K9K2G08X0A

Document Title

256M x 8 Bit NAND Flash Memory

Revision History

Revision No	History	Draft Date	Remark
0.0	1. Initial issue	May. 31. 2004	Advance
0.1	1. Technical note is changed 2. Notes of AC timing characteristics are added 3. The description of Copy-back program is changed 4. TSOP package is deleted	Oct. 25. 2004	Preliminary
0.2	1. $\overline{CE}$ access time : 23ns->35ns (p.9)	Feb. 14. 2005	
0.3	1. The value of tREA is changed. (18ns->20ns) 2. EDO mode is added.	May 4 2005	
0.4	1. The flow chart to creat the initial invalid block table is changed.	May 6 2005	
1.0	1. 1.8V FBGA spec is merged 2. 3.3V FBGA package is added 3. FBGA package size is changed to 9.5 x 12 4. Leaded part is deleted	Feb. 1 2006	

256M x 8 Bit NAND Flash Memory

PRODUCT LIST

Part Number	Vcc Range	Organization	PKG Type
K9K2G08U0A-F	2.7 ~ 3.6V	X8	WSOP1
K9K2G08X0A-J	1.65 ~ 1.95V	X8	FBGA

FEATURES

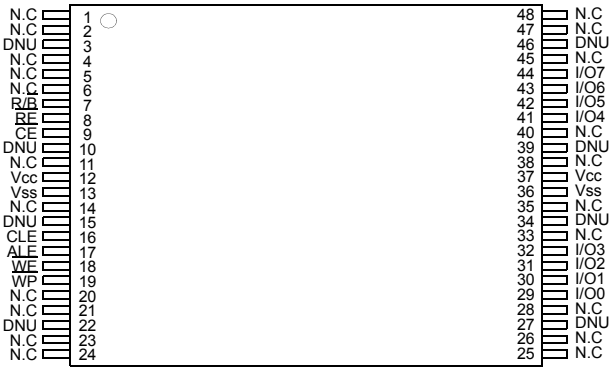
- Voltage Supply
  - 2.7 V ~3.6 V
  - 1.65V ~ 1.95V
- Organization
  - Memory Cell Array
    - (256M + 8,192K)bit x 8bit
  - Data Register
    - (2K + 64)bit x8bit
- Automatic Program and Erase
  - Page Program
    - (2K + 64)Byte
  - Block Erase
    - (128K + 4K)Byte
- Page Read Operation
  - Page Size
    - 2K-Byte
  - Random Read : 25μs(Max.)
  - Serial Access : 50ns(Min.)
- Fast Write Cycle Time
  - Program time : 300μs(Typ.)
  - Block Erase Time : 2ms(Typ.)
- Command/Address/Data Multiplexed I/O Port
- Hardware Data Protection
  - Program/Erase Lockout During Power Transitions
- Reliable CMOS Floating-Gate Technology
  - Endurance : 100K Program/Erase Cycles
  - Data Retention : 10 Years
- Command Register Operation
- Unique ID for Copyright Protection
- Package :
  - K9K2G08U0A-FIB0
    48 - Pin WSOP I (12x17x0.7mm)- Pb-free Package
  - K9K2G08X0A-JCB0/JIB0
    63- Ball FBGA (9.5x12) - Pb-free Package

GENERAL DESCRIPTION

Offered in 256Mx8bit the K9K2G08X0A is 2G bit with spare 64M bit capacity. Its NAND cell provides the most cost-effective solution for the solid state mass storage market. A program operation can be performed in typical 300μs on the 2112byte page and an erase operation can be performed in typical 2ms on a 128K-byte block. Data in the data page can be read out at 50ns cycle time per byte. The I/O pins serve as the ports for address and data input/output as well as command input. The on-chip write controller automates all program and erase functions including pulse repetition, where required, and internal verification and margining of data. Even the write-intensive systems can take advantage of the K9K2G08X0A′s extended reliability of 100K program/erase cycles by providing ECC(Error Correcting Code) with real time mapping-out algorithm. The K9K2G08X0A is an optimum solution for large nonvolatile storage applications such as solid state file storage and other portable applications requiring non-volatility.

PIN CONFIGURATION (WSOP1)

K9K2G08U0A-FIB0

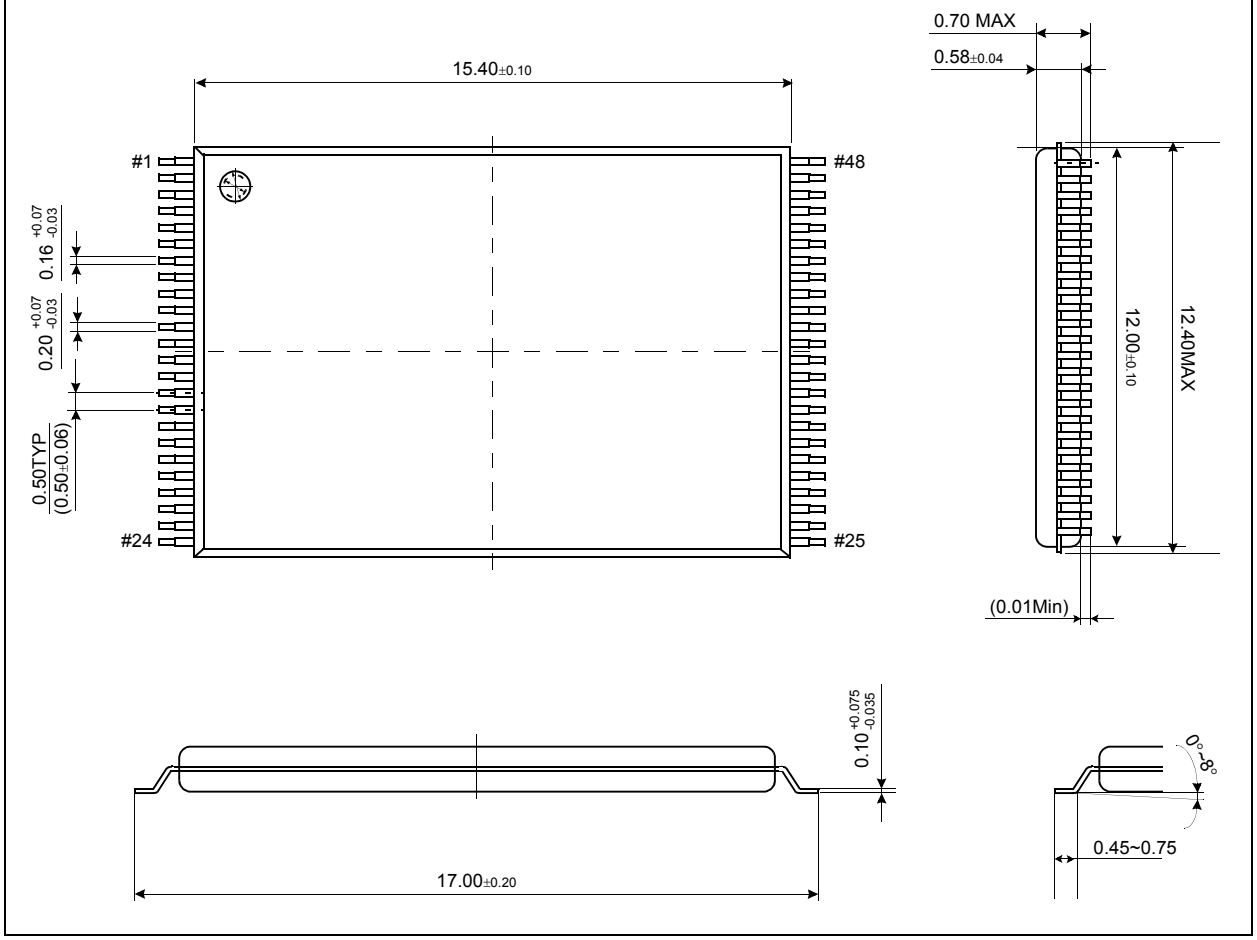


PACKAGE DIMENSIONS

48-PIN LEAD PLASTIC VERY VERY THIN SMALL OUT-LINE PACKAGE TYPE (I)

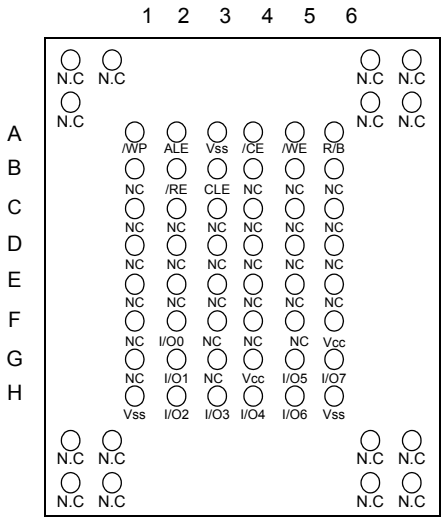
48 - WSOP1 - 1217F

Unit :mm



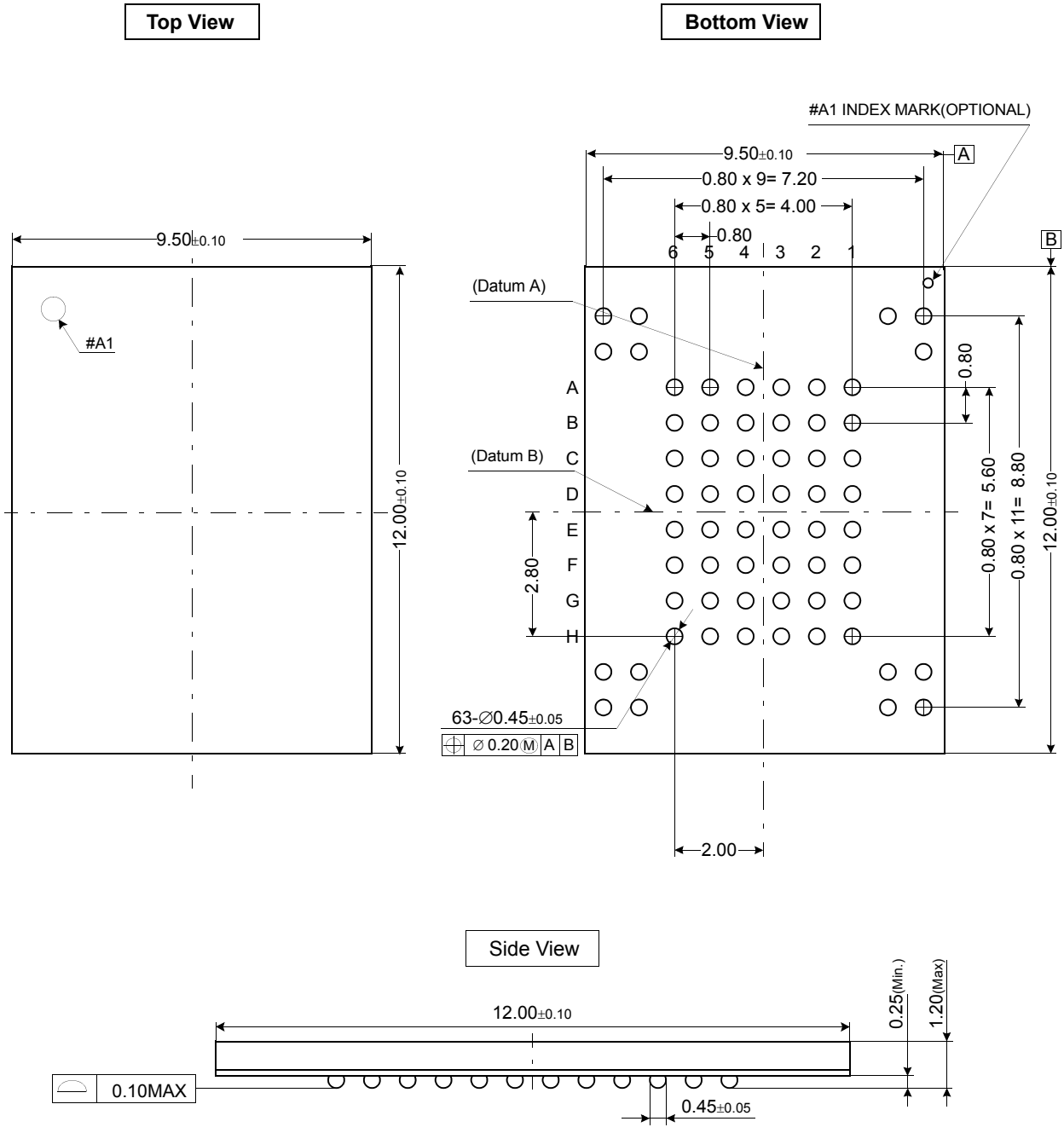
PIN CONFIGURATION (FBGA)

K9F1G08X0A-JCB0/JIB0



Top View

PACKAGE DEMENSIONS(FBGA)



PIN DESCRIPTION

Pin Name	Pin Function
I/O0 ~ I/O7	**DATA INPUTS/OUTPUTS** The I/O pins are used to input command, address and data, and to output data during read operations. The I/O pins float to high-z when the chip is deselected or when the outputs are disabled.
CLE	**COMMAND LATCH ENABLE** The CLE input controls the activating path for commands sent to the command register. When active high, commands are latched into the command register through the I/O ports on the rising edge of the $\overline{WE}$ signal.
ALE	**ADDRESS LATCH ENABLE** The ALE input controls the activating path for address to the internal address registers. Addresses are latched on the rising edge of $\overline{WE}$ with ALE high.
$\overline{CE}$	**CHIP ENABLE** The $\overline{CE}$ input is the device selection control. When the device is in the Busy state, $\overline{CE}$ high is ignored, and the device does not return to standby mode in program or erase opertion. Regarding $\overline{CE}$ control during read operation, refer to 'Page read' section of Device operation .
$\overline{RE}$	**READ ENABLE** The $\overline{RE}$ input is the serial data-out control, and when active drives the data onto the I/O bus. Data is valid tREA after the falling edge of $\overline{RE}$ which also increments the internal column address counter by one.
$\overline{WE}$	**WRITE ENABLE** The $\overline{WE}$ input controls writes to the I/O port. Commands, address and data are latched on the rising edge of the $\overline{WE}$ pulse.
$\overline{WP}$	**WRITE PROTECT** The $\overline{WP}$ pin provides inadvertent write/erase protection during power transitions. The internal high voltage generator is reset when the $\overline{WP}$ pin is active low.
R/$\overline{B}$	**READY/BUSY OUTPUT** The R/$\overline{B}$ output indicates the status of the device operation. When low, it indicates that a program, erase or random read operation is in process and returns to high state upon completion. It is an open drain output and does not float to high-z condition when the chip is deselected or when outputs are disabled.
Vcc	**POWER** VCC is the power supply for device.
Vss	**GROUND**
N.C	**NO CONNECTION** Lead is not internally connected.

NOTE:
1. Connect all VCC and VSS pins of each device to common power supply outputs.
2. Do not leave VCC or VSS disconnected.

Figure 1. Functional Block Diagram

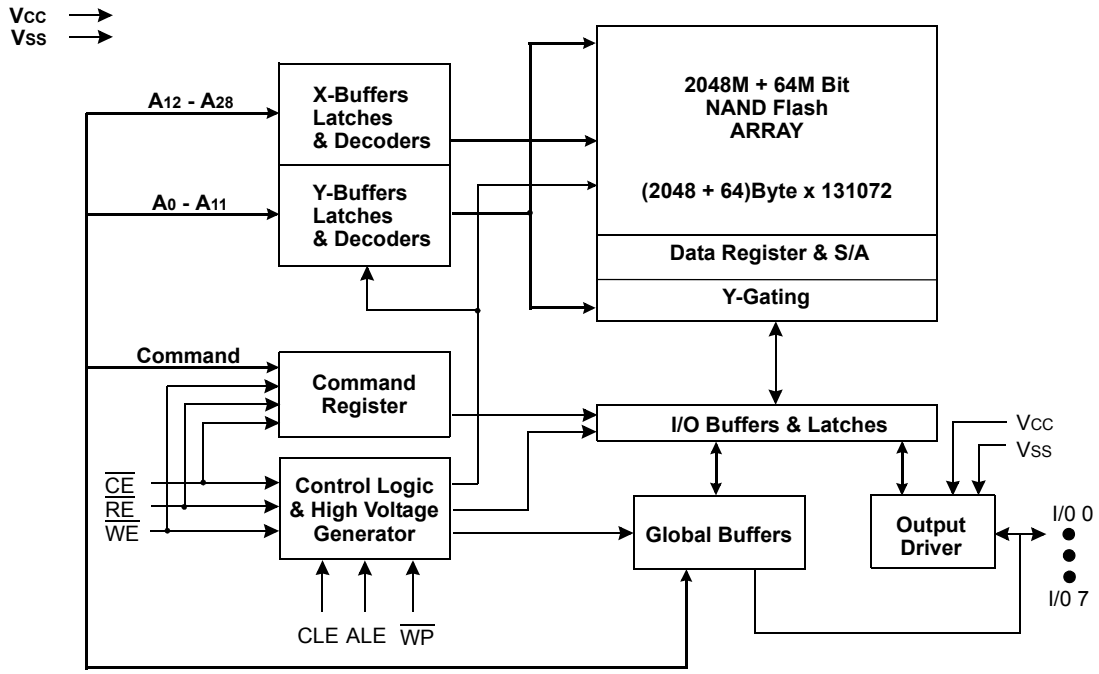
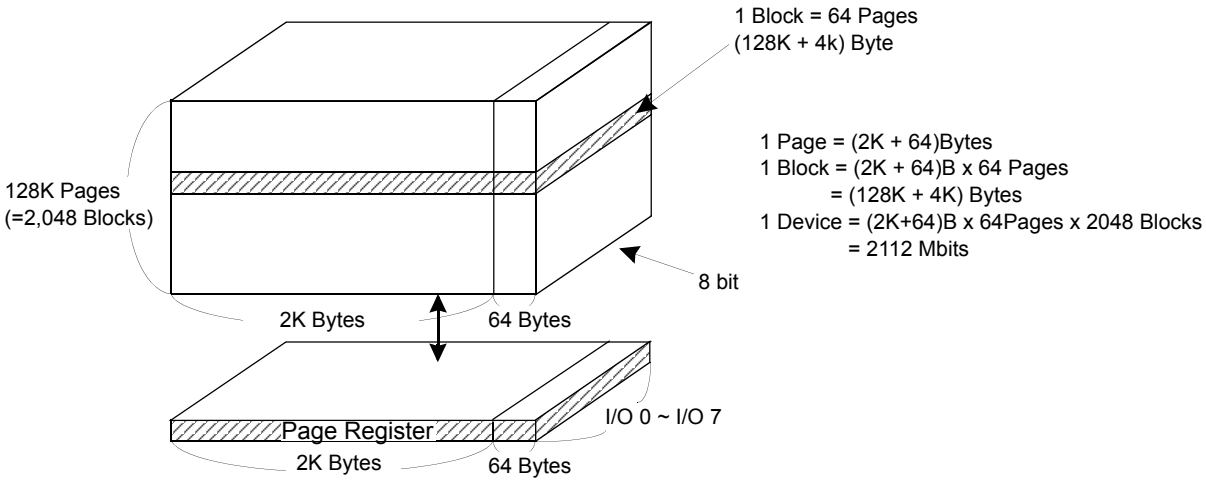


Figure 2 Array Organization



	I/O 0	I/O 1	I/O 2	I/O 3	I/O 4	I/O 5	I/O 6	I/O 7	
1st Cycle	A0	A1	A2	A3	A4	A5	A6	A7	**Column Address**
2nd Cycle	A8	A9	A10	A11	*L	*L	*L	*L	**Column Address**
3rd Cycle	A12	A13	A14	A15	A16	A17	A18	A19	**Row Address**
4th Cycle	A20	A21	A22	A23	A24	A25	A26	A27	**Row Address**
5th Cycle	A28	*L	*L	*L	*L	*L	*L	*L	**Row Address**

NOTE : Column Address : Starting Address of the Register.

* L must be set to "Low".

* The device ignores any additional input of address cycles than required.

Product Introduction

The K9K2G08X0A is a 2112Mbit(2,214,592,512 bit) memory organized as 131,072 rows(pages) by 2112x8 columns. Spare 64 columns are located from column address of 2048~2111. A 2112-byte data register is connected to memory cell arrays for accommodating data transfer between the I/O buffers and memory cells during page read and page program operations. The memory array is made up of 32 cells that are serially connected to form a NAND structure. Each of the 32 cells resides in a different page. A block consists of two NAND structures. A NAND structure consists of 32 cells. Total 135,168 NAND cells reside in a block. The program and read operations are executed on a page basis, while the erase operation is executed on a block basis. The memory array consists of 2048 separately erasable 128K-byte blocks. It indicates that the bit by bit erase operation is prohibited on the K9K2G08X0A.

The K9K2G08X0A has addresses multiplexed into 8 I/Os. This scheme dramatically reduces pin counts and allows system upgrades to future densities by maintaining consistency in system board design. Command, address and data are all written through I/O's by bringing $\overline{WE}$ to low while $\overline{CE}$ is low. Those are latched on the rising edge of $\overline{WE}$. Command Latch Enable(CLE) and Address Latch Enable(ALE) are used to multiplex command and address respectively, via the I/O pins. Some commands require one bus cycle. For example, Reset Command, Status Read Command and etc require just one cycle bus. Some other commands, like Page Read, Block Erase and Page Program, require two cycles: one cycle for setup and the other cycle for execution. The 264M byte physical space requires 29 addresses, thereby requiring five cycles for addressing: 2 cycles of column address, 3 cycles of row address, in that order. Page Read and Page Program need the same five address cycles following the required command input. In Block Erase operation, however, only the three row address cycles are used. Device operations are selected by writing specific commands into the command register. Table 1 defines the specific commands of the K9K2G08X0A.

Table 1. Command Sets

Function	1st. Cycle	2nd. Cycle	Acceptable Command during Busy
Read	00h	30h	
Read for Copy Back	00h	35h	
Read ID	90h	-	
Reset	FFh	-	O
Page Program	80h	10h	
Cache Program	80h	15h	
Copy-Back Program	85h	10h	
Block Erase	60h	D0h	
Random Data Input*1	85h	-	
Random Data Output*1	05h	E0h	
Read Status	70h		O

NOTE : 1. Random Data Input/Output can be executed in a page.

2. Cache program and Copy-Back program are supported only with 3.3V device.

Caution : Any undefined command inputs are prohibited except for above command set of Table 1.

ABSOLUTE MAXIMUM RATINGS

Parameter		Symbol	Rating		Unit
			1.8V DEVICE	3.3V DEVICE	
Voltage on any pin relative to V_{SS}		$V_{IN/OUT}$	-0.6 to + 2.45	-0.6 to + 4.6	V
		V_{CC}	-0.6 to + 2.45	-0.6 to + 4.6	
Temperature Under Bias	K9K2G08X0A-XCB0	T_{BIAS}	-10 to +125		°C
	K9K2G08X0A-XIB0		-40 to +125		
Storage Temperature	K9K2G08X0A-XCB0	T_{STG}	-65 to +150		°C
	K9K2G08X0A-XJIB0				
Short Circuit Current		Ios	5		mA

NOTE :
1. Minimum DC voltage is -0.6V on input/output pins. During transitions, this level may undershoot to -2.0V for periods <30ns.
   Maximum DC voltage on input/output pins is V_{CC}+0.3V which, during transitions, may overshoot to V_{CC}+2.0V for periods <20ns.
2. Permanent device damage may occur if ABSOLUTE MAXIMUM RATINGS are exceeded. Functional operation should be restricted to the conditions as detailed in the operational sections of this data sheet. Exposure to absolute maximum rating conditions for extended periods may affect reliability.

RECOMMENDED OPERATING CONDITIONS

(Voltage reference to GND, :T_A=0 to 70°C, K9K2G08X0A-XIB0:T_A=-40 to 85°C)

Parameter	Symbol	K9K2G08R0A(1.8V)			K9K2G08U0A(3.3V)			Unit
		Min	Typ.	Max	Min	Typ.	Max	
Supply Voltage	V_{CC}	1.65	1.8	1.95	2.7	3.3	3.6	V
Supply Voltage	V_{SS}	0	0	0	0	0	0	V

DC AND OPERATING CHARACTERISTICS(Recommended operating conditions otherwise noted.)

Parameter		Symbol	Test Conditions	K9K2G08R0A(1.8V)			K9K2G08U0A(3.3V)			Unit
				Min	Typ	Max	Min	Typ	Max	
Operating Current	Page Read with Serial Access	$I_{CC}1$	tRC=50ns, (30ns with 3.3V device) $\overline{CE}$=V_{IL} I_{OUT}=0mA	-	10	20	-	10	30	mA
	Program	$I_{CC}2$	-	-	10	20	-	10	30	
	Erase	$I_{CC}3$	-	-	10	20	-	10	30	
Stand-by Current(TTL)		$I_{SB}1$	$\overline{CE}$=V_{IH}, $\overline{WP}$=0V/V_{CC}	-	-	1	-	-	1	
Stand-by Current(CMOS)		$I_{SB}2$	$\overline{CE}$=V_{CC}-0.2, $\overline{WP}$=0V/V_{CC}	-	20	100	-	20	100	µA
Input Leakage Current		I_{LI}	V_{IN}=0 to Vcc(max)	-	-	±20	-	-	±10	
Output Leakage Current		I_{LO}	V_{OUT}=0 to Vcc(max)	-	-	±20	-	-	±10	
Input High Voltage		V_{IH}	-	0.8xVcc	-	Vcc+0.3	0.8xVcc	-	Vcc+0.3	V
Input Low Voltage, All inputs		V_{IL}	-	-0.3	-	0.2xVcc	-0.3	-	0.2xVcc	
Output High Voltage Level		V_{OH}	K9K2G08R0A: I_{OH}=-100µA K9K2G08U0A: I_{OH}=-400µA	Vcc-0.1	-	-	2.4	-	-	
Output Low Voltage Level		V_{OL}	K9K2G08R0A: I_{OL}=100mA K9K2G08U0A: I_{OL}=2.1mA	-	-	0.1	-	-	0.4	
Output Low Current(R/$\overline{B}$)		I_{OL}(R/$\overline{B}$)	K9K2G08R0A: V_{OL}=0.1V K9K2G08U0A: V_{OL}=0.4V	3	4	-	8	10	-	mA

VALID BLOCK

Parameter	Symbol	Min	Typ.	Max	Unit
Valid Block Number	N_{VB}	2008	-	2048	Blocks

NOTE :
1. The K9K2G08X0A may include invalid blocks when first shipped. Additional invalid blocks may develop while being used. The number of valid blocks is presented with both cases of invalid blocks considered. Invalid blocks are defined as blocks that contain one or more bad bits. Do not erase or program factory-marked bad blocks. Refer to the attached technical notes for appropriate management of invalid blocks.
2. The 1st block, which is placed on 00h block address, is fully guaranteed to be a valid block and does not require Error Correction up to 1K Program/ Earase cycles..

AC TEST CONDITION

(K9K2G08X0A-XCB0 :TA=0 to 70°C, K9K2G08X0A-XIB0:TA=-40 to 85°C
K9K2G08R0A : Vcc=1.65V~1.95V, K9K2G08U0A : Vcc=2.7V~3.6Vunless otherwise noted)

Parameter	K9K2G08R0A	K9K2G08U0A
Input Pulse Levels	0V to Vcc	0V to Vcc
Input Rise and Fall Times	5ns	5ns
Input and Output Timing Levels	Vcc/2	Vcc/2
Output Load	1 TTL GATE and CL=30pF	1 TTL GATE and CL=50pF

CAPACITANCE(T_A=25C, V_{CC}=1.8V/3.3V, f=1.0MHz)

Item	Symbol	Test Condition	Min	Max	Unit
Input/Output Capacitance	$C_{I/O}$	V_{IL}=0V	-	20	pF
Input Capacitance	C_{IN}	V_{IN}=0V	-	20	pF

NOTE : Capacitance is periodically sampled and not 100% tested.

MODE SELECTION

CLE	ALE	$\overline{CE}$	$\overline{WE}$	$\overline{RE}$	$\overline{WP}$	Mode	
H	L	L	⎍	H	X	Read Mode	Command Input
L	H	L	⎍	H	X		Address Input(5clock)
H	L	L	⎍	H	H	Write Mode	Command Input
L	H	L	⎍	H	H		Address Input(5clock)
L	L	L	⎍	H	H	Data Input	
L	L	L	H	⎎	X	Data Output	
X	X	X	X	H	X	During Read(Busy)	
X	X	X	X	X	H	During Program(Busy)	
X	X	X	X	X	H	During Erase(Busy)	
X	X(1)	X	X	X	L	Write Protect	
X	X	H	X	X	0V/V_{CC}(2)	Stand-by	

NOTE : 1. X can be V_{IL} or V_{IH}.
2. $\overline{WP}$ should be biased to CMOS high or CMOS low for standby.

Program / Erase Characteristics

Parameter		Symbol	Min	Typ	Max	Unit
Program Time		t_{PROG}*1	-	200	700	μs
Dummy Busy Time for Cache Program		t_{CBSY}*2		3	700	μs
Number of Partial Program Cycles in the Same Page	Main Array	Nop	-	-	4	cycles
	Spare Array		-	-	4	cycles
Block Erase Time		t_{BERS}	-	2	3	ms

NOTE : 1.Typical program time is defined as the time within which more than 50% of whole pages are programmed at Vcc of 3.3V and 25°C
2. Max. time of t_{CBSY} depends on timing between internal program completion and data in

AC Timing Characteristics for Command / Address / Data Input

Parameter	Symbol	Min		Max		Unit
		K9K2G08R0A	K9K2G08U0A	K9K2G08R0A	K9K2G08U0A	
CLE setup Time	t_{CLS}*1	25	15	-	-	ns
CLE Hold Time	t_{CLH}	10	5	-	-	ns
$\overline{CE}$ setup Time	t_{CS}*1	35	20	-	-	ns
$\overline{CE}$ Hold Time	t_{CH}	10	5	-	-	ns
$\overline{WE}$ Pulse Width	t_{WP}	25	15	-	-	ns
ALE setup Time	t_{ALS}*1	25	15	-	-	ns
ALE Hold Time	t_{ALH}	10	5	-	-	ns
Data setup Time	t_{DS}*1	20	15	-	-	ns
Data Hold Time	t_{DH}	10	5	-	-	ns
Write Cycle Time	t_{WC}	45	30	-	-	ns
$\overline{WE}$ High Hold Time	t_{WH}	15	10	-	-	ns
Address to Data Loading Time	t_{ADL}*2	100*2	100*2	-	-	ns

NOTE : 1. The transition of the corresponding control pins must occur only once while $\overline{WE}$ is held low.
2. tADL is the time from the $\overline{WE}$ rising edge of final address cycle to the $\overline{WE}$ rising edge of first data cycle.
3. **For cache program operation, the whole AC Charcateristics must be same as that of K9K2G08R0A.**

AC Characteristics for Operation

Parameter	Symbol	Min		Max		Unit
		K9K2G08R0A	K9K2G08U0A	K9K2G08R0A	K9K2G08U0A	
Data Transfer from Cell to Register	t_R	-	-	25	25	µs
ALE to $\overline{RE}$ Delay	t_{AR}	10	10	-	-	ns
CLE to $\overline{RE}$ Delay	t_{CLR}	10	10	-	-	ns
Ready to $\overline{RE}$ Low	t_{RR}	20	20	-	-	ns
RE Pulse Width	t_{RP}	25	15	-	-	ns
WE High to Busy	t_{WB}	-	-	100	100	ns
Read Cycle Time	t_{RC}	50	30	-	-	ns
$\overline{RE}$ Access Time	t_{REA}	-	-	30	20	ns
$\overline{CE}$ Access Time	t_{CEA}	-	-	45	35	ns
$\overline{RE}$ High to Output Hi-Z	t_{RHZ}	-	-	30	30	ns
$\overline{CE}$ High to Output Hi-Z	t_{CHZ}	-	-	20	20	ns
$\overline{RE}$ or $\overline{CE}$ High to Output hold	t_{OH}	15	15	-	-	ns
$\overline{RE}$ High Hold Time	t_{REH}	15	10	-	-	ns
Output Hi-Z to $\overline{RE}$ Low	t_{IR}	0	0	-	-	ns
$\overline{WE}$ High to $\overline{RE}$ Low	t_{WHR}	60	60	-	-	ns
Device Resetting Time (Read/Program/Erase)	t_{RST}	-	-	5/10/500*1	5/10/500*1	µs

NOTE: 1. If reset command(FFh) is written at Ready state, the device goes into Busy for maximum 5us.
2. **For cache program operation, the whole AC Charcateristics must be same as that of K9K2G08R0A.**

NAND Flash Technical Notes

Initial Invalid Block(s)

Initial invalid blocks are defined as blocks that contain one or more initial invalid bits whose reliability is not guaranteed by Samsung. The information regarding the initial invalid block(s) is so called as the initial invalid block information. Devices with initial invalid block(s) have the same quality level as devices with all valid blocks and have the same AC and DC characteristics. An initial invalid block(s) does not affect the performance of valid block(s) because it is isolated from the bit line and the common source line by a select transistor. The system design must be able to mask out the initial invalid block(s) via address mapping. The 1st block, which is placed on 00h block address, is fully guaranteed to be a valid block, does not require Error Correction up to 1K Program/Erase cycles.

Identifying Initial Invalid Block(s)

All device locations are erased except locations where the initial invalid block(s) information is written prior to shipping. The initial invalid block(s) status is defined by the 1st byte in the spare area. Samsung makes sure that either the 1st or 2nd page of every initial invalid block has non-FFh data at the column address of 2048. Since the initial invalid block information is also erasable in most cases, it is impossible to recover the information once it has been erased. Therefore, the system must be able to recognize the initial invalid block(s) based on the initial invalid block information and create the initial invalid block table via the following suggested flow chart(Figure 3). Any intentional erasure of the initial invalid block information is prohibited.

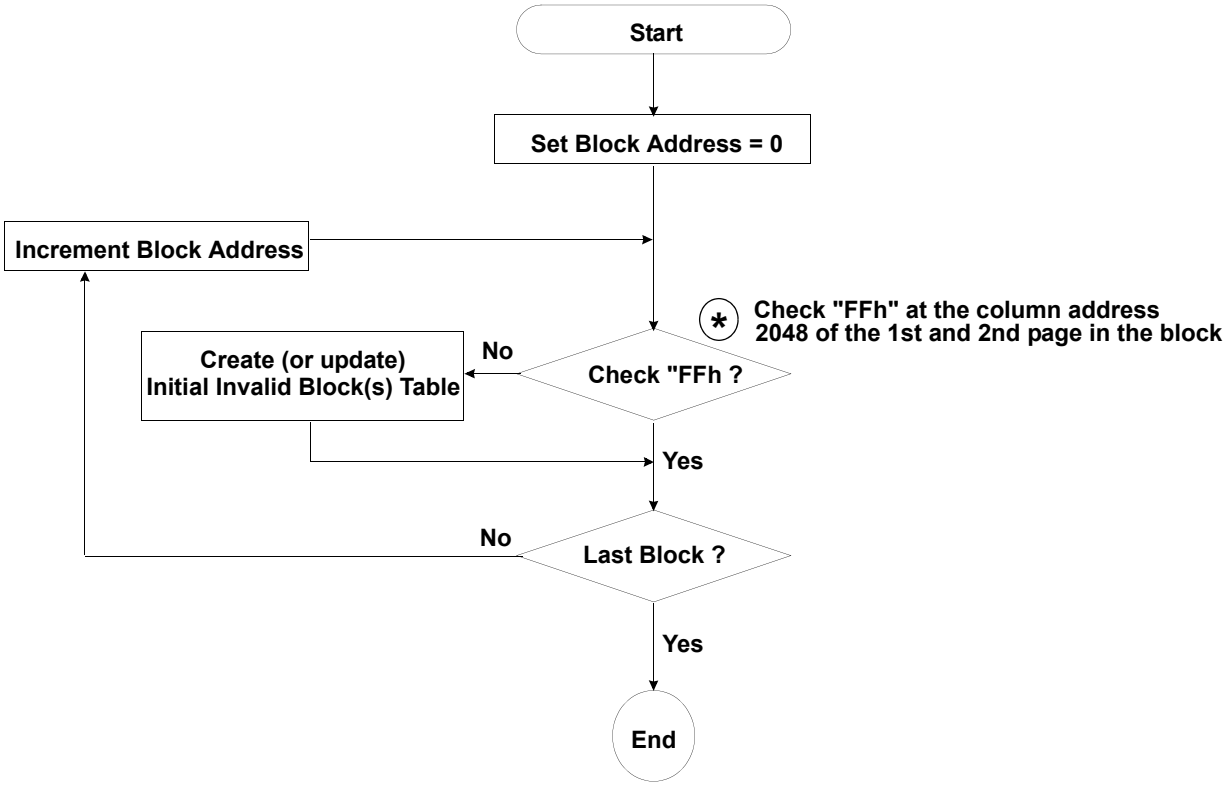


Figure 3. Flow chart to create initial invalid block table.

NAND Flash Technical Notes *(Continued)*

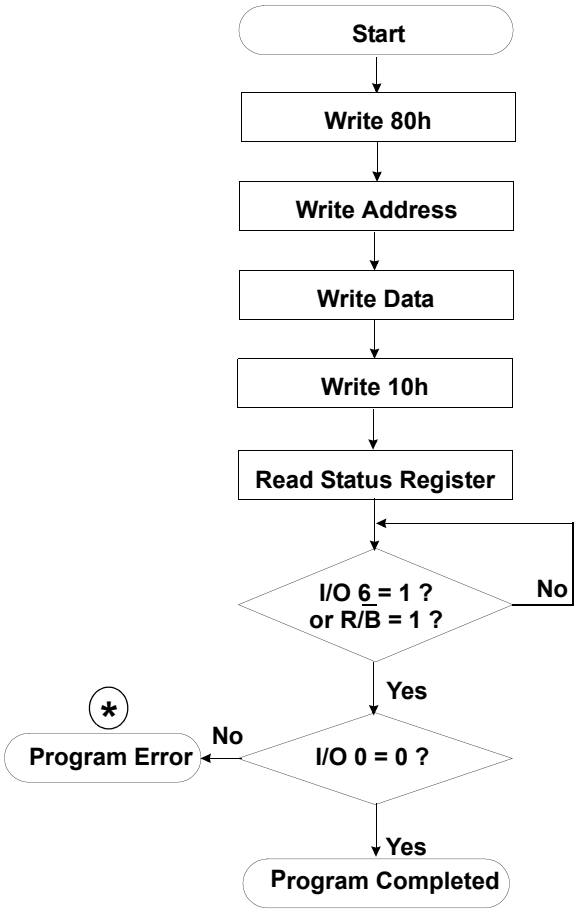
Error in write or read operation

Within its life time, additional invalid blocks may develop with NAND Flash memory. Refer to the qualification report for the block failure rate.The following possible failure modes should be considered to implement a highly reliable system. In the case of status read failure after erase or program, block replacement should be done. Because program status fail during a page program does not affect the data of the other pages in the same block, block replacement can be executed with a page-sized buffer by finding an erased empty block and reprogramming the current target data and copying the rest of the replaced block. In case of Read, ECC must be employed. To improve the efficiency of memory space, it is recommended that the read failure due to single bit error should be reclaimed by ECC without any block replacement. The block failure rate in the qualification report does not include those reclaimed blocks.

Failure Mode		Detection and Countermeasure sequence
Write	Erase Failure	Status Read after Erase --> Block Replacement
	Program Failure	Status Read after Program --> Block Replacement
Read	Single Bit Failure	Verify ECC -> ECC Correction

ECC : Error Correcting Code --> Hamming Code etc.
Example) 1bit correction & 2bit detection

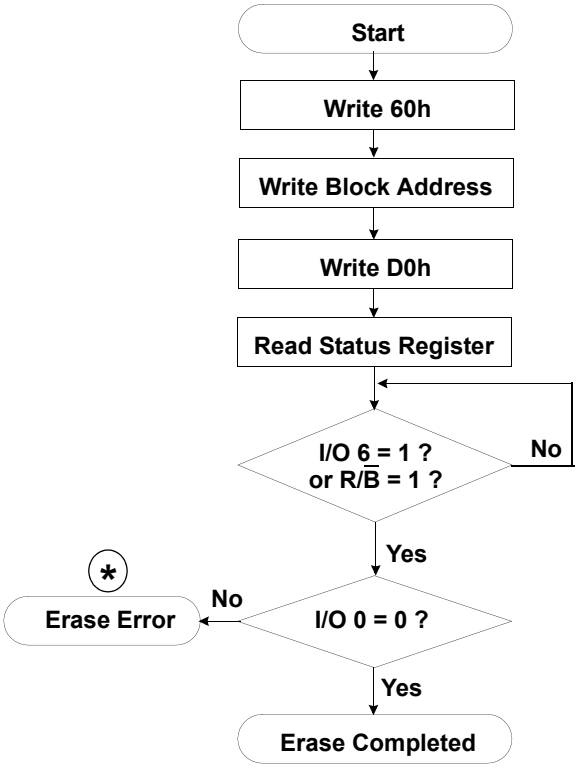
Program Flow Chart



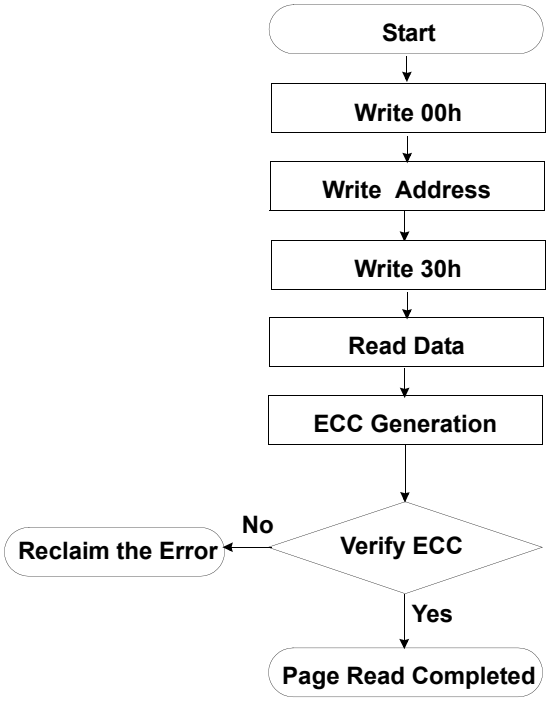
(*) : If program operation results in an error, map out the block including the page in error and copy the target data to another block.

NAND Flash Technical Notes *(Continued)*

Erase Flow Chart

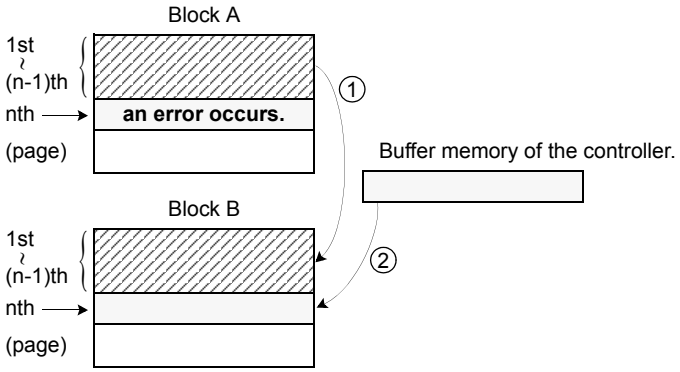


Read Flow Chart



(*) : If erase operation results in an error, map out the failing block and replace it with another block.

Block Replacement

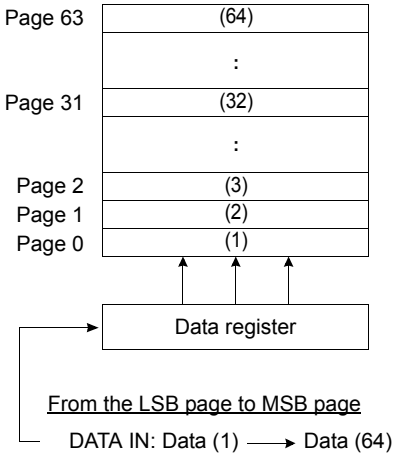


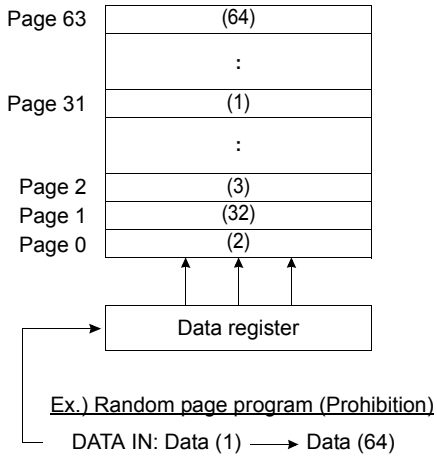
* Step1
When an error happens in the nth page of the Block 'A' during erase or program operation.
* Step2
Copy the data in the 1st ~ (n-1)th page to the same location of another free block. (Block 'B')
* Step3
Then, copy the nth page data of the Block 'A' in the buffer memory to the nth page of the Block 'B'.
* Step4
Do not erase or program to Block 'A' by creating an 'invalid Block' table or other appropriate scheme.

NAND Flash Technical Notes *(Continued)*

Addressing for program operation

Within a block, the pages must be programmed consecutively from the LSB (least significant bit) page of the block to MSB (most significant bit) pages of the block. Random page address programming is prohibited.





System Interface Using $\overline{CE}$ don't-care.

For an easier system interface, $\overline{CE}$ may be inactive during the data-loading or serial access as shown below. The internal 2112byte data registers are utilized as separate buffers for this operation and the system design gets more flexible. In addition, for voice or audio applications which use slow cycle time on the order of μ-seconds, de-activating $\overline{CE}$ during the data-loading and serial access would provide significant savings in power consumption.

Figure 4. Program Operation with $\overline{CE}$ don't-care.

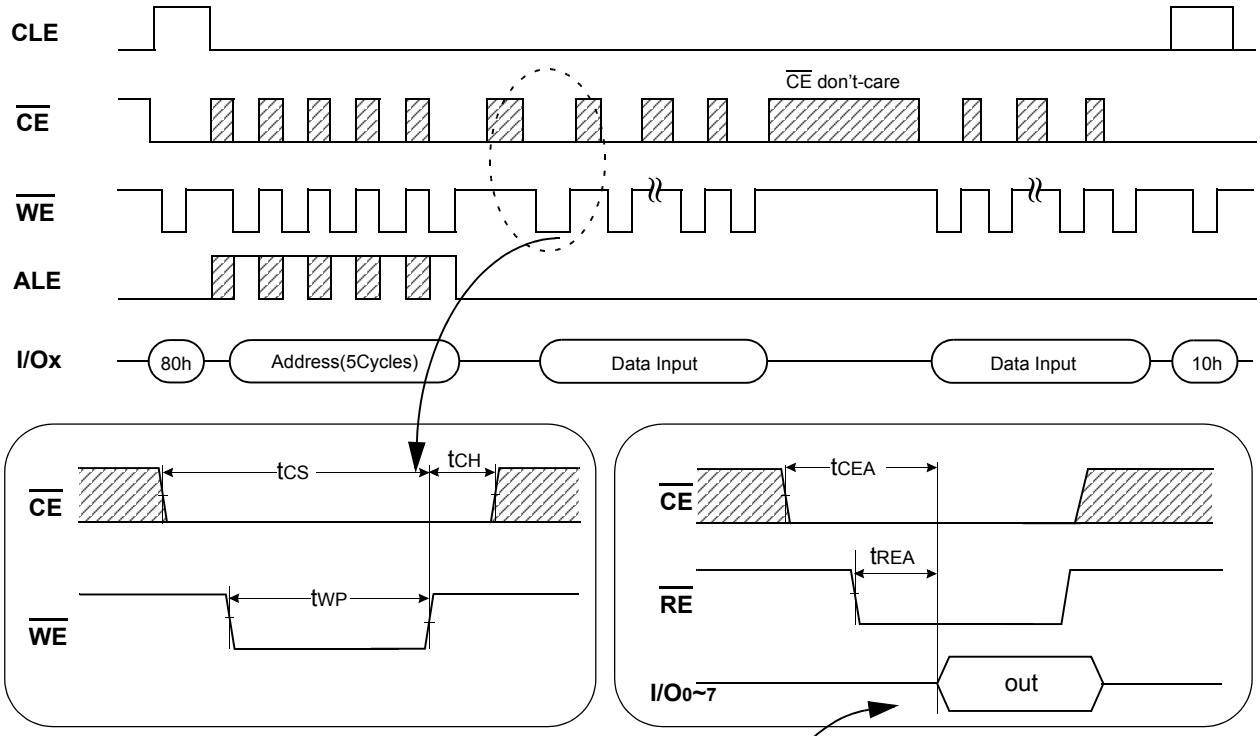
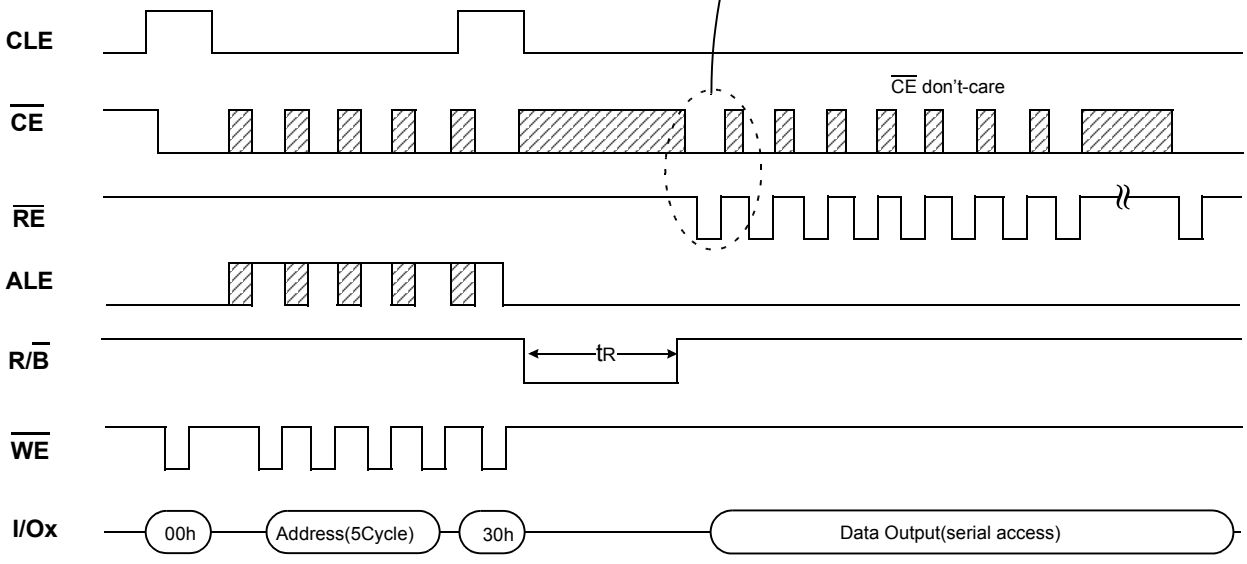


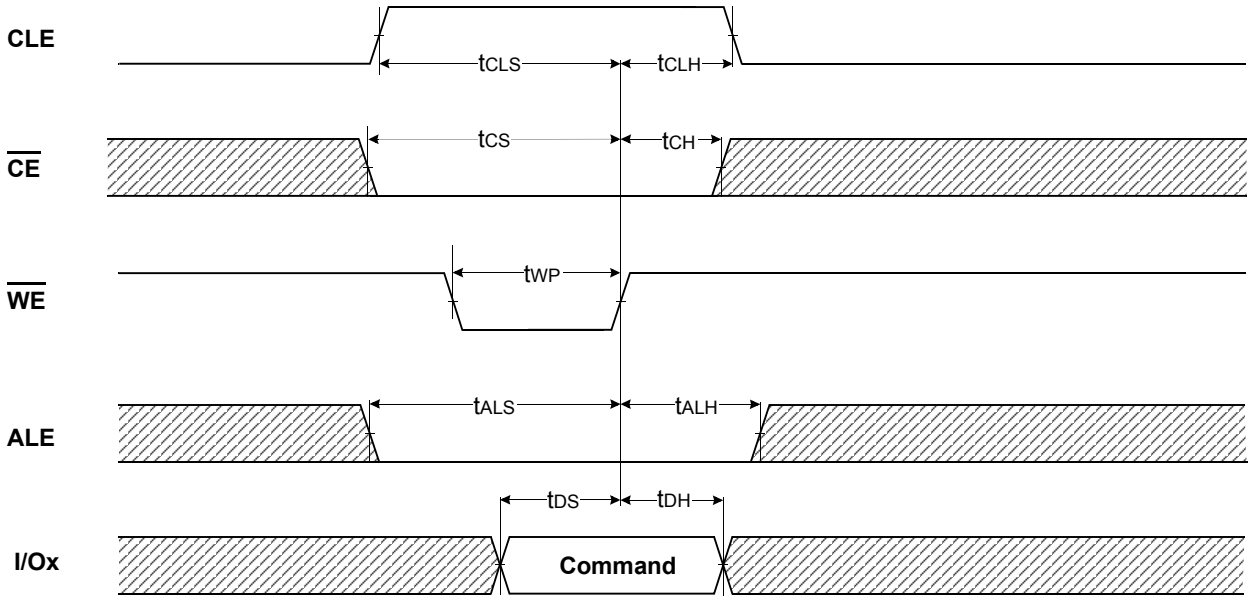
Figure 5. Read Operation with $\overline{CE}$ don't-care.



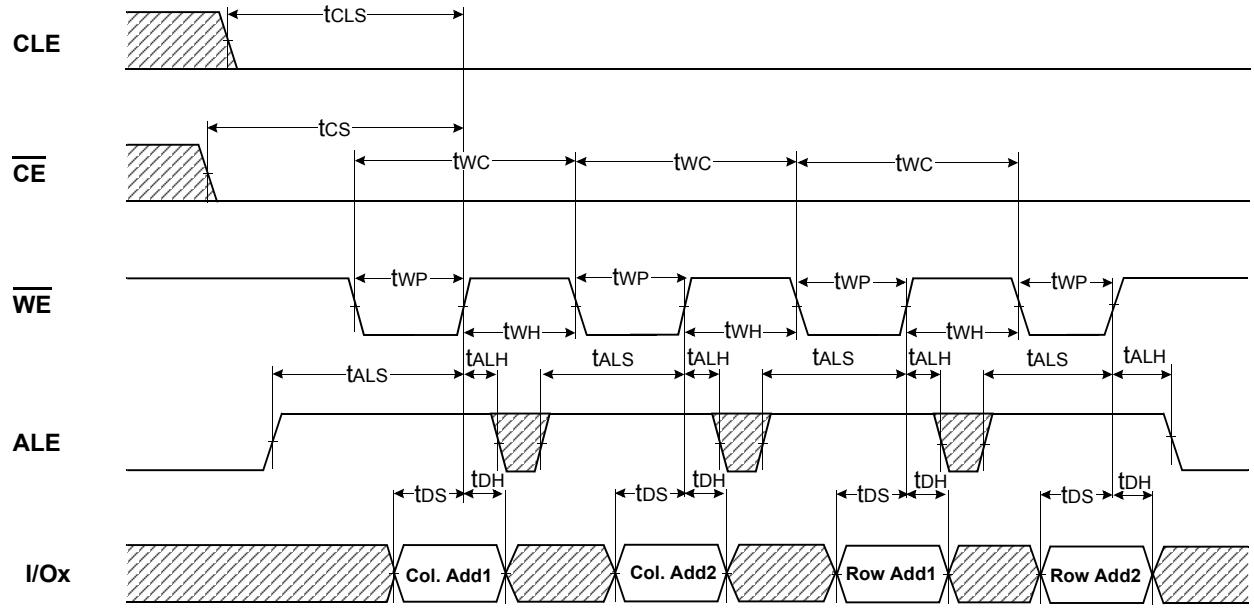
NOTE

Device	I/O	DATA	ADDRESS				
	I/Ox	Data In/Out	Col. Add1	Col. Add2	Row Add1	Row Add2	Row Add3
K9K2G08X0A	I/O 0 ~ I/O 7	~2112byte	A0~A7	A8~A11	A12~A19	A20~A27	A28

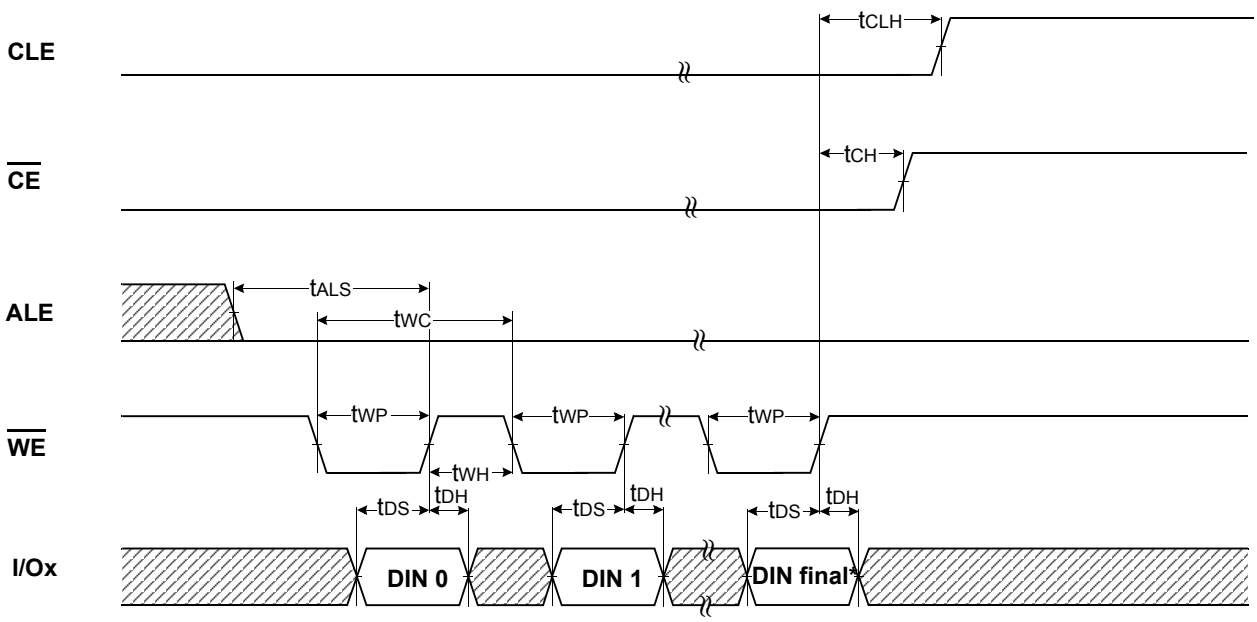
Command Latch Cycle



Address Latch Cycle

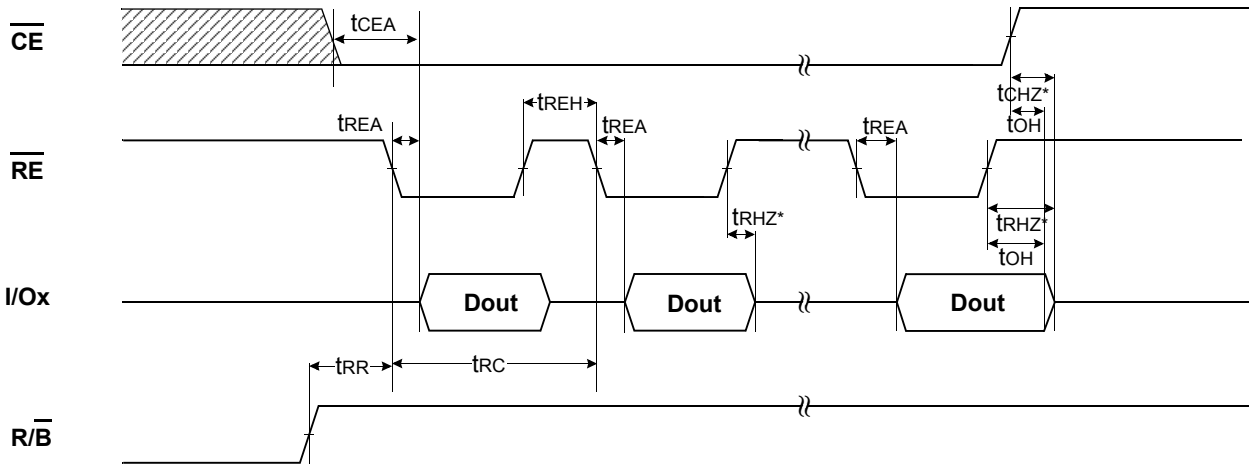


Input Data Latch Cycle



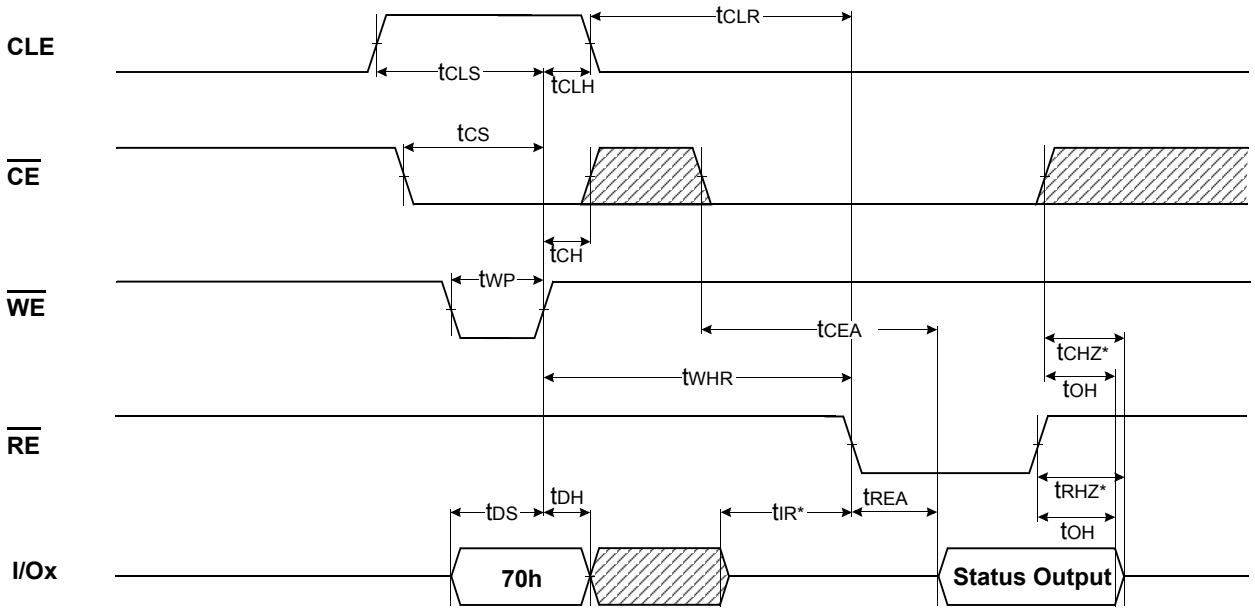
NOTES : DIN final means 2112

Serial Access Cycle after Read(CLE=L, $\overline{WE}$=H, ALE=L)

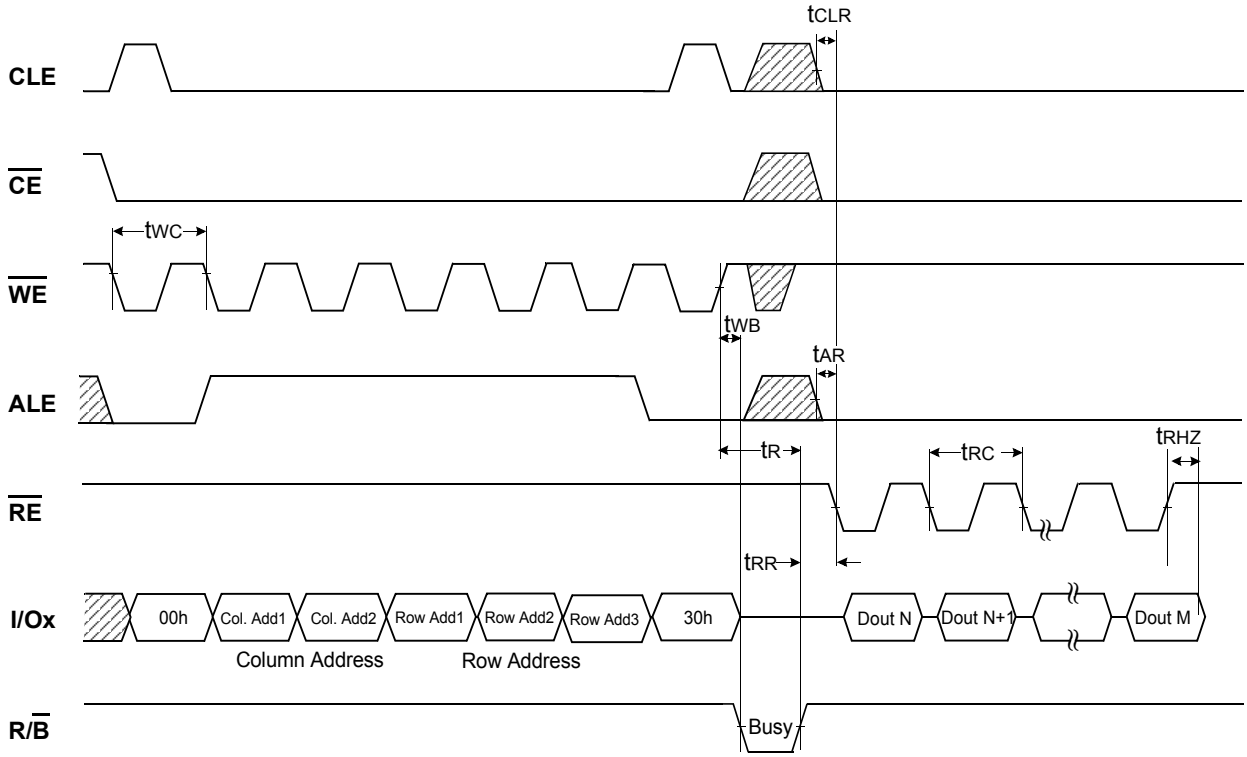


NOTES : Transition is measured ±200mV from steady state voltage with load.
This parameter is sampled and not 100% tested.

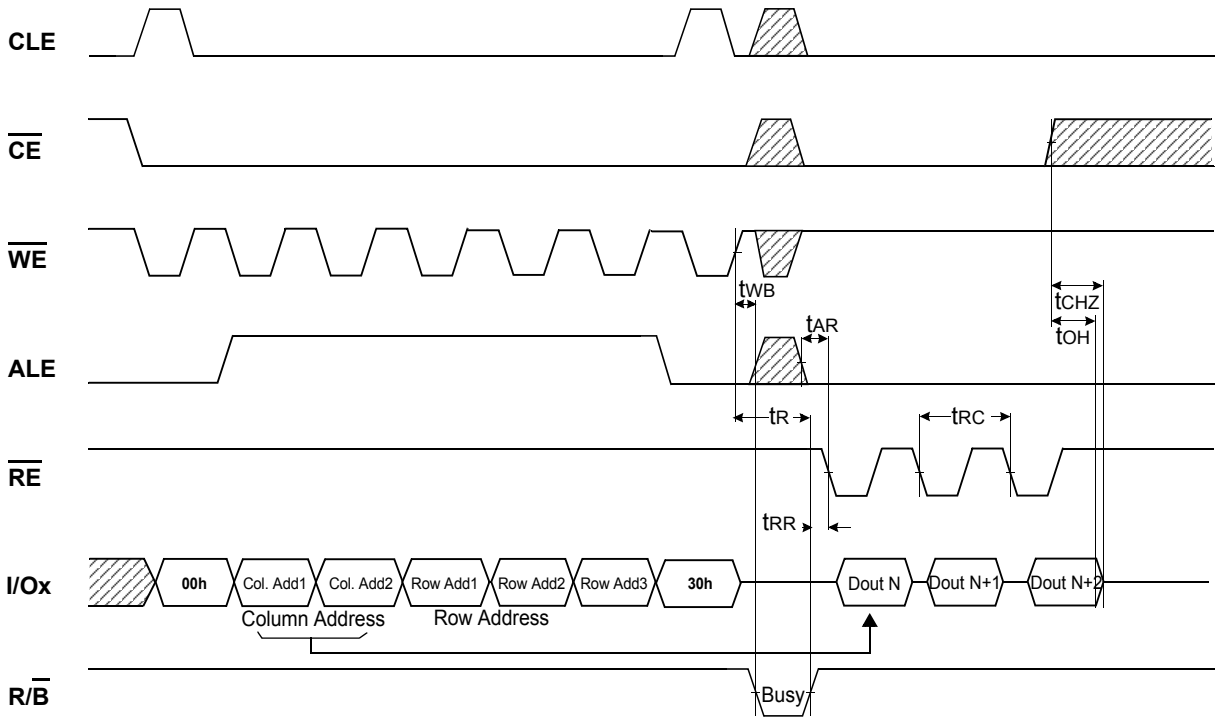
Status Read Cycle



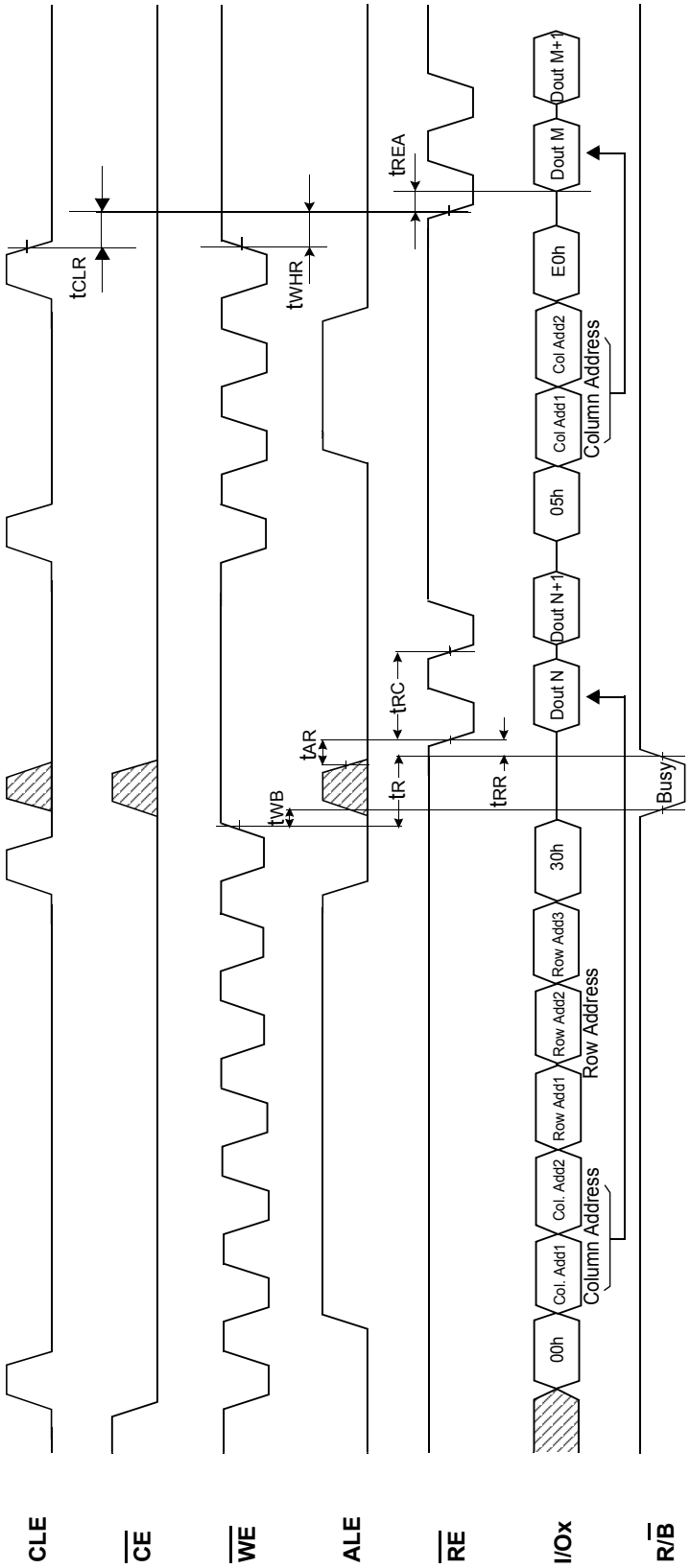
Read Operation



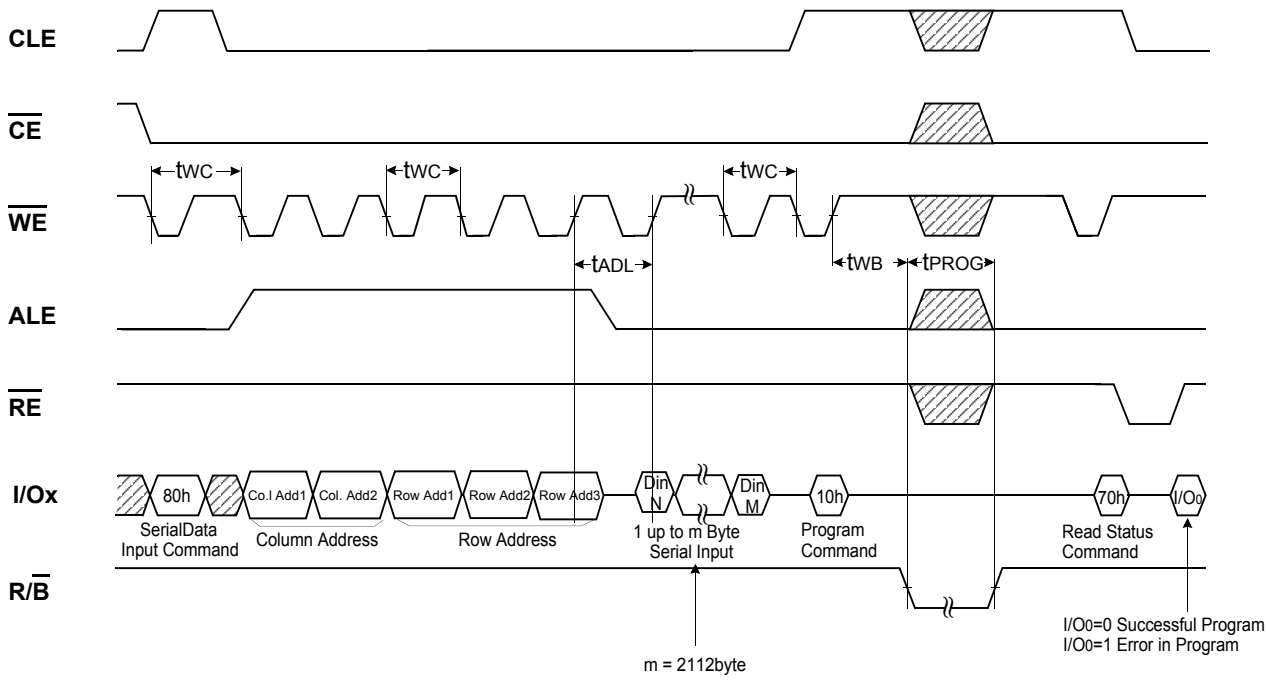
Read Operation(Intercepted by $\overline{CE}$)



Random Data Output In a Page

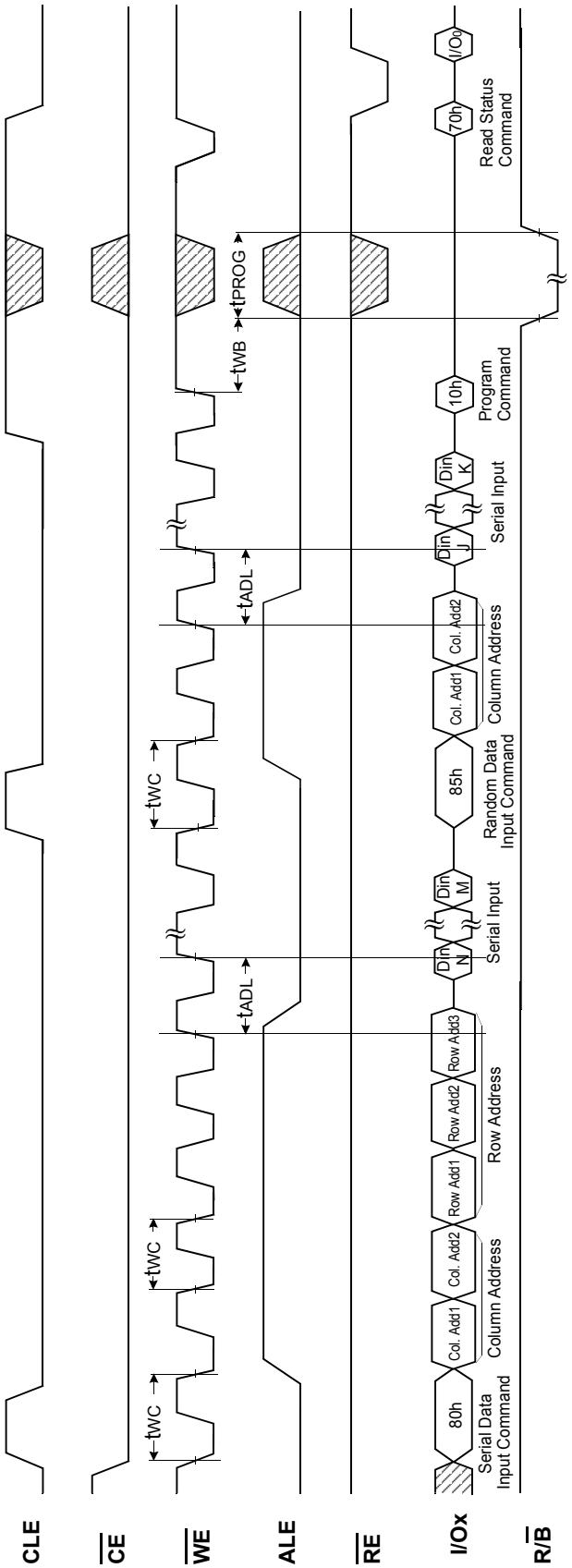


Page Program Operation



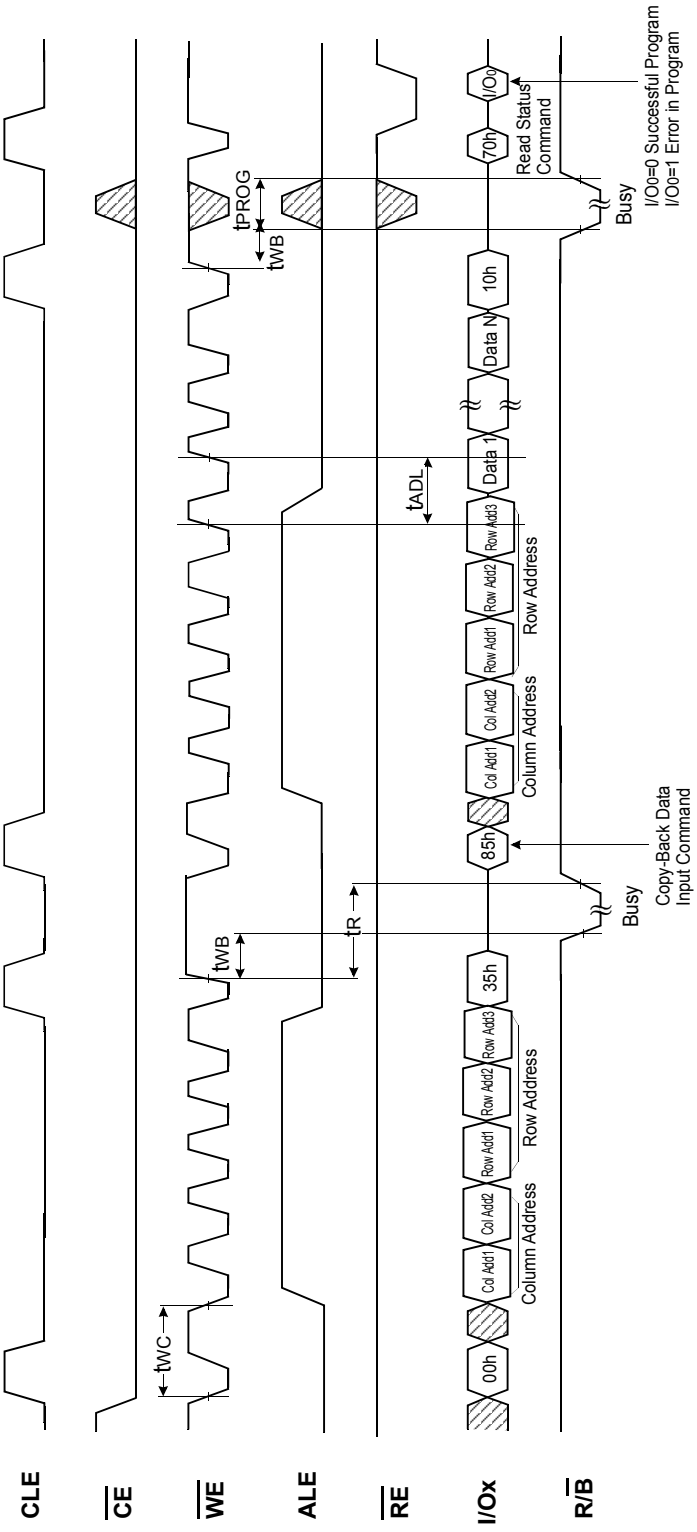
NOTES : tADL is the time from the $\overline{WE}$ rising edge of final address cycle to the $\overline{WE}$ rising edge of first data cycle.

Page Program Operation with Random Data Input



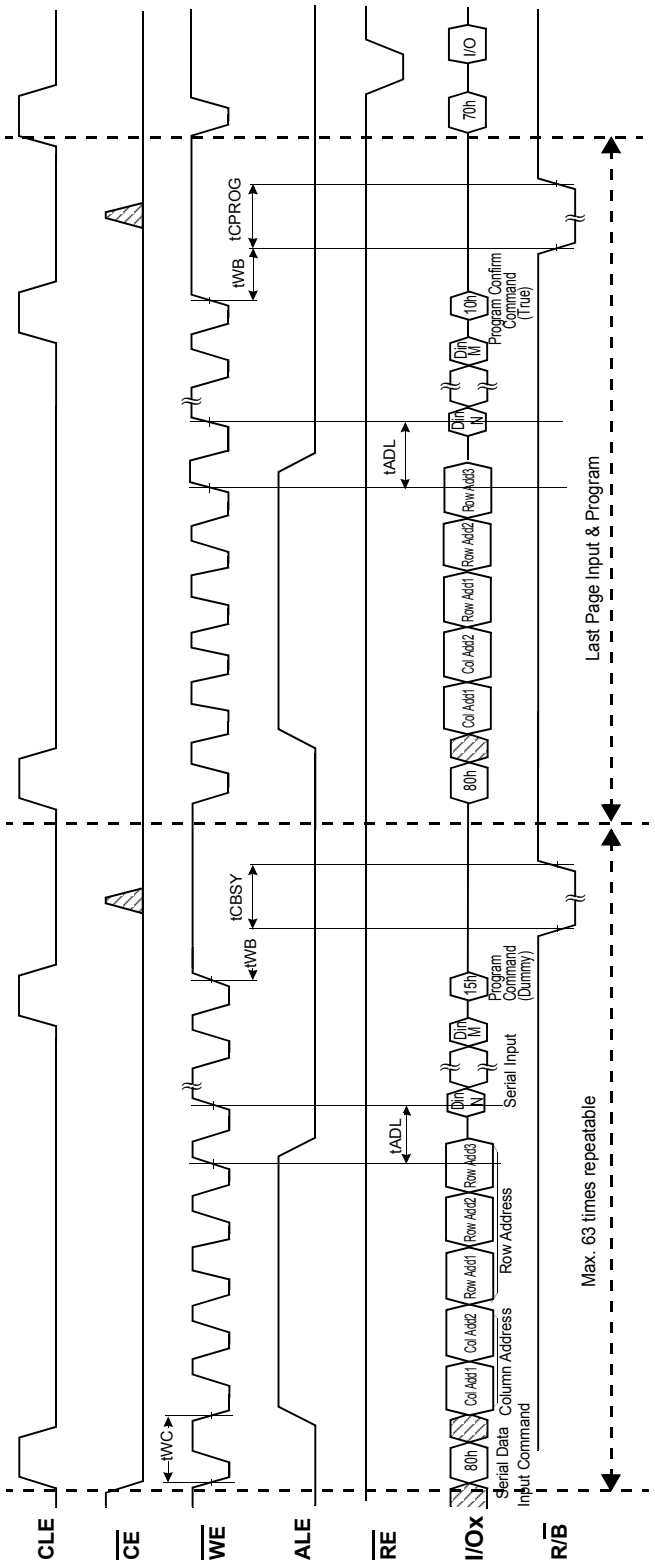
NOTES : tADL is the time from the $\overline{WE}$ rising edge of final address cycle to the $\overline{WE}$ rising edge of first data cycle.

Copy-Back Program Operation With Random Data Input



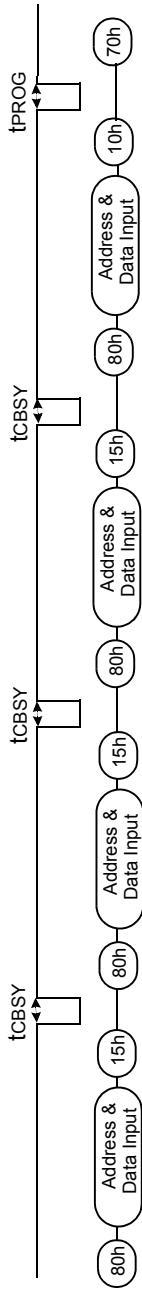
NOTES : tADL is the time from the $\overline{WE}$ rising edge of final address cycle to the $\overline{WE}$ rising edge of first data cycle.

Cache Program Operation(available only within a block)

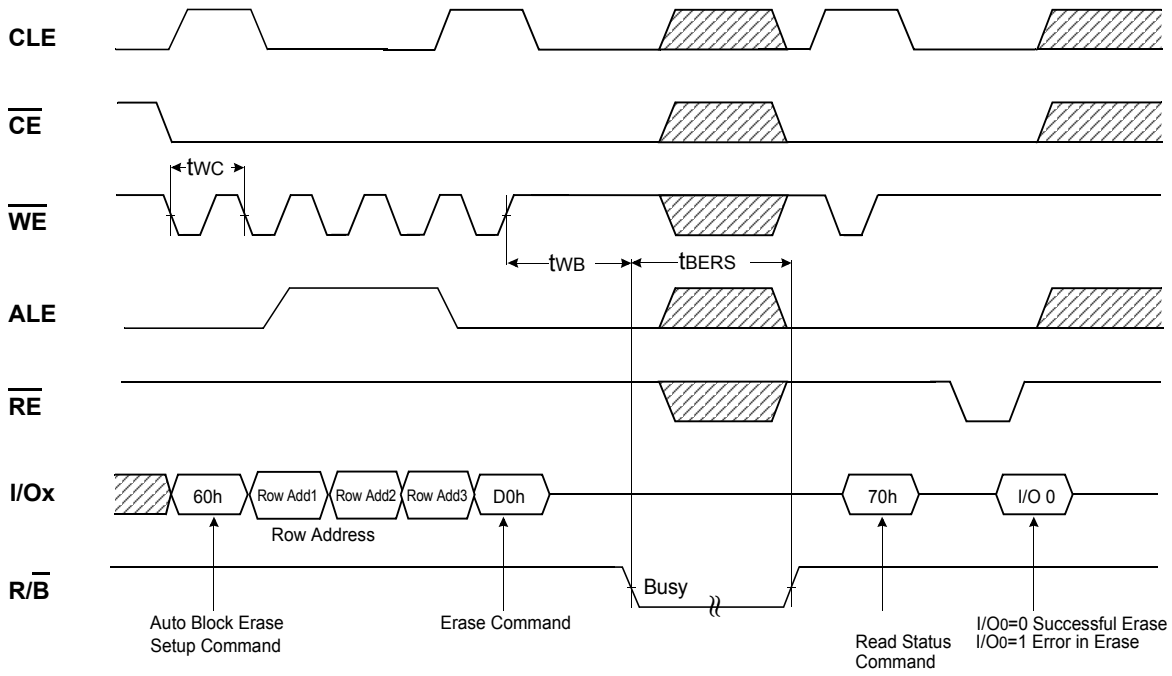


NOTES : tADL is the time from the $\overline{WE}$ rising edge of final address cycle to the $\overline{WE}$ rising edge of first data cycle.

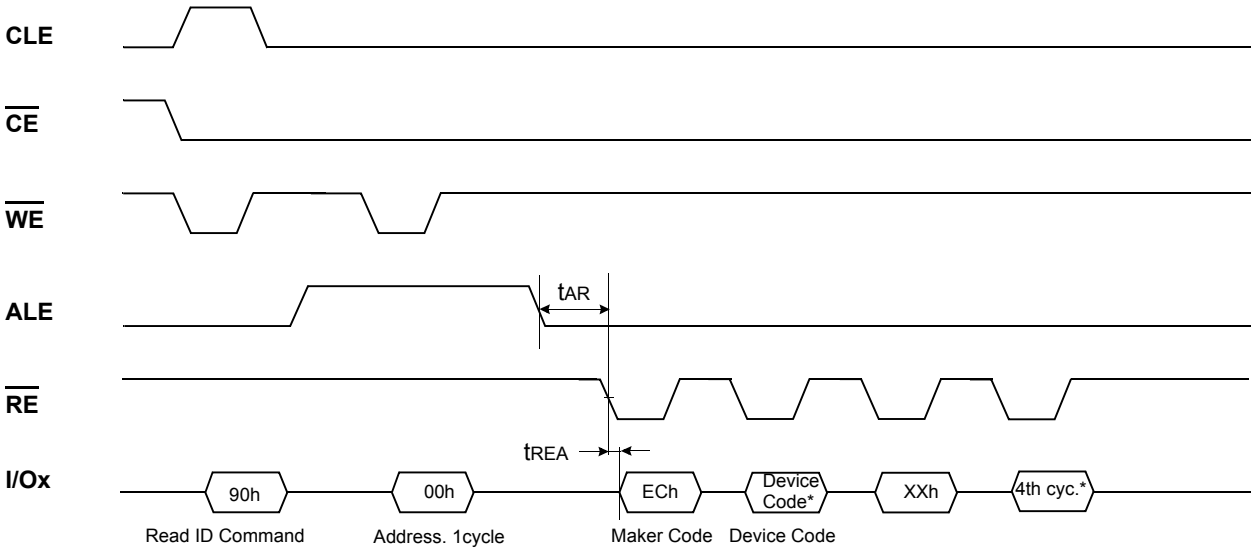
Ex.) Cache Program



Block Erase Operation



Read ID Operation



Device	Device Code*(2nd Cycle)	4th Cycle*
K9K2G08R0A	AAh	15h
K9K2G08U0A	DAh	15h

ID Defintition Table

90 ID : Access command = 90H

	Description
1st Byte	Maker Code
2nd Byte	Device Code
3rd Byte	Don't care
4th Byte	Page Size, Block Size, Spare Size, Organization

4th ID Data

	Description	I/O7	I/O6	I/O5	I/O4	I/O3	I/O2	I/O1	I/O0
Page Size (w/o redundant area)	1KB 2KB Reserved Reserved							0 0 1 1	0 1 0 1
Blcok Size (w/o redundant area)	64KB 128KB 256KB Reserved			0 0 1 1	0 1 0 1				
Redundant Area Size (byte/512byte)	8 16						0 1		
Organization	x8 x16		0 1						
Serial AccessMinimum	50ns Reserved Reserved Reserved	0 1 0 1				0 0 1 1			

Device Operation

PAGE READ

Page read is initiated by writing 00h-30h to the command register along with five address cycles. After initial power up, 00h command is latched. Therefore only five address cycles and 30h command initiates that operation after initial power up. The 2,112 bytes of data within the selected page are transferred to the data registers in less than 25μs(tR). The system controller can detect the completion of this data transfer(tR) by analyzing the output of R/$\overline{B}$ pin. Once the data in a page is loaded into the data registers, they may be read out in 50ns cycle time by sequentially pulsing $\overline{RE}$. The repetitive high to low transitions of the $\overline{RE}$ clock make the device output the data starting from the selected column address up to the last column address.

The device may output random data in a page instead of the consecutive sequential data by writing random data output command. The column address of next data, which is going to be out, may be changed to the address which follows random data output command. Random data output can be operated multiple times regardless of how many times it is done in a page.

Figure 6. Read Operation

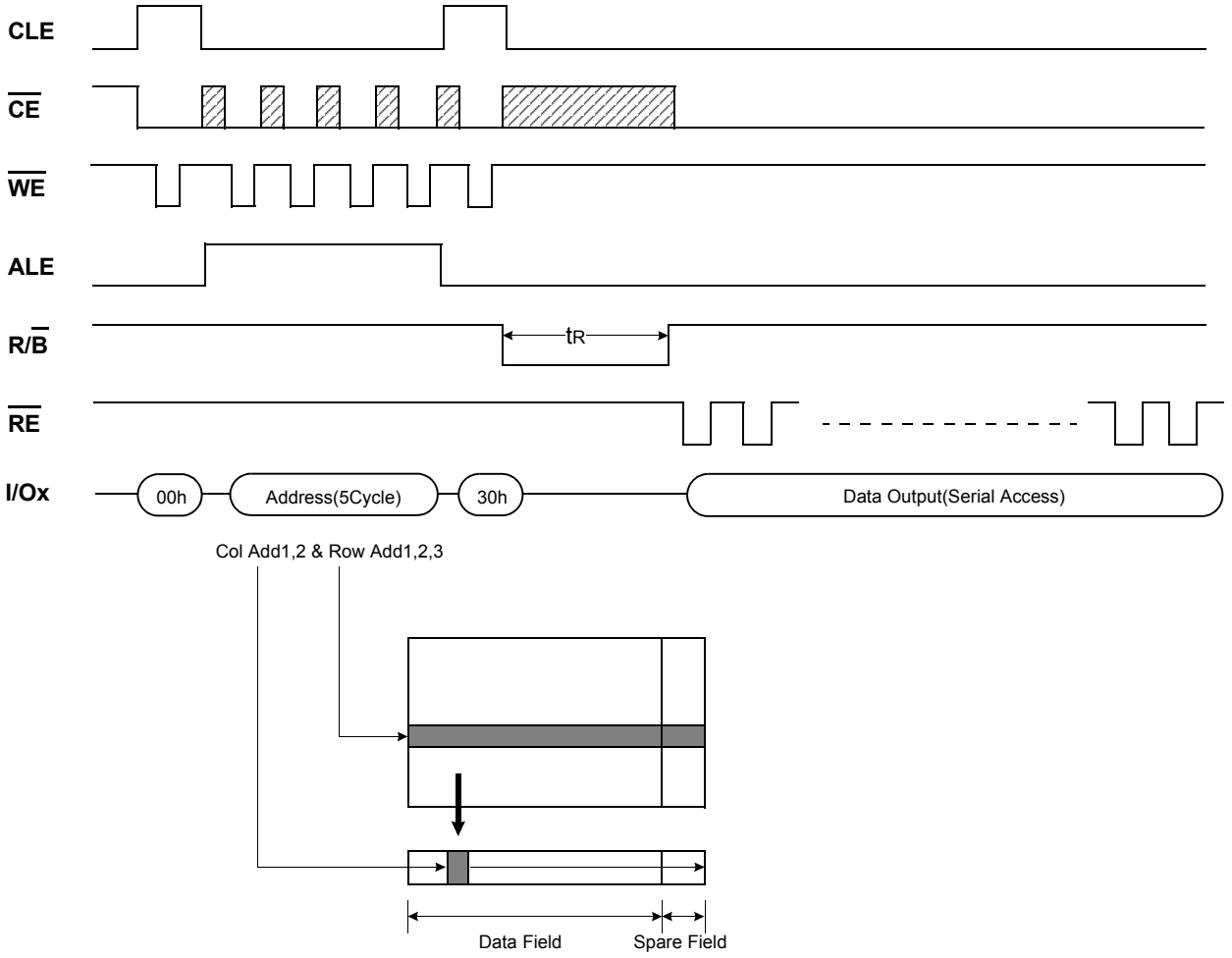
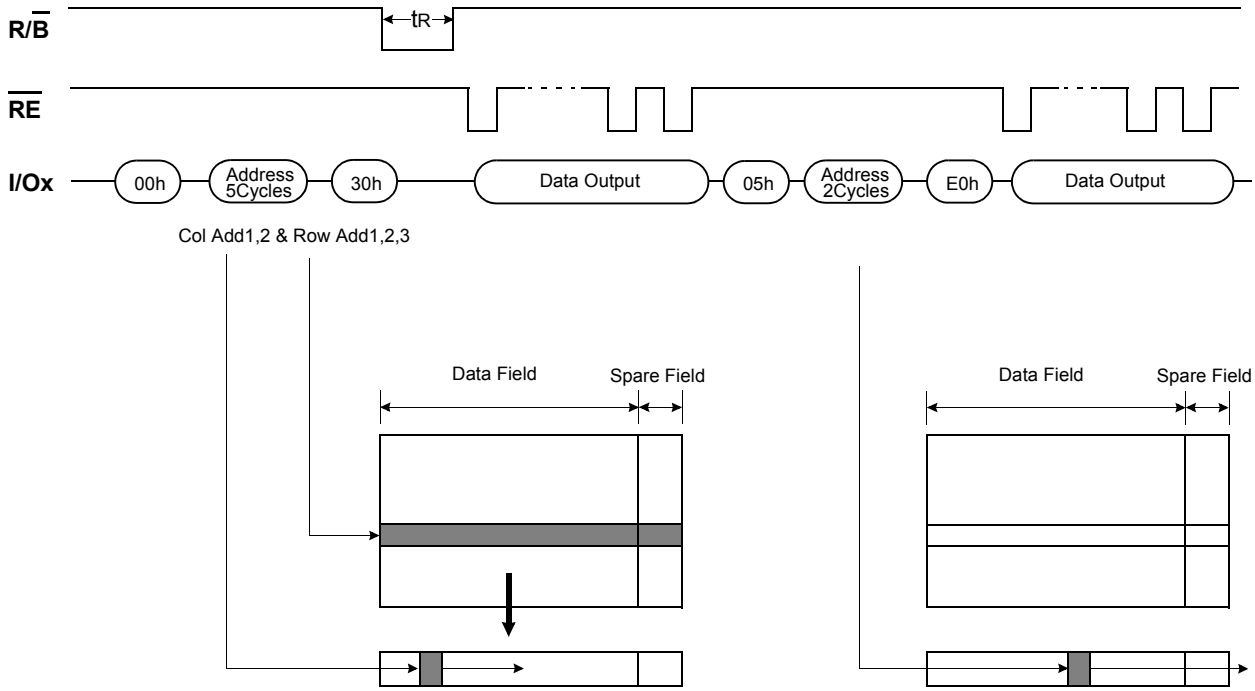


Figure 7. Random Data Output In a Page



PAGE PROGRAM

The device is programmed basically on a page basis, but it does allow multiple partial page programing of a word or consecutive bytes up to 2112, in a single page program cycle. The number of consecutive partial page programming operation within the same page without an intervening erase operation must not exceed 4 times for main array(1time/512byte) and 4 times for spare array(1time/16byte). The addressing should be done in sequential order in a block. A page program cycle consists of a serial data loading period in which up to 2112bytes of data may be loaded into the data register, followed by a non-volatile programming period where the loaded data is programmed into the appropriate cell.

The serial data loading period begins by inputting the Serial Data Input command(80h), followed by the five cycle address inputs and then serial data loading. The words other than those to be programmed do not need to be loaded. The device supports random data input in a page. The column address for the next data, which will be entered, may be changed to the address which follows random data input command(85h). Random data input may be operated multiple times regardless of how many times it is done in a page.

The Page Program confirm command(10h) initiates the programming process. Writing 10h alone without previously entering the serial data will not initiate the programming process. The internal write state controller automatically executes the algorithms and timings necessary for program and verify, thereby freeing the system controller for other tasks. Once the program process starts, the Read Status Register command may be entered to read the status register. The system controller can detect the completion of a program cycle by monitoring the R/$\overline{B}$ output, or the Status bit(I/O 6) of the Status Register. Only the Read Status command and Reset command are valid while programming is in progress. When the Page Program is complete, the Write Status Bit(I/O 0) may be checked(Figure 8). The internal write verify detects only errors for "1"s that are not successfully programmed to "0"s. The command register remains in Read Status command mode until another valid command is written to the command register.

Figure 8. Program & Read Status Operation

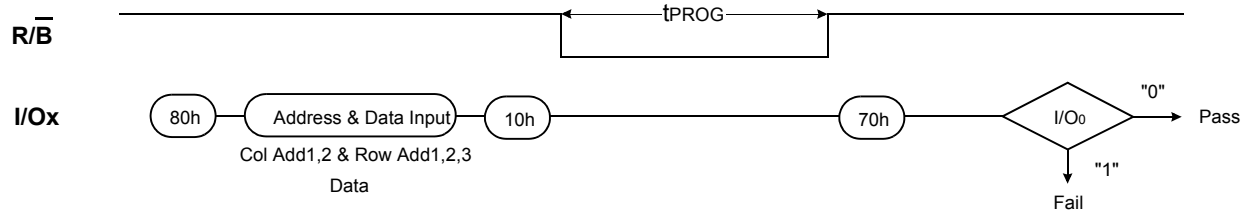
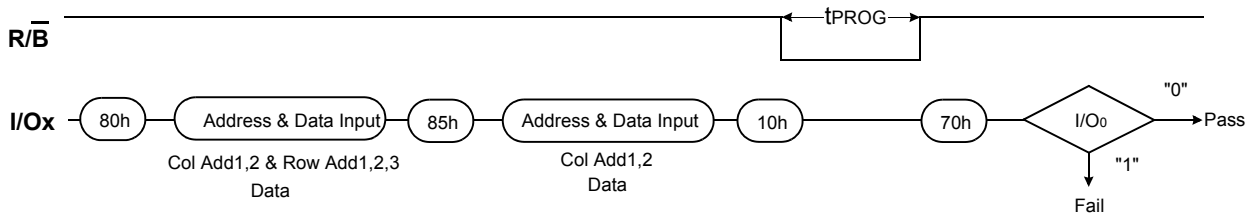


Figure 9. Random Data Input In a Page

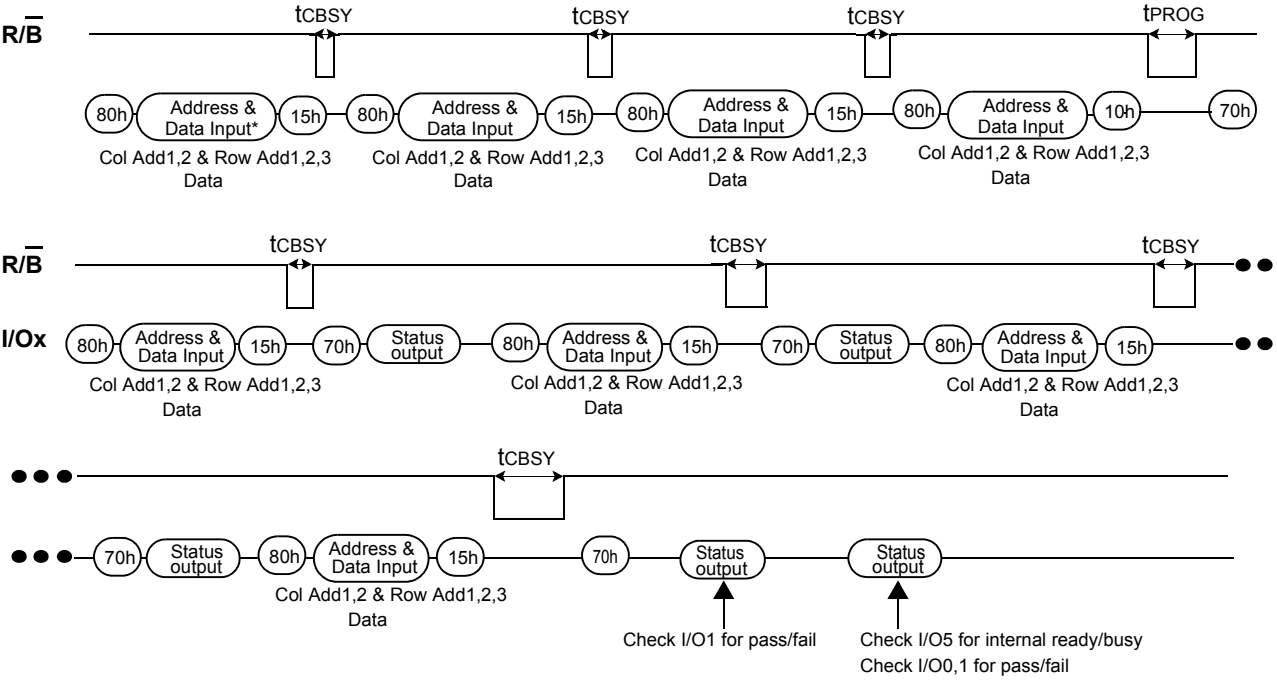


Cache Program

Cache Program is an extension of Page Program, which is executed with 2112byte data registers, and is available only within a block. Since the device has 1 page of cache memory, serial data input may be executed while data stored in data register are programmed into memory cell.

After writing the first set of data up to 2112byte into the selected cache registers, Cache Program command (15h) instead of actual Page Program (10h) is inputted to make cache registers free and to start internal program operation. To transfer data from cache registers to data registers, the device remains in Busy state for a short period of time(tCBSY) and has its cache registers ready for the next data-input while the internal programming gets started with the data loaded into data registers. Read Status command (70h) may be issued to find out when cache registers become ready by polling the Cache-Busy status bit(I/O 6). Pass/fail status of only the previouse page is available upon the return to Ready state. When the next set of data is inputted with the Cache Program command, tCBSY is affected by the progress of pending internal programming. The programming of the cache registers is initiated only when the pending program cycle is finished and the data registers are available for the transfer of data from cache registers. The status bit(I/O5) for internal Ready/Busy may be polled to identify the completion of internal programming. If the system monitors the progress of programming only with R/B, the last page of the target programming sequence must be progammed with actual Page Program command (10h). If the Cache Program command (15h) is used instead, status bit (I/O5) must be polled to find out when the last programming is actually finished before starting other operations such as read. Pass/fail status is available in two steps. I/O 1 returns with the status of the previous page upon Ready or I/O6 status bit changing to "1", and later I/O 0 with the status of current page upon true Ready (returning from internal programming) or I/O 5 status bit changing to "1". I/O 1 may be read together when I/O 0 is checked.

Figure 10. Cache Program (available only within a block)



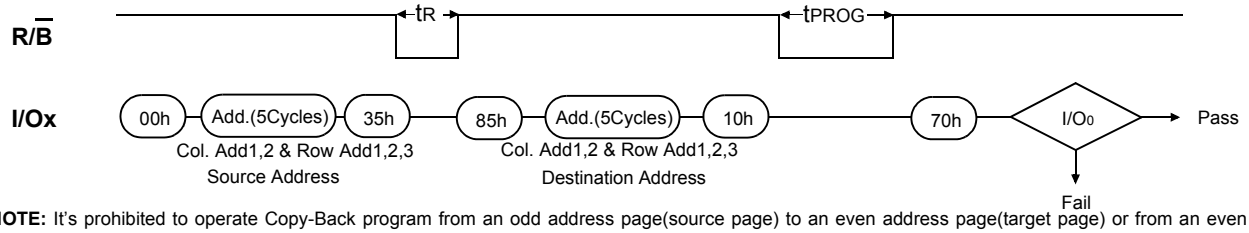
NOTE : Since programming the last page does not employ caching, the program time has to be that of Page Program. However, if the previous program cycle with the cache data has not finished, the actual program cycle of the last page is initiated only after completion of the previous cycle, which can be expressed as the following formula.

tPROG= Program time for the last page+ Program time for the (last -1)th page
- (Program command cycle time + Last page data loading time)

Copy-Back Program

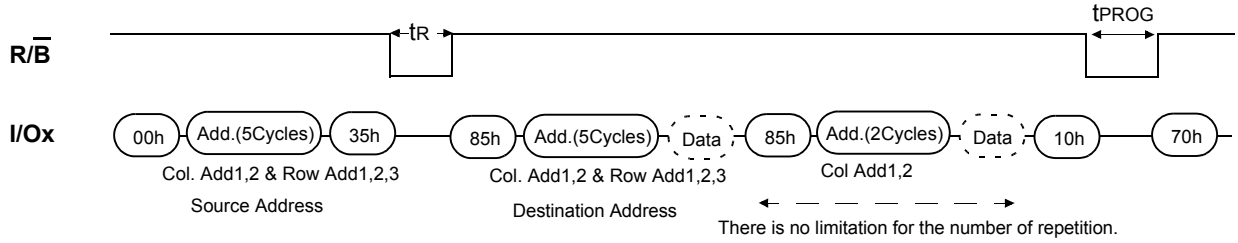
The copy-back program is configured to quickly and efficiently rewrite data stored in one page without utilizing an external memory. Since the time-consuming cycles of serial access and re-loading cycles are removed, the system performance is improved. The benefit is especially obvious when a portion of a block is updated and the rest of the block also need to be copied to the newly assigned free block. The operation for performing a copy-back program is a sequential execution of page-read without serial access and copying-program with the address of destination page. A read operation with "35h" command and the address of the source page moves the whole 2112byte data into the internal data buffer. As soon as the device returns to Ready state, Page-Copy Data-input command (85h) with the address cycles of destination page followed may be written. The Program Confirm command (10h) is required to actually begin the programming operation. Copy-Back Program operation is allowed only within the same memory plane. Once the Copy-Back Program is finished, any additional partial page programming into the copied pages is prohibited before erase. A27 must be the same between source and target page. Data input cycle for modifying a portion or multiple distant portions of the source page is allowed as shown in Figure 11. **"When there is a program-failure at Copy-Back operation, error is reported by pass/fail status. But if the soure page has an error bit by charge loss, accumulated copy-back operations could also accumulate bit errors. In this case, verifying the source page for a bit error is recommended before Copy-back program"**

Figure 11. Page Copy-Back program Operation



NOTE: It's prohibited to operate Copy-Back program from an odd address page(source page) to an even address page(target page) or from an even address page(source page) to an odd address page(target page). Therefore, the Copy-Back program is permitted just between odd address pages or even address pages.

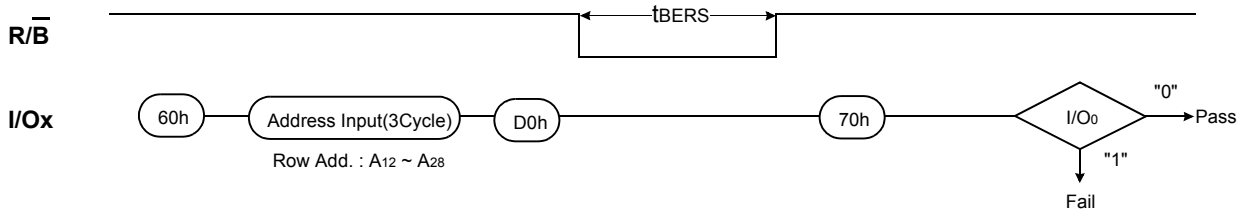
Figure 12. Page Copy-Back program Operation with Random Data Input



BLOCK ERASE

The Erase operation is done on a block basis. Block address loading is accomplished in three cycles initiated by an Erase Setup command(60h). Only address A_{18} to A_{28} is valid while A_{12} to A_{17} is ignored. The Erase Confirm command(D0h) following the block address loading initiates the internal erasing process. This two-step sequence of setup followed by execution command ensures that memory contents are not accidentally erased due to external noise conditions.
At the rising edge of $\overline{WE}$ after the erase confirm command input, the internal write controller handles erase and erase-verify. When the erase operation is completed, the Write Status Bit(I/O 0) may be checked. Figure 13 details the sequence.

Figure 13. Block Erase Operation



READ STATUS

The device contains a Status Register which may be read to find out whether program or erase operation is completed, and whether the program or erase operation is completed successfully. After writing 70h command to the command register, a read cycle outputs the content of the Status Register to the I/O pins on the falling edge of $\overline{CE}$ or $\overline{RE}$, whichever occurs last. This two line control allows the system to poll the progress of each device in multiple memory connections even when R/$\overline{B}$ pins are common-wired. $\overline{RE}$ or $\overline{CE}$ does not need to be toggled for updated status. Refer to table 2 for specific Status Register definitions. The command register remains in Status Read mode until further commands are issued to it. Therefore, if the status register is read during a random read cycle, the read command(00h) should be given before starting read cycles.

Table2. Read Staus Register Definition

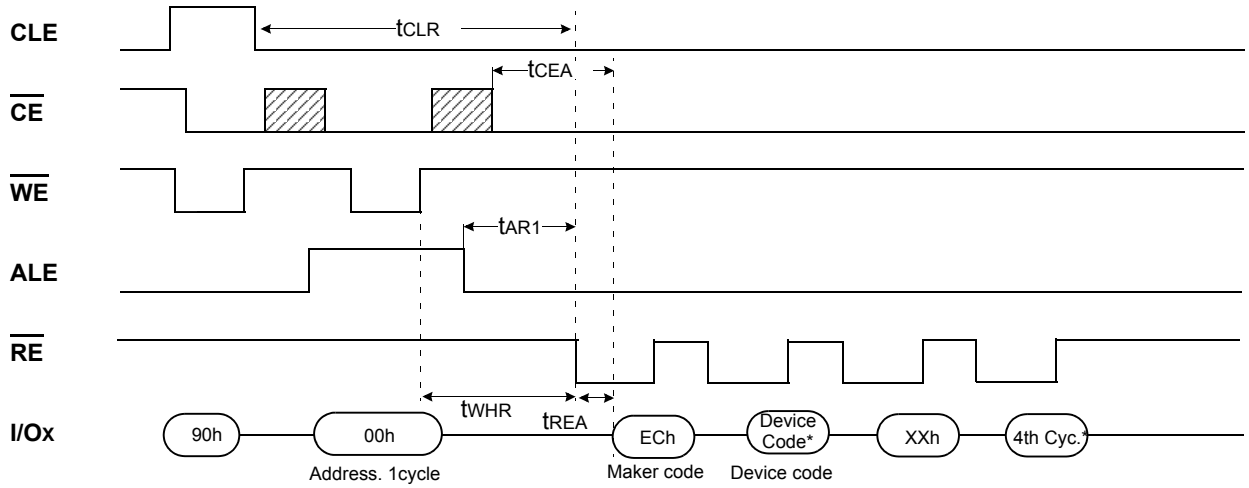
I/O No.	Page Program	Block Erase	Cache Prorgam	Read	Definition	
I/O 0	Pass/Fail	Pass/Fail	Pass/Fail(N)	Not use	Pass : "0"	Fail : "1"
I/O 1	Not use	Not use	Pass/Fail(N-1)	Not use	Pass : "0"	Fail : "1"
I/O 2	Not use	Not use	Not use	Not use	Don't -cared	
I/O 3	Not Use	Not Use	Not Use	Not Use	Don't -cared	
I/O 4	Not Use	Not Use	Not Use	Not Use	Don't -cared	
I/O 5	Ready/Busy	Ready/Busy	True Ready/Busy	Ready/Busy	Busy : "0"	Ready : "1"
I/O 6	Ready/Busy	Ready/Busy	Ready/Busy	Ready/Busy	Busy : "0"	Ready : "1"
I/O 7	Write Protect	Write Protect	Write Protect	Write Protect	Protected : "0"	Not Protected

NOTE : 1. True Ready/Busy represents internal program operation status which is being executed in cache program mode.
2. I/Os defined 'Not use' are recommended to be masked out when Read Status is being executed.

Read ID

The device contains a product identification mode, initiated by writing 90h to the command register, followed by an address input of 00h. Five read cycles sequentially output the manufacturer code(ECh), and the device code and XXh, 4th cycle ID, 15h respectively. The command register remains in Read ID mode until further commands are issued to it. Figure 11 shows the operation sequence.

Figure 14. Read ID Operation



Device	Device Code*(2nd Cycle)	4th Cycle*
K9K2G08R0A	AAh	15h
K9K2G08U0A	DAh	15h

RESET

The device offers a reset feature, executed by writing FFh to the command register. When the device is in Busy state during random read, program or erase mode, the reset operation will abort these operations. The contents of memory cells being altered are no longer valid, as the data will be partially programmed or erased. The command register is cleared to wait for the next command, and the Status Register is cleared to value C0h when $\overline{WP}$ is high. Refer to table 3 for device status after reset operation. If the device is already in reset state a new reset command will be accepted by the command register. The R/$\overline{B}$ pin transitions to low for tRST after the Reset command is written. Refer to Figure 12 below.

Figure 15. RESET Operation

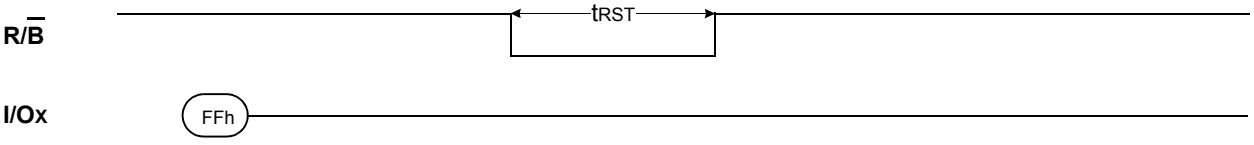


Table3. Device Status

	After Power-up	After Reset
Operation Mode	00h command is latched	Waiting for next command

READY/$\overline{BUSY}$

The device has a R/$\overline{B}$ output that provides a hardware method of indicating the completion of a page program, erase and random read. The R/$\overline{B}$ pin is normally high but transitions to low after program or erase command is written to the command register or random read is started after address loading. It returns to high when the internal controller has finished the operation. The pin is an open-drain driver thereby allowing two or more R/$\overline{B}$ outputs to be Or-tied. Because pull-up resistor value is related to tr(R/$\overline{B}$) and current drain during busy(ibusy) , an appropriate value can be obtained with the following reference chart(Fig 13). Its value can be determined by the following guidance.

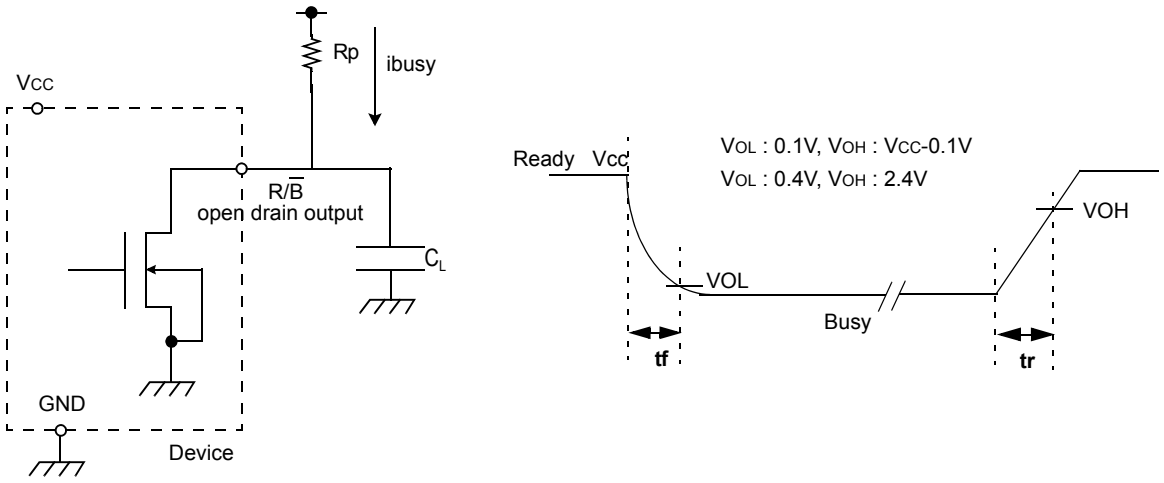
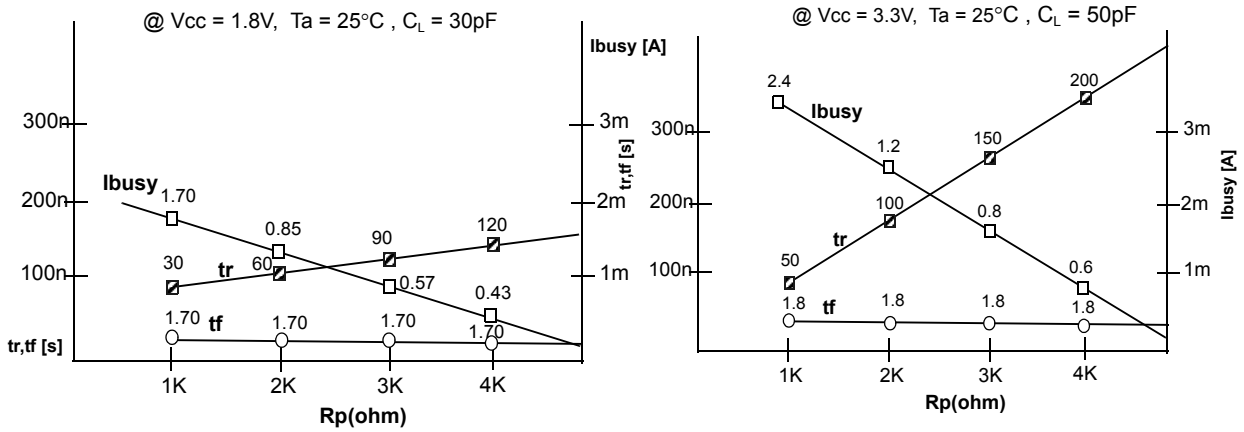


Figure 16. Rp vs tr ,tf & Rp vs ibusy



Rp value guidance

$$Rp(min, 1.8V part) = \frac{V_{CC}(Max.) - V_{OL}(Max.)}{I_{OL} + \Sigma I_L} = \frac{1.85V}{3mA + \Sigma I_L}$$

$$Rp(min, 3.3V part) = \frac{V_{CC}(Max.) - V_{OL}(Max.)}{I_{OL} + \Sigma I_L} = \frac{3.2V}{8mA + \Sigma I_L}$$

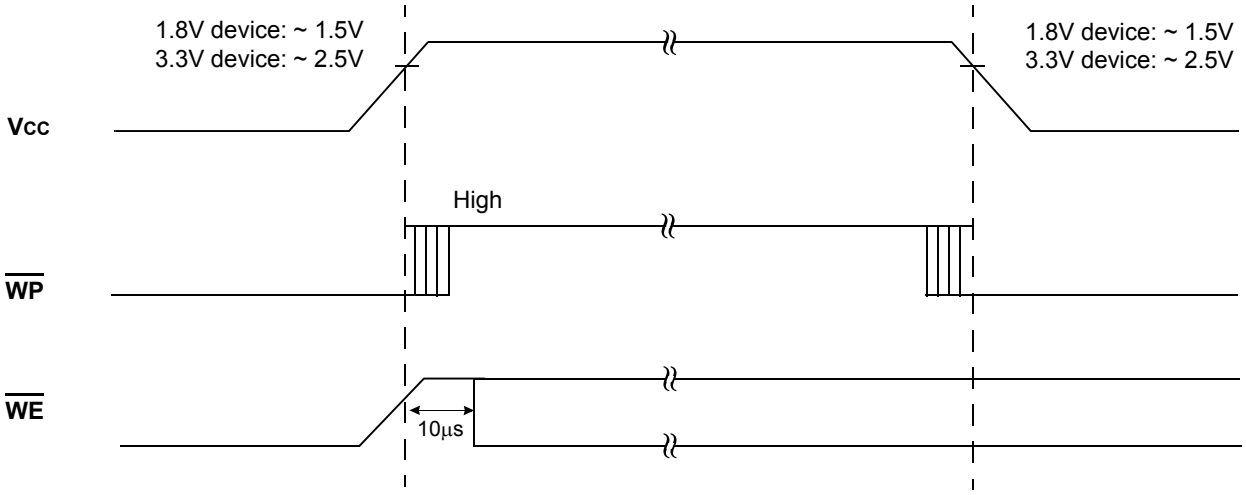
where I_L is the sum of the input currents of all devices tied to the R/$\overline{B}$ pin.

Rp(max) is determined by maximum permissible limit of tr

Data Protection & Power up sequence

The device is designed to offer protection from any involuntary program/erase during power-transitions. An internal voltage detector disables all functions whenever Vcc is below about 1.1V(1.8V device) and 2V(3.3V device). $\overline{WP}$ pin provides hardware protection and is recommended to be kept at V_{IL} during power-up and power-down. A recovery time of minimum 10μs is required before internal circuit gets ready for any command sequences as shown in Figure 14. The two step command sequence for program/erase provides additional software protection.

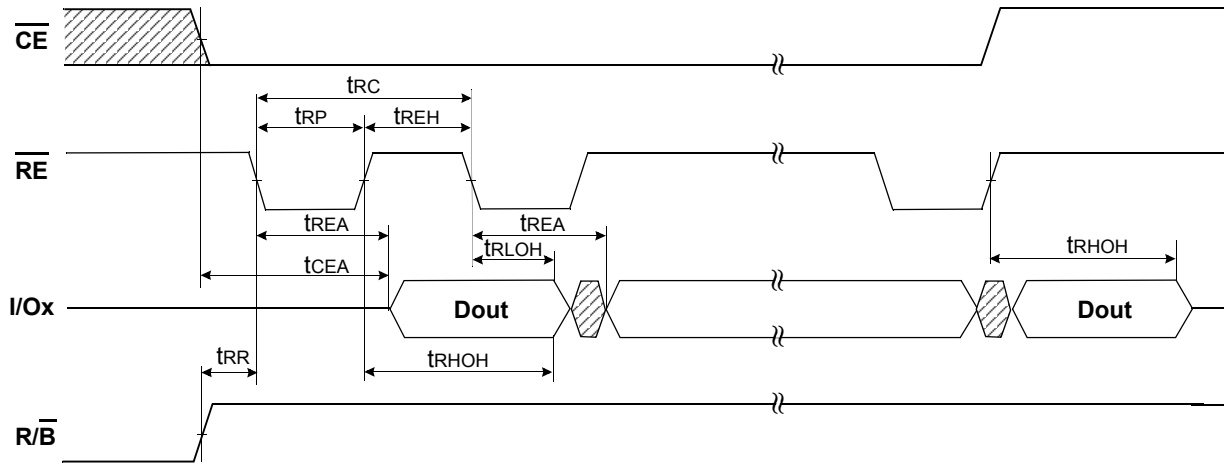
Figure 17. AC Waveforms for Power Transition



Extended Data Out Mode

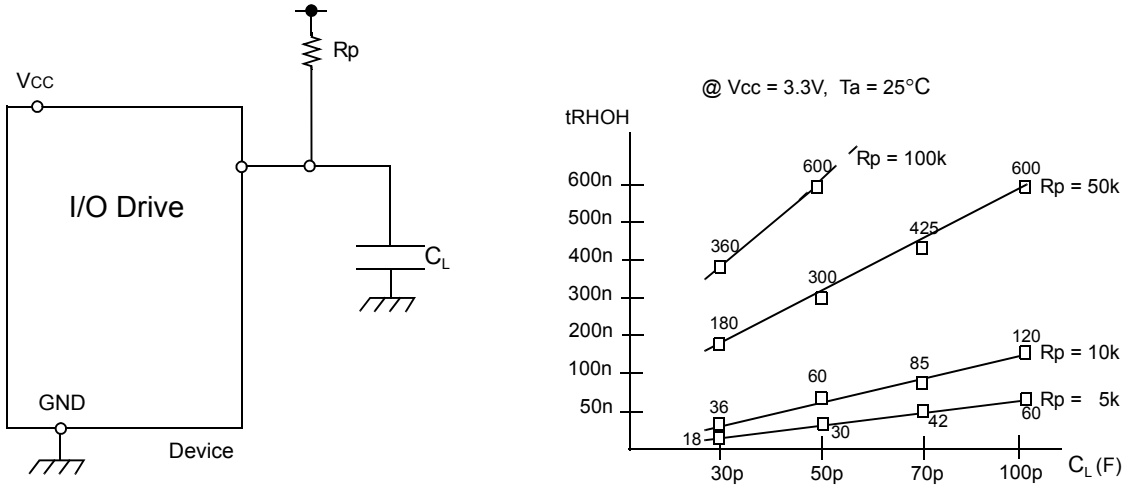
For the EDO mode, the device should hold the data on the system memory bus until the beginning of the next cycle, so that controller could fetch the data at the falling edge. However NAND flash dosen't support the EDO mode exactly.
The device stops the data input into the I/O bus after $\overline{RE}$ rising edge. But since the previous data remains in the I/O bus, the flow of I/O data seems like Figure 18 and the system can access serially the data with EDO mode. tRLOH which is the parameter for fetching data at RE falling time is necessary. Its appropriate value can be obtained with the reference chart as shown in Figure 19. The tRHOH value depands on output load(C_L) and I/O bus Pull-up resistor (Rp).

Figure 18. Serial Access Cycle after Read(EDO Type, CLE=L, $\overline{WE}$=H, ALE=L)



NOTES : Transition is measured at ±200mV from steady state voltage with load.
This parameter is sampled and not 100% tested.

Figure 19. Rp vs tRHOH vs C_L



tRLOH / tRHOH value guidance

$tRHOH = C_L * V_{OL} * Rp / Vcc$

tRLOH(min, 3.3V part) = tRHOH - tREH

Data Sheet | February 18, 2005 | FN9174.0

High Input Voltage Charger

The ISL6294 is a cost-effective, fully integrated high input voltage single-cell Li-ion battery charger. The charger uses a CC/CV charge profile required by Li-ion batteries. The charger accepts an input voltage up to 28V but is disabled when the input voltage exceeds the OVP threshold, typically 6.8V, to prevent excessive power dissipation. The 28V rating eliminates the overvoltage protection circuit required in a low input voltage charger.

The charge current and the end-of-charge (EOC) current are programmable with external resistors. When the battery voltage is lower than typically 2.55V, the charger preconditions the battery with typically 20% of the programmed charge current. When the charge current reduces to the programmable EOC current level during the CV charge phase, an EOC indication is provided by the CHG pin, which is an open-drain output. An internal thermal foldback function protects the charger from any thermal failure.

Two indication pins (PPR and CHG) allow simple interface to a microprocessor or LEDs. When no adapter is attached or when disabled, the charger draws less than 1µA leakage current from the battery.

Ordering Information

PART NUMBER	TEMP. RANGE (°C)	PACKAGE	PKG. DWG. #
ISL6294IRZ (Note)	-40 to 85	8 Ld 2x3 DFN (Pb-free)	L8.2x3
ISL6294IRZ-T (Note)	-40 to 85	8 Ld 2x3 DFN (Pb-free)	L8.2x3

NOTE: Intersil Pb-free products employ special Pb-free material sets; molding compounds/die attach materials and 100% matte tin plate termination finish, which are RoHS compliant and compatible with both SnPb and Pb-free soldering operations. Intersil Pb-free products are MSL classified at Pb-free peak reflow temperatures that meet or exceed the Pb-free requirements of IPC/JEDEC J STD-020.

Features

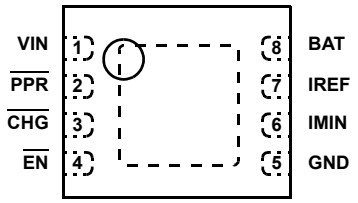
- Complete Charger for Single-Cell Li-ion/Polymer Batteries
- Integrated Pass Element and Current Sensor
- No External Blocking Diode Required
- Low Component Count and Cost
- 1% Voltage Accuracy
- Programmable Charge Current
- **Programmable End-of-Charge Current**
- **Charge Current Thermal Foldback for Thermal Protection**
- Trickle Charge for Fully Discharged Batteries
- **28V Maximum Voltage for the Power Input**
- Power Presence and Charge Indications
- Less Than 1µA Leakage Current off the Battery When No Input Power Attached or Charger Disabled
- Ambient Temperature Range: -40°C to 85°C
- **2x3 DFN-8 Packages**
- Pb-Free Available (RoHS Compliant)

Applications

- Mobile Phones
- Blue-Tooth Devices
- PDAs
- MP3 Players
- Stand-Alone Chargers
- Other Handheld Devices

Pinout

DFN 8 LEAD
TOP VIEW



Absolute Maximum Ratings (Reference to GND)

VIN . . . . . -0.3V to 30V
IMIN, IREF, BAT, CHG, EN, PPR . . . . . -0.3V to 7V
ESD Rating
Human Body Model (Per EIA JESD22 Method A114-B) 3kV
Machine Model (Per EIA JED-4701 Method C-111) 200V

Recommended Operating Conditions

Ambient Temperature Range . . . . . -40°C to 85°C
Maximum Supply Voltage (VIN Pin) 28V
Operating Supply Voltage (VIN Pin) . . . . . 4.5V to 6.5V
Programmed Charge Current . . . . . 100mA to 700mA

Thermal Information

Thermal Resistance	θ_{JA} (°C/W)	θ_{JC} (°C/W)
DFN Package (Notes 1, 2)	78	11

Maximum Junction Temperature (Plastic Package) . . . . . 150°C
Maximum Storage Temperature Range . . . . . -65°C to 150°C
Maximum Lead Temperature (Soldering 10s) . . . . . 300°C

CAUTION: Stresses above those listed in "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress only rating and operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied.

NOTES:

1. θ_{JA} is measured in free air with the component mounted on a high effective thermal conductivity test board with "direct attach" features. See Tech Brief TB379.
2. For θ_{JC}, the "case temp" location is the center of the exposed metal pad on the package underside.

Electrical Specifications

Typical Values Are Tested at VIN = 5V and the Ambient Temperature at 25°C. All Maximum and Minimum Values Are Guaranteed Under the Recommended Operating Supply Voltage Range and Ambient Temperature Range, Unless Otherwise Noted.

PARAMETER	SYMBOL	TEST CONDITIONS	MIN	TYP	MAX	UNITS
POWER-ON RESET						
Rising POR Threshold	V_{POR}	VBAT = 3.0V, use PPR to indicate the comparator output.	3.3	3.9	4.3	V
Falling POR Threshold	V_{POR}		3.1	3.6	4.15	V
VIN-BAT OFFSET VOLTAGE						
Rising Edge	V_{OS}	V_{BAT} = 4.0V, use CHG pin to indicate the comparator output (Note 3)	-	90	150	mV
Falling Edge	V_{OS}		10	50	-	mV
OVER VOLTAGE PROTECTION						
Over Voltage Protection Threshold	V_{OVP}	(Note 4) Use PPR to indicate the comparator output	6.5	6.8	7.1	V
OVP Threshold Hysteresis			100	240	400	mV
STANDBY CURRENT						
BAT Pin Sink Current	$I_{STANDBY}$	Charger disabled or the input is floating	-	-	1.0	µA
VIN Pin Supply Current	I_{VIN}	Charger disabled	-	300	400	µA
VIN Pin Supply Current	I_{VIN}	Charger enabled	-	400	600	µA
VOLTAGE REGULATION						
Output Voltage	V_{CH}	4.3V < V_{IN} < 6.5V, charge current = 20mA	4.158	4.20	4.242	V
PMOS On Resistance	$r_{DS(ON)}$	V_{BAT} = 3.8V, charge current = 0.5A	-	0.6	-	Ω
CHARGE CURRENT (Note 5)						
IREF Pin Output Voltage	I_{IREF}	V_{BAT} = 3.8V	1.18	1.22	1.26	V
Constant Charge Current	I_{CHG}	R_{IREF} = 24.3kΩ, V_{BAT} = 2.8V - 4.0V	450	500	550	mA
Trickle Charge Current	I_{TRK}	R_{IREF} = 24.3kΩ, V_{BAT} = 2.4V	70	95	130	mA
End-of-Charge Current	I_{MIN}	R_{IMIN} = 243kΩ	33	45	57	mA
EOC Rising Threshold		R_{IMIN} = 243kΩ	325	380	415	mA
PRECONDITIONING CHARGE THRESHOLD						
Preconditioning Charge Threshold Voltage	V_{MIN}		2.45	2.55	2.65	V
Preconditioning Voltage Hysteresis	V_{MINHYS}		40	100	150	mV

Electrical Specifications Typical Values Are Tested at VIN = 5V and the Ambient Temperature at 25°C. All Maximum and Minimum Values Are Guaranteed Under the Recommended Operating Supply Voltage Range and Ambient Temperature Range, Unless Otherwise Noted. **(Continued)**

PARAMETER	SYMBOL	TEST CONDITIONS	MIN	TYP	MAX	UNITS
INTERNAL TEMPERATURE MONITORING						
Charge Current Foldback Threshold (Note 6)	T_{FOLD}		100	115	130	°C
LOGIC INPUT AND OUTPUTS						
EN Pin Logic Input High			1.3	-	-	V
EN Pin Logic Input Low			-	-	0.5	V
EN Pin Internal Pull Down Resistance			100	200	400	kΩ
CHG Sink Current when LOW		Pin Voltage = 1V	10	20	-	mA
CHG Leakage Current When HIGH		V_{CHG} = 6.5V	-	-	1	µA
PPR Sink Current when LOW		Pin Voltage = 1V	10	20	-	mA
PPR Leakage Current When HIGH		V_{PPR} 6= 6.5V	-	-	1	µA

NOTES:

3. The 4.0V V_{BAT} is selected so that the CHG output can be used as the indication for the offset comparator output indication. If the V_{BAT} is lower than the POR threshold, no output pin can be used for indication.
4. For junction temperature below 100 °C.
5. The charge current can be affected by the thermal foldback function if the IC under the test setup cannot dissipate the heat.
6. This parameter is guaranteed by design, not tested.

Pin Descriptions

VIN - Power input. The absolute maximum input voltage is 28V. A 0.47µF or larger value X5R ceramic capacitor is recommended to be placed very close to the input pin for decoupling purpose. Additional capacitance may be required to provide a stable input voltage.

PPR - Open-drain power presence indication. The open-drain MOSFET turns on when the input voltage is above the POR threshold but below the OVP threshold and off otherwise. This pin is capable to sink 10mA (minimum) current to drive an LED. The maximum voltage rating for this pin is 7V. This pin is independent on the EN-pin input.

CHG - Open-drain charge indication pin. This pin outputs a logic LOW when a charge cycle starts and turns to HIGH when the end-of-charge (EOC) condition is qualified. This pin is capable to sink 10mA min. current to drive an LED. When the charger is disabled, the CHG outputs high impedance.

EN - Enable input. This is a logic input pin to disable or enable the charger. Drive to HIGH to disable the charger. When this pin is driven to LOW or left floating, the charger is enabled. This pin has an internal 200kΩ pull-down resistor.

GND - System ground.

IMIN - End-of-charge (EOC) current program pin. Connect a resistor between this pin and the GND pin to set the EOC current. The EOC current IMIN can be programmed by the following equation:

$$I_{MIN} = \frac{11000}{R_{IMIN}} \quad (mA)$$

Where R_{IMIN} is in kΩ. The programmable range covers 5% (or 10mA, whichever is higher) to 50% of IREF. When programmed to less than 5% or 10mA, the stability is not guaranteed.

IREF - Charge-current program and monitoring pin. Connect a resistor between this pin and the GND pin to set the charge current limit determined by the following equation:

$$I_{REF} = \frac{12089}{R_{IREF}} \quad (mA)$$

Where R_{IREF} is in kΩ. The IREF pin voltage also monitors the actual charge current during the entire charge cycle, including the trickle, constant-current, and constant-voltage phases. When disabled, VIREF = 0V.

BAT - Charger output pin. Connect this pin to the battery. A 1µF or larger X5R ceramic capacitor is recommended for decoupling and stability purposes. When the EN pin is pulled to logic HIGH, the BAT output is disabled.

EPAD - Exposed pad. Connect as much as possible copper to this pad either on the component layer or other layers through thermal vias to enhance the thermal performance.

Typical Applications

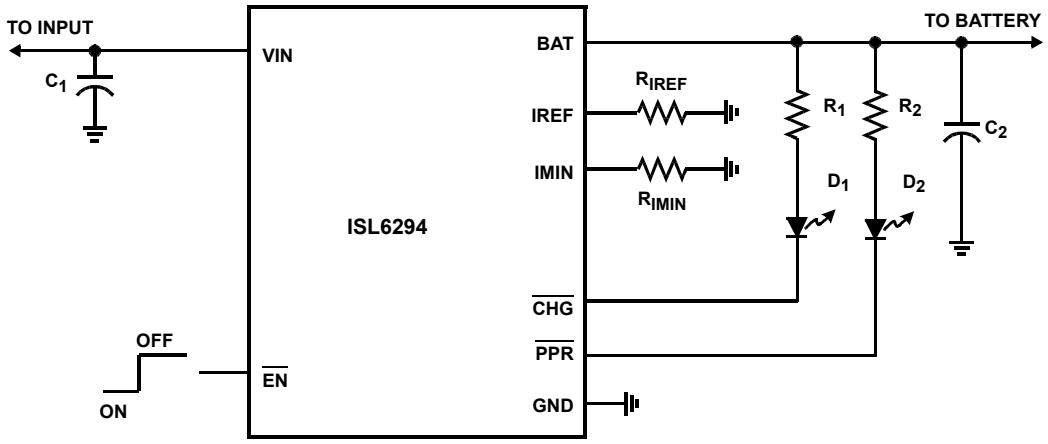


FIGURE 1. TYPICAL APPLICATION CIRCUIT INTERFACING TO INDICATION LEDs

COMPONENT DESCRIPTION FOR FIGURE 1

PART	DESCRIPTION
C_1	1µF X5R ceramic cap
C_2	1µF X5R ceramic cap
R_{IREF}	24.3kΩ, 1%, for 500mA charge current
R_{IMIN}	243kΩ, 1%, for 45mA EOC current
R_1, R_2	300Ω, 5%
D_1, D_2	LEDs for indication

COMPONENT DESCRIPTION FOR FIGURE 2

PART	DESCRIPTION
C_1	1µF X5R ceramic cap
C_2	1µF X5R ceramic cap
R_{IREF}	24.3kΩ, 1%, for 500mA charge current
R_{IMIN}	243kΩ, 1%, for 45mA EOC current
R_1, R_2	100kΩ, 5%

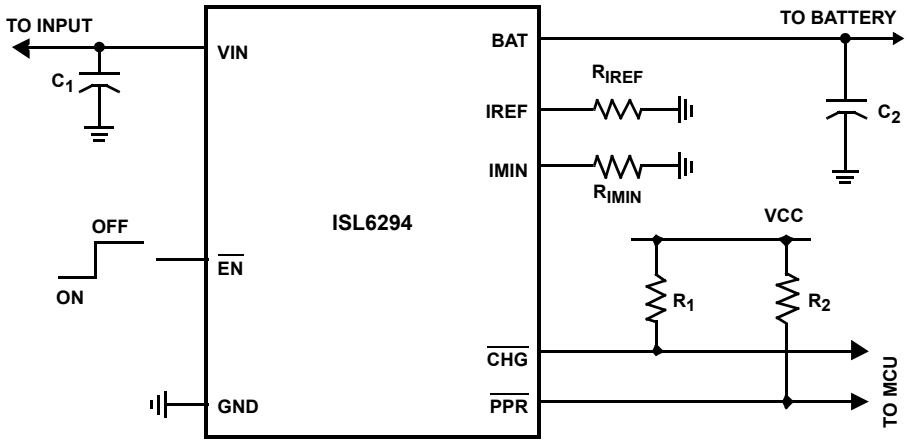


FIGURE 2. TYPICAL APPLICATION CIRCUIT WITH THE INDICATION SIGNALS INTERFACING TO A MCU

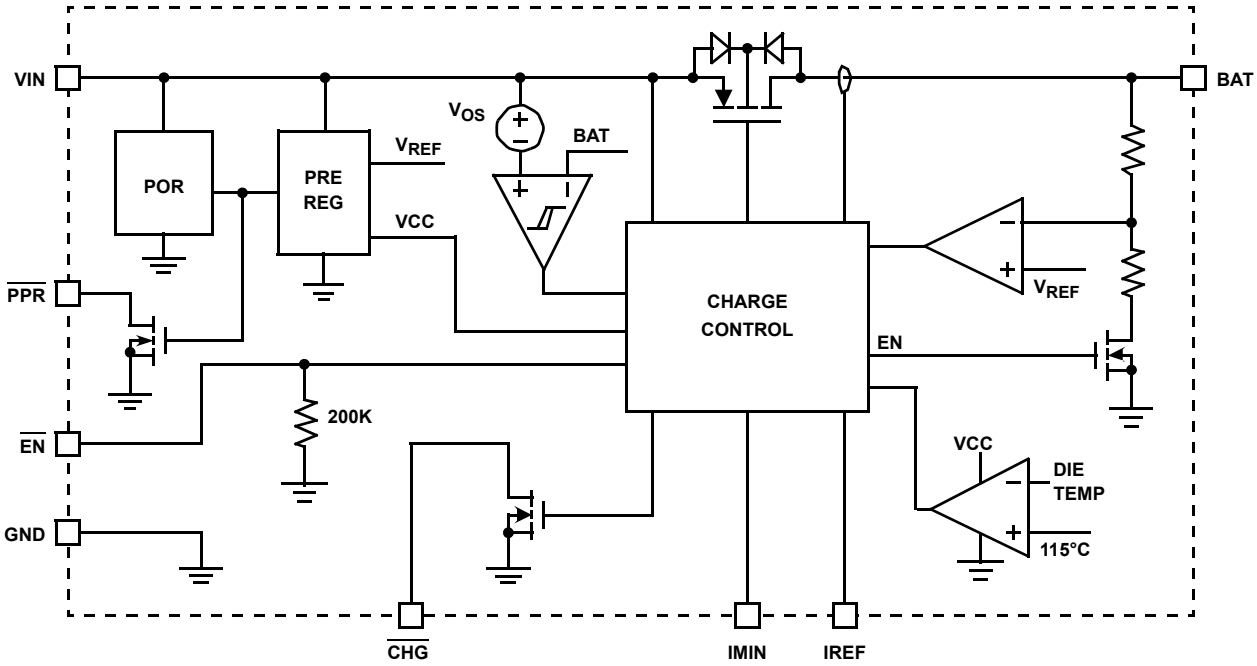


FIGURE 3. BLOCK DIAGRAM

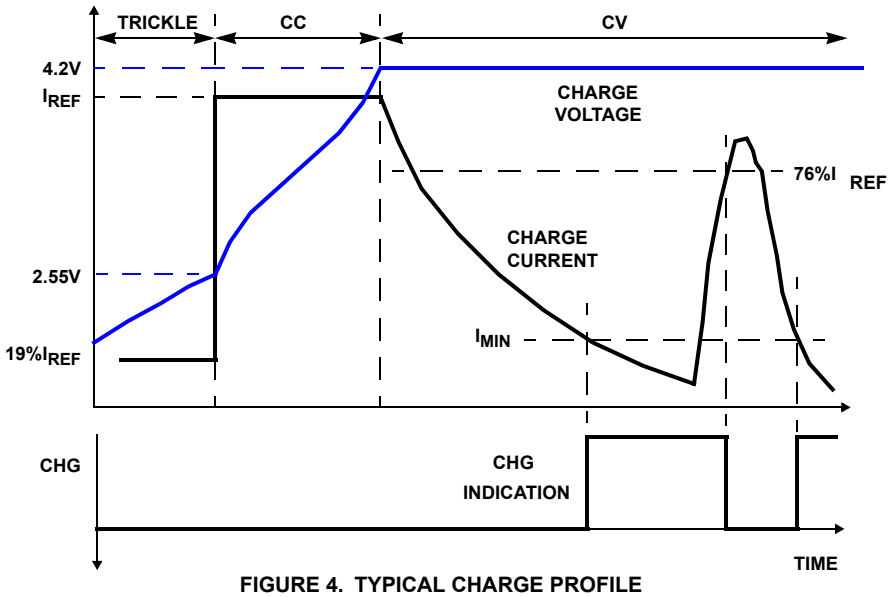


FIGURE 4. TYPICAL CHARGE PROFILE

Description

The ISL6294 charges a Li-ion battery using a CC/CV profile. The constant current I_{REF} is set with the external resistor R_{IREF} (See Figure 1) and the constant voltage is fixed at 4.2V. If the battery voltage is below a typical 2.55V trickle-charge threshold, the ISL6294 charges the battery with a trickle current of 19% of I_{REF} until the battery voltage rises above the trickle charge threshold. Fast charge CC mode is maintained at the rate determined by programming I_{REF} until the cell voltage rises to 4.2V. When the battery voltage reaches 4.2V, the charger enters a CV mode and regulates the battery voltage at 4.2V to fully charge the battery without the risk of over charge. Upon reaching an end-of-charge (EOC) current, the charger indicates the charge completion with the CHG pin, but the charger continues to output the 4.2V voltage. Figure 4 shows the typical charge waveforms after the power is on.

The EOC current level IMIN is programmable with the external resistor R_{IMIN} (See Figure 1). The CHG signal turns

to LOW when the trickle charge starts and rises to HIGH at the EOC. After the EOC is reached, the charge current has to rise to typically 76% I_{REF} for the CHG signal to turn on again, as shown in Figure 4. The current surge after EOC can be caused by a load connected to the battery.

A thermal foldback function reduces the charge current anytime when the die temperature reaches typically 115°C. This function guarantees safe operation when the printed-circuit board (PCB) is not capable of dissipating the heat generated by the linear charger. The ISL6294 accepts an input voltage up to 28V but disables charging when the input voltage exceeds the OVP threshold, typically 6.8V, to protect against unqualified or faulty ac adapters.

PPR Indication

The PPR pin is an open-drain output to indicate the presence of the ac adapter. Whenever the input voltage is higher than the POR threshold, the PPR pin turns on the internal open-drain MOSFET to indicate a logic LOW signal, independent on the EN-pin input. When the internal open-drain FET is turned off, the PPR pin should leak less than 1µA current. When turned on, the PPR pin should be able to sink at least 10mA current under all operating conditions.

The PPR pin can be used to drive an LED (see Figure 1) or to interface with a microprocessor.

Power-Good Range

The power-good range is defined by the following three conditions:

1. VIN > VPOR
2. VIN - VBAT > VOS
3. VIN < VOVP

where the VOS is the offset voltage for the input and output voltage comparator, discussed shortly, and the VOVP is the overvoltage protection threshold given in the Electrical Specification. All V_{POR}, V_{OS}, and V_{OVP} have hysteresis, as given in the Electrical Specification table. The charger will not charge the battery if the input voltage is not in the power-good range.

Input and Output Comparator

The charger will not be enabled unless the input voltage is higher than the battery voltage by an offset voltage VOS. The purpose of this comparator is to ensure that the charger is turned off when the input power is removed from the charger. Without this comparator, it is possible that the charger will fail to power down when the input is removed and the current can leak through the PFET pass element to continue biasing the POR and the Pre-Regulator blocks shown in the Block Diagram.

CHG Indication

The CHG is an open-drain output capable to at least 10mA current when the charger starts to charge and turns off when the EOC current is reached. The CHG signal is interfaced either with a micro-processor GPIO or an LED for indication.

EN Input

EN is an active-low logic input to enable the charger. Drive the EN pin to LOW or leave it floating to enable the charger. This pin has a 200kΩ internal pulldown resistor so when left floating, the input is equivalent to logic LOW. Drive this pin to HIGH to disable the charger. The threshold for HIGH is given in the ES (Electrical Specification) table.

IREF Pin

The IREF pin has the two functions as described in the Pin Description section. When setting the fast charge current, the charge current is guaranteed to have 10% accuracy with the charge current set at 500mA. When monitoring the charge current, the accuracy of the IREF pin voltage vs. the actual charge current has the same accuracy as the gain from the IREF pin current to the actual charge current. The accuracy is 10% at 500mA and is expected to drop to 30% of the actual current (not the set constant charge current) when the current drops to 50mA.

Operation Without the Battery

The ISL6294 relies on a battery for stability and is not guaranteed to be stable if the battery is not connected. With a battery, the charger will be stable with an output ceramic decoupling capacitor in the range of 1µF to 200µF. The maximum load current is limited by the dropout voltage or the thermal foldback.

Dropout Voltage

The constant current may not be maintained due to the $r_{DS(ON)}$ limit at a low input voltage. The worst case on resistance of the pass FET is 1.2Ω the maximum operating temperature, thus if tested with 0.5A current and 3.8V battery voltage, constant current could not be maintained when the input voltage is below 4.4V.

Thermal Foldback

The thermal foldback function starts to reduce the charge current when the internal temperature reaches a typical value of 115°C.

Applications Information

Input Capacitor Selection

The input capacitor is required to suppress the power supply transient response during transitions. Mainly this capacitor is selected to avoid oscillation during the start up when the input supply is passing the POR threshold and the VIN-BAT comparator offset voltage. When the battery voltage is above the POR threshold, the VIN-VBAT offset voltage dominates the hysteresis value. Typically, a 1µF X5R ceramic capacitor should be sufficient to suppress the power supply noise.

Output Capacitor Selection

The criteria for selecting the output capacitor is to maintain the stability of the charger as well as to bypass any transient load current. The minimum capacitance is a 1µF X5R ceramic capacitor. The actual capacitance connected to the output is dependent on the actual application requirement.

Charge Current Limit

The actual charge current in the CC mode is limited by several factors in addition to the set I_{REF}. Figure 5 shows three limits for the charge current in the CC mode. The charge current is limited by the on resistance of the pass element (power P-channel MOSFET) if the input and the output voltage are too close to each other. The solid curve shows a typical case when the battery voltage is 4.0V and the charge current is set to 700mA. The non-linearity on the R_{ON}-limited region is due to the increased resistance at higher die temperature. If the battery voltage increases to higher than 4.0V, the entire curve moves towards right side. As the input voltage increases, the charge current may be reduced due to the thermal foldback function. The limit caused by the thermal limit is dependent on the thermal impedance. As the thermal impedance increases, the thermal-limited curve moves towards left, as shown in Figure 5.

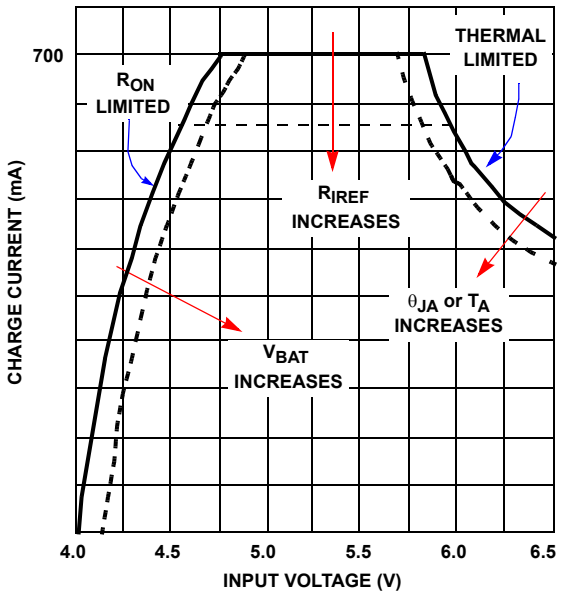


FIGURE 5. CHARGE CURRENT LIMITS IN THE CC MODE

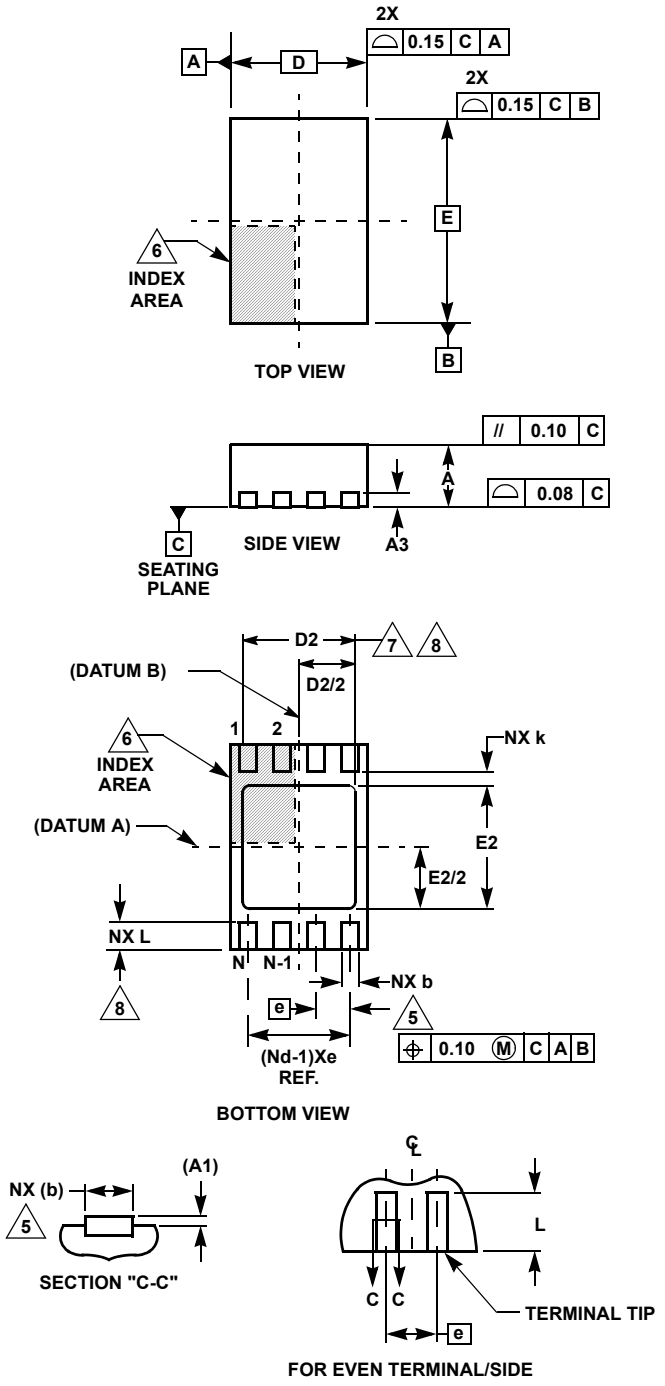
Layout Guidance

The ISL6294 uses a thermally-enhanced DFN package that has an exposed thermal pad at the bottom side of the package. The layout should connect as much as possible to copper on the exposed pad. Typically the component layer is more effective in dissipating heat. The thermal impedance can be further reduced by using other layers of copper connecting to the exposed pad through a thermal via array. Each thermal via is recommended to have 0.3mm diameter and 1mm distance from other thermal vias.

Input Power Sources

The input power source is typically a well-regulated wall cube with 1-meter length wire or a USB port. The input voltage ranges from 4.25V to 6.5V under full-load and unloaded conditions. The ISL6294 can withstand up to 28V on the input without damaging the IC. If the input voltage is higher than typically 6.8V, the charger stops charging.

Dual Flat No-Lead Plastic Package (DFN)



L8.2x3

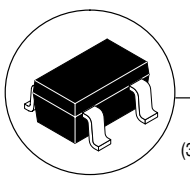
8 LEAD DUAL FLAT NO-LEAD PLASTIC PACKAGE

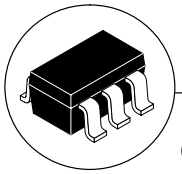
SYMBOL	MILLIMETERS			NOTES
	MIN	NOMINAL	MAX	
A	0.80	0.90	1.00	-
A1	-	-	0.05	-
A3		0.20 REF		-
b	0.20	0.25	0.32	5,8
D		2.00 BSC		-
D2	1.50	1.65	1.75	7,8
E		3.00 BSC		-
E2	1.65	1.80	1.90	7,8
e		0.50 BSC		-
k	0.20	-	-	-
L	0.30	0.40	0.50	8
N		8		2
Nd		4		3

Rev. 0 6/04

NOTES:

1. Dimensioning and tolerancing conform to ASME Y14.5-1994.
2. N is the number of terminals.
3. Nd refers to the number of terminals on D.
4. All dimensions are in millimeters. Angles are in degrees.
5. Dimension b applies to the metallized terminal and is measured between 0.25mm and 0.30mm from the terminal tip.
6. The configuration of the pin #1 identifier is optional, but must be located within the zone indicated. The pin #1 identifier may be either a mold or mark feature.
7. Dimensions D2 and E2 are for the exposed pads which provide improved electrical and thermal performance.
8. Nominal dimensions are provided to assist with PCB Land Pattern Design efforts, see Intersil Technical Brief TB389.





SLVS348C – JULY 2001 – REVISED APRIL 2002

ULTRALOW-NOISE, HIGH PSRR, FAST RF 200-mA LOW-DROPOUT LINEAR REGULATORS

FEATURES

- **200-mA Low-Dropout Regulator With EN**
- **Available in 1.8-V, 2.5-V, 2.8-V, 2.85-V, 3-V, 3.3-V, 4.75-V, and Adjustable**
- **High PSRR (70 dB at 10 kHz)**
- **Ultralow Noise (32 μV)**
- **Fast Start-Up Time (50 μs)**
- **Stable With a 2.2-μF Ceramic Capacitor**
- **Excellent Load/Line Transient**
- **Very Low Dropout Voltage (112 mV at Full Load, TPS79330)**
- **5-Pin SOT23 (DBV) Package**

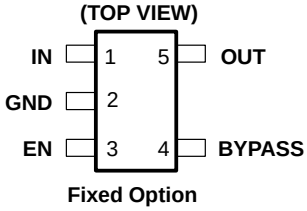
APPLICATIONS

- **Cellular and Cordless Telephones**
- **VCOs**
- **RF**
- **Bluetooth™, Wireless LAN**
- **Handheld Organizers, PDA**

DESCRIPTION

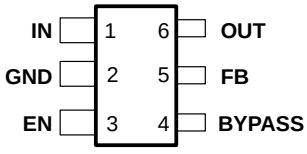
The TPS793xx family of low-dropout (LDO) low-power linear voltage regulators features high power supply rejection ratio (PSRR), ultralow noise, fast start-up, and excellent line and load transient responses in a small outline, SOT23, package. Each device in the family is stable, with a small 2.2-μF ceramic capacitor on the output. The TPS793xx family uses an advanced, proprietary BiCMOS fabrication process to yield extremely low dropout voltages (e.g., 112 mV at 200 mA, TPS79330). Each device achieves fast start-up times (approximately 50 μs with a 0.001-μF bypass capacitor) while consuming very low quiescent current (170 μA typical). Moreover, when the device is placed in standby mode, the supply current is reduced to less than 1 μA. The TPS79328 exhibits approximately 32 μV_{RMS} of output voltage noise with a 0.1-μF bypass capacitor. Applications with analog components that are noise sensitive, such as portable RF electronics, benefit from the high PSRR and low-noise features as well as the fast response time.

DBV PACKAGE
(TOP VIEW)



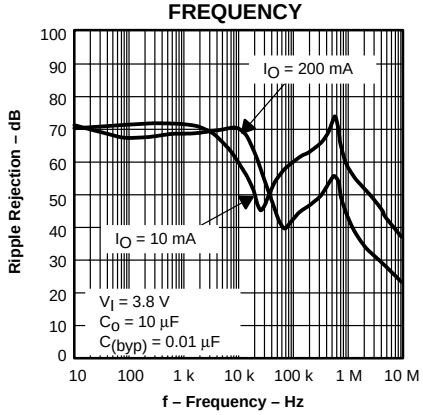
Fixed Option

DBV PACKAGE
(TOP VIEW)

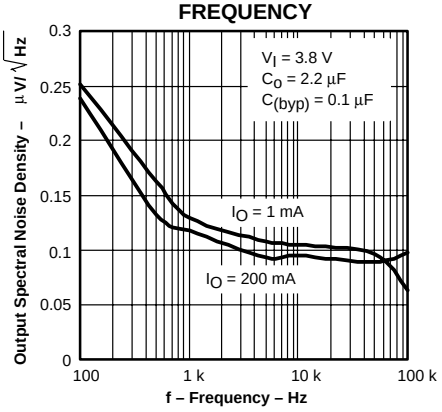


Adjustable Option

TPS79328
RIPPLE REJECTION
vs
FREQUENCY



TPS79328
OUTPUT SPECTRAL NOISE DENSITY
vs
FREQUENCY



Please be aware that an important notice concerning availability, standard warranty, and use in critical applications of Texas Instruments semiconductor products and disclaimers thereto appears at the end of this data sheet.

Bluetooth is a trademark owned by Bluetooth SIG, Inc.

AVAILABLE OPTIONS

T_J	VOLTAGE	PACKAGE	PART NUMBER	SYMBOL
–40°C to 125°C	1.2 to 5.5 V	SOT23 (DBV)	TPS79301DBVR†	PGVI
	1.8 V		TPS79318DBVR†	PHHI
	2.5 V		TPS79325DBVR†	PGWI
	2.8 V		TPS79328DBVR†	PGXI
	2.85 V		TPS793285DBVR†	PHII
	3 V		TPS79330DBVR†	PGYI
	3.3 V		TPS793333DBVR†	PHUI
	4.75 V		TPS793475DBVR†	PHJI

† The DBVR indicates tape and reel of 3000 parts.

absolute maximum ratings over operating free-air temperature range (unless otherwise noted)‡

Input voltage range (see Note 1) . . . . . –0.3 V to 6 V
Voltage range at EN . . . . . –0.3 V to V_I + 0.3 V
Voltage on OUT . . . . . –0.3 V to 6 V
Peak output current . . . . . internally limited
ESD rating, HBM . . . . . 2 kV
ESD rating, CDM . . . . . 500 V
Continuous total power dissipation . . . . . See Dissipation Rating Table
Operating virtual junction temperature range, T_J . . . . . –40°C to 150°C
Operating ambient temperature range, T_A . . . . . –40°C to 85°C
Storage temperature range, T_{stg} . . . . . –65°C to 150°C

‡ Stresses beyond those listed under “absolute maximum ratings” may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated under “recommended operating conditions” is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

NOTE 1: All voltage values are with respect to network ground terminal.

DISSIPATION RATING TABLE

BOARD	PACKAGE	$R_{\theta JC}$	$R_{\theta JA}$	DERATING FACTOR ABOVE T_A = 25°C	$T_A \le 25°C$ POWER RATING	T_A = 70°C POWER RATING	T_A = 85°C POWER RATING
Low K§	DBV	63.75 °C/W	256 °C/W	3.906 mW/°C	391 mW	215 mW	156 mW
High K¶	DBV	63.75 °C/W	178.3 °C/W	5.609 mW/°C	561 mW	308 mW	224 mW

§ The JEDEC low K (1s) board design used to derive this data was a 3-inch x 3-inch, two layer board with 2 ounce copper traces on top of the board.
¶ The JEDEC high K (2s2p) board design used to derive this data was a 3-inch x 3-inch, multilayer board with 1 ounce internal power and ground planes and 2 ounce copper traces on top and bottom of the board.

electrical characteristics over recommended operating free-air temperature range EN = V_I, T_J = –40 to 125 °C, V_I = $V_{O(typ)}$ + 1 V, I_O = 1 mA, C_o = 10 µF, $C_{(byp)}$ = 0.01 µF (unless otherwise noted)

PARAMETER		TEST CONDITIONS		MIN	TYP	MAX	UNIT
V_I Input voltage (see Note 2)				2.7		5.5	V
I_O Continuous output current (see Note 3)				0		200	mA
T_J Operating junction temperature				–40		125	°C
Output voltage	TPS79301	0 µA< I_O < 200 mA, (see Note 4)	1.22 V ≤ V_O ≤ 5.2 V,	0.98 V_O		1.02 V_O	V
	TPS79318	T_J = 25°C			1.8		V
		0 µA< I_O < 200 mA,	2.8 V < V_I < 5.5 V	1.764		1.836	
	TPS79325	T_J = 25°C			2.5		V
		0 µA< I_O < 200 mA,	3.5 V < V_I < 5.5 V	2.45		2.55	
	TPS79328	T_J = 25°C			2.8		V
		0 µA< I_O < 200 mA,	3.8 V < V_I < 5.5 V	2.744		2.856	
	TPS793285	T_J = 25°C			2.85		V
		0 µA< I_O < 200 mA,	3.85 V < V_I < 5.5 V	2.793		2.907	
	TPS79330	T_J = 25°C			3		V
		0 µA< I_O < 200 mA,	4 V < V_I < 5.5 V	2.94		3.06	
	TPS79333	T_J = 25°C			3.3		V
		0 µA ≤ I_O < 200 mA,	4.3 V < V_I < 5.5 V	3.234		3.366	
	TPS793475	T_J = 25°C			4.75		V
		0 µA< I_O < 200 mA,	5.25 V < V_I < 5.5 V	4.655		4.845	
Quiescent current (GND current)		0 µA< I_O < 200 mA,	T_J = 25°C		170		µA
		0 µA< I_O < 200 mA				220	µA
Load regulation		0 µA< I_O < 200 mA,	T_J = 25°C		5		mV
Output voltage line regulation ($\Delta V_O/V_O$) (see Note 5)		V_O + 1 V < V_I ≤ 5.5 V,	T_J = 25°C		0.05		%/V
		V_O + 1 V < V_I ≤ 5.5 V				0.12	
Output noise voltage (TPS79328)		BW = 200 Hz to 100 kHz, I_O = 200 mA, T_J = 25°C	$C_{(byp)}$ = 0.001 µF		55		μV_{RMS}
			$C_{(byp)}$ = 0.0047 µF		36		
			$C_{(byp)}$ = 0.01 µF		33		
			$C_{(byp)}$ = 0.1 µF		32		
Time, start-up (TPS79328)		R_L = 14 Ω, C_o = 1 µF, T_J = 25°C	$C_{(byp)}$ = 0.001 µF		50		µs
			$C_{(byp)}$ = 0.0047 µF		70		
			$C_{(byp)}$ = 0.01 µF		100		
Output current limit		V_O = 0 V,	See Note 4	285		600	mA
Standby current		EN = 0 V,	2.7 V < V_I < 5.5 V		0.07	1	µA
High level enable input voltage		2.7 V < V_I < 5.5 V		2			V
Low level enable input voltage		2.7 V < V_I < 5.5 V				0.7	V
Input current (EN)		EN = 0		–1		1	µA
Input current (FB) (TPS79301)		FB = 1.8 V				1	µA

NOTES: 2. To calculate the minimum input voltage for your maximum output current, use the following formula:
$V_I(min) = V_O(max) + V_{DO}$ (max load)

3. Continuous output current and operating junction temperature are limited by internal protection circuitry, but it is not recommended that the device operate under conditions beyond those specified in this table for extended periods of time.

4. The minimum IN operating voltage is 2.7 V or $V_{O(typ)}$ + 1 V, whichever is greater. The maximum IN voltage is 5.5 V. The maximum output current is 200 mA.

5. If V_O ≤ 2.5 V then V_{Imin} = 2.7 V, V_{Imax} = 5.5 V:

$$\text{Line Reg. (mV)} = (\%/V) \times \frac{V_O\left(V_{Imax} - 2.7\ V\right)}{100} \times 1000$$

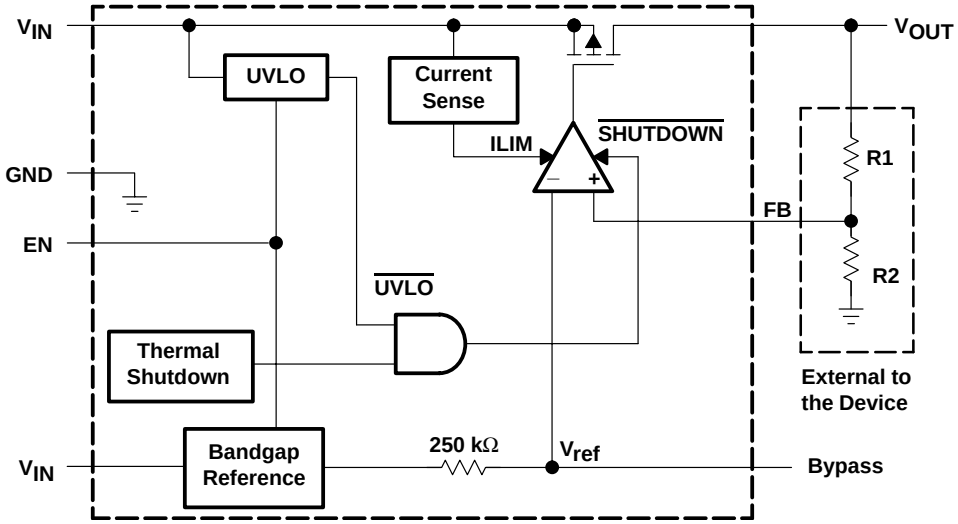
If V_O ≥ 2.5 V then $V_{Imin} = V_O$ + 1 V, V_{Imax} = 5.5 V.

electrical characteristics over recommended operating free-air temperature range EN = V_I, T_J = –40 to 125 °C, V_I = $V_{O(typ)}$ + 1 V, I_O = 1 mA, C_o = 10 μF, $C_{(byp)}$ = 0.01 μF (unless otherwise noted) (continued)

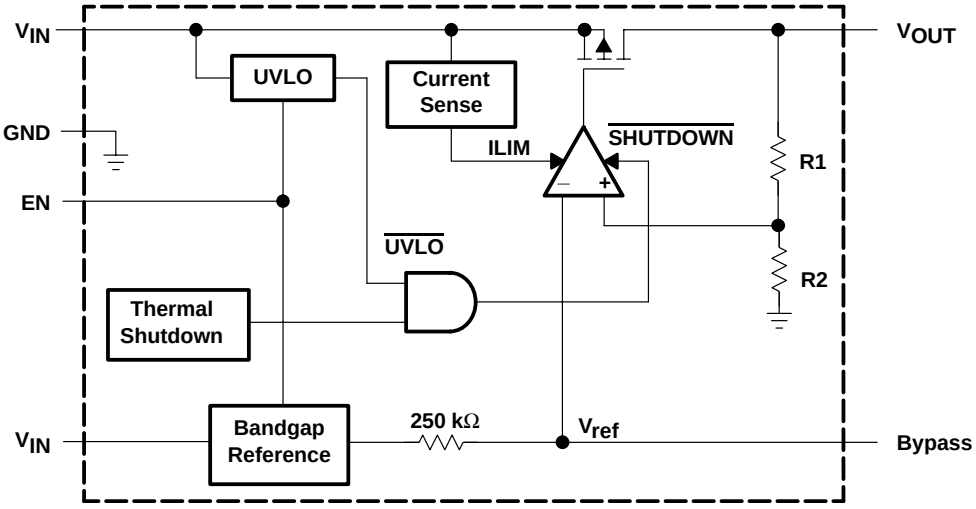
PARAMETER		TEST CONDITIONS		MIN	TYP	MAX	UNIT
Power supply ripple rejection	TPS79328	f = 100 Hz, T_J = 25°C,	I_O = 10 mA		70		dB
		f = 100 Hz, T_J = 25°C,	I_O = 200 mA		68		
		f = 10 kHz, T_J = 25°C,	I_O = 200 mA		70		
		f = 100 kHz, T_J = 25°C,	I_O = 200 mA		43		
Dropout voltage (see Note 6)	TPS79328	I_O = 200 mA,	T_J = 25°C		120		mV
		I_O = 200 mA				200	
	TPS793285	I_O = 200 mA,	T_J = 25°C		120		
		I_O = 200 mA				200	
	TPS79330	I_O = 200 mA,	T_J = 25°C		112		
		I_O = 200 mA				200	
	TPS79333	I_O = 200 mA,	T_J = 25°C		102		mV
		I_O = 200 mA				180	
	TPS793475	I_O = 200 mA,	T_J = 25°C		77		mV
		I_O = 200 mA				125	
UVLO threshold		V_{CC} rising		2.25		2.65	V
UVLO hysteresis		T_J = 25°C	V_{CC} rising		100		mV

NOTE 6: IN voltage equals V_O(typ) – 100 mV; The TPS79325 dropout voltage is limited by the input voltage range limitations.

functional block diagram—adjustable version



functional block diagram—fixed version



Terminal Functions

TERMINAL			I/O	DESCRIPTION
NAME	ADJ	FIXED		
BYPASS	4	4		An external bypass capacitor, connected to this terminal, in conjunction with an internal resistor, creates a low-pass filter to further reduce regulator noise.
EN	3	3	I	The EN terminal is an input which enables or shuts down the device. When EN goes to a logic high, the device will be enabled. When the device goes to a logic low, the device is in shutdown mode.
FB	5	N/A	I	This terminal is the feedback input voltage for the adjustable device.
GND	2	2		Regulator ground
IN	1	1	I	The IN terminal is the input to the device.
OUT	6	5	O	The OUT terminal is the regulated output of the device.

TYPICAL CHARACTERISTICS

TPS79328
OUTPUT VOLTAGE
vs
OUTPUT CURRENT

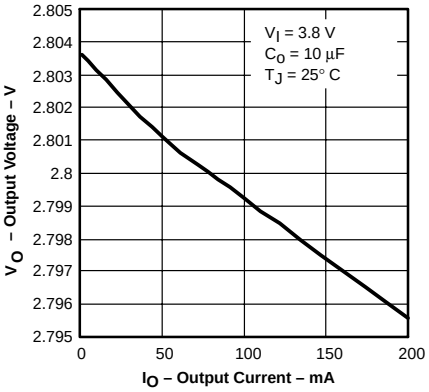


Figure 1

TPS79328
OUTPUT VOLTAGE
vs
JUNCTION TEMPERATURE

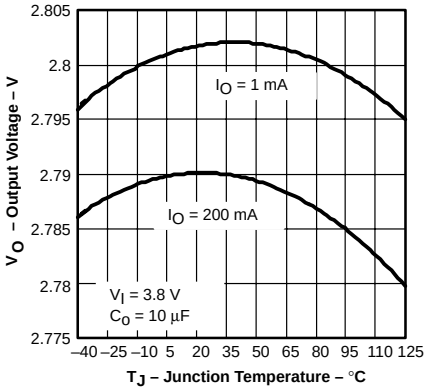


Figure 2

TPS79328
GROUND CURRENT
vs
JUNCTION TEMPERATURE

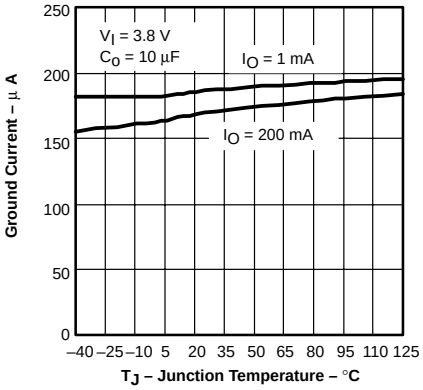


Figure 3

TPS79328
OUTPUT SPECTRAL NOISE DENSITY
vs
FREQUENCY

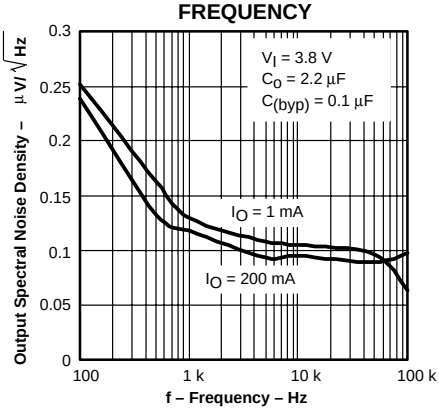


Figure 4

TPS79328
OUTPUT SPECTRAL NOISE DENSITY
vs
FREQUENCY

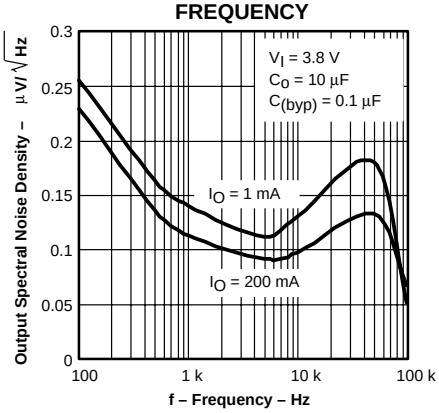


Figure 5

TPS79328
OUTPUT SPECTRAL NOISE DENSITY
vs
FREQUENCY

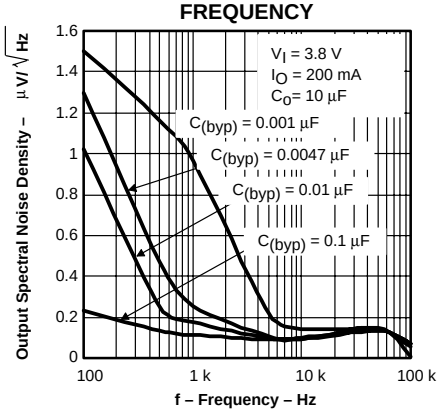


Figure 6

ROOT MEAN SQUARED OUTPUT NOISE
vs
BYPASS CAPACITANCE

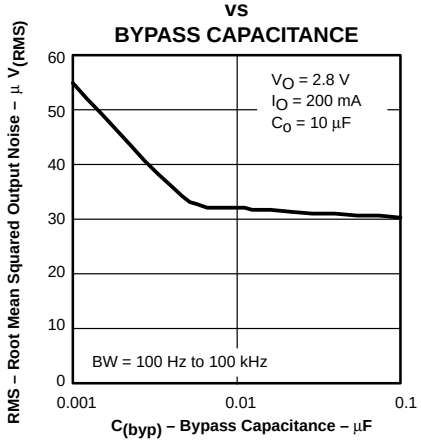


Figure 7

OUTPUT IMPEDANCE
vs
FREQUENCY

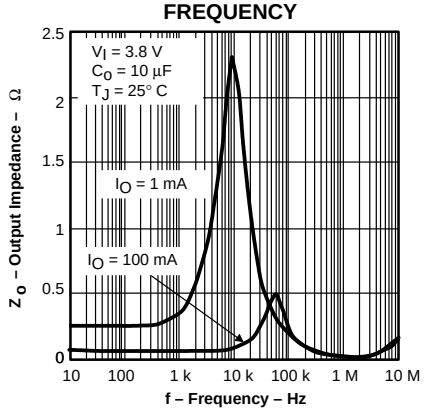


Figure 8

TPS79328
DROPOUT VOLTAGE
vs
JUNCTION TEMPERATURE

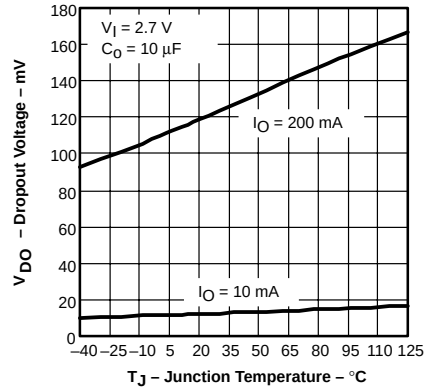


Figure 9

TYPICAL CHARACTERISTICS

TPS79328
RIPPLE REJECTION vs FREQUENCY

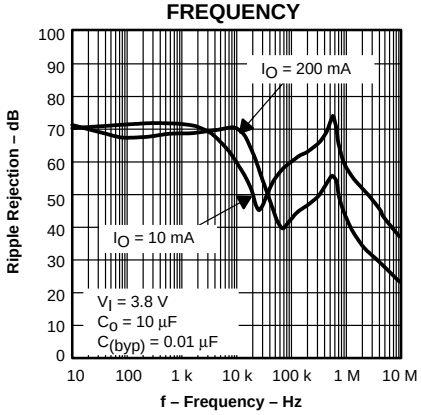


Figure 10

TPS79328
RIPPLE REJECTION vs FREQUENCY

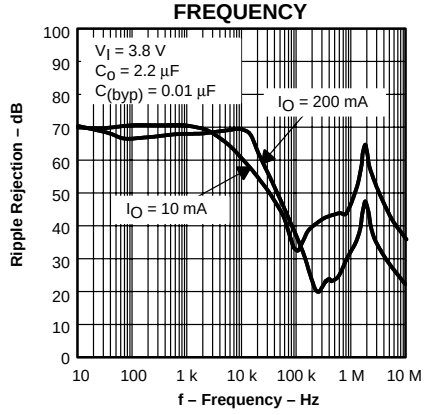


Figure 11

TPS79328
RIPPLE REJECTION vs FREQUENCY

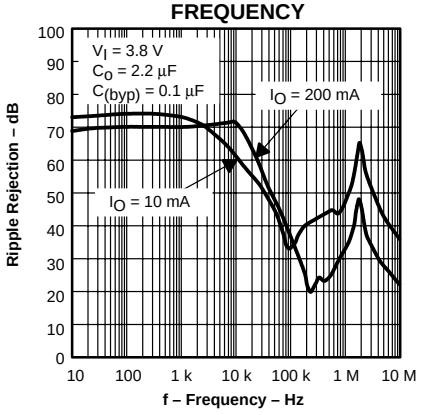


Figure 12

TPS79328
OUTPUT VOLTAGE, ENABLE VOLTAGE vs TIME (START-UP)

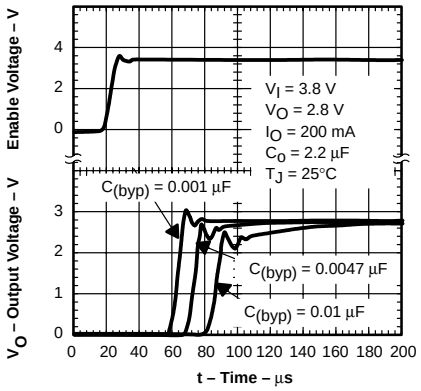


Figure 13

TPS79328
LINE TRANSIENT RESPONSE

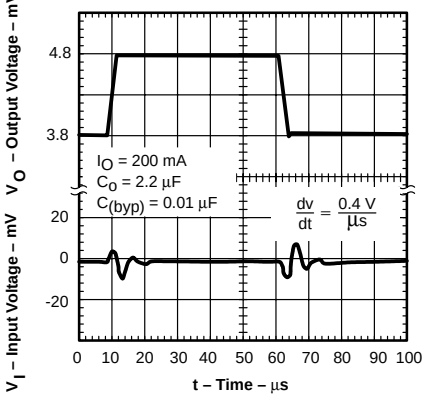


Figure 14

TPS79328
LOAD TRANSIENT RESPONSE

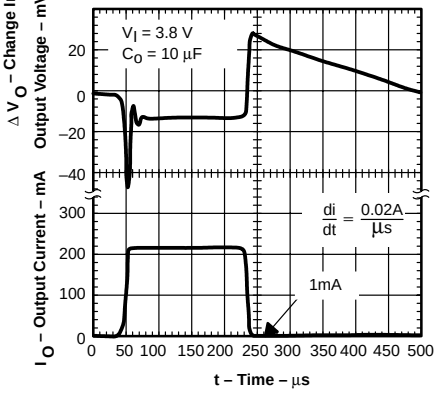


Figure 15

POWER UP / POWER DOWN

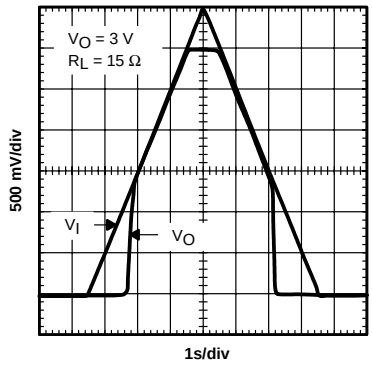


Figure 16

DC DROPOUT VOLTAGE vs OUTPUT CURRENT

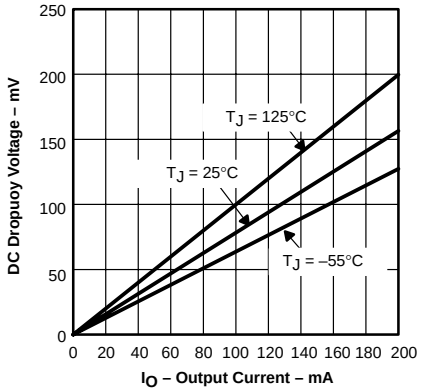


Figure 17

TPS79301
DROPOUT VOLTAGE vs INPUT VOLTAGE

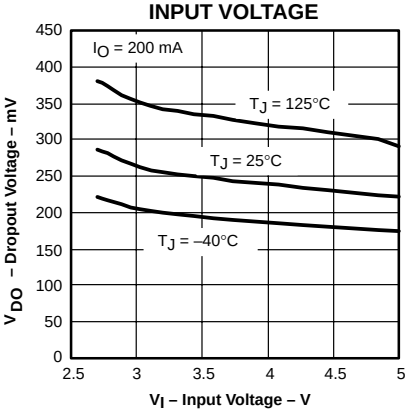


Figure 18

TYPICAL CHARACTERISTICS

MINIMUM REQUIRED INPUT VOLTAGE vs OUTPUT VOLTAGE

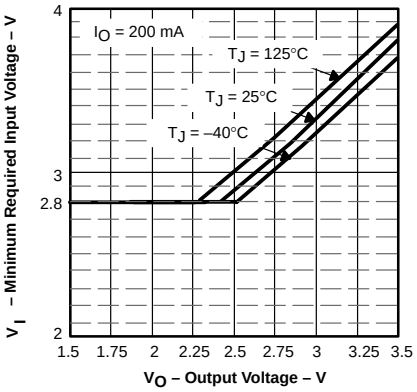


Figure 19

TYPICAL REGIONS OF STABILITY EQUIVALENT SERIES RESISTANCE (ESR) vs OUTPUT CURRENT

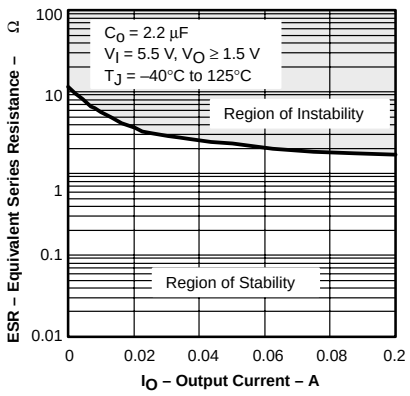


Figure 20

TYPICAL REGIONS OF STABILITY EQUIVALENT SERIES RESISTANCE (ESR) vs OUTPUT CURRENT

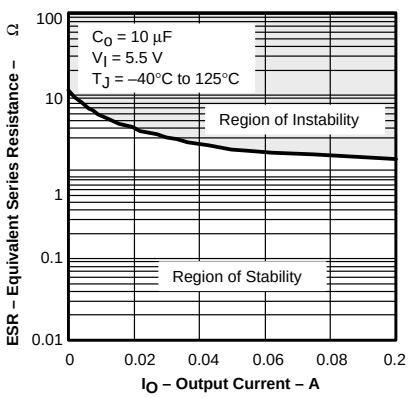


Figure 21

APPLICATION INFORMATION

The TPS793xx family of low-dropout (LDO) regulators has been optimized for use in noise-sensitive battery-operated equipment. The device features extremely low dropout voltages, high PSRR, ultralow output noise, low quiescent current (170 μA typically), and enable-input to reduce supply currents to less than 1 μA when the regulator is turned off.

A typical application circuit is shown in Figure 22.

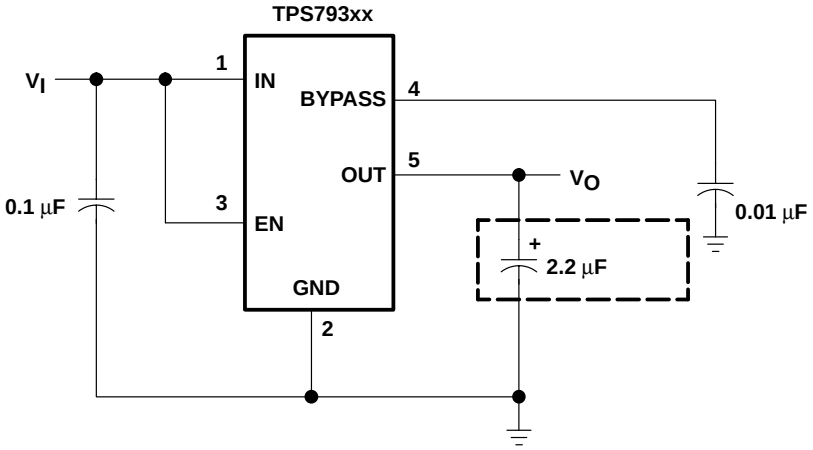


Figure 22. Typical Application Circuit

external capacitor requirements

A 0.1-μF or larger ceramic input bypass capacitor, connected between IN and GND and located close to the TPS793xx, is required for stability and will improve transient response, noise rejection, and ripple rejection. A higher-value electrolytic input capacitor may be necessary if large, fast-rise-time load transients are anticipated and the device is located several inches from the power source.

Like all low dropout regulators, the TPS793xx requires an output capacitor connected between OUT and GND to stabilize the internal control loop. The minimum recommended capacitance is 2.2 μF. Any 2.2 μF or larger ceramic capacitor is suitable, provided the capacitance does not vary significantly over temperature.

The internal voltage reference is a key source of noise in an LDO regulator. The TPS793xx has a BYPASS pin which is connected to the voltage reference through a 250-kΩ internal resistor. The 250-kΩ internal resistor, in conjunction with an external bypass capacitor connected to the BYPASS pin, creates a low pass filter to reduce the voltage reference noise and, therefore, the noise at the regulator output. In order for the regulator to operate properly, the current flow out of the BYPASS pin must be at a minimum, because any leakage current will create an IR drop across the internal resistor thus creating an output error. Therefore, the bypass capacitor must have minimal leakage current.

For example, the TPS79328 exhibits only 32 μV_{RMS} of output voltage noise using a 0.1-μF ceramic bypass capacitor and a 2.2-μF ceramic output capacitor. Note that the output starts up slower as the bypass capacitance increases due to the RC time constant at the BYPASS pin that is created by the internal 250-kΩ resistor and external capacitor.

APPLICATION INFORMATION

board layout recommendation to improve PSRR and noise performance

To improve ac measurements like PSRR, output noise, and transient response, it is recommended that the board be designed with separate ground planes for V_{IN} and V_{OUT}, with each ground plane connected only at the GND pin of the device. In addition, the ground connection for the bypass capacitor should connect directly to the GND pin of the device.

power dissipation and junction temperature

Specified regulator operation is assured to a junction temperature of 125°C; the maximum junction temperature should be restricted to 125°C under normal operating conditions. This restriction limits the power dissipation the regulator can handle in any given application. To ensure the junction temperature is within acceptable limits, calculate the maximum allowable dissipation, $P_{D(max)}$, and the actual dissipation, P_D, which must be less than or equal to $P_{D(max)}$.

The maximum-power-dissipation limit is determined using the following equation:

$$P_{D(max)} = \frac{T_J max - T_A}{R_{\theta JA}} \quad (1)$$

Where:

$T_J max$ is the maximum allowable junction temperature.

$R_{\theta JA}$ is the thermal resistance junction-to-ambient for the package, see the dissipation rating table.

T_A is the ambient temperature.

The regulator dissipation is calculated using:

$$P_D = \left(V_I - V_O\right) \times I_O \quad (2)$$

Power dissipation resulting from quiescent current is negligible. Excessive power dissipation triggers the thermal protection circuit.

programming the TPS79301 adjustable LDO regulator

The output voltage of the TPS79301 adjustable regulator is programmed using an external resistor divider as shown in Figure 23. The output voltage is calculated using:

$$V_O = V_{ref} \times \left(1 + \frac{R1}{R2}\right) \quad (3)$$

Where:

V_{ref} = 1.2246 V typ (the internal reference voltage)

APPLICATION INFORMATION

programming the TPS79301 adjustable LDO regulator (continued)

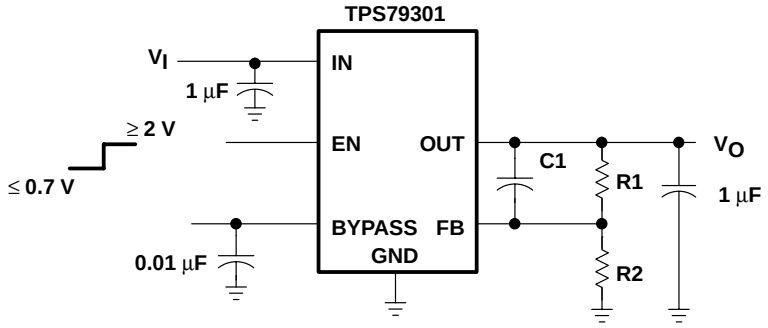
Resistors R1 and R2 should be chosen for approximately 50-μA divider current. Lower value resistors can be used for improved noise performance, but the solution consumes more power. Higher resistor values should be avoided as leakage current into/out of FB across R1/R2 creates an offset voltage that artificially increases/decreases the feedback voltage and thus erroneously decreases/increases V_O. The recommended design procedure is to choose R2 = 30.1 kΩ to set the divider current at 50 μA, C1 = 15 pF for stability, and then calculate R1 using:

$$R1 = \left(\frac{V_O}{V_{ref}} - 1\right) \times R2 \quad (4)$$

In order to improve the stability of the adjustable version, it is suggested that a small compensation capacitor be placed between OUT and FB. For voltages <1.8 V, the value of this capacitor should be 100 pF. For voltages >1.8 V, the approximate value of this capacitor can be calculated as:

$$C1 = \frac{(3 \times 10^{-7}) \times (R1 + R2)}{(R1 \times R2)} \quad (5)$$

The suggested value of this capacitor for several resistor ratios is shown in the table below. If this capacitor is not used (such as in a unity-gain configuration) or if an output voltage <1.8 V is chosen, then the minimum recommended output capacitor is 4.7 μF instead of 2.2 μF.



OUTPUT VOLTAGE PROGRAMMING GUIDE

OUTPUT VOLTAGE	R1	R2	C1
2.5 V	33.4 kΩ	30.1 kΩ	22 pF
3.3 V	53.6 kΩ	30.1 kΩ	15 pF
3.6 V	59 kΩ	30.1 kΩ	15 pF

Figure 23. TPS79301 Adjustable LDO Regulator Programming

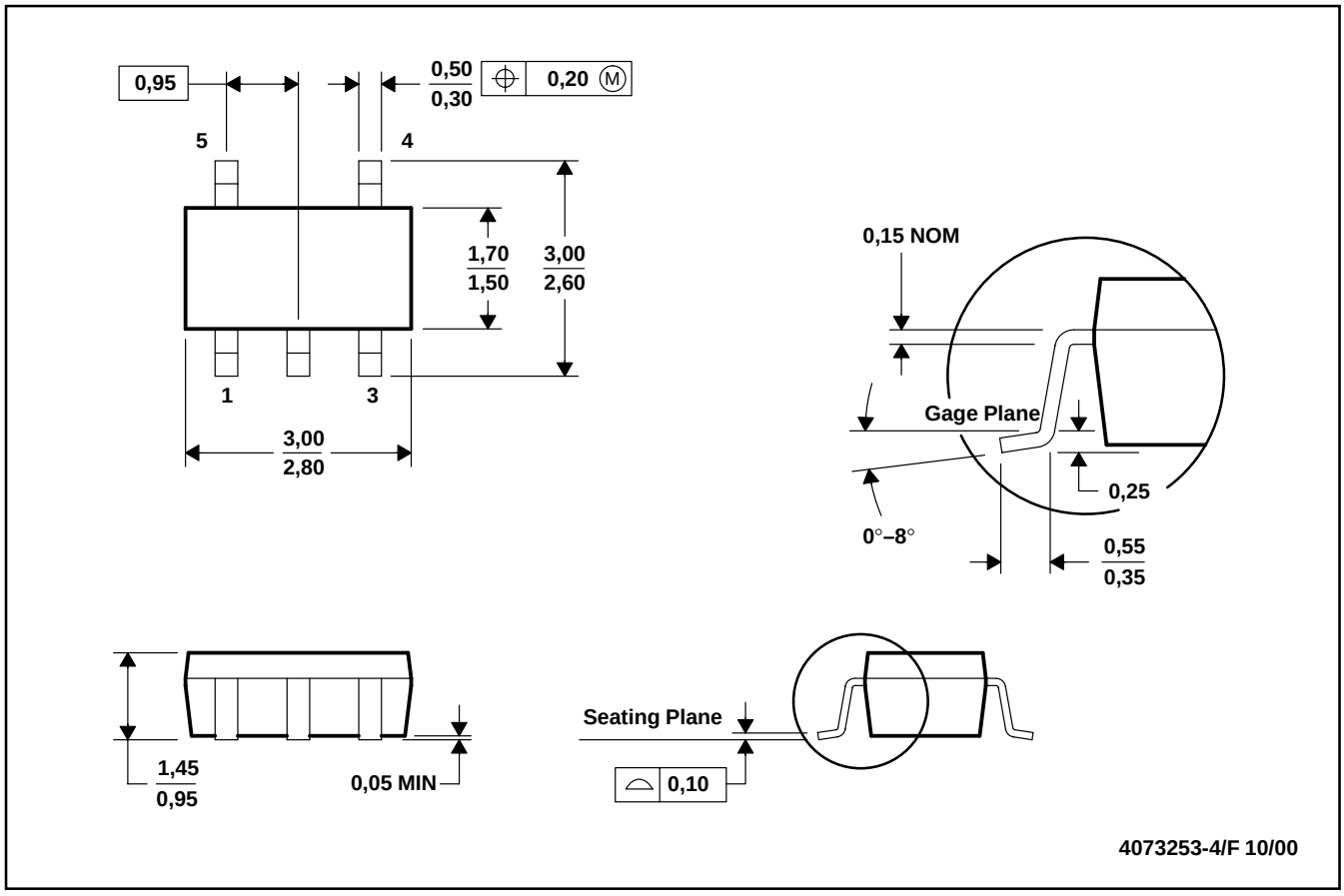
regulator protection

The TPS793xx PMOS-pass transistor has a built-in back diode that conducts reverse current when the input voltage drops below the output voltage (e.g., during power down). Current is conducted from the output to the input and is not internally limited. If extended reverse voltage operation is anticipated, external limiting might be appropriate.

The TPS793xx features internal current limiting and thermal protection. During normal operation, the TPS793xx limits output current to approximately 400 mA. When current limiting engages, the output voltage scales back linearly until the overcurrent condition ends. While current limiting is designed to prevent gross device failure, care should be taken not to exceed the power dissipation ratings of the package or the absolute maximum voltage ratings of the device. If the temperature of the device exceeds approximately 165°C, thermal-protection circuitry shuts it down. Once the device has cooled down to below approximately 140°C, regulator operation resumes.

MECHANICAL DATA

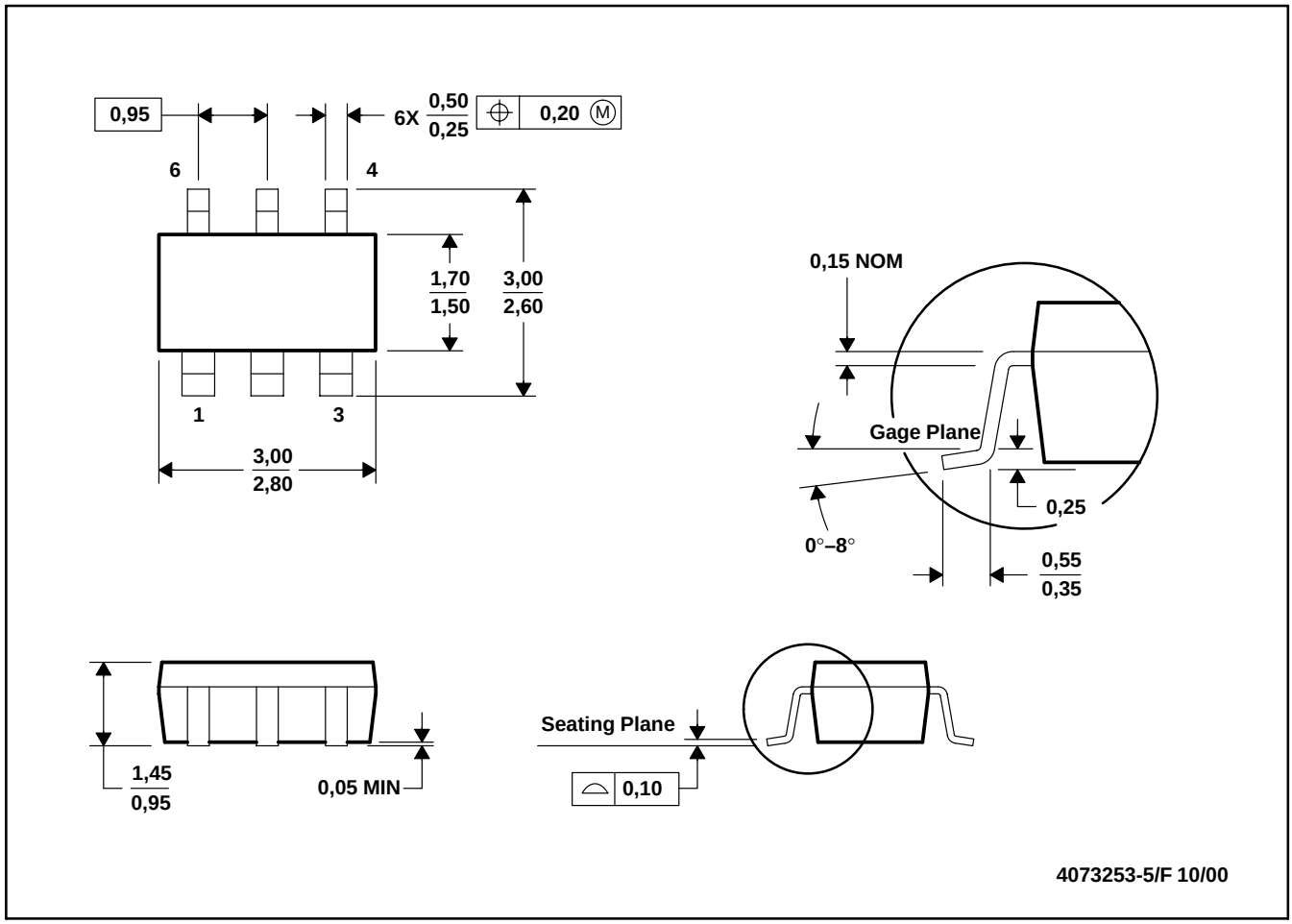
DBV (R-PDSO-G5) **PLASTIC SMALL-OUTLINE**



NOTES: A. All linear dimensions are in millimeters.
B. This drawing is subject to change without notice.
C. Body dimensions do not include mold flash or protrusion.
D. Falls within JEDEC MO-178

MECHANICAL DATA

DBV (R-PDSO-G6) **PLASTIC SMALL-OUTLINE**



NOTES: A. All linear dimensions are in millimeters.
B. This drawing is subject to change without notice.
C. Body dimensions do not include mold flash or protrusion.
D. Leads 1, 2, 3 are wider than leads 4, 5, 6 for package orientation.
E. Pin 1 is located below the first letter of the top side symbolization.

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

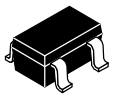
TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third–party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Mailing Address:

Texas Instruments
Post Office Box 655303
Dallas, Texas 75265



HIGH-EFFICIENCY, SOT23 STEP-DOWN, DC-DC CONVERTER

FEATURES

- **High Efficiency Synchronous Step-Down Converter With up to 95% Efficiency**
- **2.5 V to 6.0 V Input Voltage Range**
- **Adjustable Output Voltage Range From 0.7 V to V_I**
- **Fixed Output Voltage Options Available**
- **Up to 300 mA Output Current**
- **1 MHz Fixed Frequency PWM Operation**
- **Highest Efficiency Over Wide Load Current Range Due to Power Save Mode**
- **15-µA Typical Quiescent Current**
- **Soft Start**
- **100% Duty Cycle Low-Dropout Operation**
- **Dynamic Output-Voltage Positioning**
- **Available in a Tiny 5-Pin SOT23 Package**

APPLICATIONS

- **PDAs and Pocket PC**
- **Cellular Phones, Smart Phones**
- **Low Power DSP Supply**
- **Digital Cameras**
- **Portable Media Players**
- **Portable Equipment**

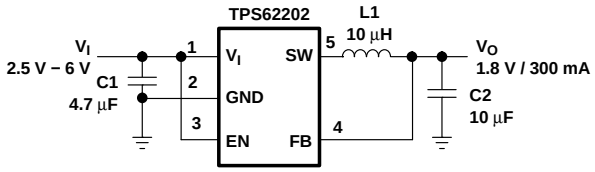
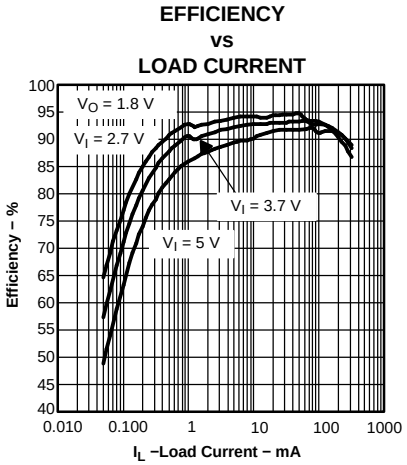


Figure 1. Typical Application (Fixed Output Voltage Version)

DESCRIPTION

The TPS6220x devices are a family of high-efficiency synchronous step-down converters ideally suited for portable systems powered by 1-cell Li-Ion or 3-cell NiMH/NiCd batteries. The devices are also suitable to operate from a standard 3.3-V or 5-V voltage rail.

With an output voltage range of 6.0 V down to 0.7 V and up to 300 mA output current, the devices are ideal to power low voltage DSPs and processors used in PDAs, pocket PCs, and smart phones. Under nominal load current, the devices operate with a fixed switching frequency of typically 1 MHz. At light load currents, the part enters the power save mode operation; the switching frequency is reduced and the quiescent current is typically only 15 µA; therefore it achieves the highest efficiency over the entire load current range. The TPS6220x needs only three small external components. Together with the tiny SOT23 package, a minimum system solution size can be achieved. An advanced fast response voltage mode control scheme achieves superior line and load regulation with small ceramic input and output capacitors.



This integrated circuit can be damaged by ESD. Texas Instruments recommends that all integrated circuits be handled with appropriate precautions. Failure to observe proper handling and installation procedures can cause damage.

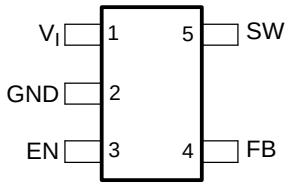
ESD damage can range from subtle performance degradation to complete device failure. Precision integrated circuits may be more susceptible to damage because very small parametric changes could cause the device not to meet its published specifications.

ORDERING INFORMATION (1)

T_A	OUTPUT VOLTAGE	SOT23 PACKAGE	SYMBOL
-40°C to 85°C	Adjustable	TPS62200DBV	PHKI
	1.2 V	TPS62207DBV	PJGI
	1.5 V	TPS62201DBV	PHLI
	1.6 V	TPS62204DBV	PHSI
	1.8 V	TPS62202DBV	PHMI
	1.875 V	TPS62208DBV	ALW
	2.5 V	TPS62205DBV	PHTI
	3.3 V	TPS62203DBV	PHNI

(1) The DBV package is available in tape and reel. Add R suffix (DBVR) to order quantities of 3000 parts. Add T suffix (DBVT) to order quantities of 250 parts

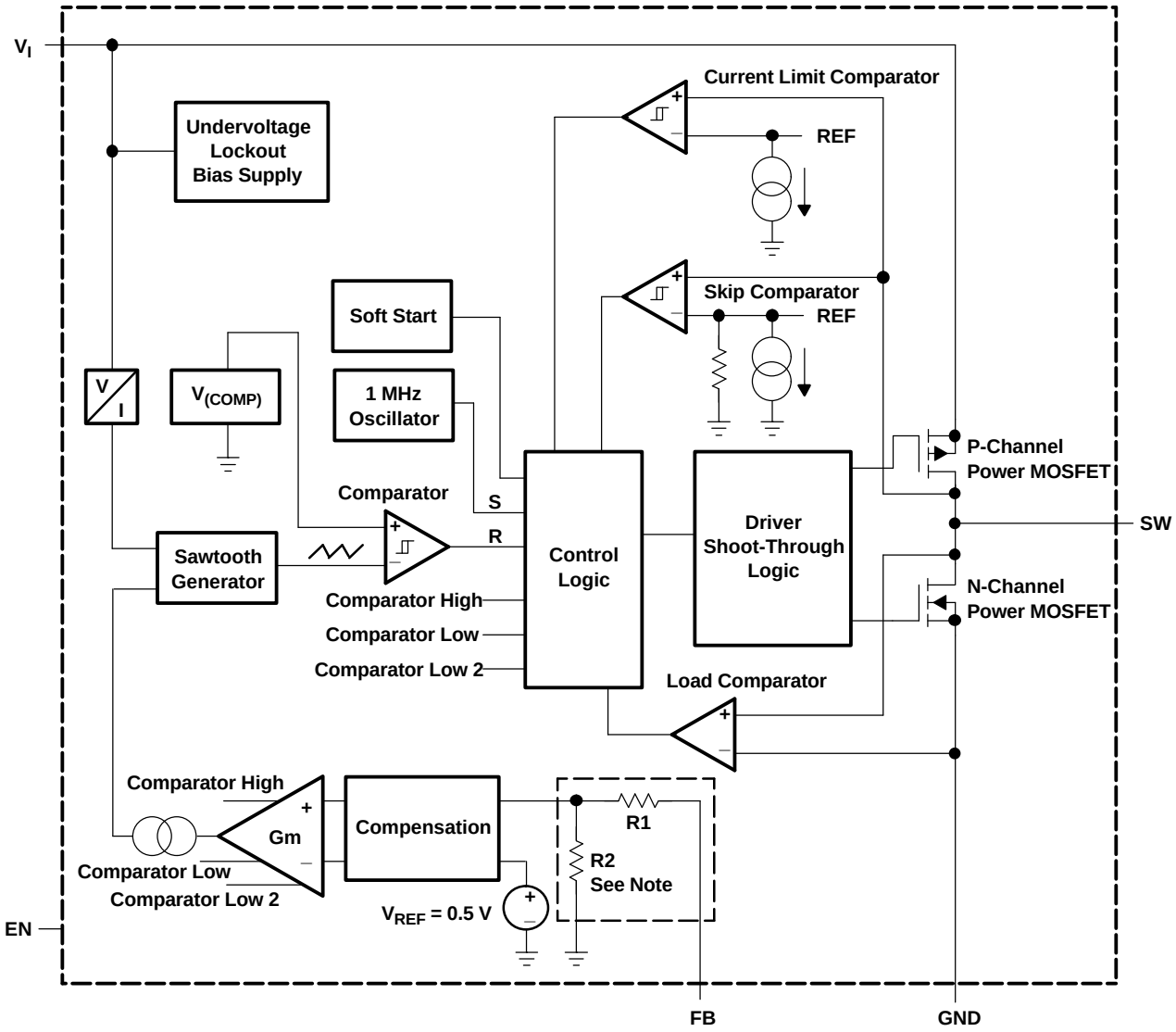
DBV PACKAGE
(TOP VIEW)



Terminal Functions

TERMINAL NAME	TERMINAL NO.	I/O	DESCRIPTION
EN	3	I	This is the enable pin of the device. Pulling this pin to ground forces the device into shutdown mode. Pulling this pin to Vin enables the device. This pin must not be left floating and must be terminated.
FB	4	I	This is the feedback pin of the device. Connect this pin directly to the output if the fixed output voltage version is used. For the adjustable version an external resistor divider is connected to this pin. The internal voltage divider is disabled for the adjustable version.
GND	2		Ground
SW	5	I/O	Connect the inductor to this pin. This pin is the switch pin and is connected to the internal MOSFET switches.
V_I	1	I	Supply voltage pin

FUNCTIONAL BLOCK DIAGRAM



#IMPLIED. For the adjustable version (TPS62200) the internal feedback divider is disabled and the FB pin is directly connected to the internal GM amplifier

DETAILED DESCRIPTION

OPERATION

The TPS6220x is a synchronous step-down converter operating with typically 1MHz fixed frequency pulse width modulation (PWM) at moderate to heavy load currents and in power save mode operating with pulse frequency modulation (PFM) at light load currents.

During PWM operation the converter uses a unique fast response, voltage mode, controller scheme with input voltage feed forward. This achieves good line and load regulation and allows the use of small ceramic input and output capacitors. At the beginning of each clock cycle initiated by the clock signal (S), the P-channel MOSFET switch is turned on, and the inductor current ramps up until the comparator trips and the control logic turns off the switch. The current limit comparator also turns off the switch in case the current limit of the P-channel switch is exceeded. Then the N-channel rectifier switch is turned on and the inductor current ramps down. The next cycle is initiated by the clock signal again turning off the N-channel rectifier and turning on the P-channel switch.

DETAILED DESCRIPTION (continued)

The GM amplifier and input voltage determines the rise time of the Sawtooth generator; therefore any change in input voltage or output voltage directly controls the duty cycle of the converter. This gives a very good line and load transient regulation.

POWER SAVE MODE OPERATION

As the load current decreases, the converter enters the power save mode operation. During power save mode, the converter operates with reduced switching frequency in PFM mode and with a minimum quiescent current to maintain high efficiency.

Two conditions allow the converter to enter the power save mode operation. One is when the converter detects the discontinuous conduction mode. The other is when the peak switch current in the P-channel switch goes below the skip current limit. The typical skip current limit can be calculated as

$$I_{skip} \leq 66\ mA + \frac{Vin}{160\ \Omega}$$

During the power save mode the output voltage is monitored with the comparator by the thresholds comp low and comp high. As the output voltage falls below the comp low threshold set to typically 0.8% above Vout nominal, the P-channel switch turns on. The P-channel switch is turned off as the peak switch current is reached. The typical peak switch current can be calculated:

$$I_{peak} = 66\ mA + \frac{Vin}{80\ \Omega}$$

The N-channel rectifier is turned on and the inductor current ramps down. As the inductor current approaches zero the N-channel rectifier is turned off and the P-channel switch is turned on again, starting the next pulse. The converter continues these pulses until the comp high threshold (set to typically 1.6% above Vout nominal) is reached. The converter enters a sleep mode, reducing the quiescent current to a minimum. The converter wakes up again as the output voltage falls below the comp low threshold again. This control method reduces the quiescent current typically to 15 µA and reduces the switching frequency to a minimum, thereby achieving the high converter efficiency. Setting the skip current thresholds to typically 0.8% and 1.6% above the nominal output voltage at light load current results in a dynamic output voltage achieving lower absolute voltage drops during heavy load transient changes. This allows the converter to operate with a small output capacitor of just 10 µF and still have a low absolute voltage drop during heavy load transient changes. Refer to Figure 2 for detailed operation of the power save mode.

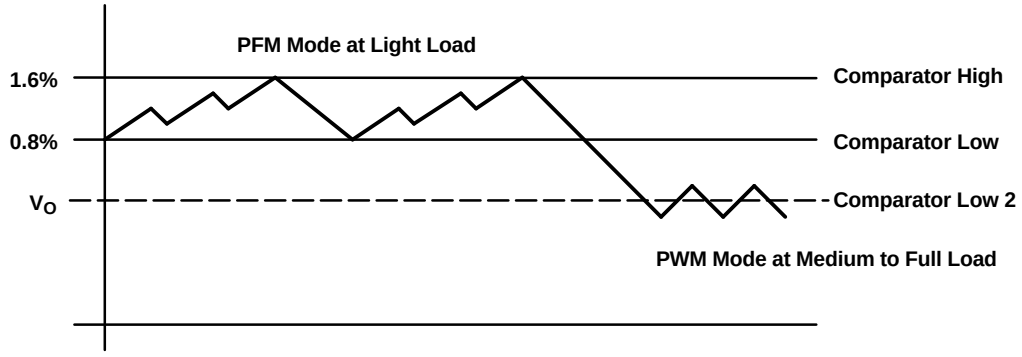


Figure 2. Power Save Mode Thresholds and Dynamic Voltage Positioning

The converter enters the fixed frequency PWM mode again as soon as the output voltage falls below the comp low 2 threshold.

DETAILED DESCRIPTION (continued)

DYNAMIC VOLTAGE POSITIONING

As described in the power save mode operation sections and as detailed in Figure 2, the output voltage is typically 0.8% above the nominal output voltage at light load currents, as the device is in power save mode. This gives additional headroom for the voltage drop during a load transient from light load to full load. During a load transient from full load to light load, the voltage overshoot is also minimized due to active regulation turning on the N-channel rectifier switch.

SOFT START

The TPS6220x has an internal soft start circuit that limits the inrush current during start-up. This prevents possible voltage drops of the input voltage in case a battery or a high impedance power source is connected to the input of the TPS6220x.

The soft start is implemented as a digital circuit increasing the switch current in steps of typically 60 mA,120 mA, 240 mA and then the typical switch current limit of 480 mA. Therefore the start-up time mainly depends on the output capacitor and load current. Typical start-up time with 10 µF output capacitor and 200 mA load current is 800 µs.

LOW DROPOUT OPERATION 100% DUTY CYCLE

The TPS6220x offers a low input to output voltage difference, while still maintaining operation with the 100% duty cycle mode. In this mode, the P-channel switch is constantly turned on. This is particularly useful in battery powered applications to achieve longest operation time by taking full advantage of the whole battery voltage range. The minimum input voltage to maintain regulation, depending on the load current and output voltage, can be calculated as

$$Vin_{min} = Vout_{max} + Iout_{max} \times \left(r_{ds}(ON)_{max} + R_L \right)$$

$Iout_{max}$ = maximum output current plus inductor ripple current
$r_{ds}(ON)_{max}$ = maximum P-channel switch $r_{ds}(ON)$
R_L = DC resistance of the inductor
$Vout_{max}$ = nominal output voltage plus maximum output voltage tolerance

ENABLE

Pulling the enable low forces the part into shutdown, with a shutdown quiescent current of typically 0.1 µA. In this mode, the P-channel switch and N-channel rectifier are turned off, the internal resistor feedback divider is disconnected, and the whole device is in shutdown mode. If an output voltage, which could be an external voltage source or super cap, is present during shutdown, the reverse leakage current is specified under electrical characteristics. For proper operation the enable pin must be terminated and must not be left floating.

Pulling the enable high starts up the TPS6220x with the soft start as previously described.

UNDERVOLTAGE LOCKOUT

The undervoltage lockout circuit prevents the device from misoperation at low input voltages. It prevents the converter from turning on the switch or rectifier MOSFET under undefined conditions.

ABSOLUTE MAXIMUM RATINGS

over operating free-air temperature (unless otherwise noted) (1)

	UNIT
Supply voltages, V_I (2)	-0.3 V to 7.0 V
Voltages on pins SW, EN, FB (2)	-0.3 V to V_{CC} +0.3 V
Continuous power dissipation, P_D	See Dissipation Rating Table
Operating junction temperature range, T_J	-40°C to 150°C
Storage temperature, T_{stg}	-65°C to 150°C
Lead temperature (soldering, 10 sec)	260°C

(1) Stresses beyond those listed under "absolute maximum ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated under "recommended operating conditions" is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.
(2) All voltage values are with respect to network ground terminal.

DISSIPATION RATING TABLE

PACKAGE	$R_{\theta JA}$	$T_A \le 25°C$ POWER RATING	$T_A = 70°C$ POWER RATING	$T_A = 85°C$ POWER RATING
DBV	250°/W	400 mW	220 mW	160 mW

RECOMMENDED OPERATING CONDITIONS

	MIN	NOM	MAX	UNIT
Supply voltage, V_I	2.5		6.0	V
Output voltage range for adjustable output voltage version, V_O	0.7		V_I	V
Output current, I_O			300	mA
Inductor, L (1)	4.7	10		µH
Input capacitor, C_I (1)		4.7		µF
Output capacitor, C_O (1)		10		µF
Operating ambient temperature, T_A	40		85	°C
Operating junction temperature, T_J	40		125	°C

(1) See the application section for further information.

ELECTRICAL CHARACTERISTICS

V_I = 3.6 V, V_O = 1.8 V, I_O = 200 mA, EN = VIN, T_A = -40°C to 85°C, typical values are at T_A = 25°C (unless otherwise noted)

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
SUPPLY CURRENT						
V_I	Input voltage range		2.5		6.0	V
I_Q	Operating quiescent current	I_O = 0 mA, Device is not switching		15	30	µA
	Shutdown supply current	EN = GND		0.1	1	µA
	Undervoltage lockout threshold		1.5		2.0	V
ENABLE						
$V_{(EN)}$	EN high level input voltage		1.3			V
	EN low level input voltage				0.4	V
$I_{(EN)}$	EN input bias current	EN = GND or VIN		0.01	0.1	µA
POWER SWITCH						
r_{ds}(ON)	P-channel MOSFET on-resistance	$V_{IN} = V_{GS}$ = 3.6 V		530	690	mΩ
		$V_{IN} = V_{GS}$ = 2.5 V		670	850	
	N-channel MOSFET on-resistance	$V_{IN} = V_{GS}$ = 3.6 V		430	540	mΩ
		$V_{IN} = V_{GS}$ = 2.5 V		530	660	

ELECTRICAL CHARACTERISTICS (continued)

V_I = 3.6 V, V_O = 1.8 V, I_O = 200 mA, EN = VIN, T_A = -40°C to 85°C, typical values are at T_A = 25°C (unless otherwise noted)

PARAMETER			TEST CONDITIONS	MIN	TYP	MAX	UNIT
SUPPLY CURRENT							
$I_{lkg\_(P)}$	P-channel leakage current		V_{DS} = 6.0 V		0.1	1	µA
$I_{lkg\_(N)}$	N-channel leakage current		V_{DS} = 6.0 V		0.1	1	µA
$I_{(LIM)}$	P-channel current limit		2.5 V < Vin < 6.0 V	380	480	670	mA
OSCILLATOR							
f_S	Switching frequency			650	1000	1500	kHz
OUTPUT							
V_O	Adjustable output voltage range	TPS62200		0.7		V_{IN}	V
V_{ref}	Reference voltage				0.5		V
	Feedback voltage (1)	TPS62200	V_I = 3.6 V to 6.0 V, I_O = 0 mA	0%		3%	
		Adjustable	V_I = 3.6 V to 6.0 V, 0 mA≤ I_O ≤ 300 mA	-3%		3%	
V_O	Fixed output voltage(1)	TPS62207	V_I = 2.5 V to 6.0 V, I_O = 0 mA	0%		3%	
		1.2 V	V_I = 2.5 V to 6.0 V, 0 mA≤ I_O ≤ 300 mA	0%		3%	
		TPS62201	V_I = 2.5 V to 6.0 V, I_O = 0 mA	0%		3%	
		1.5 V	V_I = 2.5 V to 6.0 V, 0 mA≤ I_O ≤ 300 mA	-3%		3%	
		TPS62204	V_I = 2.5 V to 6.0 V, I_O = 0 mA	0%		3%	
		1.6 V	V_I = 2.5 V to 6.0 V, 0 mA≤ I_O ≤ 300 mA	-3%		3%	
		TPS62202	V_I = 2.5 V to 6.0 V, I_O = 0 mA	0%		3%	
		1.8 V	V_I = 2.5 V to 6.0 V, 0 mA≤ I_O ≤ 300 mA	-3%		3%	
		TPS62208	V_I = 2.5 V to 6.0 V, I_O = 0 mA	0%		3%	
		1.875 V	V_I = 2.5 V to 6.0 V, 0 mA≤ I_O ≤ 300 mA	-3%		3%	
		TPS62205	V_I = 2.7 V to 6.0 V, I_O = 0 mA	0%		3%	
		2.5 V	V_I = 2.7 V to 6.0 V, 0 mA≤ I_O ≤ 300 mA	-3%		3%	
		TPS62203	V_I = 3.6 V to 6.0 V, I_O = 0 mA	0%		3%	
		3.3 V	V_I = 3.6 V to 6.0 V, 0 mA≤ I_O ≤ 300 mA	-3%		3%	
	Line regulation		V_I = 2.5 V to 6.0 V, I_O = 10 mA		0.26		%/V
	Load regulation		I_O = 100 mA to 300 mA		0.0014		%/mA
I_{lkg}	Leakage current into SW pin		Vin > Vout, 0 V ≤ Vsw≤ Vin		0.1	1	µA
I_{lkg}(Rev)	Reverse leakage current into pin SW		Vin = open, EN=GND, V_{SW} = 6.0 V		0.1	1	µA

(1) For output voltages ≤ 1.2 V a 22 µF output capacitor value is required to achieve a maximum output voltage accuracy of 3% while operating in power save mode (PFM mode)

TYPICAL CHARACTERISTICS

Table of Graphs

			FIGURES
η	Efficiency	vs Load current	3,4,5
		vs Input voltage	6
I_Q	No load quiescent current	vs Input voltage	7
f_s	Switching frequency	vs Temperature	8
V_o	Output voltage	vs Output current	9
r_{ds}(on)	r_{ds}(on) - P-channel switch,	vs Input voltage	10
	r_{ds}(on) - N-Channel rectifier switch	vs Input voltage	11
	Line transient response		12
	Load transient response		13
	Power save mode operation		14
	Start-up		15

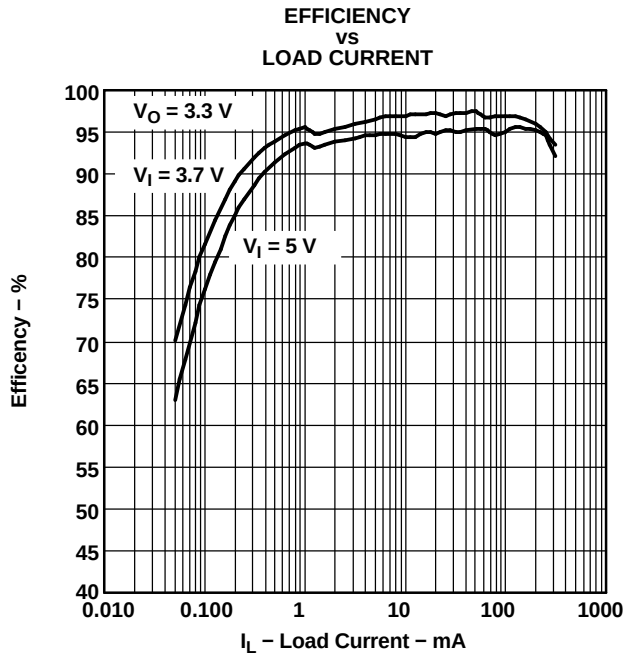


Figure 3.

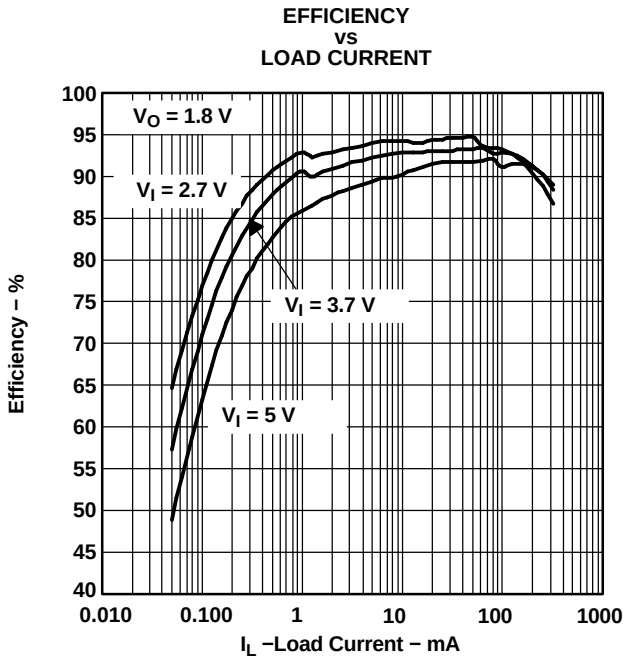


Figure 4.

TYPICAL CHARACTERISTICS (continued)

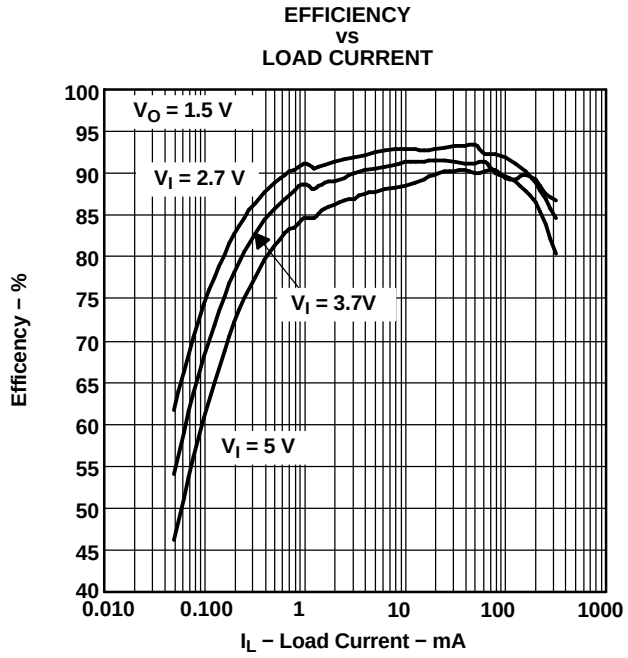


Figure 5.

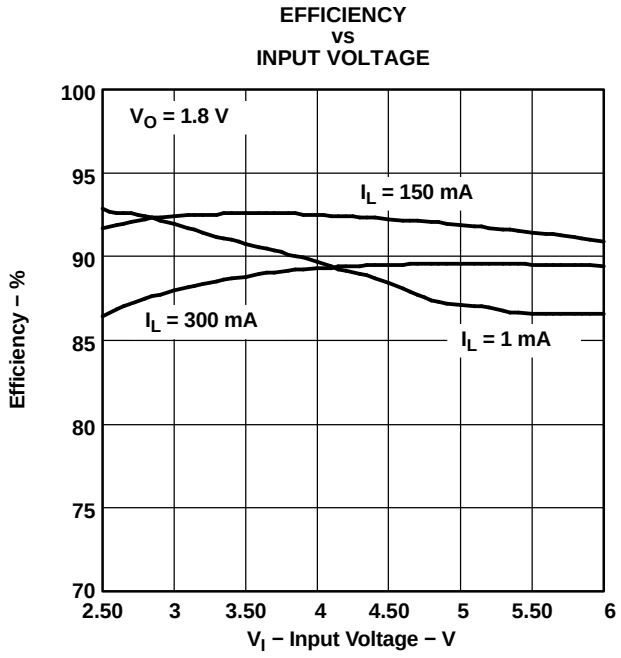


Figure 6.

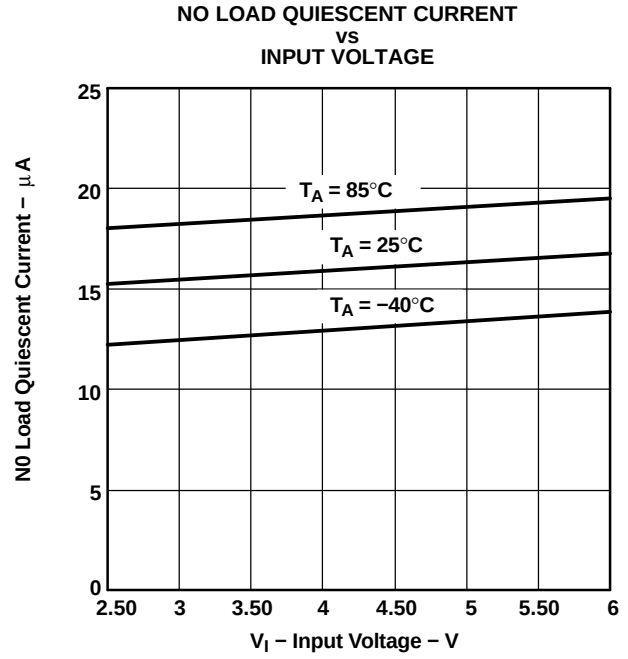


Figure 7.

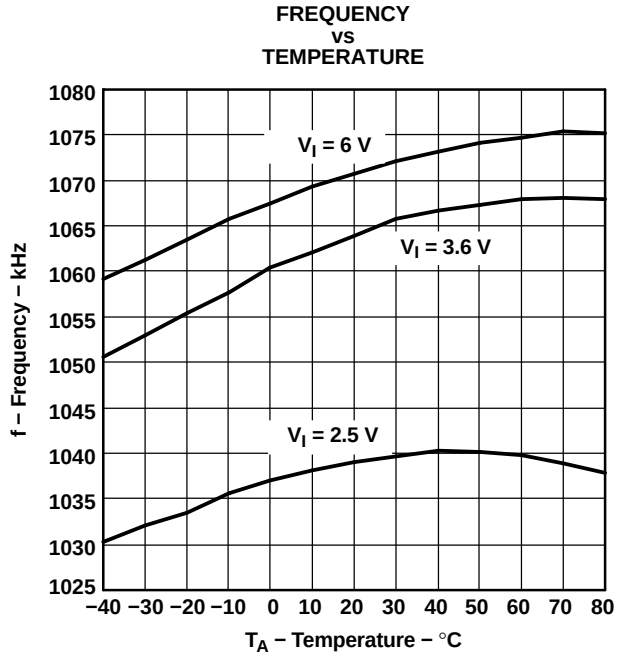


Figure 8.

TYPICAL CHARACTERISTICS (continued)

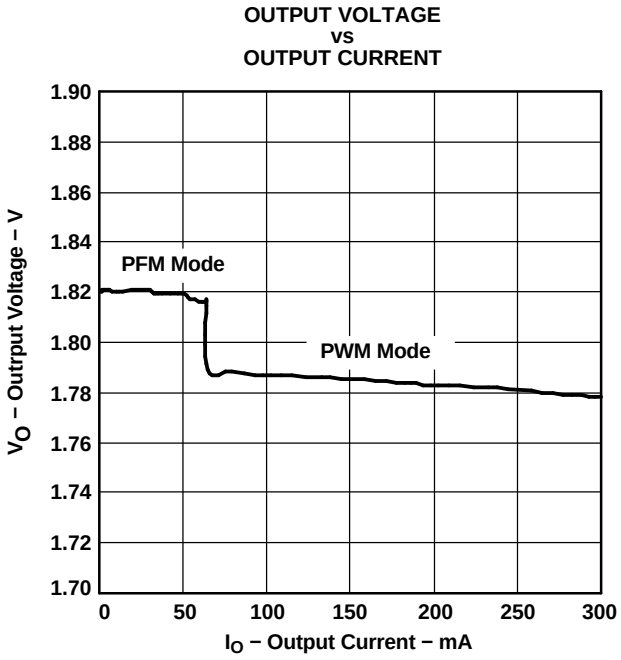


Figure 9.

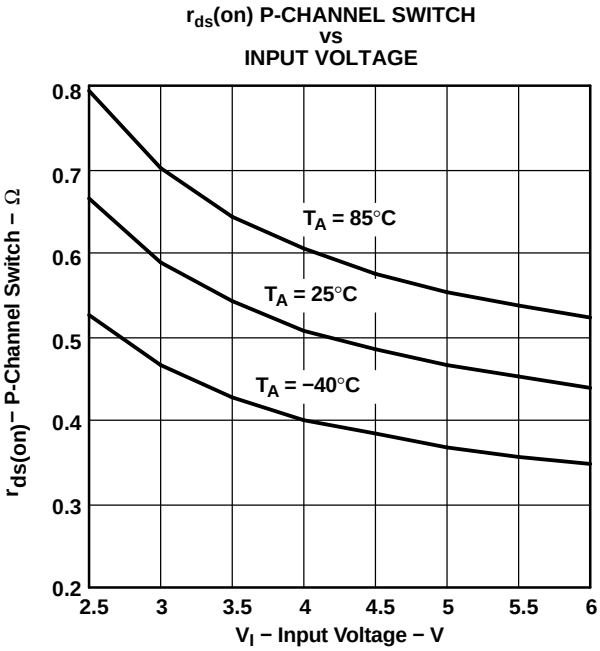


Figure 10.

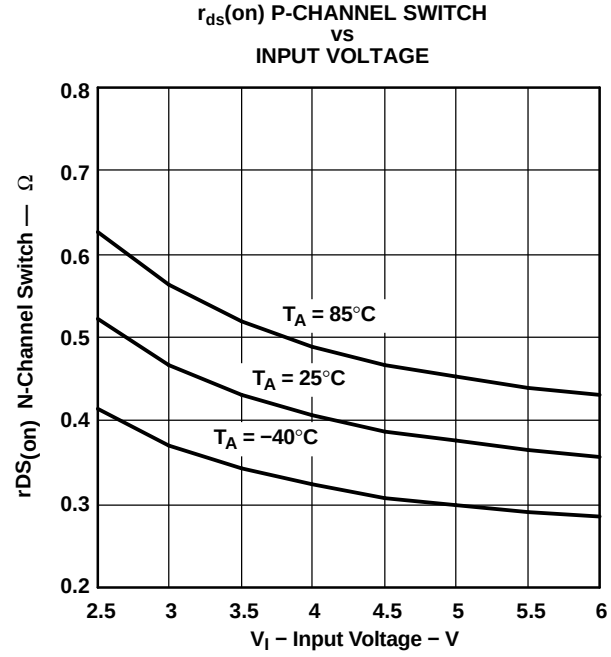


Figure 11.

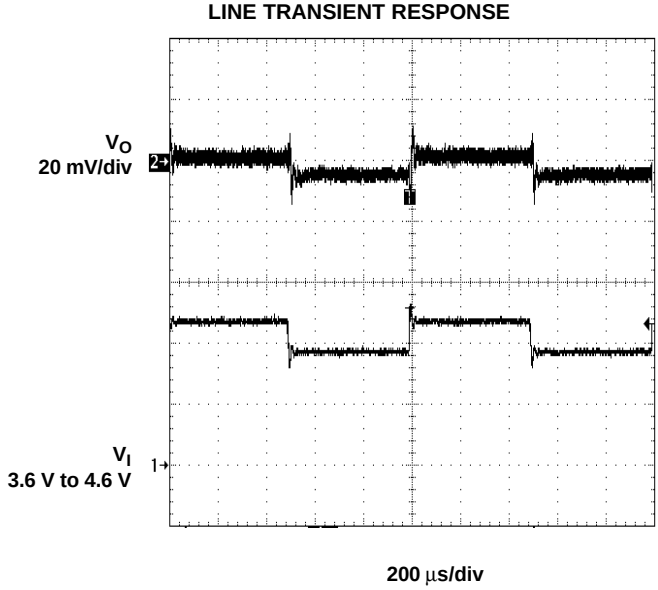


Figure 12.

TYPICAL CHARACTERISTICS (continued)

LOAD TRANSIENT RESPONSE

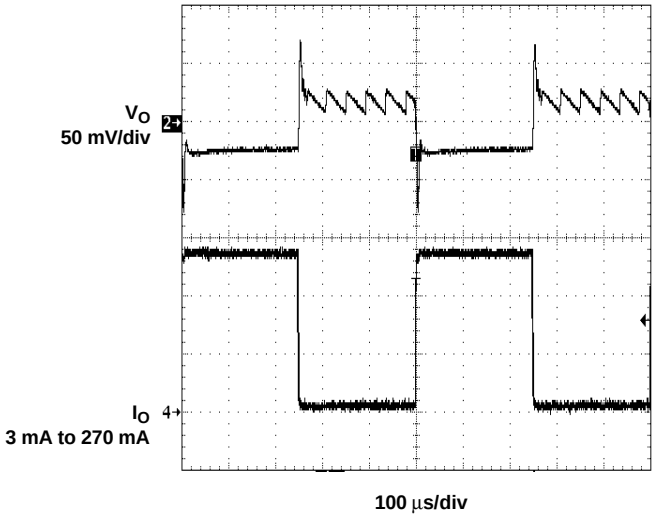


Figure 13.

POWER SAVE MODE OPERATION

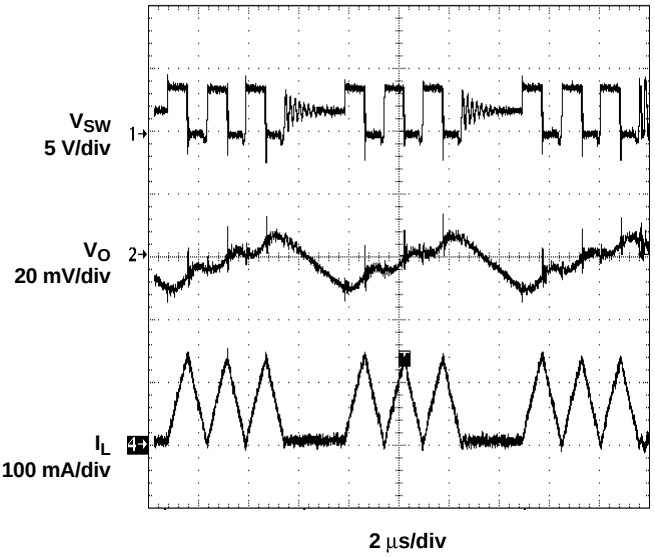


Figure 14.

START-UP

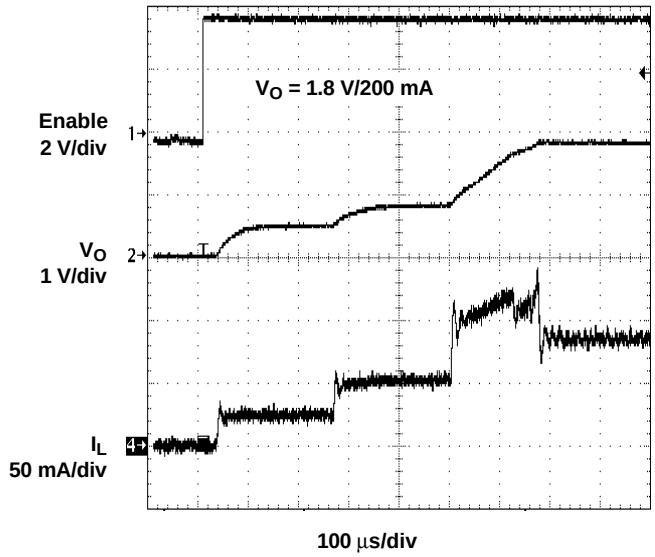


Figure 15.

APPLICATION INFORMATION

ADJUSTABLE OUTPUT VOLTAGE VERSION

When the adjustable output voltage version TPS62200 is used, the output voltage is set by the external resistor divider. See Figure 16.

The output voltage is calculated as

$$V_{out} = 0.5\ V \times \left(1 + \frac{R1}{R2}\right)$$

- R1 + R2 ≤ 1 MΩ and internal reference voltage V(ref)typ = 0.5 V

R1 + R2 should not be greater than 1 MΩ for reasons of stability. To keep the operating quiescent current to a minimum, the feedback resistor divider should have high impedance with R1+R2 ≤ 1 MΩ. Because of the high impedance and the low reference voltage of V_{ref} = 0.5 V, the noise on the feedback pin (FB) needs to be minimized. Using a capacitive divider C1 and C2 across the feedback resistors minimizes the noise at the feedback without degrading the line or load transient performance.

C1 and C2 should be selected as

$$C1 = \frac{1}{2 \times \pi \times 10\ kHz \times R1}$$

- R1 = upper resistor of voltage divider
- C1 = upper capacitor of voltage divider

For C1 a value should be chosen that comes closest to the calculated result.

$$C2 = \frac{R1}{R2} \times C1$$

- R2 = lower resistor of voltage divider
- C2 = lower capacitor of voltage divider

For C2 the selected capacitor value should always be selected larger than the calculated result. For example, in Figure 16 for C2, 100 pF are selected for a calculated result of C2 = 86.17 pF.

If quiescent current is not a key design parameter, C1 and C2 can be omitted, and a low-impedance feedback divider must be used with R1+R2 <100 kΩ. This design reduces the noise available on the feedback pin (FB) as well, but increases the overall quiescent current during operation.

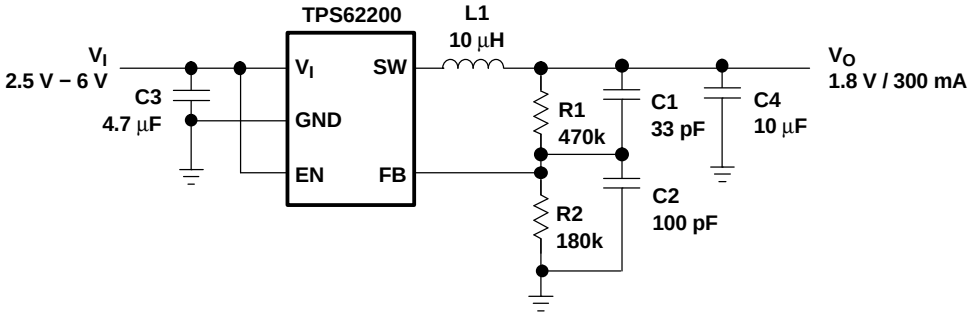


Figure 16. Typical Application Circuit for the Adjustable Output Voltage

INDUCTOR SELECTION

The TPS6220x device is optimized to operate with a typical inductor value of 10 µH.

For high efficiencies, the inductor should have a low dc resistance to minimize conduction losses. Although the inductor core material has less effect on efficiency than its dc resistance, an appropriate inductor core material must be used.

APPLICATION INFORMATION (continued)

The inductor value determines the inductor ripple current. The larger the inductor value, the smaller the inductor ripple current, and the lower the conduction losses of the converter. On the other hand, larger inductor values cause a slower load transient response. Usually the inductor ripple current, as calculated below, is around 20% of the average output current.

In order to avoid saturation of the inductor, the inductor should be rated at least for the maximum output current of the converter plus the inductor ripple current that is calculated as

$$\Delta I_L = Vout \times \frac{1-\frac{Vout}{Vin}}{L \times f} \qquad I_{Lmax} = I_{outmax} + \frac{\Delta I_L}{2}$$

f = switching frequency (1 MHz typical, 650 kHz minimal)

L = inductor valfue

ΔI_L = peak-to-peak inductor ripple current

I_{Lmax} = maximum inducator current

The highest inductor current occurs at maximum Vin.

A more conservative approach is to select the inductor current rating just for the maximum switch current of 670 mA. Refer to Table 1 for inductor recommendations.

Table 1. Recommended Inductors

INDUCTOR VALUE	COMPONENT SUPPLIER	COMMENTS
10 µH 10 µH 10 µH 10 µH	Sumida CDRH5D28-100 Sumida CDRH5D18-100 Sumida CDRH4D28-100 Coilcraft DO1608-103	High efficiency
6.8 µH 10 µH 10 µH 10 µH 10 µH	Sumida CDRH3D16-6R8 Sumida CDRH4D18-100 Sumida CR32-100 Sumida CR43-100 Murata LQH4C100K04	Smallest solution

INPUT CAPACITOR SELECTION

Because the buck converter has a pulsating input current, a low ESR input capacitor is required. This results in the best input voltage filtering and minimizing the interference with other circuits caused by high input voltage spikes. Also the input capacitor must be sufficiently large to stabilize the input voltage during heavy load transients. For good input voltage filtering, usually a 4.7 µF input capacitor is sufficient. It can be increased without any limit for better input-voltage filtering. If ceramic output capacitors are used, the capacitor RMS ripple current rating always meets the application requirements.

Ceramic capacitors show a good performance because of the low ESR value, and they are less sensitive against voltage transients and spikes compared to tantalum capacitors.

Place the input capacitor as close as possible to the input pin of the device for best performance (refer to Table 2 for recommended components).

OUTPUT CAPACITOR SELECTION

The advanced fast response voltage mode control scheme of the TPS6220x allows the use of tiny ceramic capacitors with a value of 10 µF without having large output voltage under and overshoots during heavy load transients.

Ceramic capacitors with low ESR values have the lowest output voltage ripple and are therefore recommended. If required, tantalum capacitors may be used as well (refer to Table 2 for recommended components).

At nominal load current the device operates in PWM mode and the overall output voltage ripple is the sum of the voltage spike caused by the output capacitor ESR plus the voltage ripple caused by charging and discharging the output capacitor:

$$\Delta Vout = Vout \times \frac{1-\frac{Vout}{Vin}}{L \times f} \times \left(\frac{1}{8 \times Cout \times f} + ESR\right)$$

where the highest output voltage ripple occurs at the highest input voltage Vin.

At light load currents, the device operates in power save mode, and the output voltage ripple is independent of the output capacitor value. The output voltage ripple is set by the internal comparator thresholds. The typical output voltage ripple is 1% of the output voltage Vo.

Table 2. Recommended Capacitors

CAPACITOR VALUE	CASE SIZE	COMPONENT SUPPLIER	COMMENTS
4.7 µF	0805	Taiyo Yuden JMK212BY475MG	Ceramic
10 µF	0805	Taiyo Yuden JMK212BJ106MG TDK C12012X5ROJ106K	Ceramic Ceramic
10 µF	1206	Taiyo Yuden JMK316BJ106KL TDK C3216X5ROJ106M	Ceramic
22 µF	1210	Taiyo Yuden JMK325BJ226MM	Ceramic

LAYOUT CONSIDERATIONS

For all switching power supplies, the layout is an important step in the design, especially at high peak currents and switching frequencies. If the layout is not carefully done, the regulator shows stability problems as well as EMI problems.

Therefore use wide and short traces for the main current paths, as indicated in bold in Figure 17. The input capacitor, as well as the inductor and output capacitor, should be placed as close as possible to the IC pins

The feedback resistor network must be routed away from the inductor and switch node to minimize noise and magnetic interference. To further minimize noise from coupling into the feedback network and feedback pin, the ground plane or ground traces must be used for shielding. This becomes very important especially at high switching frequencies of 1 MHz.

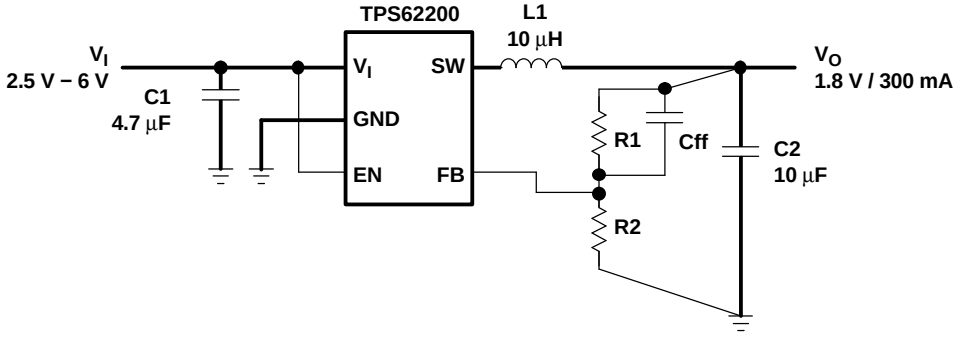


Figure 17. Layout Diagram

TYPICAL APPLICATIONS

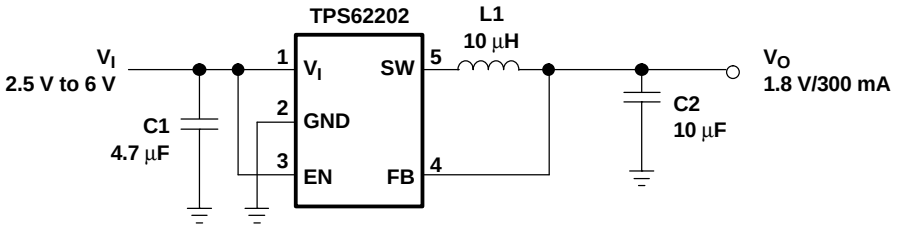


Figure 18. Li-Ion to 1.8 V Fixed Output Voltage Version

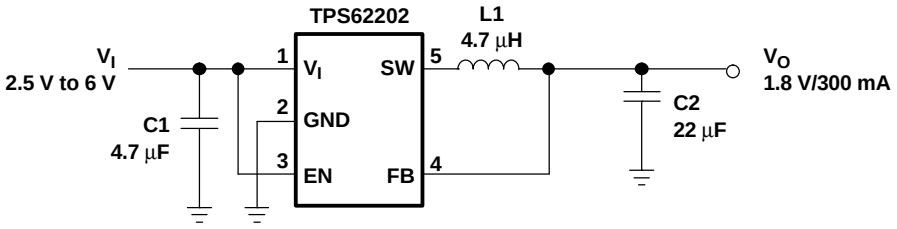


Figure 19. 1.8 V Fixed Output Voltage version Using 4.7µH Inductor

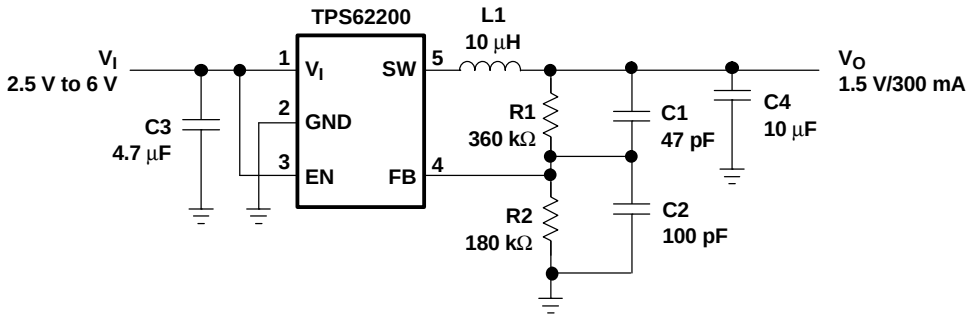


Figure 20. Adjustable Output Voltage Version Set to 1.5 V

PACKAGING INFORMATION

Orderable Device	Status (1)	Package Type	Package Drawing	Pins	Package Qty	Eco Plan (2)	Lead/Ball Finish	MSL Peak Temp (3)
TPS62200DBVR	ACTIVE	SOT-23	DBV	5	3000	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
TPS62200DBVRG4	ACTIVE	SOT-23	DBV	5	3000	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
TPS62200DBVT	ACTIVE	SOT-23	DBV	5	250	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
TPS62200DBVTG4	ACTIVE	SOT-23	DBV	5	250	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
TPS62201DBVR	ACTIVE	SOT-23	DBV	5	3000	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
TPS62201DBVRG4	ACTIVE	SOT-23	DBV	5	3000	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
TPS62201DBVT	ACTIVE	SOT-23	DBV	5	250	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
TPS62201DBVTG4	ACTIVE	SOT-23	DBV	5	250	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
TPS62202DBVR	ACTIVE	SOT-23	DBV	5	3000	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
TPS62202DBVRG4	ACTIVE	SOT-23	DBV	5	3000	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
TPS62202DBVT	ACTIVE	SOT-23	DBV	5	250	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
TPS62202DBVTG4	ACTIVE	SOT-23	DBV	5	250	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
TPS62203DBVR	ACTIVE	SOT-23	DBV	5	3000	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
TPS62203DBVRG4	ACTIVE	SOT-23	DBV	5	3000	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
TPS62203DBVT	ACTIVE	SOT-23	DBV	5	250	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
TPS62203DBVTG4	ACTIVE	SOT-23	DBV	5	250	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
TPS62204DBVR	ACTIVE	SOT-23	DBV	5	3000	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
TPS62204DBVRG4	ACTIVE	SOT-23	DBV	5	3000	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
TPS62204DBVT	ACTIVE	SOT-23	DBV	5	250	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
TPS62205DBVR	ACTIVE	SOT-23	DBV	5	3000	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
TPS62205DBVRG4	ACTIVE	SOT-23	DBV	5	3000	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
TPS62205DBVT	ACTIVE	SOT-23	DBV	5	250	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
TPS62207DBVR	ACTIVE	SOT-23	DBV	5	3000	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
TPS62207DBVRG4	ACTIVE	SOT-23	DBV	5	3000	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
TPS62207DBVT	ACTIVE	SOT-23	DBV	5	250	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM

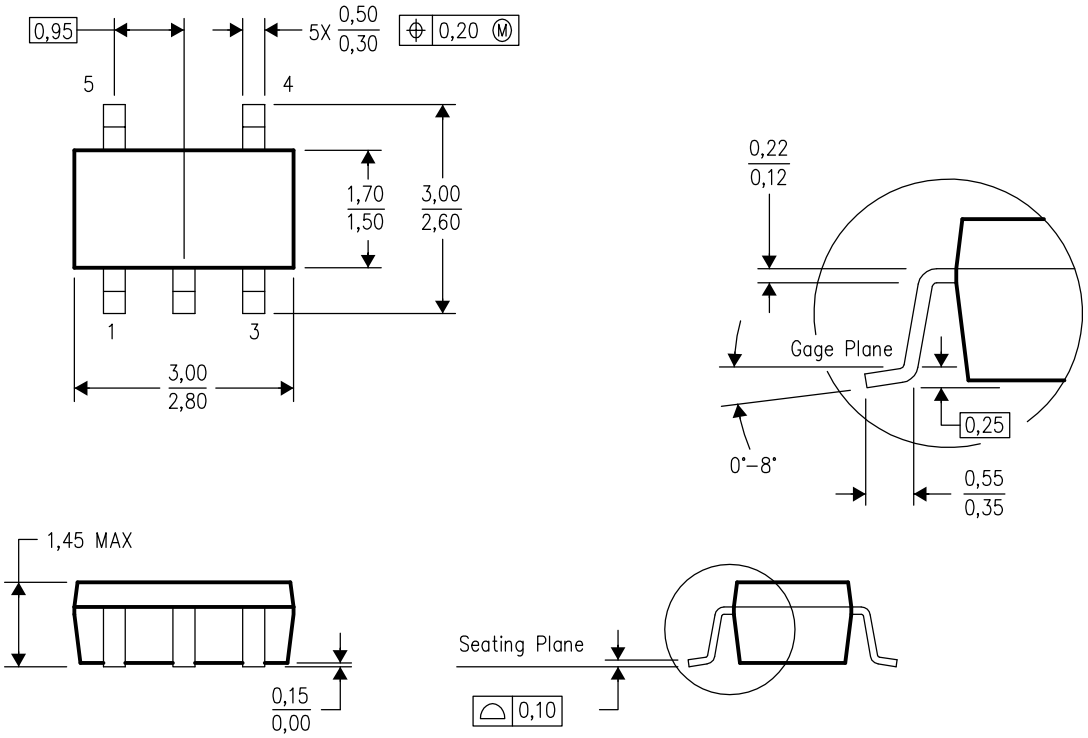
Orderable Device	Status [1]	Package Type	Package Drawing	Pins	Package Qty	Eco Plan [2]	Lead/Ball Finish	MSL Peak Temp [3]
TPS62207DBVTG4	ACTIVE	SOT-23	DBV	5	250	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
TPS62208DBVR	ACTIVE	SOT-23	DBV	5	3000	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
TPS62208DBVT	ACTIVE	SOT-23	DBV	5	250	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM

[1] The marketing status values are defined as follows:
ACTIVE: Product device recommended for new designs.
LIFEBUY: TI has announced that the device will be discontinued, and a lifetime-buy period is in effect.
NRND: Not recommended for new designs. Device is in production to support existing customers, but TI does not recommend using this part in a new design.
PREVIEW: Device has been announced but is not in production. Samples may or may not be available.
OBSOLETE: TI has discontinued the production of the device.

[2] Eco Plan - The planned eco-friendly classification: Pb-Free (RoHS) or Green (RoHS & no Sb/Br) - please check http://www.ti.com/productcontent for the latest availability information and additional product content details.
TBD: The Pb-Free/Green conversion plan has not been defined.
Pb-Free (RoHS): TI's terms "Lead-Free" or "Pb-Free" mean semiconductor products that are compatible with the current RoHS requirements for all 6 substances, including the requirement that lead not exceed 0.1% by weight in homogeneous materials. Where designed to be soldered at high temperatures, TI Pb-Free products are suitable for use in specified lead-free processes.
Green (RoHS & no Sb/Br): TI defines "Green" to mean Pb-Free (RoHS compatible), and free of Bromine (Br) and Antimony (Sb) based flame retardants (Br or Sb do not exceed 0.1% by weight in homogeneous material)

[3] MSL, Peak Temp. -- The Moisture Sensitivity Level rating according to the JEDEC industry standard classifications, and peak solder temperature.

DBV (R–PDSO–G5) PLASTIC SMALL–OUTLINE PACKAGE



4073253–4/H 10/2003

NOTES:
A. All linear dimensions are in millimeters.
B. This drawing is subject to change without notice.
C. Body dimensions do not include mold flash or protrusion.
D. Falls within JEDEC MO–178 Variation AA.

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Interface	interface.ti.com	Digital Control	www.ti.com/digitalcontrol
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
		Telephony	www.ti.com/telephony
		Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments
Post Office Box 655303 Dallas, Texas 75265

DATASHEET

SYMPHONY N1C
Solid State Audio

DRAFT Datasheet version 1.2.0 version 1.2

February 6, 2004
5:06

Philips
Semiconductors

PHILIPS

1 DOCUMENT HISTORY

Table 1 Document history

DATE	VERSION	AUTHOR	REMARKS
11-25-2002	0.1	Rachel Marée	Start of this document. Document is derived from LC_CD FRS 1.1
02-21-2003	0.2	Rachel Marée	Updated the document
28-07-2003	0.3	John van Tol	First internal draft review version
25-09-2003	1.0	John van Tol	Symphony N1A first release version
29-01-2004	1.2	John van Tol	Symphony N1C

2 FEATURES

2.1 General Features

- One chip solution for compressed audio players using flash memory
- Programmable architecture enables flexible support of “up-coming” digital music formats
  - This hold for running the decoder on the ARM
  - The decoders on the low-power DSP, are running from mask ROM
- FM Radio input and control support
- Embedded program flash for easy upgrading and increased program security
- Support for Philips LifeVibesTM audio enhancement algorithms
- Small footprint package TBGA180 10x10mm 0.5pitch

2.2 Hardware Features

- ARM7TDMI + 8kByte cache
- Embedded 64kByte RAM and 32 kByte ROM
- Integrated embedded program Flash (4M bit)
- Ultra low power Audio DSP for support of Philips LifeVibesTM audio enhancement algorithms (storede in teh DSP’s ROM).
- External memory support: Nand Flash/Compact flash/MMC/SMC/SRAM/ROM/SDRAM
- Integrated MCI interface
- Integrated USB 2.0 FS compliant slave interface (for firmware upgrade, data support from/to PC, streaming audio)
- Intelligent Configuration Power Management
- Single battery operated embedded DC/DC converter
- Integrated 6800/8080 compatible LCD interface
- General-Purpose IO pins (nearly all pins can be configured as GPIO pins)
- Integrated Master/Slave IIC interface
- Integrated ADC with line input and voice input (with recording possibility)
- Built-in ADC for level measurement & control (5-inputs)
- Integrated DAC with line output, headphone output with short-circuit protection
- Integrated IIS input and output interface
- Integrated SPDIF receiver
- Integrated UART + IRDA
- Integrated Real Time Clock with alarm
- Boundary scan

2.3 Possible software features

- MP3 encoding/decoding (*) => Support for MPEG 1 layer 3 and MPEG 2 layer 2.5 and layer 3 audio decoding (MP3)
- WMA decoding (*)
- AAC decoding (*)
- Ogg Vorbis decoding (*)
- Voice recording using ADPCM
- Intelligent power management software
- USB Mass Storage Class

- USB Device Firmware Upgrade
- Philips LifeVibesTM sound enhancement software including bass/treble/volume control.

(*) Audio decoders/encoders do need appropriate licenses.

3 GENERAL DESCRIPTION

The PNX0101 (ARM-based Solid State Audio IC) is an IC based on an embedded RISC processor. The device is designed for hand-held Solid State Audio applications like portable MP3 players. The high level of integration, low power consumption and high processor performances make the PNX0101 very suitable for portable hand-held devices.

The PNX0101 is based on the powerful ARM7TDMI CPU core, which is a full 32-bit RISC processor with 8 kbyte dedicated cache.

4 APPLICATIONS

- Portable Solid State Audio player

5 ORDERING INFORMATION

TYPE NUMBER	PACKAGE		
	NAME	DESCRIPTION	VERSION
PNX0101ET/N1	TFBGA180	Plastic low profile fine-pitch ball grid array package; 180 balls; body 10 x 10 x 0.8 mm	SOT640-1

6 GENERAL DESCRIPTION

The PNX0101 (ARM-based Solid State Audio IC) is an IC based on an embedded RISC processor. The device is designed for hand-held Solid State Audio applications like portable MP3 players. The high level of integration, low power consumption and high processor performances make the PNX0101 very suitable for portable hand-held devices.

The PNX0101 is based on the powerful ARM7TDMI CPU core, which is a full 32-bit RISC processor with 8 kbyte dedicated cache.

7 APPLICATIONS

- Portable Solid State Audio player

8 ORDERING INFORMATION

TYPE NUMBER	PACKAGE		
	NAME	DESCRIPTION	VERSION
PNX0101ET/N102	TFBGA180	Plastic low profile fine-pitch ball grid array package; 180 balls; body 10 x 10 x 0.8 mm	SOT640-1

WARNING
The package is manufactured with GREEN (environmental friendly materials) but has LEAD CONTAINING soldering balls.

9 PINNING

9.1 Pin Description PNX0101

WARNING
Pin JTAG_TMS initially was a pull-up pad .. in the latest version of the PNX0101, this is no longer the case .. an external pull-up resistor is however needed!!!
Pin DAO_WS is NO longer an GPIO pin .. switching this pin into the wrong mode, can cause problems with audio output.

Table 2 Pin list PNX0101

BGA PIN NAME	SYMBOL[1]	BGA BALL	PIN NO.	DIGITAL I/O LEVEL	APPL. FUNC	PIN STATE AFTER RESET [2]	CELLTYPE	DESCRIPTION
12 MHz oscillator (fixed: 4 pins)								
XTALH_IN	ffast_in	T10	72		A		apio (ZI)	12 MHz clock input
XTALH_OUT	ffast_out	V9	71		A		apio (IO)	12 MHz clock output
XTALH_VDDA18	vdda_fastosc	U9	70				vddco	Analog supply Oscillators
XTALH_VSSA	vssa_fastosc	T9	69				vssco	Analog ground Oscillators
32.768 kHz oscillator (fixed: 4 pins)								
XTALL_IN	fslow_in	V7	65		A		apio (ZI)	32.768 kHz clock input
XTALL_OUT	fslow_out	T8	66		A		apio (ZI)	32.768 kHz clock output
XTALL_VDDA18	XTALL_VDD18	U8	67				vddco	Analog supply Oscillators/PLL's
XTALL_GNDA	XTALL_GNDA	V8	68				vssco	Analog ground Oscillators/PLL's
bitslicer/SPDIF (fixed: 3 pins)								
SPDIF_IN	SPDIF_IN	T12	78		A		apio (IO)	SPDIF input
SPDIF_VDDA33	SPDIF_VDDA33	U11	76				vddco	Analog supply SPDIF input
SPDIF_GNDA	SPDIF_GNDA	T11	75				vssco	Analog ground SPDIF input
10-bit ADC (fixed: 7 pins)								
ADC10B_GPA4	ADC10B_GPA4	U5	58		A		apio (ZI)	Analog General Purpose pins
ADC10B_GPA3	ADC10B_GPA3	T6	59		A		apio (ZI)	Analog General Purpose pins
ADC10B_GPA2	ADC10B_GPA2	U6	61		A		apio (ZI)	Analog General Purpose pins
ADC10B_GPA1	ADC10B_GPA1	T7	63		A		apio (ZI)	Analog General Purpose pins
ADC10B_GPA0	ADC10B_GPA0	U7	64		A		apio (ZI)	Analog General Purpose pins
ADC10B_VDDA33	ADC10B_VDDA33	V10	74				vddco	Analog supply 10-bit ADC
ADC10B_GNDA	ADC10B_GNDA	U10	73				vssco	Analog ground 10-bit ADC
DAC (fixed: 14pins)								
DAC_VOUTR	DAC_VOUTR	M3	35		A		apio (IO)	SDAC Right Analog Output
DAC_VOUTL	DAC_VOUTL	M2	33		A		apio (IO)	SDAC Left Analog Output
DAC_VDDA33	DAC_VDDA33	L1	31				vddco	SDAC Positive Voltage

BGA PIN NAME	SYMBOL(1)	BGA BALL	PIN NO.	DIGITAL I/O LEVEL	APPL. FUNC	PIN STATE AFTER RESET (2)	CELLTYPE	DESCRIPTION
DAC_VREFP	DAC_VREFP	L2	30		A		apio (IO)	SDAC Positive Reference Voltage
DAC_VREFN	DAC_VREFN	M1	34		A		apio (IO)	SDAC Negative Reference Voltage
HP_OUTR	HP_OUTR	P3	39		A		apio (IO)	SDAC Right Headphone Output
HP_OUTL	HP_OUTL	N3	38		A		apio (IO)	SDAC Left Headphone Output
HP_OUTCA	HP_OUTC	N2	36		A		apio (IO)	HEADPHONE common output reference
HP_OUTCB	HP_OUTC	N1	37		A		apio (IO)	HEADPHONE common output reference
HP_VDDA33A	HP_VDDA33	R1	43				vddco	HEADPHONE analog supply
HP_VDDA33B	HP_VDDA33	R2	44				vddco	HEADPHONE analog supply
HP_GNDAA	HP_GNDA	P2	41				vssco	HEADPHONE analog ground
HP_GNDAB	HP_GNDA	P1	40				vssco	HEADPHONE analog ground
ADC (fixed: 11pins)								
ADC_MIC_LNA	ADC_MIC_LNA	T2	47		A		apio (IO)	Output of LNA of microphone input. To be connected to ADC_VINL or ADC_VINR via external capacitor if used.
ADC_VCOM	ADC_VCOM	T3	48		A		apio (IO)	ADC Common Reference Voltag e + HEADPHONE Reference Voltage combined on-chip.
ADC_VREFP	ADC_VREFP	U2	50		A		apio (IO)	ADC Positive Reference Voltage
ADC_VREFN	ADC_VREFN	V1	51		A		apio (IO)	ADC Negative Reference Voltage
ADC_VDDA18	ADC_VDDA18	V3	54				vddco	Analog supply ADC
ADC_VDDA33	ADC_VDDA33	U3	53				vddco	Analog supply ADC
ADC_GNDA	ADC_GNDA	V2	52				vssco	Analog ground ADC
ADC_VREF	ADC_VREF	U1	49		A		apio (IO)	ADC Reference Voltage
ADC_VINR	ADC_VINR	T1	46		A		apio (IO)	SADC Right Analog Input
ADC_VINL	ADC_VINL	T4	45		A		apio (IO)	SADC Left Analog Input
ADC_MIC	ADC_MIC	R3	42		A		apio (IO)	Microphone Input
LCD Interface (fixed: 12 pins)								
LCD_RW_WR	LCD_RW_WR	G2	18	0-5 VDC tolerant	I/O		bpts10tht5v	6800 read/write select 8080 active ‘high’ write enable
	GPIO_LCD_11							General Purpose IO pin
LCD_E_RD	LCD_E_RD	F2	15	0-5 VDC tolerant	I/O		bpts10tht5v	6800 enable 8080 active ‘high’ read enable
	GPIO_LCD_10							General Purpose IO pin

BGA PIN NAME	SYMBOL[1]	BGA BALL	PIN NO.	DIGITAL I/O LEVEL	APPL. FUNC	PIN STATE AFTER RESET [2]	CELLTYPE	DESCRIPTION
LCD_DB_7	LCD_DB_7	E3	14	0-5 VDC tolerant	I/O		bpts10tht5v	Data input 7/Data output 7/Serial data output/ 4-bit data 3
	GPIO_LCD_9							General Purpose IO pin
LCD_DB_6	LCD_DB_6	E2	12	0-5 VDC tolerant	I/O		bpts10tht5v	Data input 6/Data output 6/Serial data input/ 4-bit data 2
	GPIO_LCD_8							General Purpose IO pin
LCD_DB_5	LCD_DB_5	D3	11	0-5 VDC tolerant	I/O		bpts10tht5v	Data input 5/Data output 5/Serial clock output/ 4-bit data 1
	GPIO_LCD_7							General Purpose IO pin
LCD_DB_4	LCD_DB_4	D1	10	0-5 VDC tolerant	I/O		bpts10tht5v	Data input 4/Data output 4/ 4-bit data 0
	GPIO_LCD_6							General Purpose IO pin
LCD_DB_3	LCD_DB_3	D2	9	0-5 VDC tolerant	I/O		bpts10tht5v	Data input 3/Data output 3
	GPIO_LCD_5							General Purpose IO pin
LCD_DB_2	LCD_DB_2	C3	8	0-5 VDC tolerant	I/O		bpts10tht5v	Data input 2/Data output 2
	GPIO_LCD_4							General Purpose IO pin
LCD_DB_1	LCD_DB_1	C1	7	0-5 VDC tolerant	I/O		bpts10tht5v	Data input 1/Data output 1
	GPIO_LCD_3							General Purpose IO pin
LCD_DB_0	LCD_DB_0	C2	6	0-5 VDC tolerant	I/O		bpts10tht5v	Data input 0/Data output 0
	GPIO_LCD_2							General Purpose IO pin
LCD_CSB	LCD_CSB	B3	5	0-5 VDC tolerant	I/O		bpts10tht5v	Chip Select
	GPIO_LCD_1							General Purpose IO pin
LCD_RS	LCD_RS	F3	17	0-5 VDC tolerant	I/O		bpts10tht5v	‘high’ Data register select ‘low’ Instruction register select
	GPIO_LCD_0							General Purpose IO pin
Memory Card Interface (fixed: 6 pins)								
MCI_DAT_3	MCI_DAT_3	J3	26	0-5 VDC tolerant	I/O		bpts10tht5v	Data input/Data output
	GPIO_MCI_5							General Purpose IO pin
MCI_DAT_2	MCI_DAT_2	J1	25	0-5 VDC tolerant	I/O		bpts10tht5v	Data input/Data output
	GPIO_MCI_4							General Purpose IO pin
MCI_DAT_1	MCI_DAT_1	J2	24	0-5 VDC tolerant	I/O		bpts10tht5v	Data input/Data output
	GPIO_MCI_3							General Purpose IO pin
MCI_DAT_0	MCI_DAT_0	H3	23	0-5 VDC tolerant	I/O		bpts10tht5v	Data input/Data output
	GPIO_MCI_2							General Purpose IO pin
MCI_CLK	MCI_CLK	G3	20	0-5 VDC tolerant	I/O		bpts10tht5v	MCI clock output
	GPIO_MCI_1							General Purpose IO pin
MCI_CMD	MCI_CMD	H2	21	0-5 VDC tolerant	I/O		bpts10tht5v	Command input/Command output
	GPIO_MCI_0							General Purpose IO pin
USB Interface (fixed: 4 pins + (8 pins reserved for future))								

BGA PIN NAME	SYMBOL(1)	BGA BALL	PIN NO.	DIGITAL I/O LEVEL	APPL. FUNC	PIN STATE AFTER RESET (2)	CELLTYPE	DESCRIPTION
USB_CONNECT_N	USBAPB_TOP_CON_N	T15	87	0-5 VDC tolerant	I/O		bpts10tht5v	Soft connect output usb 2.0 FS
	GPIO_USB_1							General Purpose IO pin
	USB_RPU				A		apio (IO)	Reserved for usb 2.0 HS
USB_DP	USBAPB_TOPDP	U17	96		A		usb11	Positive USB data line usb 2.0 FS
	USB_DP						apio	Positive USB data line usb 2.0 HS (IO) usb_dp; (ZI) usb_dp_ioring;
USB_DM	USBAPB_TOPDM	T17	97		A		usb11	Negative USB data line usb 2.0 FS
	USB_DM						apio	Negative USB data line usb 2.0 HS (IO) usb_dm; (ZI) usb_dm_ioring;
USB_VBUS	USBAPB_TOP_VBUS	U14	85	0-5 VDC tolerant	I/O		bpts10tht5v	USB Supply detection line usb 2.0 FS & usb 2.0
	GPIO_USB_0							General Purpose IO pin
USB-RREF	USB_RREF	P16	94		I/O		bpts10tht5v	Reserved for usb 2.0 HS
Reserved6	USB_GNDA	R17	99				vssco	Reserved for usb 2.0 HS
Reserved5	USB_VSSA_REF	R16	91				vssco	Reserved for usb 2.0 HS
Reserved4	USB_VSSA_TERM	T16	90				vssco	Reserved for usb 2.0 HS
Reserved3	USB_VDDA18_PLL	U15	88				vddco	Reserved for usb 2.0 HS
Reserved2	USB_VDDA18_BG	U16	93				vddco	Reserved for usb 2.0 HS
Reserved1	USB_VDDA33	U18	100				vddco	Reserved for usb 2.0 HS
Reserved0	USB_VDDA33_DRV	V18	98				vddco	Reserved for usb 2.0 HS
DAI Interface (fixed: 3 pins)								
DAI_BCK	DAI_BCK	H17	121	0-5 VDC tolerant	I/O		bpts10tht5v	DAI Bitclock
	GPIO_DAI_2							General Purpose IO pin
DAI_WS	DAI_WS	G17	124	0-5 VDC tolerant	I/O		bpts10tht5v	DAI Wordselect
	GPIO_DAI_1							General Purpose IO pin
DAI_DATA	DAI_DATA	G16	123	0-5 VDC tolerant	I/O		bpts10tht5v	DAI Serial data input
	GPIO_DAI_0							General Purpose IO pin
DAO Interface (fixed: 3 pins)								
DAO_CLK	DAO_CLK	F16	126	0-5 VDC tolerant	I/O		bpts10tht5v	256 fs clock output
DAO_BCK	DAO_BCK	G18	125	0-5 VDC tolerant	I/O		bpts10tht5v	DAO Bitclock
	GPIO_DAO_2							General Purpose IO pin
DAO_WS	DAO_WS	F18	128	0-5 VDC tolerant	I/O		bpts10tht5v	DAO Wordselect
DAO_DATA	DAO_DATA	F17	127	0-5 VDC tolerant	I/O		bpts10tht5v	DAO Serial data output
	GPIO_DAO_0							General Purpose IO pin
JTAG (fixed: 6 pins)								
JTAG_TRST_N	trst_n	T13	81	0-5 VDC tolerant	I		ipthdt5v	JTAG Reset Input (pull-down)

BGA PIN NAME	SYMBOL(1)	BGA BALL	PIN NO.	DIGITAL I/O LEVEL	APPL. FUNC	PIN STATE AFTER RESET (2)	CELLTYPE	DESCRIPTION
JTAG_TCK	tck	V4	57	0-5 VDC tolerant	I		ipthut5v	JTAG Clock Input (pull-up)
JTAG_TMS	JTAG_TMS	U12	79	0-5 VDC tolerant	I		iptht5v	JTAG Mode Select Input
JTAG_TDI	tdi	T5	56	0-5 VDC tolerant	I		ipthut5v	JTAG Data Input (pull-up)
JTAG_TDO	tdo	U13	82	0-5 VDC tolerant	I/O		bpts10tht5v	JTAG Data Output
JTAG_SEL_ARM	jtagsel	U4	55	0-5 VDC tolerant	I		ipthdt5v	Jtag selection (pull-down)
IIC master/slave Interface (fixed: 2pins)								
IIC_SCL	i2c_scl_n	H16	120	0-5 VDC tolerant	I/O		iic400kt5v	Serial clock IIC Slave
IIC_SDA	i2c_sda_n	J17	118	0-5 VDC tolerant	I/O		iic400kt5v	Serial data IIC Slave
MPMC (fixed: 52 pins)								
MPMC_D_15	MPMC_D_15	B8	172		I/O		bpts10th	MPMC data input/output 15
	GPIO_MPMC_50							General Purpose IO pin
MPMC_D_14	MPMC_D_14	C8	173		I/O		bpts10th	MPMC data input/output 14
	GPIO_MPMC_49							General Purpose IO pin
MPMC_D_13	MPMC_D_13	B7	175		I/O		bpts10th	MPMC data input/output 13
	GPIO_MPMC_48							General Purpose IO pin
MPMC_D_12	MPMC_D_12	C7	176		I/O		bpts10th	MPMC data input/output 12
	GPIO_MPMC_47							General Purpose IO pin
MPMC_D_11	MPMC_D_11	B6	178		I/O		bpts10th	MPMC data input/output 11
	GPIO_MPMC_46							General Purpose IO pin
MPMC_D_10	MPMC_D_10	C6	179		I/O		bpts10th	MPMC data input/output 10
	GPIO_MPMC_45							General Purpose IO pin
MPMC_D_9	MPMC_D_9	C5	182		I/O		bpts10th	MPMC data input/output 9
	GPIO_MPMC_44							General Purpose IO pin
MPMC_D_8	MPMC_D_8	C4	186		I/O		bpts10th	MPMC data input/output 8
	GPIO_MPMC_43							General Purpose IO pin
MPMC_D_7	MPMC_D_7	B5	181		I/O		bpts10th	MPMC data input/output 7
	GPIO_MPMC_42							General Purpose IO pin
MPMC_D_6	MPMC_D_6	A5	180		I/O		bpts10th	MPMC data input/output 6
	GPIO_MPMC_41							General Purpose IO pin
MPMC_D_5	MPMC_D_5	B4	184		I/O		bpts10th	MPMC data input/output 5
	GPIO_MPMC_40							General Purpose IO pin
MPMC_D_4	MPMC_D_4	A4	183		I/O		bpts10th	MPMC data input/output 4
	GPIO_MPMC_39							General Purpose IO pin
MPMC_D_3	MPMC_D_3	A3	185		I/O		bpts10th	MPMC data input/output 3
	GPIO_MPMC_38							General Purpose IO pin
MPMC_D_2	MPMC_D_2	B2	3		I/O		bpts10th	MPMC data input/output 2
	GPIO_MPMC_37							General Purpose IO pin

BGA PIN NAME	SYMBOL[1]	BGA BALL	PIN NO.	DIGITAL I/O LEVEL	APPL. FUNC	PIN STATE AFTER RESET [2]	CELLTYPE	DESCRIPTION
MPMC_D_1	MPMC_D_1	A2	2		I/O		bpts10th	MPMC data input/output 1
	GPIO_MPMC_36							General Purpose IO pin
MPMC_D_0	MPMC_D_0	A1	1		I/O		bpts10th	MPMC data input/output 0
	GPIO_MPMC_35							General Purpose IO pin
MPMC_A_20	MPMC_A_20	C13	158		I/O		bpts10th	MPMC address 20
	GPIO_MPMC_34							General Purpose IO pin
MPMC_A_19	MPMC_A_19	B13	157		I/O		bpts10th	MPMC address 19
	GPIO_MPMC_33							General Purpose IO pin
MPMC_A_18	MPMC_A_18	A13	156		I/O		bpts10th	MPMC address 18
	GPIO_MPMC_32							General Purpose IO pin
MPMC_A_17	MPMC_A_17	C14	155		I/O		bpts10th	MPMC address 17
	GPIO_MPMC_31							General Purpose IO pin
MPMC_A_16	MPMC_A_16	B14	154		I/O		bpts10th	MPMC address 16
	GPIO_MPMC_30							General Purpose IO pin
MPMC_A_15	MPMC_A_15	A14	153		I/O		bpts10th	MPMC address 15
	GPIO_MPMC_29							General Purpose IO pin
MPMC_A_14	MPMC_A_14	C15	152		I/O		bpts10th	MPMC address 14 SDRAM memories use bits [14:0] Static memories use bits [20:0]
	GPIO_MPMC_28							General Purpose IO pin
MPMC_A_13	MPMC_A_13	B15	151		I/O		bpts10th	MPMC address 13
	GPIO_MPMC_27							General Purpose IO pin
MPMC_A_12	MPMC_A_12	C16	149		I/O		bpts10th	MPMC address 12
	GPIO_MPMC_26							General Purpose IO pin
MPMC_A_11	MPMC_A_11	B16	148		I/O		bpts10th	MPMC address 11
	GPIO_MPMC_25							General Purpose IO pin
MPMC_A_10	MPMC_A_10	C17	145		I/O		bpts10th	MPMC address 10
	GPIO_MPMC_24							General Purpose IO pin
MPMC_A_9	MPMC_A_9	B17	143		I/O		bpts10th	MPMC address 9
	GPIO_MPMC_23							General Purpose IO pin
MPMC_A_8	MPMC_A_8	C18	139		I/O		bpts10th	MPMC address 8
	GPIO_MPMC_22							General Purpose IO pin
MPMC_A_7	MPMC_A_7	B18	137		I/O		bpts10th	MPMC address 7
	GPIO_MPMC_21							General Purpose IO pin
MPMC_A_6	MPMC_A_6	A18	135		I/O		bpts10th	MPMC address 6
	GPIO_MPMC_20							General Purpose IO pin
MPMC_A_5	MPMC_A_5	D18	134		I/O		bpts10th	MPMC address 5
	GPIO_MPMC_19							General Purpose IO pin
MPMC_A_4	MPMC_A_4	D17	133		I/O		bpts10th	MPMC address 4
	GPIO_MPMC_18							General Purpose IO pin
MPMC_A_3	MPMC_A_3	D16	132		I/O		bpts10th	MPMC address 3
	GPIO_MPMC_17							General Purpose IO pin

BGA PIN NAME	SYMBOL(1)	BGA BALL	PIN NO.	DIGITAL I/O LEVEL	APPL. FUNC	PIN STATE AFTER RESET (2)	CELLTYPE	DESCRIPTION
MPMC_A_2	MPMC_A_2	E18	131		I/O		bpts10th	MPMC address 2
	GPIO_MPMC_16							General Purpose IO pin
MPMC_A_1	MPMC_A_1	E17	130		I/O		bpts10th	MPMC address 1
	GPIO_MPMC_15							General Purpose IO pin
MPMC_A_0	MPMC_A_0	E16	129		I/O		bpts10th	MPMC address 0
	GPIO_MPMC_14							General Purpose IO pin
MPMC_NSTCS_2	MPMC_NSTCS_2	B11	163		I/O		bpts10th	Static memory chip select 2. Default active LOW. Used for static memory device.
	GPIO_MPMC_13							General Purpose IO pin
MPMC_NSTCS_1	MPMC_NSTCS_1	A8	171		I/O		bpts10th	Static memory chip select 1. Default active LOW. Used for static memory device.
	GPIO_MPMC_12							General Purpose IO pin
MPMC_NSTCS_0	MPMC_NSTCS_0	C9	170		I/O		bpts10th	Static memory chip select 0. Default active LOW. Used for static memory device.
	GPIO_MPMC_11							General Purpose IO pin
MPMC_NDYCS	MPMC_NDYCS	B9	169		I/O		bpts10th	SDRAM chip select. Active LOW. Used for SDRAM device.
	GPIO_MPMC_10							General Purpose IO pin
MPMC_CLKOUT	MPMC_CLKOUT	A10	165		O		bpt4mt	Memory clock output. Connect to the clock input of SDRAM and SyncFlash devices.
MPMC_CKE	MPMC_CKE	B10	166		I/O		bpts10th	SDRAM clock enable. Active HIGH. Used for SDRAM devices.
	GPIO_MPMC_9							General Purpose IO pin
MPMC_NWE	MPMC_NWE	C11	164		I/O		bpts10th	Write enable for memories. Active LOW. Used for SDRAM and static memories.
	GPIO_MPMC_8							General Purpose IO pin
MPMC_NRAS	MPMC_NRAS	A9	168		I/O		bpts10th	Row address strobe for SDRAM devices and SyncFlash Devices. Active LOW. Used for SDRAM devices.
	GPIO_MPMC_7							General Purpose IO pin
MPMC_NCAS	MPMC_NCAS	C10	167		I/O		bpts10th	Column address strobe for SDRAM devices and SyncFlash Devices. Active LOW. Used for SDRAM devices.
	GPIO_MPMC_6							General Purpose IO pin

BGA PIN NAME	SYMBOL[1]	BGA BALL	PIN NO.	DIGITAL I/O LEVEL	APPL. FUNC	PIN STATE AFTER RESET [2]	CELLTYPE	DESCRIPTION
MPMC_DQM_1	MPMC_DQM_1	A11	162		I/O		bpts10th	Data mask output to SDRAMs. Active HIGH. The signal MPMCDQMOUT[1] mask byte [15:8] on the data bus. Used for SDRAM devices.
	GPIO_MPMC_5							General Purpose IO pin
MPMC_DQM_0	MPMC_DQM_0	C12	161		I/O		bpts10th	Data mask output to SDRAMs. Active HIGH. The signal MPMCDQMOUT[0] mask byte [7:0] on the data bus. Used for SDRAM devices.
	GPIO_MPMC_4							General Purpose IO pin
MPMC_NOE	MPMC_NOE	A17	141		I/O		bpts10th	Output enable for static memories. Active LOW. Used for static memory devices.
	GPIO_MPMC_3							General Purpose IO pin
MPMC_BLOUT_1	MPMC_BLOUT_1	B12	160		I/O		bpts10th	The signals nMPMCBLSOUT[1] select byte lane [15:8] on the data bus. Used for static memories.
	GPIO_MPMC_2							General Purpose IO pin
MPMC_BLOUT_0	MPMC_BLOUT_0	A12	159		I/O		bpts10th	The signals nMPMCBLSOUT[0] select byte lane [7:0] on the data bus. Used for static memories.
	GPIO_MPMC_1							General Purpose IO pin
MPMC_RPOUT	MPMC_RPOUT	B1	4		I/O		bpts10th	Reset power down to SyncFlash memory. Active LOW. Used for the synchronous memory controller.
	GPIO_MPMC_0							General Purpose IO pin
UART (fixed: 4 pins)								
UART_TXD	uart_txd	L3	32	0-5 VDC tolerant	I/O		bpts10tht5v	Serial output
	GPIO_UART_3							General Purpose IO pin
UART_RXD	uart_rxd	K3	29	0-5 VDC tolerant	I/O		bpts10tht5v	Serial input
	GPIO_UART_2							General Purpose IO pin
UART_NCTS	uart_cts_n	K2	27	0-5 VDC tolerant	I/O		bpts10tht5v	Clear to send (active low)
	GPIO_UART_1							General Purpose IO pin
UART_NRTS	uart_rts_n	K1	28	0-5 VDC tolerant	I/O		bpts10tht5v	Ready to send
	GPIO_UART_0							General Purpose IO pin
Mode Selection pins (fixed: 2 pins)								
GPIO_3	gpio3	J16	117	0-5 VDC tolerant	I/O		bpts10thdt5v	Start up MODE PIN2 (pull down)
GPIO_2	gpio2	K18	116	0-5 VDC tolerant	I/O		bpts10thdt5v	Start up MODE PIN1 (pull down)

BGA PIN NAME	SYMBOL[1]	BGA BALL	PIN NO.	DIGITAL I/O LEVEL	APPL. FUNC	PIN STATE AFTER RESET [2]	CELLTYPE	DESCRIPTION
GPIO (fixed: 2 pins)								
GPIO_1	gpio1	K17	115	0-5 VDC tolerant	I/O		bpts10tht5v	General Purpose IO pin (DFU)
GPIO_0	gpio0	K16	114	0-5 VDC tolerant	I/O		bpts10tht5v	General Purpose IO pin (STOP)
Reset input pin (fixed: 1 pin)								
RSTIN_N	rstin_n	T14	84	0-5 VDC tolerant	I		ipthut5v	System Reset Input (active low)
Flash pins: fixed: 1 pin								
FLASH_VDD_HV	FLASH_VDD_HV	V15	89				vddco	
Digital supplies (fixed: 6 pins)								
VDDI1	VDDI1	H1	22				vddco	Core supply (Mem)
VDDI2	VDDI2	V11	77				vddco	Core supply (Core)
VDDI3	VDDI3	V16	92				vddi	Core supply (Flash)
VSSI1	VSSI1	G1	19				vssco	Core ground (Mem)
VSSI2	VSSI2	V12	80				vssco	Core ground (Core)
VSSI3	VSSI3	V17	95				vssis	Core ground and substrate (Flash)
Peripheral supplies (fixed: 12 pins)								
VDDE1	VDDE1	E1	13				vdde3v3	Peripheral (I/O) supply (3.3V)
VDDE2	VDDE2	V5	60				vdde3v3	Peripheral (I/O) supply (3.3V)
VDDE3	VDDE3	V14	86				vdde3v3	Peripheral (I/O) supply (3.3V)
VDDE4	VDDE4	J18	119				vdde3v3	Peripheral (I/O) supply (3.3V)
VSSE1	VSSE1	F1	16				vsse3v3	Peripheral (I/O) ground
VSSE2	VSSE2	V6	62				vsse3v3	Peripheral (I/O) ground
VSSE3	VSSE3	V13	83				vsse3v3	Peripheral (I/O) ground
VSSE4	VSSE4	H18	122				vsse3v3	Peripheral (I/O) ground
VDDE5	VDDE5	A16	147				vdde3v3	MPMC Peripheral (I/O) supply (1.8V .. 3.3V)
VDDE6	VDDE6	A7	174				vdde3v3	MPMC Peripheral (I/O) supply (1.8V .. 3.3V)
VSSE5	VSSE5	A15	150				vsse3v3	MPMC Peripheral (I/O) ground
VSSE6	VSSE6	A6	177				vsse3v3	MPMC Peripheral (I/O) ground
DC/DC pins (fixed: 13 pins)								
DCDC_PLAY	dcdc_play	L17	112		A		apio	Play button input
DCDC_STOP	dcdc_stop	L18	113		A		apio	Stop signal input
DCDC_LX2	dcdc_lx2	N17	106		A		apio	connection to DC/DC2 external coil
DCDC_LX1	dcdc_lx1	P17	103		A		apio	connection to DC/DC1 external coil
DCDC_VUSB	dcdc_vusb_dcdc	T18	101		A		apio	USB supply voltage

BGA PIN NAME	SYMBOL[1]	BGA BALL	PIN NO.	DIGITAL I/O LEVEL	APPL. FUNC	PIN STATE AFTER RESET [2]	CELLTYPE	DESCRIPTION
DCDC_VBAT	dcdc_vbat	M17	109				vddco	battery supply voltage
DCDC_VOUT33A	dcdc_vout1	R18	102				vddco	DC/DC1 3.3v output voltage
DCDC_VOUT33B	dcdc_vout1b	M16	108				vddco	DC/DC1 3.3v input voltage
DCDC_VOUT18	dcdc_vout2	N18	107				vddco	DC/DC2 1.8v output voltage
DCDC_VSS1	dcdc_vss1	P18	104				vssco	ground for DC/DC1 Nswitch (no substrate connection)
DCDC_VSS2	dcdc_vss2	N16	105				vssco	ground for DC/DC2 Nswitch (no substrate connection)
DCDC_GND	dcdc_vssa	L16	111				vssis	Core ground and substrate
DCDC_CLEAN	dcdc_vssa_clean	M18	110				vssis	Reference circuit ground not connected to substrate

1. Pin positions are fixed, not mentioned pins are not connected.
2. Depending on application

9.2 Cell Type Explanation

Table 3 Cell Types Explanation

CELL NAME	EXPLANATION
ipthtt5v	Input Pad; Push Pull; TTL with Hysteresis; 5 Volt Tolerant
ipthut5v	Input Pad; Push Pull; TTL with Hysteresis; Pull Up; 5 Volt Tolerant
ipthdt5v	Input Pad; Push Pull; TTL with Hysteresis; Pull Down; 5 Volt Tolerant
ots10ct5v	Output Pad; 3state; 10ns Slew Rate Control; 5 Volt Tolerant
bpt4mt	Bidirectional Pad; Plain Input; 3state Output; 4mA Output Drive; **Not** 5 Volt Tolerant
bpts10tht5v	Bidirectional Pad; Plain Input; 3state Output; 10ns Slew Rate Control; TTL with Hysteresis; 5Volt Tolerant
bpts10thdt5v	Bidirectional Pad; Plain Input; 3state Output; 10 ns Slew Rate Control; TTL with Hysteresis; Pull Down; 5 Volt Tolerant
bpts10th	Bidirectional Pad; Plain Input; 3state Output; 10ns Slew Rate Control; TTL with Hysteresis; **Not** 5 Volt Tolerant
iic400kt5v	IIC Pad; 400 kHz IIC specification; 5Volt Tolerant
usb11	Universal Serial Bus Pad; Revision 1.1 specification
apio	Analog Pad; Analog Input/Output
vddi	Vdd Pad Connected to Core Vdd and internal Vdd Supply Rail in IO Ring
vddco	Vdd Pad Connected to Core Vdd
vdde3v3	Vdd Pad Connected to External (Noisy) 3.3 Volt Vdd Supply Rail
vssi	Vss Pad Connected to Core Vss and internal Vss Supply Rail in IO ring
vssco	Vss Pad Connected to Core Vss
vsse3v3	Vss Pad Connected to External (Noisy) 3.3 Volt Vss Supply Rail
vssis	Vss Pad Connected to Core Vss, internal Vss Supply Rail in IO Ring and Substrate Rail in IO Ring

10 BLOCK DIAGRAM

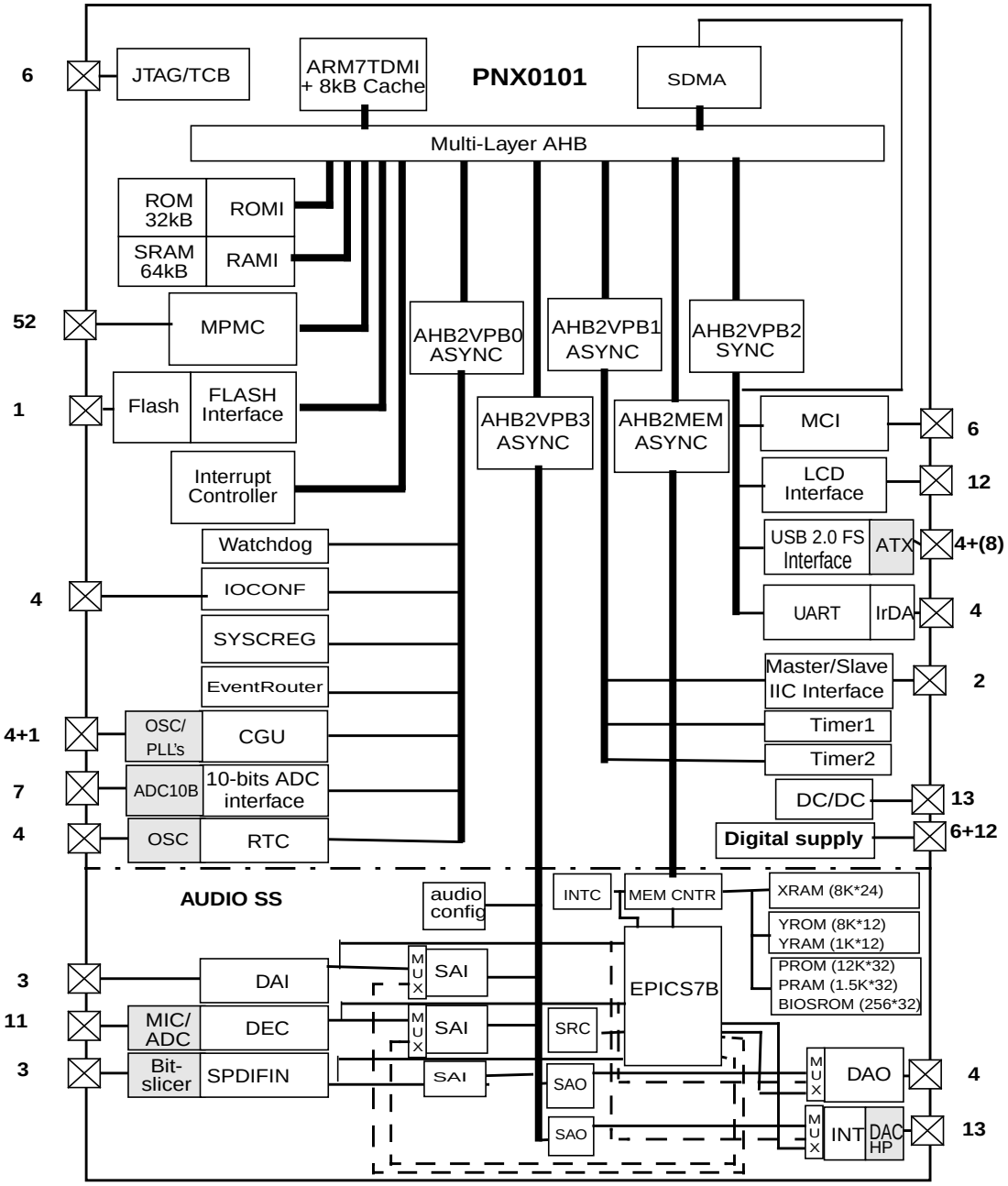


Fig. 1 Block diagram PNX0101 ET

11 MEMORY MAP

11.1 Memory Map Organisation

This section will concentrate on the PNX0101 memory map with the internal registers and memory definitions for both internal and external memories. For more detailed information use the module chapters of this document.

11.1.1 Memory Map

The memory map of PNX0101 can be found in Fig.2 and table 4 on page 33.

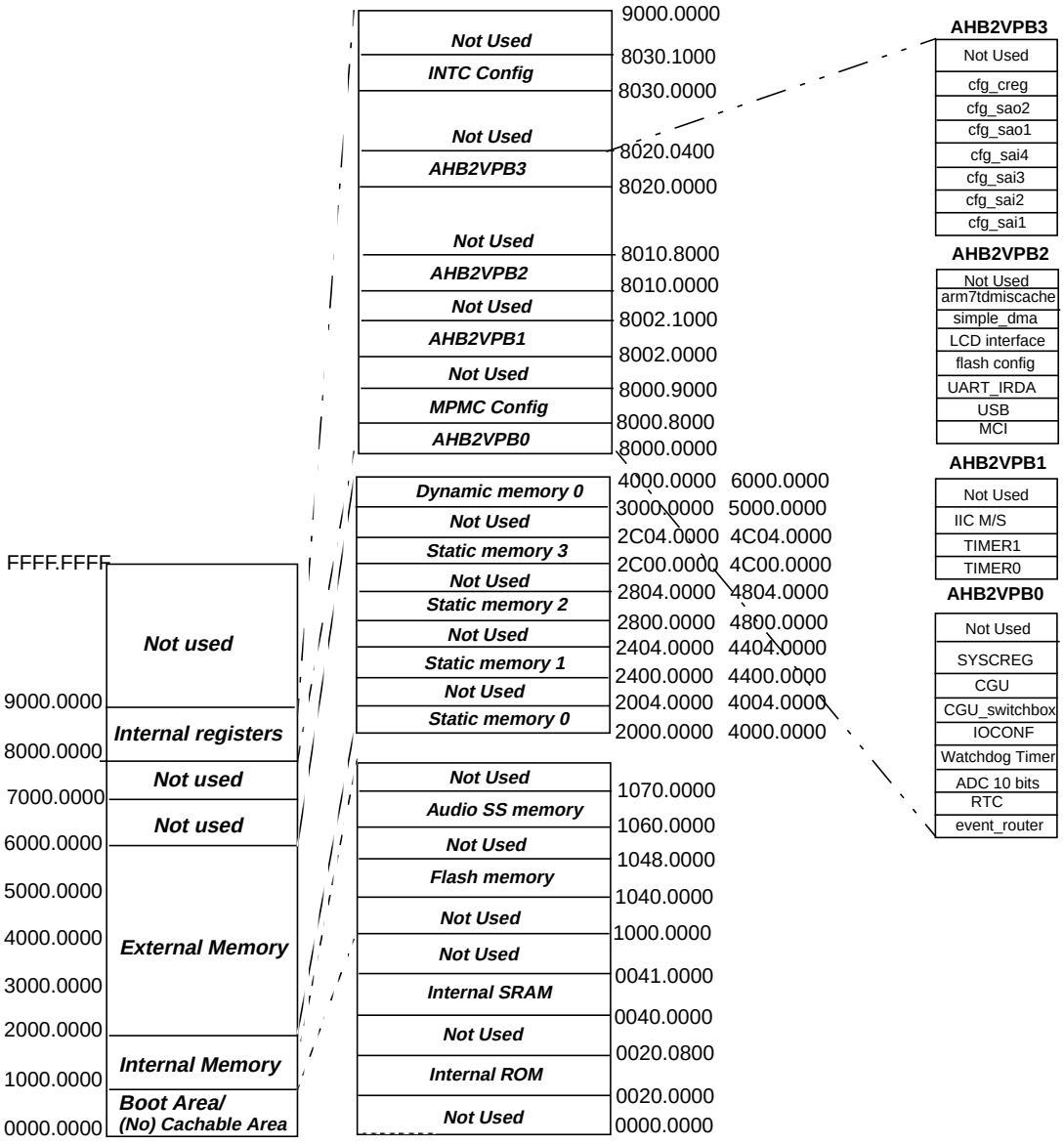


Fig. 2 PNX0101 Memory Map

11.1.2 Address Map

The next table will show the complete list of the PNX0101 address spaces, translated from Fig.2

Table 4 PNX0101 general address space

MODULE		MAX ADDRESS SPACE		ALLOCATED	DEVICE SIZE	REMARK
INTERNAL						
	Internal ROM	0x0020_0000	- 0x0020_7FFF	8K * 32 bits	32KByte	
	-	0x0020_8000	- 0x003F_FFFF	-	-	
	Internal SRAM	0x0040_0000	- 0x0040_FFFF	16K * 32 bits	64KByte	
	-	0x0041_0000	- 0x103F_FFFF	-	-	
	Internal Flash	0x1040_0000	- 0x1047_FFFF	32K * 128 bits	512 KByte	
	-	0x1048_0000	- 0x105F_FFFF	-	-	
	X-memory RAM_1	0x1060_0000	- 0x1060_FFFF	16K * 24 bits	30 KByte	10K words
	X-memory RAM_2	0x1061_0000	- 0x1061_FFFF	16K * 24 bits	-	
	X-memory RAM_3	0x1062_0000	- 0x1062_FFFF	16K * 24 bits	-	
	X-memory RAM_4	0x1063_0000	- 0x1063_FEFF	16K * 24 bits	-	
	DSP control-registers	0x1063_FF00	- 0x1063_FFFF	64 * 24 bits	256 Bytes	
	Y-memory ROM	0x1064_0000	- 0x1065_FFFF	16K * 12 bits	13.5 KByte	9K words
	Y-memory RAM_1	0x1066_0000	- 0x1066_FFFF	16K * 12 bits	1.5 KByte	1K words
	Y-memory RAM_2	0x1067_0000	- 0x1067_FFFF	16K * 12 bits	-	
	P-memory ROM	0x1068_0000	- 0x1069_FFFF	16K * 32 bits	56 KByte	14K words
	P-memory RAM_1	0x106A_0000	- 0x106B_FFFF	16K * 32 bits	8 KByte	2K words
	P-memory RAM_2	0x106C_0000	- 0x106D_FFFF	16K * 32 bits	-	
	DSP DIO-registers	0x106F_FF00	- 0x106F_FC48	64 * 24 bits	64 * 24 bits	
	DSP DMA-registers	0x106F_FFF8	- 0x106F_FFFE	64 * 24 bits	64 * 24 bits	
EXTERNAL	Static Memory 0	0x2000_0000	- 0x201F_FFFF	64 MByte	2048 KByte	
	-	0x2020_0000	- 0x23FF_FFFF	-	-	
	Static Memory 1	0x2400_0000	- 0x241F_FFFF	64 MByte	2048 KByte	
	-	0x2420_0000	- 0x27FF_FFFF	-	-	
	Static Memory 2	0x2800_0000	- 0x281F_FFFF	64 MByte	2048 KByte	
	-	0x2820_0000	- 0x2BFF_FFFF	-	-	
	Static Memory 3	0x2C00_0000	- 0x2C1F_FFFF	64 MByte	2048 KByte	
	-	0x2C20_0000	- 0x2FFF_FFFF	-	-	
	Dynamic Memory 0	0x3000_0000	- 0x3FFF_FFFF	512 MBit	512 MBit	
EXTERNAL	Static Memory 0	0x4000_0000	- 0x401F_FFFF	64 MByte	2048 KByte	mirror
	-	0x4020_0000	- 0x43FF_FFFF	-	-	
	Static Memory 1	0x4400_0000	- 0x441F_FFFF	64 MByte	2048 KByte	mirror
	-	0x4420_0000	- 0x47FF_FFFF	-	-	
	Static Memory 2	0x4800_0000	- 0x481F_FFFF	64 MByte	2048 KByte	mirror
	-	0x4820_0000	- 0x4BFF_FFFF	-	-	
	Static Memory 3	0x4C00_0000	- 0x4C1F_FFFF	64 MByte	2048 KByte	mirror
	-	0x4C20_0000	- 0x4FFF_FFFF	-	-	
	Dynamic Memory 0	0x5000_0000	- 0x5FFF_FFFF	512 MBit	512 MBit	mirror

MODULE		MAX ADDRESS SPACE		ALLOCATED	DEVICE SIZE	REMARK
INTERNAL	Peripherals	0x8000_0000	- 0x8000_7FFF	32 bits wide		AHB2VPB0
		0x8000_8000	- 0x8010_8FFF	32 bits wide		MPMC Config
		0x8002_0000	- 0x8002_0FFF	32 bits wide		AHB2VPB1
		0x8010_0000	- 0x8010_7FFF	32 bits wide		AHB2VPB2
		0x8020_0000	- 0x8020_03FF	32 bits wide		AHB2VPB3
		0x8030_0000	- 0x8030_0FFF	32 bits wide		INTC Config

11.1.3 Peripheral Address Map

Table 5 PNX0101 E7B Address Map

MODULE	BASE	OFFSET	REG NAME
E7B Memory	0x1060_0000 0x106F_FFFF	0x0_0000 0x0_9FFC	X-memory (DSP address 0x0000 - 0x27FF)
		0x3_FF00 0x3_FFFC	DSP Control Registers (DSP address 0xFFC0 - 0xFFFF)
		0x4_0000 0x4_8FFC	Y-memory ROM (DSP address 0x0000 - 0x23FF)
		0x6_0000 0x6_0FFC	Y-memory RAM (DSP address 0x8000 - 0x83FF)
		0x8_0000 0x8_FFFF	P-memory ROM (DSP address 0x0000 - 0x3800)
		0xA_0000 0xA_1FFC	P-memory RAM (DSP address 0x8000 - 0x87FF)
		0xF_FC00 0xF_FC48	DIO registers (DSP address 0xFF00 - 0xFF11)
		0xF_FF00 0xF_FFFF	DMA access registers

MODULE	BASE	OFFSET	REG NAME
DSP Control	0x1063_FF00	0x30	VH_E7B_AHB_IF_REG_INPUT_CC
	0x1063_FFFF	0x34	VH_E7B_AHB_IF_REG_INPUT_CD
		0x38	VH_E7B_AHB_IF_REG_INPUT_CE
		0x3C	VH_E7B_AHB_IF_REG_INPUT_CF
		0x40	VH_E7B_AHB_IF_REG_INPUT_D0
		0x44	VH_E7B_AHB_IF_REG_INPUT_D1
		0x48	VH_E7B_AHB_IF_REG_INPUT_D2
		0x4C	VH_E7B_AHB_IF_REG_INPUT_D3
		0x50	VH_E7B_AHB_IF_REG_OUTPUT_D4
		0x54	VH_E7B_AHB_IF_REG_OUTPUT_D5
		0x58	VH_E7B_AHB_IF_REG_OUTPUT_D6
		0x5C	VH_E7B_AHB_IF_REG_OUTPUT_D7
		0x60	VH_E7B_AHB_IF_REG_OUTPUT_D8
		0x64	VH_E7B_AHB_IF_REG_OUTPUT_D9
		0x68	VH_E7B_AHB_IF_REG_OUTPUT_DA
		0x6C	VH_E7B_AHB_IF_REG_OUTPUT_DB
		0x70	VH_E7B_AHB_IF_REG_OUTPUT_DC
		0x74	VH_E7B_AHB_IF_REG_OUTPUT_DD
		0x78	VH_E7B_AHB_IF_REG_OUTPUT_DE
		0x7C	VH_E7B_AHB_IF_REG_OUTPUT_DF
		0xC8	VH_E7B_AHB_IF_REG_DMA_IRQ_CNTR
		0xCC	VH_E7B_AHB_IF_REG_INTC_IRQ_USERFLAG
		0xD0	VH_E7B_AHB_IF_REG_INTC_SW_CLR
		0xD4	VH_E7B_AHB_IF_REG_INTC_TEST
		0xD8	VH_E7B_AHB_IF_REG_INTC_STATUS
		0xDC	VH_E7B_AHB_IF_REG_INTC_MASK
		0xE0	VH_E7B_AHB_IF_REG_INTC_MODE
		0xE4	VH_E7B_AHB_IF_REG_INTC_POLARITY
		0xE8	VH_E7B_AHB_IF_REG_DSP_CONFIG_2
		0xEC	VH_E7B_AHB_IF_REG_DSP_CONFIG_1
		0xF0	VH_E7B_AHB_IF_REG_DSP_IRQ_STACK
		0xF4	VH_E7B_AHB_IF_REG_DSP_STATUS_2
		0xF8	VH_E7B_AHB_IF_REG_DSP_STATUS_1
		0xFC	VH_E7B_AHB_IF_REG_DSP_PROG_CNTR

MODULE	BASE	OFFSET	REG NAME
DIO Registers	0x106F_FC00	0x00	VH_E7B_AHB_IF_REG_DIO_INPUT0
	0x106F_FC48	0x00	VH_E7B_AHB_IF_REG_DIO_OUTPUT0
		0x04	VH_E7B_AHB_IF_REG_DIO_INPUT1
		0x04	VH_E7B_AHB_IF_REG_DIO_OUTPUT1
		0x08	VH_E7B_AHB_IF_REG_DIO_INPUT2
		0x08	VH_E7B_AHB_IF_REG_DIO_OUTPUT2
		0x0C	VH_E7B_AHB_IF_REG_DIO_INPUT3
		0x0C	VH_E7B_AHB_IF_REG_DIO_OUTPUT3
		0x10	VH_E7B_AHB_IF_REG_DIO_INPUT4
		0x10	VH_E7B_AHB_IF_REG_DIO_OUTPUT4
		0x14	VH_E7B_AHB_IF_REG_DIO_INPUT5
		0x14	VH_E7B_AHB_IF_REG_DIO_OUTPUT5
		0x18	VH_E7B_AHB_IF_REG_DIO_INPUT6
		0x18	VH_E7B_AHB_IF_REG_DIO_OUTPUT6
		0x1C	VH_E7B_AHB_IF_REG_DIO_INPUT7
		0x1C	VH_E7B_AHB_IF_REG_DIO_OUTPUT7
		0x20	VH_E7B_AHB_IF_REG_DIO_INPUT8
		0x20	VH_E7B_AHB_IF_REG_DIO_OUTPUT8
		0x24	VH_E7B_AHB_IF_REG_DIO_INPUT9
		0x24	VH_E7B_AHB_IF_REG_DIO_OUTPUT9
		0x28	VH_E7B_AHB_IF_REG_DIO_INPUT10
		0x28	VH_E7B_AHB_IF_REG_DIO_OUTPUT10
		0x2C	VH_E7B_AHB_IF_REG_DIO_INPUT11
		0x2C	VH_E7B_AHB_IF_REG_DIO_OUTPUT11
		0x30	VH_E7B_AHB_IF_REG_DIO_INPUT12
		0x30	VH_E7B_AHB_IF_REG_DIO_OUTPUT12
		0x34	VH_E7B_AHB_IF_REG_DIO_INPUT13
		0x34	VH_E7B_AHB_IF_REG_DIO_OUTPUT13
		0x38	VH_E7B_AHB_IF_REG_DIO_INPUT14
		0x3C	VH_E7B_AHB_IF_REG_DIO_INPUT15
		0x40	VH_E7B_AHB_IF_REG_DIO_INPUT16
		0x44	VH_E7B_AHB_IF_REG_DIO_INPUT17
DMA Acess registers	0x106F_FF00 0x106F_FFFF	0xF8	VH_E7B_AHB_IF_REG_DSP_EIR
		0xFC	VH_E7B_AHB_IF_REG_DSP_CONTROL

Table 6 PNX0101 MMIO INTC Address Map

MODULE	BASE	OFFSET	REG NAME
MMIO INTC	0x8030_0000	0x000	VH_VINTC_PRIORITY_MASK0_REG
	0x8030_0FFF	0x004	VH_VINTC_PRIORITY_MASK1_REG
		0x100	VH_VINTC_VECTOR0_REG
		0x104	VH_VINTC_VECTOR1_REG
		0x200	VH_VINTC_PENDING_1_31_REG
		0x204	VH_VINTC_PENDING_32_63_REG
		0x208	VH_VINTC_PENDING_64_95_REG
		0x20C	VH_VINTC_PENDING_96_127_REG
		0x210	VH_VINTC_PENDING_128_159_REG
		0x214	VH_VINTC_PENDING_160_191_REG
		0x218	VH_VINTC_PENDING_192_223_REG
		0x21C	VH_VINTC_PENDING_224_255_REG
		0x300	VH_VINTC_FEATURES_REG
		0x404	VH_VINTC_REQUEST_1_REG
		0x7FC	VH_VINTC_REQUEST_255_REG
		0xFFC	VH_VINTC_MOD_ID_REG

Table 7 PNX0101 MPMC CFG Address Map

MODULE	BASE	OFFSET	REG NAME
MPMC PL172	0x8000_8000	0x000	VH_MPMC_CONTROL_REG
	0x8000_8FFF	0x004	VH_MPMC_STATUS_REG
		0x008	VH_MPMC_CONFIG_REG
		0x020	VH_MPMC_DYCNTL_REG
		0x024	VH_MPMC_DYREF_REG
		0x028	VH_MPMC_DYRDCFG_REG
		0x030	VH_MPMC_DYTRP_REG
		0x034	VH_MPMC_DYTRAS_REG
		0x038	VH_MPMC_DYTSREX_REG
		0x03C	VH_MPMC_DYTAPR_REG
		0x040	VH_MPMC_DYTDAL_REG
		0x044	VH_MPMC_DYTWR_REG
		0x048	VH_MPMC_DYTRC_REG
		0x04C	VH_MPMC_DYTRFC_REG
		0x050	VH_MPMC_DYTXSR_REG
		0x054	VH_MPMC_DYTRRD_REG
		0x058	VH_MPMC_DYTMRD_REG
		0x080	VH_MPMC_STEXDWT_REG
		0x100	VH_MPMC_DYCONFIG0_REG
		0x104	VH_MPMC_DYRASCAS0_REG
		0x120	VH_MPMC_DYCONFIG1_REG
		0x124	VH_MPMC_DYRASCAS1_REG
		0x140	VH_MPMC_DYCONFIG2_REG
		0x144	VH_MPMC_DYRASCAS2_REG
		0x160	VH_MPMC_DYCONFIG3_REG
		0x164	VH_MPMC_DYRASCAS3_REG
		0x200	VH_MPMC_STCONFIG0_REG
		0x204	VH_MPMC_STWTWEN0_REG
		0x208	VH_MPMC_STWTOEN0_REG
		0x20C	VH_MPMC_STWTRD0_REG
		0x210	VH_MPMC_STWTPG0_REG
		0x214	VH_MPMC_STWTWR0_REG
		0x218	VH_MPMC_STWTTURN0_REG
		0x220	VH_MPMC_STCONFIG1_REG
		0x224	VH_MPMC_STWTWEN1_REG
		0x228	VH_MPMC_STWTOEN1_REG
		0x22C	VH_MPMC_STWTRD1_REG
		0x230	VH_MPMC_STWTPG1_REG

MODULE	BASE	OFFSET	REG NAME
		0x234	VH_MPMC_STWTWR1_REG
		0x238	VH_MPMC_STWTTURN1_REG
		0x240	VH_MPMC_STCONFIG2_REG
		0x244	VH_MPMC_STWTWEN2_REG
		0x248	VH_MPMC_STWTOEN2_REG
		0x24C	VH_MPMC_STWTRD2_REG
		0x250	VH_MPMC_STWTPG2_REG
		0x254	VH_MPMC_STWTWR2_REG
		0x258	VH_MPMC_STWTTURN2_REG
		0x260	VH_MPMC_STCONFIG3_REG
		0x264	VH_MPMC_STWTWEN3_REG
		0x268	VH_MPMC_STWTOEN3_REG
		0x26C	VH_MPMC_STWTRD3_REG
		0x270	VH_MPMC_STWTPG3_REG
		0x274	VH_MPMC_STWTWR3_REG
		0x278	VH_MPMC_STWTTURN3_REG
		0xF00	VH_MPMC_ITCR_REG
		0xF20	VH_MPMC_ITIP_REG
		0xF40	VH_MPMC_ITOP_REG
		0xFD0	VH_MPMC_PERIPHID4_REG
		0xFD4	VH_MPMC_PERIPHID5_REG
		0xFD8	VH_MPMC_PERIPHID6_REG
		0xFDC	VH_MPMC_PERIPHID7_REG
		0xFE0	VH_MPMC_PERIPHID0_REG
		0xFE4	VH_MPMC_PERIPHID1_REG
		0xFE8	VH_MPMC_PERIPHID2_REG
		0xFEC	VH_MPMC_PERIPHID3_REG
		0xFF0	VH_MPMC_PCELLID0_REG
		0xFF4	VH_MPMC_PCELLID1_REG
		0xFF8	VH_MPMC_PCELLID2_REG
		0xFFC	VH_MPMC_PCELLID3_REG

Table 8 PNX0101 AHB2VPB0 Peripheral Address Map

MODULE	BASE	OFFSET	REG NAME
Event Router	0x8000_0000	0x0000	VH_CCP_EVENT_ROUTER_INT_REQ_REG
	0x8000_1FFF	0x0400	VH_CCP_EVENT_ROUTER_TARGET_REG
		0x0800	VH_CCP_EVENT_ROUTER_DTR_REG
		0x0C00	VH_CCP_EVENT_ROUTER_PEND_REG
		0x0C20	VH_CCP_EVENT_ROUTER_INT_CLR_REG
		0x0C40	VH_CCP_EVENT_ROUTER_INT_SET_REG
		0x0C60	VH_CCP_EVENT_ROUTER_MASK_REG
		0x0C80	VH_CCP_EVENT_ROUTER_MASK_CLR_REG
		0x0CA0	VH_CCP_EVENT_ROUTER_MASK_SET_REG
		0x0CC0	VH_CCP_EVENT_ROUTER_APR_REG
		0x0CE0	VH_CCP_EVENT_ROUTER_ATR_REG
		0x0D00	VH_CCP_EVENT_ROUTER_DEBOUNCE_REG
		0x0D20	VH_CCP_EVENT_ROUTER_RSR_REG
		0x0D40	VH_CCP_EVENT_ROUTER_INTOUT_REG
		0x0E00	VH_CCP_EVENT_ROUTER_FEATURES_REG
		0x0FFC	VH_CCP_EVENT_ROUTER_MODULE_ID_REG
		0x1000	VH_CCP_EVENT_ROUTER_INTOUT_PEND_REG
		0x1400	VH_CCP_EVENT_ROUTER_INTOUT_MASK_REG
		0x1800	VH_CCP_EVENT_ROUTER_INTOUT_MASK_CLR_REG
		0x1C00	VH_CCP_EVENT_ROUTER_INTOUT_MASK_SET_REG
Real Time Clock	0x8000_2000	0x00	SSARTCIntLoc
	0x8000_23FF	0x04	SSARTCPrescale
		0x08	SSARTCClk
		0x0c	SSARTCInc
		0x10	SSARTCAlrmMsk
		0x14	SSARTCCTime0
		0x18	SSARTCCTime1
		0x1c	SSARTCCTime2
		0x20	SSARTCSec
		0x24	SSARTCMin
		0x28	SSARTCHrs
		0x2c	SSARTCDOM
		0x30	SSARTCDOW
		0x34	SSARTCDOY
		0x38	SSARTCMon
		0x3c	SSARTCYrs
		0x60	SSARTCSecAlrm
		0x64	SSARTCMinAlrm
		0x68	SSARTCHrsAlrm
		0x6c	SSARTCDomAlrm
		0x70	SSARTCDowAlrm
		0x74	SSARTCDoyAlrm
		0x78	SSARTCMonAlrm
		0x7c	SSARTCYrsAlrm

MODULE	BASE	OFFSET	REG NAME
10 BIT ADC Interface	0x8000_2400	0x00	VH_ADC_R0_REG
	0x8000_27FF	0x04	VH_ADC_R1_REG
		0x08	VH_ADC_R2_REG
		0x0C	VH_ADC_R3_REG
		0x10	VH_ADC_R4_REG
		0x14	VH_ADC_R5_REG
		0x18	VH_ADC_R6_REG
		0x1C	VH_ADC_R7_REG
		0x20	VH_ADC_CON_REG
		0x24	VH_ADC_CSEL_RES_REG
		0x28	VH_ADC_INT_ENABLE_REG
		0x2C	VH_ADC_INT_STATUS_REG
		0x30	VH_ADC_INT_CLEAR_REG
Watchdog Timer	0x8000_2800	0x00	VH_WDOG_IR_REG
	0x8000_2BFF	0x04	VH_WDOG_TCR_REG
		0x08	VH_WDOG_TC_REG
		0x0C	VH_WDOG_PR_REG
		0x10	VH_WDOG_PC_REG
		0x14	VH_WDOG_MCR_REG
		0x18	VH_WDOG_MR0_REG
		0x1C	VH_WDOG_MR1_REG
		0x20	VH_WDOG_MR2_REG
		0x24	VH_WDOG_MR3_REG
		0x28	VH_WDOG_CCR_REG
		0x2C	VH_WDOG_CR0_REG
		0x30	VH_WDOG_CR1_REG
		0x34	VH_WDOG_CR2_REG
		0x38	VH_WDOG_CR3_REG
		0x3C	VH_WDOG_EMR_REG
IO CONF	0x8000_3000		block offset from base
	0x8000_3FFF	0x000	IOCONF_AHB_MPMC_PL1720
		0x040	IOCONF_AHB_MPMC_PL1721
		0x080	IOCONF_GPIO
		0x0c0	IOCONF_MELODY_ADSS
		0x100	IOCONF_SSA1_LCD_INTERFACE
		0x140	IOCONF_SSA1_MCI_PL180
		0x180	IOCONF_UART
		0x1c0	IOCONF_USBAPB_TOP
			register offset from block base
		0x00	IOCONF_REG_PINS
		0x10	IOCONF_REG_MODE0
		0x14	IOCONF_REG_MODE0_SET
		0x18	IOCONF_REG_MODE0_RESET
		0x20	IOCONF_REG_MODE1
		0x24	IOCONF_REG_MODE1_SET
		0x28	IOCONF_REG_MODE1_RESET

MODULE	BASE	OFFSET	REG NAME
CGU Switchbox	0x8000_4000 0x8000_4BFF	0	SCR_OFFSET (+ baseid)
		12	FS1_OFFSET (+ baseid)
		24	FS2_OFFSET (+ baseid)
		36	SSR_OFFSET (+ baseid)
		48	PCR_OFFSET (+ clkid)
		121	PSR_OFFSET (+ clkid)
		194	ESR_OFFSET (+ esrid)
		261	BCR_OFFSET (+ bcrid)
		264	FDC_OFFSET (+ fdid)
CGU (Clock Control Unit)	0x8000_4C00 0x8000_4FFF	0x000	CGU_CONFIG_POWERMODE
		0x004	CGU_CONFIG_WD_BARK
		0x008	CGU_CONFIG_F32KHZ_ON
		0x00c	CGU_CONFIG_F32KHZ_BYPASS
		0x010	CGU_CONFIG_FFAST_ON
		0x014	CGU_CONFIG_FFAST_BYPASS

MODULE	BASE	OFFSET	REG NAME
		0x018	CGU_CONFIG_VPB0_RESETN_SOFT
		0x01c	CGU_CONFIG_AHB2VPB0_PNRES_SOFT
		0x020	CGU_CONFIG_VPB1_RESETN_SOFT
		0x024	CGU_CONFIG_AHB2VPB1_PNRES_SOFT
		0x028	CGU_CONFIG_VPB2_RESETN_SOFT
		0x02c	CGU_CONFIG_VPB3_RESETN_SOFT
		0x030	CGU_CONFIG_AHB2VPB3_PNRES_SOFT
		0x034	CGU_CONFIG_AHB2MMIO_RESETN_SOFT
		0x038	CGU_CONFIG_AHB0_RESETN_SOFT
		0x03C	CGU_CONFIG_SSA1_TIMER0_PNRES_SOFT
		0x040	CGU_CONFIG_SSA1_TIMER1_PNRES_SOFT
		0x044	CGU_CONFIG_SSA1_MCI_PL180_PNRES_SOFT
		0x048	CGU_CONFIG_SSA1_MCI_PL180_NMCIRST_SOFT
		0x04c	CGU_CONFIG_USBAPB_TOP_PNRES_SOFT
		0x050	CGU_CONFIG_UART_SYS_RST_AN_SOFT
		0x054	CGU_CONFIG_I2C_PNRES_SOFT
		0x058	CGU_CONFIG_MELODY_ADSS_HRESETN_SOFT
		0x05c	CGU_CONFIG_MELODY_ADSS_PNRES_SOFT
		0x060	CGU_CONFIG_MELODY_ADSS_AHBIF_RST_N_SOFT
		0x064	CGU_CONFIG_MELODY_ADSS_DAI_RST_N_SOFT
		0x068	CGU_CONFIG_MELODY_ADSS_DAI2_RST_N_SOFT
		0x06c	CGU_CONFIG_MELODY_ADSS_DAO_RST_N_SOFT
		0x070	CGU_CONFIG_MELODY_ADSS_DEC_RST_N_SOFT
		0x074	CGU_CONFIG_MELODY_ADSS_EDET_RST_N_SOFT
		0x078	CGU_CONFIG_MELODY_ADSS_INT_RST_N_SOFT
		0x07c	CGU_CONFIG_MELODY_ADSS_SAI1_RST_N_SOFT
		0x080	CGU_CONFIG_MELODY_ADSS_SAI2_RST_N_SOFT
		0x084	CGU_CONFIG_MELODY_ADSS_SAI3_RST_N_SOFT
		0x088	CGU_CONFIG_MELODY_ADSS_SAI4_RST_N_SOFT
		0x08c	CGU_CONFIG_MELODY_ADSS_SAO1_RST_N_SOFT
		0x090	CGU_CONFIG_MELODY_ADSS_SAO2_RST_N_SOFT
		0x094	CGU_CONFIG_MELODY_ADSS_SPDIF_RST_N_SOFT
		0x098	CGU_CONFIG_IP_BLAS_2111_HRESETN_SOFT
		0x09c	CGU_CONFIG_SSA1_LCD_INTERFACE_PNRES_SOFT
		0x0a0	CGU_CONFIG_SIMPLE_DMA_PNRES_SOFT
		0x0a4	CGU_CONFIG_AHB_MPMC_PL172_HRESETN_SOFT
		0x0a8	CGU_CONFIG_MMIOINTC_RESETN_SOFT

MODULE	BASE	OFFSET	REG NAME
		0x0ac	CGU_CONFIG_HP0_FIN_SELECT
		0x0b0	CGU_CONFIG_HP0_MDEC
		0x0b4	CGU_CONFIG_HP0_NDEC
		0x0b8	CGU_CONFIG_HP0_PDEC
		0x0bc	CGU_CONFIG_HP0_MODE
		0x0c0	CGU_CONFIG_HP0_STATUS
		0x0c4	CGU_CONFIG_HP0_ACK
		0x0c8	CGU_CONFIG_HP0_REQ
		0x0cc	CGU_CONFIG_HP0_INSELR
		0x0d0	CGU_CONFIG_HP0_INSELI
		0x0d4	CGU_CONFIG_HP0_INSELP
		0x0d8	CGU_CONFIG_HP0_SELR
		0x0dc	CGU_CONFIG_HP0_SELI
		0x0e0	CGU_CONFIG_HP0_SELP
		0x0e4	CGU_CONFIG_LP0_FIN_SELECT
		0x0e8	CGU_CONFIG_LP0_PWD
		0x0ec	CGU_CONFIG_LP0_BYPASS
		0x0f0	CGU_CONFIG_LP0_LOCK
		0x0f4	CGU_CONFIG_LP0_DIRECT
		0x0f8	CGU_CONFIG_LP0_MSEL
		0x0fc	CGU_CONFIG_LP0_PSEL

MODULE	BASE	OFFSET	REG NAME
Syscreg	0x8000_5000	0x000	SYSCREG_ACTIVATE_TESTPINS
	0x8000_53FF	0x004	SYSCREG_DCDC_CONVERTER_DCDC1
		0x008	SYSCREG_DCDC_CONVERTER_DCDC2
		0x00c	SYSCREG_DCDC_CONVERTER_CLK
		0x010	SYSCREG_FUSEBOX_FO0
		0x014	SYSCREG_FUSEBOX_FO1
		0x018	SYSCREG_FUSEBOX_FO2
		0x01c	SYSCREG_FUSEBOX_FO3
		0x020	SYSCREG_SEL
		0x024	SYSCREG_SSA1_RTC_CFG
		0x028	SYSCREG_SSA1_ADC_PD_ADC10BITS
		0x02c	SYSCREG_SSA1_MCI_PL180_PRESERVED
		0x030	SYSCREG_USBAPB_TOP_STS
		0x034	SYSCREG_USBAPB_TOP_CFG
		0x038	SYSCREG_SW_IP_BLAS_2111_PWD_N
		0x03c	SYSCREG_IP_BLAS_2111_RSTBUSY
		0x040	SYSCREG_CGU_DYN_HP0
		0x044	SYSCREG_CGU_DYN_LP0
		0x048	SYSCREG_SYS_CREG_SDMA_EXT_EN_3
		0x04c	SYSCREG_SYS_CREG_SDMA_EXT_EN_5
		0x050	SYSCREG_SELECTION_CFG
		0x054	SYSCREG_ISRAM_LATENCY_CFG
		0x058	SYSCREG_ISROM_LATENCY_CFG
		0x05c	SYSCREG_AHB_MPMC_PL172_MISC
		0x060	SYSCREG_MPMP_DELAYMODES
		0x064	SYSCREG_WIRE_EBI_MSIZE_INIT
		0x068	SYSCREG_AHB_BOOT
		0x06c	SYSCREG_ARM7TDMISCACHE_SHADOW_POINTER
		0x070	SYSCREG_SLEEPSTATUS
		0x074	SYSCREG_CHIP_ID

Table 9 PNX0101 AHB2VPB1 Peripheral Address Map

MODULE	BASE	OFFSET	REG NAME
Timer0	0x8002_0000	0x00	SSATimerLoad
	0x8002_03FF	0x04	SSATimerValue
		0x08	SSATimerCtrl
		0x0C	SSATimerClear
		0x10	SSATimerTest
Timer1	0x8002_0400	0x00	SSATimerLoad
	0x8002_07FF	0x04	SSATimerValue
		0x08	SSATimerCtrl
		0x0C	SSATimerClear
		0x10	SSATimerTest
IIC M/S	0x8002_0800 0x8002_0BFF	0x00	I2C_RX_FIFO_REG
		0x00	I2C_TX_FIFO_REG
		0x04	I2C_STS_REG
		0x08	I2C_CTL_REG
		0x0C	I2C_CKL_REG
		0x10	I2C_CKH_REG
		0x14	I2C_ADR_REG
		0x18	I2C_RFL_REG
		0x1C	I2C_TFL_REG
		0x20	I2C_RXB_REG
		0x24	I2C_TXB_REG
		0x28	I2C_TXS_REG
		0x2C	I2C_STFL_REG

Table 10 PNX0101 AHB2VPB2 Peripheral Address Map

MODULE	BASE	OFFSET	REG NAME
MCI Controller	0x8010_0000 0x8010_0FFF	0x00	VH_SSA1_MCIPOWER_REG
		0x04	VH_SSA1_MCICLOCK_REG
		0x08	VH_SSA1_MCIARGUMENT_REG
		0x0C	VH_SSA1_MCICOMMAND_REG
		0x10	VH_SSA1_MCIRESPCMD_REG
		0x14	VH_SSA1_MCIRESPONSE0_REG
		0x18	VH_SSA1_MCIRESPONSE1_REG
		0x1C	VH_SSA1_MCIRESPONSE2_REG
		0x20	VH_SSA1_MCIRESPONSE3_REG
		0x24	VH_SSA1_MCIDATATIMER_REG
		0x28	VH_SSA1_MCIDATALENGTH_REG
		0x30	VH_SSA1_MCIDATACTRL_REG
		0x34	VH_SSA1_MCIDATACNT_REG
		0x38	VH_SSA1_MCISTATUS_REG
		0x3C	VH_SSA1_MCICLEAR_REG
		0x40	VH_SSA1_MCIMASK0_REG
		0x44	VH_SSA1_MCIMASK1_REG
		0x48	VH_SSA1_MCISELECT_REG
		0x4C	VH_SSA1_MCIFIFOCNT_REG
		0x4C .. 0x7C	VH_SSA1_MCIRESERVED_REG
		0x80 .. 0xBC	VH_SSA1_MCIFIFO_REG
		0xFE0	VH_SSA1_MCIPERIPHI0_REG
		0xFE4	VH_SSA1_MCIPERIPHI1_REG
		0xFE8	VH_SSA1_MCIPERIPHI2_REG
		0xFEC	VH_SSA1_MCIPERIPHI3_REG
		0xFF0	VH_SSA1_MCIPCELLID0_REG
		0xFF4	VH_SSA1_MCIPCELLID1_REG
		0xFF8	VH_SSA1_MCIPCELLID2_REG
		0xFFC	VH_SSA1_MCIPCELLID3_REG
USB	0x8010_1000 0x8010_13FF	0x00	vh_usbapb_topINT_STAT_REG
		0x04	vh_usbapb_topINT_EN_REG
		0x08	vh_usbapb_topINT_CLR_REG
		0x0C	vh_usbapb_topINT_SET_REG
		0x10	vh_usbapb_topCMD_CODE_REG
		0x14	vh_usbapb_topCMD_DATA_REG
		0x18	vh_usbapb_topRX_DATA_REG
		0x1C	vh_usbapb_topTX_DATA_REG
		0x20	vh_usbapb_topRX_PLENGTH_REG
		0x24	vh_usbapb_topTX_PLENGTH_REG
		0x28	vh_usbapb_topUSB_CNTRL_REG
		0x2C	vh_usbapb_topFIQ_SEL_REG

MODULE	BASE	OFFSET	REG NAME
UART	0x8010_2000 0x8010_2FFF	0x00	VH_UART_RBR_REG
		0x00	VH_UART_THR_REG
		0x00	VH_UART_DLL_REG
		0x04	VH_UART_DLM_REG
		0x04	VH_UART_IER_REG
		0x08	VH_UART_IIR_REG
		0x08	VH_UART_FCR_REG
		0x0C	VH_UART_LCR_REG
		0x10	VH_UART_MCR_REG
		0x14	VH_UART_LSR_REG
		0x18	VH_UART_MSR_REG
		0x1C	VH_UART_SCR_REG
		0x20	VH_UART_ACR_REG
		0x24	VH_UART_ICR_REG
		0x28	VH_UART_FDR_REG
		0x2C	VH_UART_POP_REG
		0x30	VH_UART_MODE_REG
		0x34	VH_UART_CFG_REG
		0x38	VH_UART_INTCE_REG
		0x3C	VH_UART_INTSE_REG
		0x40	VH_UART_INTS_REG
		0x44	VH_UART_INTE_REG
		0x48	VH_UART_INTCS_REG
Flash interface	0x8010_3000 0x8010_3FFF	0x000	EFLASH_CTRL_REG
		0x004	EFLASH_STATUS_REG
		0x008	EFLASH_PROGRAM_TIME_REG
		0x00C	EFLASH_TEST_CTRL_REG
		0x010	EFLASH_BRIDGE_WSTATE_REG
		0x018	EFLASH_MEM_STATUS_REG
		0x01C	EFLASH_CLOCK_DIV_REG
		0x020	EFLASH_BIST_ADD_START_REG
		0x024	EFLASH_BIST_ADD_STOP_REG
		0x028	EFLASH_BIST_16BIT_SIGN_REG
		0x02C	EFLASH_BIST_128BIT_SIGN_W0_REG
		0x030	EFLASH_BIST_128BIT_SIGN_W1_REG
		0x034	EFLASH_BIST_128BIT_SIGN_W2_REG
		0x038	EFLASH_BIST_128BIT_SIGN_W3_REG
		0xFD8	EFLASH_CLR_ENABLE_INT_REG
		0xFDC	EFLASH_SET_ENABLE_INT_REG
		0xFE0	EFLASH_RD_STATUS_INT_REG
		0xFE4	EFLASH_RD_ENABLE_INT_REG
		0xFE8	EFLASH_CLR_STATUS_INT_REG
		0xFEC	EFLASH_SET_STATUS_INT_REG
		0xFFC	EFLASH_MODULE_ID_REG

MODULE	BASE	OFFSET	REG NAME
LCD Interface	0x8010_4000 0x8010_43FF	0x000	VH_LCD_INTERFACE_STATUS_REG
		0x004	VH_LCD_INTERFACE_CONTROL_REG
		00x08	VH_LCD_INTERFACE_INT_RAW_REG
		0x00C	VH_LCD_INTERFACE_INT_CLEAR_REG
		0x010	VH_LCD_INTERFACE_INT_MASK_REG
		0x014	VH_LCD_INTERFACE_READ_CMD_REG
		0x020	VH_LCD_INTERFACE_INST_BYTE_REG
		0x030	VH_LCD_INTERFACE_DATA_BYTE_REG
		0x040	VH_LCD_INTERFACE_INST_WORD_REG
		0x080	VH_LCD_INTERFACE_DATA_WORD_REG

MODULE	BASE	OFFSET	REG NAME
Simple DMA	0x8010_4800	0x000	VH_SIMPLE_DMA_SOURCE_REG_0
	0x8010_4FFF	0x004	VH_SIMPLE_DMA_DESTINATION_REG_0
		0x008	VH_SIMPLE_DMA_LENGTH_REG_0
		0x00C	VH_SIMPLE_DMA_CONFIG_REG_0
		0x010	VH_SIMPLE_DMA_ENABLE_REG_0
		0x01C	VH_SIMPLE_DMA_COUNTER_REG_0
		0x020	VH_SIMPLE_DMA_SOURCE_REG_1
		0x024	VH_SIMPLE_DMA_DESTINATION_REG_1
		0x028	VH_SIMPLE_DMA_LENGTH_REG_1
		0x02C	VH_SIMPLE_DMA_CONFIG_REG_1
		0x030	VH_SIMPLE_DMA_ENABLE_REG_1
		0x03C	VH_SIMPLE_DMA_COUNTER_REG_1
		0x040	VH_SIMPLE_DMA_SOURCE_REG_2
		0x044	VH_SIMPLE_DMA_DESTINATION_REG_2
		0x048	VH_SIMPLE_DMA_LENGTH_REG_2
		0x04C	VH_SIMPLE_DMA_CONFIG_REG_2
		0x050	VH_SIMPLE_DMA_ENABLE_REG_2
		0x05C	VH_SIMPLE_DMA_COUNTER_REG_2
		0x060	VH_SIMPLE_DMA_SOURCE_REG_3
		0x064	VH_SIMPLE_DMA_DESTINATION_REG_3
		0x068	VH_SIMPLE_DMA_LENGTH_REG_3
		0x06C	VH_SIMPLE_DMA_CONFIG_REG_3
		0x070	VH_SIMPLE_DMA_ENABLE_REG_3
		0x07C	VH_SIMPLE_DMA_COUNTER_REG_3
		0x080	VH_SIMPLE_DMA_SOURCE_REG_4
		0x084	VH_SIMPLE_DMA_DESTINATION_REG_4
		0x088	VH_SIMPLE_DMA_LENGTH_REG_4
		0x08C	VH_SIMPLE_DMA_CONFIG_REG_4
		0x090	VH_SIMPLE_DMA_ENABLE_REG_4
		0x09C	VH_SIMPLE_DMA_COUNTER_REG_4
		0x0a0	VH_SIMPLE_DMA_SOURCE_REG_5
		0x0a4	VH_SIMPLE_DMA_DESTINATION_REG_5
		0x0a8	VH_SIMPLE_DMA_LENGTH_REG_5
		0x0aC	VH_SIMPLE_DMA_CONFIG_REG_5
		0x0b0	VH_SIMPLE_DMA_ENABLE_REG_5
		0x0BC	VH_SIMPLE_DMA_COUNTER_REG_5
		0x0c0	VH_SIMPLE_DMA_SOURCE_REG_6
		0x0c4	VH_SIMPLE_DMA_DESTINATION_REG_6
		0x0c8	VH_SIMPLE_DMA_LENGTH_REG_6
		0x0cC	VH_SIMPLE_DMA_CONFIG_REG_6
		0x0d0	VH_SIMPLE_DMA_ENABLE_REG_6
		0x0dC	VH_SIMPLE_DMA_COUNTER_REG_6
		0x0e0	VH_SIMPLE_DMA_SOURCE_REG_7
		0x0e4	VH_SIMPLE_DMA_DESTINATION_REG_7
		0x0e8	VH_SIMPLE_DMA_LENGTH_REG_7
		0x0eC	VH_SIMPLE_DMA_CONFIG_REG_7

MODULE	BASE	OFFSET	REG NAME
		0x0F0	VH_SIMPLE_DMA_ENABLE_REG_7
		0x0FC	VH_SIMPLE_DMA_COUNTER_REG_7
		0x200	VH_SIMPLE_DMA_ALT_SOURCE_REG_0
		0x204	VH_SIMPLE_DMA_ALT_DESTINATION_REG_0
		0x208	VH_SIMPLE_DMA_ALT_LENGTH_REG_0
		0x20C	VH_SIMPLE_DMA_ALT_CONFIG_REG_0
		0x210	VH_SIMPLE_DMA_ALT_SOURCE_REG_1
		0x214	VH_SIMPLE_DMA_ALT_DESTINATION_REG_1
		0x218	VH_SIMPLE_DMA_ALT_LENGTH_REG_1
		0x21C	VH_SIMPLE_DMA_ALT_CONFIG_REG_1
		0x220	VH_SIMPLE_DMA_ALT_SOURCE_REG_2
		0x224	VH_SIMPLE_DMA_ALT_DESTINATION_REG_2
		0x228	VH_SIMPLE_DMA_ALT_LENGTH_REG_2
		0x22C	VH_SIMPLE_DMA_ALT_CONFIG_REG_2
		0x230	VH_SIMPLE_DMA_ALT_SOURCE_REG_3
		0x234	VH_SIMPLE_DMA_ALT_DESTINATION_REG_3
		0x238	VH_SIMPLE_DMA_ALT_LENGTH_REG_3
		0x23C	VH_SIMPLE_DMA_ALT_CONFIG_REG_3
		0x240	VH_SIMPLE_DMA_ALT_SOURCE_REG_4
		0x244	VH_SIMPLE_DMA_ALT_DESTINATION_REG_4
		0x248	VH_SIMPLE_DMA_ALT_LENGTH_REG_4
		0x24C	VH_SIMPLE_DMA_ALT_CONFIG_REG_4
		0x250	VH_SIMPLE_DMA_ALT_SOURCE_REG_5
		0x254	VH_SIMPLE_DMA_ALT_DESTINATION_REG_5
		0x258	VH_SIMPLE_DMA_ALT_LENGTH_REG_5
		0x25C	VH_SIMPLE_DMA_ALT_CONFIG_REG_5
		0x260	VH_SIMPLE_DMA_ALT_SOURCE_REG_6
		0x264	VH_SIMPLE_DMA_ALT_DESTINATION_REG_6
		0x268	VH_SIMPLE_DMA_ALT_LENGTH_REG_6
		0x26C	VH_SIMPLE_DMA_ALT_CONFIG_REG_6
		0x270	VH_SIMPLE_DMA_ALT_SOURCE_REG_7
		0x274	VH_SIMPLE_DMA_ALT_DESTINATION_REG_7
		0x278	VH_SIMPLE_DMA_ALT_LENGTH_REG_7
		0x27C	VH_SIMPLE_DMA_ALT_CONFIG_REG_7
		0x400	VH_SIMPLE_DMA_ALT_ENABLE_REG
		0x404	VH_SIMPLE_DMA_IRQ_STATUS_CLEAR_REG
		0x408	VH_SIMPLE_DMA_IRQ_MASK_REG
		0x40C	VH_SIMPLE_DMA_TEST_FIFO_RESP_STATUS_REG
		0x410	VH_SIMPLE_DMA_SOFT_INT_REG

MODULE	BASE	OFFSET	REG NAME
Arm7tdmiscache	0x8010_5000 0x8010_50FF	0x00	ARM_SYSCREG_RESET_STATUS
		0x04	ARM_SYSCREG_GENERAL
		0x08	ARM_SYSCREG_CACHABLE_AREA_INDICATORS
		0x0C	ARM_SYSCREG_LINE_READS
		0x10	ARM_SYSCREG_LINE_WRITES
		0x14	ARM_SYSCREG_WRITE_MISSES
		0x18	ARM_SYSCREG_VIRTUAL_ADDRESS_PAGE0
		0x1C	ARM_SYSCREG_VIRTUAL_ADDRESS_PAGE1
		0x20	ARM_SYSCREG_VIRTUAL_ADDRESS_PAGE2
		0x24	ARM_SYSCREG_VIRTUAL_ADDRESS_PAGE3
		0x28	ARM_SYSCREG_VIRTUAL_ADDRESS_PAGE4
		0x2C	ARM_SYSCREG_VIRTUAL_ADDRESS_PAGE5
		0x30	ARM_SYSCREG_VIRTUAL_ADDRESS_PAGE6
		0x34	ARM_SYSCREG_VIRTUAL_ADDRESS_PAGE7
		0x38	ARM_SYSCREG_VIRTUAL_ADDRESS_PAGE8
		0x3C	ARM_SYSCREG_VIRTUAL_ADDRESS_PAGE9
		0x40	ARM_SYSCREG_VIRTUAL_ADDRESS_PAGE10
		0x44	ARM_SYSCREG_VIRTUAL_ADDRESS_PAGE11
		0x48	ARM_SYSCREG_VIRTUAL_ADDRESS_PAGE12
		0x4C	ARM_SYSCREG_VIRTUAL_ADDRESS_PAGE13
		0x50	ARM_SYSCREG_VIRTUAL_ADDRESS_PAGE14
		0x54	ARM_SYSCREG_VIRTUAL_ADDRESS_PAGE15
		0x58	ARM_SYSCREG_CLK_GATE_ENABLE
		0x5C	ARM_SYSCREG_SYNCHRONOUS_UNEQUAL_CLOCKS
		0x60	ARM_SYSCREG_ARM_HIGH_SPEED
		0x64	ARM_SYSCREG_ARM_CLK_SPEED_BOOST

Table 11 PNX0101 AHB2VPB3 Peripheral Address Map

MODULE	BASE	OFFSET	REG NAME
CFG_SAI1	0x8020_0000 0x8020_007F	0x00	VH_SAI_LEFT_16BIT
		0x04	VH_SAI_RIGHT_16BIT
		0x08	VH_SAI_LEFT_24BIT
		0x0C	VH_SAI_RIGHT_24BIT
		0x10	VH_SAI_INT_STATUS
		0x14	VH_SAI_INT_MASK
		0x20	VH_SAI_LEFT_32BIT_0
		0x24	VH_SAI_LEFT_32BIT_1
		0x28	VH_SAI_LEFT_32BIT_2
		0x2C	VH_SAI_LEFT_32BIT_3
		0x30	VH_SAI_LEFT_32BIT_4
		0x34	VH_SAI_LEFT_32BIT_5
		0x38	VH_SAI_LEFT_32BIT_6
		0x3C	VH_SAI_LEFT_32BIT_7
		0x40	VH_SAI_RIGHT_32BIT_0
		0x44	VH_SAI_RIGHT_32BIT_1
		0x48	VH_SAI_RIGHT_32BIT_2
		0x4C	VH_SAI_RIGHT_32BIT_3
		0x50	VH_SAI_RIGHT_32BIT_4
		0x54	VH_SAI_RIGHT_32BIT_5
		0x58	VH_SAI_RIGHT_32BIT_6
		0x5C	VH_SAI_RIGHT_32BIT_7
		0x60	VH_SAI_INTERLEAVED_0
		0x64	VH_SAI_INTERLEAVED_1
		0x68	VH_SAI_INTERLEAVED_2
		0x6C	VH_SAI_INTERLEAVED_3
		0x70	VH_SAI_INTERLEAVED_4
		0x74	VH_SAI_INTERLEAVED_5
		0x76	VH_SAI_INTERLEAVED_6
		0x78	VH_SAI_INTERLEAVED_7

MODULE	BASE	OFFSET	REG NAME
CFG_SAI2	0x8020_0080 0x8020_00FF	0x00	VH_SAI_LEFT_16BIT
		0x04	VH_SAI_RIGHT_16BIT
		0x08	VH_SAI_LEFT_24BIT
		0x0C	VH_SAI_RIGHT_24BIT
		0x10	VH_SAI_INT_STATUS
		0x14	VH_SAI_INT_MASK
		0x20	VH_SAI_LEFT_32BIT_0
		0x24	VH_SAI_LEFT_32BIT_1
		0x28	VH_SAI_LEFT_32BIT_2
		0x2C	VH_SAI_LEFT_32BIT_3
		0x30	VH_SAI_LEFT_32BIT_4
		0x34	VH_SAI_LEFT_32BIT_5
		0x38	VH_SAI_LEFT_32BIT_6
		0x3C	VH_SAI_LEFT_32BIT_7
		0x40	VH_SAI_RIGHT_32BIT_0
		0x44	VH_SAI_RIGHT_32BIT_1
		0x48	VH_SAI_RIGHT_32BIT_2
		0x4C	VH_SAI_RIGHT_32BIT_3
		0x50	VH_SAI_RIGHT_32BIT_4
		0x54	VH_SAI_RIGHT_32BIT_5
		0x58	VH_SAI_RIGHT_32BIT_6
		0x5C	VH_SAI_RIGHT_32BIT_7
		0x60	VH_SAI_INTERLEAVED_0
		0x64	VH_SAI_INTERLEAVED_1
		0x68	VH_SAI_INTERLEAVED_2
		0x6C	VH_SAI_INTERLEAVED_3
		0x70	VH_SAI_INTERLEAVED_4
		0x74	VH_SAI_INTERLEAVED_5
		0x76	VH_SAI_INTERLEAVED_6
		0x78	VH_SAI_INTERLEAVED_7

MODULE	BASE	OFFSET	REG NAME
CFG_SAI3	0x8020_0100 0x8020_017F	0x00	VH_SAI_LEFT_16BIT
		0x04	VH_SAI_RIGHT_16BIT
		0x08	VH_SAI_LEFT_24BIT
		0x0C	VH_SAI_RIGHT_24BIT
		0x10	VH_SAI_INT_STATUS
		0x14	VH_SAI_INT_MASK
		0x20	VH_SAI_LEFT_32BIT_0
		0x24	VH_SAI_LEFT_32BIT_1
		0x28	VH_SAI_LEFT_32BIT_2
		0x2C	VH_SAI_LEFT_32BIT_3
		0x30	VH_SAI_LEFT_32BIT_4
		0x34	VH_SAI_LEFT_32BIT_5
		0x38	VH_SAI_LEFT_32BIT_6
		0x3C	VH_SAI_LEFT_32BIT_7
		0x40	VH_SAI_RIGHT_32BIT_0
		0x44	VH_SAI_RIGHT_32BIT_1
		0x48	VH_SAI_RIGHT_32BIT_2
		0x4C	VH_SAI_RIGHT_32BIT_3
		0x50	VH_SAI_RIGHT_32BIT_4
		0x54	VH_SAI_RIGHT_32BIT_5
		0x58	VH_SAI_RIGHT_32BIT_6
		0x5C	VH_SAI_RIGHT_32BIT_7
		0x60	VH_SAI_INTERLEAVED_0
		0x64	VH_SAI_INTERLEAVED_1
		0x68	VH_SAI_INTERLEAVED_2
		0x6C	VH_SAI_INTERLEAVED_3
		0x70	VH_SAI_INTERLEAVED_4
		0x74	VH_SAI_INTERLEAVED_5
		0x76	VH_SAI_INTERLEAVED_6
		0x78	VH_SAI_INTERLEAVED_7

MODULE	BASE	OFFSET	REG NAME
CFG_SAI4	0x8020_0180 0x8020_01FF	0x00	VH_SAI_LEFT_16BIT
		0x04	VH_SAI_RIGHT_16BIT
		0x08	VH_SAI_LEFT_24BIT
		0x0C	VH_SAI_RIGHT_24BIT
		0x10	VH_SAI_INT_STATUS
		0x14	VH_SAI_INT_MASK
		0x20	VH_SAI_LEFT_32BIT_0
		0x24	VH_SAI_LEFT_32BIT_1
		0x28	VH_SAI_LEFT_32BIT_2
		0x2C	VH_SAI_LEFT_32BIT_3
		0x30	VH_SAI_LEFT_32BIT_4
		0x34	VH_SAI_LEFT_32BIT_5
		0x38	VH_SAI_LEFT_32BIT_6
		0x3C	VH_SAI_LEFT_32BIT_7
		0x40	VH_SAI_RIGHT_32BIT_0
		0x44	VH_SAI_RIGHT_32BIT_1
		0x48	VH_SAI_RIGHT_32BIT_2
		0x4C	VH_SAI_RIGHT_32BIT_3
		0x50	VH_SAI_RIGHT_32BIT_4
		0x54	VH_SAI_RIGHT_32BIT_5
		0x58	VH_SAI_RIGHT_32BIT_6
		0x5C	VH_SAI_RIGHT_32BIT_7
		0x60	VH_SAI_INTERLEAVED_0
		0x64	VH_SAI_INTERLEAVED_1
		0x68	VH_SAI_INTERLEAVED_2
		0x6C	VH_SAI_INTERLEAVED_3
		0x70	VH_SAI_INTERLEAVED_4
		0x74	VH_SAI_INTERLEAVED_5
		0x76	VH_SAI_INTERLEAVED_6
		0x78	VH_SAI_INTERLEAVED_7

MODULE	BASE	OFFSET	REG NAME
CFG_SAO1	0x8020_0200 0x8020_027F	0x00	VH_SAO_LEFT_16BIT
		0x04	VH_SAO_RIGHT_16BIT
		0x08	VH_SAO_LEFT_24BIT
		0x0C	VH_SAO_RIGHT_24BIT
		0x10	VH_SAO_INT_STATUS
		0x14	VH_SAO_INT_MASK
		0x20	VH_SAO_LEFT_32BIT_0
		0x24	VH_SAO_LEFT_32BIT_1
		0x28	VH_SAO_LEFT_32BIT_2
		0x2C	VH_SAO_LEFT_32BIT_3
		0x30	VH_SAO_LEFT_32BIT_4
		0x34	VH_SAO_LEFT_32BIT_5
		0x38	VH_SAO_LEFT_32BIT_6
		0x3C	VH_SAO_LEFT_32BIT_7
		0x40	VH_SAO_RIGHT_32BIT_0
		0x44	VH_SAO_RIGHT_32BIT_1
		0x48	VH_SAO_RIGHT_32BIT_2
		0x4C	VH_SAO_RIGHT_32BIT_3
		0x50	VH_SAO_RIGHT_32BIT_4
		0x54	VH_SAO_RIGHT_32BIT_5
		0x58	VH_SAO_RIGHT_32BIT_6
		0x5C	VH_SAO_RIGHT_32BIT_7
		0x60	VH_SAO_INTERLEAVED_0
		0x64	VH_SAO_INTERLEAVED_1
		0x68	VH_SAO_INTERLEAVED_2
		0x6C	VH_SAO_INTERLEAVED_3
		0x70	VH_SAO_INTERLEAVED_4
		0x74	VH_SAO_INTERLEAVED_5
		0x78	VH_SAO_INTERLEAVED_6
		0x7c	VH_SAO_INTERLEAVED_7

MODULE	BASE	OFFSET	REG NAME
CFG_SAO2	0x8020_0280 0x8020_02FF	0x00	VH_SAO_LEFT_16BIT
		0x04	VH_SAO_RIGHT_16BIT
		0x08	VH_SAO_LEFT_24BIT
		0x0C	VH_SAO_RIGHT_24BIT
		0x10	VH_SAO_INT_STATUS
		0x14	VH_SAO_INT_MASK
		0x20	VH_SAO_LEFT_32BIT_0
		0x24	VH_SAO_LEFT_32BIT_1
		0x28	VH_SAO_LEFT_32BIT_2
		0x2C	VH_SAO_LEFT_32BIT_3
		0x30	VH_SAO_LEFT_32BIT_4
		0x34	VH_SAO_LEFT_32BIT_5
		0x38	VH_SAO_LEFT_32BIT_6
		0x3C	VH_SAO_LEFT_32BIT_7
		0x40	VH_SAO_RIGHT_32BIT_0
		0x44	VH_SAO_RIGHT_32BIT_1
		0x48	VH_SAO_RIGHT_32BIT_2
		0x4C	VH_SAO_RIGHT_32BIT_3
		0x50	VH_SAO_RIGHT_32BIT_4
		0x54	VH_SAO_RIGHT_32BIT_5
		0x58	VH_SAO_RIGHT_32BIT_6
		0x5C	VH_SAO_RIGHT_32BIT_7
		0x60	VH_SAO_INTERLEAVED_0
		0x64	VH_SAO_INTERLEAVED_1
		0x68	VH_SAO_INTERLEAVED_2
		0x6C	VH_SAO_INTERLEAVED_3
		0x70	VH_SAO_INTERLEAVED_4
		0x74	VH_SAO_INTERLEAVED_5
		0x78	VH_SAO_INTERLEAVED_6
		0x7c	VH_SAO_INTERLEAVED_7
AUDIOSS_CREG	0x8020_0380 0x8020_03FF	0x00	VH_AUDIOSS_CREG_IIS_FORMAT_SETTINGS
		0x04	VH_AUDIOSS_CREG_AUDIOSS_MUX_SETTINGS
		0x08	VH_AUDIOSS_CREG_SPDIF_IN_STATUS
		0x0C	VH_AUDIOSS_CREG_SPDIF_IN_IRQ_EN
		0x10	VH_AUDIOSS_CREG_SPDIF_IN_IRQ_STATUS
		0x14	VH_AUDIOSS_CREG_SPDIF_IN_IRQ_CLEAR
		0x18	VH_AUDIOSS_CREG_SDAC_CTRL_INTI
		0x1C	VH_AUDIOSS_CREG_SDAC_CTRL_INTO
		0x20	VH_AUDIOSS_CREG_SDAC_SETTINGS
		0x24	VH_AUDIOSS_CREG_SADC_CTRL_SDC
		0x28	VH_AUDIOSS_CREG_SADC_CTRL_ADC
		0x2C	VH_AUDIOSS_CREG_SADC_CTRL_DECI
		0x30	VH_AUDIOSS_CREG_SADC_CTRL_DECO
		0x34	VH_AUDIOSS_CREG_E7B_INTERRUPT_REQ
		0x38	VH_AUDIOSS_CREG_E7B_AHB_SETTINGS
		0x3C	VH_AUDIOSS_CREG_N_SOF_COUNTER

12 ARM7TDMI MICROCONTROLLER

12.1 Overview

ARM7TDMI is a general-purpose 32-bit microprocessor. The ARM7TDMI is software compatible with the ARM processor family.

ARM7TDMI is a fully static part and has been designed to minimize power requirements. This makes it ideal for portable applications, where both these features are essential.

The ARM7TDMI architecture is based on Reduced Instruction Set Computer (RISC) principles, and the instruction set and related decode mechanism are greatly simplified compared with micro programmed Complex Instruction Set Computers (CISC).

More information regarding the ARM7 and its instructions can be retreived from the ARM website, HTTP://WWW.ARM.COM

12.2 ARM7TDMI: The THUMB Concept

The ARM7TDMI processor employs a unique architectural strategy known as THUMB, which makes it ideally suited to high-volume applications with memory restrictions, or applications where code density is an issue.

The THUMB instruction set is a subset of the ARM instruction set. The THUMB is designed to increase the performance of ARM implementations that uses a 16-bit memory data bus, and may allow better code density than ARM instruction set.

The THUMB set's 16-bit instruction length allows it to approach twice the density of standard ARM code while retaining most of the ARM's performance advantage over a traditional 16-bit processor using 16-bit registers. This is possible because THUMB code operates on the same 32-bit register set as ARM code.

THUMB code is able to provide up to 65% of the code size of ARM, and 160% of the performance of an equivalent ARM processor connected to a 16-bit memory system.
All internal memories of the PNX0101 are 32-bit, only when executing via EBI the accesses are 16-bit.

When ARM instruction caching is enabled the performance advantage might be different, since the cache is 32 bit wide. The slower speed of the 16-bit bus will only be noticed during cache misses. However, more THUMB code fits into the internal cache, reducing the cache-miss ratio which may improve execution speed.
This can be proven by using different 'builds', with one THUMB build, and a normal build, and compare these two solutions in a system benchmark.

12.3 Cache Architecture

The ARM7TDMI-S inside the PNX0101 has been extended with a 2-way set-associative cache controller. The cache is 8Kbyte wide and can store both data and instruction code.

The biggest benefit of having this cache, is that if code is run from non-zero-waitstate memory, like for instance the internal FLASH controller, that these memories still behalve like they are almost zero-waitstate memory.
If code is executed from the cache, the ARM will run at 1 instruction/clockcycle for most of the time.

The tradeoff introducing this cache is that each AHB access that bypasses the cache will have an extra waitstate inserted. Therefore it is advised to always enable instruction caching (and preferably data caching as well) for all memories, to allow zero-waitstate behaviour which gives the ARM the highest operational speed.

12.3.1 Cache description

This cache works as follows, for each page of which the cache is enabled:

- If data is read, and not in cache (cache MISS), a line of eight 32-bit words will be read from the AHB bus. In the mean time, the ARM is 'idle' (and in low power if the 'clock gating' is enabled.)
- If data is read, and is found in cache, (cache HIT) data is read from cache, with 0 ws access
- If data is written, and the location is not in cache(cache MISS), the data will be written directly to memory.
- If data is written, and the location is in cache, because this location has been read before (cache HIT), then data is written in cache with 0 ws, and the line is marked as 'dirty'.
- If a 'dirty' line is about to be discarded because of a cache-miss on a total different memory area, this line will first be written back to memory. (cache-line FLUSH)

The cache can be set to data-only, instruction-only or combined (unified) caching. The ARM7+cache as used in the PNX0101 has 16 configurable pages, each being 2 Megabyte wide.
Each page can be configured as follows:

- The virtual address.
- Cache enabled / disabled.

These 16 pages occupy the bottom 32 Megabyte of the memory map.

The whole 32-bit address range can be addressed inside the bottom 32 Megabytes of memory. This allows memories to be placed outside the first 32 Megabyte limit, and still be cachable.
Every access outside the first 32 Megabytes will be directly read or written from/to that location bypassing the cache (and introducing a 1-waitstate performance loss for each transfer).

Cache flushing is performed by filling the cache with 8 KByte read-only data (for instance ROM data). In hardware this results in that every cache-line is checked if it is dirty, and if so, writes this data towards memory.

12.4 Software Interface Specification

This chapter is about the software settings of the ARM, and how to configure the cache.

12.4.1 PROGRAMMING

12.4.1.1 ARM cache registers.

This part describes the register settings of the cache controller.
The cache controller is configured by one of the PNX0101 VPB interfaces.

The VPB interface of the ARM configuration hardware must run at the same BASE_CLK frequency as the AHB interface of the ARM, before any register is written. This is true for PNX0101.

All registers are static, and will be unaltered by the cache interface. It is therefore adviced to enable external clock enabling (=clock-gating) option in the CGU for the VPB interface of the ARM, to reduce powerconsumption.

Table 12 ARM7TDMIS + cache register map

ADDRESS SPACE	OFFSET	REGISTER NAME	R/W ACCESS	RESET
<ARM base address>	0x00	cache_reset_status	R	0x0
	0x04	general_cache_settings	R/W	0x0
	0x08	cacheable_area_indicators	R/W	0x0
	0x0C	nr_of_line_reads	R	0x0
	0x10	nr_of_line_writes	R	0x0
	0x14	nr_of_write_misses	R	0x0
	0x18	virtual_address_page0	R/W	0x0
Note: SROM	0x1C	virtual_address_page1	R/W	0x1
Note: SRAM	0x20	virtual_address_page2	R/W	0x2
Note: SRAM	0x24	virtual_address_page3	R/W	0x2
Note: eFLASH	0x28	virtual_address_page4	R/W	0x82
Note: External SRAM	0x2C	virtual_address_page5	R/W	0x100
Note: External SRAM	0x30	virtual_address_page6	R/W	0x100
Note: External SDRAM	0x34	virtual_address_page7	R/W	0x180
Note: External SDRAM	0x38	virtual_address_page8	R/W	0x180
Note: VPB0, VPB1, VPB2, PL172_CFG	0x3C	virtual_address_page9	R/W	0x400
Note: VPB3, MMIO_INTC	0x40	virtual_address_page10	R/W	0x401
	0x44	virtual_address_page11	R/W	0x102
	0x48	virtual_address_page12	R/W	0x104
	0x4C	virtual_address_page13	R/W	0x106
	0x50	virtual_address_page14	R/W	0xE
	0x54	virtual_address_page15	R/W	0xF
	0x58	clk_gate_enable	R/W	0x0
	0x5C	synchronous_unequal_clocks	R/W	0x0
	0x60	arm_high_speed	R/W	0x0
	0x64	Clock_speed_boost	R/W	0x0

cache_reset_status

If the ‘reset_cache’ bit is first set and then unset inside the ‘general_cache_settings’ register this bit indicates the status of the ongoing reset.

‘0’ - The cache reset is finished
‘1’ - The cache reset is ongoing. Its best for the ARM to poll this signal until it is ‘0’

The reset of the cache TAG memory will take 128 ARM clock-cycles to complete.

General_cache_settings

Table 13 General_cache_settings

BIT	LABEL	RESET	R/W
0	cache_reset	0x0	R/W
1	cache_data_enable	0x0	R/W
2	cache_instruction_enable	0x0	R/W
3	cache_performance_analysis_reset	0	R/W
4	cache_performance_analysis_enabled	000	R/W

Cache_reset

This bit resets the cache. Hardware internally, the 128*32 tag memory is pre-programmed to value ‘0x0’ , so that the whole cache is assigned ‘empty’. This takes about 128 clockcycles to complete.
This reset bit MUST be set before the cache may be enabled. It is recommended to include this procedure at startup of the system.

The reset progress can be followed by reading register ‘cache_reset_status’.

Reset procedure: First program this bit to ‘1’, then to ‘0’.

Cache_data_enable

This bit enables that CPU data can be stored in cache. This data will be shared with the instruction cache if the ‘cache_instruction_enable’ bit is set.
Please read the extra’s involved with setting this bit, in the ‘cache procedures’ chapter.

‘0’ - Data caching disabled. This involves all 16 pages.
‘1’ - Data caching enabled. This involves all 16 pages.

Note: In register ‘Cachable_area_indicators’ each page can be individually enabled or disabled.

Cache_instruction_enable

This bit enables that CPU instructions can be stored in cache. This data will be shared with data cache, if data caching is enabled. Since instructions are read_only (assuming there is no self-generating-code), no cache flushing is required if ‘data_cache_enable’ is ‘0’.

‘0’ - Instruction caching disabled. This involves all 16 pages.
‘1’ - instruction caching enabled. This involves all 16 pages.

Note: In register ‘Cachable_area_indicators’ each page can be individually enabled or disabled.

Cache_performance_analysis_reset

This bit resets the following registers to ‘0x0’:

‘nr_of_line_reads’
‘nr_of_line_writes’
‘nr_of_write_misses’

For this to work, bit ‘cache_performance_analysis_enabled’ has to be set to ‘1’

Cache_performance_analysis_enabled

When this bit is enabled, software can read from the cache_analysis registers:

‘nr_of_line_reads’
‘nr_of_line_writes’
‘nr_of_write_misses’

Its advisable that when cache analysis is not required, to keep this bit disabled. This reduces powerconsumption the cache system.

‘0’ - Cache performance analysis is disabled. Powersaving mode.
‘1’ - Cache performance analysis is enabled.

Cacheable_area_indicators

Table 14 Cacheable area indicators register

BIT	LABEL	RESET	R/W
0	enable cache for page 0	0	R/W
1	enable cache for page 1	0	R/W
2	enable cache for page 2	0	R/W
3	enable cache for page 3	0	R/W
4	enable cache for page 4	0	R/W
5	enable cache for page 5	0	R/W
6	enable cache for page 6	0	R/W
7	enable cache for page 7	0	R/W
8	enable cache for page 8	0	R/W
9	enable cache for page 9	0	R/W
10	enable cache for page 10	0	R/W
11	enable cache for page 11	0	R/W
12	enable cache for page 12	0	R/W
13	enable cache for page 13	0	R/W
14	enable cache for page 14	0	R/W
15	enable cache for page 15	0	R/W

Each bit of this 16-bit register represents the enabling or disabling of one page of cache.
If data caching is enabled, and the programmer wants to disable the page of a writable memory location, he or she must be aware that there might be dirty data in the cache that still needs to be written to memory.

Nr_of_line_reads

This register can be used to perform cache performance analysis. Cache performance analysis inside the general register must be enabled for this to function.

This register gives the number of times a line is read from memory and also represents the number of cache read misses.

Nr_of_line_writes

This register can be used to perform cache performance analysis. Cache performance analysis inside the general register must be enabled for this to function.

This register gives the number of times a complete cache line is written back to memory (this will only happen if some of the contents of the line changed)

Nr_of_write_misses

This register can be used to perform cache performance analysis. Cache performance analysis inside the general register must be enabled for this to function.

This register reads the number of times that a write happens to an address that is not in the cache.

The value will replace the top 11 bits of the 32-bit address, to create a REAL address.

Virtual_address_page0

The value inside this register and the following 15 registers will replace the top 11 bits of the 32-bit address, to create a REAL address.
So when the ARM performs an access to this page, the address which will be placed on the AHB bus will depend on the value of this register.

This value will remap address ‘0x0 to 0x200000’ to any address, in steps of 2 megabytes.

Virtual_address_page1

This value will remap address ‘0x200000 to 0x400000’ to any address, in steps of 2 megabytes.

Virtual_address_page2

This value will remap address ‘0x400000 to 0x600000’ to any address, in steps of 2 megabytes.

Virtual_address_page3

This value will remap address ‘0x600000 to 0x800000’ to any address, in steps of 2 megabytes.

Virtual_address_page4

This value will remap address ‘0x800000 to 0xA00000’ to any address, in steps of 2 megabytes.

Virtual_address_page5

This value will remap address ‘0xA00000 to 0xC00000’ to any address, in steps of 2 megabytes.

Virtual_address_page6

This value will remap address ‘0xC00000 to 0xE00000’ to any address, in steps of 2 megabytes.

Virtual_address_page7

This value will remap address ‘0xE00000 to 0x1000000’ to any address, in steps of 2 megabytes.

Virtual_address_page8

This value will remap address ‘0x1000000 to 0x1200000’ to any address, in steps of 2 megabytes.

Virtual_address_page9

This value will remap address ‘0x1200000 to 0x1400000’ to any address, in steps of 2 megabytes.

Virtual_address_page10

This value will remap address ‘0x1400000 to 0x1600000’ to any address, in steps of 2 megabytes.

Virtual_address_page11

This value will remap address ‘0x1600000 to 0x1800000’ to any address, in steps of 2 megabytes.

Virtual_address_page12

This value will remap address ‘0x1800000 to 0x1A00000’ to any address, in steps of 2 megabytes.

Virtual_address_page13

This value will remap address ‘0x1A00000 to 0x1C00000’ to any address, in steps of 2 megabytes.

Virtual_address_page14

This value will remap address ‘0x1C00000 to 0x1E00000’ to any address, in steps of 2 megabytes.

Virtual_address_page15

This value will remap address ‘0x1E00000 to 0x2000000’ to any address, in steps of 2 megabytes.

Clk_gate_enable

Setting this bit to ‘1’ makes the ARM more power efficient, because the clock of the pure ARM core is switched off when it is waiting for bus-access.
Synphony N1A specific: When this bit is enabled, **it is not possible to use the multi-ice**!

‘0’ - The clock is free running (default)
‘1’ - The ARM clock is in power-efficient mode.

Synchronous_unequal_clocks

This bit must be set when the ARM + cache has to run at a higher clock frequency then the AHB bus.
This bit must be set **BEFORE** the ‘clock_speed_boost’ register can be set.

‘0’ - The ARM runs synchronous and with the same clock with the AHB bus. (default)
‘1’ - The ARM is allowed to run higher then the AHB bus, using the ‘clock_speed_boost’ register.

Arm_high_speed

This bit may improve the timing of the ARM+cache design, when register ‘clk_gate-enable’ is set. More details will be added in a later stage.

‘0’ - The ARM can run at maximum 60 Mhz (85 degrees celcius)
‘1’ - The ARM can run at maximum ??? Mhz. (to be corrected)

Clock_speed_boost

This register can be used to add extra clock speed to the ARM. The ARM can run 25% faster then the AHB bus, up to 32 times faster, depending the settings of the ‘clock_speed_boost’ register.

The register ‘Synchronous unequal clocks’ must be set first, before this register may be used.

Table 15 Clock speed boost register

BIT	LABEL	RESET	R/W
4-0	Clock_multiply	0	R/W
6-5	Clock_devide	0	R/W
7	Allow_double_base_clk <reserved N1A>	0	R/W

Clock_multiply

Setting this register higher then 0x0 results in multiplying the ARM clock relative to the AHB_reference_clock_enable signal.

Clock_devide

Setting this register higher then 0x0 results in a devision of the multiplied clock.
‘Clock_multiply’ needs to be higher then 0x0, else these bits will have no effect.

Allow_double_base_clk

By setting this bit, There will be a 1 clock-cycle gap in between added clocks, allowing the AHB-base_clock to run twice the speed then the maximum speed of the ARM+cache system (see ‘arm_high_speed’ register).

Setting this bit devides the ‘clock_multiply’ by 2, rounded downwards.

The resulting clock speed can be calculated with the following formula:

If Allow_double_base_clk = 0 :
    Multiplication = (‘clock_multiply’ / ‘clock_devide + 1’)) + 1;
else
    Multiplication = (‘clock_multiply >>1’ / ‘clock_devide + 1’) +1 ; // Clock_multiply shifted 1 bit to the right

12.4.1.2 Cache programming procedures.

CACHE INITIALIZATION

- 1: Clear the cache:

  Set and reset the ’cache_reset’ signal (one clock cycle is long enough)

  The signal ’cache_reset_status’ indicates if a cache reset is ongoing, so the CPU has to poll this bit before the cache may be enabled.

- 2: Program the cacheable memory map:

  Now you enable those parts in the memory map that are allowed to end up in the cache, using the ’cacheable_area_indicators’ register.
  Each bit represents one page and 2Mbyte of memory space:

  bit 0 enables 0x0 to 0x200000 as cached (page 0),
  bit 1 enables 0x200000 to 0x400000 as cached (page 1),
  bit 2 enables 0x400000 to 0x600000 as cached (page 2),
  etc etc.

- 3: Program the virtual address for each page.

  The 11 bits programmed for each page, represents the top 11 bits of a 32-bit address that will be put on the AHB bus. So the whole 32 bit address range can be remapped inside the bottom 32 megabytes, in blocks/pages of 2 Megabyte.

  Example:

  Say address location 0x10400000 (eFLASH)must be mapped for page 3. That can be done this way::

  **virtual_address_page3 = (0x10400000 >> 21); // = 0x082;*

  Now, when the ARM reads address 0x00600004 (an address inside page 1) , then address 0x10400004 will be read on the AHB bus.

  Note: BE CAREFULL about remapping the page where the current code is run from, or the page that is used for R/W, stack or heap storage.

- 4: Enable the cache for data and/or instructions

  Enable the cache by setting (active high) signals ’cache_data_enable’ and/or ’cache_instruction_enable’.

  These two bits effect all 16 pages.

  So the whole cache can be programmed to:
  only instruction-cache,
  only data-cache,
  both enabled, to form a unified cache.

  If none of these two signals are set, nothing will end up in the cache.

CACHE FLUSHING

Cache flushing can be required if data_caching is enabled, or when a virtual address a page must be changed while this page is cache-enabled.
If data is written to cached memory, it may end up inside the cache, marked as dirty. If data is written in cache, this data is not yet stored on its physical location. So if another AHB busmaster like the SDMA is programmed to copy this data, it will copy the old data, not the new data which is inside cache.

If the programer wants to guarantee that the data inside the cache is written to memory, the programmer has to ‘flush’ the cache.

There is no hardware register or CPU_instruction to flush the cache.
The cache can only be flushed by filling up the cache with 8 Kbyte of read-only data from a cached-enabled page.
Important: Cache flushing is only possible if data-caching is enabled!

Only 1 out of 8 words from each cache line will have to be read to flush one cache line. A total of 256 cache lines has to be read to fully flush the cache. Here is a C-example to replace the cache contents, and flush its contents. Cache data enabling must be enabled for this to work.

```
void flush_cache (int * cache_start) {

volatile int * flush_pointer = (volatile int *)cache_start;
volatile int cache_dummy;
int i;

    for (i=0;i<2048;i+=8) cache_dummy=flush_pointer[i];
}
```

Example: Calling ‘flush_cache((int *) 0x1200);’ will read 8 kbytes of read-only code starting from 0x1200 into the cache, effectively flushing all dirty data from the cache. Its advisable to cache the routine that is about to be used. This results in better cache performance for the next code lines.

Cache flushing is almost mandatory in the following cases:

- 1) When data_caching is enabled for a page, and another busmaster like the SDMA uses this data as well.
- 2) When data_caching is enabled for a page, and caching for this page is about to be disabled.
- 3) When data_caching in the ‘general_cache_setting’ register is about to be disabled
- 4) when the virtual address is about to be changed of a page that is cached. This goes for both instruction and data caching.

1)

If for instance the ARM writes audio-samples to a cached page, and the SDMA transfers this audio to the IIS output, then corrupt data might be send.

This is because some audio-samples might still be inside the cache, and not in memory. The SDMA fetches data from memory and knows nothing about the existance of a cache.

To be sure that correct data is always available for the SMDA, the cache has to be flushed before a DMA channel using this data may be enabled.

2)

If suddenly the caching for a page is disabled, then every single word is read directly from memory, bypasssing the cache. If there is still valid RW-data in the cache, then the ARM may read the wrong data.

To avoid this, the cache must be flushed right before or right after the page is disabled.

3)

The same reasons as (2) apply If data_caching is disabled in general.

4)

The cache is unaware of any changes made to the virtual address. If the virtual address is changed, the programmer must ensure the contents of the cache are flushed, also if code was executed from the page that is about to be remapped.

Flushing means not only that dirty data is written to memory, but to remove active cachelines from cache that may contain code of the old virtual address.

AVOIDING CACHE FLUSHING

In some cases it may be better to avoid cache flushing, especially if data is involded that is not ment to be stored in cache. Below are a couple of methods described to avoid cache flushes.

1) Do not enable data caching in the 'general_cache_setting'

The best way to avoid data flushing, is not to enable data caching in general. This may have a performance impact, and has to be benchmarked.
Note: Bechmarking using the ARM MP3 decoder indicate that disabling data caching has a big performance impact. Performance reductions of 20-30% were measured when data caching was disabled.

2) Have 2 pages reserved for the same memory location

A more delicate way to avoid data flushing this is to have 2 pages that point towards the same memory address. One page is set as cachable, the other page as not cachable. It doesn't matter to which page the data is written to, it will eventually end up at the same memory location.

The difference is that data written to the non-cached page is written directly to memory, so the SDMA can make use of this data, without software have to worry about flushing the cache.

Two precautions:

* Don't write data to one page, and read the same data from the other page.

* Make sure that cached data and non-cached data don't overlap the same cache-line.
Example: Say page 0 and 1 are remapped to the same virtual address, and cache is enabled for page 0.

Say cached data is written to 0x24 (page 0), and non-cache data is written to 0x200028 (page 1).

First data to 0x24 will be written into cache, then the other data is written directly to memory location 0x200028
After some time, the cache-line belonging to address 0x24 will be written back to memory, and overwriting the non-cached data becuase 8 words are written at once.

To avoid this, keep non-cached data and cached data a cache-line apart. This can be controlled at the 'linking' stage of the software.

3) Use larger data buffers

Lets take as example MP3 decoding. Mp3 decoding is frame based. After a frame is decoded, some audio-samples might stil remain in cache. But if another frame is decoded, it is almost certain that previous audio samples have been flushed into memory.

So if the SDMA is running at least more then one frame behind MP3 decoding, still correct samples will be written to the headphones, without the need for cache flushing, because the running application also flushes cache contents.

If the programmer is not certain about this, then please avoid this constrution.

12.5 ARM clocking and power optimizations

The ARM clocking is slightly different then the rest of the AHB system:
Where the rest of the AHB system is clocked by the CGU (the AHB-BASE_CLOCK devided by a ‘frac_div’), the ARM+cache system uses the AHB clock as a reference to generate its internal clocks from the AHB_BASE_CLOCK.

The ARM receives 2 clock from the CGU:

- ARM7TDMISCACHE_CLK is the clock the whole ARM+cache system runs on, and must always be configured in the CGU as a direct connection to the AHB_BASE_CLOCK, and this clock may not be switched by a fracdiv. The result is a free-running clock for the ARM.
- ARM7TDMISCACHE_DUMMY_HCLK is a reference clock, which is not used as a clock but as a ‘indicator’ of the AHB clock. This clock must be programmed with the same configuration settings as the ‘AHB0’ clock, and must have its ‘external enabling’ bit enabled inside the CGU switchbox PCR register.

Inside the cache system is a clock-gate that uses the ‘AHB-reference-clock’ signal to enable or disable the freerunning clock that clocks the whole ARM+cache system.
This way, the ARM+cache system always runs at the same clock as the AHB0 clock, no matter the settings of the AHB0 fractional devider.

A special piece of hardware called ‘clock_speed_boost’ can add extra enable-cycles for this clock-gate. It always passes the ‘AHB-reference-clock-enable’ signal, but it can add extra enable cycles so the ARM+Cache receives more clocks then the AHB0 bus.
The ‘boosting’ can be set in the ‘clock_speed_boost’ register. It’s internal counter is reset at each AHB0 clock-cycle

At the next page are a couple of timing pictures of different settings of the clock selection the ARM. Below is a description of each timing picture.

1)

In this picture there is no ARM clock-gating and the fractional devider for the HCLK is not selected.This results in a freerunning clock for the AHB, Cache and ARM, all running at its maximum frequency.This is the reset condition of the system (Assuming nothing has been done to the clock settings in ROM).

2)

In this picture the AHB fractional devider has been set to generate a clock once every 7 clock-cycles.
The CGU generates a corresponding signal, that ‘announces’ the AHB clock, one clockcycle in advance. The behaviour of this signal can be seen at signal ‘CACHE_CLKEN’, which is a logical ‘OR’ of the AHB_CLKEN signal from the CGU and the clock-speed-boosting.
Notice that the CACHE is always clocked when the AHB is clocked.

3)

In this picture, the ARM clock-enabling has been enabled inside the cache configuration block. The ARM only receives a clock when the signal ‘ARM_CLKEN’ signal is active.
This signal is generated by the cache, and means the ARM must wait because:
a) The cache is fetching/sending data from/to the AHB bus
b) The cache is jumping in between cache lines, which adds a single waitstate.

Enabling this option reduces the amount of clocks the ARM receives, therefore reduces the powerconsumption of the ARM.
Notice that the CACHE is always clocked when the AHB is clocked

4)

In this picture, the ARM_SPEED_BOOSTING is enabled, adding 3 cache clockcycles at every occurance of HCLK. Since division has been set to ‘1’, the adding will be skip every one occurance of HCLK. If this had been set to ‘3’, then this would have been skipped 3 times.

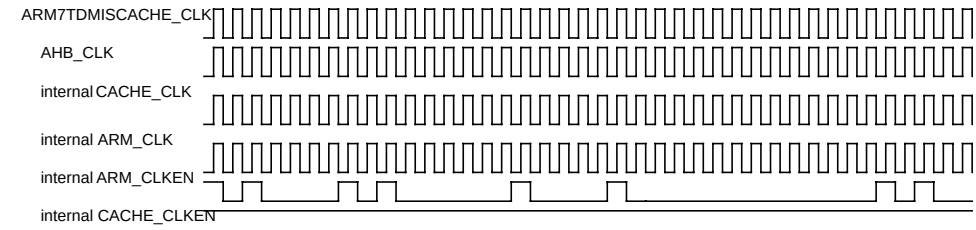
As long as cache-hits occur, the ARM can continue executing code from cache without having to suffer from a slower HCLK clock. This slower HCLK will have impact as soon as the Cache needs to fetch cache-lines from the AHB bus. The CACHE_ENABLE signal cosists of 2 parts, which are indicated in the picture:

1) The AHB_CLKEN signal which is always passed, and resets the ‘arm_speed_boost’ multiply counter
2) The enable signals generated by the clock_speed_boost hardware

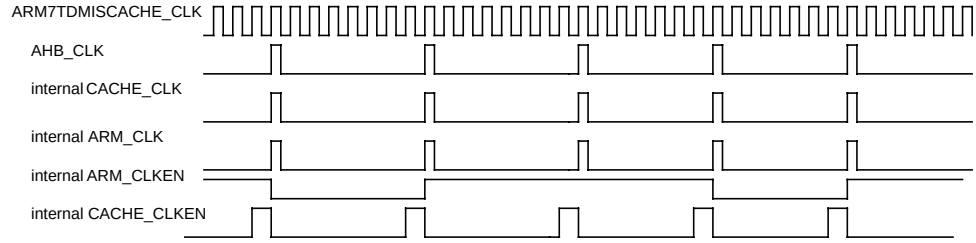
Notice that the CACHE is always clocked when the AHB is clocked.

5)

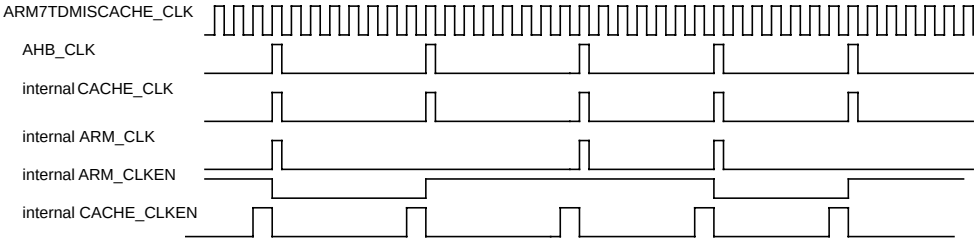
Same as (4) but now the boost-option: ‘allow-double_base_clk’ is enabled. This creates a single clock-cycle ‘gap’ in between the enabling which is added by the ‘clock_speed_boost’ hardware. So effectively, the cache always runs at least twice as slow as the base_clock, so ARM+cache timing can be met if the base_clock need to run faster then 64 Mhz.



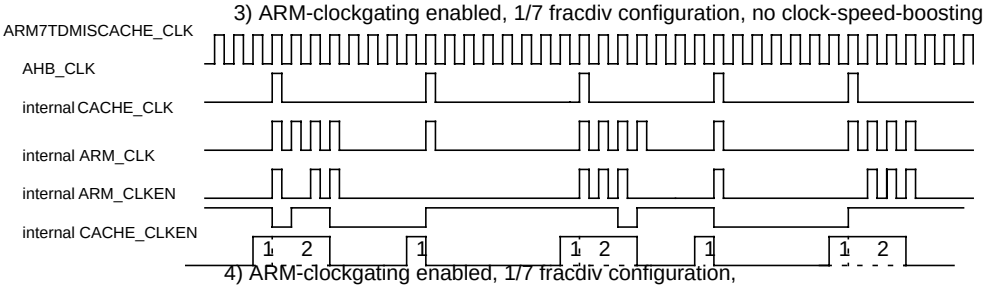
1) No ARM-clockgating, no fracdiv configuration, no clock-speed-boosting



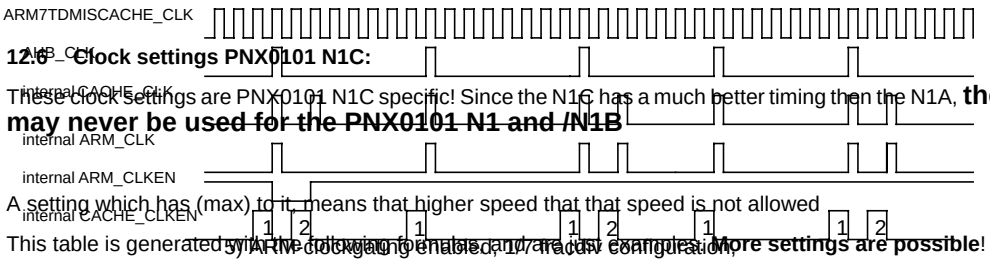
2) No ARM-clockgating, 1/7 fracdiv configuration, no clock-speed-boosting



3) ARM-clockgating enabled, 1/7 fracdiv configuration, no clock-speed-boosting



4) ARM-clockgating enabled, 1/7 fracdiv configuration,
clock-speed-boosting : Mul=3, div=1, allow_double_baseclk=0



5) ARM-clockgating enabled, 1/7 fracdiv configuration,
clock-speed-boosting : Mul=3, div=1, allow_double_baseclk=1

12.8 Clock settings PNX0101 N1C:

These clock settings are PNX0101 N1C specific! Since the N1C has a much better timing then the N1A, **these values may never be used for the PNX0101 N1 and /N1B**

A setting which has (max) to it, means that higher speed that that speed is not allowed

This table is generated with the following formulas and are just examples. **More settings are possible**!

BASE_CLK smaller or equal to 60Mhz:
Max arm_clock_speed_boost = fracdiv
BASE_CLK higher then 60 Mhz up to 120 Mhz:
Fracdivs must be programmed with an even factor (so 2,4,6,8,..)
Allow_double_base_clk = 1
Max arm_clock_speed_boost = fracdiv * 2

BASE_CLK	EFFECTIVE AHB FREQUENCY	EFFECTIVE ARM FREQUENCY	FRACDIV	ARM CLOCK SPEED BOOST	ALLOW DOUBLE BASE CLOCK	
12	12	12	-	0	0	Reset condition
36	12	36	1/3	3	0	
36	9	18	1/4	2	0	
36	9	36	1/4	4	0	
36	9	9	1/4	1	0	
48	12	48	1/4	4	0	
48	8	48	1/6	6	0	“
48	8	24	1/6	3	0	
60(max)	8.57	60 (max)	1/7	7	0	“
60(max)	10	50	1/6	5	0	
60(max)	30	30	1/2	1	0	
60(max)	10	20	1/6	2	0	
60(max)	60(max)	60(max)	1	1	0	Highest speed setting possible for the ARM and AHB
72	9	36 (max)	1/8(even!)	8(max)	1	>60 Mhz: Allow_double_base_clk = 1 Effective maximum ARM speed is half the BASE_CLK speed Only even fracdivs allowed
72	12	24	1/6(even!)	4	1	
84	14	42(max)	1/6(even!)	6	1	
84	21	42(max)	1/4(even!)	4	1	
84	10.5	31.5	1/8(even!)	6	1	
84	8.4	25.2	1/10(even!)	6	1	
96	24	48 (max)	1/4(even!)	4	1	
96	12	36	1/8(even!)	6	1	
96	8	16	1/12(even!)	4	1	
108	9	27	1/12(even!)	4	1	
108	27	54(max)	1/4(even!)	4 (max)	1	
120(max)	60(max)	60(max)	1/2	2	1	Maximum ARM, AHB and BASE_CLK speed allowed

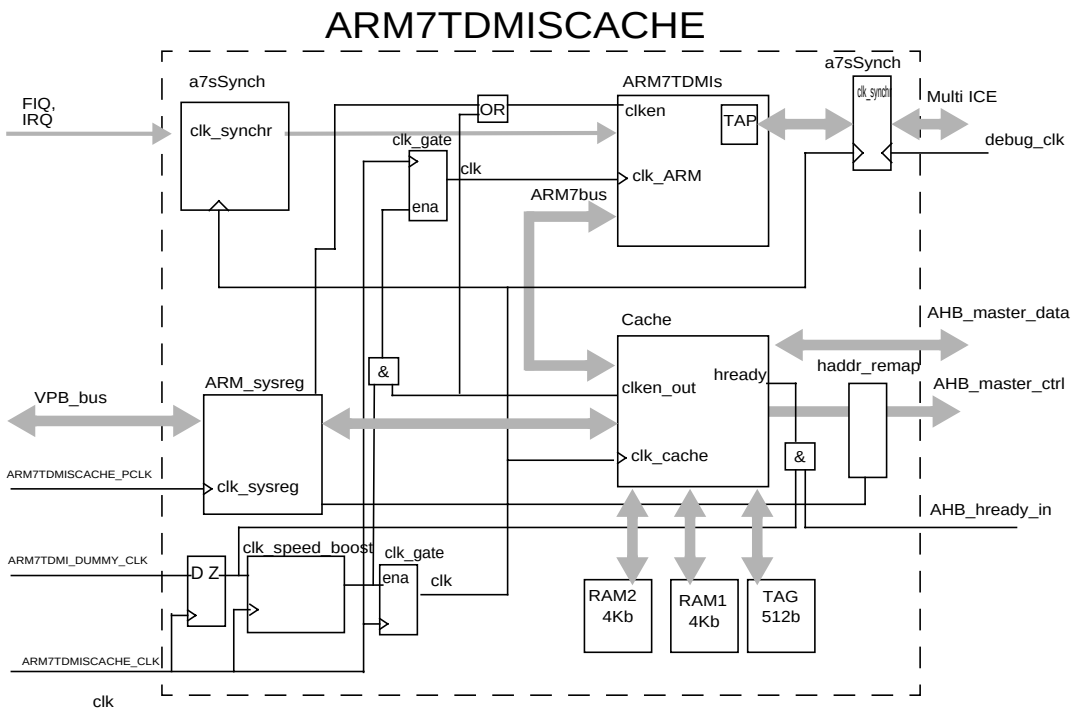


Fig. 3 ARM7TDMIS + Cache controller

Differences N1A and N1C regarding the ARM7TDMIScache

1) Changed the initial reset values of unmapped virtual pages. See the ARM configuration register map.
This allows code to be directly linked to one of these pages. These pages link to RAM, ROM, Eflash, External SRAM and external SDRAM. Software can ofcourse overwrite these values, which was required for the N1A anyway.

2) The layout of the N1C has a much better timing then the N1A. For the N1A there were ARM clock exceptions depending on the AHB clock where the ‘arm_clock _speed_boost’ could not be written with the maximum value. For the N1C, there are no more exceptions. Both the ARM and the AHB bus may run at 60 Mhz, which gives the PNX0101 more maximum available computational power.

13 MULTI-LAYER AHB

13.1 Overview

The Multi-layer AHB is an interconnection scheme, based on the AHB protocol that enables parallel access paths between multiple masters and slaves in a system. This is achieved by using a more complex interconnection matrix and gives the benefit of increased overall bus bandwidth, and a more flexible system architecture.

The different features of the PNX0101 multi-layer are described bellowed:

- Designed to work according to the Multi-layer AMBA Advanced System Bus (AHB Lite) concept.
- Supports up to 3 masters and 10 slave ports
- Supports only needed combination of 32 bit masters and slaves in according to the interconnect matrix.
- Bus implementation includes address decoding, arbitration and signal muxing.
- Zero-wait state operation, up to 100% bandwidth usage possible

13.2 Functional Description

The selection of master-to-slave-port path is done without waitstates.The address decoding of a master to a slave port and routing of the address and command signals to the slave is done within one cycle.

The interconnect matrix contains fully automatic 32-bit master to 64-bit slave port width conversion as well as 64-bit master to 32-bit slave width conversion.

Each master has a full address decoder for all slave peripherals in the system. A key issue in a multi-layer AHB system is the overall system complexity, arising from the number of concurrent actions possible in the system. To reduce the system complexity, the AHB multi-layer system only supports a unified memory map. This means all masters and slaves share a single, global 32-bit address space, and any master can select any slave in the system.

To avoid breaking the unified memory space, a specific section of the unified memory map is assigned as a shadow memory section. This memory section is virtual, i.e. no actual memory is present at the shadow address. It can be seen as a copy of a section of unified memory, specific to each master.

The bus arbiter is integrated in the AHB Multilayer and provides bus arbitration for a total of seven bus masters. Each of seven masters is represented by a three-bit priority that is supplied from three sources. The most significant bit is the bus request, the next from the external bit, and finally, the least significant bit assigned by the scheduler.

The following summarizes the rules which determine which master for the slave x is granted the bus:

1. A master requesting the bus is given the priority over a master not requesting the bus.

2. If only one master with the highest externally assigned priority is requesting the bus, it will receive the bus regardless of the underlying scheduling algorithm.

3. If no external priority is specified or all masters are of the same external priority, then the master selected by the scheduler is given the bus.

4. If no master is requesting the bus layer then the layer will generate idle cycles

5. A master that locks the bus can keep it indefinitely.

The user set the external priority bits in the configuration register block. The re-arbitration is done in one cycle (at the same time as the next access to a slave port), i.e there are no waitstates.

14 AHB2VPB BRIDGE

14.1 Overview

The AHB2VPB is a bus bridge between AMBA Advanced High-performance Bus (AHB) and the VPB/APB interface.

14.2 Features

- Supports a uni-directional VPB data bus interface
- 1 word deep write buffer
- Supports Single Clock or Dual Clock implementation
  - Single Clock Architecture (only ahb_clk): AHB2VPB Bridge 2
  - Dual Clock Architecture (separate ahb_clk and vpb_clk): AHB2VPB Bridge 0,1 and 3

14.3 Pin Description

The AHB2VPB Bus Bridge contains AHB and VPB interfaces that are naming, protocol and timing compliant. The other signals that are not part of either interface are listed in. All AHB/VPB interface pins are listed in Table 16.

Table 16 Pin description of AHB2VPB Bridge

NAME	TYPE	DESCRIPTION
AHB Interface		
ahb_clk	Input	AHB-bus clock
ahb_rst_an	Input	AHB-bus reset
ahb_addr[(n-1):0]	Input	AHB-bus address
ahb_write	Input	AHB-bus write
ahb_sel	Input	AHB-bus select
ahb_trans[1:0]	Input	AHB_bus transfer type
ahb_mastlock	Input	AHB-bus master lock signal
ahb_master[3:0]	Input	AHB-bus master
ahb_split_out[15:0]	Output	AHB-bus split completion request
ahb_wdata[31:0]	Input	AHB write data bus
ahb_rdata_out[31:0]	Output	AHB read data bus
ahb_resp_out[1:0]	Output	AHB-bus response signal
ahb_ready_out	Output	AHB slave ready output signal
ahb_ready	Input	AHB-bus global ready signal
ahb_prot	Input	AHB protection signal. ahb_prot[2] should be connected to this signal. 0: write command is not buffered 1: write command is buffered (provided there is no vpb_err signal for the corresponding VPB slave)
VPB Interface		
vpb_clk	Input	VPB-bus clock
vpb_rst_an	Input	VPB_bus reset
vpb_a_out[(m-1):0]	Output	VPB-bus address
vpb_write_out	Output	VPB-bus write signal
vpb_stb_out	Output	VPB-bus strobe signal

NAME	TYPE	DESCRIPTION
vpb_rdy_i	Input	Ready signal from the *ith* slave
vpb_err_i	Input	Error signal from the *ith* slave
vpb_sel_i_out	Input	VPB-bus slave select signal for the *ith* slave
vpb_di[(d-1):0]	Input	VPB-bus data input from VPB-bus slave
vpb_do_out[(d-1):0]	Output	VPB-bus data output from bridge

14.4 Architecture

The AHB2VPB Bus Bridge supports two architectures: Single Clock Architecture and Dual Clock Architecture.

Table 17 SYMPHONY AHB2VPB bridges

AHB2VPB	TYPE	PERIPHERALS
0	Dual Clock	Watch Dog
		RTC
		10 Bit ADC
		SYSReg
		CGU
		IOCONFIG
		EVENT ROUTER
1	Dual Clock	Timer1
		Timer2
		IIC Master/SLave
2	Single Clock	UART/IrDA
		USB
		MCI
		LCD Interface
		SDMA
		Flash Controller
3	Dual Clock	Audio Sub System

14.4.1 VPB DATA Bus Implementation

The VPB data bus implementation can only be uni-directional. The uni-directional implementation contains separate data input bus, vpb_di, and data output bus, vpb_do.

14.4.2 Memory Endianness

The bus bridge operation is independent of the endianness memory format.

14.4.3 Data Steering

Data steering for peripherals that have a narrow data bus (8 or 16 bits) is not supported. These peripherals are assumed to be accessed with word (32-bit) aligned addresses.

Peripherals with sub-word aligned addressing can be connected to the AHB2VPB Bridge, by shifting the address bits. Bits [x:0] of a byte aligned peripheral must be connected to bits [x+2:2] of the bridge. Bits [x:0] of a halfword (16-bit) aligned peripheral must be connected to bits [x+1:1] of the bridge. In both cases, the master on the AHB side of the bridge (software) must use word aligned addressing.

14.4.4 Write Buffer

The AHB2VPB Bridge contains a one word deep write buffer. Any VPB device that DOES NOT have a signal (like all APB devices) will take advantage of the write buffer provided ahb_prot is 1, if ahb_prot is 0 the devices won't able to use the write buffer. ahb_prot should be connected to ahb_prot[2] pin from AHB master. Devices with a signal won't be able to use the write buffer. The write buffer alleviates putting wait states on AHB. However, consecutive write access or write-read accesses to the bridge will insert some wait states because the write buffer is only one word deep.

14.4.5 Address Assignment

The AHB2VPB Bus Bridge allows the user to enter system specifications and information about each of the peripherals connected to the bridge. The address space allocated to each peripheral is described in "Memory Map". The bridge will assign the VPB memory map based on these parameters.

15 AHB EMBEDDED MEMORY CONTROLLER

15.1 Overview

The memory interface appears as a single slave to the bus, thus limiting the load on the bus itself. At extraction time the user can choose the memory type to be driven being SRAM or ROM.

The controller will always respond with an AHB OK or ERROR response, thus can be safely used in an AHB Multilayer system.

15.2 Functional Description

A block diagram of the memory controller is depicted in Fig.4. Note that the picture does not provide a detailed description of the design (for example the bsel lines are not drawn). The main purpose of the picture is to show that is possible, via the latency_cfg_a pins, to insert a level of pipeline between the memory and the AHB bus both on read data and control lines. The controller will insert zero, one or two AHB wait states accordingly with the value of the configurations pins.

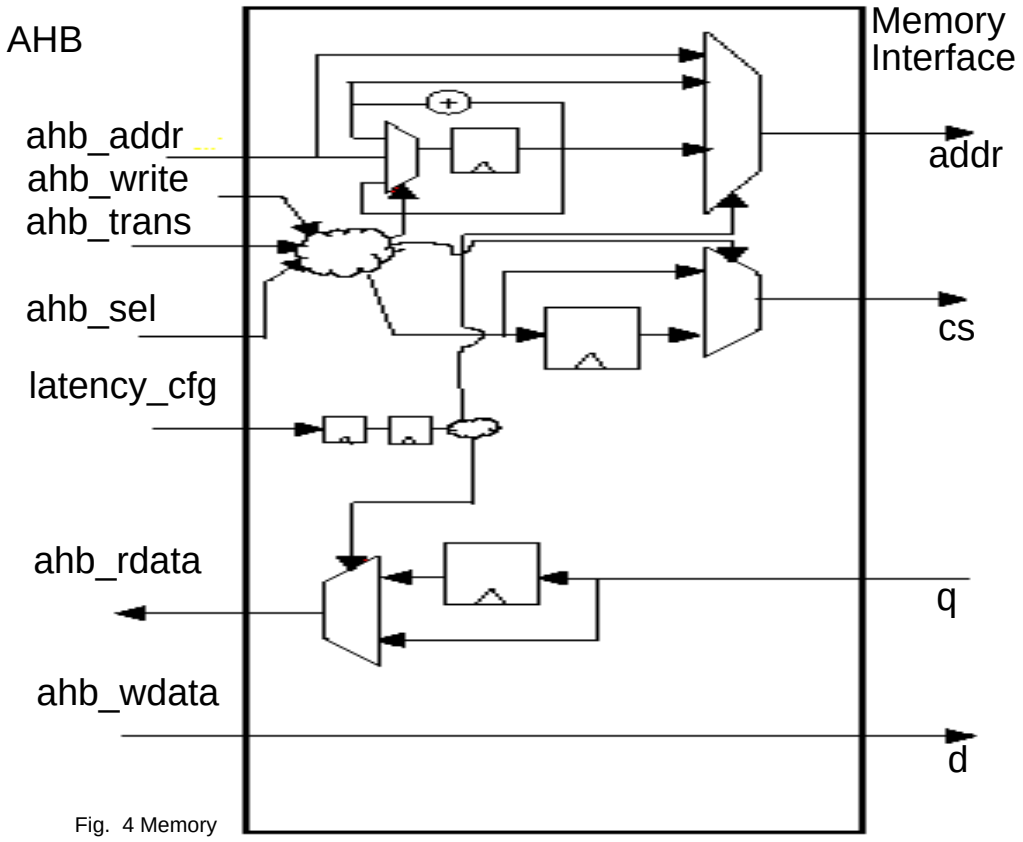


Fig. 4 Memory Controller Block

The controller supports the whole AHB protocol. The controller will respond with an error only if a write transaction is attempted towards a ROM.

It is possible to have the controller inserting a different number of wait cycles in order to achieve an optimal trade-off between latency and clock frequency. This is done by appropriate drive of the latency_cfg_a pins.

There are three possibilities:

- Zero wait cycle: the memory control signals are derived directly from the AHB bus and the AHB read data are derived directly from the memory read data signals. Thus, the maximum frequency is likely to be limited by the clock-to-data-out delay of the memory and, in second place, the speed at which the control signals arrive from the AHB master to the controller.
- One wait cycle: the memory control signals are still derived directly from the AHB bus but a pipeline stage is inserted between the memory read data and the AHB read data. All non-sequential read transfers will require one wait state to be inserted. All sequential transfers in a burst will be executing with no wait states.
- Two wait cycles: a level of pipeline is inserted both between the memory read data and the AHB read data and between the AHB control signals and the memory control ones. The internal timing of the controller is separated from the bus timing, allowing for maximum clock frequency at expenses of two wait states for each non sequential read transfer.

Note: Because of the different nature of latching address (in the controller for write operation, in the memory for read operation) a wait state has to be inserted if a read transaction has to be performed immediately following a write. During the wait state the controller will end the previous transaction writing data into the memory and in the next cycle it will process the AHB control signals and forward the address to the RAM for the read access.

The controller implements a configurable endianess. A pin is provided on the interface (bigend_a), when high the controller will act in big endian mode otherwise it will act in little endian mode. For read access of any size the behaviour in big or little endianess is the same. This means that for controller that will drive ROM's the pin has no effect to the design at all.

This value is the value that is driven when the signal is not active (outside the protocol).

- ahb_resp_out = 00
- ahb_ready = 1
- ahb_rdata_out = 0x0000.0000

15.3 Pin Description

Table 18 Pin Description

NAME	TYPE	CLOCK DOMAIN	DESCRIPTION
AHB			
ahb_clk	clock input	n.a.	AHB clock; please note that a clock that is identical to AHB clock must drive the memory clock input
ahb_rst_n	input	ahb_clk	AHB reset (active low); de-asserted synchronously to ahb_clk; assertion may be done asynchronously to ahb_clk. NOTE: this signal must be asserted upon a power-on reset.
ahb_addr[18:0]	input		AHB address, a is an integer dependent on the word depth of the memories addressed
ahb_write	input		0: read operation, 1: write operation
ahb_trans[1:0]	input		Type of transfer i.e NONSEQ,IDLE, BUSY, SEQ
ahb_burst[1:0]	input		Burst type: SINGLE, INCR, WRAP4, INCR4, WRAP8, INCR8, WRAP16, INCR16
ahb_size[2:0]	input		data size: byte, half-word, word, double word
ahb_se	input		AHB select
ahb_ready	input		AHB ready, HIGH when transfer has finished on the bus
ahb_wdata[31:0]	input		AHB write data. The signals are not present for ROM controllers.

NAME	TYPE	CLOCK DOMAIN	DESCRIPTION
ahb_resp_out[1:0]	output	ahb_clk	transfer response: OKAY, ERROR
ahb_rdata_out[31:0]	output		AHB read data
ahb_ready_out	output		HIGH when memory transaction is completed, depending on the value of latency configuration pins 0,1 or 2 wait states will be inserted.
Memory Signals			
q0[31:0]	input	ahb_clk	Memory read data
d[31:0]	output	ahb_clk	Memory write data. The signals are not present for ROM controllers.
cs	output		Memory enables
a[m:0]	output		Memory address lines, m =13 for RAM, m = 12 for ROM
web	output		Memory write enable (active low). The signal is not present for ROM controllers.
bsel[31:0]	input		Memory bit write enable. This signal allows writing only specified bits of a memory word, here are used to implement sub-word write AHB transaction. It is up to the user to ensure (e.g. in the decoding process) that no sub-word transaction are attempted toward the controller. The signal is qualified, for each instance, by the relative cs and web signals. The signal is not present for ROM controller.
gb	output		Memory output enable. The functionality of the pin is not used and the pin is driven by a flip-flop which always gives a 0 value. It is provided only to offer an interface to the integrator as complete as possible.
Configuration Signals			
bigend_a	input	any	Programmable Endianess- When HIGH, memory is treated as Big Endian; when LOW, memory is treated as Little Endian.
latency_cfg_a[1:0]	input	any	Programmable latency.
current_config[2:0]	output	ahb_clk	Current configuration, this value is the one actually stored in the internal flip-flops which determine the current configuration for the internal module logic. The value is coded as follows: current_config(0) represents the current bigend_a value stored current_config(2:1) represent the current latency_cfg_a value stored.
Others			
stall_req_a	input	any	Stall request for redundancy programming.
Test interface			
se	input	ahb_clk	scan enable; 1:shift, 0: operate
si	input		Scan chain input
so	output	ahb_clk	Scan chain output

15.4 Design Consideration

15.4.1 CLOCKING

The controller is fully synchronous to the rising edge of the ahb_clk signal.

15.4.2 RESET

The ahb_rst_n signal is used as an active low reset signal synchronous with the rising edge of ahb_clk and it must be asserted upon a power-on reset.

15.4.3 CONFIGURATION PINS

It is possible to configure the controller endianess and latency through dedicated configuration pins.

The first pin bigend_a set the module in big endianess mode when high, little endianess mode when low.

The two latency_cfg_a pins control the insertion of pipeline stages on memory read data out and AHB control signal. According to the value of those pins zero, one or two AHB wait states will be inserted.

The value is coded as follows:

“00” : there are no pipeline stages neither on AHB control signals nor on memory read data. This allows minimum latency at expenses of lower maximum clock frequency. The controller will not insert AHB wait states when in this mode.

“01” : memory select and addresses are derived from AHB signals in a combinatorial fashion while the memory read data is pipe lined. The controller will then insert a wait state for each non sequential read access.

“11” : both memory read data and AHB control signals are pipe lined. This allows maximizing the clock frequency at the expense of two wait states for each non sequential read access. Since those pins may be driven by logic on different clock domain a two stage synchronizer is inserted on every pin.

15.4.4 CHANGES CONFIGURATION AT RUN-TIME

To ensure that a configuration change does not happen while the controller is performing a transaction, the internal value of the flip-flops that represent the current configuration is updated only when the controller is not accessed with non sequential or sequential transaction for three clock cycles.

The value of those internal flip-flops is presented to the outside world through the current_config outputs.

Please note that the two stage synchronizer does not guarantee that all the values on latency_cfg_a pins are updated on the same clock cycle. Therefore it is not safe only allowing a fixed number of idle cycle (e.g. three) between the reconfiguration and the next transaction towards the controller.

The recommended safe procedure to change configuration mode in run time is then the following:

- Update the value of the configuration pins
- Poll the value on current_config outputs

Once you see that the new value is actually present on current_config outputs hen it is safe to start allowing masters to access the module.

15.4.5 LATENCY HIDING

In a situation where the CPU is performing a transaction towards a VPB peripheral, such a transaction is likely to be stretched by the AHB to VPB bridge inserting wait states. While ahb_ready signal is kept low by the bridge, another master may get the bus (e.g. DMA) and already provide the control information and address for the next transaction towards the embedded memory controller.

Since the master is not allowed to change address and control lines status without violating the AHB protocol, the controller will then hide latency meaning that it will use the informations already present on the bus to be able of performing the next transaction without wait states, even in two wait state mode.

Another scenario is when two non sequential read transfers are performed in a row then, while the first is waited, the information related to the second one will be processed allowing to complete these without inserting any wait state. Note that in this case, it will not be possible to hide the latency for a third transfer.

15.4.6 POWER CONSIDERATION

Intelligent clock gating (like providing a clock request output) is not directly supported since the number of flip-flops that can be switched off is very low. To save power, the clock may be switched off while no transfers are done. The IP_2107 has no need for an always active clock.

Internal power consumption is minimized by extensive use of enable signals, thus limiting the switching power dissipated.

15.4.7 REDUNDANCY PROGRAMMING

Memories with redundancy will require an initialization phase at start-up. During initialization the fuse contents is shifted from the fuse box to the memory redundancy registers.

To ensure that the controller is not performing memory transaction during initialization, a stall_req_a pin is provided. If, on reset de-assertion, the pin is driven high the controller will hold any transaction attempted from an AHB master (e.g. the CPU) and will stretch such a transaction inserting wait states until the stall_req_a pin is released to a zero value.

Thus, the stall_req_a input can be kept high as long as the last value in the chain is shifted in.

Note that, in functional mode after de-assertion, the value of the pin must be zero.

The signal is internally synchronized so it may be driven by flip flop belonging to a different clock domain. The reset value of those internal flip-flops is high so it is possible that the controller will insert wait states if a transaction is attempted towards the flip flop in the first two cycle after the reset is de-asserted even if the value of stall_req_a pin is zero.

15.4.8 DECREASING THE CLOCK FREQUENCY

In this example we assume a system capable of running at 150 MHz in which is possible to lower the clock frequency at run time, leaving PVT point unmodified.

1) The path going through memory read data (q0..q7) and AHB read data will then consume:

- clk_to_q_delay + 0.6ns + ahb_rdata_to_clk + clk_skew - clk_to_q_delay is depending on the memory instance, the value can be provided by the user at extraction time.

- ahb_rdata_to_clk is assumed 70% of AHB clock period..

- clk_skew is assumed 0.15 ns, the user is free to modify the value according to his own environment in the provided synopsys/constraints/tech_setup.scr file.

Assuming 150 Mhz AHB clock frequency and 4.0ns clk_to_q_delay for the memory, the path consumes about 9 ns. Thus, it is safe to use the controller with in zero wait state mode for clock frequencies up to about 105 MHz.

2) The path going through AHB control signals and memory control signals (CS,A). will then consume:

- clk_to_ahb_sel + 0.8ns + cs_to_clk_setup + clk_skew - cs_to_clk_setup is depending on the memory instance, the value can be provided by the user at extraction time.

- clk_to_ahb_sel is assumed 80% of AHB clock.

- clk_skew is assumed 0.15 ns as above. Assuming 150 Mhz AHB clock frequency and 1.5ns cs_to_clk_setup for the memory, the path consumes about 7.7 ns.

Thus, it is safe to use the controller with in zero wait state mode for clock frequencies up to about 125 MHz.

15.4.9 LOW LATENCY MODE

In Low Latency mode (latency_cfg_a = 00) the Controller will use the unregistered AHB control signals to generate the appropriate memory control signal.

When reading, the address will be latched into the SRAM using the rising edge of ahb_clk (start of Data phase) and the data on ahb_rdata_out will arrive by the following rising edge of ahb_clk (end of Data phase).

When writing the address will be registered on the rising edge of ahb_clk by the controller. On the next rising edge (end of Data phase), ahb_wdata and the registered address will be clocked into the SRAM.

Because of the different nature of latching address (in the controller for write operation, in the memory for read operation) a wait state has to be inserted if a read transaction has to be performed immediately following a write. During the wait state the controller will end the previous transaction writing data into the memory and in the next cycle it will process the AHB control signals and forward the address to the RAM for the read access.

15.4.10 One Wait State Mode

In one wait state mode (latency_cfg_a = 01) the controller will latch memory data output before sending the value on AHB bus. To allow this, one wait state will be introduced for each non sequential read transfer.

When writing, the write data and control signals are registered on the rising edge of ahb_clk by the controller as well as in low latency mode so no wait states will be introduced.

To allow a high throughput for burst transfers, the controller will generate the next address of the burst internally, based on the value of ahb_burst and ahb_size signals. It may be interesting to note:

- how the memory addresses for the sequential accesses are already fed to the memory before they actually appear on the bus.
- how the module reacts on BUSY transaction in the middle of a burst
- how the module reacts on early burst termination

15.4.11 Two Wait States Mode

In two wait states mode (latency_cfg_a = 11) a level of pipeline will be introduced on both memory data output and AHB control signal, thus two wait cycles will be introduced for each non sequential read transfer.

When writing, the write data and control signals are registered on the rising edge of ahb_clk by the controller as well as in low latency mode so no wait states will be introduced.

When writing, the write data and control signals are registered on the rising edge of ahb_clk by the controller as well as in low latency mode so no wait states will be introduced.

15.4.12 Redundancy programming

As mentioned before, if stall_req_a is asserted high when exiting from reset, the controller will hold any transaction possibly attempted, inserting wait states until the request is released. The controller will not select the memory instances while stalled.

Note as a few extra wait states are inserted following the release of stall_req_a because of the synchronization flip-flops.

15.4.13 Latency Hiding

As mentioned before, the controller will hide latency when possible. It means that it will use the informations already present on the bus, during a stretched data phase, to be able of performing the next transaction without wait states, even in two wait state mode.

16 DCDC CONVERTER

16.1 Overview

Latest DC/DC document can be found on the PNX0101 teamroom!

The PNX0101, is a device which, together with a content FLASH chip, is used in MP3 players. The player is powered by a single cell AA(A) battery. The process is CMOS18SFLV. This chapter describes the PNX0101 DC/DC converter with LDO's and auxiliary circuits, together called PNX0101_a1.The purpose of this document is to inform the customer and the design team about the technical implementation of the PNX0101 DC/DC converters.

16.2 Fuctional description

PNX0101 needs two supply voltages: between 2.7 and 3.3 V for analog functions and between 1.65 and 1.95Vfor digital functions. For normal operation one AA(A) battery of 1.5V will be used as an energy source, from which the DC/DC converter must generate the required voltage levels in an energy-efficiently manner. PNX0101 can communicate with a computer through USB. In that case the device can be powered by the provided USB supply of 5V (standard), which allows for less energy-efficient conversion.

The principle of operation is drawn in figure 2. Depicted are two inductive DC/DC converters, which will be used when the chip is battery operated. The VDDE pads of PNX0101 are externally connected to VOUT1, the VDDI pads (thus the supply for the Digital Core) are externally connected to VOUT2.

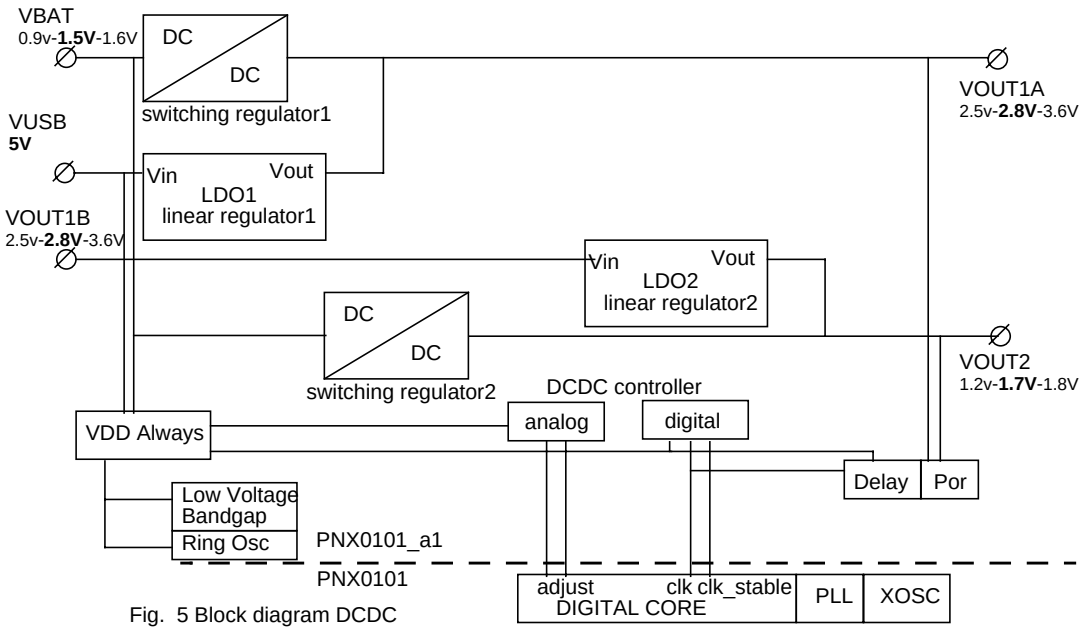


Fig. 5 Block diagram DCDC

In figure Fig.5 on page 87 , a block diagram of the DCDC converter depicted.

When the chip is supplied from USB, the outputs of the DC/DC converters will be overruled by two linear regulators; (the DC/DC converters are switched off). In that case the analog and digital supply voltages will become 3.3V and 1.8V respectively, in order to support both 1.8V (Samsung) and 3.3V (Toshiba, Samsung) FLASH. This is independent from the USB voltage (4.0 to 5.5V) so a reference circuit is needed. 5V is beyond the maximum voltage of the process and as

such requires the use of dedicated design techniques. By cascading the regulators only one of them needs to be designed for 5V operation. A few auxiliary circuits as the Low Voltage Bandgap and the Ring Osc are directly connected to the battery supply. During the start-up sequence the DC/DC-Controller uses the Ring Osc to control the switching regulators. After start-up the 12MHz from the Digital Core is fed to the DC/DC-Controller. In battery operation mode the output voltage of VOUT1 and VOUT2 can be controlled by 3 adjust bits. In the whole design care has to be taken with signal levels (level shifters) and the start-up, and shut-down, from battery to USB and from USB to battery transitions. A Delay circuit uses Ring Osc clocks to generate a delay of about 1ms for the RESET_B pulse. In the USB mode maybe the delay can be generated otherwise.

16.3 Registers

The DCDC interface function provides 3 software accessible registers, see syscreg Register map.

Table 19 SYSCREG Interface Register map belonging to DCDC converter

ADDRESS SPACE	OFFSET	REGISTER NAME	R/W ACCESS	RESET
0x8000_5000 - 0x8000_53FF	0x004	SYSCREG_DCDC_CONVERTER1	R/W	0x0000.0000
	0x008	SYSCREG_DCDC_CONVERTER2	R/W	0x0000.0000
	0x00C	SYSCREG_DCDC_CONVERTER_CLK	R/W	0x0000.0000

16.3.1 SYSCREG_DCDC_CONVERTER1 Register

Table 20 SYSCREG_DCDC_CONVERTER1 registers

BIT	VARIABLE	DESCRIPTION	RESET	R/W
2 - 0	dcdc1_adjust	Digital conversion data with respect to output voltage from DCDC1	101	R/W
31 - 3		Reserved		

Table 21 DCDC1_ADJUST registers for VDC1

BIT2	BIT1	BIT0	LOW THRESHOLD (V)	HIGH THRESHOLD (V)	AVERAGE (V)	VDELTA (MV)
0	0	0	3.562	3.710	3.636	148
0	0	1	3.406	3.548	3.477	142
0	1	0	3.250	3.385	3.318	135
0	1	1	3.094	3.223	3.159	129
1	0	0	2.938	3.060	2.999	122
1	0	1	2.782	2.898	2.840	116
1	1	0	2.626	2.735	2.681	109
1	1	1	2.470	2.573	2.522	103

16.3.2 SYSCREG_DCDC_CONVERTER2 Register

Table 22 SYSCREG_DCDC_CONVERTER2 registers

BIT	VARIABLE	DESCRIPTION	RESET	R/W
2 - 0	dcdc2_adjust	Digital conversion data with respect to output voltage from DCDC2	001	R/W
31 - 3		Reserved		

Table 23 DCDC2_ADJUST registers for VDCDC2

BIT2	BIT1	BIT0	LOW THRESHOLD (V)	HIGH THRESHOLD (V)	AVERAGE (V)	VDELTA (MV)
0	0	0	1.742	1.815	1.779	73
0	0	1	1.664	1.733	1.699	69
0	1	0	1.586	1.652	1.619	66
0	1	1	1.508	1.571	1.540	63
1	0	0	1.430	1.490	1.460	60
1	0	1	1.352	1.408	1.380	56
1	1	0	1.274	1.327	1.300	53
1	1	1	1.196	1.246	1.221	50

16.3.3 SYSCREG_DCDC_CONVERTER_CLK

This register defines which analog input channels are included and defines resolution in an A/D conversion.

Table 24 SYSCREG_DCDC_CONVERTER_CLK registers

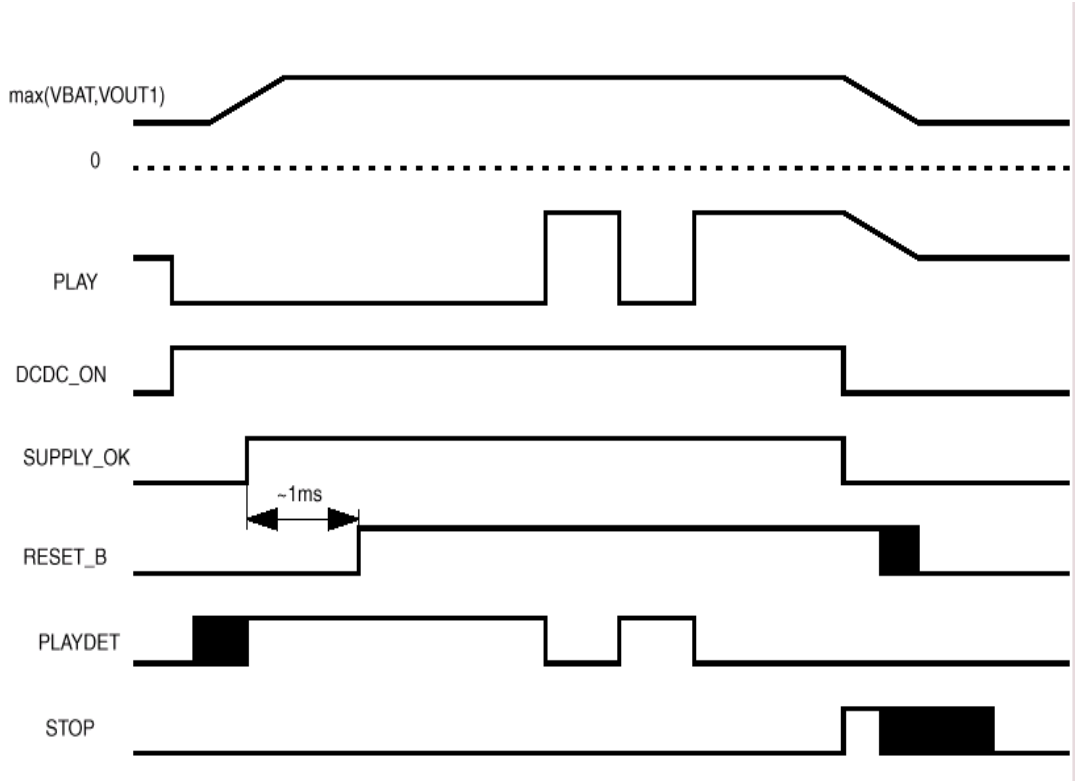
BIT	VARIABLE	DESCRIPTION	RESET	R/W
0	Clk_stable	Indication to DCDC converter that the internal clock stable is	0	R/W
31 - 1		Reserved		

Indication to DCDC converter that the internal clock stable is and the DCDC converter will switch from internal ring oscillator clock to 12 MHz clock.

16.4 Timing specification

16.4.1 PLAY/STOP PNX0101 INTERNAL DCDC(1) BATTERY POWERED.

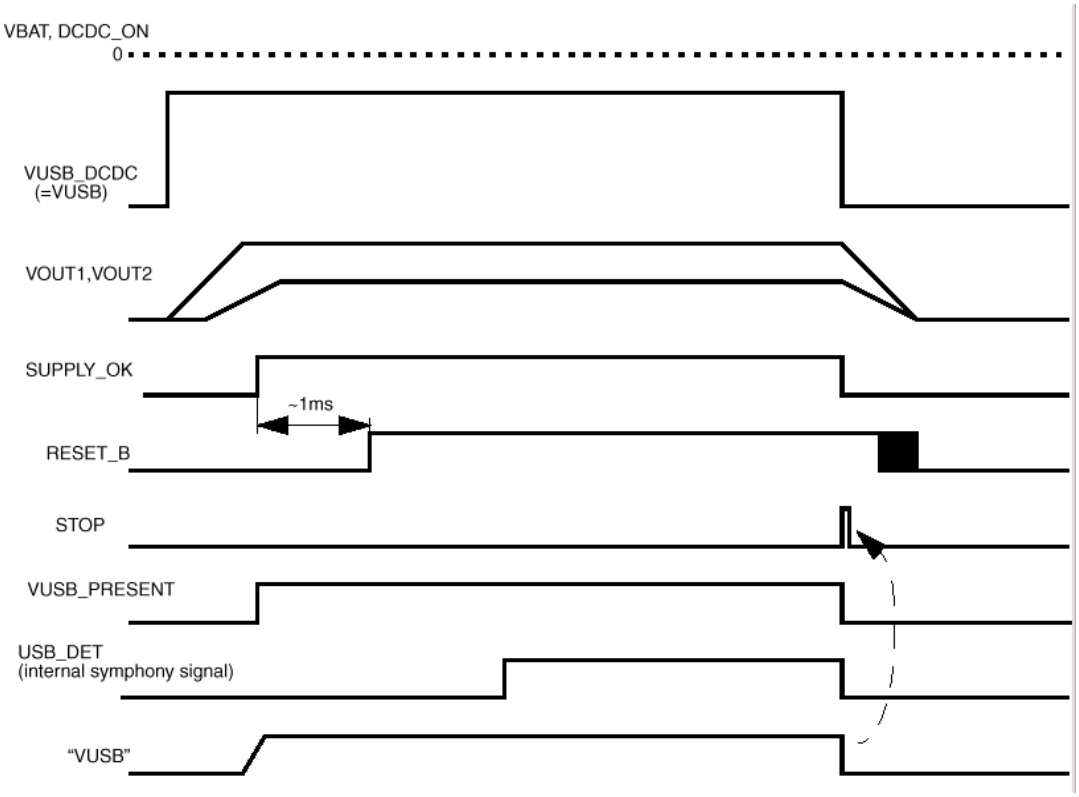
A start-up from the USB supply gives also a RESET_B pulse. The pin VUSB is externally connected to VUSB_DCDC. The signal “VUSB” is shaped by the bondpad supplies VDDI and VDDE.The disconnection of VUSB_DCDC generates a STOP.



The reference circuit, RingOsc and the POR (comparator function) will be fed by VDD_ALWAYS. The RESET_B stays low for about 1ms for proper reset of the PNX0101. Not shown in this figure is the CLK_STABLE signal showing the moment for the core clock to become available to the DC/DC converters. As soon as a stable core clock is detected, the DC/DC converters will switch to this clock in order to be in-phase with the DAC-clock, which will minimise interference into the audio signal. When this has happened, the chip has started up.

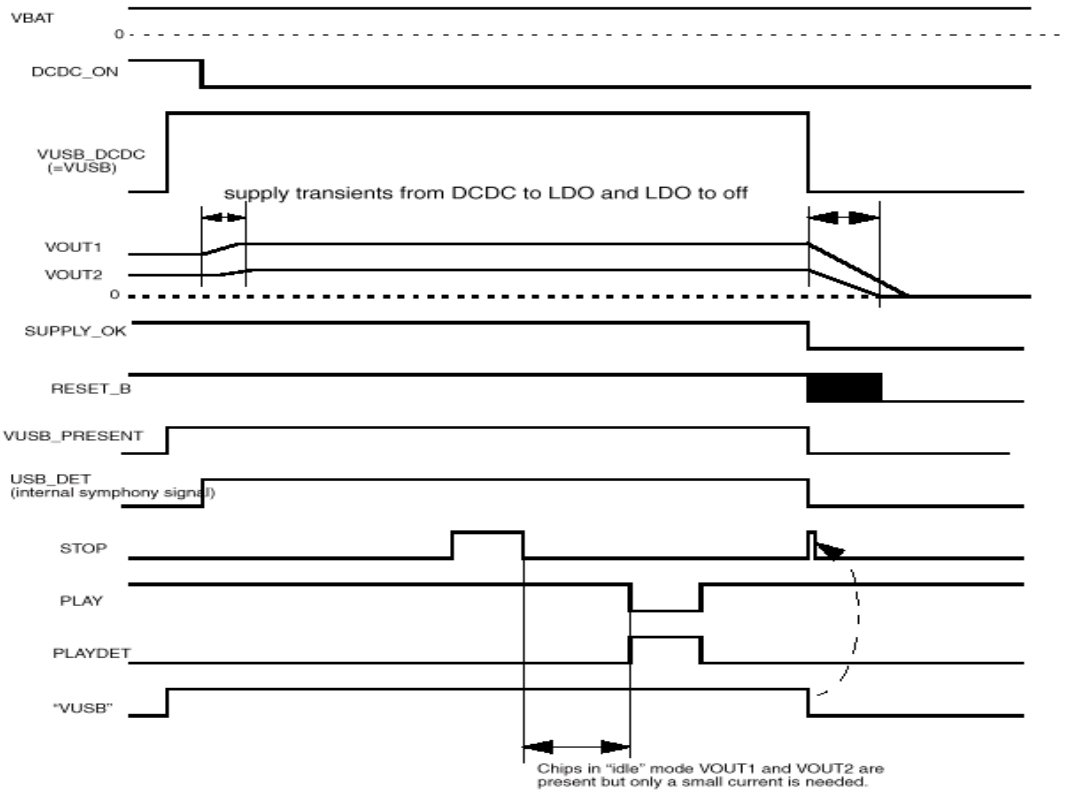
16.4.2 PLAY/STOP PNX0101 INTERNAL DCDC(2) USB POWERED

A start-up from the USB supply gives also a RESET_B pulse. The pin VUSB is externally connected to VUSB_DCDC. The signal “VUSB” is shaped by the bondpad supplies VDDI and VDDE.The disconnection of VUSB_DCDC generates a STOP.



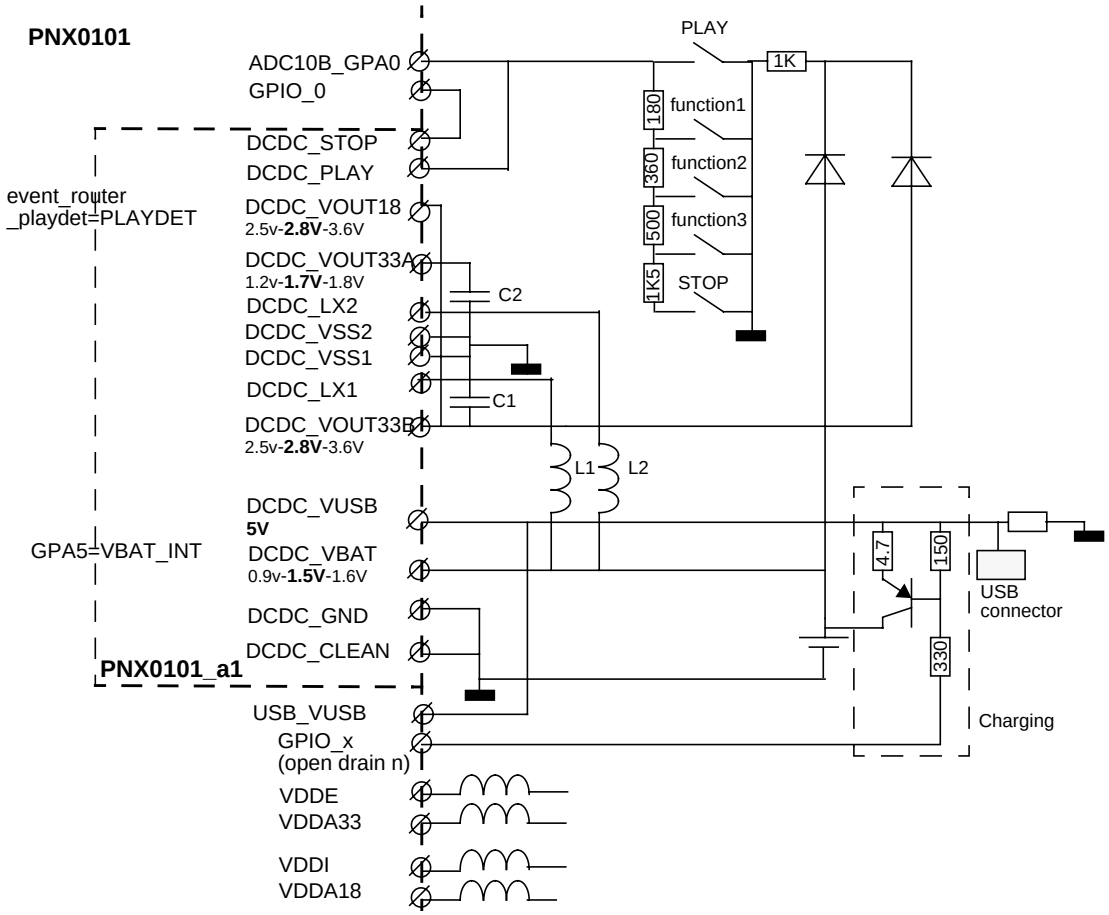
16.4.3 CHANGE FROM BATTERY TO USB SUPPLY

This graph show the timing diagram with an active (DCDC_ON) MP3 player changed to the USB supply. The USB supply has the priority. Care has to be taken that supply transients give not allowed spikes in the supplies or resets. It is not yet clear how to realise this exactly. An idea is that with USB supply and battery supply all DCDC switches are off but the controller remains active. When the USB-plug is disconnected the device goes to the off state. (A stop is generated)



In the “idle” mode the supplies VOUT1 and VOUT2 has to be still present, but the LDO’s has to deliver only a small current. The total device may not draw more than 500mA from the USB supply so a quiescent current of the few active circuits has to be less then 100mA each.

16.5 Application description


PNX0101
ADC10B_GPA0
GPIO_0
DCDC_STOP
DCDC_PLAY
event_router
_playdet=PLAYDET
DCDC_VOUT18
2.5v-**2.8V**-3.6V
DCDC_VOUT33A
1.2v-**1.7V**-1.8V
DCDC_LX2
DCDC_VSS2
DCDC_VSS1
DCDC_LX1
DCDC_VOUT33B
2.5v-**2.8V**-3.6V
DCDC_VUSB
5V
GPA5=VBAT_INT
DCDC_VBAT
0.9v-**1.5V**-1.6V
DCDC_GND
DCDC_CLEAN
PNX0101_a1
USB_VUSB
GPIO_x
(open drain n)
VDDE
VDDA33
VDDI
VDDA18
C2
C1
L1
L2
PLAY
1K
180
function1
360
function2
500
function3
1K5
STOP
4.7
150
330
USB
connector
Charging

16.6 Specification

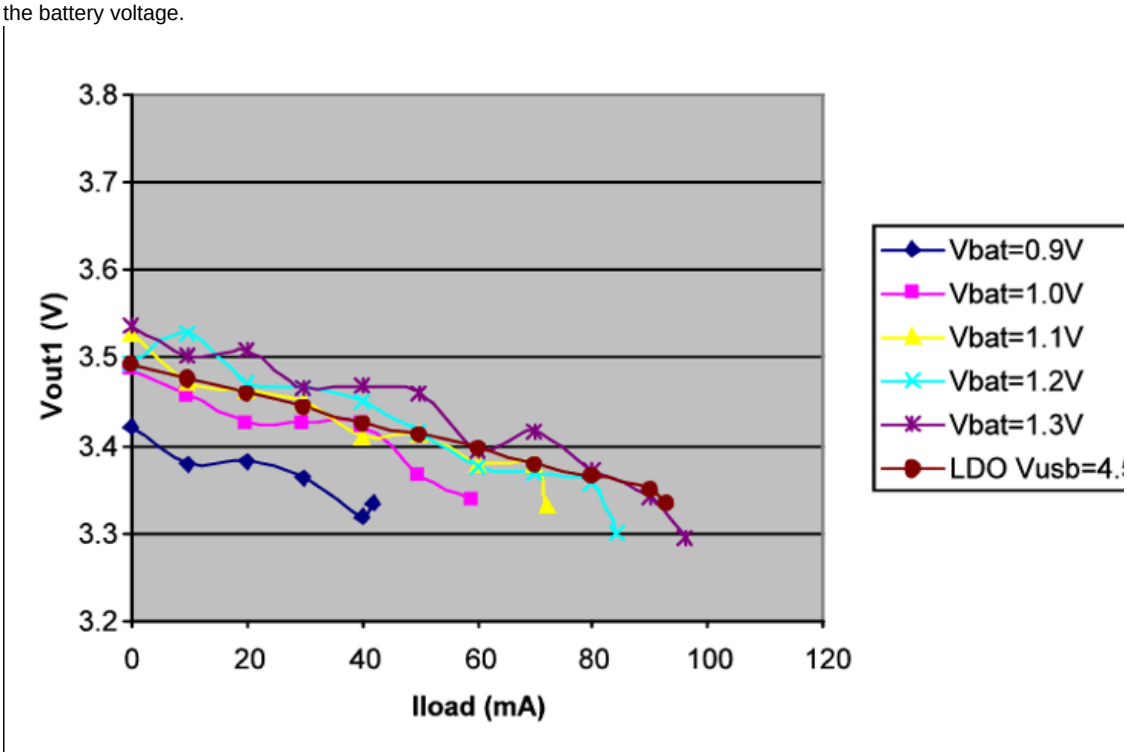
In this chapter, data is presented concerning the build in DC-DC convertor of the PNX0101. This data is based on measuring a limited amount of samples at ambient temperature.

16.6.1 SPECIFICATION TABLE

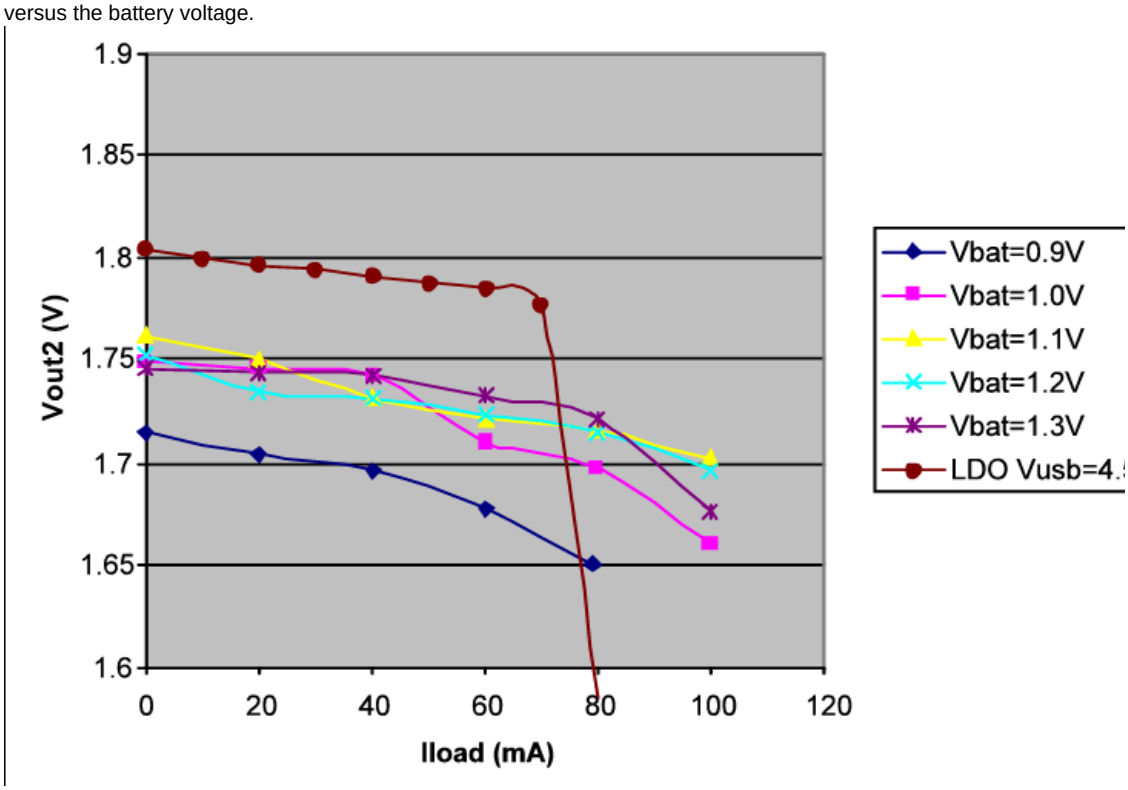
SYMBOL	PARAMETER	CONDITION	MIN	TYP	MAX	UNIT
General						
fsw	Switching frequency	during start-up	600	900	1200	kHz
		bit "clck_stable" set high.		1000		kHz
Ishutdown	current from battery in STOP mode (also called dark-current			500		µA
DC-DC convertor 1						
Vbat	Battery voltage		0.9		1.6	V
	Start-up voltage	Iload <25mA resistive load		0.95		V
Max Iload	Maximum load current	Vout1=3v3; Vbat=1.3V	100			mA
		Vout1=3v3; Vbat=1.2V		90		mA
		Vout1=3v3; Vbat=1.1V		80		mA
		Vout1=3v3; Vbat=1.0V		65		mA
		Vout1=3v3; Vbat=0.9V		50		mA
η	Efficiency	see figures in section 16.6.3				%
DC-DC convertor 2						
Vbat	Battery voltage		0.9		1.6	V
Max Iload	Maximum load current	Vout1=1V68; Vbat=1.3V	100			mA
		Vout1=1V68; Vbat=1.2V	100			mA
		Vout1=1V68; Vbat=1.1V	100			mA
		Vout1=1V68; Vbat=1.0V	100			mA
		Vout1=1V68; Vbat=0.9V		75		mA
η	Efficiency	see figures in section 16.6.3				%
LDO's						
Vusb	USB voltage		4.0		5.5	V
Isuspend	Current in suspend mode			700		µA
Vout1	Output voltage LDO1	Iload = 0mA		3.5		V
		Iload = 100mA		3.33		V
Vout2	Output LDO2	Iload = 0mA		1.8		V
		Iload = 100mA		1.78		V

16.6.2 BATTERY VOLTAGE VERSUS LOAD CURRENTS

In the picture below is the load graph versus output voltage of the 3V3 DC-DC convertor ads well as the LDO given versus the battery voltage.

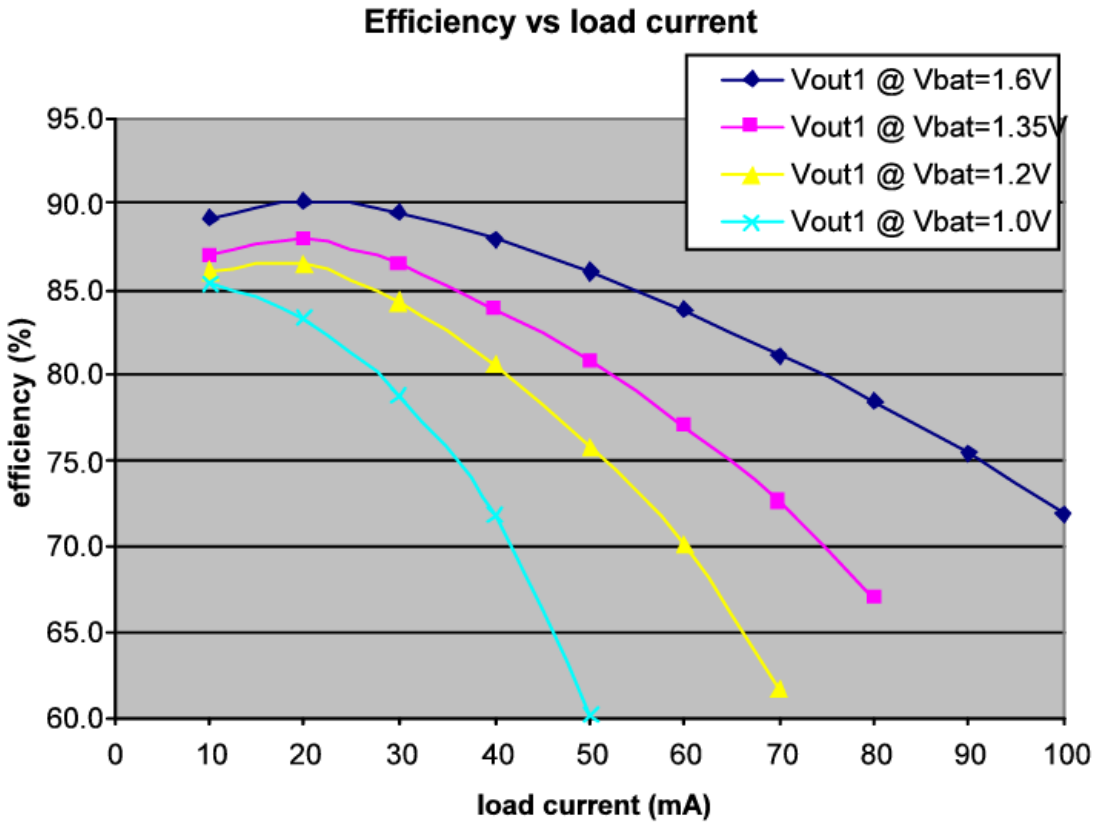


In the picture below is the load graph versus output voltage of the 1V8 DC-DC convertor ads well as the LDO given versus the battery voltage.

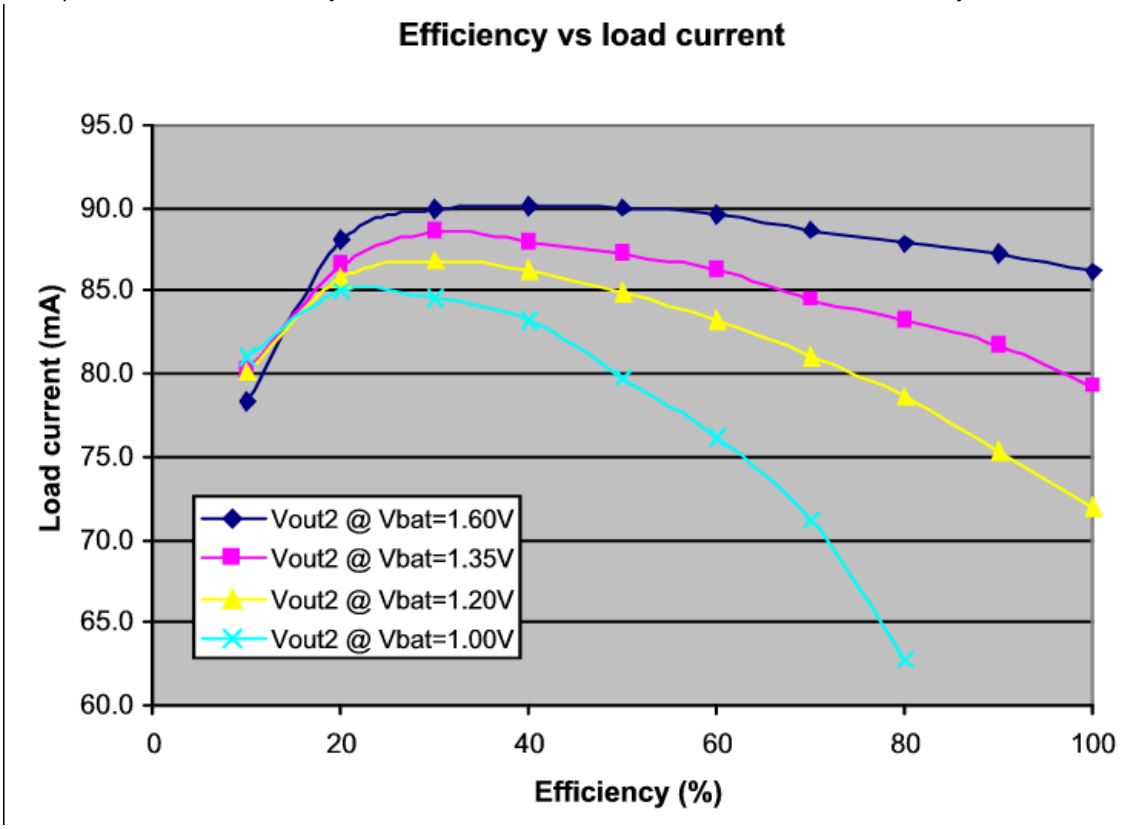


16.6.3 EFFICIENCY CURVES

In the picture below is the efficiency versus load curves of the 3V3 DC-DC convertor, wrt to the battery level..



In the picture below is the efficiency versus load curves of the 3V3 DC-DC convertor, wrt to the battery level.



17 Clock Generation Unit (CGU)

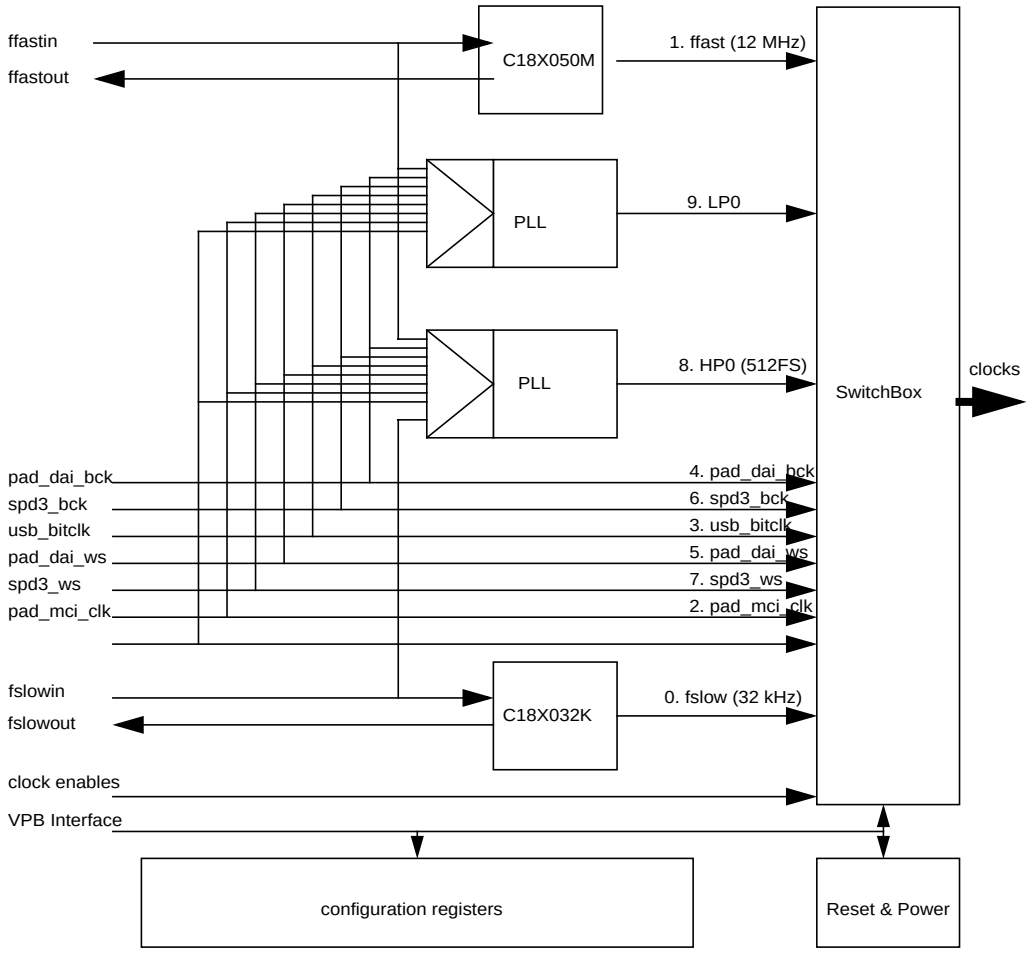
17.1 Overview

The clock Generation Unit generate all clock signals required for the PNX0101 chip.

The CGU contains:

- Two oscillators (12 MHz and 32.768 kHz)
- A PLL to generate audio sample frequencies
- A PLL to generate the clocks for the system
- A clockswitch block
- A configuration register block
- A reset and power block

The following figure shows the CGU block diagram:



There are two oscillators, one oscillator of 12 MHz and one of 32.768 KHz

The 32.768 KHz oscillator generates the clock for the Real Time Clock Module.

The 12 MHz oscillator is used in combination with the two pll's and the external clocks to generate the system frequencies.

The two PLL's are:

AUDIOPLL - Generate audio sample frequencies

MASTERPLL - Generate the bus clocks for AHB,VPB and DSP subsystem

All PLL's are programmed with the registers in the register configuration block (no software decoder is implemented)

17.1.1 THE OSCILLATOR

The C18XO50M oscillator is a 50 MHz Pierce crystal oscillator with amplitude control. It can be used in many applications e.g. as a digital reference for digital circuits, A/D and D/A clocking, etc. It is a robust design and can be used across a large frequency range.

Features:

- On-chip biasing resistance
- Amplitude controlled
- Large frequency range: 1 MHz - 20 MHz (reduced because hf pin is made low)
- Slave mode
- Power Down mode
- Bypass Test mode

17.1.1.1 Oscillation mode

In oscillation mode the oscillator gain stage can have a normal or large transconductance, determined by the hf pin. A large transconductance is required for higher oscillation frequencies, higher series resistance of the crystal and higher external load capacitors. In table below the values of the external components for frequency ranges between 1 and 20 MHz are given. Since the feedback resistance is integrated on chip, only a crystal and the capacitances Cx1 and Cx2 need to be connected externally in the case of fundamental mode oscillation.

Table 25 Crystal oscillator interface description

FREQUENCY	CRYSTAL LOAD CAPACITANCE	MAXIMUM CRYSTAL SERIE RESISTANCE	EXTERNAL LOAD CAPACITANCE
1/5 MHz	10 pF	300 ohm	18 pF
	20 pF	300 ohm	39 pF
	30 pF	300 ohm	58 pF
5/10 MHz	10 pF	300 ohm	18 pF
	20 pF	100 ohm	39 pF
	30 pF	100 ohm	57 pF
10/15 MHz	10 pF	160 ohm	18 pF
	20 pF	60 ohm	39 pF
15/20 MHz	10 pF	80 ohm	18 pF

17.1.1.2 Slave mode

The oscillator can be used as a slave device for an external clock. In this mode the external clock is applied

on input terminal osc_in and is transferred to the output terminal clkout via the comparator and multiplexer / buffer. Since the comparator has a high pass characteristic the input clock frequency must be higher than 1 MHz. In Slave mode the input signal should be AC coupled by means of a capacitor of 100 pF, with an amplitude of at least 200 mVrms.

17.1.2 THE AUDIO PLL

The C18PL550M is a multi purpose Phase Locked Loop (PLL).

features:

- Integrated PLL with no external components for clock generation
- Input frequency range: 100 kHz - 150 MHz
- CCO output frequency: 275 MHz - 550 MHz
- Output frequency range: 4.3 MHz - 550 MHz
- Programmable pre-divider, feedback-divider and post-divider
- On the fly adjustment of the clock possible
- Positive edge locking
- Frequency limiter to avoid hang-up of the PLL
- Lock detector
- Power down mode
- Possibility to bypass whole PLL, the post-divider or the pre-divider
- Possibility to disable the output clock
- Skew mode
- Free running mode
- Scan mode
- Maximum Peak Cycle-to-Cycle Output Jitter # 200 ps

17.1.2.1 Functional Description

The clock input has to be fed to Fin, Fout is the PLL output. The analog part of the PLL consists of a Phase Frequency Detector (PFD), filter and a Current Controlled Oscillator (CCO). The PFD has two inputs, a reference input from the (divided) external clock and one input from the divided CCO output clock. The PFD compares the phase/frequency of these input signals and generates a control signal if they don.t match. This control signal is fed to a filter which drives the current controlled oscillator (CCO).

The PLL contains three programmable dividers: pre-divider (N), feedback-divider (M) and post-divider (P). Every divider contains a bus to load a divider ratio. Besides the normal operating mode it is also possible to use the skew mode. In this mode a phase error can be corrected.

In power down mode the oscillator will be stopped and the output will be zero. With the bypass mode the input clock will be fed to the output of the PLL. In normal operation mode the output clock of the PLL will disappear very fast if the input clock disappears. To avoid this, the free running mode can be used. To test the digital part there are also scan-chain test pins. In enable mode the output clock of the PLL is enabled.

The Audio PLL will be used to generate the audio sampling frequencies and works in normal operating mode with pre-divider and with post-divider, the output frequency of the Audio PLL is 512FS where FS is the sampling frequency.

Table 26 AUDIO PLL DIVIDER RATIO SETTINGS FOR 12 MHz

FS (KHZ)	FOUT (MHZ)	FCCO (MHZ)	PREDI-VIDER (N_{SEL})	N_{DEC}	FEED-BACK-DI VIDER (M_{SEL})	M_{DEC}	POST-DI VIDER (P_{SEL})	P_{DEC}	SELR	SELI	SELP
96	98.304	393.216	111	82	1834	9649	1	66	14	2	31
88.2	90.3168	361.267	18	208	285	24587	1	66	2	4	31
64	65.536	393.216	111	82	1834	9649	2	42	14	2	31
48	49.152	491.52	24	63	511	16416	4	5	3	2	31
44.1	**45.1584**	**451.584**	86	251	1636	9099	4	5	12	2	31
32	32.768	327.68	74	102	1023	8194	4	5	7	2	31
24	24.576	491.52	24	63	511	16416	9	14	3	2	31
22.05	22.5792	406.426	60	131	1032	1408	8	23	8	2	31
16	16.384	327.68	74	102	1023	8194	9	14	7	2	31
12	12.288	368.64	24	63	383	16973	14	24	2	3	31
11.025	11.2896	406.426	60	131	1032	1408	17	7	8	2	31
8	8.192	327.68	74	102	1023	8194	19	31	7	2	31

17.1.2.2 Application

Table 27 Status of external pins in all different modes

MODE	PD	CLK_EN	BYPASS	DIRECTI	DIRECTO	SKEW_EN	FRM	SCAN_EN	SCAN
Supply pins									
1: Normal operating	0	1	0	1/0	1/0	0	0	0	0
2: Reserved	0	1	0	1/0	1/0	1	0	0	0
3: Power Down	1	x	x	x	x	x	x	x	0
4: Bypass	0	1	1	x	x	x	x	0	x
5: Reserved	0	x	0	x	x	x	1	0	x
6: Scan	1	x	x	x	x	x	x	1	1
7: Enable	x	1/0	x	x	x	x	x	x	x

Mode 1: Normal operating mode

Mode 1 is the normal operating mode. Table 27 gives the status of the external pins to get the PLL in normal operating mode.

The pre- and post-divider can be selected (table 28) giving:

- Mode 1a: Normal operating mode without post-divider and without pre-divider
- Mode 1b: Normal operating mode with post-divider and without pre-divider
- Mode 1c: Normal operating mode without post-divider and with pre-divider
- Mode 1d: Normal operating mode with post-divider and with pre-divider

The best phase-noise and jitter performance to get at the output of the PLL (Fout) a high as possible reference clock (clkref) at the PFD has to be used. Therefore mode 1b and 1c are recommended, if it is possible to make the right output frequency without pre-divider.

By using the post-divider the clock at the output of the PLL (Fout) the divider ratio is always even because the divide-by-2 divider after the post-divider.

Table 28 Modes within the normal operating mode

MODE	DIRECTI	DIRECTO
1a	1	1
1b	1	0
1c	0	1
1d	0	0

In normal operating mode 1a **without post-divider and without pre-divider**. The operating frequencies are:

(1)

$$Fout = Fcco = 2 \cdot (M + 1) \cdot Fin \wedge (275MHz \leq Fcco \leq 550MHz) \wedge (100KHz \leq Fin \leq 150MHz)$$

The divider ratios is programmable: Feedback divider M (0 -> 2047)

In normal operating mode 1b **with post-divider and without pre-divider**. The operating frequencies are:

(2)

$$Fout = \frac{Fcco}{2 \cdot (P+1)} = \frac{(M+1)}{(P+1)} \cdot Fin \wedge (275MHz \leq Fcco \leq 550MHz) \wedge (100KHz \leq Fin \leq 150MHz)$$

The divider ratios is programmable: Feedback divider M (0 -> 2047), Post-divider P (0 -> 31)

In normal operating mode 1c **without post-divider and with pre-divide**r. The operating frequencies are:

(3)

$$Fout = Fcco = \frac{2 \cdot (M+1)}{(N+1)} \cdot Fin \wedge (275MHz \leq Fcco \leq 550MHz) \wedge (100KHz \leq \frac{Fin}{(N+1)} \leq 150MHz)$$

The divider ratios is programmable: Feedback divider M (0 -> 2047), Pre-divider N (0 -> 255)

In normal operating mode 1d **with post-divider and with pre-divider**. The operating frequencies are:

(4)

$$Fout = \frac{Fcco}{2 \cdot (P+1)} = \frac{(M+1)}{(N+1) \cdot (P+1)} \cdot Fin \wedge (275MHz \leq Fcco \leq 550MHz) \wedge (100KHz \leq \frac{Fin}{(N+1)} \leq 150MHz)$$

The divider ratios are programmable: Feedback divider M (0 -> 2047), Pre-divider N (0 -> 255), Post-divider P (0 -> 31)

Mode 2: Reserved

Mode 3: Power Down Mode (pd)

In this mode (pd = ‘1’), the oscillator will be stopped, the lock output will be made low, and the internal current reference will be turned off. During power down mode it is also possible to load new divider ratios at the input buses. The lock signal will be made high once the PLL has regained lock on the input clock.

Mode 4: Bypass mode (bypass)

In the bypass mode the input clock (Fin) will be bypassed to the output of the PLL. Precaution has to be taken that no spikes will occur at the output of the PLL (Fout) by switching into and out of the bypass mode. To avoid spikes the output has to be disabled during switching into and out of the bypass mode. This can be done with pin clken. During bypass mode the PLL can also set into power down mode. The clken mode has a higher priority than the bypass mode which indicates that for bypass also clken should be high.

Mode 5: Reserved

Mode 6: Scan mode

In this mode the digital logic in the PLL can be scanned on faults. All the digital circuitry in the PLL is connected to one scan-chain with the input and output pins scan_in and scan_out and the enable pin scan_en and test clock clk_test. By setting the PLL into test mode (scan = ’1’), the test clock is connected to the scan-chain. During scan mode the PLL can be set into power down by the pd-bit.

Mode 7: Enable mode (clken)

In the enable mode the output clkout of the PLL is enabled. If clken= ’0’ the output of the PLL is low (clkout = ’0’). Precaution is already taken that no spikes will occur at the output of the PLL (clkout) by switching into and out of the enable mode.

Selecting the divider ratios

The PLL contains three dividers: pre-divider (N), feedback-divider (M) and post-divider (P). The divider ratios can be selected in three different ways:

1. When the digital and analog power supply become high the divider ratios at the input busses xsel (msel, psel, nsel) are loaded into the divider.

2. The divider ratio of the different dividers (M, N, P) can also be selected on the fly with help of a handshake protocol. First a new divider ratio has to be put at the divider input bus (xsel), after that a request (xreq="1") must be given. As soon as the divider gives his acknowledge (xack='1) the new divider ratio is loaded and the request can be made low (xreq='0'). To avoid forbidden divider ratio combinations it is not recommended to change the divider ratios of the different dividers at the same time. Because the divider ratios internally are not loaded at the same time.

3. The third possibility to select new divider ratios, is giving a power down signal (pd). After power down the new divider ratios at the input buses xsel (msel, psel, nsel) will be loaded into the dividers.

Remark:

The lock signal of the PLL isn't stable when the PLL has a frequency lock, so when the PLL does have a small phase shift the lock signal is becoming low and indicating that the lock has been removed.

17.1.3 The Master PLL

The Master PLL works in normal operating mode with feedback-divider and with post-divider, this means that the base for the clock signal is the current controlled oscillator (Fout = Fcco/P), running on 384MHz. The output clock (Fout) is divided by 4 to generate a 96 MHz clock.

FEATURES Masterpll (C18PLL160M)

- Integrated PLL with no external components for clock generation
- Functional down to 1.2 V (with reduced frequency range)
- 10 - 25 MHz input frequency range
- 9.75 - 160 MHz selectable output frequency with 50% output duty cycle
- 156 - 320 MHz Current Controlled Oscillator (CCO) frequency range
- Power down mode
- Input clock bypass mode
- Lock detector available
- Current consumption max. 1 mA
- Peak-to-peak cycle jitter max. 300 ps

17.1.3.1 Application

The C18PL160M has five different operating modes. Table 72 shows the available modes, together with the state of the control signals that must be applied to get the PLL in each of the modes.

Table 29 Status of external pins in all different modes

MODE	PD	BYPASS	DIRECT	P23EN	SCAN_EN
1: Normal operating	0	0	0	x	0
2: Direct CCO mode	0	0	1	0	0
3: Power Down mode	1	x	x	0	x
4: Bypass mode	0	1	0	x	0
5: Direct bypass mode	0	1	1	x	x

Post divider

The division ratio of the post divider is controlled by the psel[1:0] input. The division ratio is two times the value of P selected by psel[1:0] as shown in table 73. This guarantees an output clock with a 50% duty cycle.

Table 30 Status of external pins in all different modes

PSEL<1:0>	VALUE OF P	DIVISION RATIO (2P)
00	1	2
01	2	4
10	4	8
11	8	16

Feedback divider

The feedback divider's division ratio is controlled by the msel[4:0] input. The division ratio between the PLL's output clock and the input clock is the decimal value on msel[4:0] plus one, as specified in table 74.

Table 31 Status of external pins in all different modes

MSEL<4:0>	DIVISION RATIO (M)
00000	1
00001	2
00010	3
00011	4
.....	.
11110	31
11111	32

Changing the divider values

Changing the divider ratio while the PLL is running is not recommended. As there is no way to synchronize the change of the msel and psel values with the dividers, the risk exists that the counter will read in an undefined value, which could lead to unwanted spikes or drops in the frequency of the output clock. The recommended way of changing between divider settings is to power down the PLL, adjust the divider settings and then let the PLL start up again.

Frequency selection

Mode 1 - Normal mode
In this mode the post divider is enabled, giving a 50% duty cycle clock with the following frequency relations:

(5)

$$Fout = \frac{Fcco}{2 \cdot P} = M \cdot Fin \wedge (156MHz \leq Fcco \leq 320MHz)$$

The divider ratios are programmable: Feedback divider M (1 -> 32), Post-divider P (1 -> 4)

Mode 2 - Direct CCO mode
In this mode the post divider is bypassed and the CCO clock is sent directly to the output(s), leading to the following frequency equation:

(6)

$$Fout = Fcco = M \cdot Fin \wedge (156MHz \leq Fcco \leq 320MHz)$$

The divider ratios are programmable: Feedback divider M (1 -> 32)

To select the appropriate values for M and P, it is recommended to follow these steps:
1. Specify the input clock frequency Fclkin
2. Calculate M to obtain the desired output frequency Fclkout with
3. Verify that all frequencies and the divider value conform to the limits specified in tables 3, 7 and 8.
Note that although the post divider is not used, it is still running in this mode. To reduce the current consumption to the lowest possible value, it is recommended to set psel<1:0> to ‘00’. This will set the post divider to divide by two, which causes it to consume the least amount of current.

Mode 3 - Power down mode
In this mode, the internal current reference will be turned off, the oscillator and the phase-frequency detector will be stopped and the dividers will enter a reset state. While in power down mode, the lock output will be low, to indicate that the PLL is not in lock. When the power down mode is terminated by making pd low, the PLL will resume its normal operation, and will make the lock signal high once it has regained lock on the input clock.
Mode 4 - Bypass mode
In this mode, the analog part is placed in power down, and the input clock is sent to the post divider. This mode can be used e.g. to perform a functional test of the post divider and/or the feedback divider. The various frequencies are given by:

(7)

$$Fout = \frac{Fcco}{2 \cdot P} \wedge Fdivo = \frac{Fout}{M} \wedge (156MHz \leq Fcco \leq 320MHz)$$

The divider ratios are programmable: Feedback divider M (1 -> 32), Post-divider P (1 -> 4)

Mode 5 - Direct bypass mode
In this mode, the analog part is placed in power down, the post divider is disabled and the input clock is sent directly to the output(s). This mode can e.g. be used to perform either a function test and/or a scan test on the feedback divider. In this mode, the frequency of the feedback clock output is given by:

(8)

$$Fdivo = \frac{Fout}{M} \wedge (156MHz \leq Fcco \leq 320MHz)$$

The divider ratios are programmable: Feedback divider M (1 -> 32), Post-divider P (1 -> 4)

Performance / current consumption trade-off
In some cases it will be possible to achieve the same input and output frequencies with different combinations of internal frequencies and divider settings. Which setting to use in such cases depends on whether performance or current consumption is most important.
Minimum current consumption
1. A lower CCO frequency reduces the analog part's current consumption and that of the post-divider.
2. A lower input frequency (Fclkin) reduces the current consumption of the analog part and the feedback divider but it will increase the startup time.
Maximum performance
1. A higher input frequency (Fclkin) increases performance by making the PLL correct phase and frequency errors more often. It also decreases the time it takes to start up and lock onto the input clock.
2. A higher CCO frequency increases the swing of internal signals which reduces the noise sensitivity.

17.2 Functional Description

17.2.1 The Clock Generator offers support for individual AHB master disabling for the SDMA and ARM.

AHB Masters must not be performing bus accesses when put into sleepmode. To ensure the master is removed from the bus a disable request is generated. This signal is used by the bus (ahb_multilayer) to deny any new access requests. After the master has finished its current request a disable grant is generated by the bus. Only when all grants have been received is the internal wakeup_i signal going to the clockswitchbox allowed to become low.

- Clock disabling is preceded by a software setup in which for a set of clocks the wakeup_en control bit is set in their PCR (see clockswitchbox). It also configures the event_router to respond to certain events which can generate a wakeup towards the CGU.
- The software then activates the clock disabling by writing 'b11 to the powermode register.
- The CGU will before entering powermode ensure that all masters for which the wakeup_en control bit has been set and which support master disabling are no longer performing bus accesses. To do this it sets the corresponding master_disable_req signal high.
- The multilayer uses the disable request to mask bus access requests from the master. The master is allowed to finish its current activities but will not be able to initiate a new access.
- When the master is no longer performing accesses (this might be immediately) the bus will return a master_disable_grant signal to the CGU.
- As soon as all master, for which the wakeup_en control bit has been set, are disabled the clocks will be disabled.
- Clocks will become quiescent after two of their active edges.
- When an wakeup event occurs the wakeup signal output of the event router becomes high and clears (asynchronously) the power mode register in the CGU, thereby reenabling the clocks after two of their active edges.
- One cycle after a master clock starts running its master_disable_req signal becomes low and the bus will again process its access requests.

17.3 Clock Config block

17.3.1 Power up reset and reset domains

Power up reset is initiated by RSTIN_N, an external signal, or by 'wdr', an internal signal from the watchdog. The RSTIN_N signal is intended as a "battery insertion" type of reset, active low. Both the 'RSTIN_N' and the 'wdr' initialize the clockgen module into its reset state. A 'vddalways_resetn' reset signal is created, which is stretched until the first rising 'pclk' edge after the power up signal has become inactive. 'vddalways_resetn' activates the other reset signal(s), defined as reset domains.

17.3.2 Watchdog identification register

To find out whether a power-on reset was caused by an external 'RSTIN_N', or a watchdog reset, a special 'wd_bark' register is used. If a watch dog reset has occurred the 'wd_bark' register will be 1 after reset.

17.3.3 Reset domains

The Clock Generation can generate as many resets, synchronized to specific clock domains, as required. All resets domains specified have a re-synchronisation clock input, as shown in the figure below. This is used to keep the reset active until the second rising edge of the resync clock, after 'vddalways_resetn' has become inactive. In case the reset is asynchronous it is kept active for another half clock cycle to prevent hold violations. The reset can be selected either as active high or active low polarity. It can also be selected for synchronous or asynchronous usage. The implementation difference lies with the handling of the signal during scan test mode and its deactivation timing. During production test (scantestmode=1), all asynchronous reset outputs are directly, asynchronously connected to RSTIN_N. All synchronous signals remain controlled by the synchronisation flip flops.Optionally a software reset, active low, can be generated. This will add a configuration register, this signal is 'AND'ed with the normal reset and serves as data input to the synchronisation flip flops.

17.3.4 Controlling the frequency sources

There are four types of analog devices in the clock generation unit. These are used as frequency sources.

- C18OSC32K: 32 kHz oscillator, generates a fixed 32kHz square wave.
- C18OSC50: 10-50 MHz peripheral oscillator.
- C18PL160: LPPLL (Phase Locked Loop)
- C18PL550: HPPLL (Phase Locked Loop).

17.3.5 Programming clocks

The sequence to follow when programming a device is:

- Disable device by activating the power down mode or place it in reset mode if the module is digital.
- Set the correct operating mode and multiplication/division factor.
- Enable the device by placing it in functional mode.
- Wait until generated frequency is stable.

When reprogramming a frequency source make sure no clock is being generated from it.

17.4 Switchbox module

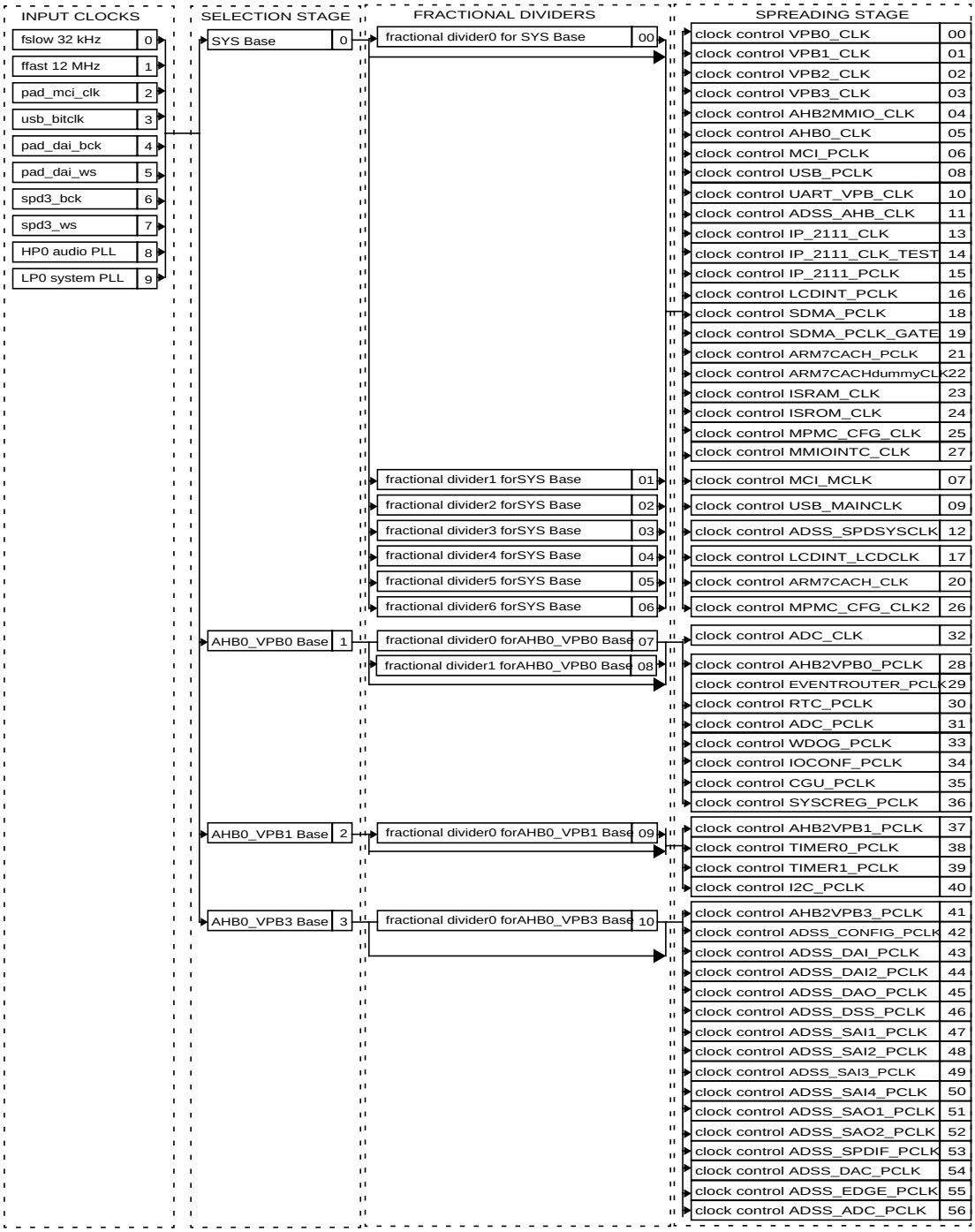
17.4.1 Overview Switchbox Module

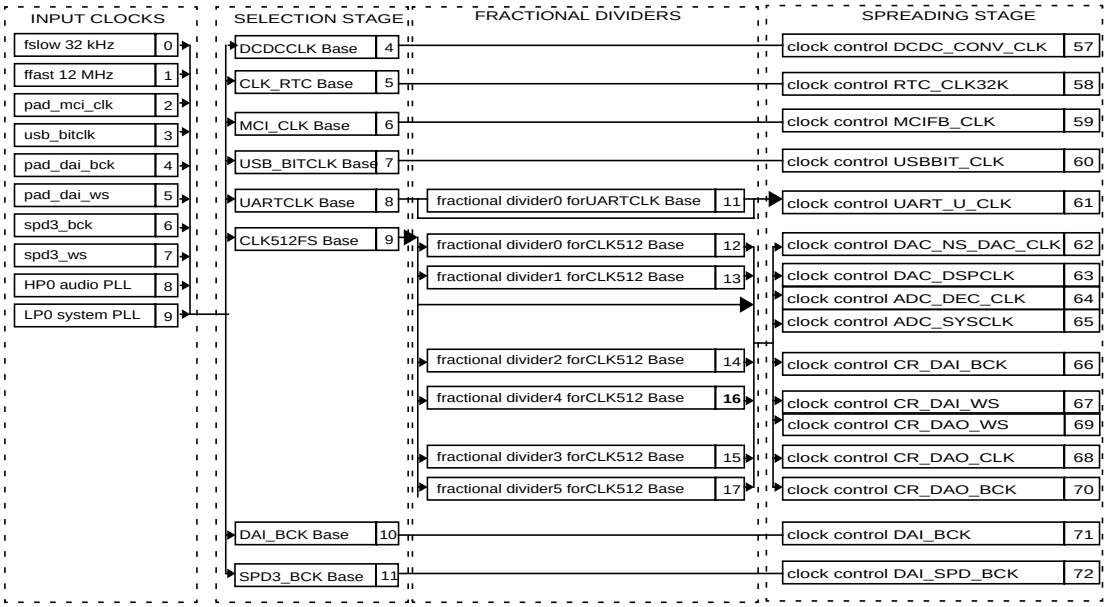
The switchbox is composed of:

- A selection stage which allows a selection between a number of reference into a number of base frequencies.
- A spreading stage which for each base frequency provides individual enabling towards a set of module clocks

These two stages are controlled via VPB configuration registers.

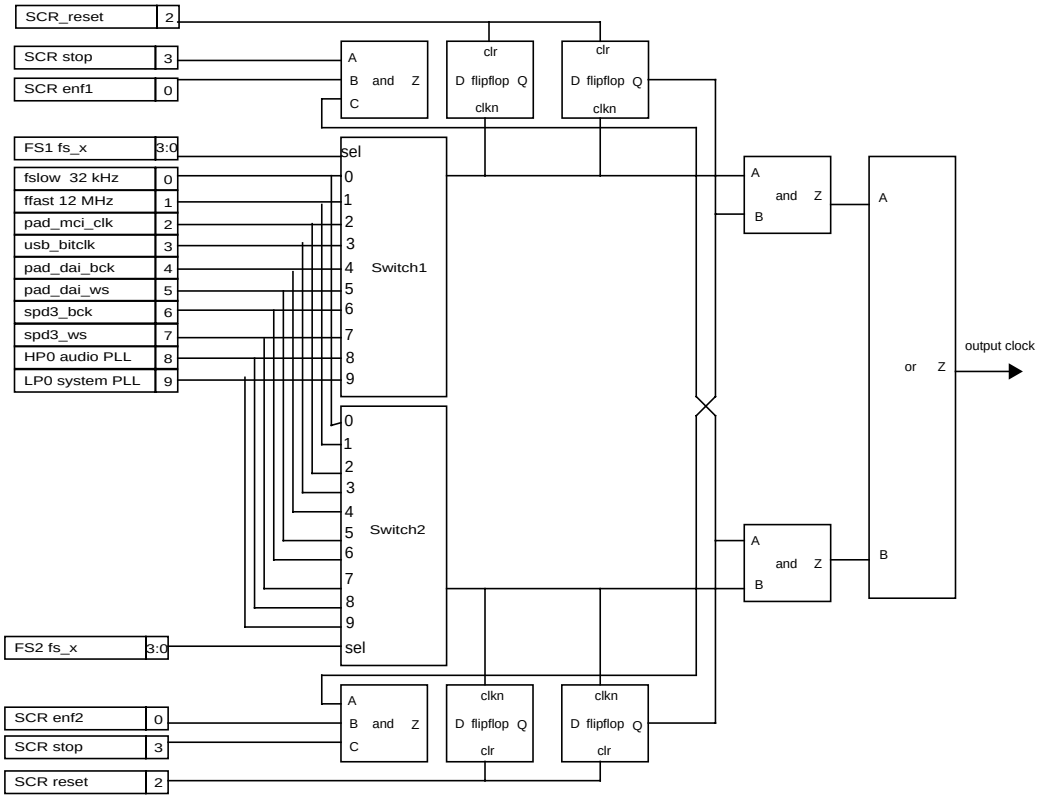
The following block diagram describes the configuration of the switchbox for PNX0101:


INPUT CLOCKS
fslow 32 kHz 0
ffast 12 MHz 1
pad_mci_clk 2
usb_bitclk 3
pad_dai_bck 4
pad_dai_ws 5
spd3_bck 6
spd3_ws 7
HP0 audio PLL 8
LP0 system PLL 9
SELECTION STAGE
SYS Base 0
AHB0_VPB0 Base 1
AHB0_VPB1 Base 2
AHB0_VPB3 Base 3
FRACTIONAL DIVIDERS
fractional divider0 for SYS Base 00
fractional divider1 forSYS Base 01
fractional divider2 forSYS Base 02
fractional divider3 forSYS Base 03
fractional divider4 forSYS Base 04
fractional divider5 forSYS Base 05
fractional divider6 forSYS Base 06
fractional divider0 forAHB0_VPB0 Base 07
fractional divider1 forAHB0_VPB0 Base 08
fractional divider0 forAHB0_VPB1 Base 09
fractional divider0 forAHB0_VPB3 Base 10
SPREADING STAGE
clock control VPB0_CLK 00
clock control VPB1_CLK 01
clock control VPB2_CLK 02
clock control VPB3_CLK 03
clock control AHB2MMIO_CLK 04
clock control AHB0_CLK 05
clock control MCI_PCLK 06
clock control USB_PCLK 08
clock control UART_VPB_CLK 10
clock control ADSS_AHB_CLK 11
clock control IP_2111_CLK 13
clock control IP_2111_CLK_TEST 14
clock control IP_2111_PCLK 15
clock control LCDINT_PCLK 16
clock control SDMA_PCLK 18
clock control SDMA_PCLK_GATE 19
clock control ARM7CACH_PCLK 21
clock control ARM7CACHdummyCLK 22
clock control ISRAM_CLK 23
clock control ISROM_CLK 24
clock control MPMC_CFG_CLK 25
clock control MMIOINTC_CLK 27
clock control MCI_MCLK 07
clock control USB_MAINCLK 09
clock control ADSS_SPDSYSCLK 12
clock control LCDINT_LCDCLK 17
clock control ARM7CACH_CLK 20
clock control MPMC_CFG_CLK2 26
clock control ADC_CLK 32
clock control AHB2VPB0_PCLK 28
clock control EVENTROUTER_PCLK 29
clock control RTC_PCLK 30
clock control ADC_PCLK 31
clock control WDOG_PCLK 33
clock control IOCONF_PCLK 34
clock control CGU_PCLK 35
clock control SYSCREG_PCLK 36
clock control AHB2VPB1_PCLK 37
clock control TIMER0_PCLK 38
clock control TIMER1_PCLK 39
clock control I2C_PCLK 40
clock control AHB2VPB3_PCLK 41
clock control ADSS_CONFIG_PCLK 42
clock control ADSS_DAI_PCLK 43
clock control ADSS_DAI2_PCLK 44
clock control ADSS_DAO_PCLK 45
clock control ADSS_DSS_PCLK 46
clock control ADSS_SAI1_PCLK 47
clock control ADSS_SAI2_PCLK 48
clock control ADSS_SAI3_PCLK 49
clock control ADSS_SAI4_PCLK 50
clock control ADSS_SAO1_PCLK 51
clock control ADSS_SAO2_PCLK 52
clock control ADSS_SPDIF_PCLK 53
clock control ADSS_DAC_PCLK 54
clock control ADSS_EDGE_PCLK 55
clock control ADSS_ADC_PCLK 56


INPUT CLOCKS
SELECTION STAGE
FRACTIONAL DIVIDERS
SPREADING STAGE
fslow 32 kHz 0
ffast 12 MHz 1
pad_mci_clk 2
usb_bitclk 3
pad_dai_bck 4
pad_dai_ws 5
spd3_bck 6
spd3_ws 7
HP0 audio PLL 8
LP0 system PLL 9
DCDCCLK Base 4
CLK_RTC Base 5
MCI_CLK Base 6
USB_BITCLK Base 7
UARTCLK Base 8
CLK512FS Base 9
DAI_BCK Base 10
SPD3_BCK Base 11
fractional divider0 forUARTCLK Base 11
fractional divider0 forCLK512 Base 12
fractional divider1 forCLK512 Base 13
fractional divider2 forCLK512 Base 14
fractional divider4 forCLK512 Base 16
fractional divider3 forCLK512 Base 15
fractional divider5 forCLK512 Base 17
clock control DCDC_CONV_CLK 57
clock control RTC_CLK32K 58
clock control MCIFB_CLK 59
clock control USBBIT_CLK 60
clock control UART_U_CLK 61
clock control DAC_NS_DAC_CLK 62
clock control DAC_DSPCLK 63
clock control ADC_DEC_CLK 64
clock control ADC_SYSCLK 65
clock control CR_DAI_BCK 66
clock control CR_DAI_WS 67
clock control CR_DAO_WS 69
clock control CR_DAO_CLK 68
clock control CR_DAO_BCK 70
clock control DAI_BCK 71
clock control DAI_SPD_BCK 72

17.4.2 Selection Stage

Selection muxes towards switches allow each reference frequency to be passed to each base frequency. Frequency switches allow safe run-time changes of base frequency selection.



A two path switch is used because of the absence of guaranteed clocks, making the use of state machines difficult. The frequency selector uses two multiplexers on all the reference frequencies inputs resulting in the frequency F1 and F2. The reason for having two multiplexers is to avoid glitches. Glitches can occur either when multiplexing or because the two frequencies are asynchronous (which they usually are) and the switching results in pulse clipping. The switch as implemented above resolves both these issues. The trick being that the new frequency is first selected on the multiplexer towards the switch side which is not activated. Only then will the switching take place, by both removing the enable on one side and activating the enable on the other side. The construction of the switch is such that first base_clk (the base frequency) is stopped when its level is low and after (at least) a complete period, at the falling edge of the new frequency, base_clk will start to run again with this new frequency. The delay of at least a complete period in both the disabling and enabling ensures that no pulse width violations can occur.

Because it takes a certain amount of time to switch between F1 and F2, care must be taken when switching to or from very low-speed clocks such as 32kHz. There may be a significant delay, in ARM clock cycles, between the clock switch programming and the actual clock activation.

The SSR register can be used by software to wait for a clock switch to complete. If an active F1 or F2 is externally stopped (by stopping a PLL or as result of external activity) then the switch will enter a deadlock. (e.g. If f2enabled is high, and F2 is stopped, then you cannot switch to F1.) Although this should never happen in practice, it is possible to recover from

this deadlock. This deadlock can be detected (after a software time-out) by looking at the SSR register, both F1STAT and F2STAT will be ‘0’. To recover from this deadlock, the frequency switch must be reset in software using the RESET bit of the SCR register.

All frequency select registers FS1 and FS2 in the clockswitchbox are reset to the external ‘fsboot’ value. fsboot can be 0 to N-1, where N is the number of reference frequencies.

17.4.3 Spreading Stage

- Each base frequency can be used to drive a set of module clocks.
- Each clock has its own enabling which can be controlled by a selectable fractional divider, via an (optional) external enable input, by its configuration register or with the wakeup signal.
- Fractional dividers on a base can be synchronized.
- In test mode base clocks are overruled by a test clock.
- Positive and inverted clocks sharing the same enabling controls.

17.4.3.1 Fractional dividers

Fractional dividers give the possibility to generate derivatives of a base frequency. A derivative is generated by enabling/masking clock pulses and optionally stretching these pulses to obtain 50% duty cycle clock approximations can be done by software control.
The fraction n/m must always be smaller than one and greater than zero.

- When using clock stretching the fraction must be smaller or equal to 1/2.
- To obtain the best possible 50% duty cycle when clock stretching is used, n/m should equal a division by a 2 power value (i.e. 1/2, 1/4, 1/8, ..). Using other fractions will result in a best approximation.
- To minimize power consumption madd and msub should be chosen to have as many trailing zero’s as possible, i.e. shift values left until the bitwidth boundary reached.
- In case of multiple fractional dividers exist on a base and they need to run in sync use the base control register fd_run bit to disable the fractional dividers. Then program the dividers and set the base control register fd_run bit to high. This ensures that all fractional dividers on this base will start running at the same instant.

In order to minimize power consumption madd and msub must be as large as possible. The limit of their values is determined by the madd/msub bitwidth.

17.4.3.2 External enabling

External enabling provides the ability to use signals from outside the switchbox to enable module clocks. These signals are latched with the base reference to ensure correct timing.
This functionality is typically used to reduce power consumption by disabling a clock whenever it is not required. It allows clock control from module level as opposed to the power control register which represents clock control from the system level. Figure x shows for which signals this has been implemented.

17.4.3.3 Wakeup feature

To support power modes a clock can be made controllable with the wakeup signal via its PCR wakeup_en bit. When active this clock will then, together with all other clocks which have the wakeup_en bit set, be disabled when wakeup becomes low and enabled when wakeup becomes high.
Through the wakeup feature a group of clocks can be made controllable with a single signal, wakeup. The intention for this signal is that it is used in combination with an wakeup event detection mechanism and a power mode selection which will then disable clocks which are wakeup enabled until a wakeup event has occurred.

Clock enabling/disabling takes two to three base periods to take effect.

The wakeup feature will be disabled for a clock when its auto bit in its pcr register is set to ‘0’.

17.4.4 Maximum speed possible on PNX0101_N1C

Table 32 Maximum speed possible on PNX0101

	WORST CASE	NOMINAL CASE	BEST CASE
Corner Voltage (V) Temperature (C)	ss 1.65 125	typ (*1.47) 1.8 25	ff (*2.19) 1.95 -40
Max SYS_BASE frequency (MHz)	57.2	84.1	125.3
Max AHB0_VPB0_ASYNC_PCLK_BASE frequency (MHz)	47.8	70.2	104.7
Max AHB0_VPB1_ASYNC_PCLK_BASE frequency (MHz)	96.7	142.1	211.8
Max AHB0_VPB3_ASYNC_PCLK_BASE frequency (MHz)	77.2	113.5	169.1
Max DCDCCLK_12MHZ_BASE frequency (MHz)		12	
Max CLK_RTC_BASE frequency (kHz)		32	
Max XT_MCI_CLK_BASE frequency (MHz)		4	
Max USBAPB_TOP_USB_BITCLK_BASE frequency (MHz)		12	
Max UARTCLK_BASE frequency (MHz)	109	160.2	238.7
Max CLK512FS_BASE frequency (MHz)	77.2	113.5	169
Max DAI_BCK_BASE frequency (MHz)		49.15	
Max SPD3_BCK_BASE frequency (MHz)		12.28	

The maximum frequencies of the clocks in the PNX0101 are depending on process parameters, voltage and temperature.

17.4.5 Debug mode

For evaluation modes and scantest will the defining of the clock frequency be different from application mode.

The different evaluation modes are:

- Bypass of the PLL’s
- Observing clocks
- Controlling/observing of oscillators

For information of the evaluation modes should be looked at the evaluation document.

17.5 Registers

Table 33 PNX0101 CGU Switchbox Peripheral Address Map

BASE	OFFSET	REG NAME	OFFSET	REG NAME
0x8000_4000 0x8000_4BFF	0	SCR_OFFSET (+ baseid)	0X000	SCR for SYS_BASE
			0X004	SCR for AHB0_VPB0_ASYNC_PCLK_BASE
			0X008	SCR for AHB0_VPB1_ASYNC_PCLK_BASE
			0X00C	SCR for AHB0_VPB3_ASYNC_PCLK_BASE
			0X010	SCR for DCDCCLK_12MHZ_BASE
			0X014	SCR for CLK_RTC_BASE
			0X018	SCR for XT_MCI_CLK_BASE
			0X01C	SCR for USBAPB_TOP_USB_BITCLK_BASE
			0X020	SCR for UARTCLK_BASE
			0X024	SCR for CLK512FS_BASE
			0X028	SCR for DAI_BCK_BASE
			0X02C	SCR for SPD3_BCK_BASE
	12	FS1_OFFSET (+ baseid)	0X030	FS1 for SYS_BASE
			0X034	FS1 for AHB0_VPB0_ASYNC_PCLK_BASE
			0X038	FS1 for AHB0_VPB1_ASYNC_PCLK_BASE
			0X03C	FS1 for AHB0_VPB3_ASYNC_PCLK_BASE
			0X040	FS1 for DCDCCLK_12MHZ_BASE
			0X044	FS1 for CLK_RTC_BASE
			0X048	FS1 for XT_MCI_CLK_BASE
			0X04C	FS1 for USBAPB_TOP_USB_BITCLK_BASE
			0X050	FS1 for UARTCLK_BASE
			0X054	FS1 for CLK512FS_BASE
			0X058	FS1 for DAI_BCK_BASE
			0X05C	FS1 for SPD3_BCK_BASE
	24	FS2_OFFSET (+ baseid)	0X060	FS2 for SYS_BASE
			0X064	FS2 for AHB0_VPB0_ASYNC_PCLK_BASE
			0X068	FS2 for AHB0_VPB1_ASYNC_PCLK_BASE
			0X06C	FS2 for AHB0_VPB3_ASYNC_PCLK_BASE
			0X070	FS2 for DCDCCLK_12MHZ_BASE
			0X074	FS2 for CLK_RTC_BASE
			0X078	FS2 for XT_MCI_CLK_BASE
			0X07C	FS2 for USBAPB_TOP_USB_BITCLK_BASE
			0X080	FS2 for UARTCLK_BASE
			0X084	FS2 for CLK512FS_BASE
			0X088	FS2 for DAI_BCK_BASE
			0X08C	FS2 for SPD3_BCK_BASE

BASE	OFFSET	REG NAME	OFFSET	REG NAME
	36	SSR_OFFSET (+ baseid)	0X090	SSR for SYS_BASE
			0X094	SSR for AHB0_VPB0_ASYNC_PCLK_BASE
			0X098	SSR for AHB0_VPB1_ASYNC_PCLK_BASE
			0X09C	SSR for AHB0_VPB3_ASYNC_PCLK_BASE
			0X0A0	SSR for DCDCCLK_12MHZ_BASE
			0X0A4	SSR for CLK_RTC_BASE
			0X0A8	SSR for XT_MCI_CLK_BASE
			0X0AC	SSR for USBAPB_TOP_USB_BITCLK_BASE
			0X0B0	SSR for UARTCLK_BASE
			0X0B4	SSR for CLK512FS_BASE
			0X0B8	SSR for DAI_BCK_BASE
			0X0BC	SSR for SPD3_BCK_BASE
	48	PCR_OFFSET (+ clkid)	0X0C0	PCR for VPB0_CLK
			0X0C4	PCR for VPB1_CLK
			0X0C8	PCR for VPB2_CLK
			0X0CC	PCR for VPB3_CLK
			0X0D0	PCR for AHB2MMIO_CLK
			0X0D4	PCR for AHB0_CLK
			0X0D8	PCR for MCI_PL180_PCLK
			0X0DC	PCR for MCI_PL180_MCLK
			0X0E0	PCR for UABABP_TOP_PCLK
			0X0E4	PCR for UABAPB_TOP_MAINCLK
			0X0E8	PCR for UART_VPB_CLK
			0X0EC	PCR for ADSS_AHB_CLK
			0X0F0	PCR for ADSS_SPD_SYSCLK
			0X0F4	PCR for IP_BLAS_2111_CLK
			0X0F8	PCR for IP_BLAS_2111_CLK_TESTSHELL
			0X0FC	PCR for IP_BLAS_2111_PCLK
			0X100	PCR for LCD_INTERFACE_PCLK
			0X104	PCR for LCD_INTERFACE_LCDCLK
			0X108	PCR for SIMPLE_DMA_PCLK
			0X10C	PCR for SIMPLE_DMA_CLK_GATED
			0X110	PCR for ARM7TDMISCACHE_CLK
			0X114	PCR for ARM7TDMISCACHE_PCLK
			0X118	PCR for ARM7TDMISCACHE_DUMMY_HCLK
			0X11C	PCR for ISRAM_CLK
			0X120	PCR for ISROM_CLK
			0X124	PCR for AHB_MPMC_PL172_CFG_CLK
			0X128	PCR for AHB_MPMC_PL172_CFG_CLK2
			0X12C	PCR for MMIOINTC_CLK

BASE	OFFSET	REG NAME	OFFSET	REG NAME
			0X130	PCR for AHB2VPB0_ASYNC_PCLK
			0X134	PCR for EVENT_ROUTER_PCLK
			0X138	PCR for RTC_PCLK
			0X13C	PCR for ADC_PCLK
			0X140	PCR for ADC_CLK
			0X144	PCR for WDOG_PCLK
			0X148	PCR for IOCONF_PCLK
			0X14C	PCR for CGU_PCLK
			0X150	PCR for SYSCREG_PCLK
			0X154	PCR for AHB2VPB1_ASYNC_PCLK
			0X158	PCR for TIMER0_PCLK
			0X15C	PCR for TIMER1_PCLK
			0X160	PCR for I2C_PCLK
			0X164	PCR for AHB2VPB3_ASYNC_PCLK
			0X168	PCR for ADSS_CONFIG_PCLK
			0X16C	PCR for ADSS_DAI_PCLK
			0X170	PCR for ADSS_DAI2_PCLK
			0X174	PCR for ADSS_DAO_PCLK
			0X178	PCR for ADSS_DSS_PCLK
			0X17C	PCR for ADSS_SAI1_PCLK
			0X180	PCR for ADSS_SAI2_PCLK
			0X184	PCR for ADSS_SAI3_PCLK
			0X188	PCR for ADSS_SAI4_PCLK
			0X18C	PCR for ADSS_SAO1_PCLK
			0X190	PCR for ADSS_SAO2_PCLK
			0X194	PCR for ADSS_SPDIF_PCLK
			0X198	PCR for ADSS_DAC_IPOL_PCLK
			0X19C	PCR for ADSS_EDGE_DET_PCLK
			0X1A0	PCR for ADSS_ADC_DEC_PCLK
			0X1A4	PCR for DCDC_CONVERTER_CLK
			0X1A8	PCR for RTC_CLK32K
			0X1AC	PCR for MCI_PL180_MCIFBCLK
			0X1B0	PCR for USBAPB_TOP_USB_BITCLK
			0X1B4	PCR for UART_U_CLK
			0X1B8	PCR for ADSS_DAC_NS_DAC_CLK
			0X1BC	PCR for ADSS_DAC_DSPCLK
			0X1C0	PCR for ADSS_ADC_DECCLK
			0X1C4	PCR for ADSS_ADC_SYSCLK
			0X1C8	PCR for CR_DAI_BCK
			0X1CC	PCR for CR_DAI_WS
			0X1D0	PCR for CR_DAO_CLK
			0X1D4	PCR for CR_DAO_WS
			0X1D8	PCR for CR_DAO_BCK
			0X1DC	PCR for DAI_BCK
			0X1E0	PCR for ADSS_DAI_SPD_BCK

BASE	OFFSET	REG NAME	OFFSET	REG NAME
	121	PSR_OFFSET (+ clkid)	0X1E4	PSR for VPB0_CLK
			0X1E8	PSR for VPB1_CLK
			0X1EC	PSR for VPB2_CLK
			0X1F0	PSR for VPB3_CLK
			0X1F4	PSR for AHB2MMIO_CLK
			0X1F8	PSR for AHB0_CLK
			0X1FC	PSR for MCI_PL180_PCLK
			0X200	PSR for MCI_PL180_MCLK
			0X204	PSR for UABABP_TOP_PCLK
			0X208	PSR for UABAPB_TOP_MAINCLK
			0X20C	PSR for UART_VPB_CLK
			0X210	PSR for ADSS_AHB_CLK
			0X214	PSR for ADSS_SPD_SYSCLK
			0X218	PSR for IP_BLAS_2111_CLK
			0X21C	PSR for IP_BLAS_2111_CLK_TESTSHELL
			0X220	PSR for IP_BLAS_2111_PCLK
			0X224	PSR for LCD_INTERFACE_PCLK
			0X228	PSR for LCD_INTERFACE_LCDCLK
			0X22C	PSR for SIMPLE_DMA_PCLK
			0X230	PSR for SIMPLE_DMA_CLK_GATED
			0X234	PSR for ARM7TDMISCACHE_CLK
			0X238	PSR for ARM7TDMISCACHE_PCLK
			0X23C	PSR for ARM7TDMISCACHE_DUMMY_HCLK
			0X240	PSR for ISRAM_CLK
			0X244	PSR for ISROM_CLK
			0X248	PSR for AHB_MPMC_PL172_CFG_CLK
			0X24C	PSR for AHB_MPMC_PL172_CFG_CLK2
			0X250	PSR for MMIOINTC_CLK
			0X254	PSR for AHB2VPB0_ASYNC_PCLK
			0X258	PSR for EVENT_ROUTER_PCLK
			0X25C	PSR for RTC_PCLK
			0X260	PSR for ADC_PCLK
			0X264	PSR for ADC_CLK
			0X268	PSR for WDOG_PCLK
			0X26C	PSR for IOCONF_PCLK
			0X270	PSR for CGU_PCLK
			0X274	PSR for SYSCREG_PCLK
			0X278	PSR for AHB2VPB1_ASYNC_PCLK
			0X27C	PSR for TIMER0_PCLK
			0X280	PSR for TIMER1_PCLK
			0X284	PSR for I2C_PCLK

BASE	OFFSET	REG NAME	OFFSET	REG NAME
			0X288	PSR for ADSS_DAI2_PCLK
			0X28C	PSR for ADSS_DAO_PCLK
			0X290	PSR for ADSS_DSS_PCLK
			0X294	PSR for ADSS_SAI1_PCLK
			0X298	PSR for ADSS_SAI2_PCLK
			0X29C	PSR for ADSS_SAI3_PCLK
			0X2A0	PSR for ADSS_SAI4_PCLK
			0X2A4	PSR for ADSS_SAO1_PCLK
			0X2A8	PSR for ADSS_SAO2_PCLK
			0X2AC	PSR for ADSS_SPDIF_PCLK
			0X2B0	PSR for ADSS_SAO1_PCLK
			0X2B4	PSR for ADSS_SAO2_PCLK
			0X2B8	PSR for ADSS_SPDIF_PCLK
			0X2BC	PSR for ADSS_DAC_IPOL_PCLK
			0X2C0	PSR for ADSS_EDGE_DET_PCLK
			0X2C4	PSR for ADSS_ADC_DEC_PCLK
			0X2C8	PSR for DCDC_CONVERTER_CLK
			0X2CC	PSR for RTC_CLK32K
			0X2D0	PSR for MCI_PL180_MCIFBCLK
			0X2D4	PSR for USBAPB_TOP_USB_BITCLK
			0X2D8	PSR for UART_U_CLK
			0X2DC	PSR for ADSS_DAC_NS_DAC_CLK
			0X2E0	PSR for ADSS_DAC_DSPCLK
			0X2E4	PSR for ADSS_ADC_DECCLK
			0X2E8	PSR for ADSS_ADC_SYSCLK
			0X2EC	PSR for CR_DAI_BCK
			0X2F0	PSR for CR_DAI_WS
			0X2F4	PSR for CR_DAO_CLK
			0X2F8	PSR for CR_DAO_WS
			0X2FC	PSR for CR_DAO_BCK
			0X300	PSR for DAI_BCK
			0X304	PSR for ADSS_DAI_SPD_BCK

BASE	OFFSET	REG NAME	OFFSET	REG NAME
	194	ESR_OFFSET (+ clkid)	0X308	ESR for VPB0_CLK
			0X30C	ESR for VPB1_CLK
			0X310	ESR for VPB2_CLK
			0X314	ESR for VPB3_CLK
			0X318	ESR for AHB2MMIO_CLK
			0X31C	ESR for AHB0_CLK
			0X320	ESR for MCI_PL180_PCLK
			0X324	ESR for MCI_PL180_MCLK
			0X328	ESR for UABABP_TOP_PCLK
			0X32C	ESR for UABAPB_TOP_MAINCLK
			0X330	ESR for UART_VPB_CLK
			0X334	ESR for ADSS_AHB_CLK
			0X338	ESR for ADSS_SPD_SYSCLK
			0X33C	ESR for IP_BLAS_2111_CLK
			0X340	ESR for IP_BLAS_2111_CLK_TESTSHELL
			0X344	ESR for IP_BLAS_2111_PCLK
			0X348	ESR for LCD_INTERFACE_PCLK
			0X34C	ESR for LCD_INTERFACE_LCDCLK
			0X350	ESR for SIMPLE_DMA_PCLK
			0X354	ESR for SIMPLE_DMA_CLK_GATED
			0X358	ESR for ARM7TDMISCACHE_CLK
			0X35C	ESR for ARM7TDMISCACHE_PCLK
			0X360	ESR for ARM7TDMISCACHE_DUMMY_HCLK
			0X364	ESR for ISRAM_CLK
			0X368	ESR for ISROM_CLK
			0X36C	ESR for AHB_MPMC_PL172_CFG_CLK
			0X370	ESR for AHB_MPMC_PL172_CFG_CLK2
			0X374	ESR for MMIOINTC_CLK
			0X378	ESR for AHB2VPB0_ASYNC_PCLK
			0X37C	ESR for EVENT_ROUTER_PCLK
			0X380	ESR for RTC_PCLK
			0X384	ESR for ADC_PCLK
			0X388	ESR for ADC_CLK
			0X38C	ESR for WDOG_PCLK
			0X390	ESR for IOCONF_PCLK
			0X394	ESR for CGU_PCLK
			0X398	ESR for SYSCREG_PCLK
			0X3AC	ESR for AHB2VPB1_ASYNC_PCLK
			0X3A0	ESR for TIMER0_PCLK
			0X3A4	ESR for TIMER1_PCLK
			0X3A8	ESR for I2C_PCLK

BASE	OFFSET	REG NAME	OFFSET	REG NAME
			0X3AC	ESR for AHB2VPB3_ASYNC_PCLK
			0X3B0	ESR for ADSS_CONFIG_PCLK
			0X3B4	ESR for ADSS_DAI_PCLK
			0X3B8	ESR for ADSS_DAI2_PCLK
			0X3BC	ESR for ADSS_DAO_PCLK
			0X3C0	ESR for ADSS_DSS_PCLK
			0X3C4	ESR for ADSS_SAI1_PCLK
			0X3C8	ESR for ADSS_SAI2_PCLK
			0X3CC	ESR for ADSS_SAI3_PCLK
			0X3D0	ESR for ADSS_SAI4_PCLK
			0X3D4	ESR for ADSS_SAO1_PCLK
			0X3D8	ESR for ADSS_SAO2_PCLK
			0X3DC	ESR for ADSS_SPDIF_PCLK
			0X3E0	ESR for ADSS_DAC_IPOL_PCLK
			0X3E4	ESR for ADSS_EDGE_DET_PCLK
			0X3E8	ESR for ADSS_ADC_DEC_PCLK
			0X3EC	ESR for UART_U_CLK
			0X3F0	ESR for ADSS_DAC_NS_DAC_CLK
			0X3F4	ESR for ADSS_DAC_DSPCLK
			0X3F8	ESR for ADSS_ADC_DECCLK
			0X3FC	ESR for ADSS_ADC_SYSCLK
			0X400	ESR for CR_DAI_BCK
			0X404	ESR for CR_DAI_WS
			0X408	ESR for CR_DAO_CLK
			0X40C	ESR for CR_DAO_WS
			0X410	ESR for CR_DAO_BCK
	261	BCR_OFFSET (+ bcrid)	0X414	BCR for SYS_BASE
			0X418	BCR for AHB0_VPB0_ASYNC_PCLK_BASE
			0X41C	BCR for CLK512FS_BASE
	264	FDC_OFFSET (+ fdid)	0X420	FDC for FRACDIV0 (SYS_BASE)
			0X424	FDC for FRACDIV1 (SYS_BASE)
			0X428	FDC for FRACDIV2 (SYS_BASE)
			0X42C	FDC for FRACDIV3 (SYS_BASE)
			0X430	FDC for FRACDIV4 (SYS_BASE)
			0X434	FDC for FRACDIV5 (SYS_BASE)
			0X438	FDC for FRACDIV6 (SYS_BASE)
			0X43C	FDC for FRACDIV7 (AHB0_VPB0_ASYNC_PCLK_BASE)
			0X440	FDC for FRACDIV8 (AHB0_VPB0_ASYNC_PCLK_BASE)
			0X444	FDC for FRACDIV9 (AHB0_VPB1_ASYNC_PCLK_BASE)
			0X448	FDC for FRACDIV10 (AHB0_VPB3_ASYNC_PCLK_BASE)
			0X44C	FDC for FRACDIV11 (UARTCLK_BASE)
			0X450	FDC for FRACDIV12 (CLK512FS_BASE)
			0X454	FDC for FRACDIV13 (CLK512FS_BASE)
			0X458	FDC for FRACDIV14 (CLK512FS_BASE)
			0X45C	FDC for FRACDIV15 (CLK512FS_BASE)
			0X460	FDC for FRACDIV16 (CLK512FS_BASE)
			0X464	FDC for FRACDIV17 (CLK512FS_BASE)

Remark about clock switching:

On the figure it is in the CGU configuration registers (pclk) communicate with all bases. E.g. base 0 communicate with base1 (clock crossing). The clock crossing is not real problem for application because it are more or less static signals. So as general point if the fracdiv is being configured (madd.msub,stretch -> pclk domain base 1) take care that the run bit from the fracdiv is disabled or that the bcr bit from the base is disabled (if you don't do that in sdf simulation a 'x' will be occur in system and in application).

There are also re synchronisation domains used between configuration domain and the bases. It is important the bcr bit sync is used for more fracdivs, as those bits are arriving can a problem occur for the sync's of the fracdivs. The solution in the cgu is risky but simulations with sdf shows correct behaviour (the edge of the base1 clock (pclk_cgu) occurring by base 0 (sdf simulations are going correct).

Table 34 SCR for <name>_BASE register

BIT	POR VALUE	NAME	DESCRIPTION
0	1	ENF1	Enable side #1 of switch
1	0	ENF2	Enable side #2 of switch
2	0	RESET	Asynchronous reset of both switches
3	cfg	STOP	forces switch in disable mode
31:4	Reserved	Reserved	

It is illegal for ENF1 and ENF2 to be high simultaneous as this might result in select oscillations of the switch between the two selection paths.

Table 35 FS1 for <name>_BASE register

BIT	POR VALUE	NAME	DESCRIPTION
FSR_WIDTH-1:0	fs boot value	FS1	The value of FS1 selects the input frequency for side #1 of the frequency switch. Its reset value is determined by the external fsboot input signal. At reset, this side of the switch is enabled.
31:FSR_WIDTH	Reserved	Reserved	

Table 36 FS2 for <name>_BASE register

BIT	POR VALUE	NAME	DESCRIPTION
FSR_WIDTH-1:0	0	FS2	The value of FS2 selects the input frequency for side #2 of the frequency switch. At reset, this side of the switch is disabled.
31:FSR_WIDTH	Reserved	Reserved	

Table 37 SSR for <name>_BASE register

BIT	POR VALUE	NAME	DESCRIPTION
0		F1STAT	If true side #1 is currently enabled.
1		F2STAT	If true side #2 is currently enabled.
FSR_WIDTH-1+2:2		FS	Feedback of currently used frequency selection.
31:FSR_WIDTH+2	Reserved	Reserved	

Constant: FSR_WIDTH = 4, fs boot value = 1 (ffast = 12 MHz)

Table 38 PCR for <name>_CLK register

BIT	POR VALUE	NAME	DESCRIPTION
0	1	RUN	When ‘0’ clock is disabled.
1	1	AUTO	If falseWakeup and External enable are overruled, only internal enabling via configuration and fractional divider remain active. This control is primary meant for debugging purposes as when it is set to zero the clock is no longer affected by power saving modes.
2	1	WAKE_EN	If ‘0’ wake up is overruled and the module clock remains active when wakeup is low. This control exists to support for power down modes. With the input signal ‘wakeup’ all the clocks which have wake_en set can be centrally controlled, when wakeup becomes low the clocks will be disabled, when wakeup is set high they will be enabled.
3	0	EXTEN_EN	Enable external enabling (= enable generated from outside module). This control is optional (configurable through HDLi), it allows a clock to be controlled by an input signal which the template names ‘<clockname>_enable’. An example are ‘pclk’ of VPB busses used for configuration registers, these need only to be active when the register is accessed and can be controlled by the VPB psel (only for 3 clock cycle accesses) signal through the external enable input. (can be used for ioconf_pclk, cgu_pclk, syscreg_pclk, ip_blas_2111_pclk, ccp_event_router_pclk, simple_dma_clk_gated, arm7tdmiscache_pclk, ssa1_adc_PCLK) (should be kept zero (not used) for ssa1_mci_pl180_pclk, usbapb_top_PCLK, usbapb_top_mainclk, uart_vpb_clk, ssa1_lcd_interface_pclk, wdog_pclk, i2c_pclk, usbapb_top_USB_BitClk)
4	0	ENOUT_EN	If true the clk enabling preview signal <clk>_enableout reflects the enable state for the second active edge of <clk>. (only valid for ahb0_clk, arm7tdmiscache_dummy_hclk, isram_clk, isrom_clk)
31:5	Reserved	Reserved	

Table 39 PSR for <name>_CLK register

BIT	POR VALUE	NAME	DESCRIPTION
0		ACTIVE	Indicates clock is functional.
1		WAKEUP	Indicates the wakeup condition for this clock.
31:2	Reserved	Reserved	

Table 40 ESR for <name>_CLK register

BIT	POR VALUE	NAME	DESCRIPTION
0	configure	ESR_EN	If the ESR_EN is true an enable is generated from the fractional divider indexed by ESR_SEL.
log2(NR_FRACDIV):2	configure	ESR_SEL	When only one fractional divider has been assigned to the clock's base frequency ESR_SEL has 'zero' bits since there are no selection options.
31:log2(NR_FRACDIV) +1	Reserved	Reserved	

Note that the enable selection is towards the fractional dividers assigned to this base frequency only. Other fractional dividers are not accessible.

SYS Base NR_FRACDIV = 7;

AHB0_VPB0 Base NR_FRACDIV = 2;

AHB0_VPB1 Base NR_FRACDIV = 1;

AHB0_VPB3 Base NR_FRACDIV = 1;

UARTCLK Base NR_FRACDIV = 1;

CLK512FS Base NR_FRACDIV = 6;

Table 41 FDC for <name>_BASE register

BIT	POR VALUE	NAME	DESCRIPTION
0	configure	FDCTRL_RUN	Enables the fractional divider
1	configure	FDCTRL_RESET	Asynchronous reset of the fractional divider
2	configure	FDCTRL_STRETCH	Enables the stretching option. When stretching the generated clocks will have approximative 50% duty cycles
maddsubt_btw+2:3	configure	MADD	The madd and msub values can be calculated according: Fdiv = n/m * f Madd = m-n Msub = -n
maddsubt_btw*2+2:maddsubt_btw+3	configure	MSUB	
31:maddsubt_btw*2+3	Reserved	Reserved	

maddsubt_btw = 8 for all fracdivs except CLK512FS Base fracdiv 4 where maddsubt_btw= 10 this fracdiv should be used for generating ws.

Table 42 BCR for <name>_BASE register

BIT	POR VALUE	NAME	DESCRIPTION
0		FDRUN	When true fractional dividers belonging to this base are allowed to function. This bit overrules the run bit in the control register of the fractional dividers. So if fd_run is set low all fractional dividers will be disabled. Its purpose is to be able to activate all fractional dividers in a base simultaneously so that they run in sync.
31:1	Reserved	Reserved	

Table 43 PNX0101 CGU Clock Control Unit Address Map

BASE	OFFSET	REG NAME
0x8000_4C00	0x000	CGU_CONFIG_POWERMODE
0x8000_4FFF	0x004	CGU_CONFIG_WD_BARK
	0x008	CGU_CONFIG_F32KHZ_ON
	0x00c	CGU_CONFIG_F32KHZ_BYPASS
	0x010	CGU_CONFIG_FFAST_ON
	0x014	CGU_CONFIG_FFAST_BYPASS

BASE	OFFSET	REG NAME
	0x018	CGU_CONFIG_VPB0_RESETN_SOFT
	0x01c	CGU_CONFIG_AHB2VPB0_PNRES_SOFT
	0x020	CGU_CONFIG_VPB1_RESETN_SOFT
	0x024	CGU_CONFIG_AHB2VPB1_PNRES_SOFT
	0x028	CGU_CONFIG_VPB2_RESETN_SOFT
	0x02c	CGU_CONFIG_VPB3_RESETN_SOFT
	0x030	CGU_CONFIG_AHB2VPB3_PNRES_SOFT
	0x034	CGU_CONFIG_AHB2MMIO_RESETN_SOFT
	0x038	CGU_CONFIG_AHB0_RESETN_SOFT
	0x03C	CGU_CONFIG_SSA1_TIMER0_PNRES_SOFT
	0x040	CGU_CONFIG_SSA1_TIMER1_PNRES_SOFT
	0x044	CGU_CONFIG_SSA1_MCI_PL180_PNRES_SOFT
	0x048	CGU_CONFIG_SSA1_MCI_PL180_NMCIRST_SOFT
	0x04c	CGU_CONFIG_USBAPB_TOP_PNRES_SOFT
	0x050	CGU_CONFIG_UART_SYS_RST_AN_SOFT
	0x054	CGU_CONFIG_I2C_PNRES_SOFT
	0x058	CGU_CONFIG_MELODY_ADSS_HRESETN_SOFT
	0x05c	CGU_CONFIG_MELODY_ADSS_PNRES_SOFT
	0x060	CGU_CONFIG_MELODY_ADSS_AHBIF_RST_N_SOFT
	0x064	CGU_CONFIG_MELODY_ADSS_DAI_RST_N_SOFT
	0x068	CGU_CONFIG_MELODY_ADSS_DAI2_RST_N_SOFT
	0x06c	CGU_CONFIG_MELODY_ADSS_DAO_RST_N_SOFT
	0x070	CGU_CONFIG_MELODY_ADSS_DEC_RST_N_SOFT
	0x074	CGU_CONFIG_MELODY_ADSS_EDET_RST_N_SOFT
	0x078	CGU_CONFIG_MELODY_ADSS_INT_RST_N_SOFT
	0x07c	CGU_CONFIG_MELODY_ADSS_SAI1_RST_N_SOFT
	0x080	CGU_CONFIG_MELODY_ADSS_SAI2_RST_N_SOFT
	0x084	CGU_CONFIG_MELODY_ADSS_SAI3_RST_N_SOFT
	0x088	CGU_CONFIG_MELODY_ADSS_SAI4_RST_N_SOFT
	0x08c	CGU_CONFIG_MELODY_ADSS_SAO1_RST_N_SOFT
	0x090	CGU_CONFIG_MELODY_ADSS_SAO2_RST_N_SOFT
	0x094	CGU_CONFIG_MELODY_ADSS_SPDIF_RST_N_SOFT
	0x098	CGU_CONFIG_IP_BLAS_2111_HRESETN_SOFT
	0x09c	CGU_CONFIG_SSA1_LCD_INTERFACE_PNRES_SOFT
	0x0a0	CGU_CONFIG_SIMPLE_DMA_PNRES_SOFT
	0x0a4	CGU_CONFIG_AHB_MPMC_PL172_HRESETN_SOFT
	0x0a8	CGU_CONFIG_MMIOINTC_RESETN_SOFT

BASE	OFFSET	REG NAME
	0x0ac	CGU_CONFIG_HP0_FIN_SELECT
	0x0b0	CGU_CONFIG_HP0_MDEC
	0x0b4	CGU_CONFIG_HP0_NDEC
	0x0b8	CGU_CONFIG_HP0_PDEC
	0x0bc	CGU_CONFIG_HP0_MODE
	0x0c0	CGU_CONFIG_HP0_STATUS
	0x0c4	CGU_CONFIG_HP0_ACK
	0x0c8	CGU_CONFIG_HP0_REQ
	0x0cc	CGU_CONFIG_HP0_INSELR
	0x0d0	CGU_CONFIG_HP0_INSELI
	0x0d4	CGU_CONFIG_HP0_INSELP
	0x0d8	CGU_CONFIG_HP0_SELR
	0x0dc	CGU_CONFIG_HP0_SELI
	0x0e0	CGU_CONFIG_HP0_SELP
	0x0e4	CGU_CONFIG_LP0_FIN_SELECT
	0x0e8	CGU_CONFIG_LP0_PWD
	0x0ec	CGU_CONFIG_LP0_BYPASS
	0x0f0	CGU_CONFIG_LP0_LOCK
	0x0f4	CGU_CONFIG_LP0_DIRECT
	0x0f8	CGU_CONFIG_LP0_MSEL
	0x0fc	CGU_CONFIG_LP0_PSEL

Table 44 CGU_CONFIG_POWERMODE register

BIT	POR VALUE	NAME	DESCRIPTION
1:0	1	Powermode	00 Unsupported, results in unpredictable behavior.
			01 Normal operational mode.
			10 Unsupported, results in unpredictable behavior.
			11 Wakeup enabled clocks are disabled until a wakeup event occurs.
31:2	Reserved	Reserved	

Table 45 CGU_CONFIG_WD_BARK register

BIT	POR VALUE	NAME	DESCRIPTION
0	0	WD_BARK	Watch dog bark register. Is set when a watch dog reset has occurred (read only). This bit is cleared only by a power on reset.
31:1	Reserved	Reserved	

Table 46 CGU_CONFIG_F32KHZ_ON register

BIT	POR VALUE	NAME	DESCRIPTION
0	1	F32KHZ_ON	32kHz oscillator power-on.
31:1	Reserved	Reserved	

Table 47 CGU_CONFIG_F32KHZ_BYPASS register

BIT	POR VALUE	NAME	DESCRIPTION
0	0	F32KHZ_BYPASS	Bypass comparator in 32kHz oscillator, used for testmode only.
31:1	Reserved	Reserved	

Table 48 CGU_CONFIG_FFAST_ON register

BIT	POR VALUE	NAME	DESCRIPTION
0	1	FFFAST_ON	activate fast oscillator
31:1	Reserved	Reserved	

Table 49 CGU_CONFIG_FFAST_BYPASS register

BIT	POR VALUE	NAME	DESCRIPTION
0	0	FFFAST_BYPASS	Bypass comparator in fast oscillator -testmode only
31:1	Reserved	Reserved	

Table 50 CGU_CONFIG_<name>_RESETN_SOFT register

BIT	POR VALUE	NAME	DESCRIPTION
0	1	CGU_CONFIG_VPB0_RESETN_SOFT	reserved (reset of vpb0 bridge, cgu, syscreg, event_router, rtc, adc10b, wdog, ioconf)
0	1	CGU_CONFIG_AHB2VPB0_PNRES_SOFT	reserved (reset of vpb0 bridge)
0	1	CGU_CONFIG_VPB1_RESETN_SOFT	reset of vpb1 bridge
0	1	CGU_CONFIG_AHB2VPB1_PNRES_SOFT	reset of vpb1 bridge
0	1	CGU_CONFIG_VPB2_RESETN_SOFT	reset of vpb2 bridge
0	1	CGU_CONFIG_VPB3_RESETN_SOFT	reset of vpb3 bridge
0	1	CGU_CONFIG_AHB2VPB3_PNRES_SOFT	reset of vpb3 bridge
0	1	CGU_CONFIG_AHB2MMIO_RESETN_SOFT	reset of intc interface (1) (mmio_intc)
0	1	CGU_CONFIG_AHB0_RESETN_SOFT	reserved (reset of ahb0 domain, arm7, isram interface, isrom interface)
0	1	CGU_CONFIG_SSA1_TIMER0_PNRES_SOFT	reset of timer0 block
0	1	CGU_CONFIG_SSA1_TIMER1_PNRES_SOFT	reset of timer1 block
0	1	CGU_CONFIG_SSA1_MCI_PL180_PNRES_SOFT	reset of mci
0	1	CGU_CONFIG_SSA1_MCI_PL180_NMCIRST_SOFT	reset of mci
0	1	CGU_CONFIG_USBAPB_TOP_PNRES_SOFT	reset of usb block
0	1	CGU_CONFIG_UART_SYS_RST_AN_SOFT	reset of uart block
0	1	CGU_CONFIG_I2C_PNRES_SOFT	reset of i2c block
0	1	CGU_CONFIG_MELODY_ADSS_HRESETN_SOFT	reset of ADSS_AHBIF hclk
0	1	CGU_CONFIG_MELODY_ADSS_PNRES_SOFT	reset of adss config block
0	1	CGU_CONFIG_MELODY_ADSS_AHBIF_RST_N_SOFT	reset of ADSS_AHBIF dss_pclk
0	1	CGU_CONFIG_MELODY_ADSS_DAI_RST_N_SOFT	reset of ADSS_DAI block
0	1	CGU_CONFIG_MELODY_ADSS_DAI2_RST_N_SOFT	reset of ADSS_DAI2 block
0	1	CGU_CONFIG_MELODY_ADSS_DAO_RST_N_SOFT	reset of ADSS_DAO block
0	1	CGU_CONFIG_MELODY_ADSS_DEC_RST_N_SOFT	reset of ADSS_DECIMATOR block
0	1	CGU_CONFIG_MELODY_ADSS_EDET_RST_N_SOFT	reset of ADSS_EDGE DETECTION block
0	1	CGU_CONFIG_MELODY_ADSS_INT_RST_N_SOFT	reset of ADSS_INTERPOLATOR block
0	1	CGU_CONFIG_MELODY_ADSS_SAI1_RST_N_SOFT	reset of ADSS_SAI1 block
0	1	CGU_CONFIG_MELODY_ADSS_SAI2_RST_N_SOFT	reset of ADSS_SAI2 block
0	1	CGU_CONFIG_MELODY_ADSS_SAI3_RST_N_SOFT	reset of ADSS_SAI3 block
0	1	CGU_CONFIG_MELODY_ADSS_SAI4_RST_N_SOFT	reset of ADSS_SAI4 block
0	1	CGU_CONFIG_MELODY_ADSS_SAO1_RST_N_SOFT	reset of ADSS_SAO1 block

BIT	POR VALUE	NAME	DESCRIPTION
0	1	CGU_CONFIG_MELODY_ADSS_SAO2_RST_N_SOFT	reset of ADSS_SAO2 block
0	1	CGU_CONFIG_MELODY_ADSS_SPDIF_RST_N_SOFT	reset of ADSS_SPDIF block
0	1	CGU_CONFIG_IP_BLAS_2111_HRESETN_SOFT	reset of ip_2111(flash interface) block
0	1	CGU_CONFIG_SSA1_LCD_INTERFACE_PNRES_SOFT	reset of lcd interface block
0	1	CGU_CONFIG_SIMPLE_DMA_PNRES_SOFT	reset of sdma block
0	1	CGU_CONFIG_AHB_MPMC_PL172_HRESETN_SOFT	reset of MPMC block
0	1	CGU_CONFIG_MMIOINTC_RESETN_SOFT	reset of intc interface (2) (mmio_intc)
31:1	Reserved		

Table 51 CGU_CONFIG_HP0_FIN_SELECT register

BIT	POR VALUE	NAME	DESCRIPTION
2:0	0	hp0_fin_select	Select input to high power pll
			0000 reserved
			0001 FFAST (12 MHz)
			0010 PAD_MCI_CLK
			0011 USB_BITCLK
			0100 PAD_DAI_BCK
			0101 PAD_DAI_WS
			0110 SPD3_BCK
			0111 SPD3_WS
			1000 reserved
			1001 LP0
			1x1x reserved
			11xx reserved
31:3	Reserved	Reserved	

Table 52 CGU_CONFIG_HP0_MDEC register

BIT	POR VALUE	NAME	DESCRIPTION
9:0	0		Feedback multiplier selects (actual multiplication = hp_msel + 1)
31:10	Reserved	Reserved	

Table 53 CGU_CONFIG_HP0_NDEC register

BIT	POR VALUE	NAME	DESCRIPTION
7:0	0		Pre scale input divider selects (actual division = hp_nsel + 1)
31:8	Reserved	Reserved	

Table 54 CGU_CONFIG_HP0_PDEC register

BIT	POR VALUE	NAME	DESCRIPTION
3:0	0		Post scale output divider selects (actual division = hp_psel + 1)
31:4	Reserved	Reserved	

Table 55 CGU_CONFIG_HP0_MODE register

BIT	POR VALUE	NAME	DESCRIPTION
8:0		MODE	MODE = {bypass, limup_off, bandsel, frm, directi, directo, pd, skew_en, clken}
31:9	Reserved	Reserved	

Table 56 CGU_CONFIG_HP0_STATUS register

BIT	POR VALUE	NAME	DESCRIPTION
0			
31:1	Reserved	Reserved	

Table 57 CGU_CONFIG_HP0_ACK register

BIT	POR VALUE	NAME	DESCRIPTION
0			
31:1	Reserved	Reserved	

Table 58 CGU_CONFIG_HP0_REQ register

BIT	POR VALUE	NAME	DESCRIPTION
0			
31:1	Reserved	Reserved	

Table 59 CGU_CONFIG_HP0_INSELR register

BIT	POR VALUE	NAME	DESCRIPTION
0			
31:1	Reserved	Reserved	

Table 60 CGU_CONFIG_HP0_INSELI register

BIT	POR VALUE	NAME	DESCRIPTION
0			
31:1	Reserved	Reserved	

Table 61 CGU_CONFIG_HP0_INSELP register

BIT	POR VALUE	NAME	DESCRIPTION
0			
31:1	Reserved	Reserved	

Table 62 CGU_CONFIG_HP0_SELR register

BIT	POR VALUE	NAME	DESCRIPTION
0			
31:1	Reserved	Reserved	

Table 63 CGU_CONFIG_HP0_SELI register

BIT	POR VALUE	NAME	DESCRIPTION
0			
31:1	Reserved	Reserved	

Table 64 CGU_CONFIG_HP0_SELP register

BIT	POR VALUE	NAME	DESCRIPTION
0			
31:1	Reserved	Reserved	

Table 65 CGU_CONFIG_LP0_FIN_SELECT register

BIT	POR VALUE	NAME	DESCRIPTION
3:0	0001	lp<id>_fin_select	Select input to LPPLL<id>, see frequency selection table for details.
			0000 FSLOW (32.768 kHz)
			0001 FFAST (12 MHz)
			0010 PAD_MCI_CLK
			0011 USB_BITCLK
			0100 PAD_DAI_BCK
			0101 PAD_DAI_WS
			0110 SPD3_BCK
			0111 SPD3_WS
			1000 HP0
			1001 reserved
			1x1x reserved
			11xx reserved
31:4	Reserved	Reserved	

Table 66 CGU_CONFIG_LP0_PWD register

BIT	POR VALUE	NAME	DESCRIPTION
0	1	lp<id>_pwd	Power down when high
31:1	Reserved	Reserved	

Table 67 CGU_CONFIG_LP0_BYPASS register

BIT	POR VALUE	NAME	DESCRIPTION
0	0		
31:1	Reserved	Reserved	

Table 68 CGU_CONFIG_LP0_LOCK register

BIT	POR VALUE	NAME	DESCRIPTION
0			
31:1	Reserved	Reserved	

Table 69 CGU_CONFIG_LP0_DIRECT register

BIT	POR VALUE	NAME	DESCRIPTION
0	0		
31:1	Reserved	Reserved	

Table 70 CGU_CONFIG_LP0_MSEL register

BIT	POR VALUE	NAME	DESCRIPTION
4:0	0	lp<id>_msel	msel[4:0] defines multiplication ratio (actual multiplication = lp1_sel + 1)
31:5	Reserved	Reserved	

Table 71 CGU_CONFIG_LP0_PSEL register

BIT	POR VALUE	NAME	DESCRIPTION
1:0	0	lp<id>_psel	psel
31:2	Reserved	Reserved	

17.6 Open items of this document

- Start-up. That all blocks are running on 12 Mhz clock, reset two clock pulse at the end of start-up, reprogramming

17.7 Programming CGU

Program order CGU to set the clocks in application mode

```
///*****************************************************************************
/// Initialise OSCILLATOR
///*****************************************************************************
// Init FFAST OSCILLATOR 12 Mhz
// FFAST = ON
// Write 1 to bit 0 (=ON) (POR value)

///*****************************************************************************
/// Initialise LP0 PLL
///*****************************************************************************
// LP0 PLL REGISTERS
//CGU_LP0_FIN_SELECT Default = 0000 (Oscillator)
//CGU_LP0_PWD Default = 1 (power down)
//CGU_LP0_BYPASS Default = 0 (Don't bypass PLL)
//CGU_LP0_LOCK Default = undefined
//CGU_LP0_DIRECT Default = 0 (use post divider)
//CGU_LP0_MSEL Default = 00000 (feedback divider setting)
//CGU_LP0_PSEL Default = 00 (post divider setting)

// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   // POWER DOWN LP PLL
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
// LP0_PWD ()

// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   // SELECT INPUT FREQUENCY
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
// LP0_FIN_SELECT

// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
  // WRITE MSEL
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
// LP0_MSEL
```

```
  // bits4:0 = 01010 (Msel=10, M=11, 11.025*11 = 121.275MHz)
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   // WRITE PSEL
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
// LP0_PSEL
  // bits4:0 = 00 (Psel=0, P=1 +> div by 2)
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   // POWER UP LP PLL WRITE LP0_PWD
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
// LP0_PWD
  // bit0 = 0 (Enable LP0 PLL)
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   // WAIT SYSTEM/DSP PLL LOCK SIGNAL
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
WaitSystemPllLock:

///*****************************************************************************
/// Initialise HP0 PLL
///*****************************************************************************
// HP0 CGU AUDIO PLL Registers
//CGU_HP0_FIN_SELECT Default = 0000 (Oscillator)
//CGU_HP0_MDEC Default = 0 0000 0000 0000 0000 (17 bits)
//CGU_HP0_NDEC Default = 00 0000 0000 (10 bits)
//CGU_HP0_PDEC Default = 000 0000 (7 bits)
//CGU_HP0_MODE Default = 0 0000 0110 (9 bits)
//CGU_HP0_STATUS
//CGU_HP0_ACK
//CGU_HP0_REQ Default = 000 (3 bits)div change request
//CGU_INSELR     Default = 0000 (4 bits) bandwidth selection
//CGU_INSELI Default = 0000 (4 bits)bandwidth selection
//CGU_INSELP Default = 0000 (4 bits)bandwidth selection
//CGU_SELR Default = 0000 (4 bits) audio inclock PLL BW select
//CGU_SELI     Default = 0000 (4 bits) audio inclock PLL BW select
//CGU_SELP     Default = 0000 (4 bits) audio inclock PLL BW select

// Select Oscillator as input frequency: Master Mode
```

```
// CGU_HP0_FIN_SELECT
// Write MDEC feedback divider
// CGU_HP0_MDEC
// Write NDEC pre divider
// CGU_HP0_NDEC
// Write PDEC post divider
// CGU_HP0_PDEC
// Write HP0_SELR
// Write HP0_SELI
// Write HP0_SELP

// Write HP0 MODE
// CGU_HP0_MODE : Normal mode
//bit0 = 1 enable output clock
//bit1 = 0 enable skew mode
//bit2 = 0 power down
//bit3 = 0 directo: by-pass post-divider
//bit4 = 0 directi: by-pass pre-divider
//bit5 = 0 enable free running mode
//bit6 = 0 enable external control of bandwidth
//bit7 = 0 disable uplimiter for fractional PLL applications
//bit8 = 0 bypass audio PLL

// Request to load new divider values via CGU_HP0_REQ
// CGU_HP0_REQ
//bit0 = 1new M div value
//bit1 = 1new N div value
//bit2 = 1new P div value

// Wait for CGU_HP0_ACK on M, N and P
// Reset CGU_HP0_REQ for M, N and P

// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   // WAIT AUDIO PLL LOCK SIGNAL
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
WaitAudioPllLock:
```

```
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
// CGUINITIALISATION FOR ALL SYSTEM/DSP CLOCKS (LP0 as input)
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
StartSetSystemClocks:

// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   // FRAC DIVIDERS CONTROL
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
// First disable all LP0 FracDiv's
   // CGU_BCR_LP0 Base Control Register
// Synchronise all fracdiv's on System/DSP PLL Base

// Fractional Divider Control 0: FRACDIV SPD_SYSCLK
// SPD_SYSCLK => SPDIF_SYSCLK (36MHz < SPDIF_SYSCLK < 69 MHz)
// SPD_SYSCLK / clk00
// e.g 121.275 MHz / 3 -> 40 MHz
//    LP0 = 121.275 MHz
//    ==================
// 8 bits FD
// n =1, m = 3
//    madd= m-n = 2
//    msub -n = -1
//    power optimalisation
//    madd = 2 * 64 = 1000 0000 (128)
//    msub = -1 * 64    = 1100 0000 (-64)
//    fdctrl_run     bit0 = 1
 //    fdctrl_reset   bit1 = 1
 //    fdctrl_stretch bit2 = 1
//    madd = m-n bits10:3 = 1000 0000 (128)
//    msub = -n   bits18:11 = 1100 0000 (-64)
//    FDC_0 = 110 0000 0100 0000 0111
//    FDC_0 = 0x060407
//    #define CGU_FDC_0 #02CC
// #define CGU_FDC_SPD_SYSCLK#02CC
```

```
// Now disable fdctrl_reset = 0
//    FDC_0 = 110 0000 0100 0000 0101
//    FDC_0 = 0x060405

// Fractional Divider Control 4: FRACDIV 4/7 ADC_CLK
// Clock for 10bitsADC. Default = 4 MHz

// +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   // END OF LP0 BASE FRAC DIVIDERS CONTROL
// +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

// +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   // ENABLE SPREADING CONTROL
// +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
// Enable Selection Register: SPD_SYSCLK
//   CGU_ESR_SPD_SYSCLK
//   CGU_ESR_0
//   ESR_enable bit0 = 1
//   ESR_select bits3:1 = 000 (Fracdiv 0/6)

// Enable Selection Register : UART_UCLK
// Enable Selection Register : VPB1_PCLK
// Enable Selection Register : UART_PCLK
// Enable Selection Register : VPB0_PCLK
// Enable Selection Register : CGU_PCLK
// Enable Selection Register : redo for all other clocks

// Then enable all LP0 FracDiv's
   // CGU_BCR_LP0 Base Control Register
// Synchronise all fracdiv's on System/DSP PLL Base

// +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   // END ENABLE SPREADING CONTROL
// +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

// +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
   // POWER CONTROL
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

   // POWER ON RESET VALUE IS ALL CLOCKS RUN

// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   // END POWER CONTROL
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
  // CGUINITIALISATION FOR ALL AUDIO CLOCKS
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   // FRAC DIVIDERS CONTROL
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

// First disable all HP0 FracDiv's
   // CGU_BCR_HP0 Base Control Register
// Synchronise all frac div's on Audio PLL Base

// Fractional Divider Control 9: FRACDIV 2/7 DAI1_BCK
// CR_DAI1_BCK
// 8 bits FD
//    512Fs -> 64Fs
// n =1, m =8
//    madd= m-n = 7
//    msub -n = -1
//    power optimalisation
//    madd = 7 * 16 = 0111 0000 (112)
//    msub = -1 * 16    = 1111 0000 (-16)
//    fdctrl_run     bit0 = 1
 //    fdctrl_reset   bit1 = 1
 //    fdctrl_stretch bit2 = 1
//    madd = m-n bits10:3 = 0111 0000 (112)
//    msub = -n   bits18:11 = 1111 0000 (-16)
//    FDC_9 = 111 1000 0011 1000 0111
//    FDC_9 = 0x078387
```

```
//    #define CGU_FDC_9 #02F0
// #define CGU_FDC_DAI1_BCK      #02F0

// Fractional Divider Control: FRACDIV DAO_BCK
// Fractional Divider Control: FRACDIV DAO_WS
// Fractional Divider Control: FRACDIV SDAC_NS_CLK
// Fractional Divider Control: FRACDIV AUDIO_SYSCLK
// Fractional Divider Control: FRACDIV DCDC_CONV_CLK
// Fractional Divider Control: FRACDIV DAO_CLK

// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   // END FRAC DIVIDERS CONTROL
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   // ENABLE SPREADING CONTROL
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
// Enable Selection Register : CR_DAI1_BCK
// Enable Selection Register : CR_DAO_BCK
// Enable Selection Register : CR_DAI1_WS
// Enable Selection Register : CR_DAO_WS
// Enable Selection Register : SDAC_NS_CLK
// Enable Selection Register : DAC_DSPCLK
// Enable Selection Register : SADC_DECCLK
// Enable Selection Register : SADC_SYSCLK
// Enable Selection Register : DCDC_CONV_CLK
// Enable Selection Register : DAO_CLK

// Then enable all HP0 FracDiv's
   // CGU_BCR_HP0 Base Control Register
// Synchronise all frac div's on Audio PLL Base

// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   // END ENABLE SPREADING CONTROL
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   // POWER CONTROL
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   // POWER ON RESET VALUE IS ALL CLOCKS RUN
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   // END POWER CONTROL
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   // BASES CONTROL
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   // MULTIPLEXER SWITCHES BASES CONTROL
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

// CGU_FS1_x BASE: LP0 DSP SYSTEM
// FS1_LP0
// SHOULD BE system_PLL_output

// CGU_FS1_x BASE: HP0 AUDIO
// FS1_HP0
// AUDIO MASTER MODE // audio_PLL_output

// CGU_FS1_x BASE: DCDC
// FS1_DCDC // oscillator_output

// CGU_FS1_x BASE: SPDIF
// FS1_SPDIF // spdif_bck

// CGU_FS1_x BASE: DAI1_BCK
// FS1_DAI1_BCK
// AUDIO MASTER MODE // pad_dai1_bck

// CGU_FS1_x BASE: DAO_BCK
// FS1_DAO_BCK
// AUDIO MASTER MODE // pad_dao_bck
```

```
// CGU_FS1_x BASE: DAO_CLK
// FS1_CR_DAO_CLK
// AUDIO MASTER MODE // audio_PLL_output
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   // ENABLING/DISABLING   BASES CONTROL
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

// CGU_SCR_LP0 SWITCH CONTROL REGISTER
// CGU_SCR_LP0 (0x0000)
// AUDIO MASTER MODE
//    enable F1 bit0 = 1
//    enable F2 bit1 = 0
//    reset      bit2 = 0
//    stop       bit3 = 0

// CGU_SCR_HP0 SWITCH CONTROL REGISTER
// CGU_SCR_HP0 // AUDIO MASTER MODE

// CGU_SCR_DCDC SWITCH CONTROL REGISTER
// CGU_SCR_DCDC // AUDIO MASTER MODE

// CGU_SCR_SPDIF SWITCH CONTROL REGISTER
// CGU_SCR_SPDIF // AUDIO MASTER MODE

// CGU_SCR_DAI1 SWITCH CONTROL REGISTER
// CGU_SCR_DAI1 // AUDIO MASTER MODE

// CGU_SCR_CR_DAO_CLK SWITCH CONTROL REGISTER
// CGU_SCR_CR_DAO_CLK (0x0020)

// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   // END   BASES CONTROL
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   // ENABLE AND SYNCHRONISING FRAC DIVIDERS
// ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

// Then enable all LP0 FracDiv's
   // CGU_BCR_LP0 Base Control Register
// Synchronise all frac div's on System/DSP PLL Base
```

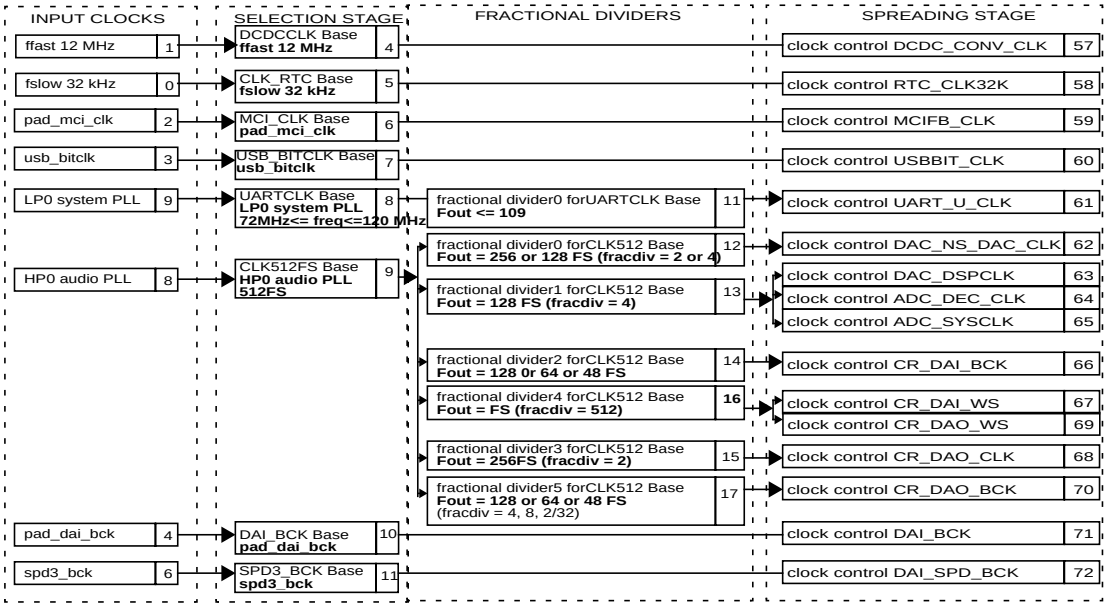
General remarks:

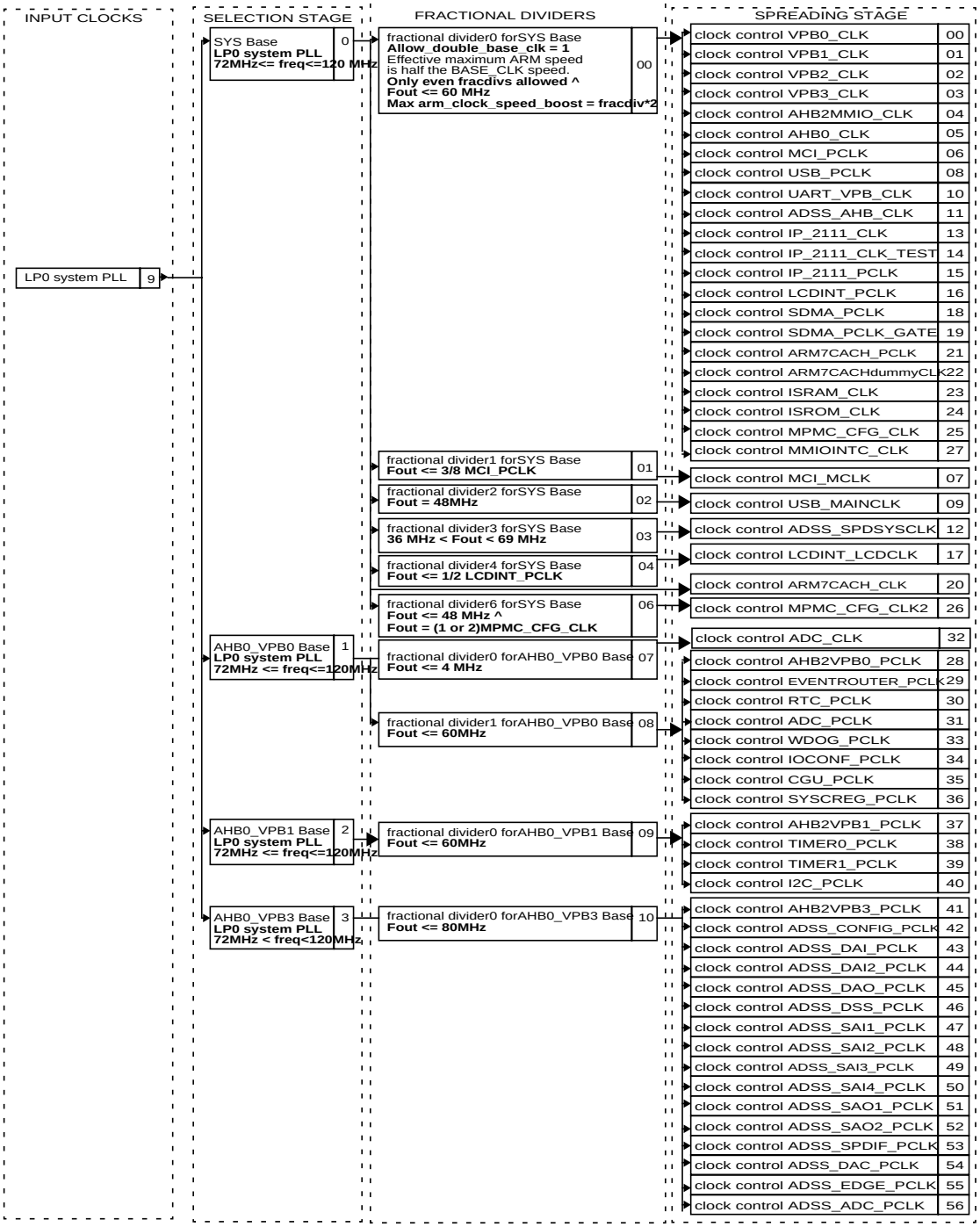
1. As default all clocks are programmed on ffast (12 MHz)

2. // allows higher ARM then AHB speed (see ARM7TDMI Microcontroller chapter)

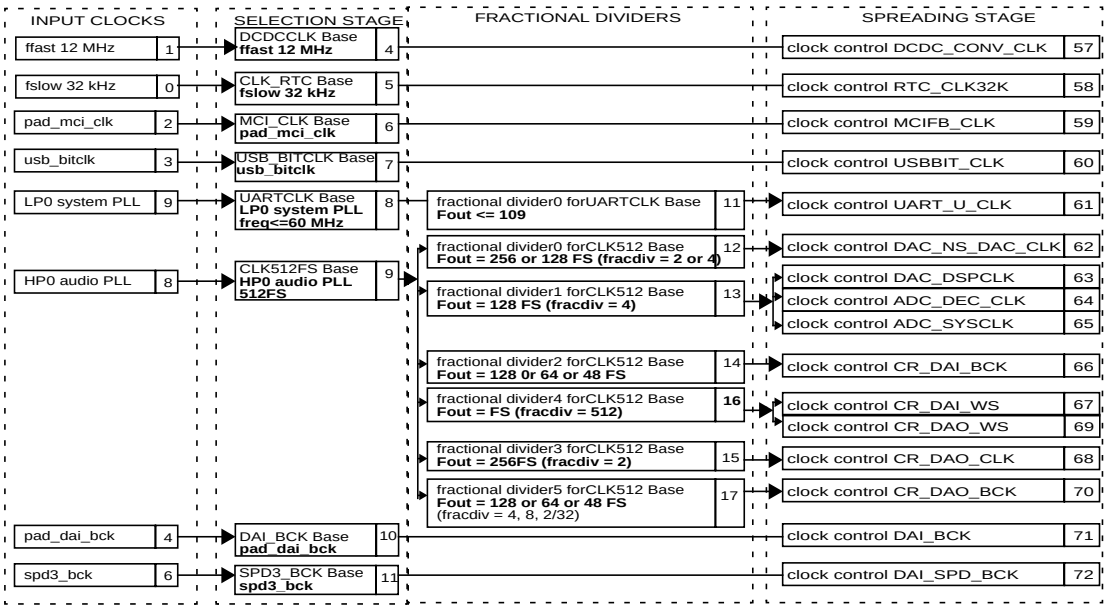
```
ARM_REGS->synchronous_unequal_clocks = 0x1; // allows higher ARM then AHB speed
ARM_REGS->arm_clk_speed_boost=0x4 | 0x80;
SWITCHBOX_REGS -> clk_pcr[CGU_SWITCHBOX_ARM7TDMISCACHE_DUMMY_HCLK_ID] |= PCR_ENOUT_EN;
```

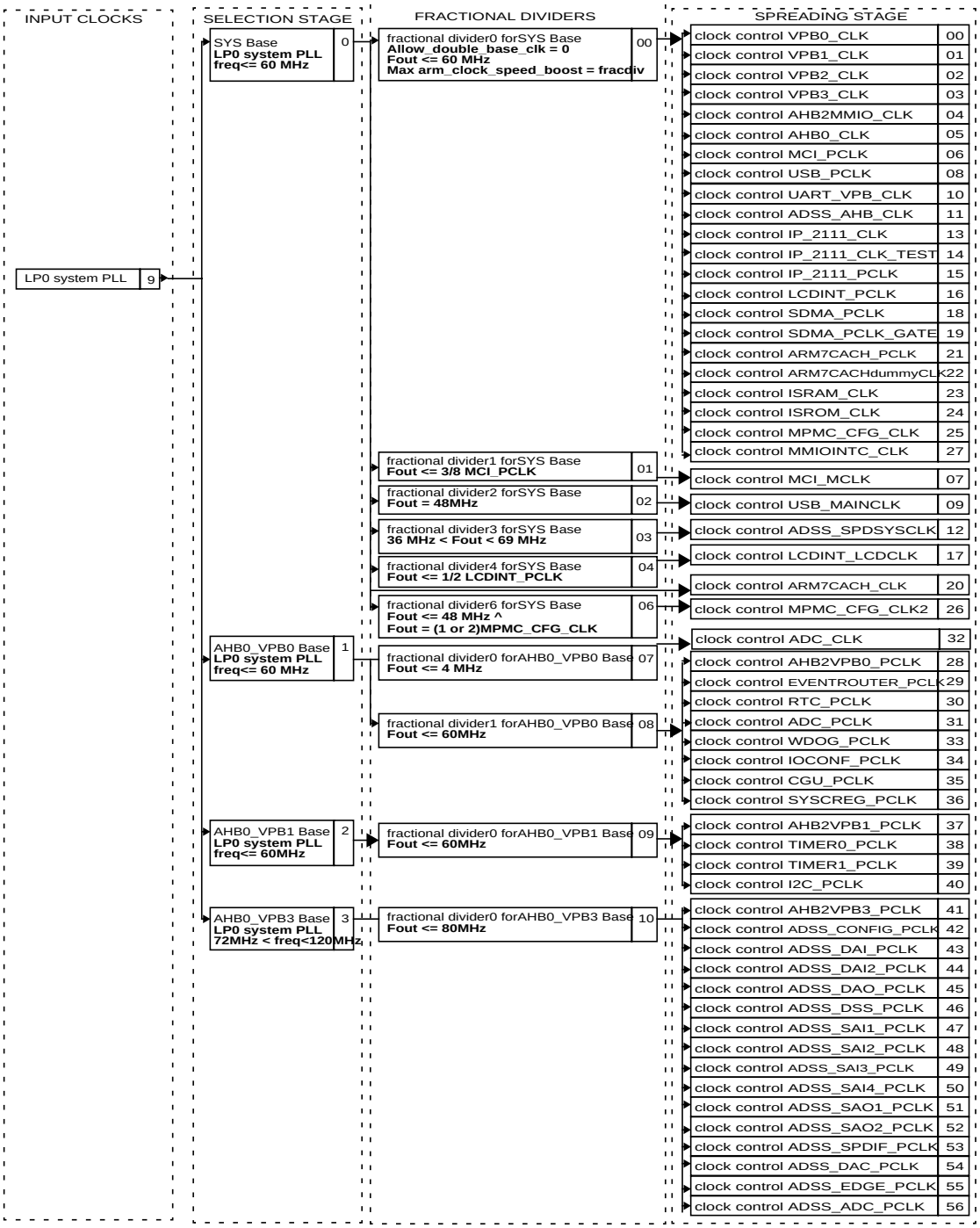
17.7.1 MAXIMUM PERFORMANCE SETTINGS FOR CGU (SYS BASE CLOCK > 64 MHZ ^ SYS BASE CLOCK <=120 MHZ)




INPUT CLOCKS
SELECTION STAGE
FRACTIONAL DIVIDERS
SPREADING STAGE
LP0 system PLL 9
SYS Base 0
LP0 system PLL
72MHz<= freq<=120 MHz
fractional divider0 forSYS Base 00
Allow_double_base_clk = 1
Effective maximum ARM speed is half the BASE_CLK speed.
Only even fracdivs allowed ^
Fout <= 60 MHz
Max arm_clock_speed_boost = fracdiv*2
fractional divider1 forSYS Base 01
Fout <= 3/8 MCI_PCLK
fractional divider2 forSYS Base 02
Fout = 48MHz
fractional divider3 forSYS Base 03
36 MHz < Fout < 69 MHz
fractional divider4 forSYS Base 04
Fout <= 1/2 LCDINT_PCLK
fractional divider6 forSYS Base 06
Fout <= 48 MHz ^
Fout = (1 or 2)MPMC_CFG_CLK
AHB0_VPB0 Base 1
LP0 system PLL
72MHz <= freq<=120MHz
fractional divider0 forAHB0_VPB0 Base 07
Fout <= 4 MHz
fractional divider1 forAHB0_VPB0 Base 08
Fout <= 60MHz
AHB0_VPB1 Base 2
LP0 system PLL
72MHz <= freq<=120MHz
fractional divider0 forAHB0_VPB1 Base 09
Fout <= 60MHz
AHB0_VPB3 Base 3
LP0 system PLL
72MHz < freq<120MHz
fractional divider0 forAHB0_VPB3 Base 10
Fout <= 80MHz
clock control VPB0_CLK 00
clock control VPB1_CLK 01
clock control VPB2_CLK 02
clock control VPB3_CLK 03
clock control AHB2MMIO_CLK 04
clock control AHB0_CLK 05
clock control MCI_PCLK 06
clock control USB_PCLK 08
clock control UART_VPB_CLK 10
clock control ADSS_AHB_CLK 11
clock control IP_2111_CLK 13
clock control IP_2111_CLK_TEST 14
clock control IP_2111_PCLK 15
clock control LCDINT_PCLK 16
clock control SDMA_PCLK 18
clock control SDMA_PCLK_GATE 19
clock control ARM7CACH_PCLK 21
clock control ARM7CACHdummyCLK 22
clock control ISRAM_CLK 23
clock control ISROM_CLK 24
clock control MPMC_CFG_CLK 25
clock control MMIOINTC_CLK 27
clock control MCI_MCLK 07
clock control USB_MAINCLK 09
clock control ADSS_SPDSYSCLK 12
clock control LCDINT_LCDCLK 17
clock control ARM7CACH_CLK 20
clock control MPMC_CFG_CLK2 26
clock control ADC_CLK 32
clock control AHB2VPB0_PCLK 28
clock control EVENTROUTER_PCLK 29
clock control RTC_PCLK 30
clock control ADC_PCLK 31
clock control WDOG_PCLK 33
clock control IOCONF_PCLK 34
clock control CGU_PCLK 35
clock control SYSCREG_PCLK 36
clock control AHB2VPB1_PCLK 37
clock control TIMER0_PCLK 38
clock control TIMER1_PCLK 39
clock control I2C_PCLK 40
clock control AHB2VPB3_PCLK 41
clock control ADSS_CONFIG_PCLK 42
clock control ADSS_DAI_PCLK 43
clock control ADSS_DAI2_PCLK 44
clock control ADSS_DAO_PCLK 45
clock control ADSS_DSS_PCLK 46
clock control ADSS_SAI1_PCLK 47
clock control ADSS_SAI2_PCLK 48
clock control ADSS_SAI3_PCLK 49
clock control ADSS_SAI4_PCLK 50
clock control ADSS_SAO1_PCLK 51
clock control ADSS_SAO2_PCLK 52
clock control ADSS_SPDIF_PCLK 53
clock control ADSS_DAC_PCLK 54
clock control ADSS_EDGE_PCLK 55
clock control ADSS_ADC_PCLK 56

17.7.2 SYS BASE CLOCK <= 60 MHz



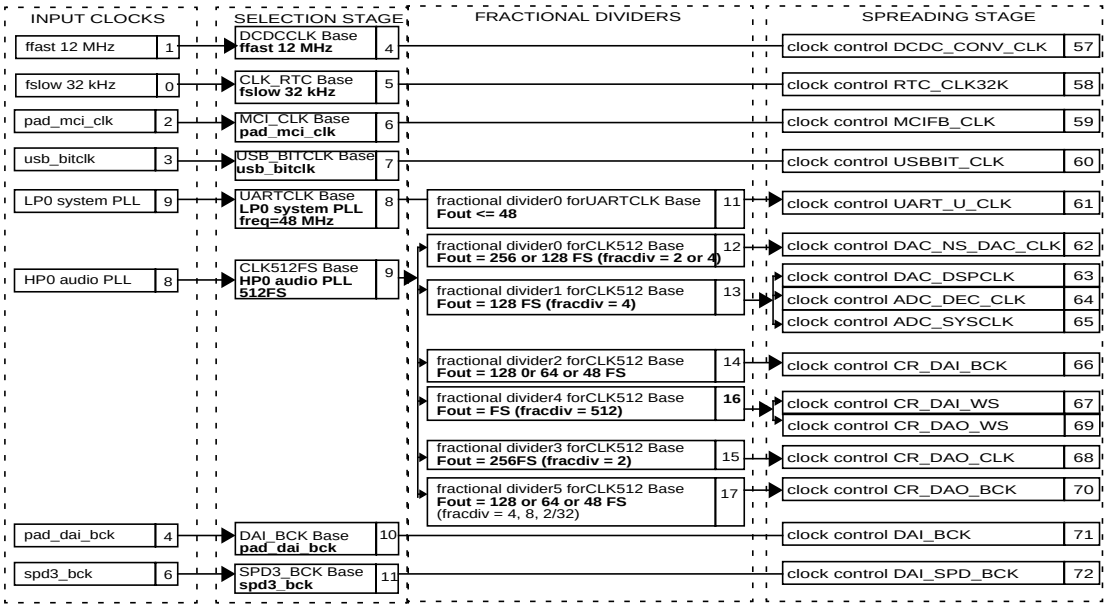

INPUT CLOCKS
SELECTION STAGE
FRACTIONAL DIVIDERS
SPREADING STAGE
LP0 system PLL 9
SYS Base LP0 system PLL freq<= 60 MHz 0
AHB0_VPB0 Base LP0 system PLL freq<= 60 MHz 1
AHB0_VPB1 Base LP0 system PLL freq<= 60MHz 2
AHB0_VPB3 Base LP0 system PLL 72MHz < freq<120MHz 3
fractional divider0 forSYS Base Allow_double_base_clk = 0 Fout <= 60 MHz Max arm_clock_speed_boost = fracdiv 00
fractional divider1 forSYS Base Fout <= 3/8 MCI_PCLK 01
fractional divider2 forSYS Base Fout = 48MHz 02
fractional divider3 forSYS Base 36 MHz < Fout < 69 MHz 03
fractional divider4 forSYS Base Fout <= 1/2 LCDINT_PCLK 04
fractional divider6 forSYS Base Fout <= 48 MHz ^ Fout = (1 or 2)MPMC_CFG_CLK 06
fractional divider0 forAHB0_VPB0 Base Fout <= 4 MHz 07
fractional divider1 forAHB0_VPB0 Base Fout <= 60MHz 08
fractional divider0 forAHB0_VPB1 Base Fout <= 60MHz 09
fractional divider0 forAHB0_VPB3 Base Fout <= 80MHz 10
clock control VPB0_CLK 00
clock control VPB1_CLK 01
clock control VPB2_CLK 02
clock control VPB3_CLK 03
clock control AHB2MMIO_CLK 04
clock control AHB0_CLK 05
clock control MCI_PCLK 06
clock control USB_PCLK 08
clock control UART_VPB_CLK 10
clock control ADSS_AHB_CLK 11
clock control IP_2111_CLK 13
clock control IP_2111_CLK_TEST 14
clock control IP_2111_PCLK 15
clock control LCDINT_PCLK 16
clock control SDMA_PCLK 18
clock control SDMA_PCLK_GATE 19
clock control ARM7CACH_PCLK 21
clock control ARM7CACHdummyCLK 22
clock control ISRAM_CLK 23
clock control ISROM_CLK 24
clock control MPMC_CFG_CLK 25
clock control MMIOINTC_CLK 27
clock control MCI_MCLK 07
clock control USB_MAINCLK 09
clock control ADSS_SPDSYSCLK 12
clock control LCDINT_LCDCLK 17
clock control ARM7CACH_CLK 20
clock control MPMC_CFG_CLK2 26
clock control ADC_CLK 32
clock control AHB2VPB0_PCLK 28
clock control EVENTROUTER_PCLK 29
clock control RTC_PCLK 30
clock control ADC_PCLK 31
clock control WDOG_PCLK 33
clock control IOCONF_PCLK 34
clock control CGU_PCLK 35
clock control SYSCREG_PCLK 36
clock control AHB2VPB1_PCLK 37
clock control TIMER0_PCLK 38
clock control TIMER1_PCLK 39
clock control I2C_PCLK 40
clock control AHB2VPB3_PCLK 41
clock control ADSS_CONFIG_PCLK 42
clock control ADSS_DAI_PCLK 43
clock control ADSS_DAI2_PCLK 44
clock control ADSS_DAO_PCLK 45
clock control ADSS_DSS_PCLK 46
clock control ADSS_SAI1_PCLK 47
clock control ADSS_SAI2_PCLK 48
clock control ADSS_SAI3_PCLK 49
clock control ADSS_SAI4_PCLK 50
clock control ADSS_SAO1_PCLK 51
clock control ADSS_SAO2_PCLK 52
clock control ADSS_SPDIF_PCLK 53
clock control ADSS_DAC_PCLK 54
clock control ADSS_EDGE_PCLK 55
clock control ADSS_ADC_PCLK 56

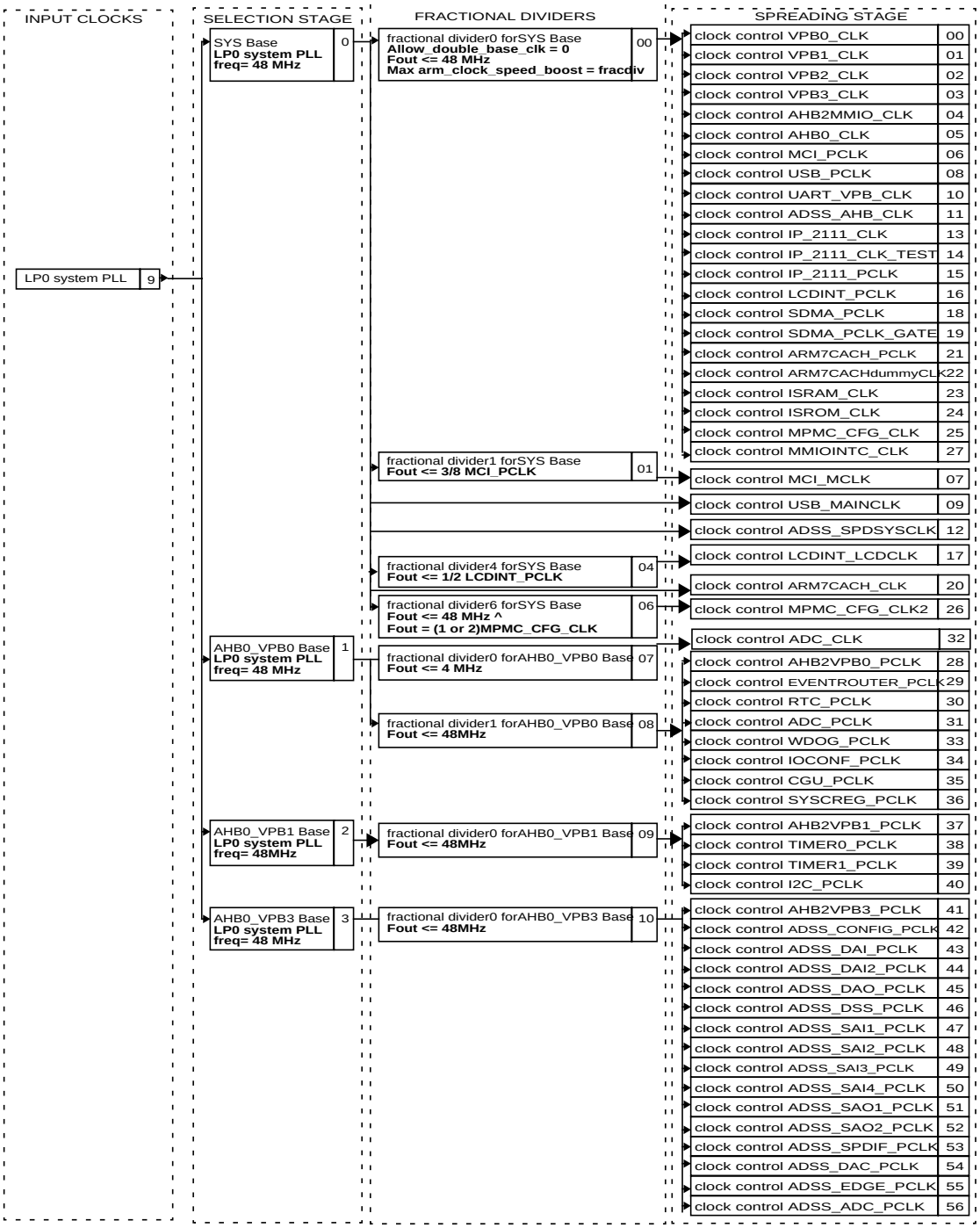
17.7.3 USB DOWNLOAD (LPPLL = 48 MHz)

1. USB mainclk is designed with some restrictions it need 50% duty-cycle. But using stretching for the USB fracdiv is designed with some restrictions; stretching can only be used when fracdiv setting < 1/2.

2. With LPPLL (Core) of 60, need the fracdiv by programmed on 4/5 to generate for the USB block the needed clock of 48 MHz, this means that with a of 4/5 which is higher than 1/2 no stretching will be possible or no 50% duty-cycle so that this solution cannot used.

As result of that for USB transfers need the LPPLL (Core) reprogrammed to 48 MHz. So dynamical switching from the LPPLL is needed if there is a switch between USB transfers and other modes which are requested a higher core clock and as a result of that also the fracdivs probably need to be changed. For USB transfers will the Core always run on 48 MHz which mean that it will not be possible to get USB download speed boost.




INPUT CLOCKS
SELECTION STAGE
FRACTIONAL DIVIDERS
SPREADING STAGE
LP0 system PLL 9
SYS Base LP0 system PLL freq= 48 MHz 0
AHB0_VPB0 Base LP0 system PLL freq= 48 MHz 1
AHB0_VPB1 Base LP0 system PLL freq= 48MHz 2
AHB0_VPB3 Base LP0 system PLL freq= 48 MHz 3
fractional divider0 forSYS Base Allow_double_base_clk = 0 Fout <= 48 MHz Max arm_clock_speed_boost = fracdiv 00
fractional divider1 forSYS Base Fout <= 3/8 MCI_PCLK 01
fractional divider4 forSYS Base Fout <= 1/2 LCDINT_PCLK 04
fractional divider6 forSYS Base Fout <= 48 MHz ^ Fout = (1 or 2)MPMC_CFG_CLK 06
fractional divider0 forAHB0_VPB0 Base Fout <= 4 MHz 07
fractional divider1 forAHB0_VPB0 Base Fout <= 48MHz 08
fractional divider0 forAHB0_VPB1 Base Fout <= 48MHz 09
fractional divider0 forAHB0_VPB3 Base Fout <= 48MHz 10
clock control VPB0_CLK 00
clock control VPB1_CLK 01
clock control VPB2_CLK 02
clock control VPB3_CLK 03
clock control AHB2MMIO_CLK 04
clock control AHB0_CLK 05
clock control MCI_PCLK 06
clock control USB_PCLK 08
clock control UART_VPB_CLK 10
clock control ADSS_AHB_CLK 11
clock control IP_2111_CLK 13
clock control IP_2111_CLK_TEST 14
clock control IP_2111_PCLK 15
clock control LCDINT_PCLK 16
clock control SDMA_PCLK 18
clock control SDMA_PCLK_GATE 19
clock control ARM7CACH_PCLK 21
clock control ARM7CACHdummyCLK 22
clock control ISRAM_CLK 23
clock control ISROM_CLK 24
clock control MPMC_CFG_CLK 25
clock control MMIOINTC_CLK 27
clock control MCI_MCLK 07
clock control USB_MAINCLK 09
clock control ADSS_SPDSYSCLK 12
clock control LCDINT_LCDCLK 17
clock control ARM7CACH_CLK 20
clock control MPMC_CFG_CLK2 26
clock control ADC_CLK 32
clock control AHB2VPB0_PCLK 28
clock control EVENTROUTER_PCLK 29
clock control RTC_PCLK 30
clock control ADC_PCLK 31
clock control WDOG_PCLK 33
clock control IOCONF_PCLK 34
clock control CGU_PCLK 35
clock control SYSCREG_PCLK 36
clock control AHB2VPB1_PCLK 37
clock control TIMER0_PCLK 38
clock control TIMER1_PCLK 39
clock control I2C_PCLK 40
clock control AHB2VPB3_PCLK 41
clock control ADSS_CONFIG_PCLK 42
clock control ADSS_DAI_PCLK 43
clock control ADSS_DAI2_PCLK 44
clock control ADSS_DAO_PCLK 45
clock control ADSS_DSS_PCLK 46
clock control ADSS_SAI1_PCLK 47
clock control ADSS_SAI2_PCLK 48
clock control ADSS_SAI3_PCLK 49
clock control ADSS_SAI4_PCLK 50
clock control ADSS_SAO1_PCLK 51
clock control ADSS_SAO2_PCLK 52
clock control ADSS_SPDIF_PCLK 53
clock control ADSS_DAC_PCLK 54
clock control ADSS_EDGE_PCLK 55
clock control ADSS_ADC_PCLK 56

18 SYSCREG

18.1 Overview

This module is a VPB peripheral and is meant for controlling the chip. In this unit only registers are located which will control muxes and mode settings

18.2 Registers

SYSCREG Memory Map Overview(0x80005000 - 0x800053ff)

DESCRIPTION	REG	RESET
reserved	0x000	0x0
SYSCREG_DCDC_CONVERTER_DCDC1	0x004	0x5
SYSCREG_DCDC_CONVERTER_DCDC2	0x008	0x1
SYSCREG_DCDC_CONVERTER_CLK	0x00c	0x0
SYSCREG_FUSEBOX_FO0	0x010	0xffffffff
SYSCREG_FUSEBOX_FO1	0x014	0xffffffff
SYSCREG_FUSEBOX_FO2	0x018	0xffffffff
SYSCREG_FUSEBOX_FO3	0x01c	0xffffffff
SYSCREG_SEL	0x020	0x0
SYSCREG_SSA1_RTC_CFG	0x024	0x0
SYSCREG_SSA1_ADC_PD_ADC10BITS	0x028	0x0
SYSCREG_SSA1_MCI_PL180_PRESERVED	0x02c	0x0
SYSCREG_USBAPB_TOP_STS	0x030	0x0
SYSCREG_USBAPB_TOP_CFG	0x034	0x0
SYSCREG_SW_IP_BLAS_2111_PWD_N	0x038	0x1
SYSCREG_IP_BLAS_2111_RSTBUSY	0x03c	0x0
SYSCREG_CGU_DYN_HP0	0x040	0x0
SYSCREG_CGU_DYN_LP0	0x044	0x0
SYSCREG_SYS_CREG_SDMA_EXT_EN_3	0x048	0x0
SYSCREG_SYS_CREG_SDMA_EXT_EN_5	0x04c	0x0
SYSCREG_SELECTION_CFG	0x050	0x1
SYSCREG_ISRAM_LATENCY_CFG	0x054	0x0
SYSCREG_ISROM_LATENCY_CFG	0x058	0x0
SYSCREG_AHB_MPMC_PL172_MISC	0x05c	0x4
SYSCREG_MPMP_DELAYMODES	0x060	0x1
SYSCREG_WIRE_EBI_MSIZE_INIT	0x064	0x1
SYSCREG_AHB_BOOT	0x068	0x0
SYSCREG_ARM7TDMISCACHE_SHADOW_POINTER	0x06c	0x200000
SYSCREG_SLEEPSTATUS	0x070	0x0
SYSCREG_CHIP_ID	0x074	0x101100c

18.2.1 DCDC_CONVERTER REGISTERS

Adjust ranges for DC/DC converter, see also chapter DC/DC.

1) DCDC1_ADJUST VDC1(V)

Table 72 Adjust Range VDC1 seven equal steps from 000 to 111

BITS	LOW THRESH.	HIGH THRESH	VDELTA(MV)
000	3.562	3.710	148
001	3.406	3.548	142
010	3.250	3.385	135
011	3.094	3.223	129
100	2.938	3.306	122
101	2.782	2.898	116
110	2.626	2.735	109
111	2.470	2.573	103

2) DCDC2_ADJUST VDC2(V)

Table 73 Adjust range VDC2

BITS	LOW THRESH.	HIGH THRESH	VDELTA(MV)
000	1.742	1.815	73
001	1.664	1.733	69
010	1.586	1.652	66
011	1.508	1.571	63
100	1.430	1.490	60
101	1.352	1.408	56
110	1.274	1.327	53
111	1.196	1.246	50

3) SYSCREG_DCDC_CONVERTER_CLK

The DC/DC converter wil start with a ringoscillator to overule this you have to set the SYSCREG_DCDC_CONVERTER_CLK to '1', at that moment the the DC/DC converter will use a 12 MHz clk from the CGU. (for more inforamtion see DC/DC).

18.2.2 SYSCREG_FUSEBOX registers

Table 74 fusebox registers fo0 to fo3

BITS	VARIABLE	RESET	R/W
31:0	fusebox register bits	0xffffffff	R

Fusebox bits are used for DRM.

18.2.3 SYSCREG_SEL register

Table 75 sel register

BITS	VARIABLE	RESET	R/W
0	sel_usb20fs	0	R/W
1	sel_ata	0	R/W

Selection bits are used for the ATA glue and external USB2.0 see also documentation about this glue and external Usb2.0.

18.2.4 SYSCREG_SSA1_RTC_CFG register

Table 76 SSA1_RTC_CFG register

BITS	VARIABLE	RESET	R/W
0	ssa1_rtc_cfg_pwr_up	0	R/W

Powerbit rtc bit set '1' activates the RTC. See also chapter RTC.

18.2.5 SYSCREG_SSA1_ADC_PD_ADC10BITS register

Table 77 SSA1_ADC_PD_ADC10BITS register

BITS	VARIABLE	RESET	R/W
0	ssa1_adc_pd_adc10bits	0	R/W

Powerbit 10 bits adc set '0' activates the 10 bits adc. See also chapter 10 bits ADC.

18.2.6 SYSCREG_SSA1_MCI_PL180_PRESERVED register

Table 78 SSA1_MCI_PL180_PRESERVED register

BITS	VARIABLE	RESET	R/W
0	SSA1_MCI_PL180_PRESERVED	0	R/W

18.2.7 SYSCREG_USBAPB_TOP_STS register

USB status bits, see also chapter USB

Table 79 USB top sts register

BITS	VARIABLE	RESET	R/W
0	VBus power	0	R
1	Up_Led_n	1	R
2	Suspend	0	R

18.2.8 SYSCREG_USBAPB_TOP_CFG register

USB cfg bits, see also chapter USB

Table 80 USB top cfg register

BITS	VARIABLE	RESET	R/W
0	ATX Poweron	0	R/W
1	not connected	0	R/W
2	ATX RCV_Mux Ctrl	0	R/W
3	USB input Bypass Ctrl	0	R/W

18.2.9 SYSCREG_IRDA_SIRXT_UART_RXD_EN register

Table 81 IRDA_SIRXT_UART_RXD_EN register

BITS	VARIABLE	RESET	R/W
0	irda_sirxt_uart_rxd_en	0	R/W

Enable bit for irda selection, set to ’1’ enables the IRDA interface, see also chapter uart/irda.

18.2.10 SYSCREG_SW_IP_BLAS_2111_PWD_N register

Table 82 SW_IP_BLAS_2111_PWD_N register

BITS	VARIABLE	RESET	R/W
0	SW_IP_BLAS_2111_PWD_N	1	R/W

Powerbit Embedded Flash bit set ’1’ activates the Flash. See also chapter Embedded Flash.

18.2.11 SYSCREG_IP_BLAS_2111_RSTBUSY register

Table 83 IP_BLAS_2111_RSTBUSY register

BITS	VARIABLE	RESET	R/W
0	IP_BLAS_2111_RSTBUSY	-	R

Acknowledgement to the system controller that the Embedded Flash is being reset or that the Flash module is being initialized. See also chapter Embedded Flash.

18.2.12 SYSCREG_CGU_DYN_HP0 register

Table 84 CGU_DYN_HP0 register

BITS	VARIABLE	RESET	R/W
0	CGU_DYN_HP0	0	R/W

18.2.13 SYSCREG_CGU_DYN_LP0 register

Table 85 CGU_DYN_LP0 register

BITS	VARIABLE	RESET	R/W
0	CGU_DYN_LP0	0	R/W

18.2.14 SYSCREG_SYS_CREG_SDMA_EXT_EN_3 register

Table 86 SYS_CREG_SDMA_EXT_EN_3 register

BITS	VARIABLE	RESET	R/W
0	SYS_CREG_SDMA_EXT_EN_3	0	R/W

18.2.15 SYSCREG_SYS_CREG_SDMA_EXT_EN_5 register

Table 87 SYS_CREG_SDMA_EXT_EN_5 register

BITS	VARIABLE	RESET	R/W
0	SYS_CREG_SDMA_EXT_EN_5	0	R/W

18.2.16 SYSCREG_SELECTION_CFG register

Table 88 SELECTION_CFG register

BITS	VARIABLE	RESET	R/W
0	SELECTION_CFG	1	R/W

18.2.17 SYSCREG_ISRAM_LATENCY_CFG register

Table 89 ISRAM_LATENCY_CFG register

BITS	VARIABLE	R/W
00	0 waitstates	R/W
01	1 waitstate	R/W
11	2 waitstates	R/W

Waitstates programming for the internal Ram, see also chapter isram.

18.2.18 SYSCREG_ISROM_LATENCY_CFG register

Table 90 ISROM_LATENCY_CFGregister

BITS	VARIABLE	R/W
00	0 waitstates	R/W
01	1 waitstate	R/W
11	2 waitstates	R/W

Waitstates programming for the internal Rom, see also chapter isrom.

18.2.19 SYSCREG_AHB_MPMC_PL172_MISCregister

Table 91 AHB_MPMC_PL172_MISC register

BITS	VARIABLE	RESET	R/W
0	ahb_mpmc_pl172_misc_srefreq	0	R/W
1	reserved	0	R/W
2	reserved	1	R/W
3	ahb_mpmc_pl172_misc_stcs0pol	0	R/W
4	ahb_mpmc_pl172_misc_stcs1pol	0	R/W
5	ahb_mpmc_pl172_misc_stcs2pol	0	R/W
6	ahb_mpmc_pl172_misc_stcs3pol	0	R/W
7	ahb_mpmc_pl172_misc_stcs1pb	0	R/W
8	ahb_mpmc_pl172_misc_rel1config	0	R/W

see also chapter MPMC.

1) ahb_mpmc_pl172_misc_srefreq

Self refresh request, when ’1’ it indicates a self refresh request from the PMU.

2) ahb_mpmc_pl172_misc_stcs0pol

Polarity of static memory CS0. This power on reset value can be over written through the reigister interface. When ’1’, it indicates active HIGH chip select and when ’0’, it indicates an active LOW chip select.

3) ahb_mpmc_pl172_misc_stcs1pol

Polarity of static memory CS1. This power on reset value can be over written through the reigister interface. When ’1’, it indicates active HIGH chip select and when ’0’, it indicates an active LOW chip select.

4) ahb_mpmc_pl172_misc_stcs2pol

Polarity of static memory CS2. This power on reset value can be over written through the reigister interface. When ’1’, it indicates active HIGH chip select and when ’0’, it indicates an active LOW chip select.

5) ahb_mpmc_pl172_misc_stcs3pol

Polarity of static memory CS3. This power on reset value can be over written through the reigister interface. When ’1’, it indicates active HIGH chip select and when ’0’, it indicates an active LOW chip select.

6) ahb_mpmc_pl172_misc_stcs1pb

Polarity of byte lane select for static memory CS1. This power on reset value can be over written through the register interface. When ’1’, for reads and write, the respective active bits of nMPMCBLSOUT[3:0] are LOW. When ’0’, for reads, all the bits of nMPMCBLSOUT[3:0] are HIGH and for writes, the respective active bits of nMPMCBLSOUT[3:0] are LOW.

7) ahb_mpmc_pl172_misc_rel1config

Static memory address mode select.

When ’1’, it indicates the static memory addressing mode of PL172 release 1v0 in which the static memory address should be connected as follows:

- When external memory width is 8 bits, MPMCADDROUT[27:0] should be connected to the A[27:0] pins of the static memory device.

- When external memory width is 16 bits, MPMCADDROUT[27:1] should be connected to the A[26:0] pins of the static memory device and MPMCADDROUT[0] is not used.

- When external memory width is 32 bits, MPMCADDROUT[27:2] should be connected to the A[25:0] pins of the static memory device and MPMCADDROUT[1:0] is not used.

When ’0’, it indicates that the static memory addresses should be connected as follows:

- When external memory width is 8 bits or 16-bits or 32 bits, MPMCADDROUT[27:0] should be connected to the A[27:0] pins of the static memory device.

18.2.20 SYSCREG_MPMP_DELAYMODES register

Table 92 MPMC_DELAYMODES register

BITS	VARIABLE	RESET	R/W
2:0	MPMC_delaymodes	1	R/W
5:3	MPMC_delaymodes	0	R/W
8:6	MPMC_delaymodes	0	R/W

MPMC Memory controller feedback clock delay, see also MPMC chapter.

1) MPMC_delaymodes[2:0]

Configures the amount of delay cells between MPMCCLK and MPMCFBCLKIN. MPMCFBCLKIN is the feedback clock for SDRAM.

- 0xx: no delay cells

- 100: one delay cell

- 101: two delay cells

- 110: three delay cells

- 111: four delay cells

(delay cell = dly5x8pdv)

2) MPMC_delaymodes[5:3]

Configures the amount of delay cells, between MPMCCLK and MPMCCLKDELAY. This clock is used in Command Delayed strategy

.- 0xx: no delay cells

- 100: one delay cell

- 101: two delay cells

- 110: three delay cells

- 111: four delay cells

(delay cell = dly5x8pdv)

3) MPMC_delaymodes[8:6]

Configures the amount of delay cells, used for delaying MPMCCLKOUT. This 'clock out' goes to the external SDRAM.

.- 0xx: no delay cells

- 100: one delay cell

- 101: two delay cells

- 110: three delay cells

- 111: four delay cells

(delay cell = dly5x8pdv)

18.2.21 SYSCREG_WIRE_EBI_MSIZE_INIT register

Table 93 WIRE_EBI_MSIZE_INIT register

BITS	VARIABLE	RESET	R/W
[1:0]	wire_ebi_msize_init	0	R/W

wire_ebi_msize_init = ahb_mpmc_pl172_misc_stcs1mw[1:0], memory width of CS1. This power on reset value can be over written through the register interface. "00" indicates 8-bits, "01" indicates 16-bits and "10" indicates 32-bits.

18.2.22 SYSCREG_AHB_BOOT register

Table 94 AHB_BOOT register

BITS	VARIABLE	RESET	R/W
0	ahb_boot	0	R/W

Boot bit:

"0" indicates internal ROM (0x200000)

"1" indicates internal RAM (0x400000)

You can use it as a remap bit, you will always boot from internal ROM (POR value).

18.2.23 SYSCREG_ARM7TDMISCACHE_SHADOW_POINTER register

Table 95 ARM7TDMISCACHE_SHADOW_POINTER register

BITS	VARIABLE	RESET	R/W
[31:0]	arm7tdmiscache_shadow_pointer	0	R/W

A shadow option is provided to change the memory mapping.

The actual re-mapping pointer is a 32-bitvector,of which the lower10 bits are '0'. It is freely programmable in increments of 1 kByte.

see also chapter Multilayer.

18.2.24 SYSCREG_SLEEPSTATUS registers

Table 96 SLEEPSTATUS register

BITS	VARIABLE	RESET	R/W
31:0	SLEEPSTATUS bits	0xffffffff	R/W

18.2.25 SYSCREG_CHIP_ID register

Table 97 CHIP_ID register

BITS	VARIABLE	RESET	R/W
0	CHIP_ID	0x101100c	R

Chip ID for the PNX0101 N1C.

19 MMIO INTERRUPT CONTROLLER

19.1 Overview

The interrupt controller decodes all the interrupt requests issued by the on-chip peripherals. It supports up to 28 interrupt sources and has two outputs (NFIQ, NIRQ). The ARM core can have two distinct levels of priority on all interrupt sources, NFIQ the higher priority and NIRQ the lower. the interrupt controller shall operate as AHB slave.

Common features are:

- level active of all interrupt are high level
- software interrupt request capability associated to each request input
- observability of interrupt request state before masking
- support for nesting of interrupt service routines

Table 98 Available Interrupts

MODULE	INTERRUPT SOURCE	INTERRUPT BIT	INTERRUPT
EVENT ROUTER	CASCADED_IRQ_0	0	EVENT ROUTER IRQ0
	CASCADED_IRQ_1	1	EVENT ROUTER IRQ1
	CASCADED_IRQ_2	2	EVENT ROUTER IRQ2
	CASCADED_IRQ_3	3	EVENT ROUTER IRQ3
TIMER0	TIMER0_INTCT1	4	COUNT INT TIMER0
TIMER1	TIMER1_INTCT1	5	COUNT INT TIMER1
RTC	RTC_NINTR	6	RTC COUNTER INCREMENT INTERRUPT
	RTC_ALARM_LP	7	RTC ALARM CLOCK INTERRUPT
ADC 10 BIT	ADC 10 BIT	8	ADC INT
MULTIMEDIA CARD INTERFACE	MCI	9/10	COMMAND RESPONSE RECEIVE, CRC FAILED
			DATA BLOCK IS SENT/RECEIVE, CRC FAILED
			COMMAND RESPONSE TIMEOUT
			DATA TIMEOUT
			TRANSMIT FIFO UNDERRUN ERROR
			RECEIVE FIFO OVERRUN ERROR
			COMMAND RESPONSE RECEIVE, CRC OK
			COMMAND SENT
			DATA END
			START BIT NOT DETECTED
			COMMAND TRANSFER
			DATA TRANSMIT
			DATA RECEIVE
			TRANSMIT FIFO IS HALF EMPTY
			RECEIVE FIFO IS HALF FULL
			TRANSMIT FIFO IS FULL
			RECEIVE FIFO IS FULL
			TRANSMIT FIFO EMPTY
			RECEIVE FIFO EMPTY
			TRANSMIT FIFO DATA AVAILABLE
			RECEIVE FIFO DATA AVAILABLE
USB	USB REG-IRQ USB REG-FIQ FRAMETOGGLE	11/12/13	USB FRAME
			ENDPOINT 0 - 7
			DEVICE STATUS
			COMMAND CODE IS EMPTY
			COMMAND DATA IS FULL
			EOP FOR OUT TRANSFER IS REACHED
			EOP FOR IN TRANSFER IS REACHED
			ENDPOINT 1 BUFFER FULL
			ENDPOINT 3 BUFFER FULL
			ENDPOINT 5 BUFFER FULL
			ENDPOINT 7 BUFFER FULL
UART	UART-INT	14	RECEIVER ERROR FLAG
			RECEIVE DATA AVAILABLE
			TIME-OUT
			TRANSMIT HOLDING EMPTY

MODULE	INTERRUPT SOURCE	INTERRUPT BIT	INTERRUPT
IIC MASTER/SLAVE	IIC_MASTER	15	TRANSMIT DONE
			TRANSMIT ARBITRATION FAILURE
			TRANSMIT NO ACK
			MASTER TRANSMIT DATA REQUEST
			RECEIVE FIFO FULL
			RECEIVE FIFO EMPTY
			TRANSMIT FIFO FULL
			TRANSMIT FIFO EMPTY
AUDIO SUBSYSTEM	E7B2ARM_1	16	EPICS7B INT (FLAG 16)
	E7B2ARM_2	17	EPICS7B INT (FLAG 17)
	SAI1_IRQ	18	SAI1 INTERRUPT
	SAI2_IRQ	19	SAI2 INTERRUPT
	SAI3_IRQ	20	SAI3 INTERRUPT
	SAI4_IRQ	21	SAI4 INTERRUPT
	SAO1_IRQ	22	SAO1 INTERRUPT
	SAO2_IRQ	23	SAO2 INTERRUPT
	SPDIFIN_IRQ	24	SPDIFin INTERRUPT
FLASH INTERFACE	Flash	25	FINISHING BURNING OR ERASING INTERRUPT
LCD INTERFACE	LCD	26	LCD FIFO EMPTY
			LCD FIFO HALF EMPTY
			LCD FIFO OVERRUN
			LCD READ VALID
SDMA	IRQ	27	SDMA DATA TRANSFER COMPLETE

Interrupts routed to NIRQ and to NFIQ are vectored. That is to say that the processor can execute the interrupt handler corresponding to the current interrupt without testing each interrupt individually. Thus the software is minimized.

The interrupt vector register contains the start address of a specific ISR and an associated priority limiter value can be delivered if nesting of ISR shall be performed.

In this interrupt controller, a set of software accessible variables is provided to control interrupt request generation. It is essentially used in debug mode.

The interrupt controller supports interrupts which are level sensitive, asynchronous and can be active low or high.

19.2 Architecture

Fig.19 presents the functionality of the interrupt controller.

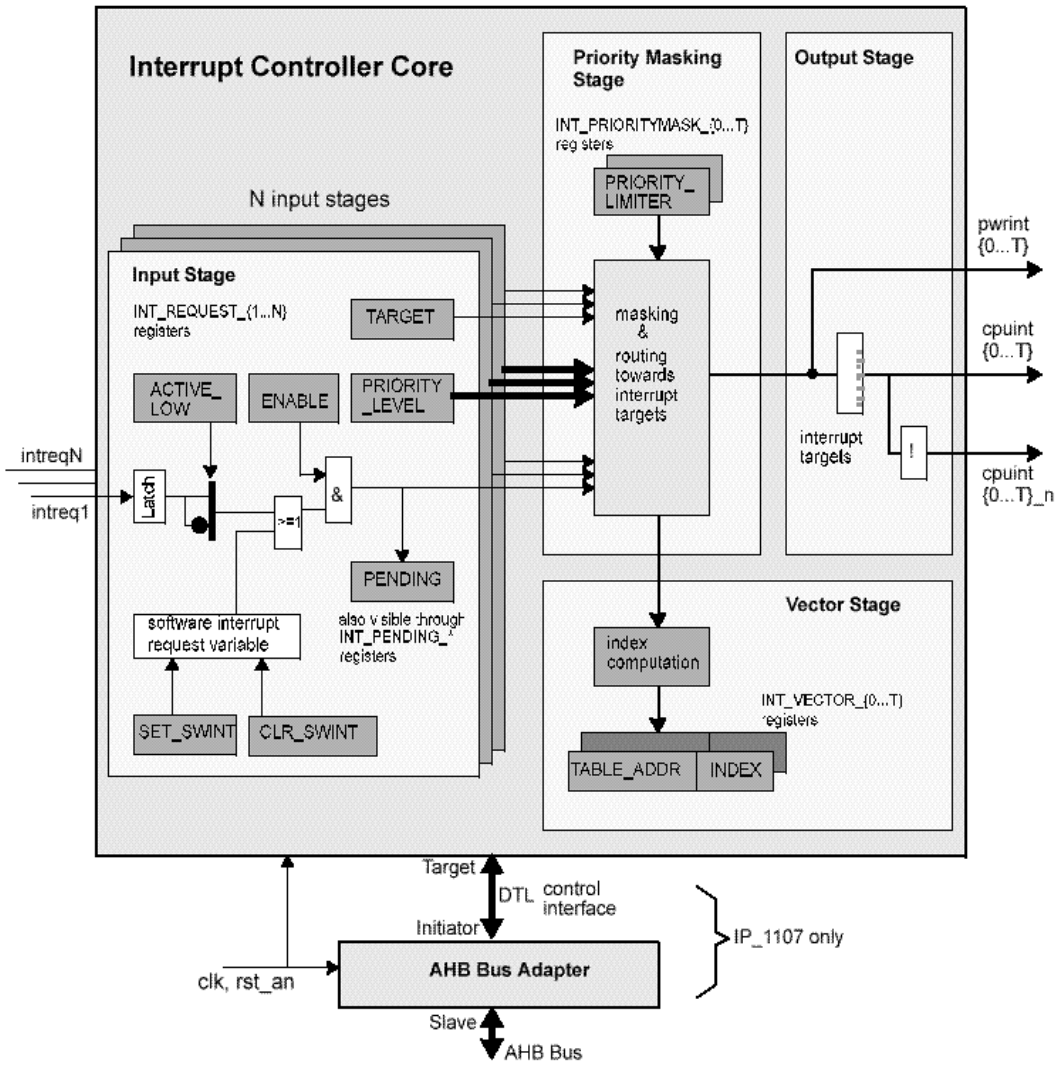


Fig. 19 Interrupt controller block diagram

19.2.1 INPUT STAGE

An input stage performs the following tasks (see Fig.19 on page 164):

- input of one interrupt request (intreq) signal,
- latch the interrupt request state during computation of the interrupt vector, otherwise keep the latch transparent,
- invert the request polarity if the interrupt request signal is active low (controlled by variable ACTIVE_LOW),
- combine the interrupt request with the state of a local software interrupt request variable,
- enable or disable the resulting interrupt request (controlled by variable ENABLE), and finally
- forward the request to the priority masking stage together with attributes characterising the interrupt request. These attributes are: - the priority level assigned to the request (variable PRIORITY_LEVEL) - the interrupt target defined for the request (variable TARGET).

In addition, the input stage provides means to set and clear the software interrupt request variable (SET_SWINT and CLR_SWINT commands) and to observe the request status before priority masking (variable PENDING).

While no interrupt vector is being computed, the signal path of an interrupt request throughout the input stage (including the latch) is asynchronous and requires no active interrupt controller clock. Immediately before vector computation the latch synchronously captures the state of the intreq line and thereby blocks any signal changes to propagate into the vector stage potentially leading to incorrect index computation. Sufficient time to resolve potential metastability of the latched request will be allowed. After vector computation, the latch becomes transparent again.

There is one input stage per interrupt request input. The number of input stages is scalable between 1 and 255. The vector stage references an input stage through an index. There is no input stage defined for index 0 as this index is reserved for a special purpose by the vector stage.

19.2.2 POLARITY MASKING STAGE

The priority masking stage performs the following tasks:

- for each interrupt target, input all interrupt requests selected for the target and mask pending interrupts which are at lower or equal priority than a target specific priority threshold (PRIORITY_LIMITER),
- for each interrupt target, combine pending interrupt requests with priority above the priority threshold through a logical OR and route the result towards the interrupt target.

The signal path of interrupt requests throughout the priority masking stage towards the output stage is asynchronous and requires no active interrupt controller clock.

19.2.3 OUTPUT STAGE

The output stage performs the following tasks:

- for each interrupt target, produce processor interrupt request output signals cpuint{0...1}(_n) at both active high and low level by registering the interrupt request information of the priority masking stage,

The interrupt controller introduces an interrupt latency (measured from assertion of an intreq signal to an assertion of cpuint{0..1}(_n)) of less than 2 clock periods.

19.2.4 VECTOR STAGE

The vector stage provides one vector register per interrupt target (INT_VECTOR_{0...1}). It performs the following tasks triggered by a read action to one of these registers:

- for a read of register INT_VECTOR_t, process the PRIORITY information of input states with pending interrupt requests selected for TARGET = t,
- then identify the input stage with the highest PRIORITY value above the target specific PRIORITY_LIMITER threshold (if this condition is true for a multitude of input stages, then the input stage with the highest index is taken), and finally

- present the index of that input stage through the INDEX variable in the INT_VECTOR_t register. If no interrupt request exceeds the PRIORITY_LIMITER threshold, then INDEX=0 is given.

The above process is performed upon any INT_VECTOR_* read action - there is no storage of a previously computed vector.

The information from the INT_VECTOR_* register facilitates a generic software ISR in identifying the interrupt requesting device to be serviced. To invoke the ISR of that device, the INDEX variable can be taken as offset into a table of address pointers towards device specific ISR. Alternatively, the total content of the INT_VECTOR_* register, consisting of a table base address (variable TABLE_ADDR) plus INDEX, can be taken as pointer into a table.

INDEX = 0 identifies the special case that no interrupt request requires service when the INT_VECTOR_* register is read.

For correct vector computation, it is required that the ISR always reads the INT_VECTOR_* register that corresponds to the interrupt target.

19.2.5 CONFIGURATION

The interrupt module is extracted from HDLi with the configuration defined in table 99 on page 166

Table 99 Interrupt Controller Configuration of PNX0101

PARAMETER	DESCRIPTION	RANGE	CONFIGURATION
Number of IRQ requests	Number of interrupt request inputs	integer range3 < N < 256	N = 28
Number of priority bits	Number of priority levels supported minus one	P=3, 7, 15, 31, 63, 127, 255 and P <= N	P = 15
Number of IRQ targets	Number of interrupt output targets minus one	0 <= T < 16	T = 1, number of targets output is 2
Clock gating	Provide clock gating for test purposes	YES|NO	NO
DFT implementation	Provide the design with scanchains for testability	YES|NO	NO
Target clock frequency	What is clock frequency of the interrupt controller	MHz	80MHz

19.3 Software Interface

19.3.1 Register Overview

The interrupt controller provides the registers described in Table 100 to give software access to its internal variables.

Table 100 Interrupt Controller Software Interface

REGISTER NAME	OFFSET TO MMIO BASE ADDRESS OF INTERRUPT CONTROLLER	DESCRIPTION
INT_PRIORITYMASK_0	0x000	interrupt target 0 priority threshold
INT_PRIORITYMASK_1	0x004	interrupt target 1 priority threshold
INT_VECTOR_0	0x100	interrupt target 0 = nIRQ
INT_VECTOR_1	0x104	interrupt target 1 = nFIQ
INT_PENDING_1_28	0x200	status of interrupt request 1..28
INT_FEATURES	0x300	interrupt controller configuration features
INT_REQUEST_1	0x404	interrupt request 1 configuration features
INT_REQUEST_2	0x408	interrupt request 2 configuration features
INT_REQUEST_3	0x40C	interrupt request 3 configuration features
INT_REQUEST_4	0x410	interrupt request 4 configuration features
INT_REQUEST_5	0x414	interrupt request 5 configuration features
INT_REQUEST_6	0x418	interrupt request 6 configuration features
INT_REQUEST_7	0x41C	interrupt request 7 configuration features
INT_REQUEST_8	0x420	interrupt request 8 configuration features
INT_REQUEST_9	0x424	interrupt request 9 configuration features
INT_REQUEST_10	0x428	interrupt request 10 configuration features
INT_REQUEST_11	0x42C	interrupt request 11 configuration features
INT_REQUEST_12	0x430	interrupt request 12 configuration features
INT_REQUEST_13	0x434	interrupt request 13 configuration features
INT_REQUEST_14	0x438	interrupt request 14 configuration features
INT_REQUEST_15	0x43C	interrupt request 15 configuration features
INT_REQUEST_16	0x440	interrupt request 16 configuration features
INT_REQUEST_17	0x444	interrupt request 17 configuration features
INT_REQUEST_18	0x448	interrupt request 18 configuration features
INT_REQUEST_19	0x44C	interrupt request 19 configuration features
INT_REQUEST_20	0x450	interrupt request 20 configuration features
INT_REQUEST_21	0x454	interrupt request 21 configuration features
INT_REQUEST_22	0x458	interrupt request 22 configuration features
INT_REQUEST_23	0x45C	interrupt request 23 configuration features
INT_REQUEST_24	0x460	interrupt request 24 configuration features
INT_REQUEST_25	0x464	interrupt request 25 configuration features
INT_REQUEST_26	0x468	interrupt request 26 configuration features
INT_REQUEST_27	0x46C	interrupt request 27 configuration features
INT_REQUEST_28	0x470	interrupt request 28 configuration features
INT_MOD_ID	0xFFC	interrupt controller module ID

19.3.2 INTERRUPT TARGETS

The interrupt controller can route incoming interrupt requests towards a potential multitude of processor interrupt request outputs, also referred as "interrupt targets". Interrupt targets can be assigned to a single processor - then allowing to trigger distinct types of interrupt service from that processor, or to different processors.

The application of interrupt targets is not prescribed by the architecture. It may be specific to a system hardware/software design and may depend on the capabilities of the processor handling the interrupts. The interrupt architecture recommends the following use of interrupt targets:

- ARM processor: target 0 = nIRQ (standard interrupt service with full context state save/restore) target 1 = nFIQ (fast interrupt service with minimal context save/restore)

The interrupt target is configured for each interrupt request input of the interrupt controller through the TARGET variable in the INT_REQUEST_* registers.

19.3.3 INTERRUPT PRIORITY

Interrupt request masking is performed individually per interrupt target by comparing the priority level assigned to a specific interrupt request input (variable PRIORITY_LEVEL in the INT_REQUEST_* registers) with a target specific priority threshold (variable PRIORITY_LIMITER in the INT_PRIORITYMASK_* registers).

Priority levels are defined as follows:

- Priority level 0 corresponds to 'masked'. Interrupt requests with priority 0 will never lead to an interrupt request towards processor or power management controller.
- Priority level 1 corresponds to lowest priority.
- Priority level P corresponds to highest priority.

Programming the INT_REQUEST_* register variable ENABLE = 0 is an alternative to PRIORITY_LEVEL = 0 which is typically applied when an interrupt request input shall be temporarily disabled without the need to save and restore the current PRIORITY_LEVEL setting.

19.3.4 SOFTWARE INTERRUPTS

Software interrupt support is provided through variables in the INT_REQUEST_* registers. Software interrupts can be applied for:

- test the RTOS interrupt handling without using a device specific ISR
- software emulation of an interrupt requesting device, including interrupts
- inter-processor signalling: e.g. if the interrupt controller supports 2 interrupt targets (target 0 connected to processor 0 and target 1 connected to processor 1), then processor 0 can issue a software interrupt request to target 1 for signalling purpose. Processor 1 then incurs an interrupt and clears the software interrupt request during the ISR. A prerequisite for this application is that both processors have access to the interrupt controller.

19.3.5 INTERRUPT PRIORITY MASK REGISTER

inter-processor signalling: e.g. if the interrupt controller supports 2 interrupt targets (target 0 connected to processor 0 and target 1 connected to processor 1), then processor 0 can issue a software interrupt request to target 1 for signalling purpose. Processor 1 then incurs an interrupt and clears the software interrupt request during the ISR. A prerequisite for this application is that both processors have access to the interrupt controller.

Table 101INT_PRIORITYMASK_{0..1}

BITS	READ/WRITE	RESET VALUE	NAME	DESCRIPTION
[7:0]	R/W	X	PRIORITY_LIMITER	Priority Limiter: this variable determines a priority threshold that incoming interrupt requests must exceed to trigger interrupt requests towards processor and power management controller. Legal PRIORITY_LIMITER values are 0 .. P; other values are reserved and lead to undefined behaviour. PRIORITY_LIMITER = 0: incoming interrupt requests with priority >0 can trigger interrupt requests towards processor and power management controller PRIORITY_LIMITER = n: only interrupt requests at a priority level above n can trigger interrupt requests towards processor and power management controller ... PRIORITY_LIMITER = P: no incoming interrupt requests can trigger interrupt requests towards processor and power management controller. high order bits not required for PRIORITY_LIMITER encoding are read-only 0
[31:8]	R	X		Reserved for future extensions; should be written as 0

The number of priority levels P is implementation specific.

The PRIORITY_LIMITER variable can be used to define the minimum priority level for nesting interrupts: typically, the PRIORITY_LIMITER variable is set to the priority level of the ISR that is currently being executed. By doing this, only interrupt requests at a higher priority level will lead to a nested interrupt service. Nesting can be disabled by setting PRIORITY_LEVEL = P or by disabling interrupt exceptions within the processor.

19.3.6 INTERRUPT VECTOR REGISTER

These registers identify - individually for each interrupt target - the highest priority enabled pending interrupt request that is present at the time when a register is being read.

Table 102INT_VECTOR_{0..1} Registers

BITS	READ/WRITE	RESET VALUE	NAME	DESCRIPTION
[2:0]	R	0	NULL	bit field always read as zero
[10:3]	R	X	INDEX	Index: indicates the intreq line number of the interrupt request to be served by the processor: INDEX = 0: no interrupt request to be served INDEX = 1: serve interrupt request at input intreq1 INDEX = 2: serve interrupt request at input intreq2 ... INDEX = N: serve interrupt request at input intreqN
[31:11]	R/W	X	TABLE_ADDR	Table start address: indicates the lower address boundary of a 2048 byte aligned interrupt vector table in memory

The software ISR must always read the vector register that corresponds to the interrupt tar-get, e.g.:

- read INT_VECTOR_0 for interrupt target 0 service,
- read INT_VECTOR_1 for interrupt target 1 service.

The INT_VECTOR_* register content can be used as a vector into a memory based table. This table has N+1 entries. To be able to use the register content as a full 32 bit address pointer, the table must be aligned to a 2048 byte address boundary. If only the INDEX variable is used as offset into the table, then this address alignment is not required.

Each table entry has 64 bit data. It is recommended to pack per table entry:

- the start address of a device specific ISR, plus
- the associated priority limiter value (if nesting of ISR shall be performed).

64 bit packing will optimize the speed of nested interrupt handling by processors that implement caches. In the (likely) case of a cache miss when reading data from the table, the priority limiter value to be programmed into the INT_PRIORITYMASK register will be loaded into the cache along with the ISR start address, saving several clock cycles interrupt processing time compared to a solution where the priority limiter value would have to be established from an INT_REQUEST_* register.

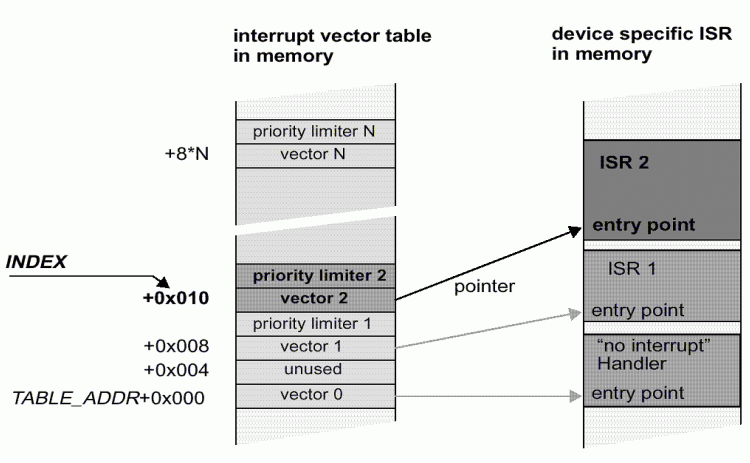


Fig. 20 Memory Based Interrupt Vector/Priority Table

A vector with INDEX = 0 indicates that no interrupt with priority above the priority threshold is pending. The vector table should implement for this entry a "no interrupt" handler to treat this special case.

Note that due to the special purpose of INDEX = 0 no interrupt request input intreq0 and thus no INT_REQUEST_0 register exists.

19.3.7 INTERRUPT REQUEST REGISTERS (INT_REQUEST_{1..28}

This set of registers holds configuration information related to interrupt request inputs of the interrupt controller and allows to issue software interrupt requests.

There are N registers, one for each intreq input signal. N is a configuration parameter.

Table 103INT_REQUEST_{0..28} Registers

BITS	READ/ WRITE	RESET VALUE	NAME	DESCRIPTION
31	R	X	PENDING	Pending interrupt request: This variable reflects the state of the intreq line (if needed, converted to active high) OR'ed by the state of the local soft-ware interrupt request variable at the time the register is read. Note that the PENDING variable is also visible from the INT_PENDING_* registers. PENDING = 0: no interrupt request PENDING = 1: interrupt request pending
30	W	0	SET_SWINT	Set software interrupt request SET_SW_INT = 0 (write): no effect on the state of the local software interrupt request variable SET_SWINT = 1 (write): set the state of the local software interrupt request variable to '1' SET_SWINT is always reads as 0
29	W	0	CLR_SWINT	Clear software interrupt request: CLR_SWINT = 0 (write): clear the state of the local software interrupt request variable to '0' CLR_SWINT is always read as 0
28	W	n.a.	WE_PRIORITY_LEVEL	Write Enable PRIORITY_LEVEL WE_PRIORITY_LEVEL = 0 (write): no change of PRIORITY_LEVEL variable state WE_PRIORITY_LEVEL = 1 (write): PRIORITY_LEVEL variable state may be changed WE_PRIORITY_LEVEL is always read as 0
27	W	n.a.	WE_TARGET	Write Enable TARGET WE_TARGET = 0 (write): no change of TARGET variable state WE_TARGET = 1 (write): TARGET variable state may be changed WE_TARGET is always read as 0
26	W	n.a.	WE_ENABLE	Write Enable ENABLE WE_ENABLE = 0 (write): no change of ENABLE variable state WE_ENABLE = 1 (write): ENABLE variable state may be changed WE_ENABLE is always read as 0

BITS	READ/ WRITE	RESET VALUE	NAME	DESCRIPTION
25	W	n.a.	WE_ACTIVE_LOW	Write Enable ACTIVE_LOW WE_ACTIVE_LOW = 0 (write): no change of ACTIVE_LOW variable state WE_ACTIVE_LOW = 1 (write): ACTIVE_LOW variable state may be changed WE_ACTIVE_LOW is always read as 0
[24:18]	R	X		Reserved; should be written as zeros
17	R/W	0	ACTIVE_LOW	Active Low This variable selects the polarity of the interrupt request input signal. See also WE_ACTIVE_LOW. ACTIVE_LOW = 1: the intreq signal is interpreted as active low ACTIVE_LOW = 0: the intreq signal is interpreted as active high
16	R/W	0	ENABLE	Enable interrupt request This variable controls whether an interrupt request is enabled for further processing by the interrupt controller. See also WE_ENABLE. ENABLE = 0: the interrupt request is discarded. It cannot cause a processor or power management interrupt request. ENABLE = 1: the interrupt request may cause a processor or power management interrupt request when further conditions for this become true.

BITS	READ/ WRITE	RESET VALUE	NAME	DESCRIPTION
[15:14]	R	X		Reserved; should be written as zeros
[13:8]	R/W	0	TARGET	Interrupt target: This variable defines the interrupt target of an interrupt request. Legal values are 0 ... 1; other values are reserved and lead to undefined behaviour. See also WE_TARGET. TARGET = 0: the interrupt request shall lead to a processor interrupt request 0 (cpuint0) TARGET = 1: the interrupt request shall lead to a processor interrupt request 1 (cpuint1) ... TARGET = T: the interrupt request shall lead to a processor interrupt request T (cpuintT) High order bits not required for TARGET encoding are read-only 0. For a single-target configuration (T = 0), the TARGET field must always be written as 0.
[7:0]	R/W	X	PRIORITY_LEVEL	Priority level This variable determines the priority level of the interrupt request. Legal values are 0 ... P; other values are reserved and lead to undefined behaviour. See also WE_PRIORITY_LEVEL. PRIORITY_LEVEL = 0: the interrupt request has priority level 0 (masked); it is ignored PRIORITY_LEVEL = 1: the interrupt request has priority level 1 (lowest) ... PRIORITY_LEVEL = P: the interrupt request has priority level P (highest) High order bits not required for PRIORITY_LEVEL encoding are read-only 0

The number of priority levels P and interrupt targets T is implementation specific (see section 19.2.5 “Configuration”).

Note that there is no INT_REQUEST_0 register.

For changing the TARGET variable state dynamically, software must first disable the interrupt request (ENABLE = 0), then change TARGET and finally re-enable the request (ENABLE = 1).

Write enable commands are provided to allow the modification of individual INT_REQUEST_* variables by simple write operations instead of atomic read-modify-write operations. This feature allows to access INT_REQUEST_* registers simultaneously by multiple software threads.

19.3.8 Interrupt Pending Registers

This set of registers gathers the PENDING variables of all interrupt requests.

Software can make use of INT_PENDING registers to gain a faster overview on pending interrupt requests than by reading individual INT_REQUEST_* registers. For certain ISR software this may lead to a benefit in interrupt processing time.

There are (N div 32)+1 registers, with N equal to the number of interrupt request inputs of the interrupt controller. N is a configuration parameter.

INT_PENDING_1_28 reflects the state of signals intreq1...28.

Table 104INT_PENDING_28 Registers

BITS	READ/ WRITE	RESET VALUE	NAME	DESCRIPTION
i	R	X	PENDING[i]	Pending interrupt request: This variable reflects the state of the intreq[i] line (if needed, converted to active high) OR'ed by the state of the local soft-ware interrupt request variable at the time the register is read. Note that the pending variables are also reflected by the INT_REQUEST_28 registers individually present for each interrupt request input. PENDING = 0: no interrupt request PENDING = 1: interrupt request is pending

19.3.9 Interrupt Controller Features Register (INT_FEATURES)

This register indicates the hardware configuration parameters chosen during the creation of the interrupt controller. A generic ISR can make use of the INT_FEATURES register to implement interrupt controller configuration specific behaviour.

Table 105INT_FEATURES Registers

BITS	READ/ WRITE	RESET VALUE	NAME	DESCRIPTION
[31:22]	R	X		Reserved
[21:16]	R	T	T	Configuration parameter T: number of interrupt targets supported (minus one)
[15:8]	R	P	P	Configuration parameter P: number of priority levels supported
[7:0]	R	N	N	Configuration parameter N: number of interrupt request inputs

19.3.10 Interrupt Controller Module ID Register

This register identifies the type and revision of the interrupt controller. A generic ISR can make use of the INT_MOD_ID register to implement interrupt controller type or revision specific behaviour.

Table 106INT_MOD

BITS	R/W	RESET VALUE	NAME	DESCRIPTION
31 : 16	R	0x1106	ID	**Identification number** Unique module identifier indicating the type of interrupt controller.
15 : 12	R	1	MAJOR_REV	**Major Revision** Major revision of module implementation Note that there is no guaranteed software compatibility.
11:8	R	o	MINOR_REV	**Minor Revision** Minor revision of module implementation, starting at 0. There is software compatibility between minor revisions.
7:0	R	0	APERTURE_SIZE	**Aperture Size** 4 KByte address aperture

20 IOCONF

20.1 Block Diagram

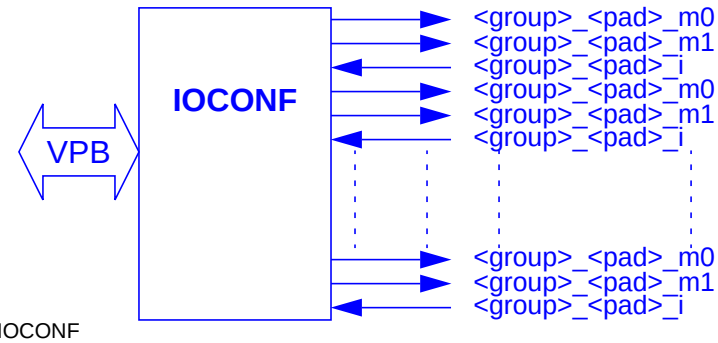


Fig. 20 IOCONF

20.2 Features

- Designed to be used with template PADMUX.
- Provides control/observe signals to/from a pad multiplexor.
- Each controlled pad can be configured for 4 operation modes. Typical: controlled by a function block (normal operation), driven low, high or high-Z.
- A pad can be observed (read) in any mode.
- Each controlled pad can be set to any mode during system reset.
- Conforms to VPB interface specification.
- Up to 32 pads can be defined and grouped into one function block.
- Provides set and clear access methods.

20.3 Overview

The Input/Output configuration (IOCONF) is designed to provide developers a set of registers. This can be used for configuration of various on chip components especially a pad multiplexer. This block conforms to the VLSI Peripheral Bus (VPB) specification for ease of use with other VPB peripherals. The register in this block are clocked by pclk.

The IOCONF block is used to provide individual control and visibility for a (relatively large) set of pads. In conjunction with a set of pad multiplexers, individual pads can be switched either in normal operation mode, or in "GPIO" mode. In GPIO mode, a pad is fully controllable. Through the IOCONF, individual pad levels can be observed in both "normal" and "GPIO" modes.

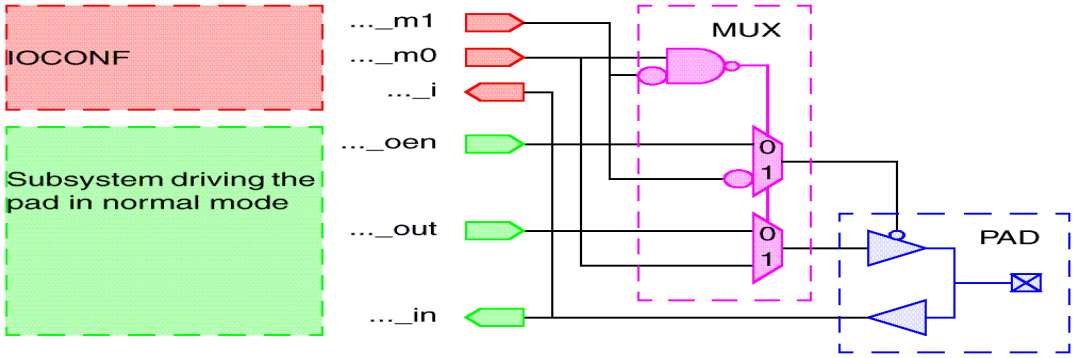
Functional pads can be grouped into function blocks. For each function block 1 to 32 pads can be defined. 1 to 64 function blocks can be defined. All output values in a function block can be set simultaneously by accessing a single register. Changing modes for all pads within a function block requires at most two register access. All input values in a function block can be read simultaneously by accessing a single register. Input values are not registered and always read directly from the pad’s input driver regardless of the mode of the pad.

For each function block there are two registers holding the control mode. Mode bit 1 leaves the IOCONF inverted as it is intended to be used as inverted (output-) enable. Each register is writeable and readable, has configurable pad names per bit (32 max) and provides set and clear access methods (set/clear bit when ’1’), and configurable reset value. Configurable pad names are provided in order to enhance readability and consistency of both HDL and generated C header file.

For an application to a single multiplexed pad, please refer to the "PADMUX" example schematic in the figure below. This shows a simplified solution, a more sophisticated pad logic will probably include boundary scan, production scan test logic, provision for multiple power domains and wire delays.

In this example IOCONF should control the selection (sel) between "GPIO" and "normal" mode. It also needs to control output (out) and inverted output

A separate part of the GPIO block is an interrupt controller which can be programmed to generate IRQ's on any change of most of the input pins.



In this example IOCONF should control the selection (sel) between "GPIO" and "normal" mode. It also needs to control output (out) and inverted output enable (oen) for GPIO mode. This gives us 3 wires that control 4 modes. 2 wires are sufficient to control 4 modes. Using two wires reduces the amount of top level wiring from IOCONF to pads with 33%. Because "out" is redundant when "oen"='1' we can use "out" to switch between "Normal" and "GPIO" High-Z mode.

Note that "m0" is used as GPIO (Generic Port I/O) "out", "m1" as GPIO "oe" and the special case "m0"=1 and "m1"=0 switches to "out" and "oen" from the subsystem driving the pad in normal (none-GPIO) mode. When output is enabled, "m0" controls the data being output. When output is disabled "m0" controls selection between GPIO and normal mode. Refer to following table for the "PADMUX" function. This table shows the PAD state related to the mode register inside IOCONF bits 0 and 1.

Table 107MODE0/1

MODE1 (OE)	MODE0 (DATA)	PAD
0	0	GPIO, high impedance
0	1	Normal operation, controlled by sub system
1	0	GPIO, drive low
1	1	GPIO, drive high

20.4 Pin Description

The GPIO interfaces to a standard VPB bus with unidirectional data bus. The data bus is 32-bits wide. Refer to VPB Peripherals document for a description of the VPB bus.

The non-VPB signals are *_i, *_m0, *_m1. The prefix for these signals is <block>_<pad)>. Table 108 gives a brief summary of each of these IOCONF pins.

20.5 Register Definition

Table below shows the structure "IOCONF_REGS" for any IOCONF function block:

Table 108 Register Definition

IOCONF_REGS FIELD NAME	OFFSET	WRITE OPERATION	READ OPERATION
pins	0x00	(Invalid)	Read inputs
reserved	0x04	-	-
reserved	0x08	-	-
reserved	0x0C	-	-
mode0	0x10	Load	Read mode 0
mode0_set	0x14	Set bits	Read mode 0
mode0_reset	0x18	Reset bits	Read mode 0
reserved	0x1C	-	-
mode1	0x20	Load	Read mode 1
mode1_set	0x24	Set bits	Read mode 1
mode1_reset	0x28	Reset bits	Read mode 1
reserved	0x2C	-	-

Table below shows the structure "IOCONF_BLOCKS" when IOCONF is configured for three function blocks:

Table 109 IOCONFIG Function blocks

BLOCK NAME	OFFSET	BIT	PIN NAME
AHB_MPMC_PL172_0	0x000	0	MPMC_D_0
		1	MPMC_D_1
		2	MPMC_D_2
		3	MPMC_D_3
		4	MPMC_D_4
		5	MPMC_D_5
		6	MPMC_D_6
		7	MPMC_D_7
		8	MPMC_D_8
		9	MPMC_D_9
		10	MPMC_D_10
		11	MPMC_D_11
		12	MPMC_D_12
		13	MPMC_D_13
		14	MPMC_D_14
		15	MPMC_D_15
		16	MPMC_A_0
		17	MPMC_A_1
		18	MPMC_A_2
		19	MPMC_A_3
		20	MPMC_A_4
		21	MPMC_A_5
		22	MPMC_A_6
		23	MPMC_A_7
		24	MPMC_A_8
		25	MPMC_A_9
		26	MPMC_A_10
		27	MPMC_A_11
		28	MPMC_A_12
		29	MPMC_A_13
		30	MPMC_A_14
		31	MPMC_A_15
AHB_MPMC_PL172_1	0x040	0	MPMC_A_16
		1	MPMC_A_17
		2	MPMC_A_18
		3	MPMC_A_19
		4	MPMC_A_20
		5	MPMC_NSTCS_0
		6	MPMC_NSTCS_1
		7	MPMC_NSTCS_2

BLOCK NAME	OFFSET	BIT	PIN NAME
		8	MPMC_NDYCS
		9	MPMC_CKE
		10	MPMC_DQM_0
		11	MPMC_DQM_1
		12	MPMC_BLOUT_0
		13	MPMC_BLOUT_1
		14	MPMC_CLKOUT
		15	MPMC_NWE
		16	MPMC_NCAS
		17	MPMC_NRAS
		18	MPMC_NOE
		19	MPMC_RPOUT
GPIO	0x080	0	GPIO_0
		1	GPIO_1
		2	GPIO_2
		3	GPIO_3
MELODY_ADSS	0x0C0	0	DAI_DATA
		1	DAI_BCK
		2	DAI_WS
		3	DAO_CLK
		4	Reserved
		5	DAO_BCK
		6	DAO_DATA
LCD_INTERFACE	0x100	0	LCD_CSB
		1	LCD_RS
		2	LCD_RW_WR
		3	LCD_E_RD
		4	LCD_DB_0
		5	LCD_DB_1
		6	LCD_DB_2
		7	LCD_DB_3
		8	LCD_DB_4
		9	LCD_DB_5
		10	LCD_DB_6
		11	LCD_DB_7
MCI_PL180	0x140	0	MCI_CLK
		1	MCI_CMD
		2	MCI_DAT_3
		3	MCI_DAT_2
		4	MCI_DAT_1
		5	MCI_DAT_0

BLOCK NAME	OFFSET	BIT	PIN NAME
UART_IP3106	0x180	0	UART_RXD
		1	UART_TXD
		2	UART_CTS_N
		3	UART_RTS_N
USB	0x1C0	0	USBAPB_TOP_VBUS
		1	USBAPB_TOP_CON_N

20.6 PNX0101 Boot Flow

20.6.1 General

- The boot code is responsible for establishing start-up context and initializing the system to a pre-determined and well known state which allows code to be executed.
- The boot code should allow for quick verification of IC functionality

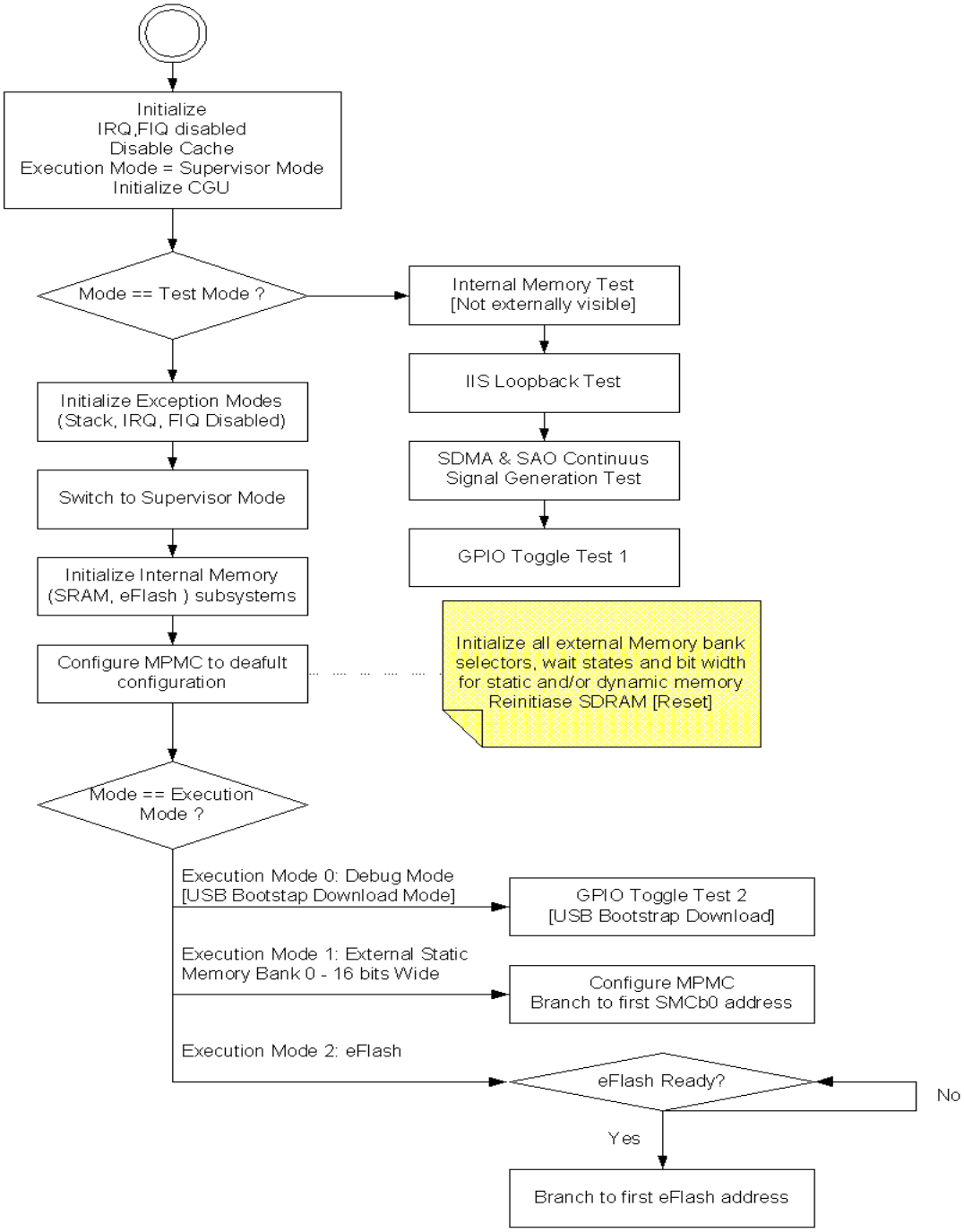
20.6.2 Boot Code Overview

Table 110 Mode Pins PNX0101

MODE PINS M [1:0]	MODE
0b00	Execution 2: Internal eFlash
0b01	Execution 1: External Static Memory Bank 0 -16 bits Wide Download [Debug] Mode USB
0b10	[USB Bootstrap download to internal SRAM]
0b11	autoTest Mode

Mode Pin M[0] = GPIO_2

Mode Pin M[1] = GPIO_3


Initialize
IRQ,FIQ disabled
Disable Cache
Execution Mode = Supervisor Mode
Initialize CGU
Mode == Test Mode ?
Internal Memory Test
[Not externally visible]
IIS Loopback Test
SDMA & SAO Continuus
Signal Generation Test
GPIO Toggle Test 1
Initialize Exception Modes
(Stack, IRQ, FIQ Disabled)
Switch to Supervisor Mode
Initialize Internal Memory
(SRAM, eFlash) subsystems
Configure MPMC to deafult
configuration
Initialize all external Memory bank
selectors, wait states and bit width
for static and/or dynamic memory
Reinitiase SDRAM [Reset]
Mode == Execution
Mode ?
Execution Mode 0: Debug Mode
[USB Bootstap Download Mode]
GPIO Toggle Test 2
[USB Bootstrap Download]
Execution Mode 1: External Static
Memory Bank 0 - 16 bits Wide
Configure MPMC
Branch to first SMCb0 address
Execution Mode 2: eFlash
eFlash Ready?
No
Yes
Branch to first eFlash address

21 EVENT_ROUTER

21.1 Overview

The event router provides bus control over the interrupt system. The bus interface is designed to the VPB specification. The event router is configured for 103 event sources. The event sources can be defined, their polarity and activation type to be selected. It also allows each input to be routed to any output(s) at reset. The outputs are defined by name.
This module can be used in low power systems to request power-up or start a clock on an external or internal event. It can also be used to generate interrupts as a result the events.

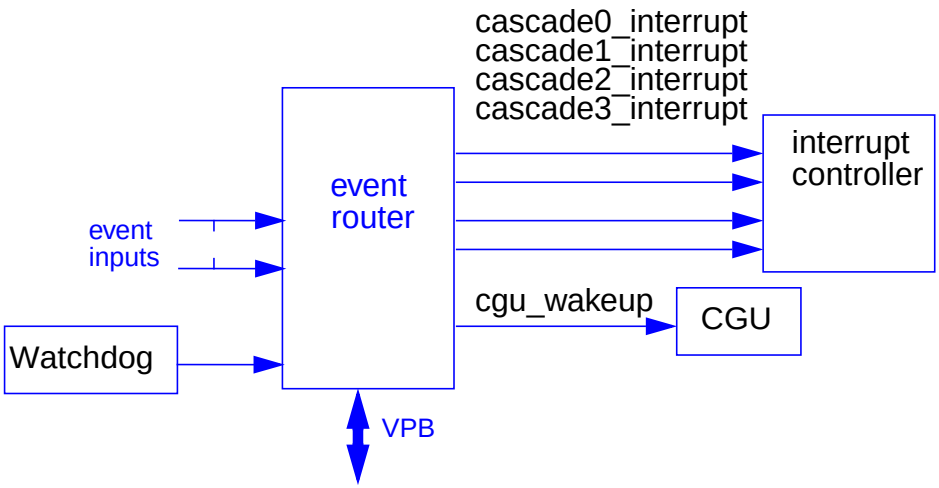


Fig. 23 Block Diagram

In table 115 on page 191all events source are listed which are connected to the event inputs. All events are coming from the pins of the design. Except for the RTC, this block can generate an counter interrupt which can cause an event trigger on the event router.

21.2 Features

- Provides bus controlled routing of input events to multiple outputs for use as interrupts or wakeup signals.
- Conforms to the VPB interface specification.
- Input events can be used either directly or latched (edge detected) as an interrupt source.
  - Direct interrupts will disappear when the event becomes inactive.
  - Latched interrupts will remain active until they are explicitly cleared.
- Interrupt events can be inverted (programmable).
- Each interrupt can be masked on event level.
- Interrupt event detect status can be read per interrupt type.
- Interrupt detection is fully asynchronous (no active clk required)

21.3 Architecture

The event router block is accessible through a VPB interface. Due to the maximum VPB data size of 32bits this also forms the limit to the number of outputs that can be generated.

Input events are processed in event slices, one for each event signal. Each of these slices generates one event signal and is visible in the rsr register. These events are then anded with enables from the mask register to give pending event status. All events are connected to an output slice for each output. In an output slice the signals from all inputs can be enabled or disabled to generate that output. There is a separate pending, mask, maskClr and maskSet register for each output slice.

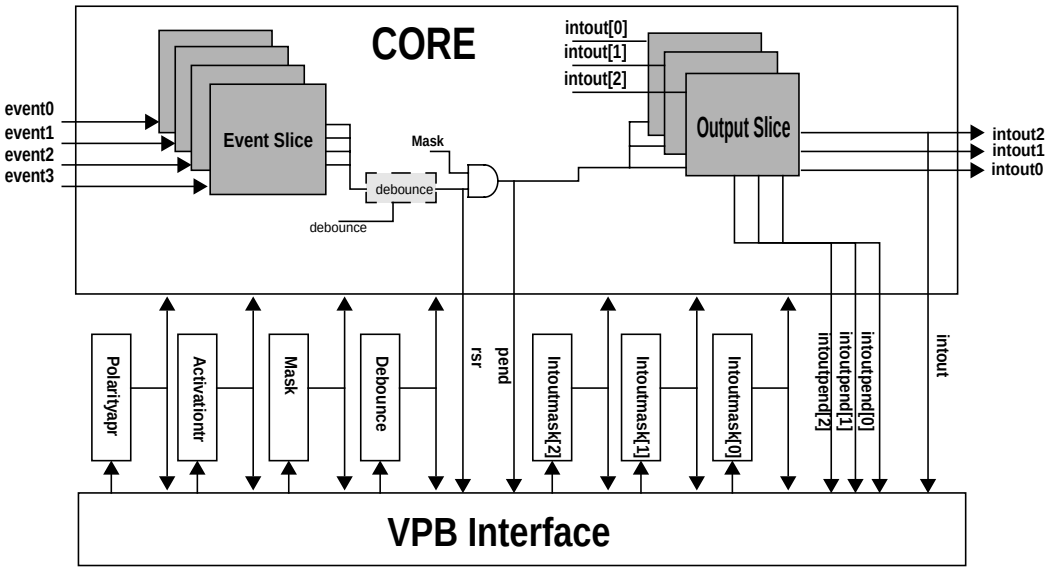


Fig. 24 extIntC Block Diagram

An event slice is controlled through bits in the polarity, activation, intSet and intClr registers.

- The polarity setting conditionally inverts the interrupt input event.
- The activation setting selects between latched or direct event.
- The resulting interrupt event is visible through a read action on the raw status register.
- Edge detection is performed by two registers with set functions. One detects the signal being inactive, the other detects the signal being active Latched.
- These interrupt values are visible through read actions on the status registers.
- A write ’1’ on the corresponding slice index in the intClr will clear the latched interrupt event synchronously.

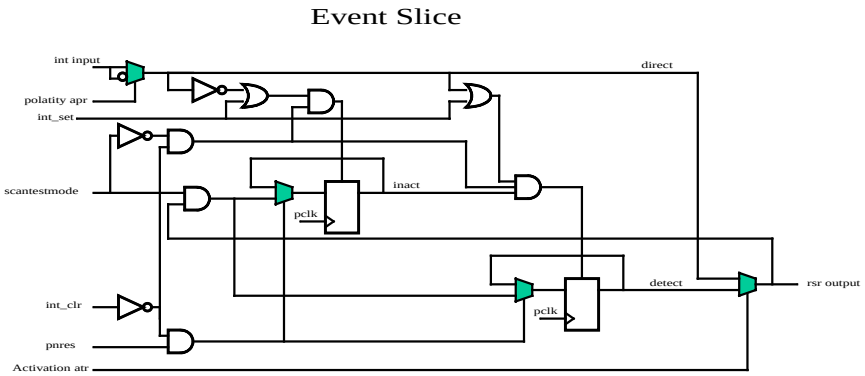


Fig. 25 Event Slice

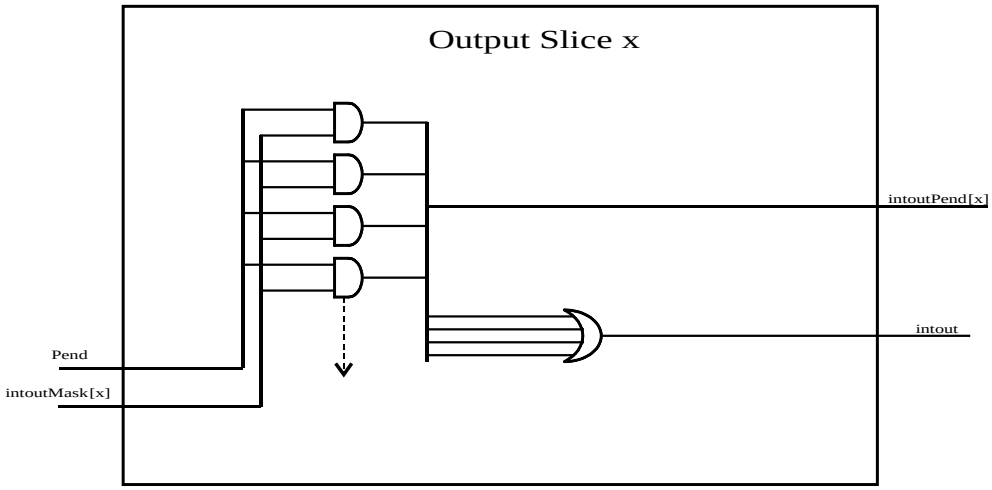


Fig. 26 Output Slice x

For individual outputs, the results can be read on their specific intoutPend status register.

21.4 Registers

The event router maintains the registers shown in Table 111. More detailed descriptions follow. The VPB width is a byte of data for 8 or less interrupt sources, a half word for 9 to 16 interrupt sources, and a word for 17 to 32 interrupt sources.

The bits in these registers relate to the order in which the interrupt events where declared. So the first declared interrupt signal is accessed by bit 0, the next by bit 1 and so on.

n = number of inputs = 105, m = number of outputs = 5, k = integer(n/32) + 1 = 4.

Table 111 Register Definition

OFFSET	REGISTER NAME	FUNCTION	IO
0x000 - 0xBFF	Reserved	Reserved	-
0x0C00	pend[0]	input event pending (masked) status, one bit per input	R
0x0C04	pend[1]	input event pending (masked) status, one bit per input	R
0x0C08	pend[2]	input event pending (masked) status, one bit per input	R
0x0C0C	pend[3]	input event pending (masked) status, one bit per input	R
0x0C20	int_clr[0]	interrupt clear register. write any bit '1' to clear that interrupt latch. (1 bit per input)	W
0x0C24	int_clr[1]	interrupt clear register. write any bit '1' to clear that interrupt latch. (1 bit per input)	W
0x0C28	int_clr[2]	interrupt clear register. write any bit '1' to clear that interrupt latch. (1 bit per input)	W
0x0C2C	int_clr[3]	interrupt clear register. write any bit '1' to clear that interrupt latch. (1 bit per input)	W
0x0C40	int_set[0]	interrupt set register. write any bit '1' to set that interrupt latch. (1 bit per input)	W
0x0C44	int_set[1]	interrupt set register. write any bit '1' to set that interrupt latch. (1 bit per input)	W
0x0C48	int_set[2]	interrupt set register. write any bit '1' to set that interrupt latch. (1 bit per input)	W
0x0C4C	int_set[3]	interrupt set register. write any bit '1' to set that interrupt latch. (1 bit per input)	W
0x0C60	mask[0]	global input event mask, one bit per input 1=enable, 0=disable an input	R/W
0x0C64	mask[1]	global input event mask, one bit per input 1=enable, 0=disable an input	R/W
0x0C68	mask[2]	global input event mask, one bit per input 1=enable, 0=disable an input	R/W
0x0C6C	mask[3]	global input event mask, one bit per input 1=enable, 0=disable an input	R/W
0x0C80	mask_clr[0]	mask clear register. write any bit '1' to clear an input mask. (1 bit per input)	W
0x0C84	mask_clr[1]	mask clear register. write any bit '1' to clear an input mask. (1 bit per input)	W
0x0C88	mask_clr[2]	mask clear register. write any bit '1' to clear an input mask. (1 bit per input)	W
0x0C8C	mask_clr[3]	mask clear register. write any bit '1' to clear an input mask. (1 bit per input)	W
0x0CA0	mask_set[0]	mask set register. write any bit '1' to set an input mask bit. (1 bit per input)	W
0x0CA4	mask_set[1]	mask set register. write any bit '1' to set an input mask bit. (1 bit per input)	W
0x0CA8	mask_set[2]	mask set register. write any bit '1' to set an input mask bit. (1 bit per input)	W

OFFSET	REGISTER NAME	FUNCTION	IO
0x0CAC	mask_set[3]	mask set register. write any bit '1' to set an input mask bit. (1 bit per input)	W
0x0CC0	apr[0]	Activation polarity register	R/W
0x0CC4	apr[1]	Activation polarity register	R/W
0x0CC8	apr[2]	Activation polarity register	R/W
0x0CCC	apr[3]	Activation polarity register	R/W
0x0CE0	atr[0]	Activation type register 1=latched(edge), 0 = direct	R/W
0x0CE4	atr[1]	Activation type register 1=latched(edge), 0 = direct	R/W
0x0CE8	atr[2]	Activation type register 1=latched(edge), 0 = direct	R/W
0x0CEC	atr[3]	Activation type register 1=latched(edge), 0 = direct	R/W
0x0D20	rsr[0]	Raw status of input events (or event latches in latched mode)	R
0x0D24	rsr[1]	Raw status of input events (or event latches in latched mode)	R
0x0D28	rsr[2]	Raw status of input events (or event latches in latched mode)	R
0x0D2C	rsr[3]	Raw status of input events (or event latches in latched mode)	R
0x0D40	intout	status of interrupt output pins (useful for testing)	R
0x0E00	features	indication of template settings to Software	R
0x0FFC	ModuleID	indication of module id and version to Software	R
0x1000	intoutPend[0][0]	An array of status bits, 1 bit per input showing which events are pending for each intout target.[intout][bank]	R
0x1004	intoutPend[0][1]	An array of status bits, 1 bit per input showing which events are pending for each intout target. [intout][bank]	R
0x1008	intoutPend[0][2]	An array of status bits, 1 bit per input showing which events are pending for each intout target. [intout][bank]	R
0x100C	intoutPend[0][3]	An array of status bits, 1 bit per input showing which events are pending for each intout target. [intout][bank]	R
0x1020	intoutPend[1][0]	An array of status bits, 1 bit per input showing which events are pending for each intout target. [intout][bank]	R
0x1024	intoutPend[1][1]	An array of status bits, 1 bit per input showing which events are pending for each intout target. [intout][bank]	R
0x1028	intoutPend[1][2]	An array of status bits, 1 bit per input showing which events are pending for each intout target. [intout][bank]	R
0x102C	intoutPend[1][3]	An array of status bits, 1 bit per input showing which events are pending for each intout target. [intout][bank]	R
0x1040	intoutPend[2][0]	An array of status bits, 1 bit per input showing which events are pending for each intout target. [intout][bank]	R

OFFSET	REGISTER NAME	FUNCTION	IO
0x1044	intoutPend[2][1]	An array of status bits, 1 bit per input showing which events are pending for each intout target. [intout][bank]	R
0x1048	intoutPend[2][2]	An array of status bits, 1 bit per input showing which events are pending for each intout target. [intout][bank]	R
0x104C	intoutPend[2][3]	An array of status bits, 1 bit per input showing which events are pending for each intout target. [intout][bank]	R
0x1060	intoutPend[3][0]	An array of status bits, 1 bit per input showing which events are pending for each intout target. [intout][bank]	R
0x1064	intoutPend[3][1]	An array of status bits, 1 bit per input showing which events are pending for each intout target. [intout][bank]	R
0x1068	intoutPend[3][2]	An array of status bits, 1 bit per input showing which events are pending for each intout target. [intout][bank]	R
0x106C	intoutPend[3][3]	An array of status bits, 1 bit per input showing which events are pending for each intout target. [intout][bank]	R
0x1080	intoutPend[4][0]	An array of status bits, 1 bit per input showing which events are pending for each intout target. [intout][bank]	R
0x1084	intoutPend[4][1]	An array of status bits, 1 bit per input showing which events are pending for each intout target. [intout][bank]	R
0x1088	intoutPend[4][2]	An array of status bits, 1 bit per input showing which events are pending for each intout target. [intout][bank]	R
0x108C	intoutPend[4][3]	An array of status bits, 1 bit per input showing which events are pending for each intout target. [intout][bank]	R
0x1100	intoutPend[5][0]	An array of status bits, 1 bit per input showing which events are pending for each intout target. [intout][bank]	R
0x1104	intoutPend[5][1]	An array of status bits, 1 bit per input showing which events are pending for each intout target. [intout][bank]	R
0x1108	intoutPend[5][2]	An array of status bits, 1 bit per input showing which events are pending for each intout target. [intout][bank]	R
0x110C	intoutPend[5][3]	An array of status bits, 1 bit per input showing which events are pending for each intout target. [intout][bank]	R
0x1400	intoutMask[0][0]	Enable bits for each output, connecting input events to that output. [intout][bank]	R/W
0x1404	intoutMask[0][1]	Enable bits for each output, connecting input events to that output. [intout][bank]	R/W
0x1408	intoutMask[0][2]	Enable bits for each output, connecting input events to that output. [intout][bank]	R/W
0x140C	intoutMask[0][3]	Enable bits for each output, connecting input events to that output. [intout][bank]	R/W
0x1420	intoutMask[1][0]	Enable bits for each output, connecting input events to that output. [intout][bank]	R/W
0x1424	intoutMask[1][1]	Enable bits for each output, connecting input events to that output. [intout][bank]	R/W
0x1428	intoutMask[1][2]	Enable bits for each output, connecting input events to that output. [intout][bank]	R/W
0x142C	intoutMask[1][3]	Enable bits for each output, connecting input events to that output. [intout][bank]	R/W

OFFSET	REGISTER NAME	FUNCTION	IO
0x1440	intoutMask[2][0]	Enable bits for each output, connecting input events to that output. [intout][bank]	R/W
0x1444	intoutMask[2][1]	Enable bits for each output, connecting input events to that output. [intout][bank]	R/W
0x1448	intoutMask[2][2]	Enable bits for each output, connecting input events to that output. [intout][bank]	R/W
0x144C	intoutMask[2][3]	Enable bits for each output, connecting input events to that output. [intout][bank]	R/W
0x1460	intoutMask[3][0]	Enable bits for each output, connecting input events to that output. [intout][bank]	R/W
0x1464	intoutMask[3][1]	Enable bits for each output, connecting input events to that output. [intout][bank]	R/W
0x1468	intoutMask[3][2]	Enable bits for each output, connecting input events to that output. [intout][bank]	R/W
0x146C	intoutMask[3][3]	Enable bits for each output, connecting input events to that output. [intout][bank]	R/W
0x1480	intoutMask[4][0]	Enable bits for each output, connecting input events to that output. [intout][bank]	R/W
0x1484	intoutMask[4][1]	Enable bits for each output, connecting input events to that output. [intout][bank]	R/W
0x1488	intoutMask[4][2]	Enable bits for each output, connecting input events to that output. [intout][bank]	R/W
0x148C	intoutMask[4][3]	Enable bits for each output, connecting input events to that output. [intout][bank]	R/W
0x1800	intoutMaskClr[0][0]	mask clear register for each output. The bit format matches intoutMask. [intout][bank]	W
0x1804	intoutMaskClr[0][1]	mask clear register for each output. The bit format matches intoutMask. [intout][bank]	W
0x1808	intoutMaskClr[0][2]	mask clear register for each output. The bit format matches intoutMask. [intout][bank]	W
0x180C	intoutMaskClr[0][3]	mask clear register for each output. The bit format matches intoutMask. [intout][bank]	W
0x1820	intoutMaskClr[1][0]	mask clear register for each output. The bit format matches intoutMask. [intout][bank]	W
0x1824	intoutMaskClr[1][1]	mask clear register for each output. The bit format matches intoutMask. [intout][bank]	W
0x1828	intoutMaskClr[1][2]	mask clear register for each output. The bit format matches intoutMask. [intout][bank]	W
0x182C	intoutMaskClr[1][3]	mask clear register for each output. The bit format matches intoutMask. [intout][bank]	W
0x1840	intoutMaskClr[2][0]	mask clear register for each output. The bit format matches intoutMask. [intout][bank]	W
0x1844	intoutMaskClr[2][1]	mask clear register for each output. The bit format matches intoutMask. [intout][bank]	W
0x1848	intoutMaskClr[2][2]	mask clear register for each output. The bit format matches intoutMask. [intout][bank]	W

OFFSET	REGISTER NAME	FUNCTION	IO
0x184C	intoutMaskClr[2][3]	mask clear register for each output. The bit format matches intoutMask. [intout][bank]	W
0x1860	intoutMaskClr[3][0]	mask clear register for each output. The bit format matches intoutMask. [intout][bank]	W
0x1864	intoutMaskClr[3][1]	mask clear register for each output. The bit format matches intoutMask. [intout][bank]	W
0x1868	intoutMaskClr[3][2]	mask clear register for each output. The bit format matches intoutMask. [intout][bank]	W
0x186C	intoutMaskClr[3][3]	mask clear register for each output. The bit format matches intoutMask. [intout][bank]	W
0x1880	intoutMaskClr[4][0]	mask clear register for each output. The bit format matches intoutMask. [intout][bank]	W
0x1884	intoutMaskClr[4][1]	mask clear register for each output. The bit format matches intoutMask. [intout][bank]	W
0x1888	intoutMaskClr[4][2]	mask clear register for each output. The bit format matches intoutMask. [intout][bank]	W
0x188C	intoutMaskClr[4][3]	mask clear register for each output. The bit format matches intoutMask. [intout][bank]	W
0x1C00	intoutMaskSet[0][0]	mask set register for each output. The bit format matches intoutMask [intout][bank]	W
0x1C04	intoutMaskSet[0][1]	mask set register for each output. The bit format matches intoutMask [intout][bank]	W
0x1C08	intoutMaskSet[0][2]	mask set register for each output. The bit format matches intoutMask [intout][bank]	W
0x1C0C	intoutMaskSet[0][3]	mask set register for each output. The bit format matches intoutMask [intout][bank]	W
0x1C20	intoutMaskSet[1][0]	mask set register for each output. The bit format matches intoutMask [intout][bank]	W
0x1C24	intoutMaskSet[1][1]	mask set register for each output. The bit format matches intoutMask [intout][bank]	W
0x1C28	intoutMaskSet[1][2]	mask set register for each output. The bit format matches intoutMask [intout][bank]	W
0x1C2C	intoutMaskSet[1][3]	mask set register for each output. The bit format matches intoutMask [intout][bank]	W
0x1C40	intoutMaskSet[2][0]	mask set register for each output. The bit format matches intoutMask [intout][bank]	W
0x1C44	intoutMaskSet[2][1]	mask set register for each output. The bit format matches intoutMask [intout][bank]	W
0x1C48	intoutMaskSet[2][2]	mask set register for each output. The bit format matches intoutMask [intout][bank]	W
0x1C4C	intoutMaskSet[2][3]	mask set register for each output. The bit format matches intoutMask [intout][bank]	W
0x1C60	intoutMaskSet[3][0]	mask set register for each output. The bit format matches intoutMask [intout][bank]	W
0x1C64	intoutMaskSet[3][1]	mask set register for each output. The bit format matches intoutMask [intout][bank]	W

OFFSET	REGISTER NAME	FUNCTION	IO
0x1C68	intoutMaskSet[3][2]	mask set register for each output. The bit format matches intoutMask [intout][bank]	W
0x1C6C	intoutMaskSet[3][3]	mask set register for each output. The bit format matches intoutMask [intout][bank]	W
0x1C80	intoutMaskSet[4][0]	mask set register for each output. The bit format matches intoutMask [intout][bank]	W
0x1C84	intoutMaskSet[4][1]	mask set register for each output. The bit format matches intoutMask [intout][bank]	W
0x1C88	intoutMaskSet[4][2]	mask set register for each output. The bit format matches intoutMask [intout][bank]	W
0x1C8C	intoutMaskSet[4][3]	mask set register for each output. The bit format matches intoutMask [intout][bank]	W

21.4.1 **Pending register**

These registers indicate when an event is active with one bit per input event. If more than 32 inputs are specified the next address is used, so if events 0 -31 are in address then event 32 is bit 0 in address +1 and so on. For multi-thread applications separate addresses are also provided for clearing and setting of latch bits, removing the need for read-modify-write operations.

21.4.2 **Interrupt clear register**

These registers allow latched events to be cleared by writing a ’1’ to any bits corresponding to interrupts to be cleared. The bits are arranged in the same way as the pendng register.

21.4.3 **Interrupt set register**

These registers allow event latches to be set by writing a ’1’ to any bits corresponding to interrupts to be set. This will only have an effect if the event is programmed to latched operation. The bits are arranged in the same way as the pendng register.

21.4.4 **Mask register**

The mask register allows the user to enable or disable input events globally across all outputs. An event which is enabled in this register will cause activation of any outputs which have also been programmed in the intoutMask register. An event which is disabled will not cause activation of any outputs.

21.4.5 **Mask clear register**

These registers allow bits in the mask register to be clear by writing a ’1’ to any bits corresponding to the mask bits to be set. The bits are arranged in the same way as the mask register.

21.4.6 **Mask set register**

These registers allow bits in the mask register to be set by writing a ‘1’ to any bits corresponding to the mask bits to be set. The bits are arranged in the same way as the mask register.

21.4.7 **Activation polarity register apr[k]**

The APR is used to configure which level is the active state for the event sources. A high bit indicates that the event is high sensitive, a low bit that it is low sensitive.

21.4.8 **Activation type register atr[k]**

The ATR is used to configure whether an event signal is used directly or if it is latched. If it is latched, the interrupt will persist after its event source has become inactive until it is cleared by an int_clr write action. The event router slice includes an edge detection circuit which prevents reassertion of an interrupt if the input remains at the active level after the latch is cleared. A high bit written to the ATR selects the latched event as the event source, a low bit uses the event directly.

21.4.9 **Raw status registers rsr[k]**

The Raw Status Register (RSR) shows unmasked events including latched events. A high bit read from the RSR indicates an event is (or has been) generated by the particular event source. A low bit read indicates the device is not generating an event. Level sensitive events are expected to be held and removed by the interrupt source.

21.4.10 **Intout register**

These registers show the current state of the event router outputs. One bit per output (bit 0 = intout0 etc).

21.4.11 **Features register**

Indicates how many inputs and outputs were configured in the template.

n inputs: bits [7:0]

m outputs: bits [21:16]

21.4.12 **Module ID register**

Indicates Module ID and version information about the event router.

Module ID bits [31:16]

Minor Version bits [11:8]

Major version bits [15:12]

VPB address range bits [7:0] = 1 for 8k bytes.

21.4.13 **Intout pending registers IntoutPend[m][k]**

With these registers the user can, for each individual output, enable/disable an input event to be routed to that output.The register/bit arrangement matches that of intoutPend.

21.4.14 **IntoutMask registers intoutMask[m][k]**

With these registers the user can, for each individual output, enable/disable an input event to be routed to that output.The register/bit arrangement matches that of intoutPend.

21.4.15 **Intout Mask Clear registers IntoutMaskClr[m][k]**

Writing a bit to 1 in any of these registers sets the corresponding bit in the corresponding intoutMask register.

21.4.16 **Intout Mask Set registers intoutMaskSet[m][k]**

Writing a bit to 1 in any of these registers clears the corresponding bit in the corresponding intoutMask register.

21.5 Wake-up behaviour

All event sources, which are connected to to the event inputs, can cause an wake-up trigger to the CGU module.

21.6 Timing

This section will show the timing diagrams.
Note that the timing shown is for standard VPB bus transfers.

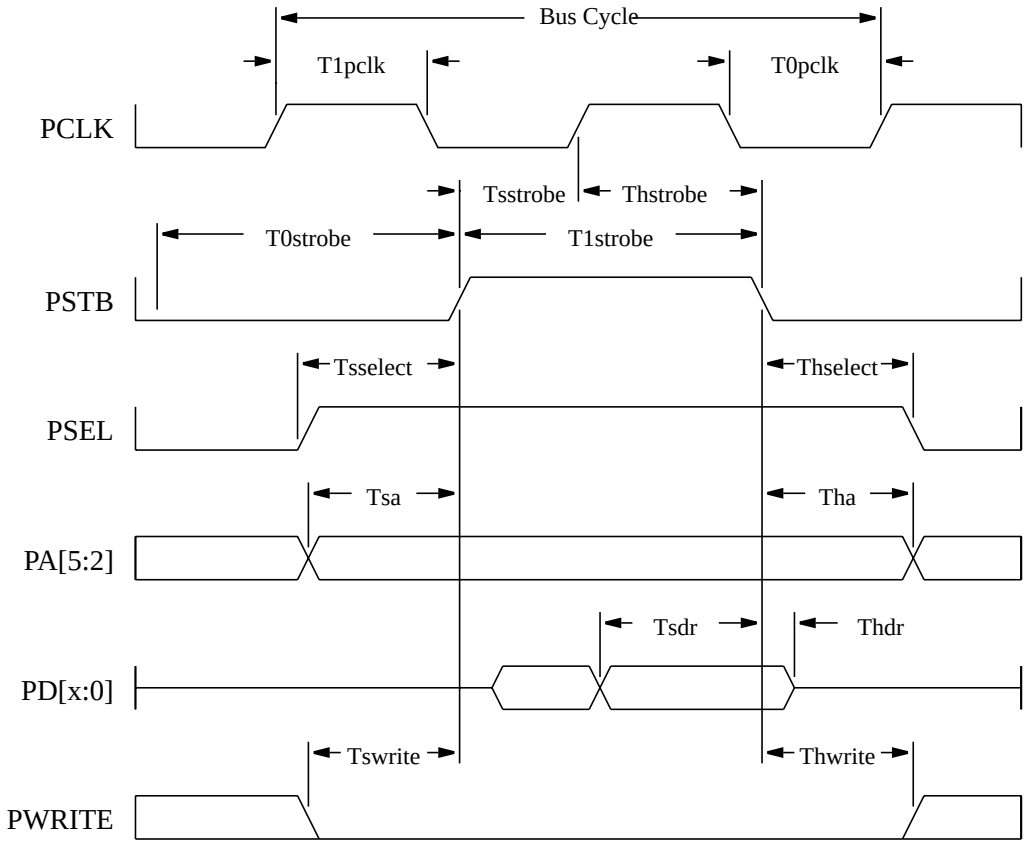


Fig. 27 Waveforms for a read from the event_router block.

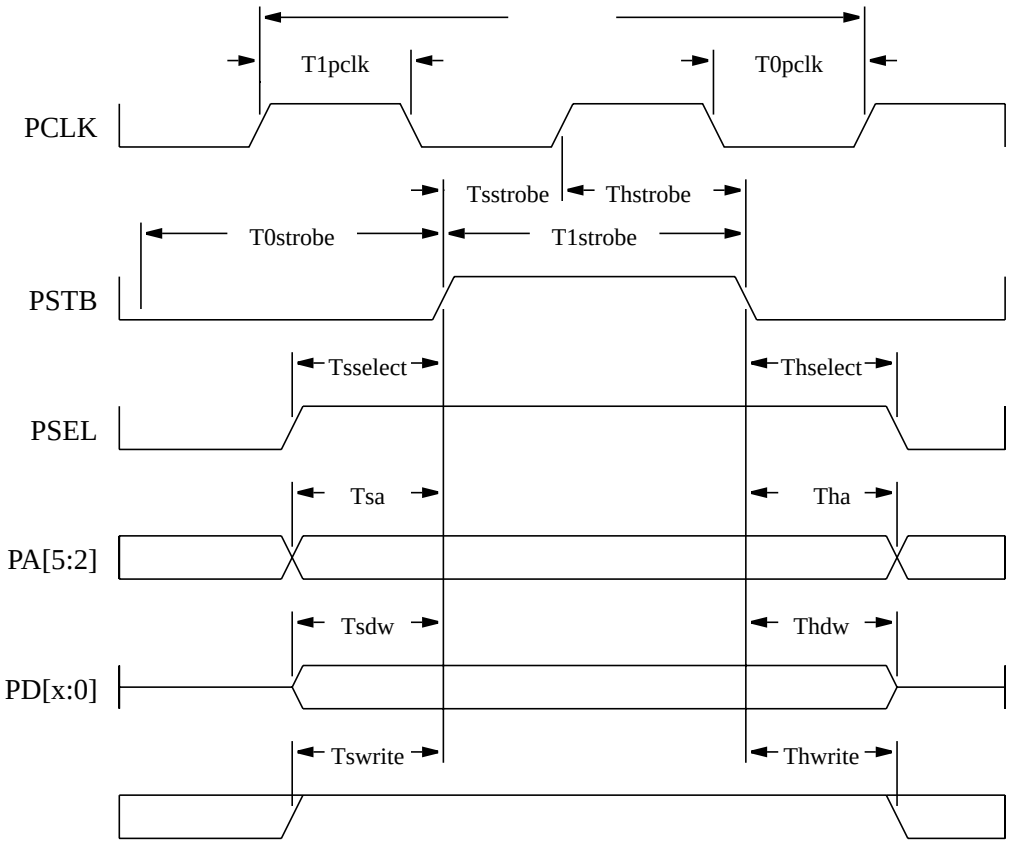


Fig. 28 Waveforms for a write to the event_router block

Table 112 Timing Parameters

PARAMETER NAME	DESCRIPTION
T1pclk	Minimum required high time for PCLK.
T0pclk	Minimum required low time for PCLK.
T1strobe	Minimum required high time for PSTB.
T0strobe	Minimum required high time for PSTB.
Tsstrobe	Minimum required setup time for PSTB.
Thstrobe	Minimum required hold time for PSTB.
Tsselect	Minimum required setup time for PSEL.
Thselect	Minimum required hold time for PSEL.
Tsa	Minimum required setup time for PA[12:2].
Tha	Minimum required hold time for PA[12:2].
Tsdr	Minimum provided setup time for a PD read.
Thdr	Minimum provided hold time for a PD read.
Tsdw	Minimum required setup time for a PD write.
Thdw	Minimum required hold time for a PD write.
Tswrite	Minimum required setup time for PWRITE.
Thwrite	Minimum required hold time for PWRITE.

22 MULTI PURPOSE MEMORY CONTROLLER (MPMC)

22.1 Overview

The multi port memory controller supports the interface to a lot of type of memories: SDRAM, LP SDRAM, Flash, Synchronous micron flash, SRAM, ROM.

Use the free available Technical Reference Manual of the ARM PrimeCell MultiPort Memory Controller of ARM, when you need more information about the MPMC than this chapter describes. You can download this manual at www.arm.com.

22.2 Configuration of MPMC in PNX0101

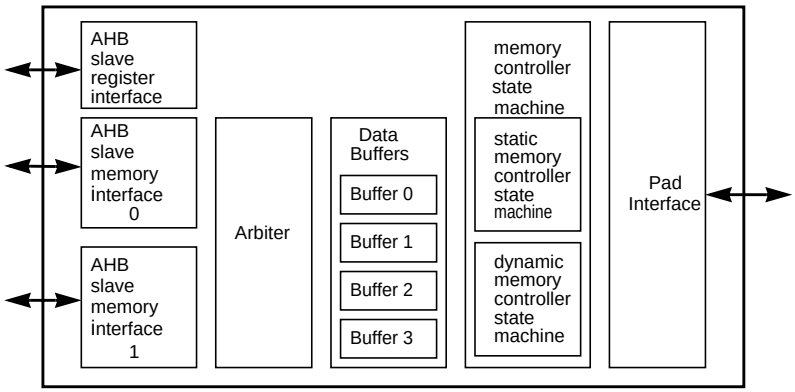


Fig.28 MPMC block diagram

The MPMC is configured as follows in the PNX0101 chip:

- 2 AHB slave memory interfaces are used.
- 3 static memory bank selects are used. A 4th static memory bank select is used intern for the glue logic. This means that only 3 static memory bank selects are going outside the chip. The addresspace of each of these 4 banks is 2 MBytes. (21 AHB address lines are used for addressing the space within a static memory bank.)
- 1 dynamic memory bank select is used. The addresspace of this bank is 64 Mbyte (maximal 512 MBit SDRAM).
- The MPMC doen't contain the Test Interface Controller.
- Data bus width to the pads is 16 bits.
- Addres bus width to the pads is 21 bits.

22.3 Configuration of MPMC in PNX0101

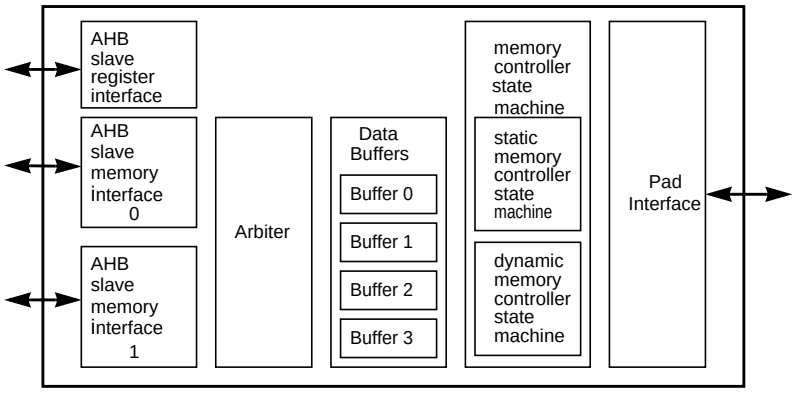


Fig.28 MPMC block diagram

The MPMC is configured as follows in the PNX0101 chip:

- 2 AHB slave memory interfaces are used.
- 3 static memory bank selects are used. A 4th static memory bank select is used intern for the glue logic. This means that only 3 static memory bank selects are going outside the chip. The addresspace of each of these 4 banks is 2 MBytes.
- 1 dynamic memory bank select is used. The addresspace of this bank is 512 MByte.
- The MPMC doen't contain the Test Interface Controller.
- Data bus width to the pads is 16 bits.
- Addres bus width to the pads is 21 bits.

22.4 Configuration of delays

It is possible to delay the following clocks: MPMCFBCLKIN (regarding to MPMCCLK), MPMCCLKDELAY (regarding to MPMCCLK) and MPMCCLKOUT. In the chapter about SYSCREG you wil find the SYSCREG_MPMP_DELAYMODES register which can be used for configuring the delays of these clocks.

22.5 Functional Description

The multiport memory controller block optimizes and controls external memory transactions. The functions of the MPMC blocks are described in this chapter.

22.5.1 AHB slave register interface

The AHB slave register interface block enables the registers of the PrimeCell MPMC to be programmed. This module also contains most of the registers and performs the majority of the register address decoding. This interface must be connected to the ARM processor AHB bus to enable the PrimeCell MPMC to be programmed.

22.5.1.1 Memory transaction endianness and transfer width towards registers

To eliminate the possibility of endianness problems, all data transfers to and from the registers of the PrimeCell MPMC must be 32 bits wide.

--------- **Note** ----------
If an access is attempted with a size other than a word (32 bits), it causes an ERROR response on HRESP and the

transfer is terminated.

22.5.2 AHB SLAVE MEMORY INTERFACES

The AHB slave memory interfaces enable devices to access the external memories. The memory interfaces are prioritized, with interface 0 having the highest priority. Having more than one memory interface enables high-bandwidth peripherals direct access to the PrimeCell MPMC, without data having to pass over the main system bus.

---------- **Note** ----------
All AHB burst types are supported, enabling the most efficient use of memory bandwidth.

The AHB interfaces do not generate SPLIT and RETRY responses.

22.5.2.1 Memory transaction endianness

The endianness of the data transfers to and from the external memories is determined by the Endian mode (N) bit in the MPMCConfig register.
The memory controller must be idle (see the busy field of the MPMCStatus register) before endianness is changed, so that the data is transferred correctly.

22.5.2.2 Memory transaction size

Memory transactions can be 8, 16, or 32 bits wide. Any access attempted with a size greater than a word (32 bits) causes an ERROR response on HRESP and the transfer is terminated.

22.5.2.3 Write protected memory areas

Write transactions to write-protected memory areas generate an ERROR response on HRESP and the transfer is terminated.

22.5.3 Arbiter

The arbiter arbitrates between the AHB slave memory interfaces. AHB interface 0 has the highest access priority, and AHB interface 3 has the lowest priority.

22.5.4 Data buffers

The AHB interfaces use read and write buffers to improve memory bandwidth and reduce transaction latency. The PrimeCell MPMC contains four 16-word buffers. The buffers are not tied to a particular AHB interface, and can be used as read buffers, write buffers, or a combination of both. The buffers are allocated automatically. Because of the way the buffers are designed they are always coherent for reads and writes, and across AHB memory interfaces.

The buffers are enabled on a memory bank basis, using the MPMCDynamicConfig or the MPMCStaticConfig registers.

Write buffers are used to:

- Merge write transactions so that the number of external transactions are minimized.
- Buffer data until the PrimeCell MPMC can complete the write transaction improving AHB write latency.
- Convert all dynamic memory write transactions into quadword bursts on the external memory interface. This enhances transfer efficiency for dynamic memory.
- Reduce external memory traffic. This improves memory bandwidth and reduces power consumption.

Write buffer operation:

- If the buffers are enabled, an AHB write operation writes into the Least Recently Used (LRU) buffer if empty.
- If the LRU buffer is not empty the contents of the buffer are flushed to memory to make space for the AHB write data.
- If a buffer contains write data it is marked as dirty, and its contents are written to memory before the buffer can be reallocated.

The write buffers are flushed whenever:

- the memory controller statemachine is not busy performing accesses to external memory
- the memory controller statemachine is not busy performing accesses to external memory, and a AHB interface is writing to a different buffer.

---------- **Note** ----------
The smallest buffer flush is a quadword of data.

Read buffers are used to:

- Buffer read requests from memory. Future read requests that hit the buffer read the data from the buffer rather than memory reduce transaction latency.
- Convert all read transactions into quadword bursts on the external memory interface. This enhances transfer efficiency for dynamic memory.
- Reduce external memory traffic. This improves memory bandwidth and reduces power consumption.

Read buffer operation:

- If the buffers are enabled, and the read data is contained in one of the buffers the read data is provided directly from the buffer.
- If the read data is not contained in a buffer, the LRU buffer is selected. If the buffer is dirty (contains write data), the write data is flushed to memory. When an empty buffer is available the read command is posted to the memory. While the memory controller is waiting for the data to be returned the memory controller can re-arbitrate to enable additional memory transactions to be processed. When the first data item is returned from memory the read data is provided to the respective AHB port. Other AHB ports can access the data in the buffer when the read transaction has completed.

A buffer filled by performing a read from memory is marked as not-dirty (not containing write data) and its contents are not flushed back to the memory controller unless a subsequent AHB transfer performs a write that hits the buffer.

22.5.5 Memory controller state machine

The memory controller state machine comprises two functional blocks:

- a static memory controller
- a dynamic memory controller

The memory controller state machine holds up to two requests in its internal buffer. It prioritizes and rearranges accesses to maximize memory bandwidth and minimize transaction latency.
For example, if AHB interfaces 3 and 2 simultaneously request a data transfer from dynamic memory, to different memory banks, and the port 2 request address is to a closed page, but port 3 address is for an already open page, the following sequence occurs:

1. The ACT command is sent to open the SDRAM row specified by the AHB interface 2 address.
2. The AHB interface 3 access is completed.
3. The AHB interface 2 access is completed.

The access priority is modified to take into account the ease of getting data to complete each transfer, but the access priority is always biased to the highest priority AHB interface.

22.5.6 Pad interface

The pad interface block provides the interface to the pads. The pad interface uses feedback clocks, MPMCFBCLKIN[3:0], to resynchronize SDRAM read data from the off-chip to on-chip domains.

22.6 MPMC register summery

The external memory is accessed using the AHB memory interface ports. Addresses are not fixed, but are determined by the AHB decoder, and can be different for any particular system implementation. Transfers to the external memories of the PrimeCell MPMC are selected by the HSELMPMC[3:0]CS[7:0] signals.

---------- **Note** ----------
[3:0] indicates the AHB port number, and [7:0] indicates the chip select to be accessed.

The PrimeCell MPMC registers are shown in Table 113 on page 204.

Table 113MPMC register map

REGISTER	Offset	Type	Reset nPOR	Description
MPMCControl	0x000	R/W	0x3	Control register
MPMCStatus	0x004	R	0x5	Status register
MPMCConfig	0x008	R/W	0x000	Configuration register
MPMCDynamicControl	0x020	R/W	0x006	Dynamic memory control register
MPMCDynamicRefresh	0x024	R/W	0x0	Dynamic memory refresh timer
MPMCDynamicReadConfig	0x028	R/W	0x0	Dynamic memory read configuration
MPMCDynamictRP	0x030	R/W	0x0F	Dynamic memory precharge command period (t_{RP})
MPMCDynamictRAS	0x034	R/W	0xF	Dynamic memory active to precharge period (t_{RAS})
MPMCDynamictSREX	0x038	R/W	0xF	Dynamic memory self-refresh exit time(t_{SREX})
MPMCDynamictAPR	0x03C	R/W	0xF	Dynamic memory last data out to active time (t_{SREX})
MPMCDynamictDAL	0x040	R/W	0xF	Dynamic memory data-in to active command time (t_{DAL} or t_{APW})
MPMCDynamictWR	0x044	R/W	0xF	Dynamic memory write recovery time (t_{WR}, t_{DPL}, t_{RWL}, or t_{RDL})
MPMCDynamictRC	0x048	R/W	0x1F	Dynamic memory active to active command period (t_{RC})
MPMCDynamictRFC	0x04C	R/W	0x1F	Dynamic memory auto refresh period, and auto refresh to active command period (t_{RFC})
MPMCDynamictXSR	0x050	R/W	0x1F	Dynamic memory exit self-refresh to active command time (t_{XSR})
MPMCDynamictRRD	0x054	R/W	0xF	Dynamic memory active bank A to active B time (t_{RRD})
MPMCDynamictMRD	0x058	R/W	0xF	Dynamic memory load mode register to active command time (t_{MRD})
MPMCStaticExtendedWait	0x080	R/W	0x0	Static memory extended wait
MPMCDynamicConfig0	0x100	R/W	0x0	Dynamic memory configuration register
MPMCDynamicRasCas0	0x104	R/W	0x303	Dynamic memory RAS and CAS delay
MPMCDynamicConfig1	0x120	R/W	0x0	Dynamic memory configuration register
MPMCDynamicRasCas1	0x124	R/W	0x303	Dynamic memory RAS and CAS delay
MPMCDynamicConfig2	0x140	R/W	0x0	Dynamic memory configuration register
MPMCDynamicRasCas2	0x144	R/W	0x303	Dynamic memory RAS and CAS delay
MPMCDynamicConfig3	0x160	R/W	0x0	Dynamic memory configuration register
MPMCDynamicRasCas3	0x164	R/W	0x303	Dynamic memory RAS and CAS delay
MPMCStaticConfig0	0x200	R/W	0x-0 [a]	Static memory configuration register
MPMCStaticWaitWen0	0x204	R/W	0x0	Static memory write enable delay
MPMCStaticWaitOen0	0x208	R/W	0x0	Static memory output enable delay
MPMCStaticWaitRd0	0x20C	R/W	0x1F	Static memory read delay
MPMCStaticWaitPage0	0x210	R/W	0x1F	Static memory page mode read delay
MPMCStaticWaitWr0	0x214	R/W	0x1F	Static memory write delay
MPMCStaticWaitTurn0	0x218	R/W	0xF	Static memory turn round delay
MPMCStaticConfig1	0x220	R/W	0x--[a]	Static memory configuration register
MPMCStaticWaitWen1	0x224	R/W	0x0	Static memory write enable delay
MPMCStaticWaitOen1	0x228	R/W	0x0	Static memory output enable delay

REGISTER	Offset	Type	Reset nPOR	Description
MPMCStaticWaitRd1	0x22C	R/W	0x1F	Static memory read delay
MPMCStaticWaitPage1	0x230	R/W	0x1F	Static memory page mode read delay
MPMCStaticWaitWr1	0x234	R/W	0x1F	Static memory write delay
MPMCStaticWaitTurn1	0x238	R/W	0xF	Static memory turn round delay
MPMCStaticConfig2	0x240	R/W	0x-0 [a]	Static memory configuration register
MPMCStaticWaitWen2	0x244	R/W	0x0	Static memory write enable delay
MPMCStaticWaitOen2	0x248	R/W	0x0	Static memory output enable delay
MPMCStaticWaitRd2	0x24C	R/W	0x1F	Static memory read delay
MPMCStaticWaitPage2	0x250	R/W	0x1F	Static memory page mode read delay
MPMCStaticWaitWr2	0x254	R/W	0x1F	Static memory write delay
MPMCStaticWaitTurn2	0x258	R/W	0xF	Static memory turn round delay
MPMCStaticConfig3	0x260	R/W	0x-0 [a]	Static memory configuration register
MPMCStaticWaitWen3	0x264	R/W	0x0	Static memory write enable delay
MPMCStaticWaitOen3	0x268	R/W	0x0	Static memory output enable delay
MPMCStaticWaitRd3	0x26C	R/W	0x1F	Static memory read delay
MPMCStaticWaitPage3	0x270	R/W	0x1F	Static memory page mode read delay
MPMCStaticWaitWr3	0x274	R/W	0x1F	Static memory write delay
MPMCStaticWaitTurn3	0x278	R/W	0xF	Static memory turn round delay
MPMCITCR	0xF00	R/W	0x0	Test control register
MPMCITIP	0xF20	R/W	-	Test input register
MPMCITOP	0xF40	R/W	-	Test output register
MPMCPeriphID4	0xFD0	R/W	0x33	Peripheral identification register bits [39:32]
MPMCPeriphID5	0xFD4	R/W	0x0	Reserved for peripheral identification register
MPMCPeriphID6	0xFD8	R/W	0x0	Reserved for peripheral identification register
MPMCPeriphID7	0xFDC	R/W	0x0	Reserved for peripheral identification register
MPMCPeriphID0	0xFE0	R/W	0x72	Peripheral identification register bits [7:0]
MPMCPeriphID1	0xFE4	R/W	0x11	Peripheral identification register bits [15:8]
MPMCPeriphID2	0xFE8	R/W	0x14 [b]	Peripheral identification register bits [23:16]
MPMCPeriphID3	0xFEC	R/W	0x07	Peripheral identification register bits [31:24]
MPMCPCellID0	0xFFO	R/W	0x0D	PrimeCell identification register bits [7:0]
MPMCPCellID1	0xFF4	R/W	0xF0	PrimeCell identification register bits [15:8]
MPMCPCellID2	0xFF8	R/W	0x05	PrimeCell identification register bits [23:16]
MPMCPCellID3	0xFFC	R/W	0xB1	PrimeCell identification register bits [31:24]

[a] Tie off dependent.

[b] Revision dependent

22.7 Register descriptions

The following PrimeCell MPMC registers are described in this section:

22.7.1 MPMCCONTROL REGISTER

The MPMCControl register is a three-bit, read/write register that controls the memory controller operation. The control bits can be altered during normal operation. This register can be accessed with zero wait states.

Table 114 on page 207 shows the bit assignments for the MPMCControl register.

Table 114MPMCControl register

BITS	NAME	TYPE	DESCRIPTION
[31:3]	Reserved		Reserved, read as zero, do not modify.
[2]	Low-power mode (L)	Read/write	Indicates normal, or low-power mode: 0 = normal mode (reset value on nPOR, and HRESETn) 1 = low-power mode. Entering low-power mode reduces memory controller power consumption. Dynamic memory is refreshed as necessary. The memory controller returns to normal functional mode by clearing the low-power mode bit (L), or by AHB, or power-on reset. ---------- **Note** ---------- The external memory cannot be accessed in low-power state. If a memory access is performed an error response is generated. The memory controller AHB register programming port can be accessed normally. The PrimeCell MPMC registers can be programmed in low-power, and, or disabled state. ----------
[1]	Address mirror (M)	Read/write	Indicates normal or reset memory map: 0 = normal memory map 1 = reset memory map. Static memory chip select 1 is mirrored onto chip select 0 and chip select 4 (reset value on nPOR). On power-on reset, chip select 1 is mirrored to both chip select 0 and chip select 1 and chip select 4 memory areas. Clearing the M bit enables chip select 0 and chip select 4 memory to be accessed.
[0]	MPMC Enable (E)	Read/write	Indicates if the PrimeCell MPMC is enabled or disabled: 0 = disabled 1 = enabled (reset value on nPOR and HRESETn). Disabling the PrimeCell MPMC reduces power consumption. When the memory controller is disabled the memory is not refreshed. The memory controller is enabled by setting the enable bit, or by AHB, or power-on reset. ---------- **Note** ---------- The external memory cannot be accessed in disable state. If a memory access is performed an error response is generated. The memory controller AHB register programming port can be accessed normally. The PrimeCell MPMC registers can be programmed in low-power, and, or disabled state. ----------

22.7.2 MPMCSTATUS REGISTER

The three-bit read-only MPMCStatus register provides PrimeCell MPMC status information. This register can be accessed with zero wait states.

Table 115 on page 208 shows the bit assignments for the MPMCStatus register.

Table 115MPMCStatus register

BITS	NAME	TYPE	DESCRIPTION
[31:3]	Reserved		Reserved, read as zero, do not modify.
[2]	Self-refresh acknowledge, MPMCSREFACK (SA)	Read	This read only bit indicates the operating mode of the PrimeCell MPMC: 0 = normal mode 1 = self-refresh mode (reset value on nPOR).
[1]	Write buffer status (S)	Read	This read-only bit enables the PrimeCell MPMC to enter low-power mode or disabled mode cleanly: 0 = write buffers empty (reset value on nPOR) 1 = write buffers contain data.
[0]	Busy (B)	Read	This read-only bit is used to ensure that the memory controller enters the low-power or disabled mode cleanly by determining if the memory controller is busy or not: 0 = MPMC is idle (reset value on HRESETn) 1 = MPMC is busy performing memory transactions, commands, auto-refresh cycles, or is in self-refresh mode (reset value on nPOR and HRESETn).

22.7.3 MPMCCONFIG REGISTER

The two-bit, read/write, MPMCConfig register configures the operation of the memory controller. It is recommended that this register is modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the PrimeCell MPMC is idle, and then entering low-power, or disabled mode. MPMCConfig is accessed with one wait state. shows the bit assignments for the MPMCConfig register.

Table 116 on page 209 shows the bit assignments for the MPMCConfig register.

Table 116MPMCConfig register

BITS	NAME	TYPE	DESCRIPTION
[31:9]			Reserved, read undefined, must be written as zeros.
[8]		Read/write	HCLK:MPMCCLKOUT[3:0] ratio: 0 = 1:1 (reset value on nPOR) 1 = 1:2.
[7:1]			Reserved, read undefined, must be written as zeros.
[0]	Endian mode (N)	Read/write	Endian mode: 0 = little-endian mode 1 = big-endian mode. The value of the endian bit on power-on reset (nPOR) is determined by the MPMCBIGENDIAN signal. This value can be overridden by software. This field is unaffected by the AHB reset (HRESETn). -------- **Note** -------- The value of the MPMCBIGENDIAN signal is reflected in this field. When programmed this register reflects the last value that is written into it. You must flush all data in the PrimeCell MPMC before switching between little-endian and big-endian modes. ----------

22.7.4 MPMCDYNAMICCONTROL REGISTER

The nine-bit, read/write, MPMCDynamicControl register is used to control dynamic memory operation. The control bits can be altered during normal operation. This register can be accessed with zero wait states.

Table 117 on page 210 shows the bit assignments for the MPMCDynamicControl register.

Table 117MPMCDynamicControl register

BITS	NAME	TYPE	DESCRIPTION
[31:16]			Reserved, read undefined, must be written as zeros.
[15]	SyncFlash reset/power down voltage signal, MPMCRPVHHOUT	Read/write	0 = normal voltage (reset value on nPOR) 1 = set MPMCRPVHHOUT high voltage. This output can be used externally in conjunction with the nMPMCRPOUT signal to indicate a high output voltage, see Table 3-6 on page 3-14.
[14]	SyncFlash reset/power down signal, nMPMCRPOUT (nRP)	Read/write	0 = n1 = set nMPMCRPOUT signal HIGH.MPMCRPOUT signal LOW (reset value on nPOR)
[13]	Low-power SDRAM deep-sleep mode (DP)	Read/write	0 = normal operation (reset value on nPOR) 1 = enter deep power down mode.
[12:9]			Reserved, read undefined, must be written as zeros.
[8:7]	SDRAM initialization (I)	Read/write	00 = issue SDRAM NORMAL operation command (reset value on nPOR) 01 = issue SDRAM MODE command 10 = issue SDRAM PALL (precharge all) command 11 = issue SDRAM NOP (no operation) command).
[6]			Reserved, read undefined, must be written as zeros.
[5]	Memory clock control (MMC)	Read/write	0 = MPMCCLKOUT enabled (reset value on nPOR) 1 = MPMCCLKOUT disabled. --------- **Note ----------** Disabling MPMCCLKOUT can be performed if there are no SDRAM memory transactions. When enabled this field can be used in conjunction with the dynamic memory clock control (CS) field. ------------------------------
[4:3]			Reserved, read undefined, must be written as zeros.

BITS	NAME	TYPE	DESCRIPTION
[2]	Self-refresh request, MPMCSREFREQ (SR)	Read/write	0 = normal mode 1 = enter self-refresh mode (reset value on nPOR). By writing 1 to this bit self-refresh can be entered under software control. Writing 0 to this bit returns the MPMC to normal mode. The self-refresh acknowledge bit in the MPMCStatus register must be polled to discover the current operating mode of the MPMC. ------- **Note** --------- The memory controller exits from power-on reset with the self-refresh bit on HIGH. To enter normal functional mode set this bit LOW. Writing to this register with a HIGH places this register into self-refresh mode. This functionality enables data to be stored over SDRAM self-refresh if the ASIC is powered down. ----------
[1]	Dynamic memory clock control (CS)	Read/write	0 = MPMCCLKOUT stops when all SDRAMs are idle and during self-refresh mode. 1 = MPMCCLKOUT runs continuously (reset value on nPOR). When clock control is LOW the output clock MPMCCLKOUT is stopped when there are no SDRAM transactions. The clock is also stopped during self-refresh mode.
[0]	Dynamic memory clock enable (CE)	Read/write	0 = clock enable of idle devices are deasserted to save power (resetvalue on nPOR) 1 = all clock enables are driven HIGH continuously. ------- **Note** -------- Clock enable must be HIGH during SDRAM initialization. ----------

Table 118 on page 211 shows the output voltage settings for different combinations of bits [15:14].
A block external to the PrimeCell MPMC can use the values of the nMPMCRPOUT and MPMCRPVHHOUT signals to generate the required voltage settings for the Micron SyncFlash reset signal. Some systems do not have an 8V output, or do not require the functionality to raise nRP to 8V. In these cases nMPMCRPVHHOUT can be ignored.

Table 118 Output voltage settings

NMPMCRPOUT	MPMCRPVHHOUT	OUTPUT
0	0	0V
0	1	0V
1	0	3V
1	1	8V

22.7.5 MPMCDynamicRefresh register

The 11-bit, read/write, MPMCDynamicRefresh register configures dynamic memory operation. It is recommended that this register is modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the PrimeCell MPMC is idle, and then entering low-power, or disabled mode. However, these control bits can, if necessary, be altered during normal operation. This register is accessed with one wait state.

-------- **Note** ----------
This register is used for all four dynamic memory chip selects. Therefore the worse case value for all of the chip selects must be programmed.

Table 119 on page 212 shows the bit assignments for the MPMCDynamicRefresh register.

Table 119MPMCDynamicRefresh register

BITS	NAME	TYPE	DESCRIPTION
[31:11]			Reserved, read undefined, must be written as zeros
[10:0]	Refresh timer (REFRESH)	Read/write	0x0 = refresh disabled (reset value on nPOR) 0x1 1(x16) = 16 HCLK ticks between SDRAM refresh cycles 0x8 8(x16) = 128 HCLK ticks between SDRAM refresh cycles 0x1 to 0x7FF n(x16) = 16n HCLK ticks between SDRAM refresh cycles

For example, for the refresh period of 16us, and an HCLK frequency of 50MHz, the following value must be programmed into this register:

16 x 10-6 x 50 x 106 /16 = 50 or 0x32

---------- **Note** ----------
The refresh cycles are evenly distributed. However, there might be slight variations when the auto-refresh command is issued depending on the status of the memory controller.

Unlike other SDRAM memory timing parameters the refresh period is programmed in the HCLK domain.

22.7.6 MPMCDynamicReadConfig register

The two-bit, read/write, MPMCDynamicReadConfig register enables you to configure the dynamic memory read strategy. This register must only be modified during system initialization. This register can be accessed with one wait state. See Pad interface methodology on page 10-12 for more information.

-------- **Note** ----------
This register is used for all four dynamic memory chip selects. Therefore the worse case value for all of the chip selects must be programmed.

Table 120 on page 213 shows the bit assignments for the MPMCDynamicReadConfig register.

Table 120MPMCDynamicReadConfig register

BITS	NAME	TYPE	DESCRIPTION
[31:2]			Reserved, read undefined, must be written as zeros
[1:0]	Read data strategy (RD)	Read/write	00 = clock out delayed strategy, using MPMCCLKOUT (command not delayed, clock out delayed). Reset value on nPOR. 01 = command delayed strategy, using MPMCCLKDELAY (command delayed, clock out not delayed). 10 = command delayed strategy plus one clock cycle, using MPMCCLKDELAY (command delayed, clock out not delayed) 11 = command delayed strategy plus two clock cycles, using MPMCCLKDELAY (command delayed, clock out not delayed).

22.7.7 MPMCDYNAMICTRP REGISTER

The four-bit, read/write, MPMCDynamictRP register enables you to program the precharge command period, t_{RP}. This register must only be modified during system initialization. This value is normally found in SDRAM data sheets as t_{RP}. This register can be accessed with one wait state.

-------- **Note** ----------
This register is used for all four dynamic memory chip selects. Therefore the worse case value for all of the chip selects must be programmed.

Table 121 on page 214 shows the bit assignments for the MPMCDynamictRP register.

Table 121MPMCDynamictRP register

BITS	NAME	TYPE	DESCRIPTION
[31:4]			Reserved, read undefined, must be written as zeros
[3:0]	Precharge command period (t_{RP})	Read/write	0x0 to 0xE = n + 1 clock cycles [a] 0xF = 16 clock cycles (reset value on nPOR)

[a] The delay is in MPMCCLK cycles.

22.7.8 MPMCDYNAMICTRAS REGISTER

The four-bit, read/write, MPMCDynamictRAS register enables you to program the active to precharge command period, t_{RAS}. It is recommended that this register is modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the PrimeCell MPMC is idle, and then entering low-power, or disabled mode. This value is normally found in SDRAM data sheets as t_{RAS}. This register can be accessed with one wait state.

-------- **Note** ----------
This register is used for all four dynamic memory chip selects. Therefore the worse case value for all of the chip selects must be programmed.

Table 122 on page 214 shows the bit assignments for the MPMCDynamictRAS register.

Table 122MPMCDynamictRAS register

BITS	NAME	TYPE	DESCRIPTION
[31:4]			Reserved, read undefined, must be written as zeros
[3:0]	Active to precharge command period (t_{RAS})	Read/write	0x0 to 0xE = n + 1 clock cycles [a] 0xF = 16 clock cycles (reset value on nPOR)

[a] The delay is in MPMCCLK cycles.

22.7.9 MPMCDYNAMICTSREX REGISTER

The four-bit, read/write, MPMCDynamictSREX register enables you to program the self-refresh exit time, t_{SREX}. It is recommended that this register is modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the PrimeCell MPMC is idle, and then entering low-power, or disabled mode. This value is normally found in SDRAM data sheets as t_{SREX}, for devices without this parameter you use the same value as t_{XSR}. This register can be accessed with one wait state.

-------- **Note** ----------
This register is used for all four dynamic memory chip selects. Therefore the worse case value for all of the chip selects must be programmed.

Table 123 on page 215 shows the bit assignments for the MPMCDynamictSREX register.

Table 123MPMCDynamictSREX register

BITS	NAME	TYPE	DESCRIPTION
[31:4]			Reserved, read undefined, must be written as zeros
[3:0]	Self-refresh exit time (t_{SREX})	Read/write	0x0 to 0xE = n + 1 clock cycles [a] 0xF = 16 clock cycles (reset value on nPOR)

[a] The delay is in MPMCCLK cycles.

22.7.10 MPMCDYNAMICTAPR REGISTER

The four-bit, read/write, MPMCDynamictAPR register enables you to program the last-data-out to active command time, t_{APR}. It is recommended that this register is modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the PrimeCell MPMC is idle, and then entering low-power, or disabled mode. This value is normally found in SDRAM data sheets as t_{APR}. This register can be accessed with one wait state.

-------- **Note** ----------
This register is used for all four dynamic memory chip selects. Therefore the worse case value for all of the chip selects must be programmed.

Table 124 on page 215 shows the bit assignments for the MPMCDynamictAPR register.

Table 124MPMCDynamictAPR register

BITS	NAME	TYPE	DESCRIPTION
[31:4]			Reserved, read undefined, must be written as zeros
[3:0]	Last-data-out to active command time (t_{APR})	Read/write	0x0 to 0xE = n + 1 clock cycles [a] 0xF = 16 clock cycles (reset value on nPOR)

[a] The delay is in MPMCCLK cycles.

22.7.11 MPMCDYNAMICTDAL REGISTER

The four-bit, read/write, MPMCDynamictDAL register enables you to program the data-in to active command time, t_{DAL}. It is recommended that this register is modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the PrimeCell MPMC is idle, and then entering low-power, or disabled mode. This value is normally found in SDRAM data sheets as t_{DAL}, or t_{APW}. This register can be accessed with one wait state.

-------- **Note** ----------
This register is used for all four dynamic memory chip selects. Therefore the worse case value for all of the chip selects must be programmed.

Table 125 on page 216 shows the bit assignments for the MPMCDynamictDAL register.

Table 125MPMCDynamictDAL register

BITS	NAME	TYPE	DESCRIPTION
[31:4]			Reserved, read undefined, must be written as zeros
[3:0]	Data-in to active command (t_{DAL})	Read/write	0x0 to 0xE = n clock cycles [a] 0xF = 15 clock cycles (reset value on nPOR)

[a] The delay is in MPMCCLK cycles.

22.7.12 MPMCDYNAMICTWR REGISTER

The four-bit, read/write, MPMCDynamictWR register enables you to program the write recovery time, t_{WR}. It is recommended that this register is modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the PrimeCell MPMC is idle, and then entering low-power, or disabled mode. This value is normally found in SDRAM data sheets as tWR, $t_{DP}L$, t_{RWL}, or t_{RDL}. This register can be accessed with one wait state.

-------- **Note** ----------
This register is used for all four dynamic memory chip selects. Therefore the worse case value for all of the chip selects must be programmed.

Table 126 on page 216 shows the bit assignments for the MPMCDynamictWR register.

Table 126MPMCDynamictWR register

BITS	NAME	TYPE	DESCRIPTION
[31:4]			Reserved, read undefined, must be written as zeros
[3:0]	Write recovery time (t_{WR})	Read/write	0x0 to 0xE = n + 1 clock cycles[a] 0xF = 16 clock cycles (reset value on nPOR)

[a] The delay is in MPMCCLK cycles.

22.7.13 MPMCDYNAMICTRC REGISTER

The five-bit, read/write, MPMCDynamictRC register enables you to program the active to active command period, t_{RC}. It is recommended that this register is modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the PrimeCell MPMC is idle, and then entering low-power, or disabled mode. This value is normally found in SDRAM data sheets as t_{RC}. This register can be accessed with one wait state.

-------- **Note** ----------
This register is used for all four dynamic memory chip selects. Therefore the worse case value for all of the chip selects must be programmed.

Table 127 on page 217 shows the bit assignments for the MPMCDynamictRC register.

Table 127MPMCDynamictRC register

BITS	NAME	TYPE	DESCRIPTION
[31:4]			Reserved, read undefined, must be written as zeros
[3:0]	Active to active command period (t_{WR})	Read/write	0x0 to 0x1E = n + 1 clock cycles[a] 0x1F = 32 clock cycles (reset value on nPOR)

[a] The delay is in MPMCCLK cycles.

22.7.14 MPMCDYNAMICTRFC REGISTER

The five-bit, read/write, MPMCDynamictRFC register enables you to program the auto-refresh period, and auto-refresh to active command period, t_{RFC}. It is recommended that this register is modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the PrimeCell MPMC is idle, and then entering low-power, or disabled mode. This value is normally found in SDRAM data sheets as t_{RFC}, or sometimes as t_{RC}. This register can be accessed with one wait state.

-------- **Note** ----------
This register is used for all four dynamic memory chip selects. Therefore the worse case value for all of the chip selects must be programmed.

Table 128 on page 217 shows the bit assignments for the MPMCDynamictRFC register.

Table 128MPMCDynamictRFC register

BITS	NAME	TYPE	DESCRIPTION
[31:5]			Reserved, read undefined, must be written as zeros
[4:0]	Auto-refresh period and auto-refresh to active command period (t_{RFC})	Read/write	0x0 to 0x1E = n + 1 clock cycles[a] 0x1F = 32 clock cycles (reset value on nPOR)

[a] The delay is in MPMCCLK cycles.

22.7.15 MPMCDYNAMICTXSR REGISTER

The five-bit, read/write, MPMCDynamictXSR register enables you to program the exit self-refresh to active command time, t_{XSR}. It is recommended that this register is modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the PrimeCell MPMC is idle, and then entering low-power, or disabled mode. This value is normally found in SDRAM data sheets as t_{XSR}. This register can be accessed with one wait state.

-------- **Note** ----------
This register is used for all four dynamic memory chip selects. Therefore the worse case value for all of the chip selects must be programmed.

Table 129 on page 218 shows the bit assignments for the MPMCDynamictXSR register.

Table 129MPMCDynamictXSR register

BITS	NAME	TYPE	DESCRIPTION
[31:5]			Reserved, read undefined, must be written as zeros
[4:0]	Exit self-refresh to active command time (t_{XSR})	Read/write	0x0 to 0x1E = n + 1 clock cycles[a] 0x1F = 32 clock cycles (reset value on nPOR)

[a] The delay is in MPMCCLK cycles.

22.7.16 MPMCDYNAMICTRRD REGISTER

The four-bit, read/write, MPMCDynamictRRD register enables you to program the active bank A to active bank B latency, t_{RRD}. It is recommended that this register is modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the PrimeCell MPMC is idle, and then entering low-power, or disabled mode. This value is normally found in SDRAM data sheets as t_{RRD}. This register can be accessed with one wait state.

-------- **Note** ----------
This register is used for all four dynamic memory chip selects. Therefore the worse case value for all of the chip selects must be programmed.

Table 130 on page 218 shows the bit assignments for the MPMCDynamictRRD register.

Table 130MPMCDynamictRRD register

BITS	NAME	TYPE	DESCRIPTION
[31:4]			Reserved, read undefined, must be written as zeros
[3:0]	Active bank A to active bank Read/write B latency (t_{RRD})	Read/write	0x0 to 0xE = n + 1 clock cycles[a] 0xF = 16 clock cycles (reset value on nPOR)

[a] The delay is in MPMCCLK cycles.

22.7.17 MPMCDynamictMRD register

The four-bit, read/write, MPMCDynamictMRD register enables you to program the load mode register to active command time, t_{MRD}. It is recommended that this register is modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the PrimeCell MPMC is idle, and then entering low-power, or disabled mode. This value is normally found in SDRAM data sheets as tMRD, or t_{RSA}. This register can be accessed with one wait state.

-------- **Note** ----------
This register is used for all four dynamic memory chip selects. Therefore the worse case value for all of the chip selects must be programmed.

Table 131 on page 219 shows the bit assignments for the MPMCDynamictMRD register.

Table 131MPMCDynamictMRD register

BITS	NAME	TYPE	DESCRIPTION
[31:4]			Reserved, read undefined, must be written as zeros
[3:0]	Load mode register to active command time (t_{MRD})	Read/write	0x0 to 0xE = n + 1 clock cycles[a] 0xF = 16 clock cycles (reset value on nPOR)

[a] The delay is in MPMCCLK cycles.

22.7.18 MPMCStaticExtendedWait register

The 10-bit, read/write, MPMCStaticExtendedWait register is used to time long static memory read and write transfers (which are longer than can be supported by the MPMCStaticWaitRd[n] or, MPMCStaticWaitWr[n] registers) when the EW bit of the MPMCStaticConfig registers is enabled. There is only a single MPMCStaticExtendedWait register. This is used by the relevant static memory chip select if the appropriate ExtendedWait (EW) bit in the MPMCStaticConfig register is set. It is recommended that this register is modified during system initialization, or when there are no current or outstanding transactions. However, if necessary, these control bits can be altered during normal operation. This register can be accessed with one wait state.

Table 132 on page 219 shows the bit assignments for the MPMCStaticExtendedWait register.

Table 132MPMCStaticExtendedWait register

BITS	NAME	TYPE	DESCRIPTION
[31:10]			Reserved, read undefined, must be written as zeros
[9:0]	External wait time out (EXTENDEDWAIT)	Read/write	0x0 = 16 clock cycles a (reset value on nPOR) 0x1 to 0x3FF = (n+1) x16 clock cycles

[a] The delay is in HCLK cycles.

For example, for a static memory read/write transfer time of 16s, and an MPMCCLK frequency of 50MHz, the following value must be programmed into this register:

(16 x 10-6 x 50 x 106 /16) - 1= 49 or 0x31

22.7.19 MPMCDYNAMICCONFIG0-3 REGISTERS

The 11-bit, read/write, MPMCDynamicConfig0-3 registers enable you to program the configuration information for the relevant dynamic memory chip select. These registers is normally only modified during system initialization. These registers can be accessed with one wait state.

Table 133 on page 220 shows the bit assignments for the MPMCDynamicConfig0-3 registers.

Table 133MPMCDynamicConfig0-3 registers

BITS	NAME	TYPE	DESCRIPTION
[31:21]			Reserved, read undefined, must be written as zeros.
[20]	Write protect (P)	Read/write	0 = writes not protected (reset value on nPOR) 1 = write protected.
[19]	Buffer enable (B)	Read/write	0 = buffer disabled for accesses to this chip select (reset value on nPOR) 1 = buffer enabled for accesses to this chip select. --------- **Note** ---------- The buffers must be disabled during SDRAM and SyncFlash initialization. They must also be disabled when performing SyncFlash commands. The buffers must be enabled during normal operation. ----------
[18:15]			Reserved, read undefined, must be written as zeros.
[14]	Address mapping (AM)	Read/write	See Table 134 on page 221. 0 = reset value on nPOR.
[13]			Reserved, read undefined, must be written as zeros.
[12:7]	Address mapping (AM)	Read/write	See Table 134 on page 221. 00000000 = reset value on nPOR. --------- **Note** ---------- The SDRAM column and row width and number of banks are computed automatically from the address mapping. ----------
[6:5]			Reserved, read undefined, must be written as zeros.
[4:3]	Memory device (MD)	Read/write	00 = SDRAM (reset value on nPOR) 01 = low-power SDRAM 10 = Micron SyncFlash 11 = reserved.
[2:0]			Reserved, read undefined, must be written as zeros.

Address mappings that are not shown in Table 134 on page 221 are reserved.

Table 134 Address mapping

[14]	[12]	[11:9]	[8:7]	DESCRIPTION
16-bit external bus high-performance address mapping (Row, Bank, Column)				
0	0	000	00	16Mb (2Mx8), 2 banks, row length = 11, column length = 9
0	0	000	01	16Mb (1Mx16), 2 banks, row length = 11, column length = 8
0	0	001	00	64Mb (8Mx8), 4 banks, row length = 12, column length = 9
0	0	001	01	64Mb (4Mx16), 4 banks, row length = 12, column length = 8
0	0	010	00	128Mb (16Mx8), 4 banks, row length = 12, column length = 10
0	0	010	01	128Mb (8Mx16), 4 banks, row length = 12, column length = 9
0	0	011	00	256Mb (32Mx8), 4 banks, row length = 13, column length = 10
0	0	011	01	256Mb (16Mx16), 4 banks, row length = 13, column length = 9
0	0	100	00	512Mb (64Mx8), 4 banks, row length = 13, column length = 11
0	0	100	01	512Mb (32Mx16), 4 banks, row length = 13, column length = 10
16-bit external bus low-power SDRAM address mapping (Bank, Row, Column)				
0	1	000	00	16Mb (2Mx8), 2 banks, row length = 11, column length = 9
0	1	000	01	16Mb (1Mx16), 2 banks, row length = 11, column length = 8
0	1	001	00	64Mb (8Mx8), 4 banks, row length = 12, column length = 9
0	1	001	01	64Mb (4Mx16), 4 banks, row length = 12, column length = 8
0	1	010	00	128Mb (16Mx8), 4 banks, row length = 12, column length = 10
0	1	010	01	128Mb (8Mx16), 4 banks, row length = 12, column length = 9
0	1	011	00	256Mb (32Mx8), 4 banks, row length = 13, column length = 10
0	1	011	01	256Mb (16Mx16), 4 banks, row length = 13, column length = 9
0	1	100	00	512Mb (64Mx8), 4 banks, row length = 13, column length = 11
0	1	100	01	512Mb (32Mx16), 4 banks, row length = 13, column length = 10

A chip select can be connected to a single memory device, in this case the chip select data bus width is the same as the device width. Alternatively the chip select can be connected to a number of external devices. In this case the chip select data bus width is the sum of the memory device databus widths.

For example, for a chip select connected to:

- a 32-bit wide memory device, choose a 32-bit wide address mapping
- a 16-bit wide memory device, choose a 16-bit wide address mapping
- four x 8-bit wide memory devices, choose a 32-bit wide address mapping
- two x 8-bit wide memory devices, choose a 16-bit wide address mapping.

22.7.20 MPMCDynamicRasCas0-3 registers

The four-bit, read/write, MPMCDynamicRasCas0-3 registers enable you to program the RAS and CAS latencies for the relevant dynamic memory. It is recommended that these registers are modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the PrimeCell MPMC is idle, and then entering low-power, or disabled mode.

The MPMCDynamicRasCas0-3 registers are accessed with one wait state. Table 135 on page 222 shows the bit assignments for the MPMCDynamicRasCas0-3 registers.

-------- **Note** ----------
The values programmed into these registers must be consistent with the values used to initialize the SDRAM memory device.

Table 135MPMCDynamicRasCas0-3 registers

BITS	NAME	TYPE	DESCRIPTION
[31:10]			Reserved, read undefined, must be written as zeros
[9:8]	CAS latency (CAS)	Read/write	00 = reserved 01 = one clock cycle [a] 10 = two clock cycles 11 = three clock cycles (reset value on nPOR)
[7:2]			Reserved, read undefined, must be written as zeros
[1:0]	RAS latency (active to read/write delay) (RAS)	Read/write	00 = reserved 01 = one clock cycle [a] 10 = two clock cycles 11 = three clock cycles (reset value on nPOR)

[a] The RAS to CAS latency (RAS) and CAS latency (CAS) are both defined in MPMCCLK cycles.

22.7.21 MPMCSTATICCONFIG0-3 REGISTERS

The eight-bit, read/write, MPMCStaticConfig0-3 registers are used to configure the static memory configuration. It is recommended that these registers are modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the PrimeCell MPMC is idle, and then entering low-power, or disabled mode.

The MPMCStaticConfig0-3 registers are accessed with one wait state.

Table 136 on page 224 shows the bit assignments for the MPMCStaticConfig0-3 registers.

Table 136MPMCStaticConfig0-3 registers

BITS	NAME	TYPE	DESCRIPTION
[31:21]			Reserved, read undefined, must be written as zeros.
[20]	Write protect (P)	Read/write	0 = writes not protected (reset value on nPOR) 1 = write protected.
[19]	Buffer enable (B)	Read/write	0 = write buffer disabled (reset value on nPOR) 1 = write buffer enabled.
[18:9]			Reserved, read undefined, must be written as zeros.
[8]	Extended wait (EW)	Read/write	0 = Extended wait disabled (reset value on nPOR) 1 = Extended wait enabled. Extended wait (EW) uses the MPMCStaticExtendedWait register to time both the read and write transfers rather than the MPMCStaticWaitRd and MPMCStaticWaitWr registers. This enables much longer transactions. ---------- **Note** --------- Extended wait and page mode cannot be selected simultaneously. ---
[7]	Byte lane state (PB)	Read/write	0 = For reads all the bits in nMPMCBLSOUT[3:0] are HIGH. For writes the respective active bits in nMPMCBLSOUT[3:0] are LOW (reset value for chip select 0, 2, and 3 on nPOR). 1 = For reads the respective active bits in nMPMCBLSOUT[3:0] are LOW. For writes the respective active bits in nMPMCBLSOUT[3:0] are LOW. The value of the chip select 1 byte lane state field on power-on reset (nPOR) is determined by the MPMCSTCS1PB signal. This value can be overridden by software. This field is unaffected by AHB reset (HRESETn). The byte lane state bit, PB, enables different types of memory to be connected. For byte-wide static memories the nMPMCBLSOUT[3:0] signal from the PrimeCell MPMC is usually connected to nWE (write enable). In this case for reads all the nMPMCBLSOUT[3:0] bits must be HIGH. This means that the byte lane state (PB) bit must be LOW. 16-bit wide static memory devices usually have the nMPMCBLSOUT[3:0] signals connected to the nUB and nLB (upper byte and lower byte) signals in the static memory. In this case a write to a particular byte must assert LOW the appropriate nUB or nLB signal. For reads, all the nUB and nLB signals must be asserted LOW so that the bus is driven. In this case the byte lane state (PB) bit must be HIGH. ---------- **Note** ---------- For chip select 1 the value of the MPMCSTCS1PB signal is reflected in this field. When programmed this register reflects the last value that is written into it. ---

BITS	NAME	TYPE	DESCRIPTION
[6]	Chip select polarity (PC)	Read/write	0 = active LOW chip select 1 = active HIGH chip select. The value of the chip select polarity on power-on reset (nPOR) is determined by the relevant MPMCSTCSxPOL signal. This value can be overridden by software. This field is unaffected by AHB reset (HRESETn). ---------- **Note** ---------- The value of the relevant MPMCSTCSxPOL signal is reflected in this field. When programmed this register reflects the last value that is written into it. ----------
[5:4]			Reserved, read undefined, must be written as zeros.
[3]	Page mode (PM)	Read/write	0 = disabled (reset value on nPOR) 1 = async page mode enabled (page length four). In page mode the PrimeCell MPMC can burst up to four external accesses. Therefore devices with asynchronous page mode burst four or higher devices are supported. Asynchronous page mode burst two devices are not supported and must be accessed normally.
[2]			Reserved, read undefined, must be written as zeros.
[1:0]	Memory width (MW)	Read/write	00 = 8-bit (reset value for chip select 0, 2, and 3 on nPOR). 01 = 16-bit 10 = 32-bit 11 = reserved. The value of the chip select 1 memory width field on power-on reset (nPOR) is determined by the MPMCSTCS1MW[1:0] signal. This value can be overridden by software. This field is unaffected by AHB reset (HRESETn). ---------- **Note** ---------- For chip select 1 the value of the MPMCSTCS1MW[1:0] signal is reflected in this field. When programmed this register reflects the last value that is written into it. ----------

---------- **Note** ----------
Synchronous burst mode memory devices are not supported.

22.7.22 MPMCSTATICWAITWEN0-3 REGISTERS

The four-bit, read/write, MPMCStaticWaitWen0-3 registers enable you to program the delay from the chip select to the write enable. It is recommended that these registers are modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the PrimeCell MPMC is idle, and then entering low-power, or disabled mode.
The MPMCStaticWaitWen0-3 registers are accessed with one wait state.

Table 137 on page 226 shows the bit assignments for the MPMCStaticWaitWen0-3 registers.

Table 137MPMCStaticWaitWen0-3 registers

BITS	NAME	TYPE	DESCRIPTION
[31:4]			Reserved, read undefined, must be written as zeros.
[3:0]	Wait write enable (WAITWEN)	Read/write	Delay from chip select assertion to write enable. 0000 = one HCLK cycle delay between assertion of chip select and write enable (reset value on nPOR) 0001 to 1111 = (n + 1) HCLK cycle delay [a] .

[a] The delay is (WAITWEN +1) x t_{HCLK}.

22.7.23 MPMCSTATICWAITOEN0-3 REGISTERS

The four-bit, read/write, MPMCStaticWaitOen0-3 registers enable you to program the delay from the chip select or address change, whichever is later, to the output enable. It is recommended that these registers are modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the PrimeCell MPMC is idle, and then entering low-power, or disabled mode.
The MPMCStaticWaitOen0-3 registers are accessed with one wait state.
Table 138 on page 226 shows the bit assignments for the MPMCStaticWaitOen0-3 registers.

Table 138MPMCStaticWaitOen0-3 registers

BITS	NAME	TYPE	DESCRIPTION
[31:4]			Reserved, read undefined, must be written as zeros.
[3:0]	Wait output enable (WAITOEN)	Read/write	Delay from chip select assertion to output enable. 0000 = No delay (reset value on nPOR) 0001 to 1111 = n cycle delay [a] .

[a] The delay is WAITOEN x t_{HCLK}.

22.7.24 MPMCStaticWaitRd0-3 registers

The five-bit, read/write, MPMCStaticWaitRd0-3 registers enable you to program the delay from the chip select to the read access. It is recommended that these registers are modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the PrimeCell MPMC is idle, and then entering low-power, or disabled mode. It is not used if the extended wait bit is enabled in the MPMCStaticConfig0-3 registers.
The MPMCStaticWaitRd0-3 registers are accessed with one wait state.
Table 139 on page 227 shows the bit assignments for the MPMCStaticWaitRd0-3 registers.

Table 139MPMCStaticWaitRd0-3 registers

BITS	NAME	TYPE	DESCRIPTION
[31:5]			Reserved, read undefined, must be written as zeros.
[4:0]	Nonpage mode read wait states or asynchronous page mode read first access wait state (WAITRD)	Read/write	Nonpage mode read or asynchronous page mode read, first read only:00000 to 11110 = (n + 1) HCLK cycles for read accesses [a] 11111 = 32 HCLK cycles for read accesses (reset value on nPOR).

[a] For nonsequential reads, the wait state time is (WAITRD + 1) x t_{HCLK}.

22.7.25 MPMCStaticWaitPage0-3 registers

The five-bit, read/write, MPMCStaticWaitPage0-3 registers enable you to program the delay for asynchronous page mode sequential accesses. It is recommended that these registers are modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the PrimeCell MPMC is idle, and then entering low-power, or disabled mode.
MPMCStaticWaitPage0-3 is accessed with one wait state.
Table 140 on page 227 shows the bit assignments for the MPMCStaticWaitPage0-3 registers.

Table 140MPMCStaticWaitPage0-3 registers

BITS	NAME	TYPE	DESCRIPTION
[31:5]			Reserved, read undefined, must be written as zeros.
[4:0]	Asynchronous page mode read after the first read wait states (WAITPAGE)	Read/write	Number of wait states for asynchronous page mode read accesses after the first read: 00000 to 11110 = (n+ 1) HCLK cycle read access time [a] 11111 = 32 HCLK cycle read access time (reset value on nPOR).

[a.] For asynchronous page mode read for sequential reads, the wait state time for page mode accesses after the first read is (WAITRD + 1) x t_{HCLK}.

22.7.26 MPMCSTATICWAITWR0-3 REGISTERS

The five-bit, read/write, MPMCStaticWaitWr0-3 registers enable you to program the delay from the chip select to the write access. It is recommended that these registers are modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the PrimeCell MPMC is idle, and then entering low-power, or disabled mode.
These registers are not used if the extended wait (EW) bit is enabled in the MPMCStaticConfig register.
The MPMCStaticWaitWr0-3 registers are accessed with one wait state.
Table 141 on page 228 shows the bit assignments for the MPMCStaticWaitWr0-3 registers.

Table 141MPMCStaticWaitWr0-3 registers

BITS	NAME	TYPE	DESCRIPTION
[31:5]			Reserved, read undefined, must be written as zeros.
[4:0]	Write wait states (WAITWR)	Read/write	SRAM wait state time for write accesses after the first read: 00000 to 11110 = (n + 2) HCLK cycle write access time [a] 11111 = 33 HCLK cycle write access time (reset value on nPOR).

[a] For asynchronous page mode read for sequential reads, the wait state time for page mode accesses after the first read is (WAITRD + 1) x t_{HCLK}.

22.7.27 MPMCSTATICWAITTURN0-3 REGISTERS

The four-bit, read/write, MPMCStaticWaitTurn0-3 registers enable you to program the number of bus turnaround cycles. It is recommended that these registers are modified during system initialization, or when there are no current or outstanding transactions. This can be ensured by waiting until the PrimeCell MPMC is idle, and then entering low-power, or disabled mode.
The MPMCStaticWaitTurn0-3 registers are accessed with one wait state.
Table 142 on page 228 shows the bit assignments for the MPMCStaticWaitTurn0-3 registers.

Table 142MPMCStaticWaitTurn0-3 registers

BITS	NAME	TYPE	DESCRIPTION
[31:4]			Reserved, read undefined, must be written as zeros.
[3:0]	Bus turnaround cycles (WAITTURN)	Read/write	0000 to 1110 = (n + 1) HCLK turnaround cycles [a] 1111 = 16 HCLK turnaround cycles (reset value on nPOR).

[a] Bus turnaround time is (WAITTURN+ 1) x t_{HCLK}.

To prevent bus contention on the external memory databus, the WAITTURN field controls the number of bus turnaround cycles added between static memory read and write accesses.
The WAITTURN field also controls the number of turnaround cycles between static memory and dynamic memory accesses.

22.7.28 MPMCPeriphID4-7 registers

The MPMCPeriphID4-7 registers are four eight-bit read-only registers, that span address locations 0xFD0-0xFDC. The registers can conceptually be treated as a single register that holds a 32-bit additional peripheral ID value.
Table 143 on page 229 shows the bit assignments for the conceptual 32-bit MPMC Additional Peripheral ID register.

Table 143 Conceptual MPMC Additional Peripheral ID register

BITS	NAME	TYPE	DESCRIPTION
[31:8]	Reserved		Reserved, read undefined, must be written as zeros.
[7:0]	Configuration1	Read	Additional peripheral configuration information

The configuration options are peripheral-specific. For MPMC, the four, eight-bit peripheral identification registers are described in the following subsections:

- MPMCPeriphID4 register on page 229
- MPMCPeriphID5-7 registers on page 229.

22.7.28.1 MPMCPeriphID4 register

The MPMCPeriphID4 register is hard-coded and the fields within the register determine the reset value. Table 144 on page 229 shows the bit assignments for the MPMCPeriphID4 register.

Table 144MPMCPeriphID4 register

BITS	NAME	TYPE	DESCRIPTION
[31:8]		Read	Reserved, read undefined must be written as zeros.
[7:4]	Configuration	Read	Number of buffers: 0000 to 1111 = n +1 buffers 0011 = 4 buffers (value for PL172).
[3:0]	Configuration	Read	Number of AHB memory ports: 0000 to 1111 = n +1 memory ports 0011 = 4 memory ports (value for PL172).

22.7.28.2 MPMCPeriphID5-7 registers

The MPMCPeriphID5-7 registers are reserved.
Table 145 on page 229 shows the bit assignments for the MPMCPeriphID5-7 registers.

Table 145MPMCPeriphID5-7 registers

BITS	NAME	TYPE	DESCRIPTION
[31:0]		Read	Reserved, read undefined, must be written as zeros

22.7.29 MPMCPERIPHID0-3 REGISTERS

The MPMCPeriphID0-3 registers are four eight-bit read-only registers, that span address locations 0xFE0-0xFEC. The registers can conceptually be treated as a single register that holds a 32-bit peripheral ID value.
Table 146 on page 230 shows the bit assignments for the conceptual 32-bit MPMC Peripheral ID register.

Table 146 Conceptual MPMC Peripheral ID register

BITS	NAME	TYPE	DESCRIPTION
[31:24]	Configuration	Read	Configuration options are peripheral-specific. See MPMCPeriphID3 register on page 3-39.
[23:20]	Revision	Read	The peripheral revision number is revision dependent.
[19:12]	Designer	Read	Designer's ID number. This is 0x41 for ARM.
[11:0]	Part number	Read	Identifies the peripheral. The part number for PL172 is 0x172.

Fig.28 on page 230 shows the correspondence between bits of the MPMCPeriphID0-3 registers and the conceptual 32-bit MPMC Peripheral ID register.

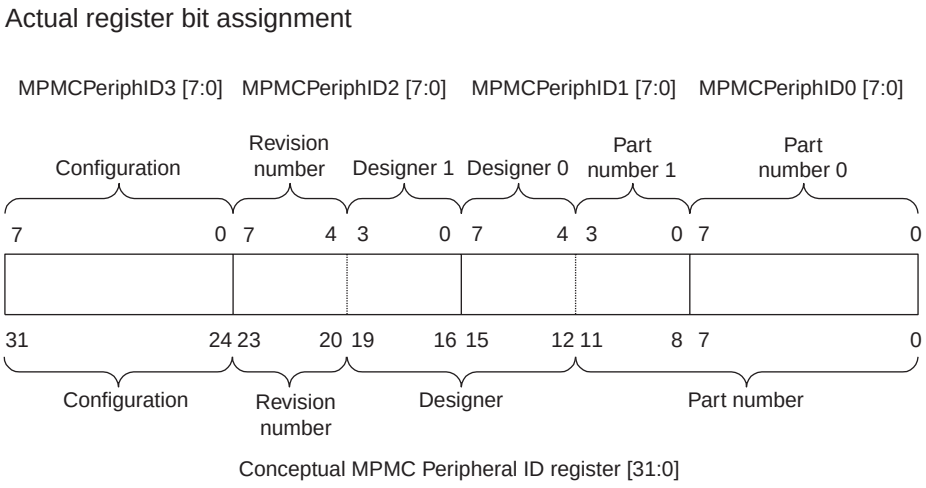


Fig.28 Peripheral identification register bit assignment.

The four eight-bit peripheral identification registers are described in the following subsections:

- MPMCPeriphID0 register on page 231
- MPMCPeriphID1 register on page 232
- MPMCPeriphID2 register on page 232
- MPMCPeriphID3 register on page 233.

22.7.29.1 MPMCPeriphID0 register

The MPMCPeriphID0 register is hard-coded and the fields within the register determine the reset value.

Table 147 on page 231 shows the bit assignments for the MPMCPeriphID0 register.

Table 147MPMCPeriphID0 register

BITS	NAME	TYPE	DESCRIPTION
[31:8]			Reserved, read undefined, must be written as zeros
[7:0]	PartNumber0	Read	These bits read back as 0x72

22.7.29.2 MPMCPeriphID1 register

The MPMCPeriphID1 register is hard-coded and the fields within the register determine the reset value. Table 148 on page 232shows the bit assignments for the MPMCPeriphID1 register.

Table 148MPMCPeriphID1 register

BITS	NAME	TYPE	DESCRIPTION
[31:8]			Reserved, read undefined, must be written as zeros
[7:4]	Designer0	Read	These bits read back as 0x1
[3:0]	PartNumber1	Read	These bits read back as 0x1

22.7.29.3 MPMCPeriphID2 register

The MPMCPeriphID2 register is hard-coded and the fields within the register determine the reset value. Table 149 on page 232 shows the bit assignments for the MPMCPeriphID2 register.

Table 149MPMCPeriphID2 register

BITS	NAME	TYPE	DESCRIPTION
[31:8]			Reserved, read undefined, must be written as zeros
[7:4]	Revision	Read	These bits read back as the revision number, that can be between 0 and 15. ---------- **Note** --------- 0 = PL172 Revision 1 1 = PL172 Revision 2 ----------
[3:0]	Designer1	Read	These bits read back as 0x4.

22.7.29.4 MPMCPeriphID3 register

The MPMCPeriphID3 register is hard-coded and the fields within the register determine the reset value. Table 150 on page 233 shows the bit assignments for the MPMCPeriphID3 register.

Table 150MPMCPeriphID3 register

BITS	NAME	TYPE	DESCRIPTION
[31:8]			Reserved, read undefined, must be written as zeros.
[5:3]	Configuration	Read	Indicates the AHB master bus width: 000 = 32-bit wide 001 = 64-bit wide 010 = 128-bit wide 011 = 256-bit wide 100 = 512-bit wide 101 = 1024-bit wide 110-111 = reserved For PL172 this field is set to 000.
[2]	Configuration	Read	TIC interface: 0 = no 1 = yes For PL172 this field is set to 1.
[1]	Configuration	Read	Data buffers: 0 = no 1 = yes For PL172 this field is set to 1.
[0]	Configuration	Read	Static memory controller:: 0 = no 1 = yes For PL172 this field is set to 1.

22.7.30 MPMCPCELLID0-3 REGISTERS

The MPMCPCellID0-3 registers are four eight-bit wide registers, that span address locations 0xFF0-0xFFC. The registers can conceptually be treated as a single register that holds a 32-bit PrimeCell ID value. The register can be used for automatic BIOS configuration. The MPMCCellID register is set to 0xB105F00D.
The register can be accessed with one wait state.
Table 151 on page 234 shows the bit assignments for the conceptual PrimeCell ID register.

Table 151 Conceptual PrimeCell ID register

BITS	RESET VALUE	REGISTER	BITS	DESCRIPTION
		MPMCPCellID3	[31:8]	Reserved, read undefined, must be written as zeros
[31:24]	0xB1	MPMCPCellID3	[7:0]	These bits read back as 0xB1
		MPMCPCellID2	[31:8]	Reserved, read undefined, must be written as zeros
[23:16]	0x05	MPMCPCellID2	[7:0]	These bits read back as 0x05
		MPMCPCellID1	[31:8]	Reserved, read undefined, must be written as zeros
[15:8]	0xF0	MPMCPCellID1	[7:0]	These bits read back as 0xF0
		MPMCPCellID0	[31:8]	Reserved, read undefined, must be written as zeros
[7:0]	0x0D	MPMCPCellID0	[7:0]	These bits read back as 0x0D

23 EMBEDDED FLASH CONTROLLER

23.1 Features

- AHB Slave port for address and data transfer
- VPB Slave port for configuration and control
- Programming by CPU via VPB
- Programming by external programmer via JTAG
- Programmable sector remapping

23.2 Place in the system

The flash memory controller has an AHB slave port for data transfer, and a VPB port for configuring the controller and for triggering programming and erasing. A JTAG port is available for programming by a TAP controller.

The embedded flash module is capable of generating an interrupt request when burning or erasing is finished. Therefore it is connected to the interrupt controller.

Fig.29 presents the application of the flash memory controller. The AHB data port and AHB configuration port share the same AHB slave port, but have separate addresses (see the ARM memory map), to enable independent address mapping for data (flash memory contents) and configuration (registers).

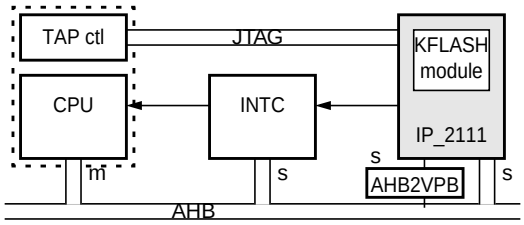


Fig. 29 Flash controller place in the system

23.3 Reading from FLASH

Reading can be done in synchronous or in asynchronous mode, and with several cache options.

23.3.1 SYNCHRONOUS VERSUS ASYNCHRONOUS

With synchronous reading, the transfer of the address to the flash module and read data from the flash module are done clock synchronous. In this mode of operation, the power consumption of the flash module is the lowest. Started reads cannot be stopped, which is the reason that speculative reading and dual caching is not supported with synchronous reading.

With asynchronous reading, the transfer of the address to the flash module and read data from the flash module are done asynchronous, yielding the fastest possible response time. The power consumption of the flash module is the highest. Started reads can be stopped.

23.3.2 CACHING

Caching is offered because the flash memory is 128 bits wide, while the controller only has a 32 bits AHB interface. With caching, a cacheline holds a complete 128 bits FlashWord, from which 4 Words can be read. Without caching, every

AHB data port read starts a flash read. A flash read is a slow process compared to the AHB cycle time. With caching, the average read time is reduced, which can improve system performance.

With a single cache, the most recently read FlashWord stays available at the output of the flash module until the next flash read. When an AHB data port read transfer requires data from the same FlashWord as the previous read transfer, there is a cache hit. No new flash read is done, and read data is given without wait states.

When an AHB data port read transfer requires data from a different FlashWord as the previous read transfer, there is a cache miss. A new flash read is done, and wait states are given until the new read data is available.

With a dual cache, a secondary cache in the flash controller is used. The output of the flash module is considered to be the primary cache. On a primary cache hit, data can be copied to the secondary cache, which allows the flash module to start a speculative read of the next FlashWord.

Both caches are invalidated after:

- initialization
- a configuration register access
- data latch reading
- index sector reading

23.3.3 MODES OF OPERATION

The modes of operation are listed in table 152

Table 152 Read modes

TIMING	CACHING	CHARACTERISTICS / FEATURES	CONFIGURATION
Synchronous	no cache	for single (nonlinear) reads, one FlashWord read per Word read	**fctr.fs_cachebyp** = 1
	single cache	**default mode of operation** most recently read FlashWord is kept until another FlashWord is required lowest power consumption	
Asynchronous	no cache	one FlashWord read per Word read	**fctr.fs_dcr** = 1 **fctr.fs_cachebyp** = 1
	single cache	most recently read FlashWord is kept until another FlashWord is required	**fctr.fs_dcr** = 1 **fbwst.cache2en**=0
	dual cache, single speculative	on a cache miss, a flash read is done, followed by at most one speculative read optimized for execution of code with small loops (< 8 words) from flash	**fctr.fs_dcr** = 1 **fbwst.specalways** = 0
	dual cache, always speculative	most recently used FlashWord is copied into second cache, next FlashWord read is started highest performance for linear reads. With 3 or less wait states, a FlashWord can be read every 4th clock cycle, which enables full speed AHB reading (one Word per clock cycle) for linear reads.	**fctr.fs_dcr** = 1

23.3.4 DATA LATCH READING

The flash memory has data latches to store the data that is to be programmed into the flash array.

Instead of the flash memory contents, the controller can read the data latch contents of the flash memory. Data latch reading is always done uncached, with **fbwst.wst** wait states on every beat of the burst. Data latch reading can be done both synchronously and asynchronously.

Data latch reading is selected with **fctr.rld**.

23.3.5 INDEX SECTOR READING

Index sector reading is always done uncached, with **fbwst.wst** wait states on every beat of the burst. Index sector reading can be done both synchronously and asynchronously.

Index sector reading is selected with **fctr.iss**.

23.3.6 WAIT STATE PROGRAMMING

The flash controller does not wait for the ready signal from the flash memory, but takes data from the memory after a predefined number of clock cycles. These clock cycles are called wait states and can be programmed in **fbwst.wst**. The optimal number of wait states depends on the clock frequency of the AHB clock, and thus could change, with a change in ARM clock frequency.

The formula to calculate the optimal number of wait states is still TBD.

23.4 VPB Programming

23.4.1 INTRODUCTION

Programming an embedded flash memory is more complex than just writing data to the appropriate address, as with embedded SRAM. A flash memory is organized in sectors that must be erased before data can be written into them. A flash memory also has sector wise protection.

Fig. 30shows the programming flow chart. The part on the dark background is automatically done by the hardware of the flash controller.

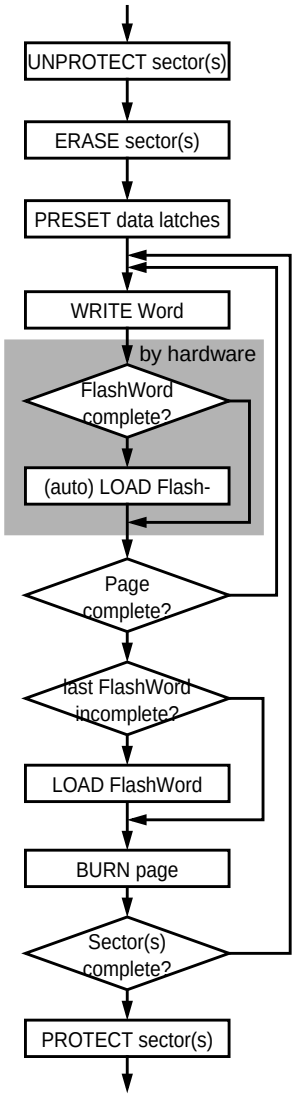


Fig. 30 FLASH AHB programming flow chart

Flash programming contains the following elements:

- unprotecting
- erasing
- presetting data latches
- writing
- loading
- burning
- protecting

In the remainder of this chapter these are described in more detail.

23.4.2 (UN)PROTECTING SECTORS

A sector gets unprotected by writing an even value to its base address, followed by writing the (un)protect trigger value to the **fctr** register. The trigger value for (un)protecting has the following bits set: **loadreq**, **wpb**, **web**, **wre**, **cs**. The other bits are zero.

A sector gets protected by writing an odd value to its base address, followed by the same trigger as for unprotecting.

23.4.3 ERASING SECTORS

Before the erasing, the erase time must be written to the timer register **fptr.tr**, and the timer must be enabled through **fptr.en_t**. During erasing, the timer register counts back to zero. Therefore, the timer register must be rewritten before every erase cycle.

The programmed erase time must obey:

$$(512 \times (fptr \cdot tr) + 2) \times t_{clk} \geq 400ms$$

A single sector gets erased by writing any value to an address within that sector, followed by writing the erase trigger value to the **fctr** register. The trigger value for erasing has the following bits set: **progreq**, **wpb**, **cs**. The other bits are zero.

Only unprotected sectors can be erased.

For erasing, the flash module needs a 66 kHz CRA clock being active. This clock is derived from the AHB clock, dividing it by a factor programmed in **fcra.fcra**. A value of zero inactivates the CRA clock.

Erasing multiple sectors can be done with only one (time consuming) erase cycle. First all sectors except the last are selected for erasure. Then the last sector is erased using the single sector erase procedure.

A sector gets selected for erasure by writing any value to an address within that sector, followed by writing the select for erase trigger value to the fctr register. The trigger value for erase selecting has the following bits set: **loadreq**, **wpb**, **web**, **cs**. The other bits are zero.

23.4.4 PRESETTING DATA LATCHES

When only a part of a page has to be programmed, the data latches for the rest of the page must be preset to logical 1's. This can be done with a single control by setting and clearing the **fctl.fs_pdl** bit.

23.4.5 WRITING AND LOADING

Writing a Word is done through the AHB data port to the data inputs of the flash controller. Every beat takes 2 clock cycles (1 wait state), and results in a partial update of the data input of the flash module.

Writing is done per Word. Byte or halfword writing is not possible. However, because writing logical 1's leaves the flash contents unchanged, it is possible to do byte writing by encapsulating this byte in a Word of logical 1's. This encapsulation must be done by the AHB master that initiates the transfer. The flash controller does not offer this feature.

Every 4th write, a FlashWord (=4 Words) is loaded automatically into the data latches of the flash module. Loading is done per FlashWord.

When for instance addresses 0x00 through 0x0c have to be loaded, loading is done automatically after writing to address 0x0C (note that these four addresses form a single complete FlashWord). This requires that values are already written to addresses 0x00.. 0x08. An AHB incremental burst starting on a FlashWord aligned address meets this requirement.

Loading can also be done manually by writing a '1' to FCTR.FS_LOADREQ.

23.4.6 BURNING

Burning is the data transfer from the data latches of the flash module to the flash array. Before the burning, the burning time must be written to the timer register **fptr.tr**, and the timer must be enabled through **fptr.en_t**. During burning, the timer register counts back to zero. Therefore, the timer register must be rewritten before every burning cycle.

The programmed burning time must obey:

$$(512 \times (fptr \cdot tr) + 2) \times t_{clk} \geq 1ms$$

The burning is started by writing a trigger value to the **fctr** register. The trigger value for burning has the following bits set: **progreq**, **wpb**, **wre**, **cs**. The other bits are zero.

The page address that is offered to the flash module during burning, is the page address of the most recent data port write transfer.

Only pages within unprotected sectors can be programmed.

For burning, the flash memory needs a 66 kHz CRA clock being active. This clock is derived from the AHB clock, dividing it by a factor programmed in FCRA.FCRA. A value of zero inactivates the CRA clock.

23.4.7 INDEX SECTOR PROGRAMMING

The index sector is (un)protected using the same procedure as normal sector (un)protecting, except that the **this** bit is also set as **fctr** trigger.

The erasable part of the index sector is erased using the same procedure as normal sector erasing, except that the **this** bit is also set as fctr trigger.

Writable pages in the index sector are burned using the same procedure as normal burning, except that the this bit is also set as **fctr** trigger. Data is loaded using the normal writing and loading procedure.

23.5 JTAG Programming

TBD

23.6 Miscellaneous

23.6.1 SIGNATURE

To verify the flash contents, a signature can be generated by the flash memory, which in turn must be compared with an expected signature. The alternative would be reading back all contents, which would be much more time and code consuming.

A signature can be generated for any part of the flash contents. The address range to be used for signature generation is defined by writing the start address to the FMSSTART register, and the stop address to the FMSSTOP register.

The signature generation is started by writing a ‘1’ to FMSSTOP.MISR_START. Starting the signature generation is typically combined with defining the stop address, which is done in another field FMSSTOP.FMSSTOP of the same register.

After signature generation, a 16 bits signature can be read from the **fms16** register, and a 128 bits signature can be read from the **fmsw0** .. **fmsw3** registers. The signatures as they are read must be equal to reference signatures. At the writing of this document, the tooling to derive the reference signatures is under development.

23.6.2 BURN / ERASE TIMER

A built-in timer is used to control the burn time or erase time. The timer is started by writing the burn or erase time to the timer register **fptr.tr**, and by enabling it through **fptr.en_t**. During burning or erasing, the timer register counts back to zero, and its current value is returned when reading the **fptr** register. This timer reading can be used to observe the progress of burning/erasing.

While the timer is counting down, the FLASH memory controller is only partly accessible:

- data port transfers are stalled, using AHB wait states
- configuration port write transfers are stalled
- configuration port read transfers are completed normally without stalling

This stalling can have significant impact on system behaviour. Therefore it is advised to read the timer register and the status register first, and only start a data port transfer or configuration port write transfer to the FLASH memory controller when **fptr.tr** is zero, and **fstat.fs_ry** is high.

Because a write transfer to the **fctr** register also stalls, suspending of burning or erasing is not possible.

23.7 Register Overview

The flash memory controller has user registers and JTAG registers. User registers can be accessed through the AHB port. All registers can be accessed through the JTAG port. All registers are listed in table 153

Table 153 Flash memory controller register overview

OFFSET	REGISTER NAME	DESCRIPTION	R/W/ RW (1)	RESET VALUE
User registers				
0xFFC	MODULE_ID	Embedded Flash module identification	R	TBD
0xFEC	INT_SET_STATUS	Set interrupt status bits	S	TBD
0xFE8	INT_CLR_STATUS	Clear interrupt status bits	C	TBD
0xFE4	INT_ENABLE	Interrupt enable bits	R	TBD
0xFE0	INT_STATUS	Interrupt status bits	R	TBD
0xFDC	INT_SET_ENABLE	Set interrupt enable bits	S	TBD
0xFD8	INT_CLR_ENABLE	Clear interrupt enable bits	C	TBD
0x038	FMSW3	FLASH 128 bit signature, word 3	R	TBD
0x034	FMSW2	FLASH 128 bit signature, word 2	R	TBD
0x030	FMSW1	FLASH 128 bit signature, word 1	R	TBD
0x02C	FMSW0	FLASH 128 bit signature, word 0	R	TBD
0x028	FMS16	FLASH 16 bit signature register	R	TBD
0x024	FMSSTOP	FLASH BIST stop address	R/W	TBD
0x020	FMSSTART	FLASH BIST start address	R/W	TBD
0x01C	FCRA	FLASH clock divider for 66 kHz generation	R/W	TBD
0x018	-	reserved		
0x010	FBWST	FLASH bridge wait state register	R/W	TBD
0x00C	-	reserved		
0x008	FPTR	FLASH program time register	R/W	TBD
0x004	FSTAT	FLASH status register	R	TBD
0x000	FCTR	FLASH control register	R/W	TBD
JTAG registers				
n.a.	TBD	TBD	TBD	TBD

1. S = set (set asserted bits, leave others unchanged), C = clear (clear asserted bits, leave others unchanged)

23.8 User registers

23.8.1 MODULE IDENTIFICATION

23.8.2 INTERRUPT REGISTERS

These registers determine when the flash memory controller gives an interrupt request. The int_req output is asserted when the bitwise AND of **int_status** and **int_enable** is nonzero.

Table 154 Flash memory controller **int_status** register

BITS	VARIABLE	DESCRIPTION	R/W/ RW	RESET VALUE
31..3	-	reserved	R	0x0000000
2	END_OF_MISR	End-of-misr interrupt status bit: Bit is set when SIGNATURE generation is finished or when a '1' is written to INT_SET_STATUS[2] Bit is cleared when a '1' is written to INT_CLR_STATUS[2]	R	0
1	END_OF_BURN	End-of-burn interrupt status bit: Bit is set when BURNING a page is finished or when a '1' is written to INT_SET_STATUS[1] Bit is cleared when a '1' is written to INT_CLR_STATUS[1]	R	0
0	END_OF_ERASE	End-of-erase interrupt status bit: Bit is set when ERASING one or more sectors is finished or when a '1' is written to INT_SET_STATUS[0] Bit is cleared when a '1' is written to INT_CLR_STATUS[0]	R	0

Table 155 Flash memory controller **int_set_status** register

BITS	VARIABLE	DESCRIPTION	R/W/ RW	RESET VALUE
31..3	-	reserved	R	0x0000000
2..0	SET_STATUS	Bitwise setting of the INT_STATUS register: 0 => leave corresponding bit unchanged 1=> set corresponding bit.	S	0

Table 156 Flash memory controller **int_clr_status** register

BITS	VARIABLE	DESCRIPTION	R/W/ RW	RESET VALUE
31..3	-	reserved	R	0x0000000
2..0	CLR_STATUS	Bitwise clearing of the INT_STATUS register: 0 => leave corresponding bit unchanged 1=> set corresponding bit.	C	0

Table 157 Flash memory controller **int_enable** register

BITS	VARIABLE	DESCRIPTION	R/W/ RW	RESET VALUE
31..3	-	reserved	R	0x0000000
2	END_OF_MISR	End-of-misr interrupt enable bit: Bit is set when a ‘1’ is written to INT_SET_ENABLE[2] Bit is cleared when a ‘1’ is written to INT_CLR_ENABLE[2]	R	0
1	END_OF_BURN	End-of-burn interrupt enable bit: Bit is set when a ‘1’ is written to INT_SET_ENABLE[1] Bit is cleared when a ‘1’ is written to INT_CLR_ENABLE[1]	R	0
0	END_OF_ERASE	End-of-erase interrupt enable bit: Bit is set when a ‘1’ is written to INT_SET_ENABLE[0] Bit is cleared when a ‘1’ is written to INT_CLR_ENABLE[0]	R	0

Table 158 Flash memory controller **int_set_enable** register

BITS	VARIABLE	DESCRIPTION	R/W/ RW	RESET VALUE
31..3	-	reserved	R	0x0000000
2..0	SET_ENABLE	Bitwise setting of the INT_ENABLE register: 0 => leave corresponding bit unchanged 1=> set corresponding bit.	S	0

Table 159 Flash memory controller **int_clr_enable** register

BITS	VARIABLE	DESCRIPTION	R/W/ RW	RESET VALUE
31..3	-	reserved	R	0x0000000
2..0	CLR_ENABLE	Bitwise clearing of the INT_ENABLE register: 0 => leave corresponding bit unchanged 1=> set corresponding bit.	C	0

23.8.3 FLASH CONTROL REGISTER

The FLASH control register is used to select read modes, and to control the programming of the flash memory.

Table 160 Flash memory controller FLASH control register (**fctr**)

BITS	VARIABLE	DESCRIPTION	R/W/ RW	RESET VALUE
31..16	-	reserved	R	0x0000
15	FS_LOADREQ	Flash data load request: 0 => no request. 1 => write register to Flash, only valid with FS_WRE = 1.Data load is automatically triggered after the last word was written to the load register.	RW	0
14	FS_CACHECLR	Clear data transfer register: 0 => no effect. 1 => set all bits to 1.	RW	0
13	FS_CACHEBYP	Bypasses the cache in the flash bridge: 0 => select cached reading. 1 => select uncached reading.	RW	0
12	FS_PROGREQ	Request arbiter for programming: 0 => no effect. 1 => request for programming.	RW	0
11	FS_RLS	Select sector latches for reading: 0 => read flash array. 1 => read sector latches.	RW	0
10	FS_PDL	Preset data latches: 0 => no effect. 1 => set all bits in the data latches to 1.	RW	0
9	-	reserved	RW	
8	DISABLE_REMAP	Address remapping: 0 => logical to physical conversion enabled. 1 => logical to physical conversion disabled.	RW	0
7	FS_WPB	Program/Erase protection: 0 => program/erase disabled. 1 => program/erase enabled.	RW	0
6	FS_ISS	Index sector selection: 0 => select flash array. 1 => select index sector.	RW	0
5	FS_RLD	Read flash data latches: 0 => read flash array. 1 => read data latches for verification of data that is loaded to be programmed.	RW	0
4	FS_DCR	DC read mode: 0 => select synchronous reading. 1 => select asynchronous reading.	RW	0
3	FS_SUS	Suspend program/erase operation for read access: 0 => continue program/erase operation. 1 => suspend program/erase operation.	RW	0

BITS	VARIABLE	DESCRIPTION	R/W/ RW	RESET VALUE
2	FS_WEB	Program/erase enable: 0 => enable program/erase. 1 => disable program/erase.	RW	1
1	FS_WRE	Program/erase selection: 0 => select erase. 1 => select program/data load.	RW	0
0	FS_CS	Flash chip select: 0 => standby mode. 1 => active mode.	RW	1

23.8.4 FLASH STATUS REGISTER

The FLASH status register is a read-only register for status information.

Table 161 Flash memory controller FLASH status register (**fstat**)

BITS	VARIABLE	DESCRIPTION	R/W/ RW	RESET VALUE
31..8	-	reserved	R	0x0000
7	FS_CHECK1/GO	All 1 check: 0 => all 1 check failed. 1 => all 1 check passed.	R	0
6	FS_CHECK0	All 0 check: 0 => all 0 check failed. 1 => all 0 check passed.	R	1
5	FS_ERR	Bit error detection: 0 => no errors detected. 1 => bit error was detected and corrected.	R	0
4	FS_HVONN	Negative high voltage presence, during program/erase: 0 => negative high voltage is not generated. 1 => negative high voltage is generated.	R	0
3	FS_HVONP	Positive high voltage presence, during program/erase: 0 => positive high voltage is not generated. 1 => positive high voltage is generated.	R	0
2	FS_RY	Flash ready indication: 0 => read or program or erase is in progress. 1 => one of the following conditions is true: a) synchronous read has completed b) program/erase is suspended, read can start c) program/erase has finished.	R	1
1	FS_PROGGNT	Flash bus lock grant: 0 => Flash bus lock request for program/erase is not granted. 1 => Flash bus lock request for program/erase is granted.	R	0
0	FS_BSYN	Program busy not: 0 => during program/erase (also during suspend). 1 => total program/erase finished. FS_BSYN is the logical AND of F_WEB, FS_SUS, F_HVONN and F_HVONP.	R	1

23.8.5 FLASH PROGRAM TIME REGISTER

The FLASH program time register controls the timer for all programming tasks (burning and erasing). It also allows to read the remaining burn or erase time.

Table 162 Flash memory controller FLASH program time register (**fptr**)

BITS	VARIABLE	DESCRIPTION	R/W/ RW	RESET VALUE
31..16	-	reserved	R	0x0000
15	EN_T	Program timer Enable: 0 => timer disabled. 1 => timer enabled.	RW	0
14..0	TR	Program timer: Remaining burn/erase time is 512 * TR clock cycles	RW	0x0000

23.8.6 FLASH WAIT STATES REGISTER

The FLASH wait state register controls the number of wait states that is inserted for AHB transfers. This register also controls the second cache (for asynchronous reading).

Table 163 Flash memory controller FLASH wait states register (**fbwst**)

BITS	VARIABLE	DESCRIPTION	R/W/ RW	RESET VALUE
31..16	-	reserved	R	0x0000
15	CACHE2EN	Enable second cache: 0 => second cache disabled. 1 => second cache enabled.	RW	1
14	SPECALWAYS	Speculative reading: 0 => single speculative reading. 1 => always speculative reading.	RW	1
13..8	-	reserved	R	0x00
7..0	WST	number of wait states	RW	0x04

23.8.7 FLASH CLOCK DIVIDER REGISTER

The FLASH clock divider register controls the clock divider for the CRA clock. This clock should be programmed to 66 kHz during burning or erasing.

Table 164 Flash memory controller FLASH clock divider register (**fcra**)

BITS	VARIABLE	DESCRIPTION	R/W/ RW	RESET VALUE
31..12	-	reserved	R	0x00000
11..0	FCRA	Clock divider setting: 0x000 => no CRA clock to FLASH memory. > 0x000 => CRA clock to FLASH memory,frequency is AHB clock frequency divided by (FCRA*3 + 1). To be programmed such that CRA clock frequency is 66 kHz +- 20%.	RW	0x000

23.8.8 FLASH BIST CONTROL

The FLASH BIST control registers control the embedded signature generation. Because the FLASH BIST operates on FlashWords, the start and stop addresses must be entered as FlashWord addresses. These can be derived from the AHB byte addresses through division by 16 (omitting 4 lowest bits).

Table 165 Flash memory controller FLASH BIST start address register (**fmsstart**)

BITS	VARIABLE	DESCRIPTION	R/W/ RW	RESET VALUE
31..17	-	reserved	R	0x0000
16..0	FMSSTART	BIST start address (divided by 16, FlashWord aligned)	RW	0x00000

Table 166 Flash memory controller FLASH BIST stop address register (**fmsstop**)

BITS	VARIABLE	DESCRIPTION	R/W/ RW	RESET VALUE
31..18	-	reserved	R	0x0000
17	MISR_START	BIST start bit	RW	0
16..0	FMSSTOP	BIST stop address (divided by 16, FlashWord aligned)	RW	0x00000

23.8.9 FLASH BIST sIGNATURE REGISTERS

The FLASH BIST signature registers return the signatures as produced by the embedded signature generators. There is a 16 bit signature, and a 128 bit signature. Some reset values are undefined because either they depend on the flash memory status or these registers are not reset at all.

Table 167 Flash memory controller FLASH BIST 16 bit signature register (**fms16**)

BITS	VARIABLE	DESCRIPTION	R/W/ RW	RESET VALUE
31..16	-	reserved	R	0x0000
15:11	-	reserved	R	0x00
10	CHECK0	indicates that all bits are zero	R	-
9	CHECK1	indicates that all bits are one	R	-
8	ERR	indicates a bit error has occurred	R	-
7..0	QPAR	parity bit output	R	-

Table 168 Flash memory controller FLASH BIST 128 bit signature register, word 0 (**fmsw0**)

BITS	VARIABLE	DESCRIPTION	R/W/ RW	RESET VALUE
31..0	FMSW0	FLASH BIST 128 bit signature, bit 31:0	R	-

Table 169 Flash memory controller FLASH BIST 128 bit signature register, word 1 (**fmsw1**)

BITS	VARIABLE	DESCRIPTION	R/W/ RW	RESET VALUE
31..0	FMSW1	FLASH BIST 128 bit signature, bit 63:32	R	-

Table 170 Flash memory controller FLASH BIST 128 bit signature register, word 2 (**fmsw2**)

BITS	VARIABLE	DESCRIPTION	R/W/ RW	RESET VALUE
31..0	FMSW2	FLASH BIST 128 bit signature, bit 95:64	R	-

Table 171 Flash memory controller FLASH BIST 128 bit signature register, word 3 (**fmsw3**)

BITS	VARIABLE	DESCRIPTION	R/W/ RW	RESET VALUE
31..0	FMSW3	FLASH BIST 128 bit signature, bit 127:96	R	-

23.9 JTAG Register Overview

The flash memory controller has user registers and JTAG registers. User registers can be accessed through the AHB port. All registers can be accessed through the JTAG port. All registers are listed in table 153

Table 172 Flash memory controller register overview

OFFSET	REGISTER NAME	DESCRIPTION	R/W/ RW (1)	RESET VALUE	SCAN SCAIN LOCATION ??
User registers					
0x038	FMSW3	FLASH 128 bit signature, word 3	R	TBD	chain8(190:159)
0x034	FMSW2	FLASH 128 bit signature, word 2	R	TBD	chain8(158:127)
0x030	FMSW1	FLASH 128 bit signature, word 1	R	TBD	chain8(126:95)
0x02C	FMSW0	FLASH 128 bit signature, word 0	R	TBD	chain8(94:63)
0x028	FMS16	FLASH 16 bit signature register	R	TBD	chain8(62:47)
0x024	FMSSTOP	FLASH BIST stop address	R/W	TBD	chain8(46:29)
0x020	FMSSTART	FLASH BIST start address	R/W	TBD	chain8(28:12)
0x018	FCHAR	FLASH memory status info	R	TBD	chain6(11:0)
0x00C	FTCTR	FLASH test control register	RW	TBD	chain1(35:4)
JTAG registers					
n.a.	FTBD	Auto address update settings	R/W	TBD	chain1(3:0)
n.a.	FBADDR	Address to flash module	R/W	TBD	chain2(16:0)
n.a.	FDBG		R/W	TBD	chain3(2:0)
n.a.	FBDATA	Data and parity to/from flash module	R/W	TBD	chain4(247:0)
n.a.	ERR	Pattern error	R/W	TBD	chain6(12)
n.a.	flash_data	Data, parity and ready inside flash module	R/W	TBD	chain7(136:0)
n.a.	CHAIN	Chain control	R/W	TBD	before all chains

1. S = set (set asserted bits, leave others unchanged), C = clear (clear asserted bits, leave others unchanged)

23.9.1 FLASH TEST CONTROL REGISTER

The FLASH test control register gives access to several pins of the flash memory instance.

Table 173 Flash memory controller FLASH test control register (**ftctr**)

BITS	VARIABLE	DESCRIPTION	R/W/ RW	RESET VALUE
31	FS_PARSEL	not used	RW	0
30	FS_TDC	Forces DCR pin to high: asynchronous read mode	RW	0
29	FS_BYPASS_R	Connected to FLASH memory pin BYPASS_R	RW	0
28	FS_BYPASS_W	Connected to FLASH memory pin BYPASS_W	RW	0
27	FS_CHECK_EN	Connected to FLASH memory pin CHECK_EN	RW	0
26	FS_FFR	Connected to FLASH memory pin FFR	RW	0
25	F_ECCTST	Connected to FLASH memory pin ECCTST	RW	0
24	FS_HVSS(2)	Selects which internal signal will be on pin DCM, see FS_HVSS(1:0)	RW	0
23	F_TRIP20	Connected to FLASH memory pin TRIP20	RW	0
22	F_TRIP5	Connected to FLASH memory pin TRIP5	RW	0
21	F_TRIP1	Connected to FLASH memory pin TRIP1	RW	0
20	F_MARI	Connected to FLASH memory pin MARI	RW	0
19	F_EVN	Connected to FLASH memory pin EVN	RW	0
18	F_MARW	Connected to FLASH memory pin MARW	RW	0
17	F_EVP	Coonnected to FLASH memory pin EVP	RW	0
16	F_IVP	Connected to FLASH memory pin IVP	RW	0
15	F_RRS	Connected to FLASH memory pin RRS	RW	0
14	F_FSS	Connected to FLASH memory pin FSS	RW	0
13..12	FS_HVSS(1:0)	FS_HVSS(2:0) selects which internal signal will be on pin DCM: 000 => enppsc 001 => enpndc 010 => enppdc 011 => hvonn 100 => hvonp 101 => go 110 => f_RY 111 => reserved	RW	0
11	F_FAS	Connected to FLASH memory pin FAS	RW	0
10	F_FOG	Connected to FLASH memory pin FOG	RW	0
9	F_FEG	Connected to FLASH memory pin FEG	RW	0
8	F_DAW	Connected to FLASH memory pin DAW	RW	0
7	F_SOW	Connected to FLASH memory pin SOW	RW	0
6	F_SEW	Connected to FLASH memory pin SEW	RW	0
5	F_DCT	Connected to FLASH memory pin DCT	RW	0
4	F_MSS	Connected to FLASH memory pin MSS	RW	0
3	FS_INVCKB	Condition to select status bit 7, all ‘1’ selects go-nogo test	RW	0
2	FS_CKB	Condition to select status bit 7, all ‘1’ selects go-nogo test	RW	0
1	FS_ALL0	Condition to select status bit 7, all ‘1’ selects go-nogo test	RW	0
0	FS_ALL1	Condition to select status bit 7, all ‘1’ selects go-nogo test	RW	0

23.9.2 FLASH MEMORY STATUS REGISTER

The FLASH memory status register returns status information from the flash memory. Some reset values are undefined because they depend on the KFLASH status.

Table 174 Flash memory controller FLASH memory status register (**fchar**)

BITS	VARIABLE	DESCRIPTION	R/W/ RW	RESET VALUE
31..12	-	reserved	R	0x00000
11..9	FS_SL	Connected to FLASH memory pins SL(2:0)	R	undefined
8	FS_SORRO	Connected to FLASH memory pin SORRO	R	undefined
7	FS_SERRO	Connected to FLASH memory pin SERRO	R	undefined
6	FS_ENPPSC	Connected to FLASH memory pin ENPPSC	R	undefined
5	FS_ENPPDC	Connected to FLASH memory pin ENPPDC	R	undefined
4	FS_ENPNDC	Connected to FLASH memory pin ENPNDC	R	undefined
3..0	-	reserved	R	undefined

23.9.3 FLASH CLOCK DIVIDER REGISTER

The FLASH clock divider register controls the clock divider for the CRA clock. This clock should be programmed to 66 kHz during burning or erasing.

Table 175 Flash memory controller FLASH clock divider register (**fcra**)

BITS	VARIABLE	DESCRIPTION	R/W/ RW	RESET VALUE
31..12	-	reserved	R	0x00000
11..0	FCRA	Clock divider setting: 0x000 => no CRA clock to FLASH memory. > 0x000 => CRA clock to FLASH memory,frequency is AHB clock frequency devided by (FCRA*3 + 1). To be programmed such that CRA clock frequency is 66 kHz +- 20%.	RW	0x000

23.10 *Operating conditions*

TBD

24 SIMPLE DMA CONTROLLER

24.1 Overview

The Simple DMA interface, or SDMA, is a small AHB bus master specifically designed for bulk memory transfers over the AHB bus.

This can be:

- Memory to Memory copy:

  Memory can be copied from the source address to the destination address with a specified length, while incrementing the address for both the source and destination. This is the same as performing a so called ‘memcpy();’
- Memory to peripheral:

  Data is transferred from incrementing memory to a fixed address of a peripheral, when the peripheral demands it.
- Peripheral to memory:

  Data is transfered from a fixed address of a peripheral to incrementing memory when the peripheral demands it.
- Peripheral to peripheral:

  Data is transfered when both peripherals demand data transfer.

For each mode, the length of the operation is specified.

When half of this length is reached, or when the end of the transfer has been reached, the CPU can be interrupted or the CPU can poll for notification of this event.

The SDMA controller has a flexible amount of channels (from 1 to 15), which is fixed after hardware-synthesis. Each channel can be configured with its own source, destination, length and control information.

For the PNX0101, the configuration is **8 channels**. (4 channels if all channels use scatter gather)

24.2 Functional Description

24.2.1 HARDWARE

- APB / VPB interface for sending control data from CPU to DMA.
- Performs mem-to-mem copies in 2 AHB cycles, and mem to peripheral or peripheral to mem in 3 AHB cycles.
- Supports byte, halfword and word transfers, and correctly alligns it over the AHB bus.
- Supports 30 peripherals for DMA flow control.
- Compatible with ARM flow control, for single requests (sreq), last single requests(lsreq), terminal count info(tc) and dma clearing (clr).
- Supports swapping in endianess of the transported data, for file reading / MP3 decoding purposes
- Contains maskable interrupts for each raw IRQ
- Uses ‘HPROT’ of the AHB bus for buffer disabling for peripheral transfers and the last DMA transfer
- Because of its simplicity, the SDMA is very small in size.
  About 4 K gates for 1 synthesized channel.
  About 20K gates for 8 synthesized channels.
- Most of the Flip-flops of the SDMA are static like source, destination, control and length registers. These can be put on a gated clock-domain to conserve power.
- Supports external enabling of DMA channels, so others sources then the CPU can enable one or more DMA channels

24.2.2 PHYSICAL INTERFACE

The following pins are on the top level entity:

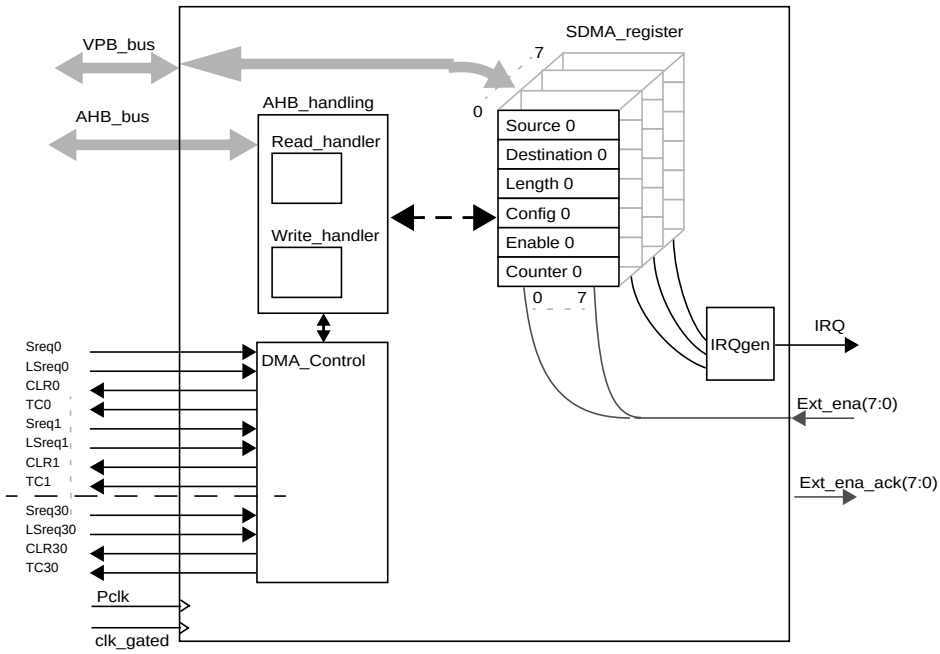


Fig. 31 DMA Context SimpleDMA-controller

APB slave bus signals:

PENABLE
PSELDMA
PADDR
PWRITE
PWDATA
PRDATA

IRQ

AHB master signals:

HPROT
HRESETN
HRDATAIN
HREADYIN
HRESPIN
HGRANTAHB
HADDROUT
HTRANSOUT
HWRITEOUT
HSIZE

HBURST
HWDATAOUT
HBUSREQAHB

Clocks:

PCLK

This is the clock the SDMA operates on. This is the clock for both the VPB and AHB bus.

CLK_GATED

This clock is the same as PCLK, but this one may be disabled during the inactive periods of the APB/VPB bus towards the SDMA, to reduce power. The clock disabling will not be done in the SMDA, and has to be done in for instance the clock-generation unit (CGU) .
On both the PNX0101 and Melody, clock gating is done on the active period of PSEL.

Slave response signals:

SDMA_SREQ (30 downto 0)
30 Pins arriving from DMA slaves, which represents flow control request signals
Inside the Configuration register, one of these pins can be checked if the read or write transfer from or to that specific peripheral can be started.
These signals may be generated on a different clock domain.
‘0’ means the peripheral is not ready
‘1’ means the peripheral has data available / can accept data.

SDMA_LSREQ (30 downto 0)
30 Pins arriving from DMA slaves(optional for slaves).
The behaviour is a mirror of SDMA_SREQ, except that this signal indicates the last transfer performed by the slave.
If this signal was active, then the DMA channel that uses this request line will be disabled, and the CPU will be interrupted.
These signals may be generated on a different clock domain.
‘0’ means the peripheral is not ready
‘1’ means the peripheral has just indicated a last transfer can be performed.

SDMA_TC (30 downto 0)
30 Pins going towards DMA slaves (Optional for slaves).
By using this signal, a slave can recognise that the last transfer has just been performed by the SDMA. Valid during the active period of ‘SDMA_CLR’.
These signals may be generated on a different clock domain.
‘0’ means the performed transfer was not the last transfer
‘1’ means the performed transfer was the last transfer.

SDMA_CLR (30 downto 0)
30 Pins going towards the DMA slaves (optional).
If the slave supports DMA flow control, this signal is used to indicate to the slave that the single word/halfword/byte transfer is complete, so the slave can restart a the request handshake.
‘0’ means the transfer is not complete
‘1’ means the transfer is complete.

Special signals:

EXT_EN (nr_of_channels downto 0)

Signals arriving from external hardware that can also enable a DMA channel. Each bit corresponds to an equivalent SDMA channel. These signals may be generated on a different clock domain.

EXT_EN_ACK (nr_of_channels downto 0)

The acknowledge of the external enabling, when a channels has finished its transfer. Each bit corresponds to a equivilent DMA channel.
If this signal will be used on a different clock domain, the receiving side must take care of the clock-crossing.

24.3 Registers

Table 176SDMA register map

ADDRESS SPACE	OFFSET	REGISTER NAME	R/W ACCESS	RESET
<IC-specific>	0x00	channel 0: SOURCE_ADDRESS	R/W	0x0
	0x04	channel 0: DESTINATION_ADDRESS	R/W	0x0
	0x08	channel 0: TRANSFER_LENGTH	R/W	<max>
	0x0C	channel 0: CONFIGURATION	R/W	0x0
	0x10	channel 0: ENABLE	R/W	0x0
	0x14-0x18	channel 0: <reserved>	-	
	0x1C	channel 0: TRANSFER_COUNTER	R/W	0x0
	0x20-0x3c	Channel1 registers	R/W	
	0x40-0x5c	Channel 2 registers	R/W	
	0x60-0x7c	Channel 3 registers	R/W	
	0x80-0x9c	Channel 4 registers	R/W	
	0xA0-0xBC	Channel 5 registers	R/W	
	0xC0-0xDC	Channel 6 registers	R/W	
	0xE0-0xFC	Channel 7 registers	R/W	
	0x100 - 0x1FC	<reserved for channels 8 to 15>	-	
	0x200	channel 0: ALT_SOURCE_ADDRESS	W	
	0x204	channel 0: ALT_DESTINATION_ADDRESS	W	
	0x208	channel 0: ALT_TRANSFER_LENGTH	W	
	0x20C	channel 0: ALT_CONFIGURATION	W	
	0x210-0x21c	ALT Channel 1,offset	W	
	0x220-0x22c	ALT Channel 2 registers	W	

ADDRESS SPACE	OFFSET	REGISTER NAME	R/W ACCESS	RESET
	0x230-0x23c	ALT Channel 3 registers	W	
	0x240-0x24c	ALT Channel 4 registers	W	
	0x250-0x25c	ALT Channel 5 registers	W	
	0x260-0x26c	ALT Channel 6 registers	W	
	0x270-0x27c	ALT Channel 7 registers	W	
	0x260 - 0x3FC	<alternative addresses reserved for channel 8-15>	-	
	0x400	ALT_ENABLE	R/W	0x0
	0x404	IRQ_STATUS_CLEAR	R/W	0x0
	0x408	IRQ_MASK	R/W	0xFFFF
	0x40c	TEST_FIFO_RESP_STATUS	R	-
	0x410	SOFT_INT	W	-

SOURCE_ADDRESS

This register contains the address from where the data is read from. This data will remain static during all memory transfers, and will only be changed by re-programming it.

DESTINATION_ADDRESS

This register contains the address to where the data is written to.This data will remain static during all memory transfers, and will only be changed by re-programming it.

TRANSFER_LENGTH

This register contains the amount of cycles to ransfer. This can be bytes, half-words or words. This data will remain static thoughout the memory transfer, and will only be changed by re-programming it.

The number of transfers performed is:
<The number programmed in this register> + 1

If for example 0x100 is programmed in this register, a total of 0x101 transfers will be performed.

Current maximum for PNX0101 is 4096K transfers.

CONFIGURATION

This register contains configuration information for the transfer. see table number 177. This data will remain static durigng all memory transfers, and will only be changed by re-programming it

Table 177CONFIGURATION_n

BIT	LABEL	RESET	R/W
4-0	WRITE_SLAVE_NR	0x0	R/W
9-5	READ_SLAVE_NR	0x0	R/W
11-10	TRANSFER_SIZE	0x0	R/W
12	INVERT_ENDIANESS	0	R/W
15-13	COMPANION_CHANNEL_NR	000	R/W
16	<reserved for table 8-15>	0	R
17	COMPANION_CHANNEL_ENABLE	0	R/W
18	CIRCULAR_BUFFER	0	R/W
32-19	<RESERVED>	0x0	R

Write_slave_nr

0x0:
When 0x0 is written to this register, means that the transfer is unconditional for each write, and the write-address is incremented each write-cycle.

0x1 - 0x1f:
If a number higher then 0x0 is programmed here, then flow control for each write is used, checking for the status of that specific slave-FIFO pin, decremented by 1.
The address is NOT incremented for each write, so the same FIFO is written each time.

Example: if a peripheral is connected to pin 2 of the SDMA, and flow control for this slave must be used, then 0x3 must be programmed (0x2 + 1 = 0x3)

Read_slave_nr

0x0:
When 0x0 is written to this register, means that the transfer is unconditional for each read, and the read-address is incremented each read-cycle.

0x1 - 0x1f: If a number higher then 0x0 is programmed here, then flow control is used for the status of that specific slave-FIFO pin, decremented by 1. The address is NOT incremented for each write, so the same FIFO is read from each time.

So if a peripheral is connected to pin 3 of the SDMA, and flow control for this slave will be used, then 0x4 must be programmed for these bits (0x3 + 1)

Transfer_size

This register contains the size of each transfer:
0x0: Transfer of words
0x1: Transfer of half-words
0x2: Transfer of bytes.

If half-words or bytes are used, these will be correctly alligned over the AHB bus.

Invert_endianess

0 no endianess inversion while transfering
1 In this setting the endianess of the word, or half-word is switched. This is for instance required for MP3 decoding from PC-files.

If the transfer is word alligned, then:
Byte 3 and byte 0 are swapped
Byte 2 and byte 1 are swapped

If the transfer is half-word alligned then:
Byte 1 of the half-word is swapped with byte 0

Note: It is allowed to use the same source and destination address to change the endianess of the data without copying it to a different location.

Companion_channel_nr

If the companion_channel_enable bit is set, the channel number programmed in this register will be enabled when the current channel has finished transfer.

This allows also the use of a linked-list / scatter-gather method. See the ‘scatter-gather’ chapter.

Companion_channel_enable

If this bit is set, the channel number programmed inside the copy_channel_number bits will be enabled a the end of the current transfer.

Circular_buffer

When this bit is set, the enable bit inside the enable register will never be cleared, and the channel will keep looping. However the CPU can still be interrupted at the end of each loop and halfway each loop.
This is a good alternative to channel-companions, since this technique requires only one channel to be enabled, saving channels

<Reserved>

Reserved for Future use. Treat as ‘0’

ENABLE

Writing a ‘1’ will enable this channel.
Writing a ‘0’ will disable this channel.

This register will be automatically be disabled when the transfer is finished, OR when the slave has indicated the last transfer using ‘SDMA_LSREQ’

If the transfer is disabled by the CPU in the middle of a transfer, software might want to reset the counter, by writing to the ‘ADDRESS_COUNTER’ register of that channel. This counts also if the SDMA was stopped because ‘LSREQ’ was active.

TRANSFER_COUNTER

Reading this register returns the current counter status of the channel.
The counter starts at ‘0’ and ends at the length programmed in the ‘length’ register.

When a transfer is finished, the counter is reset to ‘0’.
When a transfer is stopped by a slave, by using the ‘SDMA_LSREQ’ signal, the counter indicates the amount of transfers performed.

Writing this register resets the counter to ‘0’.

Example: If this counter reads 0x100, then 0x100 transfers have been performed.

NOTE: If a channel is disabled in the middle of a transfer by the CPU, OR the counter was stopped by using the ‘SDMA_LSREQ’, then writing to this register is the only method of resetting the counter to ‘0’.

ALT_SOURCE_ADDRESS

This register is a mirror of the SOURCE_ADDRESS register, and can only be written to.
See chapter ‘Using the linked-list method’ for more details.

ALT_DESTINATION_ADDRESS

This register is a mirror of the *DESTINATION_ADDRESS_n* register, and can only be written to.
See chapter ‘scatter gathering’ for more details.

ALT_TRANSFER_LENGTH

This register is a mirror of the *TRANSFER_LENGTH_n* register, and can only be written to.
See chapter ‘Using the linked-list method’ for more details.

ALT_CONFIGURATION

This register is a mirror of the CONFIGURATION register, and can only be written to.
See chapter ‘Using the linked-list method’ for more details.

ALT_ENABLE

This register allows enabling and disabling of multiple channels at the same time.
Each bit represents a channel number, so
Bit 0 = Channel 0,
bit 1 = Channel 1,
etc...

Please use the individual ‘enable’ register of each channel, in stead of using this register!

A read-modify-write to this register might write the wrong data, since there is a chance the enable register of one of the channels has updated itself in between the read-modify-write, because:
- it is either finished,
- was activated by a ‘copy-table’ setting,
- or when external enabling was involved

IRQ_STATUS_CLR

The IRQ status register contains information if a channel has fished its transfer OR if the channel is halfway. See Table 178.

A bit which has been set can only be cleared by writing a ‘1’ to this bit in this register.

The ‘finish’ bit will only be set when the channel is finished. After clearing, it will only be set again if the channel finishes again.
The ‘half-way’ bit will only be set when the channel has passed the half of the transfer. After clearing, this bit will only be set when the channels passes halfway again.

The soft interrupt bit will be set when the SOFT_INT register is written to. This is for scatter-gather interrupt control: The last transfer of a scatter-gather operation can write towards this register, so the CPU knows the scatter-gather operation has finished.

The ‘DMA_abort’ bitwill be active if any transfer done by the SDMA controller resulted in an abort on the AHB bus.

Table 178IRQ_STATUS_CLR register

BIT	LABEL	RESET	R/W
0	Channel 0 is finished	0	R/W
1	Channel 0 is more then half-way	0	R/W
2	Channel 1 is finished	0	R/W
3	Channel 1 is more then half-way	0	R/W
4	Channel 2 is finished	0	R/W
5	Channel 2 is more then half-way	0	R/W
6	Channel 3 is finished	0	R/W
7	Channel 3 is more then half-way	0	R/W
8	Channel 4 is finished	0	R/W
9	Channel 4 is more then half-way	0	R/W
10	Channel 5 is finished	0	R/W
11	Channel 5 is more then half-way	0	R/W
12	Channel 6 is finished	0	R/W
13	Channel 6 is more then half-way	0	R/W
14	Channel 7 is finished	0	R/W
15	Channel 7 is more then half-way	0	R/W
29-16	Reserved for more channels	0	R/W
30	Soft interrupt, scatter gather	0	R/W
31	DMA abort	0	R/W

Clearing interrupts: Writing a ‘1’ to a bit will clear the interrupt belonging to that bit.

IRQ_MASK

Table 179IRQ_MASK register

BIT	LABEL	RESET	R/W
0	IRQ of STATUS bit 0 is masked	1	R/W
1	IRQ of STATUS bit 1 is masked	1	R/W
2	IRQ of STATUS bit 2 is masked	1	R/W
3	IRQ of STATUS bit 3 is masked	1	R/W
4	IRQ of STATUS bit 4 is masked	1	R/W
5	IRQ of STATUS bit 5 is masked	1	R/W
6	IRQ of STATUS bit 6 is masked	1	R/W
7	IRQ of STATUS bit 7 is masked	1	R/W
8	IRQ of STATUS bit 8 is masked	1	R/W
9	IRQ of STATUS bit 9 is masked	1	R/W
10	IRQ of STATUS bit 10 is masked	1	R/W
11	IRQ of STATUS bit 11 is masked	1	R/W
12	IRQ of STATUS bit 12 is masked	1	R/W
13	IRQ of STATUS bit 13 is masked	1	R/W
14	IRQ of STATUS bit 14 is masked	1	R/W
15	IRQ of STATUS bit 15 is masked	1	R/W
29-16	Reserved for more channels	0	R/W
30	IRQ of Soft interrupt is masked	1	R/W
31	IRQ of DMA abort	1	R/W

TEST_FIFO_RESP_STAT

Test register only. Not usefull for functional operation.
This register reads out the exact status of the FIFO-response pins going towards the SDMA.

Bit 30-0 represents the connected SREQ signals. By reading this register, it can be tested if the SREQ pins are correctly connected on a system level.

SOFT_INT

Writing to this register will enable the soft_interrupt IRQ, in the IRQ status register.
This register exists purely for linked-list, so the last transfer can be a write towards this address.
This way the CPU knows exactly when a scatter-gather operation is finished.

24.4 Channel Arbitration

If the SDMA contains more then 1 channel, arbitration is required so that each channel gets equal access over the AHB bus.

The channels are checked in a round-robin (circular) motion. First channel 0 gets a transfer, then 1, then 2... then 0 again and so forth.

When the current transfer is finished, the next channel will be checked first. The arbitration goes like this:

IF (next channel = enbled)
    IF (mem-to-mem) Current_channel = this channel

else IF (peripheral to mem) OR (mem to peripheral) OR (peripheral to peripheral)
If (selected peripheral(s) = ready) Current_channel = this channel

Else IF (next channel + 1 = enabled)
IF (mem-to-mem) Current_channel = this channel
else IF (peripheral to mem) OR (mem to peripheral) OR (peripheral to peripheral)
If (selected peripheral(s) = ready) Current_channel = this channel

Else IF (next channel + 2 = enabled)

.....
.....
Else Current_channel = UNCHANGED

A new channel is known within a single clock-cycle. Every new AHB cycle the channels are re-arbitrated. As soon as a channel is ready to start a transfer, this channel gets access, and arbitration will continue from this point.

24.5 Scatter gathering / Building a linked-list

Scatter gathering is a method where data is located in lots of different areas of memory and needs to be ‘gathered’ to one location as a whole. This might be mem to mem, but also mem/per combinations.
In memory, the CPU can program a linked-list which consists of source , destination and length entries. Each entry will be executed as if a channel was programmed by these values and started, in a sequential order.

The SDMA supports a linked-list by using 2, sequential, channels.

The first channel will execute the contents of the linked list.
The second channel will perform the updating of the linked list entry.

The way it is implemented in the SDMA, is that the SDMA actually re-programs one of its own channels (the previous one), replacing the contents of a channel with the contents of the linked-list, then enables this channel.

A linked-list entry consists of 5 words. The first 4 words reprograms the first DMA channel (the one that executes the contents of the link-list). The fifth word overwrites the source address of the current DMA channel, thus updating the linked list entry to the next location. See the SDMA register map: the ALT_addresses.
You’ll notice the fifth word will overwrite the source address of the next channel.

See table below how a linked list may look like:

Table 180 Linked list example

ADDRESS	LABEL	REMARK #
n=0: LINKED_LIST_BASE_ADDESS + n*0x14 + 0x0	Source address n	0
n*0x14 + 0x4	Destination address n	
n*0x14 + 0x8	Transfer Length n	
n*0x14 + 0xc	Configuration n	1
n*0x14 + 0x10	Linked_list_Base_address + (n+1)*0x14	Addr next entry
(n+1)*0x14 + 0x0	Source address (n+1)	
(n+1)*0x14 + 0x4	Destination address (n+1)	
(n+1)*0x14 + 0x8	Transfer Length (n+1)	
(n+1)*0x14 + 0xc	Configuration (n+1)	1
(n+1)*0x14 + 0x10	Linked_list_Base_address + (n+2)*0x14	Addr next entry
-	Room for more entires	2
Last_entry (option 1):		
y=last entry number: (n+y)*0x14 + 0x0	Source address (n+y)	
(n+y)*0x14 + 0x4	Destination address (n+y)	
(n+y)*0x14 + 0x8	Transfer Length (n+y)	
(n+y)*0x14 + 0xc	Configuration (n+y)	Disable companion table option
(n+y)*0x14 + 0x10	0x0	Don’t care
Last_entry (option 2):		
y=last entry number: (n+y)*0x14 + 0x0	Source address (n+y)	
(n+y)*0x14 + 0x4	Destination address (n+y)	
(n+y)*0x14 + 0x8	Transfer Length (n+y)	
(n+y)*0x14 + 0xc	Configuration(n+y)n	
(n+y)*0x14 + 0x10	Linked_list_Base_address	Jump to the start of the linked list, making it a circular linked list.
Last_entry (option 3):		3
y=last entry number: (n+y)*0x14 + 0	<Any readable memory address>	
(n+y)*0x14 + 0x4	<SDMA SOFT_INT address>	4
(n+y)*0x14 + 0x8	0x1	5
(n+y)*0x14 + 0xC	0x0	6
(n+y)*0x14 + 0x10	0x0	Don’t care
Last_entry (option 4):		3
y=last entry number: (n+y-1)*0x14 + 0	<Any readable memory address>	
(n+y-1)*0x14 + 0x4	<SDMA SOFT_INT address>	4
(n+y-1)*0x14 + 0x8	0x1	5

ADDRESS	LABEL	REMARK #
(n+y-1)*0x14 + 0xC	<word transfer + Enable_next_table>	7
(n+y-1)*0x14 + 0x10	Linked_List_Base_address + (n+y)*0x14	Addr last entry
y=last entry number: (n+y)*0x14 + 0x0	Source address y	
(n+y)*0x14 + 0x4	Destination address y	
(n+y)n*0x14 + 0x8	Transfer Length y	
(n+y)*0x14 + 0xc	Configuration y	Disable companion table option
(n+y)*0x14 + 0x10	0x0	Don’t care

Remark 0: n = Channel entry number.
Remark 1: The configuration register can be programmed any way the programmers liks it, but should **always** have the ‘companion channel’ set to the next channel, or otherwise the linked-list execution stops. An exception in this case is when the programmer wants to stop the linked list execution (for instance in case of the last entry)
Remark 2: As many entries can be set here as the programmer would like, each entry consuming 20 bytes of local memory. Remember that every entry has 5 addresses, and that address 5 contains the address of the following entry.
Remark 3: This last entry will soft-interrupt the DMA controller, so the CPU knows the last transfer has finished. The source address is ‘dont care’, but preferably a memory mapped location (like SRAM).
Remark 4: The destination address is the soft-interrupt register of this SDMA controller.
Remark 5: Length ‘1’ is sufficient
Remark 6: This setting disables companion-channels, so basically this is the last transfer.
Remark 7: the ‘enable companion channel’ option must be set, since there is still1 transfer left.

Ending a linked-list transfer inside the linked list :

The last entry of a linked list is a special one. In the example linked-list, there are 4 methods described how to end the linked list transfer:

Method1) Just stop the linked list, by disabling the copy-table-enabling in the config setting.
This can be done for instance if the CPU is not interrested in an IRQ on this event.

Method 2) Re-start the linked list by pointing the last pointer back to the beginning of the linked list, effectively creating a circular buffer made out of linked list entries.

Method 3) Write towards the SDMA soft interrupt register, which enables the CPU to be interrupted, so that it can act on the event that a linked-list has ended.

Method 4) This is a combination of 1 and 3. Now the CPU gets a soft interrupt 1 entry before the last entry is executed.

Method x) Whatever suits the application.....

To start a linked-list operation:

* Program into memory the linked-list to <linked-list_base_address>.
* Reserve 2 sequential DMA channels/channels.
* Program the second channel as follows:

Source address	<Linked_list_base_address>
Destination address	<**The alternative source address** of the previous DMA channel>

Transfer Length	0x4 (for 5 addresses per entry)
Configuration	DMA_word + enable_companion_channel (previous channel)
Enable address	0x1 (start the transfer)

NOTE: Please do take care that the ALT_SOURCE_REG is used as the destination address, and not the normal SOURCE_REG

Ending a linked list transfer in the middle of a linked-list execution

There may be cases where the CPU needs to stop a linked-list from execution. There are more ways to do this:

1) Brutally stop the execution.

this can be done by:

* Enable [first_channel]=0
* Enable [second_channel]=0
* Enable [first_channel]=0

Now both channels are disabled, and the execution is stopped.

To continue, do the following:

```
if counter[first_channel] > 0 then
   Enable[first_channel] = 1
else
   Enable[second_channel] = 1
```

2) Stop the linked-list, but let the current entry finish first.

This can be done by read-modify-write the config[second_channel] register, and disable the ‘companion channel’ bit.

When this is done, the linked list entry will finish, and a new linked-list entry will be programmed in the previous channel. However, the first channel will not be enabled, thus stopping the transfer.

To continue, do the following:

* read-modify-write the config[second_channel] register, and enable the ‘companion channel’ bit.
* enable[first_channel] = 1

24.6 SDMA flow control

The SDMA flow control is compatible to the ARM flow control used in the ARM PL080 and PL081 DMA controllers.

Not supported by the SDMA is the ‘Burst Request’ and ‘Last Burst Request’ signal, if this signal indicates a burst of more than one word.

For information about the timing of the SDMA flow control, please refer to the ARM PL080 data sheet.

Is allowed for DMA slaves to assert an asyncronous request signal, either slower or faster then the SDMA clock. This is synchronised inside the SDMA. Please take care that the slave may receive an asynchronous ‘DMA_CLR’ signal as well, if there is an intention to use asynchronous clocks.

If a DMA slave is behind a write-buffered bridge or has its own write buffer, this DMA slave may only de-assert ‘SREQ’ when the transfer is truly completed. If the slave does not comply to this rule, then the write-buffered bridge must support the disabling of the write-buffer by reading the AHB-HPROT[2] signal.

PNX0101 connectivity:

Table 181PNX0101 SDMA connectivity

MODULE	DMA SOURCE	PERIPHERAL PIN *	INTERRUPT
MULTIMEDIA CARD INTERFACE	MCI single request MCI burst request	0 1	MCI single request indication
			MCI burst request indication
UART	UART rx uart tx	2 3	UART rx receive FIFO request information
			UART tx transmit FIFO request information
i2c	i2c	4	i2c FIFO DMA request
AUDIO SS		5	SAO1 Left channel DMA request
		6	SAO1 right channel DMA request
		7	SAO2 Left channel DMA request
		8	SAO2 Left channel DMA request
		9	SAI 1 left channel DMA request
		10	SAI 1 right channel DMA request
		11	SAI 2 left channel DMA request
		12	SAI 2 right channel DMA request
		13	SAI 3 left channel DMA request
		14	SAI 3 rightchannel DMA request
		15	SAI 4 left channel DMA request
		16	SAI 4 right channel DMA request
ssa1_lcd_interface	lcd tx	17	LCD interface FIFO transmit DMA request
MPMC_A19	Off-chip	18	Off-chip fifolevel-request at pin MPMC_A19
MPMC_A17	Off-chip	19	Off-chip fifolevel-request at pin MPMC_A17

* When programmed, add ‘0x1’ to this number.

Using FIFOlevel slaves, which have no flow control:

The SDMA can support such a slave, with the following rules:

- For a FIFOlevel slave which is read, the FIFOlevel must be updated directly after the read has taken place.
- For a FIFOlevel slave which is written, the FIFOlevel must be updated directly after the write has taken place.
  If a write-slave is behind a write-buffered bridge, then the write-buffered bridge must support the disabling of the write-buffer by reading the AHB-HPROT[2] signal.
  If this is not done, it is then possible that the SDMA sometimes performs two transfers to the slave in stead of one This may be acceptible for a slave, but it is then mandatory the FIFO of this slave can accept two transfers.
  The SDMA will never exceed its maximal transfer count.
- On top-level connectivity, the FIFOlevel request signal must be ANDed with the inverted SDMA_CLR[x] signal. This new signal is then connected to the same numbered SDMA_SREQ[x] pin.

24.7 External enable flow control

Other devices then the CPU can enable a SDMA channel, by activating one of the ‘Ext_Enable’ flow control pins.

The SDMA has an equal number of these pins available on the SDMA top-level, as there are channels available, and each external enable pin-number has a direct connection to its equivalent DMA channel number. The activation of one of these pins immediatly enables its corresponding DMA channel.

So:

Ext_en[0] can enable channel 0;
Ext_en[1] can enable channel 1;
Ext_en[2] can enable channel 2;
Ext_en[3] can enable channel 3;

Ext_en[4] can enable channel 4;
Ext_en[5] can enable channel 5;
Ext_en[6] can enable channel 6;
Ext_en[7] can enable channel 7;

Connections to SDMA EXT_EN pins:

EXT_EN	CONNECTED DEVICE
0	-
1	-
2	-
3	MPMC_A20
4	-
5	MPMC_A18
6	-
7	Epics7B

It uses the following flow control, using Ext_en[7] as an example:

- Ext_en[7] is asserted by the EPICS7B
- The SDMA starts channel 7 because of this assertion.
- Ext_en[7] must remain asserted until Ext_en_ack[7] is asserted by the SDMA.
- The SDMA asserts Ext_en_ack[7], when channel 7 is finished with all its transfers.
- Ext_en[7] can be de-asserted.
- The SDMA will de-assert Ext_en_ack[7] because of this.
- Another transfer is allowed, and the flow may be restarted.

A SDMA channel can only be enabled by this method, not disabled.
If 'Ext_en[7]' is asserted, the SDMA will only enable channel 7 once. If the transfer is complete, and Ext_en[x] remains asserted, channel 7 will remain inactive. So A toggling is required on Ext_en[7] before multiple enablings are possible, and this has to be done according the Ext_en flow control as described above.

It is allowed to have the 'Ext_en' signals on different clock domains, so the EPICS can run faster or slower then the SDMA

The 'Ext_en_ack' of a channel is always asserted for at least a single clock-cycle if that channel is finished, disregarding if Ext_en is asserted or not. Any peripheral can recognise this way if a channel is finished.

24.8 Differences between the N1A and N1C regarding the SDMA

1) Fixed 2 non-compliences:

- The SDMA can now use the PL172 for ATA and other ISA like devices.

  - With the N1A, each read from the PL172 resulted in two reads on the external memory bus. The first read is the correct read, the second a pre-emptive read. So correct data was returned, but external devices that are read-strobe based, 'saw' nota single read but two reads.
  With the N1C, only one access is performed, which is correct behaviour

- Soft interrupt was not functional. Fixed for N1C.

2) Added two flow-control signals

Two external devices connected to the PNX0101 are now capable controlling the flow control of the SDMA.
Fifo-level flow control is used, which means that as long as the extenal flow control signal stays active 'high' the SDMA will continue sending data towards the same address, just like an internal fifo-level does (For instance the LCD_INTERFACE).

To be able to use the flow control, the MPMC_A19 or MPMC_A17 pin must be set as GPIO-input.
See the SDMA flow control connections table.

3) Added two off-chip external enable signals

Two external devices are capable of enabling a DMA channel.
* Pin MPMC_A20 is capable of enabling channel 3
* Pin MPMC_A18 is capable of enabling channel 5

If this functionality is to be used, two things need to be done:

* Set the pins you want to use as ‘GPIO-input’
* Enable the correct external-enable pin the the ‘syscreg’ block. The register is called:
    - SYSCREG_SYS_CREG_SDMA_EXT_EN_3 or
    - SYSCREG_SYS_CREG_SDMA_EXT_EN_5

25 TIMER MODULE

25.1 Overview

The timer block contains two fully independant timers, where each timer has its own clock and chip-select. Each timer is a 32 bit wide down-counter with selectable pre-scalers. The pre-scaler allows either the system clock to be used directly, or the clock divided by 16 or 256 may be used. This is provided by 0, 4 or 8 stages of pre-scale. Two modes of operation are available, free-running and periodic timer. In periodic timer mode the counter will generate an interrupt at a constant interval. In free-running mode the timer will overflow after reaching its zero value and continue to count down from the maximum value.

The timer speed depends on the system clock. The system clock can change depending the operation mode, which means that the timer can not be used as a real time clock!

25.2 Functional description

The timer is loaded by writing to the Load register and then, if enabled, the timer will count down to zero. On reaching a count of zero an interrupt will be generated. The interrupt may be cleared by writing to the Clear register.

After reaching a zero count, if the timer is operating in free-running mode then the timer will continue to decrement from its maximum value. If periodic timer mode is selected then the timer will reload from the Load register and continue to decrement. In this mode the timer will effectively generate a periodic interrupt. The mode is selected by a bit in the Control register.

At any point the current timer value may be read from the Value register.

At any point the timer_load may be re-written. This will cause the timer to restart to the timer_load value.

the timer is enabled by a bit in the control register. At reset the timer will be disabled, the interrupt will be cleared and the Load register will be undefined. The mode and pre-scale value will also be undefined.

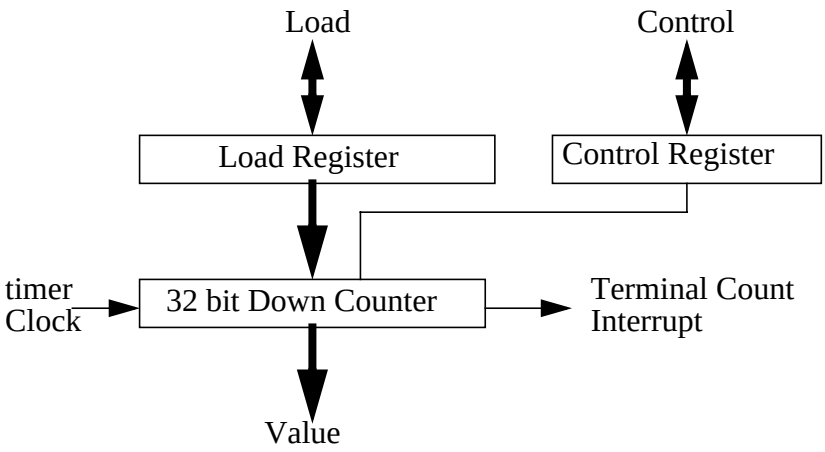


Fig. 32 Timer Block Diagram

The timer clock is generated by a pre-scale unit. The timer clock may be the system clock, the system clock divided by 16, which is generated by 4 bits of pre-scale, or the system clock divided by 256, which is generated by a total of 8 bits of pre-scale.

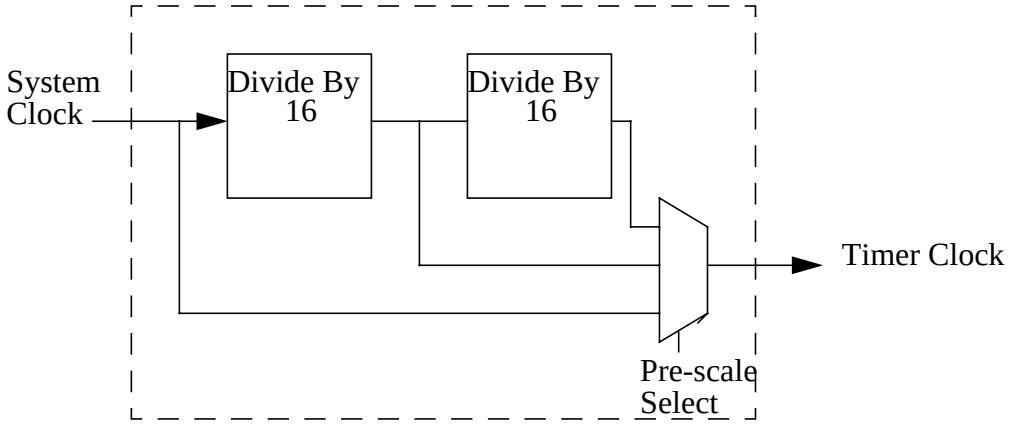


Fig. 33 Timer Pre-scape Unit

25.3 Registers

25.3.1 LOAD REGISTER

Read/write. The Load register contains the initial 32-bit value of the timer and is also used as the reload value in periodic timer mode.

Writing to the Load register results in a restart of the timer to the timer_load value.

25.3.2 VALUE REGISTER

Read only. The Value location gives the current 32- bit value of the timer.

25.3.3 CLEAR REGISTER

Write only. Writing to the Clear location clears the interrupt generated by the counter timer.

25.3.4 CONTROL REGISTER

The Control register provides enable/disable, mode and pre-scale configurations for the timer.

31 8	7	6	5	4	3 2	1	0
X	Enabe	Mode	X	X	Pre-scale	X	X

Bit 7 - Enable Bit 0-Timer Disabled
1-Timer Enabled

Bit 6 - Mode bit 0-Free running Model
1-Periodic Timer Model

Bits 3-2: Pre-scale Bits See Table 182

Table 182 Bits 3-2: Pre-scale bits

BIT 3	BIT 2	CLOCK DIVIDED BY	STAGES OF PRE-SCALE
0	0	1	0
0	1	16	4
1	0	256	8
1	1	Undefined	

Bits 31-8,4,1-0: Undefined Must be written as zero
Read as undefined

25.3.5 TEST REGISTER

25.3.6 TIMER MEMORY MAP

The base address of the Timers is not fixed and may be different for any particular system implementation. However, the offset of any particular register from the base address is fixed.

Table 183 Timer Register Map

ADDRESS	READ LOCATION	WRITE LOCATION
Timerbase0 = 0x8002_0000	Timer0Load	Timer0Load
Timerbase0+0x04	Timer0Value	Reserved
Timerbase0+0x08	Timer0Control	Timer0Control
Timerbase0+0x0C	Reserved	Timer0Clear
Timerbase0+0x10	Timer0Test	Timer0Test
Timerbase1 = 0x8002_0400	Timer1Load	Timer1Load
Timerbase1+0x04	Timer1Value	Reserved
Timerbase1+0x08	Timer1Control	Timer1Control
Timerbase1+0x0C	Reserved	Timer1Clear
Timerbase1+0x10	Timer1Test	Timer1Test

25.4 Interrupts

The timer is loaded by writing to the Load register and then, if enabled, the timer will count down to zero. On reaching a count of zero an interrupt will be generated. The interrupt may be cleared by writing to the Clear register

26 WATCHDOG TIMER

26.1 Overview

Once the watchdog is enabled, it will monitor the programmed timeout period and generates a reset request when the period expires. In normal operation the watchdog is triggered periodically, resetting the watchdog counter and ensuring that no reset is generated. In the event of a software or hardware failure preventing the CPU from triggering the watchdog, the timeout period will be exceeded and a reset requested from the CGU.

26.2 Configuration

Fig.33 on page 276 shows how the Watchdog Timer is located in the PNX0101. The interrupt pin (nint) of this Watchdog Timer is not connected to the Interrupt Controller. Instead of this, two pins m0 and m1 (see Fig.33 on page 276) are used which will generate events. Pins m0 an m1 will generate events when their Match Registers matches the Timer Counter Register (TC).

The Watchdog Timer in the PNX0101 can be used as follows):

- As watchdog
  The m1 output is used for generating an event to the CGU, which requests a reset.
- As timer.
  The m0 output is used for generating an event to the Event Router, which generates an interrupt to the Interrupt Controller. Note that the latency between the occurence of an event at pin m0 and when the IRQ or FIQ will be asserted due this event, will be longer than when the Timer Modules are used. Because the interrupts generated by the Timer Modules are directly connected to the Interrupt Controller, while the event of the Watchdog Timer goes via the Event Controller
- As watchdog and as timer.
  The value of the MCR0 (Match Register 0) has to be a lower than the value of MCR1. Otherwise not desired resets will be generated by the CGU.

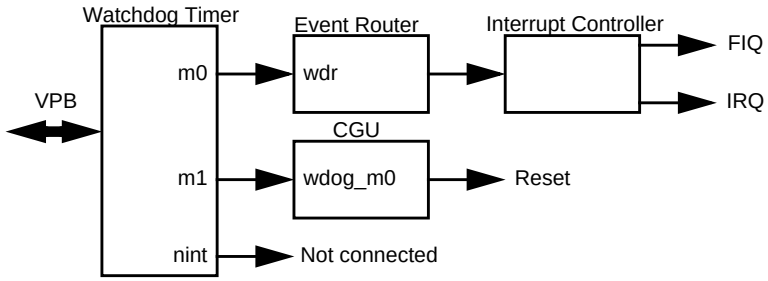


Fig.33 Watchdog timer in PNX0101

26.3 Register map

In table 184 on page 277 is the address map shown of the Watchdog Timer.

Table 184 Watchdog Register Map

ADDRESS	REGISTER NAME	DESCRIPTION
0x00	Interrupt Register (IR)	The IR can be written to clear interrupts. The IR can be read to identify which of two possible interrupt sources are generating an interrupt. Note that nint signal of Watchdog Timer is not connected to Interrupt Controller.
0x04	Timer Control Register (TCR)	The TCR is used to control the Timer Counter and Prescale Counter functions. The Timer Counter and Prescale Counter can be disabled or reset through the TCR.
0x08	Timer Counter (TC)	The TC is incremented every PR+1 cycles of pclk. If the PR is not included then the TC is incremented every cycle of pclk. The TC is controlled through the TCR.
0x0C	Prescale Register (PR)	This Prescale Register (PR) is an 32 bits register. The PR allows the user to specify that the TC be incremented every PR+1 cycles of pclk.
0x10		Reserved
0x14	Match Control Register (MCR)	The optional MCR is used to control if an interrupt is generated and if the TC is reset when one of the Match Registers matches the value in the TC.
0x18	Match Register 0 (MR0)	The optional MR0 can be enabled through MCR to reset the TC, stop both the TC and PC, and/or generate an interrupt every time MR0 matches that of the TC.
0x1C	Match Register 1 (MR1)	(See MR0)
0x20 till Watchdogbase+0x38		Reserved
0x3C	External Match Register (EMR)	The optional EMR contains control and status bits related to the external match pins m(0-1). EMR[7:4] is used to determine how EMR[1:0] and m(0-1) change when a match occurs.

26.4 Register descriptions

26.4.1 INTERRUPT REGISTER (IR)

The Interrupt Register consists of up to eight bits, four for the interrupts on the Match Register matches and four for the interrupts on capture events. If an interrupt is being generated then the corresponding bit in the IR will be one. Otherwise, the bit will be zero. Writing a logic one to the corre-sponding IR bit will reset the interrupt. Writing a zero has no effect. Writing a one instead of a zero allows the user to write the contents of the Interrupt Register to itself thus providing a quick method of clearing. Refer to table 185 on page 278 for the mapping of this register.The function of each of the bits of the IR is shown in table 186 on page 278.

An interrupt is generated if one of the Match Registers matches the contents of the Timer Counter. This interrupt signal is captured on the rising edge of pclk. The output of the flip-flop is fed back to itself to hold the interrupt at one until cleared by the user. The feed back is broken and the interrupt cleared when the user writes a one into the appropriate bit of the

Interrupt Register. The logical NOR of each of the four possible match generated interrupts along with the four possible capture interrupts is used to generate the external nint signal. Note that the nint signal of the Watchdog Timer is not connected to the Interrupt Controller.

Table 185 Interrupt Register Map

Bit	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**
Name	-	-	-	-	-	-	-	-
POR	-	-	-	-	-	-	-	-
Bit	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Name	-	-	-	-	-	-	-	-
POR	-	-	-	-	-	-	-	-
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Name	-	-	-	-	-	-	-	-
POR	-	-	-	-	-	-	-	-
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	reserved	reserved	reserved	reserved	reserved	reserved	intr_m1	intr_m0
POR	0	0	0	0	0	0	0	0

Table 186 Interrupt Register (IR)

BIT	DISCRIPTION
intr_m0	Interrupt bit for a MR0 and TC match.
intr_m1	Interrupt bit for a MR1 and TC match.

26.4.2 TIMER CONTROL REGISTER (TCR)

The Timer Control Register maintains 3-bits which control the Timer. The Timer Enable of the TCR gates the Counter Enable (CE) pin of the Timer Counter. The Reset bit of the TCR synchronously resets the TC at the rising edge of pclk.

TCR is in table 187 on page 279. The function of each of the bits of the TCR is shown in table 188 on page 279.

Table 187 Timer Control Register Map

Bit	31	30	29	28	27	26	25	24
Name	-	-	-	-	-	-	-	-
POR	-	-	-	-	-	-	-	-
Bit	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Name	-	-	-	-	-	-	-	-
POR	-	-	-	-	-	-	-	-
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Name	-	-	-	-	-	-	-	-
POR	-	-	-	-	-	-	-	-
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	-	-	-	-	-	reserved	reset	enable
POR	-	-	-	-	-	0	0	0

Table 188 Timer Control Register (TCR)

BIT	DISCRIPTION	
enable	Counter Enable	When one, the Timer Counter is enabled for counting. When zero, the counters are disabled.
reset	Counter Reset	When one, the Timer Counter is synchronously reset on the next positive edge of pclk. The counters remain reset until TCR[1] is brought back to zero.

26.4.3 TIMER COUNTER (TC)

The Timer Counter is an HDLi compiled counter. The counter size is 32 bits. The counter is clocked on the rising edge of the bus clock (pclk). The Count Enable (CE) pin of the TC is enabled through the Timer Control Register (TCR). The counter counts in modulo 2 n fashion. A maximum value other than 2 n -1can be specified by using the Reset on Match option. This option would allow the counter to count to a value specified in the Match Register. When a match occurs the TC is synchronously reset when the PC's terminal count indicator enables the TC. TC can also be synchronously reset by the user by setting the reset bit in the TCR. The TC is therefore reset on the next rising edge of pclk. The Power On Reset (POR) value is zero for all bits.

26.4.4 PRESCALE REGISTER (PR)

The Prescale Register (PR) is an 32 bits register. The PR allows the user to specify that the TC be incremented every PR+1 cycles of pclk.

26.4.5 MATCH REGISTERS AND MATCH CONTROL REGISTER (MRX AND MCR)

The counter block can be configured to have up to four Match Registers. Each Match Register is the size of the Timer Counter. Each Match Register can be configured through the Match Control Register to stop both the Timer Counter and Prescale Counter thus maintaining their value at the time of the match, restart the Timer Counter at zero, allow the counters to continue counting, and/or generate an interrupt when its contents match those of the TC. When MRx=TC, Reset on MRx is enabled through the MCR, and the TC is enabled through the TCR and the Prescale Counter's TCI pin, then the Timer Counter is reset on rising edge of pclk. It should be noted that stop on match has higher priority than reset on match. The POR value for all Match Registers MR(1-0) is zero. Register map detail for the MCR is listed in table 189 on page 280. The function of each of the bits in the MCR is shown in table 190 on page 280.

Table 189 Match Control Register Map

Bit	31	30	29	28	27	26	25	24
Name	-	-	-	-	-	-	-	-
POR	-	-	-	-	-	-	-	-
Bit	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Name	-	-	-	-	-	-	-	-
POR	-	-	-	-	-	-	-	-
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Name	-	-	-	-	reserved	reserved	reserved	reserved
POR	-	-	-	-	0	0	0	0
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	reserved	reserved	stop_1	reset_1	intr_1	stop_0	reset_0	intr_0
POR	0	0	0	0	0	0	0	0

Table 190 Match Control Register

BIT	DISCRIPTION	
intr_0	Interrupt on MR0	When one, an interrupt is generated when MR0 matches the value in the TC. When zero this interrupt feature is disabled.
reset_0	Reset on MR0	When one, the TC will be synchronously reset if MR0 matches TC. When zero this feature is disabled.
stop_0	Stop on MR0	When one, the TC and PC will stop counting and TCR[0] will be set to 0 if MR0 matches TC. When zero this feature is disabled.
intr_1	Interrupt on MR1	When one, an interrupt is generated when MR1 matches the value in the TC. When zero this interrupt feature is disabled.
reset_1	Reset on MR1	When one, the TC will be synchronously reset if MR1 matches the TC. When zero this feature is disabled.
stop_1	Stop on MR1	When one, the TC and PC will stop counting and TCR[0] will be set to 0 if MR1 matches TC. When zero this feature is disabled

26.4.6 External Match Register (EMR)

The External Match Register is provided when the user has compiled the timer with match registers and external match outputs.This register pro-vides both control and status of the external match pins M(0-3). EMR[0:3] and M(0-3) can either toggle, go to zero, go to one, or maintain state when the contents of MRx is equal to the contents of TC. EMR[4:11] are used to specify the action taken by EMR[0:3] and M(0-3). Clearing M(0-3) can be done by writing directly to EMR. Register map detail is in table 191 on page 281. The functionality of each of the EMR bits is shown in table 192 on page 281.

Table 191 External Match Register Map

Bit	31	30	29	28	27	26	25	24
Name	-	-	-	-	-	-	-	-
POR	-	-	-	-	-	-	-	-
Bit	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Name	-	-	-	-	-	-	-	-
POR	-	-	-	-	-	-	-	-
Bit	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Name	-	-	-	-	reserved	reserved	reserved	reserved
POR	-	-	-	-	0	0	0	0
Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	ctrl_1[1]	ctrl_1[0]	ctrl_0[1]	ctrl_0[0]	reserved	reserved	emr_1	emr_0
POR	0	0	0	0	0	0	0	0

Table 192 External Match Register

BIT	DISCRIPTION	
emr_0	External Match 0	External Match 0- When match register 0 (MR0) equals the timer counter (TC) this output can either toggle, go to zero, go to one, or do nothing. EMR[4:5] controls the functionality of this output.
emr_1	External Match 1	External Match 1- When match register 1 (MR1) equals the timer counter (TC) this output can either tog-gle, go to zero, go to one, or do nothing. EMR[6:7] con-trols the functionality of this output.
ctrl_0[0]	External Match Control 0	External Match Determines the functionality of External Match 0, table 193 on page 281 shows the decode of these pins.
ctrl_0[1]		
ctrl_1[0]	External Match Control 1	Determines the functionality of External Match 1, table 193 on page 281 shows the decode of these pins.
ctrl_1[1]		
ctrl_3[1]		

Table 193 External Match Control

BIT	CTRL_X[0]	DISCRIPTION
0	0	Do Nothing
0	1	Set LOW
1	0	Set HIGH
1	1	Toggle

27 REAL TIME CLOCK (RTC)

27.1 Overview

The Real-Time Clock (RTC) module consists of a counter which increments at a frequency of typically 32.768kHz. The RTC provide a set of counters to measure time during power on and power off operation. It is designed to use little power consumption in power down mode.

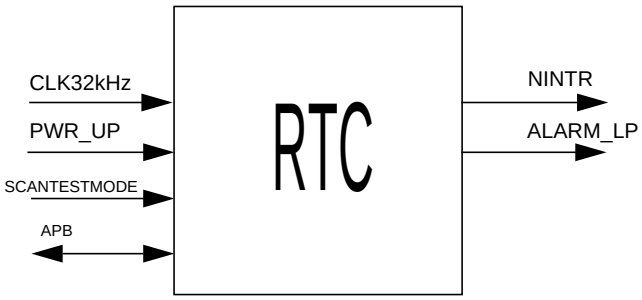


Fig. 34 RTC Block Diagram

Features:

- Measures the passage of time to maintain a calendar and clock
- Conforms to AMBA APB interface specifications
- Ultra low power design to support battery operation
- Date counters
- Leap year calculation
- Feature to allow programmable periodic interrupts every time a selected counter increments
- Alarm clock feature allows the user to specify a date and time for an interrupt
- Testmode to facilitate parallel testing

27.2 Functional Description

The RTC interfaces to a standard Advanced Peripheral Bus (APB) with a unidirectional data bus. The data bus is 32-bits wide while the consolidated time registers are included to read all time counters with only three read operations. The non-APB signals are used for other purposes like interrupt output (nintr), power indicator (pwr_up), the 32.768 kHz clock and the low-power alarm indicator (alarm_lp).

Note that the signal to the pwr_up pin is controllable in the Syscreg block. See the ssa1_rtc_cfg_pwr_up bit of the SYSCREG_SSA1_RTC_CFG register of the Syscreg block.

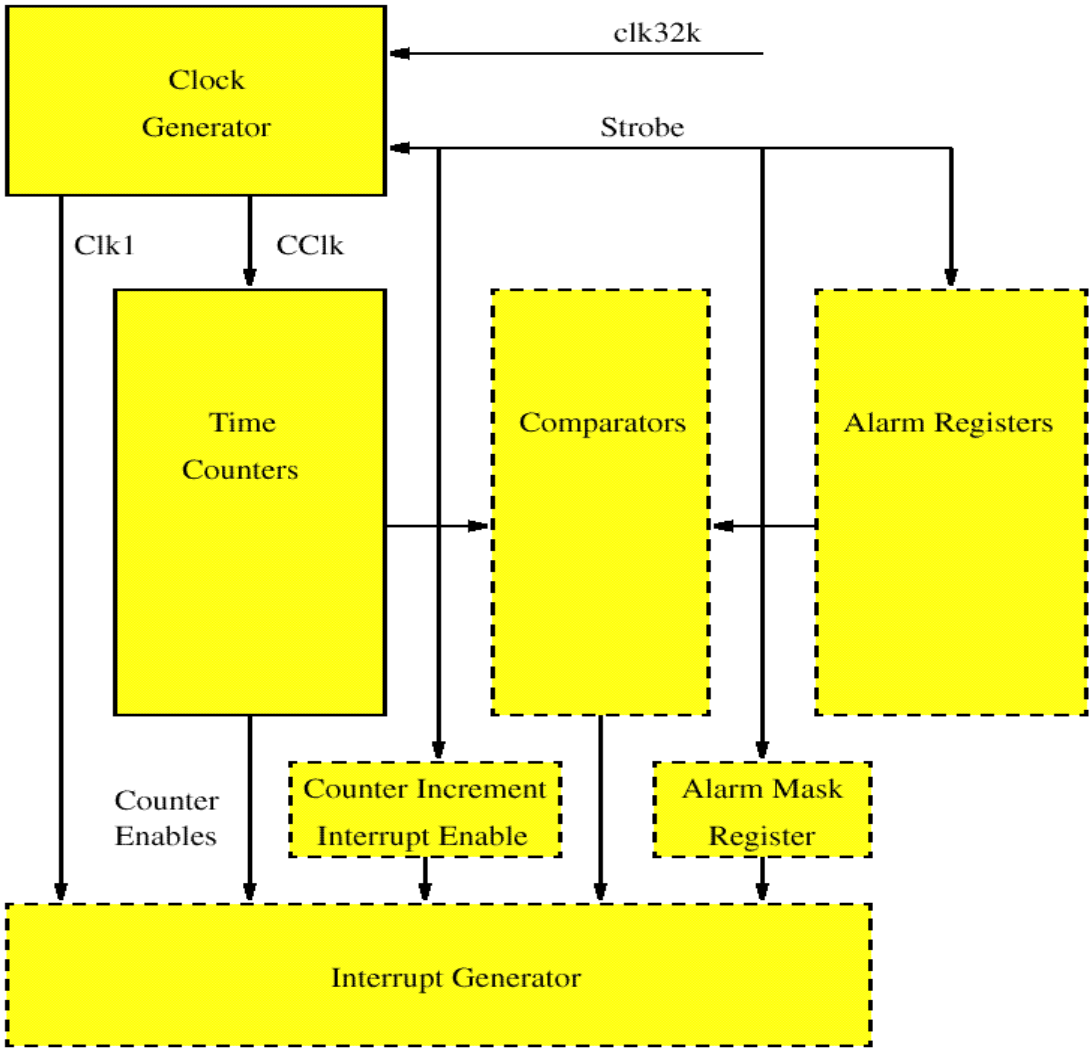


Fig. 35 RTC architecture

The RTC uses a 32.768 kHz clock (clk32k), that is divided down to a 1 Hz clock using a ripple counter (see Fig. 36). A ripple counter is used to minimize power during power down mode. If bit 2 is set in the RTC Clock Register (CR[2]), the ripple counter is divided into three identical 5-bit ripple counters clocked off of the 32.768 kHz system clock. CR[1] of the RTC Clock Register is tied to the asynchronous resets of each of the flip-flops. Driving CR[1] high will reset the flip-flops in the ripple counter, while driving CR[1] low will re-enable the ripple counters. CR[0] allows the clock divider to be disabled so the time counters can be written.

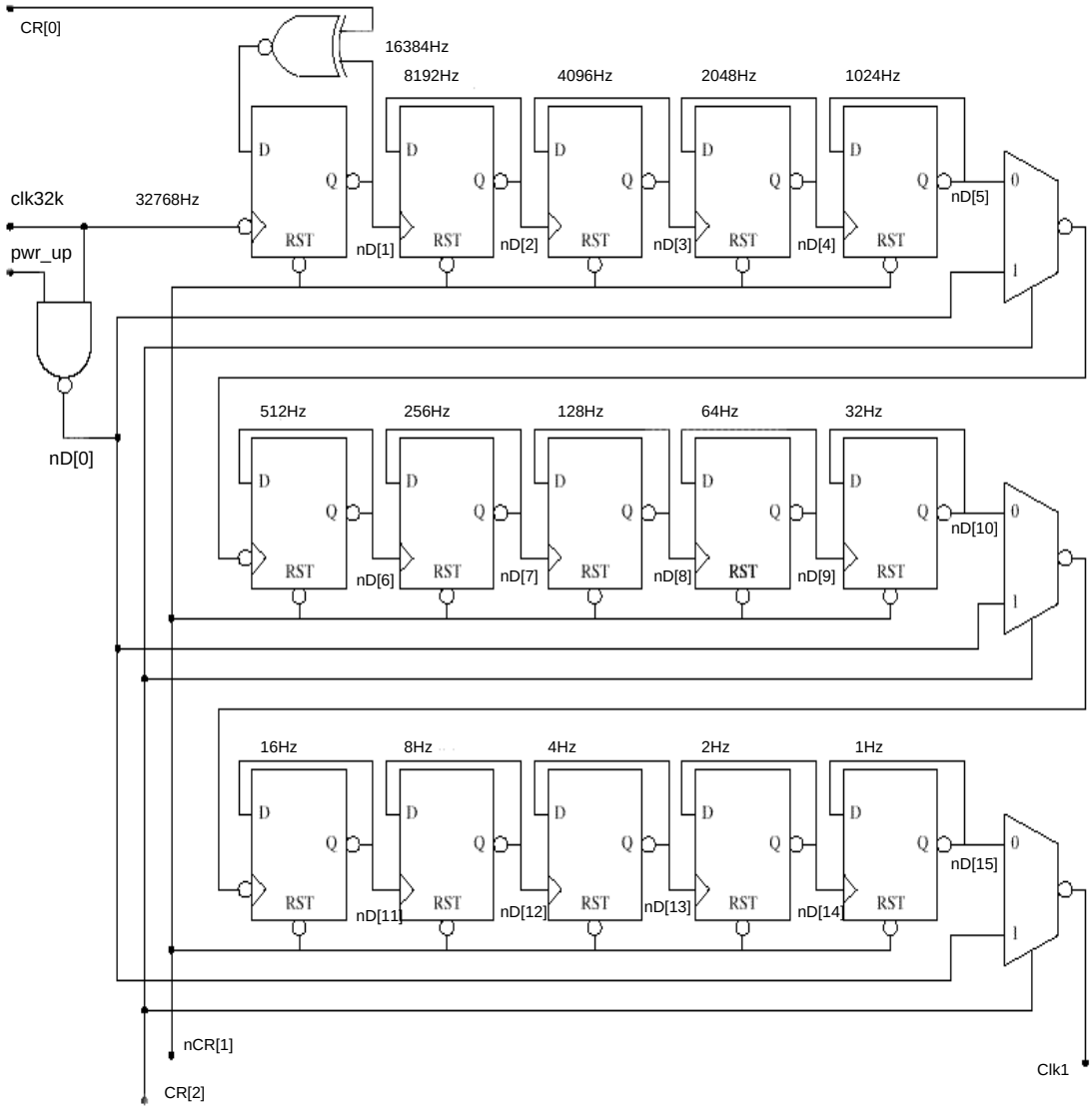


Fig. 36 Clk1 generation

The counter clock (CClk) consists of the exclusive-or of the 1 Hz clock and the counter write strobe. During a non-write operation the counters operate as a set of sequential counters clocked by the 1 Hz clock. During a write operation no event is allowed on the 1 Hz clock. To insure this condition is true the user should disable the 1 Hz clock by setting the clock enable bit (CR[0]) to zero and setting the pre scalar reset bit (CR[1]) to zero before writing to the RTC.

There are two sources that clock the counters. The first source is Clk1, the 1 Hz clock. The second source is PENABLE during write operations. This is implemented by XORing Clk1 with a signal, CWStrobe, which indicates a bus write operation to the counters. CClk, the counter clock, is either Strobe or Strobe inverted depending on if Clk1 was stopped during its low or high phase. The effect is, depending on the state Clk1, that the new counter value is either loaded on the rising or falling edge of Strobe. The CEN pin of each counter is kept low during the counter write cycle (PWRITE, PSEL, pwr_up, and PADDR), so that the CWStrobe does not increment any of the counters.

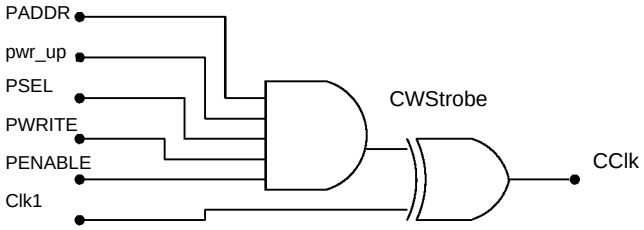


Fig. 37 Time Counter clock generation

Each counter has its count enable (CEN) gated so that during a counter write operation no counter is increment by the clock pulse generated by the write strobe. Two of the counters have dynamic maximum values, the Day of Month counter and the Day of Year counter. These maximum values are determined via combinational logic whose inputs are the Year counter (for leap year) and Month counter.

For determining a leap year, the RTC does a simple bit comparison to see if the two lowest order bits of the year counter are zero. If true, then the RTC considers that year a leap year. A more accurate algorithm would prevent years evenly divisible by 100, but not evenly divisible by 400, from being leap years (the year 2000 is a leap year, but 2100 is not). The RTC considers all years evenly divisible by 4 as a leap year. This algorithm will be accurate until the year 2100.

27.3 Registers

The address map for a Real Time Clock with all features enabled is shown in table 194 on page 286. Note that there are several unused locations. The address space is split into three sections by functionality. The first eight addresses reference Miscellaneous Registers. The next eight locations are the Consolidated Time Registers. The last eight contain the Alarm Registers. If registers or bits are not used, writes to the bit or register are ignored.

Table 194 RTC register map

ADDRESS SPACE	OFFSET	REGISTER NAME	R/W ACCESS	RESET
0x8000_2000 - 0x8000_23FF	0x00	RTC_INT_LOC	R	-
		RTC_INT_CLEAR	W	-
	0x04	RTC_PRESCALE	R	-
	0x08	RTC_CLK_REG	R/W	-
	0x0C	RTC_INC_INT	R/W	-
	0x10	RTC_ALARM_MASK	R/W	-
	0x14	RTC_CTIME0	R	-
	0x18	RTC_CTIME1	R	-
	0x1C	RTC_CTIME2	R	-
	0x20	RTC_SECONDS	R/W	-
	0x24	RTC_MINUTES	R/W	-
	0x28	RTC_HOURS	R/W	-
	0x2C	RTC_DayOfMonth	R/W	-
	0x30	RTC_DayOfWeek	R/W	-
	0x34	RTC_DayOfYear	R/W	-
	0x38	RTC_MONTHS	R/W	-
	0x3C	RTC_YEARS	R/W	-
	0x60	RTC_SEC_ALARM	R/W	-
	0x64	RTC_MIN_ALARM	R/W	-
	0x68	RTC_HRS_ALARM	R/W	-
	0x6C	RTC_DOM_ALARM	R/W	-
	0x70	RTC_DOW_ALARM	R/W	-
	0x74	RTC_DOY_ALARM	R/W	-
	0x78	RTC_MONTHS_ALARM	R/W	-
	0x7C	RTC_YEARS_ALARM	R/W	-

27.3.1 RTC_INT_LOC REGISTER

This register (see table 195) contains the RTC Interrupt Location variables. This register specifies which blocks are generating an interrupt. Writing a one to the appropriate bits clears the corresponding interrupt. Writing a zero has no effect. This allows the programmer to read this register and write back the same value to clear only the interrupt that is detected by the read.

Table 195 RTC_INT_LOC register

BITS	VARIABLE	RESET	R/W
0	Counter Increment Interrupt (ILR[0])	-	R/W
1	Alarm Registers Interrupt (ILR[1])	-	R/W
31:2	Reserved	-	-

ILR[0] When high the Counter Increment Interrupt block generated an interrupt. Writing a one to this bit location clears the counter increment interrupt.

ILR[1] When high the alarm registers generated an interrupt. Writing a one to this bit location clears the alarm interrupt.

27.3.2 RTC_PRESCALE register

This register (see table 196) contains the RTC prescaler variables. It can be reset to zero through the Clock Register (CR). The prescaler consists of the bits of the clock divider ripple counter. the 16-bit value divided by 2^{16} gives the fractional part of the second. The pre-scale counter is read-only.

Table 196 RTC_PRESCALE register

BITS	VARIABLE	RESET	R/W
15:0	Pre-scale value	-	R
31:16	Reserved	-	-

27.3.3 RTC_CLK_REG register

The clock register is a 4-bit register that controls the operation of the clock divide circuit. Each bit is described in table 197.

Table 197 RTC_CLK_REG register

BITS	VARIABLE	RESET	R/W
0	Clock Enable (CR[0])	-	R/W
1	Prescaler Reset (CR[1])	-	R/W
2	Test Enable CR[2];CR[3]	-	R/W
3		-	R/W
31:4	Reserved	-	-

Clock Enable When the value is high the time counters are enabled. When low, they are disabled

Prescaler Reset When high the elements in the clock divider are reset. The elements remain reset until this bit is brought low. When Time Counter Registers are set, this bit must be high.

Test Enable These bits should always be set low during normal operation.

27.3.4 RTC_INC_INT register

The Counter Increment Interrupt Register (CIIR) gives the user the ability to generate an interrupt every time a counter is increment. This interrupt remains valid until cleared by writing a one to bit zero of the Interrupt Location Register (ILR[0]).

Table 198 RTC_INC_INT register

BITS	VARIABLE	RESET	R/W
0	Second	-	R/W
1	Minute	-	R/W
2	Hour	-	R/W
3	Day of Month	-	R/W
4	Day of Week	-	R/W
5	Day of Year	-	R/W
6	Month	-	R/W
7	Year	-	R/W
31:8	Reserved	-	-

27.3.5 RTC_ALARM_MASK register

The Alarm Mask Register (AMR) allows the user to mask any of the alarm registers. Table 199 shows the relationship between the bits in the AMR and the alarms. For the alarm function, every non-masked alarm register must match the corresponding time counter for an interrupt to be generated. The interrupt is generated only when the counter comparison first changes from no-match to match. The interrupt is removed when a one is written to the appropriate bit of the Interrupt Clear register. If all mask bits are set, then the alarm is disabled.

Table 199 RTC_ALARM_MASK register

BITS	VARIABLE	RESET	R/W
0	Second	-	R/W
1	Minute	-	R/W
2	Hour	-	R/W
3	Day of Month	-	R/W
4	Day of Week	-	R/W
5	Day of Year	-	R/W
6	Month	-	R/W
7	Year	-	R/W
31:8	Reserved	-	-

27.3.6 Consolidated Time registers

The values of the Time Counters can be read in a consolidated format which allows the programmer to read all time counters with only three read operations. The various registers are packed into 32-bit values as shown in table 200 on page 289. The least significant bit of each register is read back at bit 0, 8,16 or 24. Unused bits are read back as zero.

Table 200 Counter Read locations

ADDRESS	VARIABLE	D[31:24]	D[23:16]	D[15:8]	D[7:0]
0x14	CTIME0	Day of Week	Hours	Minutes	Seconds
0x18	CTIME1	Year		Month	Day of Month
0x1C	CTIME2	Day of Year			

27.3.7 Time Counter registers

The RTC can consist of the eight counters shown in table 201. These counters can be read or written at the locations shown in table 194 on page 286.

Table 201 Time Counter register

ADDRESS	VARIABLE	SIZE	ENABLED BY	MIN VALUE	MAX VALUE
0x20	Second	6	1 Hz clock	0	59
0x24	Minute	6	Second	0	59
0x28	Hour	5	Minute	0	23
0x2C	Day of Month	5	Hour	1	28,29,30 or 31
0x30	Day of Week	3	Hour	0	6
0x34	Day of Year	9	Hour	1	365 or 366 (with leap yr. support)
0x38	Month	4	Day of Month	1	12
0x3C	Year	12	Month or Day of Year	0	4095

27.3.8 Alarm registers

The alarm registers are shown in table 194 on page 286. The values in these registers are compared with the time counters. If all unmasked (section 27.3.5 “RTC_ALARM_MASK register”) alarm registers match their corresponding time counters then an interrupt is generated. The interrupt is cleared when a one is written to bit one of the Interrupt Location Register (ILR[1]).

27.4 Interrupts

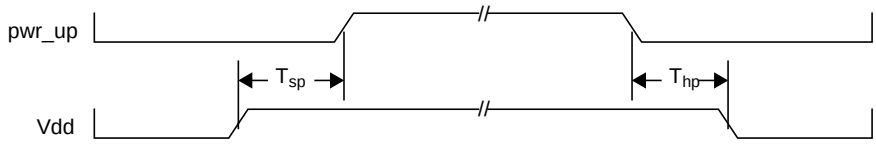
Interrupt generation is controlled through the Counter Increment Interrupt Register (CIIR), the alarm registers, and the Alarm Mask register (AMR). Interrupts are generated only by the transition into the interrupt state. Each bit in CIIR corresponds to one of the time counters. If CIIR is enabled for a particular counter, then every time the counter is increment an interrupt is generated. The alarm registers allow the user to specify a date and time for an interrupt to be generated, if alarm interrupt is generated also a Counter Increment Interrupt is generated. The AMR provides a mechanism to mask alarm compares. If all non-masked alarm registers match the value in their corresponding time counter, then an interrupt is generated.

27.5 Power Down operation

When the external signal pwr_up is low the RTC goes into power down mode. In power down mode all bus interface inputs are gated. Besides the first element in the ripple counter, and the optional alarm clock sampling flip flop, all loads to the 32.768 KHz clock are gated to reduce power.

The user can optionally specify that the alarm compare interrupt output should remain active in power down mode to allow for a power-on timer. When powered down, the synchronizing clock for alarm comparison will be the 1 Hz clock. When powered up, the bus clock is used to synchronize the alarm.

To guarantee proper operation without data loss, the pwr_up signal should not be set high until the bus has stabilized. Similarly the pwr_up signal should be taken low before the bus de-stabilizes on power down. A timing diagram is shown in Fig.38 on page 290 .



T_{sp} Minimum time the bus interface should be powered before pwr_up can be brought high
T_{hp} Minimum time pwr_up can be brought low before system power is removed.

Fig. 38 Power up requirements

28 10 BIT ADC

28.1 Overview

The 10 bitsis a successive approximation analog to digital converter with an 5 way input mux.

Features:

- Eight input channels selected through an analog multiplexer using sel[2:0] signal.
- From 2 to 10 bits resolution conversion, selected on the fly by means of strobe signal st.
- Maximum conversion rate 400k samples/s for 10 bits resolution.
- Power down mode performing minimal power dissipation.
- Internal power management to switch off unused circuitry between conversion cycles.
- No start-up cycles, no power down / power up recovery time.
- Muxed reference voltage inputs.

28.2 Functional description

This ADC is able to convert one of its 5 analog input signals with a conversion rate of 400 kSamples/s for a 10 bits resolution. However the resolution can be reduced to 2 bits and the conversion rate increased to 1500 kSamples/s. The ADC is composed of an analog mux 5:1 to select the input to convert, two 2:1 analog muxes to select the reference pair, a resistor string DAC and a successive approximation register. The selected reference pair is connected to the DAC which is controlled by the successive approximation register (SAR). One 10 bits conversion needs 11 clock cycles. During the first cycle, the selected input signal is sampled then the successive approximation determines the output code during the last 10 clock cycles.

28.3 Specification

In table 211 the general characteristics of the ADC module are shown. The specifications of the ADC are shown in table 202.

This module may be irreparably damaged if taken outside the specified absolute maximum rating range:

- Digital Supply Voltage (vdd): -0.5 to 2.0 Vdc
- Analog supply voltage (vdda): -0.5 to 3.6 Vdc
- Analog input voltage: -0.5 to 3.6 V
- Storage Temperature: -65 ^{o}C to +150 ^{o}C
- Max Junction Temperature: + 150 ^{o}C

Table 202 Specifications of the ADC

PARAMETER	SYMBOL	MIN.	TYP.	MAX.	UNIT
operation conditions					
junction temperature	Tjunc	-40	25	125	C
analog supply voltage	vdda	2.5	3.3	3.6	V
digital supply	vdd!	1.6	1.8	2.0	V
substate noise				150	mVpp
vrefn range	vrefn	vssa		vrefp-2	V
vrefp range	vrefp	vrefn+2		vdda	V
input impedance vrefp to vrefn		20		39	kOhms
input range	vin	vrefn		vrefp	V

PARAMETER	SYMBOL	MIN.	TYP.	MAX.	UNIT
clock frequency	Fclk			4.5	MHz
general					
total current consumption on vdda	Iact			400	uA
total power down current	Idle			1	uA
ADC static characteristics					
resolution	n	2		10	bits
integral non linearity	INL			+/- 1	LSB
differential non linearity	DNL			+/- 1	LSB
offset error	OSe	-20		20	mV
full scale error	FSe	-20		20	mV
ADC dynamic characteristics					
Sampling rate	Fsmpl	400(10 bits)		1500(2 bits)	kS/s
Conversion time	tconv	3(2bits)		11(10 bits)	clk cycles

29 10 BIT ADC

29.1 Overview

The 10 bitsis a successive approximation analog to digital converter with an 5 way input mux.

Features:

- Eight input channels selected through an analog multiplexer using sel[2:0] signal.
- From 2 to 10 bits resolution conversion, selected on the fly by means of strobe signal st.
- Maximum conversion rate 400k samples/s for 10 bits resolution.
- Power down mode performing minimal power dissipation.
- Internal power management to switch off unused circuitry between conversion cycles.
- No start-up cycles, no power down / power up recovery time.
- Muxed reference voltage inputs.

29.2 Functional description

This ADC is able to convert one of its 5 analog input signals with a conversion rate of 400 kSamples/s for a 10 bits resolution. However the resolution can be reduced to 2 bits and the conversion rate increased to 1500 kSamples/s. The ADC is composed of an analog mux 5:1 to select the input to convert, two 2:1 analog muxes to select the reference pair, a resistor string DAC and a successive approximation register. The selected reference pair is connected to the DAC which is controlled by the successive approximation register (SAR). One 10 bits conversion needs 11 clock cycles. During the first cycle, the selected input signal is sampled then the successive approximation determines the output code during the last 10 clock cycles.

29.3 Specification

In table 211 the general characteristics of the ADC module are shown. The specifications of the ADC are shown in table 203.

This module may be irreparably damaged if taken outside the specified absolute maximum rating range:

- Digital Supply Voltage (vdd): -0.5 to 2.0 Vdc
- Analog supply voltage (vdda): -0.5 to 3.6 Vdc
- Analog input voltage: -0.5 to 3.6 V
- Storage Temperature: -65 ^{o}C to +150 ^{o}C
- Max Junction Temperature: + 150 ^{o}C

Table 203 Specifications of the ADC

PARAMETER	SYMBOL	MIN.	TYP.	MAX.	UNIT
operation conditions					
junction temperature	Tjunc	-40	25	125	C
analog supply voltage	vdda	2.5	3.3	3.6	V
digital supply	vdd!	1.6	1.8	2.0	V
substate noise				150	mVpp
vrefn range	vrefn	vssa		vrefp-2	V
vrefp range	vrefp	vrefn+2		vdda	V
input impedance vrefp to vrefn		20		39	kOhms
input range	vin	vrefn		vrefp	V

PARAMETER	SYMBOL	MIN.	TYP.	MAX.	UNIT
clock frequency	Fclk			4.5	MHz
general					
total current consumption on vdda	Iact			400	uA
total power down current	Idle			1	uA
ADC static characteristics					
resolution	n	2		10	bits
integral non linearity	INL			+/- 1	LSB
differential non linearity	DNL			+/- 1	LSB
offset error	OSe	-20		20	mV
full scale error	FSe	-20		20	mV
ADC dynamic characteristics					
Sampling rate	Fsmpl	400(10 bits)		1500(2 bits)	kS/s
Conversion time	tconv	3(2bits)		11(10 bits)	clk cycles

30 10 BIT ADC INTERFACE

30.1 Overview

This section specifies the ADC interface, a module that connects an Analog Digital converter to an APB. The ADC interface module can be used for observing battery voltage.

The interface can be divided into 2 main modules, a 10 bit A/D converter and an ADC controller.

The A/D converter is a 10 bit successive approximation analog to digital converter . The ADC controller module is responsible for the communication between the A/D converter and APB.

The basic characteristics of the ADC interface module are:

- Five analog input channels, selected by an analog multiplexer;
- ‘Generic Interface’ towards a APB function;
- Programmable ADC resolution from 2 to 10 bits;
- Single A/D conversion scan mode and continuous A/D conversion scan mode;
- Converted digital values are stored in a 2 * 10 bits register;
- Power down mode.

30.2 Functional description

The ADC interface takes care off converting analog signals to digital data and write this data to the APB on request or via an interrupt.

In figure Fig.39 on page 296 , a block diagram of the ADC interface is depicted including the interconnect towards the A/D converter.

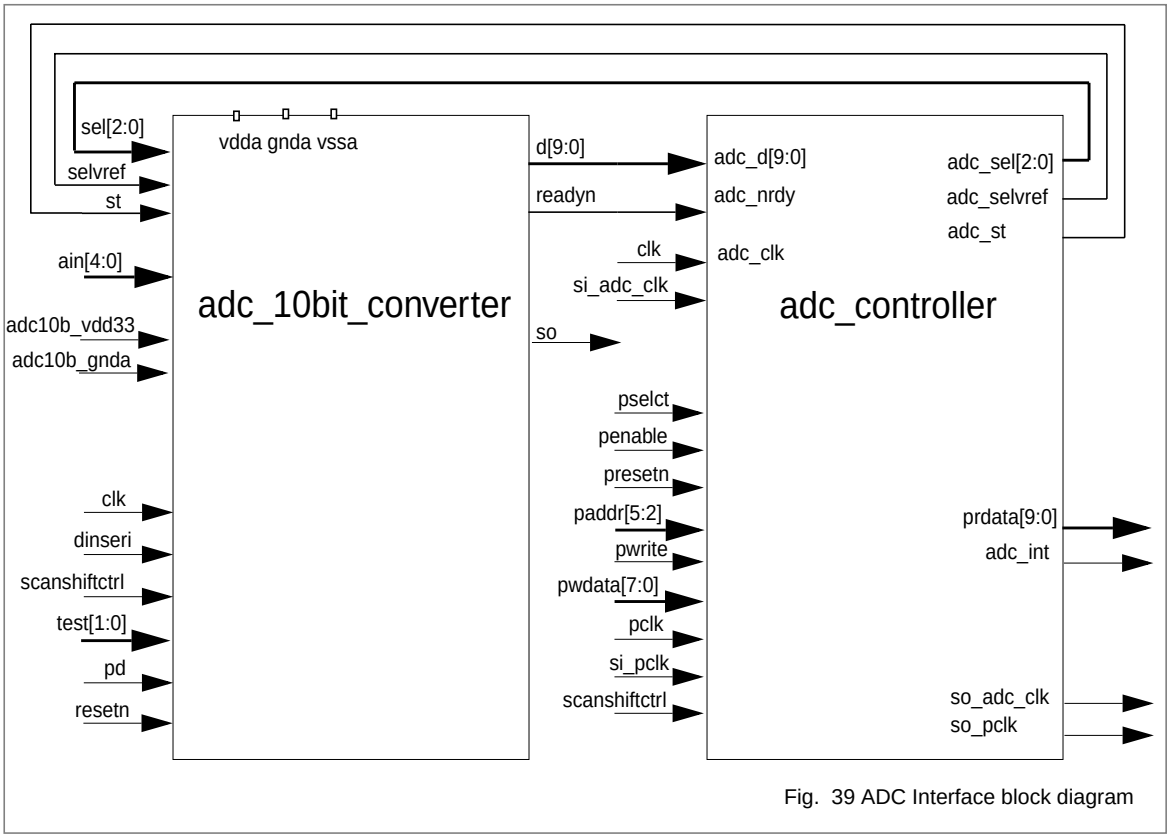


Fig. 39 ADC Interface block diagram

The functional blocks are:

- adc_10bit_converter:
- adc_controller:
  - Registers: a set of 7 software accessible registers, plus logic for address decoding and control of read/write access.
  - Control: a state machine that controls the selection of analog input channel(s), the conversion process within the AMoS A/D converter and the storage of converted digital data in a set of 5 result registers.

30.2.1 A/D CONVERSION CONTROL

The AMoS ADC core performs analog input channel multiplexing, sampling and successive digital approximation of analog signals under control of the ADC controller module. For this purpose the ADC interface provides the following signals:

- adc_sel[2..0]: select one of the five analog input channels;
- adc_st: control conversion sequence strobe signal;
- adc_selvref: select reference voltage.

The protocol sequence starts with the sampling of the selected analog input channels. This is followed by the ‘approximation loop’ in which the DAC voltage is step wise approximated to sampled input voltage. The number of loop cycles depends on whether 2,3,4,..,10 bit resolution is selected.

The A/D conversion result becomes valid when nrdy signal is low, at the next rising edge clock the result is moved to the ADC_Rx register of the corresponding channel.

30.2.2 ADC RESOLUTION

The resolution within the AD conversion process is software programmable through ADC controller variables. The resolution can be adjust between 2 and 10 bits.

The conversion rate is computed as follows:

$$\mathrm{conversionrate} = \frac{\mathrm{clockfrequency}}{(\mathrm{resolution} + 1)}$$

30.2.3 MULTI CHANNEL A/D CONVERSION SCAN

Associated to each analog input channel is a set of eight 10 bits result registers for storage of A/D conversion result. It is programmable which channels are included and which channels are excluded from the A/D conversion scan process. The A/D conversion scan process can be started by software.

There are two scan modes, ‘Continuous Scan’ mode and ‘Single Scan’ mode.

In ‘Continuous Scan’ mode, A/D conversion scans are carried out continuously: once one scan completed, the next one is started automatically. In ‘Single Scan’ mode, only a single conversion scan is carried out, the next scan must be started explicitly by software.

30.2.4 CLOCKING

ADC interface is operated on clock signal from APB(pclk). The AMoS ADC is operated on a clock from Clock Generation Unit (CGU).

The frequency of the clock (clk) doesn’t have to be very high because the number of samples can be low for measuring for exapmle a battery voltage, so the clock frequency (clk) can be programmed to 31.25kHz and is offered by the Clock Generation Unit.

30.3 Registers

The ADC interface function provides 13 software accessible registers, refer to table 204 on page 298.

Table 204ADC Interface Register map

ADDRESS SPACE	OFFSET	REGISTER NAME	R/W ACCESS	RESET
0x8000_2400 - 0x8000_27FF	0x00	ADC_R0	R	0x0000.0000
	0x04	ADC_R1	R	0x0000.0000
	0x08	ADC_R2	R	0x0000.0000
	0x0C	ADC_R3	R	0x0000.0000
	0x10	ADC_R4	R	0x0000.0000
	|			
	Reserved			
	|			
	0x20	ADC_CON	R/W	0x0000.0000
	0x24	ADC_CSEL_RES	R/W	0x0000.0000
	0x28	ADC_INT_ENABLE	R/W	0x0000.0000
	0x2C	ADC_INT_STATUS	R	0x0000.0000
	0x30	ADC_INT_CLEAR	W	
	0x34	Reserved		

30.3.1 ADC Results Register (ADC_Rx)

The ADC_Rx registers contain the output data, the output data is send to APB when CPU gives a read request. These eight registers store the result of an A/D conversion scan through 8 channels. Register ADC_R0 is associated to channel 0 and ADC_R1 to channel 1 and so on. The registers are read-only.

Table 205 ADC_Rx registers

BIT	VARIABLE	DESCRIPTION	RESET	R/W
4 - 0	ADC_Rx	Digital conversion data with respect to analog input channel	-	R
31 - 5		Reserved		

Table 206 shows the A/D conversion results with different resolutions.

Table 206 A/D conversion result table

RESOLUTION	ADC_RX[9]	ADC_RX[8]	ADC_RX[7]	ADC_RX[6]	ADC_RX[5]	ADC_RX[4]	ADC_RX[3]	ADC_RX[2]	ADC_RX[1]	ADC_RX[0]
2	valid	valid	0	0	0	0	0	0	0	0
3	valid	valid	valid	0	0	0	0	0	0	0
4	valid	valid	valid	valid	0	0	0	0	0	0
5	valid	valid	valid	valid	valid	0	0	0	0	0
6	valid	valid	valid	valid	valid	valid	0	0	0	0
7	valid	valid	valid	valid	valid	valid	valid	0	0	0
8	valid	valid	valid	valid	valid	valid	valid	valid	0	0
9	valid	valid	valid	valid	valid	valid	valid	valid	valid	0
10	valid	valid	valid	valid	valid	valid	valid	valid	valid	valid

30.3.2 ADC Control Register

This register controls the ADC operation modes and gives status information.

Table 207 ADC_CON Register

BIT	VARIABLE	RESET	R/W
0	ADC_SELVREF	0	R/W
1	ADC_ENABLE	0	R/W
2	ADC_CSCAN	0	R/W
3	ADC_START	0	R/W
4	ADC_STATUS	-	R
31: 5	Reserved		

ADC_SELVREF Select reference voltage input:

0: vrefp1 reference selected.

1: vrefp0 reference selected.

ADC_ENABLE

ADC enable

0: Non-operational mode. Power consumption is minimized. Start of A/D conversion is not possible. An ongoing A/D conversion scan is completed before the ADC is disabled.

1: Operational mode. Start of A/D conversion scans is possible.

ADC_CSCAN

ADC Continuous Scan:

0: Single A/D conversion scan mode. Triggered by write ‘1’ action to the ADC_START flag. A single A/D conversion scan is performed through the channels that are selected by the ADC_CSEL_RES register. Once all selected channels have been sampled, the process stops and the sampled data is kept in registers ADC_R0 and ADC_R1.

1: Continuous A/D conversion scan mode. After a write ‘1’ action to the ADC_START flag, A/D conversion scans are carried out continuously. Conversion results in registers ADC_R0 and ADC_R1 are continuously updated. The process stops when ADC_CSCAN is cleared.

ADC_START

ADC start command:

0: No effect.

1: Start an A/D conversion scan through the channels selected by the ADC_CSEL_RES register and use the operating mode defined by ADC_ENABLE, ADC_CSCAN and ADC_SELVREF

ADC_STATUS

ADC Status:

0: Indicates that no A/D conversion scan is in progress.

1: Indicates that an A/D conversion scan is in progress. The flag is cleared in ‘single A/D conversion scan’ mode when the A/D conversion scan through the selected channels has been completed. In ‘continuous A/D conversion’ mode the flag remains set. Power down mode is not allowed when A/D conversion scan is in progress.

30.3.3 SELECT CHANNEL AND RESOLUTION

This register defines which analog input channels are included and defines resolution in an A/D conversion.

Table 208ADC_CSEL_RES register

BIT	NAME	FUNCTION	RESET	R/W
3:0	ADC_CSEL0	Select channel 0.	0	R/W
7:4	ADC_CSEL1	Select channel 1.	0	R/W
11:8	ADC_CSEL2	Select channel 2.	0	R/W
15:12	ADC_CSEL3	Select channel 3.	0	R/W
19:16	ADC_CSEL4	Select channel 4.	0	R/W
31:20	-	Reserved	-	

Table 209 shows the resolution selection.

Table 209ADC_CSEL channel selection and resolution value.

VALUE X[3:0]	FUNCTION
0000	Channel not selected.
0001	Reserved
0010	2 bit resolution
0011	3 bit resolution
0100	4 bit resolution
0101	5 bit resolution
0110	6 bit resolution
0111	7 bit resolution
1000	8 bit resolution
1001	9 bit resolution
1010	10 bit resolution
1011	Reserved
1100	Reserved
1101	Reserved
1110	Reserved
1111	Reserved

30.3.4 ADC Interrupt Enable register

This register contains a variable to enable/disable the interrupt request generation.

Table 210 ADC_INT_ENABLE register

BIT	VARIABLE	RESET	R/W
0	ADC_SCAN_INT_ENABLE	0	R/W
31:1	Reserved	-	

ADC_SCAN_INT_ENABLE Interrupt enable:

0: Disable ADC_INT interrupt request.

1: Enable ADC_INT interrupt request. An interrupt request is generated when the ADC_SCAN_INT_STATUS flag is set.

30.3.5 ADC Interrupt Status register

This register contains interrupt status variable that indicates the presence of interrupt condition. It is read-only.

Table 211 ADC_INT_STATUS register

BIT	VARIABLE	RESET	R/W
0	ADC_SCAN_INT_STATUS	0	R
31:1	Reserved	-	

ADC_SCAN_INT_STATUS Interrupt status:

0: No scan interrupt pending.

1: Scan interrupt pending. An interrupt request is issued when ADC_SCAN_INT_ENABLE = 1.

30.3.6 ADC Interrupt Clear register

A write action to this address location allows to clear interrupt status variable in ADC_INT_STATUS register.

Table 212 ADC_INT_CLEAR register

BIT	VARIABLE	RESET	R/W
0	ADC_SCAN_INT_CLEAR	-	W
31:1	Reserved	-	

ADC_SCAN_INT_CLEAR Interrupt clear:

0: No effect.

1: clear ADC_SCAN_INT_STATUS variable.

30.4 Interrupts

The ADC interface implements one interrupt, a scan interrupt which indicates the completion of an A/D conversion scan process and the validity of the data in the result registers.

30.5 Programmer's Guide

30.5.1 INITIALISATION

Reset ADC Interface:

1. write PRESETN: reset ADC_CONTROLLER

ADC_ENABLE = 0

ADC_START = 0

ADC_SELVREF = 0

ADC_CSCAN = 0

ADC_CSEL0 = 0

ADC_CSEL1 = 0

ADC_SCAN_INT_ENABLE = 0

ADC_SCAN_INT_STATUS = 0

30.5.2 SETUP CONVERSION

Setup ADC Interface:

1. read ADC Status => ADC_STATUS = 0
2. write ADC Interrupt Enable Register => ADC_SCAN_INT_ENABLE = 1
3. read ADC Interrupt Status Register => ADC_SCAN_INT_STATUS = 0
4. select reference voltage input (vrefp0) => ADC_SELVREF = 1
5. write Select Channel and Resolution Register => ADC_CSEL0 untill ADC_CSEL4
6. write ADC enable bit => ADC_ENABLE = 1

30.5.3 RUN SINGLE MODE CONVERSION

Run Single Conversion Mode (ADC_CSCAN = 0):

1. write ADC Start Command => ADC_START = 1
2. write ADC Start Command => ADC_START = 0
3. wait for interrupt => ADC_INT
4. read ADC Interrupt Status Register => ADC_SCAN_INT_STATUS = 1
5. write ADC interrupt clear register => ADC_SCAN_INT_CLEAR = 1
6. read ADC Result Register for Channel 0 untill Channel 4=> ADC_R0 .. ADC_R4
7. wait for ADC Interrupt Status Register => ADC_SCAN_INT_STATUS = 0

Run Single Conversion Stops => Go back to Setup ADC Interface

30.5.4 RUN CONTINUOUS MODE CONVERSION

Run Continuous Conversion Mode (ADC_CSCAN = 1):

1. write ADC Start Command => ADC_START = 1
2. write ADC Start Command => ADC_START = 0
3. wait for interrupt => ADC_INT
4. read ADC Interrupt Status Register => ADC_SCAN_INT_STATUS = 1
5. write ADC interrupt clear register => ADC_SCAN_INT_CLEAR = 1
6. read ADC Result Register for Channel 0 untill Cahnnel 4=> ADC_R0 .. ADC_R4
7. wait for ADC Interrupt Status Register => ADC_SCAN_INT_STATUS = 0

Run Continuous Conversion Start at point 3.

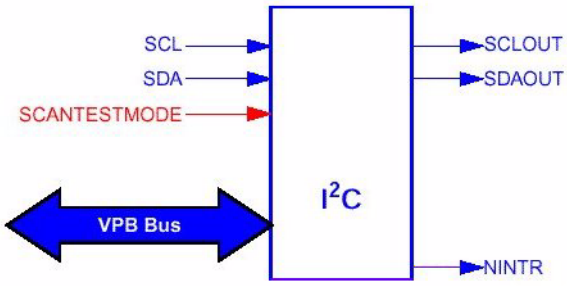
8. Stop Continuous Conversion => ADC_CSCAN = 0.

31 IIC MASTER/SLAVE INTERFACE MODULE

31.1 Overview

The I²C master/slave module provides a serial interface that meets the I²C bus specification and supports all transfer modes from and to the I²C bus. It supports the following functionality:

- It supports both the normal mode (100 kHz SCL) and the fast mode (400 kHz SCL).
- It has word (32-bits) access from the CPU side.
- Interrupt generation on received or sent byte (and some special cases).
- It has four modes of operation: master transmitter, master receiver, slave transmitter and slave reciever.



31.2 Restrictions

- With VPB frequencies below 24 MHz, 100K bits/sec is possible and 400K bits/sec is **not** possible.
- With VPB frequencies of 24 MHz and above, 100K bits/sec is possible and 400K bits/sec is possible.
- The first read of the status register, after a read of the receive fifo, is not always valid.
- There is found that there can be a problem when the Master I2C receives a byte, when the receive FIFO is full. The 8 bits of the latest received byte are received correctly, but the acknowledge bit not. During the acknowledge, SDA goes to early to '1' in the latest part of the high level of the clock. This causes an unwanted stop bit. Which is not expected. This problem To be sure, avoid that the receive FIFO becomes full.

31.3 Functional Description

The main features of the I²C-bus are:

- Serial clock synchronization allows devices with different bit rates to communicate via one serial bus
- Serial clock synchronization can be used as a handshake mechanism to suspend and resume serial transfer
- The I²C bus may be used for test and diagnostic purpose

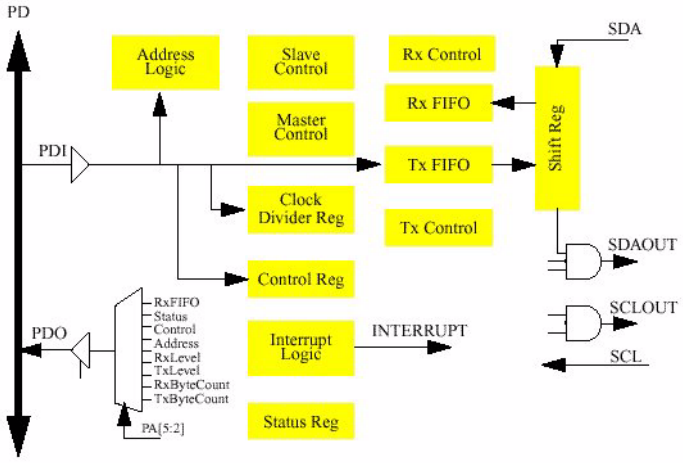
Two wires, SDA (serial data) and SCL (serial clock) carry information between devices connected to the I²C bus. Each device can operate as either a transmitter or receiver, depending on the function of the device. In addition to transmitters and receivers, devices can also be considered as masters or slaves when performing data transfers. A master is the device which initiates a data transfer on the bus and generates the clock signals to permit that transfer. Any device addressed by a master is considered a slave.

Generation of clock signals on the I²C bus is always the responsibility of the master device; each master generates its own clock signals when transferring data on the bus. Bus clock signals from a master can only be altered when they are stretched by a slow-slave device holding down the clock line, or by another master when arbitration occurs.

31.3.1 ARCHITECTURE

The VLSI Peripheral Bus interface provides a communications link between the CPU or host and the I2C controller.

The shift register handles serializing/deserializing data and counting bits of a byte. The Rx and Tx Control blocks count bytes and handling byte acknowl-edgement. The Master and Slave control blocks can each enable or disable the Rx and Tx control and also handle transfering data between the shift reg-ister and the Rx and Tx FIFOs. SDAOUT is driven either by shift register data, acknowledge signals, or the Master control for creating START or STOP con-ditions. SCLOUT is driven by the Master control generating a clock or by the Slave control extending the clock of an external master.



31.4 Registers

Separate TX FIFOs are needed in a multi-master because a controller might have a message queued for transmission when an external mas-ter addresses it to be come a slave-transmitter, a second source of data is needed. When the I2C has to function as a Master transmitter, one has to write to the TX register. And when the the I2C has to function as a Slave transmitter, one has to write to the TXS register. For Master receive and Slave receive funtionality, in both cases one has to read from the RX register.

Table 213IIC master register map

ADDRESS SPACE	OFFSET	REGISTER NAME	ACRONYM	R/W ACCESS	RESET
0x8002_0800 - 0x8002_0BFF	0x00	Receive FIFO	RX	R	
	0x00	Transmit FIFO	TX	W	
	0x04	Status Register	STS	R	0x0000.2A00
	0x08	Control Register	CTL	R/W	0x0
	0x0C	Clock Divisor High	CLKHI	R/W	0x0000.752E
	0x10	Clock Divisor Low	CLKLO	R/W	0x0000.752E
	0x14	I2C Address (slave functionality)	ADR	R/W	0x0000.001A
	0x18	Rx FIFO level	RFL	R	0x0000.0000
	0x1C	Tx FIFO level	TFL	R	0x0000.0000
	0x20	Number of bytes received. Reset when I2C transitions from inactive to active.	RXB	R	0x0000.0000
	0x24	Number of bytes transmitted. Reset when I2C transitions from inactive to active.	TXB	R	0x0000.0000
	0x28	Slave Transmit FIFO	TXS	W	0x0000.0000
	0x2C	Slave TX FIFO level	STFL	R	0x0000.0000

31.4.1 Receive FIFO (RX -' h00)

The RX is the top byte of the Rx FIFO. The top byte of the Rx FIFO contains the oldest byte received. RX[7:0] is read from data bus PD[7:0].

The RxFIFO is flushed by a hard reset or by a soft reset (CTL bit 7).

Reading an empty FIFO produces a PERR, but gives unpredictable data results.

Table 214 RX register

BITS	VARIABLE
0	data bit 0 (LSB)
1	data bit 1
2	data bit 2
3	data bit 3
4	data bit 4
5	data bit 5
6	data bit 6
7	data bit 7(MSB or oldest received data bit)

31.4.2 Transmit FIFO (TX - 'h00)

The TX is the top byte of the Tx FIFO. The top byte is the newest byte in the Tx FIFO. TX[9:0] is written from the data bus PD[9:0].

The Tx FIFO will is flushed by a hard reset, soft reset (CTL bit 7), or if an arbitration failure occurs (STS bit 3).

Writes to a full FIFO cause PERR to be asserted. The data is ignored.

Operating as a master, the TX FIFO must be written for both write and read operations to transfer each byte. Bits [7:0] are ignored for master-receive operations. The master-receiver must write a (dummy) byte to the TX FIFO for each byte it expects to receive in the RX FIFO. When the STOP bit (9) is set or the START bit (8) is set to cause a RESTART condition on a byte written to the TX FIFO as a master-receiver, then the byte read from the slave is not acknowledged. That is, the last byte of a master-receive operation is not acknowledged.

If the controller is configured as a Master/Slave and bus arbitration logic is included to operate in a multi-master environment, then only master-transmit data should be written to TX, slave transmit data should be written to TXS.

Table 215 TX register

BITS	VARIABLE
0	data bit 0 (LSB)
1	data bit 1
2	data bit 2
3	data bit 3
4	data bit 4
5	data bit 5
6	data bit 6
7	data bit 7(MSBt)
8	issue a START condition before transmitting this byte
9	issue a STOP condition after transmitting this byte

31.4.3 Status Register (STS -’ h04)

The IIC_STATUS is a read-only register that provides status information on the Tx and Rx blocks as well as the current state of the external buses. An active reset, PNRES = 0, will clear the STS register with the exception of RFE and TFE (these will be 1) and SCL and SDA (indeterminate).

Table 216 STS Register

BITS	VALUE	DESCRIPTION
0 Transaction Done (TDI).	0(default)	Transaction has not completed.
	1	Transaction completed. This flag is set if a transaction completes successfully. It is cleared by writing a 1 to bit 0 of the status register. It is unaffected by slave transactions.
1 Arbitration Failure (AFI).	0(default)	No arbitration failure on last transmission.
	1	Arbitration failure occurred on last transmission. When transmitting, if the SDA is low when SDAOUT is high, then this I2C has lost the arbitration to another device on the bus. The Arbitration Failure bit is set when this happens. It is cleared by writing a 1 to bit 1 of the status register.
2 No Acknowledge (NAI).	0(default)	Last transmission received an acknowledge.
	1	Last transmission did not receive an acknowledge. After every byte of data is sent, the transmitter expects an acknowledge from the receiver. This bit is set if the acknowledge is not received. It is cleared when a byte is written to the master Tx FIFO.
3 Master Data Request (DRMI).	0(default)	Master transmitter does not need data.
	1	Master transmitter needs data. Once a transmission is started, the transmitter must have data to transmit as long as it is not followed by a stop condition or it will hold SCL low until more data is available. The Master Data Request bit is set when the master transmitter is data-starved. If the master TX FIFO is empty and the last byte did not have a STOP condition flag, then SCL is held low until the mP writes another byte to transmit. This bit is cleared when a byte is written to the master Tx FIFO.
4 Slave Data Request (DRSI).	0(default)	Slave transmitter does not need data.
	1	Slave transmitter needs data. Once a transmission is started,the transmitter must have data to transmit as long as it isn't followed by a stop condition or it will hold SCL low until more data is available. The Slave Data Request bit is set when the slave transmitter is data-starved. If the slave TX FIFO is empty and the last byte transmitted was acknowledged, then SCL is held low until the mP writes another byte to transmit. This bit is cleared when a byte is written to the slave Tx FIFO.
5 Active.	0(default)	Indicates whether the bus is busy. This bit is set when a START condition has been seen. It is cleared when a STOP condition is seen.
6 SCL.	0(default)	The current value of the SCL signal.
7 SDA.	0(default)	The current value of the SDA signal.
8 Receive FIFO Full (RFF).	0(default)	RX FIFO is not full
	1	RX FIFO is full This bit is set when the RX FIFO is full and cannot accept any more data. It is cleared when the RX FIFO is not full. If a byte arrives when the Receive FIFO is full, the SCL is held low until the mP reads the RX FIFO and makes room for it.
9 Receive FIFO Empty (RFE).	0	RX FIFO contains data.
	1(default)	RX FIFO is empty RFE is set when the RX FIFO is empty and is cleared when the RX FIFO contains valid data.
10 Transmit FIFO Full (TFF).	0(default)	TX FIFO is not full.
	1	TX FIFO is full TFF is set when the TX FIFO is full and is cleared when the TX FIFO is not full.
11 Transmit FIFO Empty (TFE).	0	TX FIFO contains valid data.
	1(default)	TX FIFO is empty TFE is set when the TX FIFO is empty and is cleared when the TX FIFO contains valid data.
12 Slave Transmit FIFO Full (TFFS).	0 (default)	TX FIFO is not full.
	1	TX FIFO is full TFF is set when the slave TX FIFO is full and is cleared when the slave TX FIFO is not full. This bit exists only for master/slave configurations that include bus arbitration logic.
13 Slave Transmit FIFO Empty (TFES).	0	TX FIFO is not empty.
	1(default)	TX FIFO is empty TFF is set when the slave TX FIFO is empty and is cleared when the slave TX FIFO contains valid data. This bit exists only for master/slave configurations that include bus arbitration logic.

31.4.4 Control Register (CTL - 'h08)

The CTL register is used to enable interrupts and reset the I^2C state machine. CTL[n:0] are read and written from data bus PD[n:0]. If Register Style: PSTB is selected, then this register is updated on the rising edge of PSTB.

Table 217IIC_CNTRL Registers Control Register RW

BITS	VALUE	DESCRIPTION
0 Transmit Done Interrupt Enable (TDIE)	0(default)	Disable the TDI interrupt.
	1	Enable the TDI interrupt. This enables the TDI interrupt signalling that this I^2C issued a stop condition.
1 Transmitter Arbitration Failure Interrupt Enable (AFIE)	0(default)	Disable the AFI.
	1	Enable the AFI. This enables the AFI interrupt which is asserted during transmission when trying to set SDA high, but the bus is driven low by another device.
2 Transmitter No Acknowledge Interrupt Enable (NAIE)	0(default)	Disable the NAI.
	1	Enable the NAI. This enables the NAI interrupt signalling that transmitted byte was not acknowledged.
3 Master Transmitter Data Request Interrupt Enable (DRMIE)	0(default)	Disable the DRMI interrupt.
	1	Enable the DRMI interrupt. This enables the DRMI interrupt which signals that the master transmitter has run out of data, has not issued a stop, and is holding the SCL line low.
4Slave Transmitter Data Request Interrupt Enable (DRMIE)	0(default)	Disable the DRSI interrupt.
	1	Enable the DRSI interrupt. This enables the DRSI interrupt which signals that the slave transmitter has run out of data and the last byte was acknowledged, so the SCL line is being held low.
5 Receive FIFO Full Interrupt Enable(RFFIE)	0(default)	Disable the RFFI.
	1	Enable the RFFI. This enables the Receive FIFO Full interrupt to indicate that the receive FIFO cannot accept any more data.
6 Receive Data Available Interrupt Enable (DAIE)	0(default)	Disable the DAI.
	1	Enable the DAI. This enables the DAI interrupt to indicate that data is available in the receive FIFO (i.e. not empty).
7 Transmit FIFO Not Full Interrupt Enable (TFFIE)	0(default)	Disable the TFFI.
	1	Enable the TFFI. This enables the Transmit FIFO Not Full interrupt to indicate that the more data can be written to the transmit FIFO. Note that this is not full. It is intended help the mP write to the I^2C block only when there is room in the FIFO to accept it and do this without polling the status register.
8 Soft Reset	0(default)	
	1	Reset the I^2C to idle state. Self clearing. This is only needed in unusual circumstances. If a device issues a start condition without issuing a stop condition. A system timer may be used to reset the I^2C if the bus remains busy longer than the time-out period. On a soft reset, the Tx and Rx FIFOs are flushed, STS register is cleared, and all internal state machines are reset to appear idle. The CLK, CTL, and ADR registers are NOT modified by a soft reset.
9 Reserved		
10 Slave Transmit FIFO Not Full Interrupt Enable (TFFSIE)	0(default)	Disable the TFFSI.
	1	Enable the TFFSI. This enables the Slave Transmit FIFO Not Full interrupt to indicate that the more data can be written to the slave transmit FIFO. Note that this is not full. This bit exists only for master/slave configurations that include arbitration logic.

31.4.5 Clock Divider High (CLKHI - ‘h0C)

The CLK register holds a terminal count for counting PCLK cycles to create the high period of the slower I2C serial clock, SCL. CLKHI[n:0] are read and written from data bus PD[n:0]. When reset, the clock divider will be set to run at its minimum frequency. If Register Style: PSTB is selected, then this register is updated on the rising edge of PSTB.

Table 218 CLKHI Register.

BITS	VALUE	DESCRIPTION
14: 0	See text above	Clock Divisor High This value is the number of system clocks the serial clock (SCL) will be high. n is defined at compile time based on the system clock (PCLK) and minimum serial frequency. For example, if the system clock period is 50 ns and the maximum desired serial period is 10000 ns (100 kHz), then CLKHI would be set to **0.5*(f_sys/f_i2c)-2**=0.5*(20e6/100e3)-2=98. The actual value programmed into CLKHI will vary from this slightly due to variations in the output pad's rise and fall times as well as the deglitching filter length. In this example, n = 7, since eigth bits are needed to hold the clock divider count.

31.4.6 Clock Divider Low (CLKLO-' h10)

The CLK register holds a terminal count for counting PCLK cycles to create the low period of the slower I 2 C serial clock, SCL. CLKLO[n:0] are read and written from data bus PD[n:0]. When reset, the clock divider will be set to run at its minimum frequency. If “Register Style: PSTB” is selected, then this register is updated on the rising edge of PSTB.

Table 219 CLKLO Register.

BITS	VALUE	DESCRIPTION
14: 0	See text above	Clock Divisor Low This value is the number of system clocks the serial clock (SCL) will be low. This register is the same width as the CLKHI register. See text above

31.4.7 Address Register (ADR - 'h14)

The ADR is the I2C bus slave address. Bits 6:0 are implemented only if the I2C is configured for slave mode. Bits 9:7 are implemented only if it is configured for slave mode and '10-bit addressing' is selected. ADR[9:0] is read and written from data bus PD[9:0]. If "Register Style: PSTB" is selected, then this reg-ister is updated on the rising edge of PSTB.

If the address is programmable, ADR will be set to the address specified at compile time when reset.

Table 220 ADR Register

BITS	VARIABLE
0	address bit 0 (LSB)
1	address bit 1
2	address bit 2
3	address bit 3
4	address bit 4
5	address bit 5
6	address bit 6
7	address bit 7
8	address bit 8
9	address bit 9 (MSB)

31.4.8 Receive FIFO LEVEL Register (RFL - ‘h18).

This register shows the number of bytes stored in the RX FIFO.

31.4.9 Transmit FIFO LEVEL Register (TFL - ‘h1C).

This register shows the number of bytes in the TX FIFO.

31.4.10 RECEIVE FIFO BYTE REGISTER (RXB - ‘H20).

This register indicates the number of bytes received so far. Reset when I2C transitions from inactive to active.

31.4.11 Transmit FIFO BYTE Register (TXB - ‘h24).

This register indicates the number of bytes sent so far. Reset when I2C transitions from inactive to active.

31.4.12 Slave Transmit Buffer Register (TXS - ’h28).

The TXS is the top byte of the Slave Tx FIFO. The top byte is the newest byte in the Slave Tx FIFO. TX[7:0] is written from the data bus PD[9:0].

The Slave Tx FIFO will is flushed by a hard reset or soft reset (CTL bit 7).

Writes to a full FIFO cause PERR to be asserted. The data is ignored.

This register is only available for Master/Slave configurations that include arbi-tration logic for operating in a multi-master environment. In this case, all slave transmit data is written to TXS while all master transmit data is written to TX.

Table 221 TXS Register

Bits	VARIABLE	VALUE AFTER RESET
0	data bit 0 (LSB)	U
1	data bit 1	U
2	data bit 2	U
3	data bit 3	U
4	data bit 4	U
5	data bit 5	U
6	data bit 6	U
7	data bit 7 (MSB)	U

31.5 Interrupt

Active low signal indicates if an interrupt is pending. The reason for the interrupt is encoded in Status Register. There are several possible interrupt types: transfer completed, arbitration failure, missing acknowledge, need more data, Tx FIFO has room for more data, or data has been received. Status register bits are ANDed with enables in the Control register then ORed together and negated to create NINTR.

28 UART

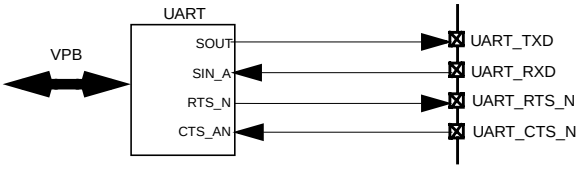
28.1 Overview

This chapter specifies the interfaces and functions of the IP_3106 VPB UART. The Universal Asynchronous Receiver Transmitter is commonly used to implement a serial interface. This UART includes a partial modem interface as well. In any case where a device needs a low overhead, standard, low performace interface, a UART can be used.

The UART includes advanced features like a fractional clock divider.

Fig 1. shows a blockdiagram of the UART.

Figure 1: UART blockdiagram.



28.2 Configuration

This UART is configured as follows:

- This UART is compatible with the industry-standard ‘650 UARTs. It configures UART transmit and receive FIFO size and trigger levels according to Table 1. Table 1 lists the FIFO sizes and trigger points corresponding to the different UART type parameters.

Table 1: UART type configuration

UART type	FIFO size	FCR. RxTrigLevel = 0	FCR. RxTrigLevel = 1	FCR. RxTrigLevel = 2	FCR. RxTrigLevel = 3
‘450	1	NA	NA	NA	NA
‘550	16	1	4	8	14
‘650	32	1	16	24	28
‘750	64	1	16	32	56

- No full modem interface is included. Only cts/rts modem interface is available. Hardware flow control is supported.
- IrDA module included.
- Signal on UART_RxD pin can be used as wake-up event.

28.3 Clocking

The UART has two clock inputs:

- the vpb_clk operates the VPB interface and the registers in the software view
- the u_clk is used for UART baud-rate generation and operates most of the UART data-path including the optional IrDA

Unless the UART is configured as a type ‘450 UART (par_sc3106_uart_type = 450), the clocks can be asynchronous i.e. need not have a frequency or phase relation. In ‘450 configuration the UART only supports synchronous identical clocks with balanced clock trees. In non ‘450 configurations the vpb_clk frequency can be smaller than, equal to or higher than the u_clk frequency.

The u_clk can be switched off by an external clock gate. In this case the UART cannot receive and transmit characters; the VPB interface is still operational and can be accessed without deadlocking the VPB bus.

The vpb_clk can be switched off by an external clock gate. In this case the UART registers are not accessible from the VPB bus. The UART can still receive and transmit data but will not forward interrupt and DMA status to the CPU and the DMA controller.

28.4 Software interface

28.4.1 Register overview

Table 2 lists the registers and the offset address of the register. Some registers can only be accessed if bit 7 of the Line Control Register is set; this is the Divisor Latch Enable bit (DLAB).

Table 2: Register overview

Name	Address Offset	DLAB	R/W/RW	Description
RBR	0x000	0	R	Receiver Buffer Register
THR	0x000	0	W	Transmitter Holding Register
DLL	0x000	1	RW	NHP Pop Register
IER	0x004	0	RW	Interrupt Enable Register
DLM	0x004	1	RW	Divisior Latch MSB
IIR	0x008	x	R	Interrupt Identification Register
FCR	0x008	x	W	FIFO Control Register
LCR	0x00C	x	RW	Line Control Register
MCR	0x010	x	RW	Modem Control Register
LSR	0x014	x	R	Line Status Register
MSR	0x018	x	R	Modem status Register
SCR	0x01C	x	RW	Scratch Register
ACR	0x020	x	RW	Auto-baud Control Register
ICR	0x024	x	RW	IrDA Control Register
FDR	0x028	x	RW	Fractional Divider Register
-	0x02C	x	R	reserved: read operation returns 0x0
POP	0x030	x	W	NHP Pop Register
MODE	0x03	x	RW	NHP Mode Selection Register
-	0x038-0xFD0	x	R	reserved: read operation returns 0x0
CFG	0xFD4	x	R	Configuration Register
INTCE	0xFD8	x	W	Interrupt Clear Enable Register
INTSE	0xFDC	x	W	Interrupt Set Enable Register
INTS	0xFE0	x	R	Interrupt Status Register
INTE	0xFE4	x	R	Interrupt Enable Register
INTCS	0xFE8	x	W	Interrupt Clear Status Register

Table 2: Register overview

Name	Address Offset	DLAB	R/W/RW	Description
INTSS	0xFEC	x	W	Interrupt Set Status Register
-	0xFF0-0xFF8	x	R	reserved: read operation returns 0x0
MID	0xFFC	x	R	Module Identification Register

28.4.2 REGISTER DESCRIPTION

28.4.2.1 Receiver Buffer Register

The Receiver Buffer Register (RBR) is located on byte address 0x000 and can only be accessed if the DLAB = 0 (Divisor Latch Access bit, situated in Line Control register LCR[7]) otherwise the DLL register will be accessed. The RBR register is read-only; writing this address will access the THR register. Table 3 lists the bit definition of the register.

Table 3: Receiver Buffer Register bit definition

Bit Range	Variable Name	Description	R/W/RW	Reset Value
31:8	reserved	reserved for future use	R	0x0
7:0	RBRFirst	In FIFO mode (FCR[0] = 1), top of Receiver FIFO; in non FIFO mode (FCR[0] = 0), value of buffer register	R	undefined

For an ‘650 configuration in non FIFO mode (i.e. when FCR[0] is not set), the RBR can store one received character. The value of the RBR can be read via this register. After reading the register the value is undefined until the next character is received.

For a ’650 UART configuration in FIFO mode i.e. when FCR[0] is set, the value in the register represents the first character in the UART FIFO. In this mode reading a character from the RBR will pop the character from the RBR FIFO. After the read operation the RBR will have the value of the next character in the FIFO. If the FIFO is empty the value of the register will be undefined until the next character is received.

If the UART is configured as Nexperia Home Platorm (NHP) compliant by setting the MODE.NHP bit then the RBR register of the UART is protected against speculative read options and reading RBR will not pop the character from the top of the FIFO. Instead the top FIFO character will be popped by writing 0x1 to the POP register.

Bit 0 is the least significant bit and is the first bit serially received. If the UART transmit/receive word length as defined in LCR[1:0] is less than 8 bits per character then the upper bits of the RBR will be 0.

28.4.2.2 Transmitter Holding Register

The Transmitter Holding Register (THR) is on byte address 0x000 and can only be accessed if DLAB = 0, otherwise the DLL register will be accessed. The THR register is write-only; reading this address will access the RBR register. Table 4 lists the bit definition of the register.

Table 4: Transmitter Holding Register bit definition

Bit Range	Variable Name	Description	R/W/RW	Reset Value
31:8	reserved	reserved for future use	R	0x0
7:0	THRLast	In FIFO mode writing will queue a byte in the transmit FIFO; in non FIFO mode writing will write the value of the transmit holding register	W	NA

For a ‘650 UART in non FIFO mode (i.e. when FCR[0] is not set), the THR can store one transmit character (In an ‘x50 configuration in non FIFO mode the full Tx FIFO depth can still be filled). The value for the THR can be written via this register. After writing the register the value will be transmitted on the UART serial out port (sout). New data can only be written in the register after old data has been transmitted. If new data is written before old data has been transmitted then the new data will be discarded.

But in the design the above mentioned characteristic is not implemented, for THR FIFO in x50 configuration in non FIFO mode FIFO can store characters till it becomes full.

For a ’650 UART configuration in FIFO mode i.e. when FCR[0] is set, the value in the register represents the tail the UART’s transmit FIFO/queue. In this mode writing a character to the THR will queue a character for transmission. If the FIFO is full any writes to the THR will be discarded.

Bit 0 is the least significant bit and is the first bit serially transmitted. If the UART transmit/receive word length as defined in LCR[1:0] is less than 8 bits per character then the upper bits of the THR will be ignored.

28.4.2.3 Interrupt Enable Register

The Interrupt Enable Register (IER) is on byte address 0x004 and can only be accessed if DLAB = 0, otherwise the DLM register will be accessed. The IER register can be read and written. Table 5 lists the bit definition of the register.

Table 5: Interrupt Enable Register bit definition

Bit Range	Variable Name	Description	R/W/RW	Reset Value
31:10	reserved	reserved for future use	R	0x0
9	ABTOIntEn	Auto-baud Time-Out Interrupt Enable.	RW	0x0
8	ABEOIntEn	End of Auto-Baud Interrupt Enable.	RW	0x0
7	CTSIntEn	If auto-cts mode is enabled this enables/disables the modem status interrupt generation on a cts_an signal transition. If auto-cts mode is disabled a cts_an transition will generate an interrupt if MSIntEn is set.	RW	0x0
6:4	reserved	reserved for future use	R	0x0
3	MSIntEn	Modem Status interrupt enable	RW	0x0
2	RLSIntEn	Receiver Line Status interrupt enable	RW	0x0
1	THREIntEn	Transmitter Holding Register Empty interrupt enable,	RW	0x0
0	RDAIntEn	Receive Data Available interrupt enable.	RW	0x0

The IER masks the interrupts from receiver ready, transmitter empty, line status and modem status registers.These interrupts would normally be seen in the IIR register and on the interrupt request output pin (uart_intreq).

In normal operation a cts_an signal transition will generate a Modem Status interrupt unless the interrupt has been disabled by clearing the MSIntEn bit in the IER register. In auto-cts mode a transition on the cts_an bit will trigger an interrupt only if both the MSIntEn and the CTSIntEn bits are set.

In Nexperia Home Platform compliant systems SW should use the INTSS and INTSE registers for implementing interrupt service routines.

28.4.2.4 Interrupt Identification Register

The Interrupt Identification Register (IIR) is on byte address 0x008 is read-only; writing this address will access the FCR register. Table 6 lists the bit definition of the register.

Table 6: Interrupt Identification Register

Bit Range	Variable Name	Description	R/W/RW	Reset Value
31:8	reserved	reserved for future use	R	0x0
9	ABTOInt	Auto-baud time-out interrupt True if auto-baud has timed out and interrupt is enabled.	R	0x0
8	ABEOInt	End of auto-baud interrupt. True if auto-baud has finished successfully and interrupt is enabled.	R	0x0
7:6	FIFOEn	copies of FCR[0]	R	0x0
5:4	reserved	reserved for future use	R	0x0
3:1	IntId	Interrupt identification	R	0x0
0	IntStatus	Interrupt status. If 0 then interrupt is pending; if 1 no interrupt is pending	R	0x1

Bits [7:6] of this register are copies of the FCR.FIFOEn bit.

Bit [9:8] are set by the auto-baud function and signal a time-out or end of auto-baud condition. The auto-baud interrupt conditions are cleared by setting the corresponding Clear bits in the Auto-baud Control Register.

If the IntStatus bit is 0, a non auto-baud interrupt is pending in which case the IntId bits identify the type of interrupt according to Table 7. If the IntStatus bit is 1 no interrupt is pending and the IntId bits will be zero.

Table 7: Interrupt identification and priorities

IntId	Priority	Type	Set	Clear
0x3	1 (high)	Receiver Line Status	Set on an overrun error, parity error, framing error or break indication	Reading the LSR
0x2	2	Received Data Available	Receiver data available in ‘450 mode or trigger level reached in FIFO mode	Reading the RBR ‘in 450 mode; in non ‘450 mode the FIFO level dropping below the trigger level
0x6	2	Character time-out; only generated if the FIFO is enabled	No characters have been received or on the serial input or removed from the receive FIFO during the last 4 character times and there is at least one character in the FIFO. The time-out counter is reset at the center of each stop bit received or each time the receive holding register (RHR) is read.	Reading the RBR
0x1	3	Transmitter Holding Register empty	Transmitter Holding Register empty (THRE)	Reading the IIR if IIR value is 0x1, or writing to THR
0x0	4 (low)	Modem status	Set on transition of cts_an	Reading the MCR

The IIR only indicates and interrupt if the corresponding bit in the IER register is set.

If multiple interrupts are pending only the highest priority interrupt will be indicated in the IntId bits of the IIR. Only after clearing a higher priority interrupt a low priority interrupt will be indicated. The UART provides four prioritized levels of interrupts:

Priority 1 – Receiver line status (highest priority)

Priority 2 – Receiver data available or receiver character time-out

Priority 3 – Transmitter holding register empty

Priority 4 – Modem status (lowest priority)

Interrupts can be cleared by reading the register causing the interrupt. The THRE can be cleared either by reading the IIR or writing to the THR register. The THRE interrupt will only be cleared by an IIR read if the THRE interrupt is the highest pending interrupt.

This interrupt is activated when THR FIFO is empty provided certain initialization conditions have been met. These initialization conditions are intended to give the THR FIFO a chance to fill up with data to eliminate many THRE interrupts from occurring at system start-up. The initialization conditions implement a one character delay minus the stop bit whenever THRE=1and there have not been at least two characters in the THR at one time since the last THRE=1 event. This delay is provided to give CPU time to write data to THR without a THRE interrupt to decode and service. A THRE interrupt is set immediately if the THR FIFO has held two or more characters at one time and currently, the THR is empty.

Note that the interrupt register can indicate multiple interrupts concurrently: the time-out and end of auto-baud interrupts and one of the non auto-baud interrupts.

In Nexperia Home Platform compliant systems SW should use the INTSS and INTSE register for implementing interrupt service routines since the IIR.THRE bit is not protected for speculative read operations.

28.4.2.5 FIFO Control Register

The FIFO Control Register (FCR) is on byte address 0x008 and is write-only; reading this address will access the IIR register. Table 8 lists the bit definition of the register. The value of the FCR should not be modified while receiving/transmitting data or data might get lost or corrupted.

Table 8: FIFO Control Register bit definition

Bit Range	Variable Name	Description	R/W/RW	Reset Value
31:8	reserved	reserved for future use	W	0x0
7:6	RxTrigLevel	Receiver trigger level selection; always 0x0 for a ‘450 configuration or if the FIFO is disabled..	W	0x0
5:4	reserved	reserved for future use	-	0x0
3	DMAMode	DMA mode select. When FIFOEn is set, setting DMAMode causes the rx_rdy_n, tx_rdy_n to change from mode 0 to mode 1.	W	0x0
2	TxFIFORst	Transmitter FIFO reset. TxFIFORst when set clears all bytes in the transmit FIFO and resets its counter to 0. The TSR is not cleared. The logic 1 that is written to this bit position is self clearing.	W	0x0
1	RxFIFORst	Receiver FIFO reset. RxFIFORst when set clears all bytes in the receiver FIFO and resets its counter. The RSR is not cleared. The logic 1 that is written to this bit position is self clearing.	W	0x0
0	FIFOEn	FIFO enable. FIFOEn when set enables the transmit and receive FIFOs. This bit must be set when other FCR bits are written to or the other bits will retain their old value. Changing this bit value clears the transmitter and receiver FIFOs.	W	0x0

The value of the FCR should not be modified while receiving/transmitting data or data might get lost or corrupted.

28.4.2.6 Line Control Register

The Line Control Register (LCR) is on byte address 0x00C. The register can be read and written. Table 9: lists the bit definition of the register.

Table 9: Line Control Register bit definition

Bit Range	Variable Name	Description	R/W/RW	Reset Value
31:8	reserved	reserved for future use	R	0x0
7	DLAB	Divisor Latch Access bit. If set enables access to the divisor latch registers, if cleared enables access to the RBR, THR, IER registers.	RW	0x0
6	BrkCtrl	break control bit. BrkCtrl is set to force a break condition; i.e., a condition where sout is forced to the spacing (low) state. When BrkCtrl is cleared, the break condition is disabled and has no affect on the transmitter logic; it only affects the serial output.	RW	0x0
5	ParStick	If parity is enabled by setting the ParEn bit the ParStick bit enables the stick parity mode.	RW	0x0
4	ParEven	If parity is enabled by setting the ParEn bit the ParEvenSel bit selects between even, odd, stick 0 and stick 1 parity.	RW	0x0
3	ParEn	Parity enable. Setting this bit appends a parity bit to each transmission and the receiver checks each word for parity errors. Clearing this bit disables parity transmission and receiver parity checking.	RW	0x0
2	StopBitNum	Number of stop bits selector. The number of stop bits depends on the value of WdLenSel and StopBitNum.	RW	0x0
1:0	WdLenSel	Word length select..	RW	0x0

Table 10 lists the parity options available in the UART.

Table 10: UART parity generation

LCR.ParEn	LCR.ParEven	LCR.ParStick	Description
0	x	x	Parity is disabled. No parity bit will be transmitted and no parity will be checked during receive.
1	0	0	Odd parity mode. data+parity bit will have an odd number of 1s
1	1	0	Even parity mode. data+parity bit will have an even number of 1s
1	0	1	Stick 1 mode: parity is always 1
1	1	1	Stick 0 mode: parity is always 0

Table 11: lists the word length and stop bit options available in the UART. Apart from the BrkCtrl bit the value of the LCR should not be modified while transmitting/receiving data or data might get lost or corrupted.

Table 11: UART character lengths and stop bit generation

LCR. WdLenSel	LCR.StopBit	Character length	Number of stop bits
0x0	0	5	1
0x1	0	6	1
0x2	0	7	1
0x3	0	8	1
0x0	1	5	1.5
0x1	1	6	2
0x2	1	7	2
0x3	1	8	2

Apart from the BrkCtrl bit the value of the LCR should not be modified while transmitting/receiving data or data might get lost or corrupted.

28.4.2.7 Modem Control Register

The Modem Control Register (MCR) is on byte address 0x010. The register can be read and written. Table 12 lists the bit definition of the register.

.

Table 12: Modem Control Register bit definition

Bit Range	Variable Name	Description	R/W/RW	Reset Value
31:8	reserved	reserved for future use	R	0x0
7	AutoCTSEn	Auto-cts flow control enable	RW	0x0
6	AutoRTSEn	Auto-rts flow control enable	RW	0x0
5	reserved	reserved for future use	RW	0x0
4[1]	LoopEn	Loop-back mode enable	RW	0x0
3	OUT2	Inverse control for the out2_n output	RW	0x0
2	OUT1	Inverse control for the out1_n output	RW	0x0
1	RTS	Request To Send. If the AutoRTSEn bit is set the RTS bit is read-only and will reflect the current state of the rts_n output. If the AutoRTSEn is cleared then the RTS bit is the inverse control for the rts_n output.	RW	0x0
0	DTR	Inverse control for the Data Terminal Ready output	RW	0x0

[1] When LoopEn is set registers MCR[3:0] are forced to “0000”

28.4.2.8 Line Status Register

The Line Status Register (LSR) is on byte address 0x014. The register is read-only. Table 13: lists the bit definition of the register. Note that in some industry standard UARTs the LSR register is read/writeable.

Table 13: Line Status Register bit definition

Bit Range	Variable Name	Description	R/W/RW	Reset Value
31:8	reserved	reserved for future use	R	0x0
7	RxErr	Error in receiver FIFO. Logic 0 = No error (normal default condition). Logic 1 = At least one parity error, framing error or break indication is in the current FIFO data. This bit is cleared when LSR register is read.	R	0x0
6	TEMT	Transmitter empty: TSR and THR are empty. This bit is set to a logic 1 whenever the transmit holding register and the transmit shift register are both empty. It is reset to logic 0 whenever either the THR or TSR contains a data character. In the FIFO mode, this bit is set to ‘1’ whenever the transmit FIFO and transmit shift register are both empty.	R	0x1
5	THRE	Transmitter Holding Register empty. This bit indicates that the UART is ready to accept a new character for transmission. In addition, this bit causes the UART to issue an interrupt to CPU when the THR interrupt enable is set. The THR bit is set to a logic 1 when a character is transferred from the transmit holding register into the transmitter shift register. The bit is reset to a logic 0 concurrently with the loading of the transmitter holding register by the CPU. In the FIFO mode, this bit is set when the transmit FIFO is empty; it is cleared when at least 1 byte is written to the transmit FIFO.	R	0x1
4	BI	Break indication. Logic 0 = No break condition (normal default condition). Logic 1 = The receiver received a break signal (RX was a logic 0 for one character frame time). In the FIFO mode, only one break character is loaded into the FIFO. This bit is cleared when the LSR register is read.	R *	0x0

Table 13: Line Status Register bit definition

Bit Range	Variable Name	Description	R/W/RW	Reset Value
3	FE	Framing error. Logic 0 = No framing error (normal default condition). Logic 1 = Framing error. The receive character did not have a valid stop bit(s). In the FIFO mode, this error is associated with the character at the top of the FIFO.	R *	0x0
2	PE	Parity error. Logic 0 = No parity error (normal default condition). Logic 1 = Parity error. The receive character does not have correct parity information and is suspect. In the FIFO mode, this error is associated with the character at the top of the FIFO. This bit is cleared when the LSR register is read.	R *	0x0
1	OE	Overrun error. Logic 0 = No overrun error (normal default condition). Logic 1 = Overrun error. A data overrun error occurred in the receive shift register. This happens when additional data arrives while the FIFO is full. In this case, the previous data in the shift register is overwritten. Note that under this condition, the data byte in the receive shift register is not transferred into the FIFO, therefore the data in the FIFO is not corrupted by the error.This bit is cleared when the LSR register is read.	R *	0x0
0	DR	Data ready Logic 0 = No data in receive holding register or FIFO (normal default condition). Logic 1 = Data has been received and is saved in the receive holding register or FIFO.	R	0x0

For these UARTs writing is only intended to be used for factory testing which is why the VPB UART does not support writing this register.

Bytes are transferred from THR to TSR as soon as 50% of the start bit has been transmitted by the TSR. The LSR.THRE bit is updated as soon as a byte has been transferred from THR to TSR or if a byte is written into the THR. The LSR.TEMT bit is updated as soon as 50% of the first stop bit has been transmitted or if a byte is written into the THR.

* In Nexperia Home Platform compliant systems SW should not use the BI/FE/PE/OE bits in the LSR since these are sensitive to speculative read operations. The corresponding INTSS bits should be used instead. Reading LSR, IIR or INTSS will not have side effects on the value of the bits in INTSS.

28.4.2.9 Modem status Register

The Modem Status Register (MSR) is on byte address 0x018. The register is read-only. Table 14 lists the bit definition of the register.

Table 14: Modem Status Register bit definition

Bit Range	Variable Name	Description	R/W/RW	Reset Value
31:8	reserved	reserved for future use	R	0x0
7	DCD	Data Carrier DetectNormally this bit is the complement of the dcd_an input. In the loop-back mode this bit is equivalent to the OUT2 bit in the MCR register.	R	0x0
6	RI	Ring IndicatorNormally this bit is the complement of the ri_an input. In the loop-back mode this bit is equivalent to the OUT1 bit in the MCR register.	R	0x0
5	DSR	Data Set ReadyNormally this bit is the complement of the dsr_an input. In loop-back mode this bit is equivalent to the DTR bit in the MCR register.	R	0x0

Table 14: Modem Status Register bit definition

Bit Range	Variable Name	Description	R/W/RW	Reset Value
4	CTS	Clear To Send CTS functions as hardware flow control signal input if it is enabled. Flow control (when enabled) allows starting and stopping the transmissions based on the external modem CTS signal. A logic 1 at the CTS pin will stop the UART transmissions as soon as current character has finished transmission.Normally CTS is the complement of the cts_an input. However, in the loop-back mode, this bit is equivalent to the RTS bit in the MCR register.	R	0x0
3	DDCD	Delta Data Carrier Detect. This bit is set as soon as DCD changes its value. The bit is cleared by a MSR read.	R*	0x0
2	TERI	Trailing Edge Ring Indicator. This bit is set as soon as RI transitions from a HIGH to a LOW value. The bit is cleared by a MSR read.	R *	0x0
1	DDSR	Delta Data Set Ready. This bit is set as soon as DSR changes it value. The bit is cleared by a MSR read.	R *	0x0
0	DCTS	Delta Clear To Send. This bit is set as soon as CTS changes its value. The bit is cleared by a MSR read.	R *	0x0

Note that in some industry standard UARTs the MSR register is read/writable. For these UARTs writing is only intended to be used for factory testing which is why the VPB UART does not support writing this register.

The modem status register is only available if the par_sc3106_modem configuration parameter value is set to ‘full’ or ‘ctsrts’. In case the parameter value is ‘no’ reading the register will return 0x0.

* In Nexperia Home Platform compliant systems SW should not use the DDCD/TERI/DDSR/DCTS bits in the MSR since these are sensitive to speculative read operations. The corresponding INTSS bits should be used instead. Reading MSR does not affect the value of the bits in INTSS.

28.4.2.10 Scratch Register

The Scratch Register (SCR) register is on byte address 0x01C. The register can be read and written. Table 15 lists the bit definition of the register.The scratch register is not used by the UART it can be used by software as a temporary storage.

Table 15: Scratch Register bit definition

Bit Range	Variable Name	Description	R/W/RW	Reset Value
31:8	reserved	reserved for future use	R	0x0
7:0	SCRVal	Scratch value	RW	0x0

28.4.2.11 Auto-baud Control Register

The Auto-baud Control Register (ACR) is on byte address 0x020. The registerbits can be partly read and written. Table 16 lists the bit definition of the register.

Table 16: Auto-baud Control Register bit definition

Bit Range	Variable Name	Description	R/W/RW	Reset Value
31:8	reserved	reserved for future use	R	0x0
9[1]	ABTOIntClr	Auto-baud time-out interrupt clear. Writing a 1 will clear the corresponding interrupt in the IIR.	W	0x0
8[1]	ABEOIntClr	End of auto-baud interrupt clear. Writing a 1 will clear the corresponding interrupt in the IIR.	W	0x0
7:3	reserved	reserved for future use	R	0x0
2	AutoRestart	Automatic restart mode: 0 - no restart 1 - restart in case of time-out (counter restarts at next sin_a falling edge)	RW	0x0
1	Mode	Auto-baud mode: 0 - mode 0 1 - mode 1	RW	0x0
0	Start	Auto-baud run 0 - auto-baud stop / auto-baud not running 1 - auto-baud start / auto-baud running the bit automatically clears after auto-baud completion	RW	0x0

[1] ACR[9:8] should be explicitely written with '0' to reset the bit

28.4.2.12 IrDA Control Register

The IrDA Control Register (ICR) is on byte address 0x024. The register can be read and written. Table 17 lists the bit definition of the register.

Table 17: IrDA Control Register bit definition

Bit Range	Variable Name	Description	R/W/RW	Reset Value
31:8	reserved	reserved for future use	R	0x0
5:3	PulseDiv	Configures of pulse in fixed pulse width mode. Only relevant if FixPulseEn is set.	RW	0x0
2	FixPulseEn	Enables IrDA fixed pulse width mode	RW	0x0
1	IrDAInv	If true the serial input is inverted; if false the input is not inverted. The serial output is not affected by the value of this bit.	RW	0x0
0	IrDAEn	If true, enable IrDA; if false disable IrDA and pass UART sin/sout transparently.	RW	0x0

The value of the ICR should not be modified while transmitting/receiving data or data may be lost or corrupted.

The IrDA Control Register is only available if the par_sc3106_irda configuration parameter value is set to ‘yes’. In case the parameter value is ‘no’ reading the register will return 0x0.

The IrDA.PulseDiv bits are used for configuring the pulse width of the fixed pulse width mode of the UART. The value of these bits should be configured such that the resulting pulse width is at least 1.63 us. Table 18 lists the PulseDiv values and their corresponding pulse width values

Table 18: IrDA pulse width

IrDA.FixPulseEn	IrDA.PulseDiv	IrDA transmitter pulse width [us]
0	x	3 / (16 * baud-rate)
1	0	2 * T_{u_clk}
1	1	4 * T_{u_clk}
1	2	8 * T_{u_clk}
1	3	16 * T_{u_clk}
1	4	32 * T_{u_clk}
1	5	64 * T_{u_clk}
1	6	128 * T_{u_clk}
1	7	256 * T_{u_clk}

28.4.2.13 Fractional Divider Register

The Fractional Divider Register (FDR) is on byte address 0x028. The register can be read and written. Table 19 lists the bit definition of the register.

Table 19: Fractional Divider Register bit definition

Bit Range	Variable Name	Description	R/W/RW	Reset Value
31:8	reserved	reserved for future use	R	0x0
7:4	MulVal	Baud-rate pre-scaler multiplier value	RW	0x1
3:0	DivAddVal	Baud-rate generation pre-scaler divisor value	RW	0x0

The FDR register controls the clock pre-scaler for the baud rate generation. The clock can be pre-scaled by a value of MulVal/(MulVal+DivAddVal). The value of MulVal and DivAddVal should comply to the following expressions:

0 < MulVal <= 15, 0 <= DivAddVal <= 15

If the register value does not comply to the above expression, then the fractional divider output is undefined. If DivAddVal is zero then the fractional divider is disabled and the clock will not be divided.

The value of the FDR should not be modified while transmitting/receiving data or data may be lost or corrupted.

28.4.2.14 NHP Pop Register

The NHP Pop Register (POP) is located on byte address 0x030. The POP register is a write-only register. Reading the register will return 0x0. Table 20 lists the bit definition of the register.

Table 20: NHP Pop Register bit definition

Bit Range	Variable Name	Description	R/W/RW	Reset Value
31:1	reserved	reserved for future use	W	0x0
0	PopRBR	Setting this bit will pop the first item from the Receiver Buffer Register's FIFO as if RBR were read in non NHP mode. The bit will clear automatically.	W	0x0

In 'NHP mode' which protects the UART from speculative reads the PopRBR bit in the POP register can be used to pop the first data element from the RBR FIFO as if the RBR register were read in non 'NHP mode'.

Typically in a NHP mode application each RBR read operation is followed by writing the PopRBR bit to remove the top read data from the FIFO.

28.4.2.15 NHP Mode Selection Register

The NHP Mode Selection Register (MODE) is located on byte address 0x034. The MODE register is a read/write register. Table 21 lists the bit definition of the register.

Table 21: NHP Pop Register bit definition

Bit Range	Variable Name	Description	R/W/RW	Reset Value
31:1	reserved	reserved for future use	RW	0x0
0	NHP	Setting this bit will switch the UART in 'NHP mode' and protect the UART RBR from speculative reads; RBR needs to be popped explicitly via the POP register. Intreq is derived from INTS instead of IIR. Clearing the bit will switch the UART in 'x50 mode	RW	0x0

After reset the UART will be in non NHP compliant, normal 'x50 compliant mode in which read operations will have side effects. After setting the NHP bit the UART will be in NHP compliant mode which will have two implications:

- RBR read operations will have no side effects. LSR, IIR and MSR reads will still have side effects, in NHP mode the speculative read sensitive bits in the LSR/IIR/MSR registers should not be used by SW. Instead SW should use the INTS register for determining the state of modem, line and interrupt.
- In NHP mode the uart_intreq output will be derived from the INTS register instead of the IIR register.

28.4.2.16 Configuration Register

The Configuration Register (CFG) is on byte address 0xFD4. The register is read-only. Table 23 lists the bit definition of the register.

Table 22: Configuration Register bit definition

Bit Range	Variable Name	Description	R/W/RW	Reset Value
31:13	reserved	reserved for future use	R	0x0
12	HasIrDA	Value reflects the par_sc3106_irda_include parameter value during extraction.	R	no: 0x0 yes: 0x1
11:10	reserved	reserved for future use	R	0x0
9	HasLevel	Value reflects the par_sc3106_fifo_level parameter value during extraction.	R	no: 0x0 yes: 0x1
8	HasDMA	Value reflects the par_sc3106_dma_interface parameter value during extraction.	R	no: 0x0 yes: 0x1
7:6	reserved	reserved for future use	R	0x0

Table 22: Configuration Register bit definition

Bit Range	Variable Name	Description	R/W/RW	Reset Value
5:4	Modem	Value reflects the par_sc3106_modem parameter value during extraction.	R	no: 0x0 rtscts: 0x1 full: 0x2
3:2	reserved	reserved for future use	R	0x0
1:0	Type	Value reflects the par_sc3106_type parameter value during extraction.	R	450: 0x0 550: 0x1 650: 0x2 750: 0x3

The CFG register reflects the hardware configuration as extracted from the IPYP. Typically the register is used for run-time configuration of the software driver.

Note: The UART contains in principle the full modem interface at block level, so when you read CFG.Modem then you will read a 0x2. But this value is not true at chip level, because at chip level only cts/rts modem interface is available. So no full modem interface is supported.

28.4.2.17 Interrupt Clear Enable Register

The Interrupt Clear Enable Register (INTCE) is on byte address 0xFD8. The register is write-only. Table 23 lists the bit definition of the register.

Table 23: Interrupt Clear Enable Register bit definition

Bit Range	Variable Name	Description	R/W/RW	Reset Value
31:16	reserved	reserved for future use	W	0x0
15	OEIntEnClr	Overrun Error Interrupt Enable Clear	W	0x0
14	PEIntEnClr	Parity Error Interrupt Enable Clear	W	0x0
13	FEIntEnClr	Frame Error Interrupt Enable Clear	W	0x0
12	BIIntEnClr	Break Indication Interrupt Enable Clear	W	0x0
11:10	reserved	reserved for future use	W	0x0
9	ABTOIntEnClr	Auto-Baud Time-Out Interrupt Enable Clear	W	0x0
8	ABEOIntEnClr	End of Auto-Baud Interrupt Enable Clear	W	0x0
7	reserved	reserved for future use	W	0x0
6	RxDAIntEnClr	Receiver Data Available Interrupt. Enable Clear	W	0x0

Table 23: Interrupt Clear Enable Register bit definition

Bit Range	Variable Name	Description	R/W/RW	Reset Value
4	THREIntEnClr	Transmitter Holding Register Empty Interrupt Enable Clear	W	0x0
3	DDCDIntEnClr	Delta Carrier Detect Interrupt Enable Clear	W	0x0
2	TERIIntEnClr	Trailing Edge Ring Indicator Interrupt Enable Clear	W	0x0
1	DDSRIntEnClr	Delta Data Set Ready Interrupt Enable Clear	W	0x0
0	DCTSIntEnClr	Delta Clear To Send Interrupt Enable Clear	W	0x0

Note: The register bits are one-shot registers and automatically cleared.

28.4.2.18 Interrupt Set Enable Register

The Interrupt Set Enable Register (INTSE) is on byte address 0xFDC. The register is write-only.The register is write-only. Table 24 lists the bit definition of the register.

Table 24: Interrupt Set Enable Register bit definition

Bit Range	Variable Name	Description	R/W/RW	Reset Value
31:16	reserved	reserved for future use	W	0x0
15	OEIntEnSet	Overrun Error Interrupt Enable Set	W	0x0
14	PEIntEnSet	Parity Error Interrupt Enable Set	W	0x0
13	FEIntEnSet	Frame Error Interrupt Enable Set	W	0x0
12	BIIntEnSet	Break Indication Interrupt Enable Set	W	0x0
11:10	reserved	reserved for future use	W	0x0
9	ABTOIntEnSet	Auto-Baud Time-Out Interrupt Enable Set	W	0x0
8	ABEOIntEnSet	End of Auto-Baud Interrupt Enable Set	W	0x0
7	reserved	reserved for future use	W	0x0
6	RxDAIntEnSet	Receiver Data Available Interrupt. Enable Set	W	0x0
5	RxTOIntEnSet	Receiver Time-Out Interrupt Enable Set	W	0x0

Table 24: Interrupt Set Enable Register bit definition

Bit Range	Variable Name	Description	R/W/RW	Reset Value
4	THREIntEnSet	Transmitter Holding Register Empty Interrupt Enable Set	W	0x0
3	DDCDIntEnSet	Delta Carrier Detect Interrupt Enable Set	W	0x0
2	TERIIntEnSet	Trailing Edge Ring Indicator Interrupt Enable Set	W	0x0
1	DDSRIntEnSet	Delta Data Set Ready Interrupt Enable Set	W	0x0
0	DCTSIntEnSet	Delta Clear To Send Interrupt Enable Set	W	0x0

Note: The register bits are one-shot registers and automatically cleared.

28.4.2.19 Interrupt Status Register

The Interrupt Status Register (INTS) is on byte address 0xFE0. The register is read-only. Table 25 lists the bit definition of the register.

Table 25: Interrupt Status Register bit definition

Bit Range	Variable Name	Description	R/W/RW	Reset Value
31:16	reserved	reserved for future use	R	0x0
15	OEInt	Overrun Error Interrupt. Set if RBR overruns. Cleared by setting INTCS.OEIntClr	R	0x0
14	PEInt	Parity Error Interrupt. Set if top of RBR has parity error. Cleared by popping RBR.	R	0x0
13	FEInt	Frame Error Interrupt. Set if top of RBR has framing error. Cleared by popping RBR.	R	0x0
12	BIInt	Break Indication Interrupt. Set if top of RBR has break indication. Cleared by popping RBR.	R	0x0
11:10	reserved	reserved for future use	R	0x0
9	ABTOInt	Auto-Baud Time-Out Interrupt. Set on auto-baud time-out. Cleared by setting INTCS.ABTOIntClr	R	0x0

Table 25: Interrupt Status Register bit definition

Bit Range	Variable Name	Description	R/W/RW	Reset Value
8	ABEOInt	End of Auto-Baud Interrupt. Set at end of auto-baud. Cleared by setting INTCS.ABEOIntClr.	R	0x0
7	reserved	reserved for future use	R	0x0
6	RxDAInt	Receiver Data Available Interrupt. Cleared by popping RBR below FIFO level.	R	0x0
5	RxTOInt	Receiver Time-Out Interrupt. Cleared by popping RBR, receiving new character or setting the INTCS.RxTOIntClr bit.	R	0x0
4	THREInt	Transmitter Holding Register Empty Interrupt. Set if THR is empty. Cleared by writing THR or setting INTCS.THREIntClr	R	0x0
3	DDCDInt	Delta Carrier Detect Interrupt. Set on change of DCD. Cleared by setting INTCS.DDCSIntClr.	R	0x0
2	TERIInt	Trailing Edge Ring Indicator Interrupt. Set on falling edge of RI. Cleared by setting INTC.TERIIntClr.	R	0x0
1	DDSRInt	Delta Data Set Ready Interrupt. Set on change of DSR. Cleared by setting INTCS.DDSRIntClr.	R	0x0
0	DCTSInt	Delta Clear To Send Interrupt. Set on change of CTS. Cleared by setting INTCS.DCTSIntClr.	R	0x0

Only a limited number of bits from the NHP Interrupt Status Register can be set and cleared by software. The INTS.RxDAInt/PEInt/FEInt/BIInt bits cannot be set/cleared from SW. These bits are controlled by HW.

28.4.2.20 Interrupt Enable Register

The Interrupt Enable Register (INTE) is on byte address 0xFE4. The register is read-only. Table 26 lists the bit definition of the register.

Table 26: Interrupt Enable Registerl bit definition

Bit Range	Variable Name	Description	R/W/RW	Reset Value
31:16	reserved	reserved for future use	R	0x0
15	OEIntEn	Overrun Error Interrupt Enable	R	0x0
14	PEIntEn	Parity Error Interrupt Enable	R	0x0
13	FEIntEn	Frame Error Interrupt Enable	R	0x0
12	BIIntEn	Break Indication Interrupt Enable	R	0x0
11:10	reserved	reserved for future use	R	0x0
9	ABTOIntEn	Auto-Baud Time-Out Interrupt Enable	R	0x0
8	ABEOIntEn	End of Auto-Baud Interrupt Enable	R	0x0
7	reserved	reserved for future use	R	0x0
6	RxDAIntEn	Receiver Data Available Interrupt. Enable	R	0x0
5	RxTOIntEn	Receiver Time-Out Interrupt Enable	R	0x0
4	THREIntEn	Transmitter Holding Register Empty Interrupt Enable	R	0x0
3	DDCDIntEn	Delta Carrier Detect Interrupt Enable	R	0x0
2	TERIIntEn	Trailing Edge Ring Indicator Interrupt Enable	R	0x0
1	DDSRIntEn	Delta Data Set Ready Interrupt Enable	R	0x0
0	DCTSIntEn	Delta Clear To Send Interrupt Enable	R	0x0

28.4.2.21 Interrupt Clear Status Register

The Interrupt Enable Register (INTE) is on byte address 0xFE4. The register is read-only. Table 26 lists the bit definition of the register. Table 27 lists the bit definition of the register.

Table 27: Interrupt Clear Status Registerl bit definition

Bit Range	Variable Name	Description	R/W/RW	Reset Value
31:16	reserved	reserved for future use	W	0x0
15	OEIntClr	Overrun Error Interrupt Clear	W	0x0
14:10	reserved	reserved for future use	W	0x0

Table 27: Interrupt Clear Status Registerl bit definition

Bit Range	Variable Name	Description	R/W/RW	Reset Value
9	ABTOIntClr	Auto-Baud Time-Out Interrupt Clear. Alias of the ACR.ABTOIntClr bit	W	0x0
8	ABEOIntClr	End of Auto-Baud Interrupt Clear. Alias of the ACR.ABEOIntClr bit	W	0x0
7:6	reserved	reserved for future use	W	0x0
5	RxTOIntClr	Receiver Time-Out Interrupt Clear	W	0x0
4	THREIntClr	Transmitter Holding Register Empty Interrupt Clear	W	0x0
3	DDCDIntClr	Delta Carrier Detect Interrupt Clear	W	0x0
2	TERIIntClr	Trailing Edge Ring Indicator Interrupt Clear	W	0x0
1	DDSRIntClr	Delta Data Set Ready Interrupt Clear	W	0x0
0	DCTSIntClr	Delta Clear To Send Interrupt Clear	W	0x0

Note: The register bits are one-shot registers and automatically cleared.

28.4.2.22 Interrupt Set Status Register

The Interrupt Set Status Register (INTSS) is on byte address 0xFEC. The register is write-only. Table 28 lists the bit definition of the register.

Table 28: Interrupt Clear Status Registerl bit definition

Bit Range	Variable Name	Description	R/W/RW	Reset Value
31:16	reserved	reserved for future use	W	0x0
15	OEIntSet	Overrun Error Interrupt Set	W	0x0
14:10	reserved	reserved for future use	W	0x0
9	ABTOIntSet	Auto-Baud Time-Out Interrupt Set	W	0x0
8	ABEOIntSet	End of Auto-Baud Interrupt Set	W	0x0
7:6	reserved	reserved for future use	W	0x0
5	RxTOIntSet	Receiver Time-Out Interrupt Set	W	0x0
4	THREIntSet	Transmitter Holding Register Empty Interrupt Set	W	0x0
3	DDCDIntSet	Delta Carrier Detect Interrupt Set	W	0x0

Table 28: Interrupt Clear Status Registerl bit definition

Bit Range	Variable Name	Description	R/W/RW	Reset Value
2	TERIIntSet	Trailing Edge Ring Indicator Interrupt Clear.	W	0x0
1	DDSRIntSet	Delta Data Set Ready Interrupt Set	W	0x0
0	DCTSIntSet	Delta Clear To Send Interrupt Set	W	0x0

Note: The register bits are one-shot registers and automatically cleared.

28.4.2.23 Module Identification Register

The Module Identification Register (MID) is on byte address 0xFFC. The register is read-only. Table 29 lists the bit definition of the register.

Table 29: Configuration Register bit definition

Bit Range	Variable Name	Description	R/W/RW	Reset Value
31:16	ID	Unique identification number, assigned by RTG	R	0x3106
15:12	MajRev	Major revision, starting at 0. No guaranteed software compatibility between major revisions.	R	Implement. Specific
11:8	MinRev	Minor revision, starting at 0. Guaranteed software compatibility between minor revisions.	R	Implement. Specific
7:0	Aperture	Aperture encoded as (aperture/size 4K)-1, so 0x00 means a 4K aperture.	R	0x0

The MID register uniquely identifies the HW version of the UART. Major and minor revision number will change between HW versions. Typically the register is used for run-time configuration of the software driver.

28.4.2.24 Divisor Latch LSB

The Divisor Latch LSB register (DLL) is on byte address 0x000 and can only be accessed if DLAB = 1. The DLL register can be read and written. If DLAB = 0 reading from and writing to this address will access RBR and THR. Table 30 lists the bit definition of the register.

Table 30: Divisor Latch LSB bit definition

Bit Range	Variable Name	Description	R/W/RW	Reset Value
31:8	reserved	reserved for future use	R	0x0
7:0	DLLVal	Least significant byte of the divisor latch value	RW	0x1

The divisor value is 16 bits of which the most significant bits are stored in the DLM register and the least significant bits are defined in the DLL register.

The value of the DLL should not be modified while transmitting/receiving data or data may be lost or corrupted.

28.4.2.25 Divisor Latch MSB

The Divisor Latch MSB register (DLM) is on byte address 0x004 and can only be accessed if DLAB = 1. The DLM register can be read and written. If DLAB = 0 reading and writing this address will access the IER register. Table 31 lists the bit definition of the register.

Table 31: Divisor Latch MSB bit definition

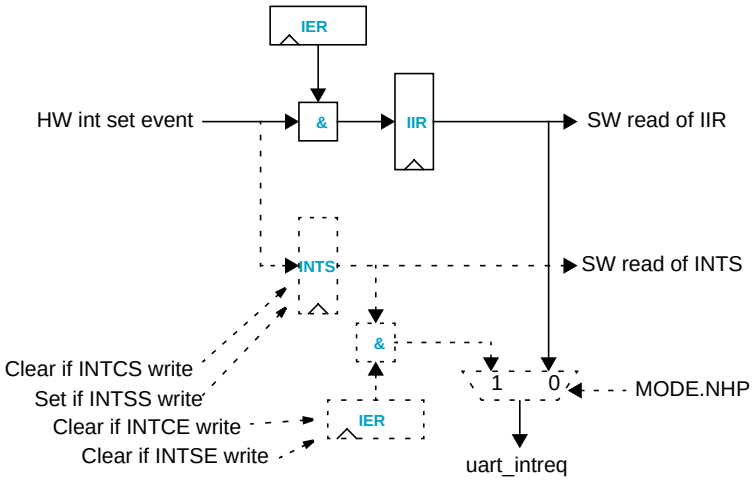
Bit Range	Variable Name	Description	R/W/RW	Reset Value
31:8	reserved	reserved for future use	R	0x0
7:0	DLMVal	Least significant byte of the divisor latch value	RW	0x1

The value of the DLM should not be modified while transmitting/receiving data or data may be lost or corrupted.

28.4.3 PROTOCOLS

The UART has two interrupt modes. If the UART is configured as NHP compliant mode by setting the MODE.NHP bit interrupt will comply to the CoReUse interrupt standard. If the UART is not configured for NHP compliance or if the MODE.NHP bit is cleared the UART is in standard ‘x50 mode and interrupts should be handled via the IIR/IER registers.

Figure 2: Two interrupt mechnisms



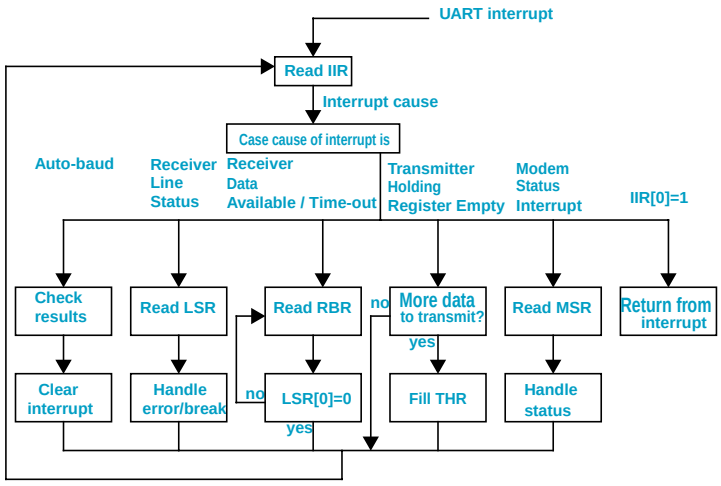
UART supports two interrupt mechnisms selectable via the MODE.NHP bit. The standard ‘x50 mechanism (not dotted lines in figure 2) and the CoReUse/NHP compliant interrupt mechnism (dotted lines, in figure 2). So when MODE.NHP bit is 1, UART is configured as NHP compliant mode then the interrupt is generated in the CoReUse/NHP compliant way.

Note that the interrupt enables in both mechnisms are different. In the ‘x50 standard disabling an interrupt in the IER will not set the interrupt in the IIR register and not propagate the interrupt to the interrupt output pin. In the CoReUse/NHP interrupt architecture disabling an interrupt in the INTE will still set the interrupt in the INTS register but not propagate the interrupt to the interrupt output pin.

28.4.3.1 Standard ‘x50 interrupt handling

This section defines the typical Interrupt Service Routine for non Nexperia Home platform compliant systems i.e. when MODE.NHP bit is not set. Fig 3. depicts a flow diagram for a typical UART SW driver Interrupt Service Routine (ISR).

Figure 3: Standard interrupt service routine



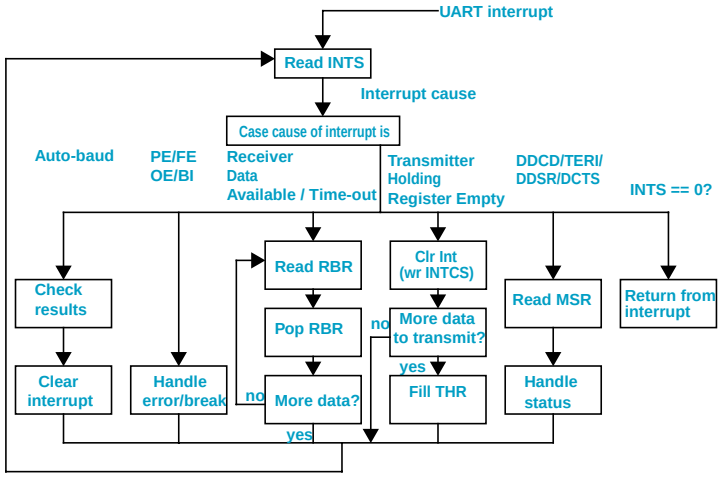
The interrupt routine starts with reading the Interrupt Identification Register to determine the cause of the interrupt. Depending on the cause of the interrupt a different branch in the ISR is taken. Taking away the cause of the interrupt will automatically clear the interrupt condition in the IIR. After taking away the interrupt cause the ISR loops back to the IIR read to process any pending lower level interrupts. Once all interrupts are processed (IIR.IntStatus = 1) the ISR can return from the interrupt.

Interrupts can be enabled and disabled by setting and clearing the appropriate bits in the IER register.

28.4.3.2 Nexperia Home Platorm compliant interrupt handlingI

Figure 4 depicts a flow diagram for a typical Nexperia Home UART SW driver Interrupt Service Routine (ISR)

Figure 4: Standard interrupt service routine



The interrupt routine starts with reading the NHP Interrupt Status Register to determine the cause of the interrupt. Multiple interrupts can be pending. Depending on the cause of the interrupt a different branch in the ISR is taken. In some cases taking away the cause of the interrupt will automatically clear the interrupt condition in the INTS in other cases the interrupt has to be cleared explicitly by SW. After clearing and handling the interrupt the ISR loops back to the INTS read to process any pending lower level interrupts. Once all interrupts are processed (INTS == 0) the ISR can return from the interrupt.

28.4.3.3 Initialization

Before receiving and transmitting characters the UART interface needs to be initialized. The baud-rate has to be specified writing the FDR, DLL and DLM registers. The values of the FDR, DLL and DLM registers should not be modified while transmitting/receiving data or data may be lost or corrupted.

In case the IrDA option is selected the IrDA needs to be configured by programming the ICR. The value of the ICR should not be modified while transmitting/receiving data or data may be lost or corrupted.

In case FIFO mode is used the FIFO needs to be enabled and the appropriate FIFO trigger levels need to be set in the FCR. Optionally DMA mode can be enabled to use a HW DMA engine for reading receive data and writing transmit data..

The number of bits per character, parity and stop bits need to be selected in the LCR. Apart from the BrkCtrl bit the value of the LCR register should not be modified while transmitting/receiving data or data may be lost or corrupted.

In case the UART has a modem interface this needs to be configured in the MCR.

In case of a Nexperia Home Platform compliant system the MODE.ReadProtect bit needs to be set.

29 LCD INTERFACE (SMALL CHANGES)

29.1 Overview

The LCD interface contains logic to interface to a 6800/8080 compatible LCD controller. The LCD interface is compatible with the 6800 bus standard and the 8080 bus standard, with one address pin (RS) for selecting the data or instruction register.

The LCD interface makes use of a configurable clock (programmed in the PMU) to adjust the speed of the 6800/8080 bus to the speed of the connected peripheral.

29.2 Functional Description

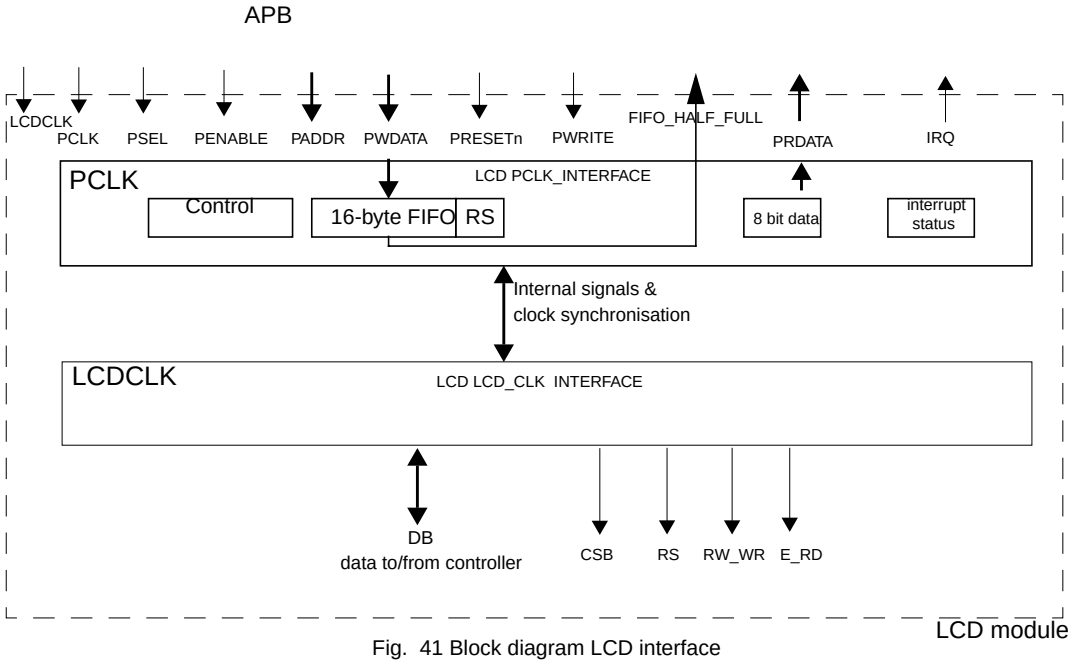


Fig. 41 Block diagram LCD interface

29.2.1 INTERFACE

- 8/4 bit parallel interface mode: 6800-series, 8080-series
- Supports multiple frequencies for the 6800/8080 bus, to support high and low speed controllers
- Supports polling the busy flag from LCD controller to off load the CPU from polling
- Contains an 16 byte FIFO for sending control and data information to the LCD controller
- Contains a serial interface which uses the same FIFO for serial transmissions.
- Contains maskable interrupts.
- Connects to the AMBA APB 2.0 bus, VPB 2-cycle or VPB 3-cycle bus.
- Supports FIFOLEVEL flow control towards the Simple DMA controller.

29.2.2 SYSTEM INTERFACE

Table 222 Various modes of the LCD interface

PS	MI	IF	CSB	RS	RW_WR	E_RD	DB0-3	DB4-	DB5	DB6	DB7
Bus mode (L)	6800-series (H)	8 bit (L)	CSB	RS	nR/W	E	DB0-3	DB4	DB5	DB6	DB7
		4 bit (H)	CSB	RS	nR/W	E	*	DB4	DB5	DB6	DB7
	8080-series (L)	8 bit (L)	CSB	RS	nWR	nRD	DB0-3	DB4	DB5	DB6	DB7
		4 bit (H)	CSB	RS	nWR	nRD	*	DB4	DB5	DB6	DB7
Serial mode (H)	*	*	CSB	RS	*	*	*	*	SCL	SI	SO

Note:

1. * Don’t care (“High”, ”Low” or “Open”)

PS = Parallel/Serial mode
CSB = Chip Select. Default low active
IF = 4 or 8-bit mode
MI = Motorola/Intel mode
RS = Register Select (also seen as A0)
E_RD = Enable / Read. Enable in 6800 mode, Read in 8080 mode.
RW_WR = ReadWrite / WRITE. Read/write in 6800 mode, Write in 8080 mode.
DB(7-0) = Data Bus.
SCL = Serial CLock.
SI = Serial Input.
SO = Serial Output.

29.2.3 TIMING DIAGRAM

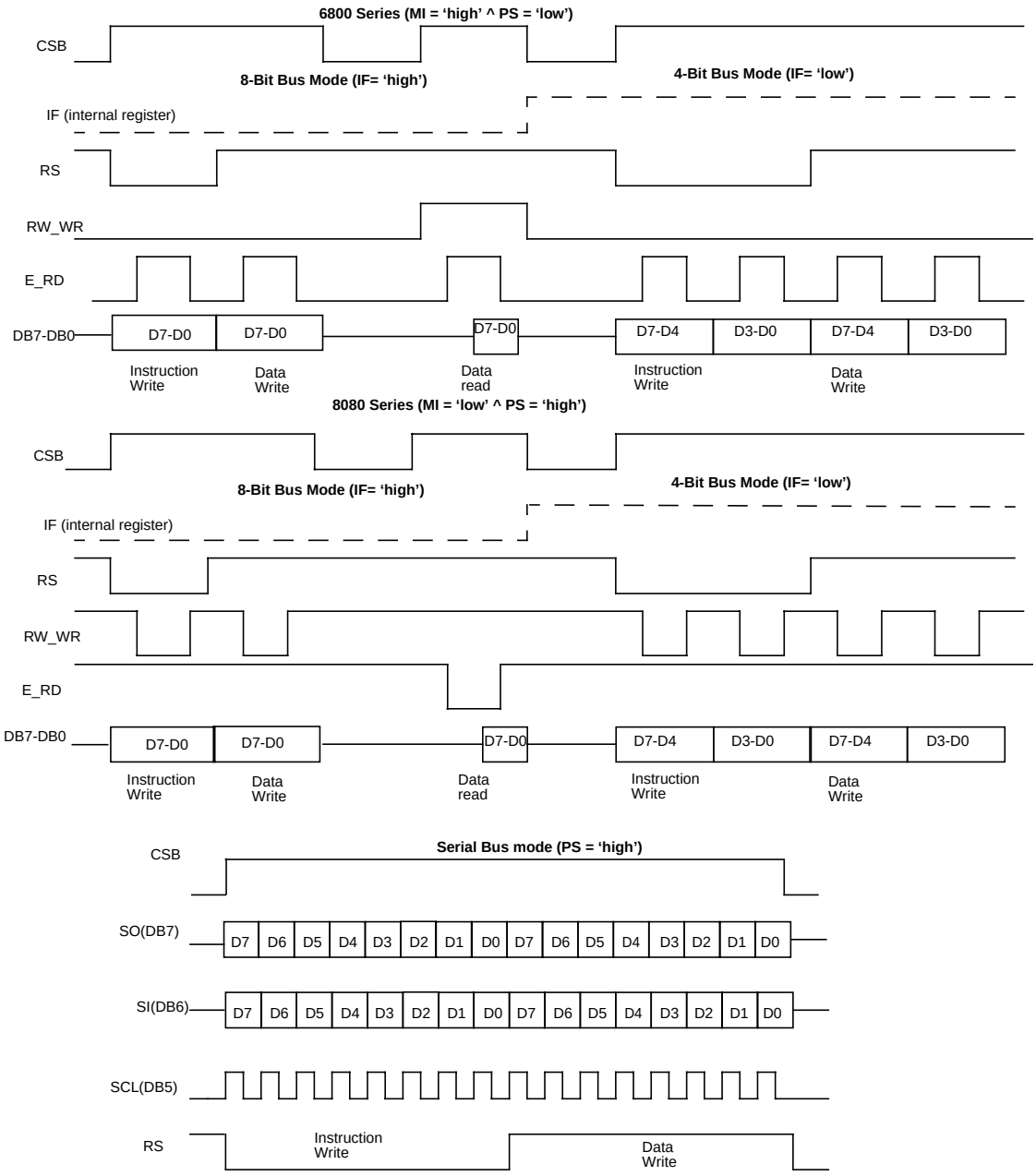


Fig. 42 Timing

29.2.4 RESETTING THE EXTERNAL LCD CONTROLLER

A GPIO pin or the system reset can be used to act as a reset signal for the external LCD controller.

In some cases a simple instruction to the controller is enough to perform the reset.

29.2.5 THE WRITE FIFO

A 16 byte write FIFO (First In First Out memory) is implemented in the LCD interface which will store both instructions and data. This way the CPU can write multiple bytes or words to the LCD interface in stead of just one. Now the CPU doesn't have to wait for the LCD controller to finish, for a new write action.

Information to the LCD controller can be written in the LCD_instruction_out or LCD_data_out register. If the total amount of data written is bigger then the FIFO can store, the 'FIFO_overrun' interrupt will be set, and the last byte or word will not be written into the FIFO.

If a word is written into the FIFO, the LSB byte (bits 7-0) will be put on the data pins first.

A status pin indicating that the FIFO is at least half empty is connected at top-level, so the Simple DMA controller can decide when it is allowed to transfer data and when it is not allowed.

29.2.6 OPERATIONAL MODES

The LCD_interface has three modes for outputting data: Byte-mode, 4-bit mode and serial-mode.

Byte mode:

At each shift of the FIFO, the last byte inside the FIFO will be put on the data pins, and pin RS will indicate if the data is an instruction or data value.
In read mode the data on pins DB_IN 7-0 will be sampled by the LCD_interface.

4-bit mode:

At each shift of the FIFO, the last byte from the FIFO will be split, where the order depends on the 'MSB_first' from the control register.
When set to '1', bit 7-4 from the FIFO byte will be put first, or read first, at the data pins, and then bit 3-0.
When set to '0' bits 3-0 will be written or read first, and then bits 7-4.

Serial mode:

At each shift of the FIFO the last FIFO byte will be split in 8 seperate bits and be put on data pin 7, where the bit-order depends on the 'MSB_first' from the control register register.
When set to '0', then first bit 0 and last bit 7 will be written/read first, else the order is from 7 downto 0.
Signal RS is included for each 8 bits and indicates a instruction or data. Not all controllers require this signal in serial mode, but can be used if required.

29.2.7 WRITING DATA

There are 2 addresses on the 6800/8080 bus. The can be addressed, by using the following registers:

LCD INST_OUT

Writing to one of these registers will add the contents to the FIFO, tagged as instruction.

- To write a word into the FIFO: write the word to address: LCD_instruction_word_out*. The LSB byte of the word will be transmitted first out of the FIFO.
- To write a byte into the FIFO: write the byte to address: LCD_instruction_byte_out*

*NOTE: The separation of bytes and words is necessary, because the APB assumes all data to be 32 bit wide.

LCD_DATA_OUT

Writing to these registers will add the contents to the FIFO, tagged as data.

To write a word into the FIFO: write the word to address: LCD_data_word_out. The LSB byte of the word will be transmitted first out of the FIFO.
To write a byte into the FIFO: write the byte to address: LCD_data_byte_out

29.2.8 READING DATA

Reads from the LCD controller are quite rare. Normally only writes are performed to a LCD controller. Therefore a simple single byte register is used to store the byte that has been read from the 6800/8080 bus.

The LCD controller is a slow peripheral. When the CPU requests information from the 6800/8080 bus, it will not be available directly. To read the data register or the instruction register it has to wait a while before the data becomes valid.

The read procedure is as follows:

1. Write the following in the LCD_READ_CMD register:
   '0x0' for initiating a read on INST_BYTE (RS=0)
   '0x1' for initiating a read on DATA_BYTE (RS=1)
2. -Wait until an IRQ arrives, and check the 'LCD_read_valid' bit of the STATUS register
   OR
   -keep polling the LCD_INTERFACE_BUSY bit of the STATUS register. '0' means valid.
3. If the returned value is valid, the byte can be read from the LCD_INST_BYTE or LCD_DATA_BYTE register. A write to LCD_READ_CMD will initiate a new read on the LCD_bus.

If the LCD_INST_BYTE or LCD_DATA_BYTE register is read, the LCD_INTERFACE_BUSY bit of the CONTROL register stays logical '0' until a new read is started.

29.2.9 COMBINED WRITING AND READING DATA

A read operation can be performed at any time. The read-request is stored (queued), and as soon as the write FIFO is empty, the read operation will be executed. Only one read request is stored, so if there were multiple read requests (which is considered invalid), the last one will be executed.

Note: A write operation after a read operation which is still queued or has not completed its cycle is considered invalid. The read operation will be discarded or aborted and the LCD interface will enter or stay (if the LCD interface did not start reading yet) in write mode. The written value is stored in the FIFO as being a regular write transfer, and the LCD interface will proceed normally, without executing the read command.

29.2.10 USING WAIT STATES

The LCD controller does not support waitstates. If the LCD bus needs to made slower, it is best to reduce the LCDCLK speed inside the CGU.

29.2.11 SERIAL MODE

The LCD interface can be put in serial mode by placing bit ‘PS’ of LCD_control register to ‘1’.

29.2.11.1 Serial writes

In this mode, the FIFO will be used to shift the data on a bit-by-bit basis on pin DB7 of the LCD interface. The ‘MSB_first’ bit of the control register controls if the MSB or LSB bit is transmitted first:

‘0’ to select LSB first (first bit 0, then 1,2,3,4,5,6,7).
‘1’ to select if the serial data is transmitted with MSB first (so first bit 7, then 6,5,4,3,2,1,0)

This is depending on how the LCD controller accepts serial information.

If the FIFO runs empty, the LCD_FIFO_empty bit of the raw-status interrupt register will be set, and the serial clock output will be disabled, until new data is available in the FIFO.

The RS output can be used to select the data or instruction register of the LCD controller.

29.2.11.2 Serial reads

Serial reading uses the same commands as parallel reading.

Reading the serial data is done on pin DB6 of the 6800/8080 bus. The ‘MSB_first’ bit controls which bit is sampled first:

‘0’ to select that the first sampled bit will be stored in the LSB bit
‘1’ to select that the first sampled bit will be stored in the MSB bit.

Eight clock cycles are generated on the serial_clock, and the serial input will be sampled for 8 cycles.

Now the information in the LCD_data_in register is valid and can be read.

29.2.11.3 Serial clock timing

The serial output clock ‘SCL’ can be set to four modes to comply to the specifications of the LCD controller. Each mode is a 25% shift of the previous mode.The clock mode is set by writing to the ‘serial_clk_shift’ location of the control register. See Fig. 43.

For reading, register location ‘serial_read_pos’ of the control register determines at which position the data is sampled by the LCD interface. This can be seen in Fig. 43, bottom part.

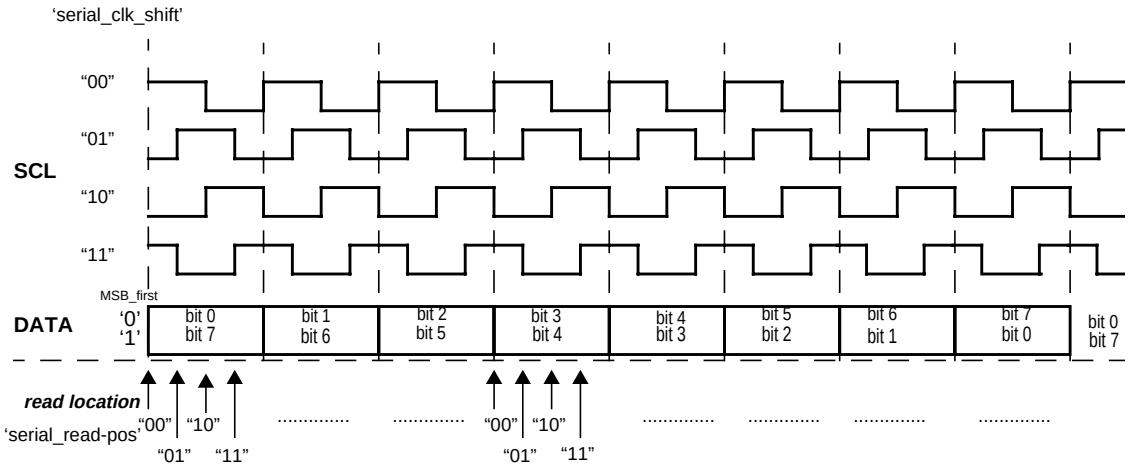


Fig. 43 Serial mode timing

29.2.12 CHECKING THE BUSY FLAG OF THE LCD CONTROLLER

Most LCD controllers contain a status register with a bit that represents the busy flag. If available, this flag has to be polled before a read or write can be performed from or to the LCD controller.

To off-load the CPU from polling the LCD controller before each access, an option is included that the LCD interface takes over this polling for data read, data write and instruction write accesses.

To enable the busy-flag checking, set the following bits in the CONTROL register:

- 'BUSY_FLAG_CHECK' : set to '1' to enable the busy-flag-checking
- 'BUSY_BIT_NR' : Set which bit number has to be checked for the busy flag. Don't care if BUSY_FLAG_CHECK = '0'
- 'BUSY_RS_VALUE' : Set which address (RS value) has to be checked for the busy flag.

  when '0' then register belonging to RS='0' will be checked.
  when '1' the register belonging to RS='1' will be checked.

  Don't care if BUSY_FLAG_CHECK = '0'
- 'BUSY_VALUE' : Set if a logic '0' or '1' represents the busy flag. Don't care if BUSY_FLAG_CHECK = '0'

Note: Reading the register from the LCD controller set by BUSY_RS_VALUE will return the value of that register, without checking for the busy-flag first.

The busy-flag-checking can only be used if one of the two LCD controller registers have a bit that represents the 'busy' value. If this bit is not available, the busy-flag-checking feature has to be disabled and a slower clock has to be used to ensure reading or writing a valid value.

29.2.13 LOOP BACK MODE

Setting the register 'LOOPBACK' of the CONTROL register to '1', will set the LCD interface in loop back-mode.

Internally, the LCD data output is connected to the LCD data input. The programmer can test correct behaviour of the LCD interface, by doing the following:

- Place the LCD interface in parallel, 8-bit mode
- Write a single byte to the LCD_DATA_BYTE register

- Write ‘0x01’ to the LCD_READ_CMD register to request a bus read
- Poll the status bit, or wait for the ‘valid’ interrupt (if MASK is cleared)
- If valid, read the byte from LCD_DATA_BYTE register
- Compare this value with the written value

To ensure correctness of the test, perform the above a couple of times, with different values.

29.3 Registers

Table 223LCD register map

ADDRESS SPACE	OFFSET	REGISTER NAME	R/W ACCESS	RESET
0x8E00_0000 - 0x8E00_00FF	0x00	LCD_STATUS	R	0x00 00 00 10
	0x04	LCD_CONTROL	R/W	0x00 01 C0 F0
	0x08	LCD_INT_RAW	R	0x0
	0x0C	LCD_INT_CLEAR	W	
	0x10	LCD_INT_MASK	R/W	0x00 00 00 0F
	0x14	LCD_READ_CMD	W	
	0x20	LCD_INST_BYTE	R/W	0x0
	0x30	LCD_DATA_BYTE	R/W	0x0
	0x40	LCD_INST_WORD	W	
	0x80	LCD_DATA_WORD	W	

LCD STATUS

Read only. This register stores the interrupt status, the busy bit and the FIFO counter value. They are described in Table 224

LCD_CONTROL

Read/write. This register stores the control bits used by the LCD interface. They can be found in Table 225

LCD_INT_RAW

This register contains the status of the interrupts, without any masking.

LCD_INT_CLEAR
Write only. Writing to this register clears the selected interrupts.

LCD_INT_MASK

Read/write. This register contains the masking information for the interrupt. If a mask bit equals to ‘1’ than that specific interrupt won’t be used as a source for the IRQ to the CPU.

LCD_READ_CMD

Write only. Writing to this register will result in a read operation on the LCD_interface bus. A write to this register during a read operation will trigger a new read, and will discard the old.

writing ‘0x00’ will result in a read on the INST_BYTE
Writing ‘0x01’ will result in a read in DATA_BYTE

If a read is finished (valid) the byte can be read in either LCD_INST_BYTE or LCD_DATA_BYTE

LCD_INST_BYTE

read/write. Writing to this register will write one byte into the FIFO, tagged as instruction.
Reading from this register will return the data read from the LCD controller. The data can be considered valid if the interrupt: LCD_INT_DATA_VALID occurred, or if bit 'LCD_INTERFACE_BUSY' reads '0'.

LCD_DATA_BYTE

Read/write. Writing to this register will write one byte into the FIFO, tagged as data.
Reading from this register will return the data read from the LCD controller. The data can be considered valid if the interrupt LCD_INT_DATA_VALID occurred, or if bit 'LCD_INTERFACE_BUSY' reads '0'

LCD_INST_WORD

Word write only. Writing to this register writes the 32-bit value into the FIFO, tagged as instruction. Burst writes are allowed. The LSB byte of the word will be transmitted out of the FIFO first.

LCD_DATA_WORD

Word write only. Writing to this register writes the 32-bit value into the FIFO, tagged as data. Burst writes are allowed. The LSB byte of the word will be transmitted out of the FIFO first.

Table 224LCD_STATUS register

BIT	LABEL	RESET	R/W
0	LCD_INT_FIFO_EMPTY	0	R
1	LCD_INT_FIFO_HALF_EMPTY	0	R
2	LCD_INT_FIFO_OVERRUN	0	R
3	LCD_INT_READ_VALID	0	R
4	LCD_INTERFACE_BUSY	1	R
5-9	LCD_COUNTER	00000	R

LCD_INT_FIFO_EMPTY

Is set when the FIFO is empty (LCD_counter = 0) and not masked in the LCD_INT_MASK register

LCD_INT_FIFO_HALF_EMPTY

Is set when the FIFO is less then half full (when LCD_counter < 8) and not masked in the LCD_INT_MASK register

LCD_INT_FIFO_OVERRUN

is set when value being written is larger that the FIFO can hold and not masked in the LCD_INT_MASK register

LCD_INT_READ_VALID

Is set when the value read from the LCD controller is valid and not masked in the LCD_INT_MASK register.

LCD_INTERFACE_BUSY

This bit indicates if the LCD interface is still reading the value from the controller.

LCD_COUNTER

The current value of the FIFO counter. 0x00 means empty.

Table 225LCD_CONTROL register

BIT	LABEL	RESET	R/W
0	reserved	0	R/W
1	PS (Parallel/Serial)	0	R/W
2	MI (Motorola 6800/Intel 8080)	0	R/W
3	IF (8bit/4bit)	0	R/W
4-5	SERIAL_CLK_SHIFT	11	R/W
6-7	SERIAL_READ_POS	11	R/W
8	BUSY_FLAG_CHECK	0	R/W
9	BUSY_VALUE	0	R/W
10-12	BUSY_BIT_NR	111	R/W
13	BUSY_RS_VALUE	0	R/W
14	INVERT_CS	0	R/W
15	INVERT_E_RD	0	R/W
16	MSB_FIRST	0	R/W
17	LOOPBACK	0	R/W

PS (Parallel/Serial)
‘0’ will put the LCD interface in parallel mode
‘1’ will put the LCD interface in serial mode

MI (Motorola 6800/Intel 8080)
‘0’ will put the LCD interface in 8080 mode.
‘1’ will put the LCD interface in 6800 mode.
Dont care if PS=’1’

IF (8bit/4bit)
‘0’ will put the LCD interface in 8 bit mode.
‘1’ will put the LCD interface in 4 bit mode.
Dont care if PS=’1’

SERIAL_CLK_SHIFT
In parallel mode (PS = 0):
Dont care

In Serial Mode (PS=1):
‘xx00’ -- clock mode 0
‘xx01’ -- clock mode 1
‘xx10’ -- clock mode 2
‘xx11’ -- clock mode 3

SERIAL_READ_POS
In parallel mode (PS = 0):
Dont care

In Serial Mode (PS=1):
‘00xx’ -- Sampling done at the beginning of the cycle
‘01xx’ -- Sampling done at 0.25 * cycle
‘10xx’ -- Sampling done at 0.5 * cycle
‘11xx’ -- Sampling done at 0.75 * cycle
See Fig. 43

BUSY_FLAG_CHECK
‘0’ will disable the busy-flag checking.
‘1’ will enable the busy-flag-checking.

BUSY_VALUE
‘0’ will mean that if the checked bit equals to ‘0’ that the LCD controller is not busy.
‘1’ will mean that if the checked bit equals to ‘1’ that the LCD controller is busy.
Dont care if BUSY_FLAG_CHECK = ‘0’

BUSY_BIT_NR
This 3 bit value stores the bit of the 6800/8080 bus which represents the busy flag.
Don’t care if BUSY_FLAG_CHECK = ‘0’

BUSY_RS_VALUE
‘0’ means that the busy check will happen on RS value ’0’.
‘1’ means that the busy check will happen on RS value ’1’.
Don’t care if BUSY_FLAG_CHECK = ‘0’

INVERT_CS
‘1’ means that the Chip_select output will be active low
‘0’ means that the Chip select output will be active high

INVERT_E_RD
‘0’ means that the E_RD output pin will be active low
‘1’ means that the E_RD output pin will be active high.

MSB_FIRST
8-bit mode:
Don’t care.

4-bit mode:
‘0’ means that bits 3-0 will be transmitted first.
‘1’ means that bits 7-4 will be transmitted first.

serial mode:
‘0’ means that bit 0 will be transmitted first.
‘1’ means that bit 7 will be transmitted first.

LOOPBACK:
‘0’ means normal operation
‘1’ mean that the data output and input are equal: loop back mode.

Table 226LCD_INT_RAW register

BIT	LABEL	RESET	R/W
0	LCD_INT_FIFO_EMPTY_RAW	0	R
1	LCD_INT_FIFO_HALF_EMPTY_RAW	0	R
2	- Reserved-	0	R
3	LCD_INT_READ_VALID_RAW	0	R

LCD_INT_FIFO_EMPTY_RAW

Is set when the FIFO is empty (LCD_counter = 0)

LCD_INT_FIFO_HALF_EMPTY_RAW

Is set when the FIFO is less then half full (when LCD_counter < 8)

- Reserved -

Ignore this bit in software.

LCD_INT_READ_VALID_RAW

Is set when the the value that has been read from the LCD controller is valid.

Table 227LCD_INT_CLEAR register

BIT	LABEL	RESET	R/W
0	LCD_INT_FIFO_EMPTY_CLR	-	W
1	LCD_INT_FIFO_HALF_EMPTY_CLR	-	W
2	LCD_FIFO_OVERRUN_CLR	-	W
3	LCD_INT_READ_VALID_CLR	-	W

Table 228LCD_INT_MASK register

BIT	LABEL	RESET	R/W
0	LCD_FIFO_EMPTY_MASK	1	R/W
1	LCD_FIFO_HALF_EMPTY_MASK	1	R/W
2	LCD_FIFO_OVERRUN_MASK	1	R/W
3	LCD_READ_VALID_MASK	1	R/W

Table 229LCD_READ_COMMAND register

BIT	LABEL	RESET	R/W
0	LCD_READ_COMMAND	-	W

Table 230LCD_INST_BYTE register

BIT	LABEL	RESET	R/W
0-7	INSTRUCTION_BYTE	0x00	R/W

Table 231LCD_DATA_BYTE register

BIT	LABEL	RESET	R/W
0-7	DATA_BYTE	0x00	R/W

Table 232LCD_INST_WORD register

BIT	LABEL	RESET	R/W
0-31	INSTRUCTION_WORD	-	W

Table 233LCD_DATA_WORD register

BIT	LABEL	RESET	R/W
0-31	DATA_WORD	-	W

29.4 Interrupt

An interrupt is generated on the following occasions:

- When the FIFO is empty (LCD_FIFO_EMPTY).
- When the FIFO is half empty. (LCD_FIFO_HALF_EMPTY)
- When the FIFO is overrun. (LCD_FIFO_OVERRUN)
- When the requested instruction/data register is valid. (LCD_READ_VALID)

Any of these interrupts can be masked individually to keep them from generating an interrupt to the CPU, by using the LCD_INT_MASK register. The interrupts after masking can be read in the LCD_STATUS register. Writing a ‘1’ in the mask register will mask the interrupt. The status of the interrupts without masking can be read in the ‘LCD_INT_RAW’ register

Clearing the interrupts:

An interrupt can be cleared by writing a ‘1’ to the respectable bit in the LCD_INT_CLR register. If the interrupt has not been solved, for instance the FIFO is still empty, this will re-set the interrupt, when not masked.

29.5 Using SDMA flow control

All data transers towards the LCD interface can be done using the SDMA and the corresponding flow control, reducing CPU interupting.
The SDMA has the ability to transfer blocks of data from memory and the LCD FIFO while checking the FIFOLEVEL flow control before sending data. this way data will only be send when the LCD interface has space left in its local FIFO.

This construction allows much larger blocks of data to be transported to the LCD interface then the maximal 16 bytes of the FIFO at a time..
If the external LCD controller supports it, a single enable command to the SDMA controller may refresh the complete LCD screen, making the LCD-contoller a very low CPU-intensive piece of hardware.

It requires a single SDMA channel to transfer data to the LCD interface. If a linked-list must be supported, then two sequential channels are required. See the SDMA chapter of this document for more info.

LCD-reading is not supported with SDMA flow control.

29.6 Clock relation HCLK and LCDCLK.

The LCDCLK and the HCLK can be totally independant from eachother. The HCLK logic samples the LCDLCK and uses this as a timing reference. Internally the LCDCLK is converted to pulses at each rising edge of the LCDCLK.

The HCLK should run at least 2 times as fast as the LCDCLK.
The LCDCLK should have the ‘clock_stretching’ option enabled when it is configured.

Every 5 LCDCLK cycles means one LCD bus cycle.

If Busy flag checking is enabled each write will take 10 cycles, since each write is accompanied by a read.

29.7 Changes in the LCD interface SAA775x -> PNX0101/Melody

- Added SDMA flow control, using the FIFOLEVEL method. The SDMA can take over LCD data traffic, to reduce CPU interruping.
- The LCDCLK can be set to any possible clock inside the CGU, creating a faster maximum external bus if required.
- The LCD interface is shared with the ISP1581 glue logic.

30 USB INTERFACE

30.1 Overview

The USB interface can be used for:

- Down loading of compressed music data (bulk) from PC to the removable flash card.
- Down loading of firmware data (bulk) from PC to the internal flash memory (field-upgradability).
- Up loading of speech data (bulk) from the removable flash card to the PC.

The USB interface for PNX0101 is a full speed USB interface (12Mbits/s) and is USB 1.1 compliant. It consist of an analog transceiver (ATX), and a Full Speed USB module (FS22).

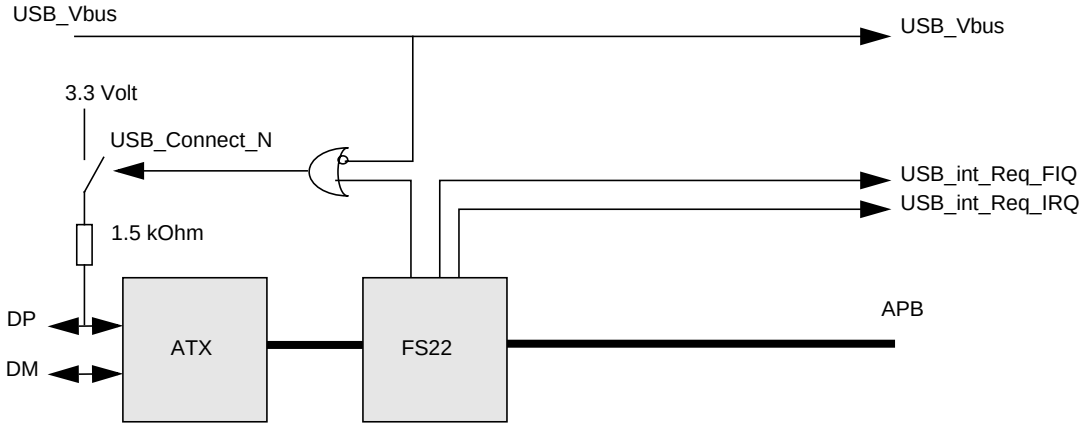


Fig. 44 USB interface

The USB_Connect_N line can be used in the situation where internal initialisation of the USB device is longer than the time needed according to USB specification:

“120ms after detection by the host of a USB device, the USB device should start responding to the transaction on the USB”.

A USB_Connect_N line can be used to switch the external pull-up resistor of 1.5kOhm under software control. This line is S/W controlled under the additional condition that USB_Vbus is high. This behaviour is modelled by the or gate in the schematic.

Important note: some external status and control bits of the USB are handled in the sys_creg register block. See the chapter on the SYSCREG for the description on the USBAPB_TOP_cfg and USBAPB_TOP_sts registers.

30.2 Functional description

30.2.1 ATX MODULE

For a complete functional description of the ATX, see “Analogue Transceiver for USB as CMOS18 bondpad”[12].

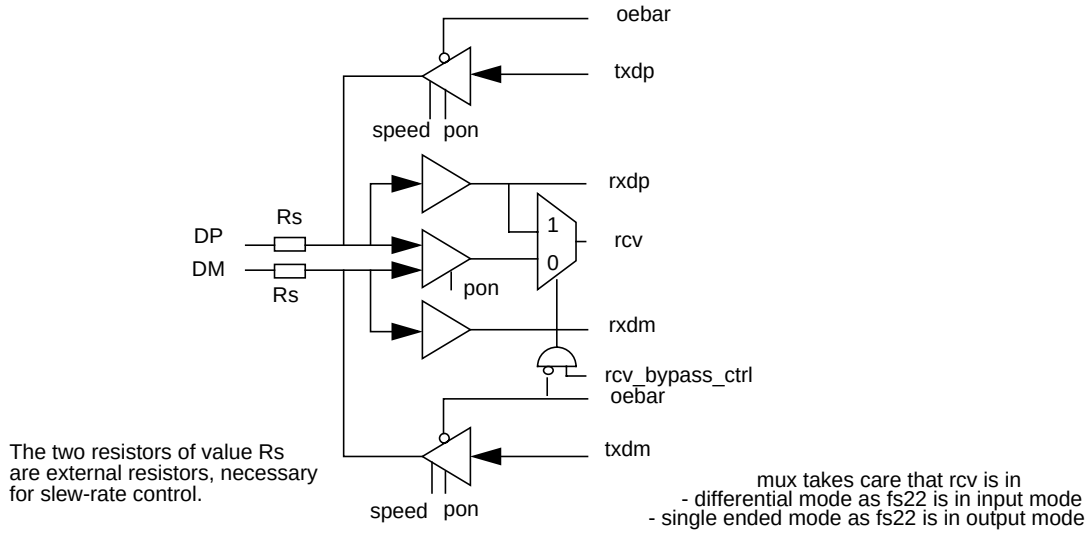


Fig. 45 Analogue Transeiver for USB (usb1.1)

The ATX can be put in a power-saving mode (suspend-state); only the single-ended receivers will be operational during this mode, in order to be able to detect activity on DP or DM.

Table 234 Status of the ATX

CONTROL SIGNALS			ATX MODES
OEBAR	PON	SPEED	
H	L	X	suspend
L	H	L	low-speed transmitting
H	H	L	low-speed receiving
L	H	H	full-speed transmitting
H	H	H	full-speed receiving

The function of the single-ended receiver is to compare the voltage on DP and DM with a certain threshold voltage. The comparison is done with some hysteresis, so it is a Schmitt-trigger buffer. The single-ended receiver is non-inverting.

The function of the differential receiver is to compare the voltages on both USB data lines DP and DM. For power saving, the differential receiver can be switched off.

Table 235 Functional description of the differential receiver

INPUTS		OUTPUTS
PON	**V(DP)-V(DM)**	**RCV**
L	X	undefined
H	> 200 mV	H
H	< -200 mV	L
H	-200 mV < V(DP)-V(DM) < 200 mV	undefined

Table 236 Functional description of the transmitters

INPUTS				OUTPUTS
OEBAR	**PON**	**TXDP (OR TXDM)**	**SPEED**	**DP (OR DM)**
H	H	X	X	high impedance
L	H	L	L	going to “gnd” with low speed fall time
L	H	L	H	going to “gnd” with full speed fall time and correct output impedance
L	H	H	L	going to “vdd” with low speed rise time
L	H	H	H	going to “vdd” with full speed rise time and correct output impedance
X	L	X	X	undefined

30.2.2 FS22 MODULE

For a complete functional description, see document “Technical Specification of USB FS22”[13].

For information about the update from version 1.3 to 1.5 see document “darwin_changes.pdf”[..].

The FS22 module requires a 48MHz input clock. This 48MHz clock is generated in the CGU by the LPPLL.

The FS22 module internally produces a 1kHz frame clock and a 12MHz recovered clock (bitclk).

Table 237 USB Endpoints

LOGIC ENDPOINT	PHYSICAL ENDPOINT	TYPE	DIRECTION	SIZE	DOUBLE BUFFERED
0	0	control	out	8	no
	1	control	in	8	no
1	2	interrupt	out	16	no
	3	interrupt	in	16	no
2	4	interrupt	out	16	no
	5	interrupt	in	16	no
3	6	bulk	out	64	yes
	7	bulk	in	64	yes
4	8	isochronous	out	294	yes
	9	isochronous	in	294	yes

Each endpoint, with the exception of the control endpoints can be enabled or disabled. The implementation of function makes use of two SRAMs of 788 bytes each(756 bytes + 32 overhead bytes). All endpoints are accessed via the APB interface.

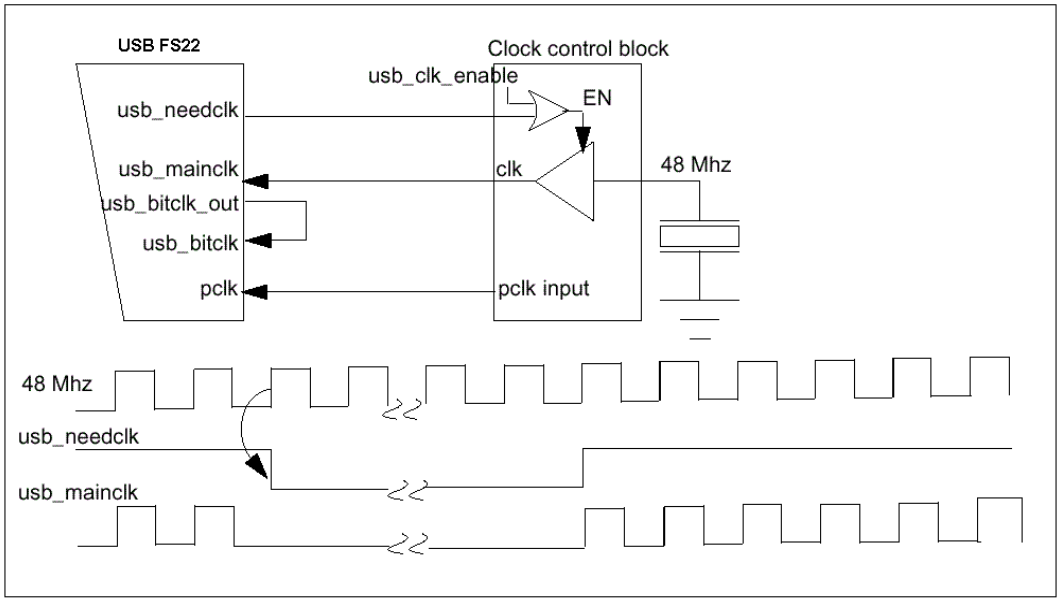
30.2.3 CLOCKS

IP_3501 has the following clock connections:

usb_mainclk: usb_mainclk is the 48 Mhz+/-500 ppm input clock. This clock need not be synchronized with the system clock (pclk). Gating of this clock is possible by external control block using usb_needclk signal. This clock will be used to recover the 12 Mhz clock from the USB bus.

usb_bitclk: usb_bitclk is the 12 Mhz recovered clock . This is internally generated in the IP and is available on the usb_bitclk_out pin. It is brought out to the top level to ease the clock tree generation and has to be fed back to usb_bitclk signal pin.

pclk: This is the VPB bus clock. Minimum clock frequency of this clock is 16 Mhz.


USB FS22
Clock control block
usb_clk_enable
EN
usb_needclk
usb_mainclk
clk
48 Mhz
usb_bitclk_out
usb_bitclk
pclk
pclk input
48 Mhz
usb_needclk
usb_mainclk

30.3 Registers

Table 238 FS22 registers

ADDRESS SPACE	OFFSET	REGISTER NAME	R/W ACCESS	RESET
0x8010_1000 - 0x8010_13FF	0x00	INTERRUPT STATUS	R	0x0
(base address may change due to CB flow)	0x04	INTERRUPT ENABLE	R/W	0x0
	0x08	INTERRUPT CLEAR	W	
	0x0C	INTERRUPT SET	W	
	0x10	COMMAND CODE	W	
	0x14	COMMAND DATA	R	0x0
	0x18	RECEIVE DATA	R	0x0
	0x1C	TRANSMIT DATA	W	
	0x20	RECEIVE PACKET LENGTH	R	0x0
	0x24	TRANSMIT PACKET LENGTH	W	
	0x28	USB CONTROL	R/W	0x0
	0x2C	FIQ SELECTION	W	

30.3.1 INTERRUPT STATUS REGISTER

The Interrupt Status register has got two parts:

1. Interrupt status part.
2. IN endpoint buffer status

Each interrupt has got one bit in the interrupt status register. This is set when the interrupt occurs. CPU upon receiving interrupt should read the interrupt status register to know which interrupt has occurred. There are seven bits for the interrupt status. The IN endpoint buffer status shows there are valid packets in the IN endpoint buffer. Any packets which is not yet transferred host is considered valid. This is a Read only register.

There are 4 additional bits in the interrupt status register to show the valid buffer status for the IN end points. There is one bit corresponding to each physical endpoint. Before attempting a new write to the buffer the software should ensure that there is no valid packet present in the buffer.

Table 239 Interrupt Status register

BITS	VARIABLE	DESCRIPTION	RESET	R/W
0		Frame interrupt. Occurs once in every millisecond	0	R
1		Interrupt for endpoint 0	0	R
2		Interrupt for endpoint 1	0	R
3		Interrupt for endpoint 2	0	R
4		Interrupt for endpoint 3	0	R
5		Interrupt for endpoint 4	0	R
6		Interrupt for endpoint 5	0	R
7		Interrupt for endpoint 6	0	R
8		Interrupt for endpoint 7	0	R
9		Device status	0	R
10		Command Code register is empty	0	R
11		Command Data register is full	0	R
12		EOP for OUT transfer is reached	0	R
13		EOP for IN transfer is reached	0	R
15-14		Reserved	-	R
16		Endpoint 1 buffer full	0	R
17		Endpoint 3 buffer full	0	R
18		Endpoint 5 buffer full	0	R
19		Endpoint 7 buffer full	0	R
31-20		Reserved	-	R

30.3.2 INTERRUPT ENABLE REGISTER

This register is for masking an interrupt. Setting 0 will mask the interrupt and setting to 1 will enable the interrupt. But the interrupt status register bit will be set irrespective of the value of the corresponding bit in the Interrupt enable register. Only the external interrupt corresponding to this bit will be affected. This is a Read- Write register.

Table 240 Interrupt Enable register

BITS	VARIABLE	DESCRIPTION	RESET	R/W
0		Enable Frame interrupt. Occurs once in every millisecond	0	R/W
1		Enable Interrupt for endpoint 0	0	R/W
2		Enable Interrupt for endpoint 1	0	R/W
3		Enable Interrupt for endpoint 2	0	R/W
4		Enable Interrupt for endpoint 3	0	R/W
5		Enable Interrupt for endpoint 4	0	R/W
6		Enable Interrupt for endpoint 5	0	R/W
7		Enable Interrupt for endpoint 6	0	R/W
8		Enable Interrupt for endpoint 7	0	R/W
9		Enable Device status	0	R/W
10		Enable Command Code register is empty	0	R/W
11		Enable Command Data register is full	0	R/W
12		Enable EOP for OUT transfer is reached	0	R/W
13		Enable EOP for IN transfer is reached	0	R/W
15-14		Reserved	-	R/W
16		Enable Endpoint 1 buffer full	0	R/W
17		Enable Endpoint 3 buffer full	0	R/W
18		Enable Endpoint 5 buffer full	0	R/W
19		Enable Endpoint 7 buffer full	0	R/W
31-20	-	Reserved	0	R/W

30.3.3 INTERRUPT CLEAR REGISTER

Setting a bit in this register will cause clearing of the corresponding interrupt. Setting 0 will not have any effect. This is a write only register.

Table 241 Interrupt Clear register

BITS	VARIABLE	DESCRIPTION	RESET	R/W
0		Clear Frame interrupt. Occurs once in every millisecond	0	W
1		Clear Interrupt for endpoint 0	0	W
2		Clear Interrupt for endpoint 1	0	W
3		Clear Interrupt for endpoint 2	0	W
4		Clear Interrupt for endpoint 3	0	W
5		Clear Interrupt for endpoint 4	0	W
6		Clear Interrupt for endpoint 5	0	W
7		Clear Interrupt for endpoint 6	0	W
8		Clear Interrupt for endpoint 7	0	W
9		Clear Device status	0	W
10		Clear Command Code register is empty	0	W
11		Clear Command Data register is full	0	W
12		Clear EOP for OUT transfer is reached	0	W
13		Clear EOP for IN transfer is reached	0	W
15-14		Reserved	-	W
16		Clear Endpoint 1 buffer full	0	W
17		Clear Endpoint 3 buffer full	0	W
18		Clear Endpoint 5 buffer full	0	W
19		Clear Endpoint 7 buffer full	0	W
31-20	-	Reserved	0	W

30.3.4 INTERRUPT SET REGISTER

Setting a bit in this register will cause corresponding interrupt to occur. This is mainly for debugging purpose. This is a write only register.

Table 242 Interrupt Set register

BITS	VARIABLE	DESCRIPTION	RESET	R/W
0		Set Frame interrupt. Occurs once in every millisecond	0	W
1		Set Interrupt for endpoint 0	0	W
2		Set Interrupt for endpoint 1	0	W
3		Set Interrupt for endpoint 2	0	W
4		Set Interrupt for endpoint 3	0	W
5		Set Interrupt for endpoint 4	0	W
6		Set Interrupt for endpoint 5	0	W
7		Set Interrupt for endpoint 6	0	W
8		Set Interrupt for endpoint 7	0	W
9		Set Device status	0	W
10		Set Command Code register is empty	0	W
11		Set Command Data register is full	0	W
12		Set EOP for OUT transfer is reached	0	W
13		Set EOP for IN transfer is reached	0	W
15-14		Reserved	-	W
16		Set Endpoint 1 buffer full	0	W
17		Set Endpoint 3 buffer full	0	W
18		Set Endpoint 5 buffer full	0	W
19		Set Endpoint 7 buffer full	0	W
31-20	-	Reserved	0	W

30.3.5 COMMAND CODE REGISTER

This register is used for writing the commands. The commands written here will get propagated to the FS22 and will be executed there. After executing the command the register will be empty and the CPU will get an interrupt “command_code_register_empty”. This is a 11 bit register with the following field definition.

Table 243 Command Code register

BITS	VARIABLE	DESCRIPTION	RESET	R/W
7-0	-	Reserved	00000000	-
15 - 8	Phase_identifier	Phase Identifier 0x01: Write data phase 0x02: Read data phase 0x05: Command phase	00000000	W
23 - 16	Command_code	USB Command Code	00000000	W
31 -24	-	Reserved	00000000	-

30.3.6 COMMAND DATA REGISTER

This is a byte wide read only register which will carry the data retrieved after executing a command. When this register is full, the CPU will get an interrupt “command_data_register_full” upon which it is supposed to read this register.

Table 244 Command Data register

BITS	VARIABLE	DESCRIPTION	RESET	R/W
7-0	D(7:0)	Command_Data	0	R
31-8	-	Reserved	0	-

30.3.7 RECEIVE DATA REGISTER

For an OUT transaction, the CPU reads the endpoint data from this register. Data from the RxRAM is fetched and filled in this register. This is a 4 byte register which is read at a peek rate of once in two clocks. There is no interrupt when the register is full.

Table 245 Receive Data register

BITS	VARIABLE	DESCRIPTION	RESET	R/W
31-0	D(31:0)	received data	0	R

30.3.8 TRANSMIT DATA REGISTER

For an IN transaction, the CPU writes the data into this 4 byte wide register. This data will be transferred into the TxRAM before the next writing occurs. There is no interrupt when the register is empty.

Table 246 Transmit Data register

BITS	VARIABLE	DESCRIPTION	RESET	R/W
31-0	D(31:0)	data to be transmitted	0	W

30.3.9 RECEIVE PACKET LENGTH REGISTER

This gives the number of bytes remaining in the RAM for the current packet being transferred and indicates whether the packet is valid or not. This register will be updated at every word that get transferred to the system. The CPU can use this register to get the number of bytes to be transferred. When the number of bytes reaches to zero it will give an End Of Packet interrupt. The first 10 bits carries the information on the number of bytes and the 11th bits indicates the data is valid or not. This is a 11 bits read only register.

Table 247 Receive packet length register

BITS	VARIABLE	DESCRIPTION	RESET	R/W
9-0	D(9:0)	Length of received Packet	0	R
10	D(10)	Data is valid (HIGH), Data is not valid (LOW)	0	R
31-11	-	Reserved	0	-

30.3.10 TRANSMIT PACKET LENGTH REGISTER

This counts the number of bytes transferred from the CPU to the RAM. The CPU has to program this register with the Max packet size value before writing into the RAM. CPU can read this register to determine the number of bytes it has transferred to the memory. This is a 10 bits read/write register.

Table 248 Transmit packet length register

BITS	VARIABLE	DESCRIPTION	RESET	R/W
9-0	D(9:0)	Length of Packet to be transmitted	0	R/W
31-10	-	Reserved	0	-

30.3.11 USB CONTROL REGISTER

This register controls the overall operation of the FS22.

When the CPU wants to read the data from an endpoint it should make the read enable bit high and should program the logical endpoint number. The control logic will start fetching data for this endpoint from the RAM and fill the receive data register. When the End Of Packet (EOP) is reached the read enable bit will be disabled by the control logic and the CPU will receive an interrupt. In between if the CPU makes the read enable bit low then also the reading will be terminated. In this case the data will remain in RAM. When the read enable signal is made high again for this endpoint, the data will be read from the beginning.

For writing into the RAM, the write enable signal should be made high and the CPU should write to the transmit data register. Prior to this it should write to the Tx length register the number of bytes it is going to send in the packet. When the control logic receives this many number of bytes, it will reset the write enable bit. If the CPU resets this bit in the midway, writing will start again from the beginning.

Both the read enable and write enable bit can be high at the same time. In that case read will take the priority. This feature gives the flexibility to stop a write transfer for some time and do a read transfer and again come back to the same write transfer. But it is not possible to stop a read transfer and go to a write transfer.

Table 249 USB Control register

BITS	VARIABLE	DESCRIPTION	RESET	R/W
0	-	Read enable 1 = enables the data reading for the endpoint 0 = disables the data reading for the endpoint	0	R/W
1	-	Write enable 1 = enables the data writing for the endpoint 0 = disables the data writing for the endpoint	0	R/W
6-2	-	Logical endpoint number	0	R/W
31-7	-	Reserve	0	-

30.3.12 FIQ SELECTION REGISTER

The FS22 has got 2 interrupt outputs namely USB_int_req_FIQ and USB_int_req_IRQ. The FIQ interrupt is from a single source and the IRQ interrupt is the combined effect of all other interrupt sources. Any of the following interrupt can be routed to FIQ:

- Frame interrupt (EP0)
- Bulk out interrupt (EP6)
- Bulk in interrupt (EP7)

The FIQ selection register acts as the selection register for these 3 interrupts. This is a 3-bit write only register. Only one bit can be high at a time. Setting all bits ‘0’ removes FIQ interrupt generation.

Table 250 USB FIQ selection register

BITS	VARIABLE	DESCRIPTION	RESET	R/W
0	-	Frame interrupt 1 = frame interrupt is selected to be the FIQ 0 = frame interrupt is not selected to be the FIQ	0	W
1	-	Bulk out interrupt 1 = Bulk out is selected to be the FIQ 0 = Bulk out is not selected to be the FIQ	0	W
2	-	Bulk in interrupt 1 = Bulk in interrupt is selected to be the FIQ 0 = Bulk in interrupt is not selected to be the FIQ	0	W
31-3	-	Reserve	0	-

30.4 Interrupts

There are three interrupts to the system:

- USB_int_req_FIQ
- USB_int_req_IRQ
- Frame_Toggle

30.4.1 USB_INT_REQ_FIQ

This is the high priority interrupt to the system. The frame interrupt, Bulk OUT interrupt or Bulk IN interrupt can be routed to generate the FIQ. It is a must that this interrupt should have only one source at a time.

30.4.2 USB_INT_REQ_IRQ

This is the low priority interrupt to the system. The data transfer for all endpoints other than the FIQ source is initiated through this interrupt. This interrupt has got multiple sources, sources different from the source that created the FIQ at that point in time.

30.4.3 FRAME_TOGGLE

The Frame_Toggle output of the USB is directly connected to the interrupt controller as a seperate interrupt line.

30.4.4 OTHER INTERRUPT BIT DEFINITIONS

30.4.4.1 Frame interrupt

This interrupt occurs once in every millisecond if it is enabled. It is used for ISO endpoints.

30.4.4.2 Endpoint interrupts

These are the interrupts from the endpoints. The OUT endpoint of the USB generates an interrupt when its buffer is full and hence means CPU has to transfer the data from its buffer. In the case of IN endpoints interrupts are generated according to the setting in the Mode register of the USB core (Set_mode command) for the value of InterruptOnNAK bit. If this bit is set to ‘0’ only successful in transaction generate the interrupt. If it is set to ‘1’ successful transaction as well as NAK can cause the interrupt. On this interrupt CPU has to fill the corresponding IN endpoint buffer with Data.

30.4.4.3 Cmd_code_empty_int

This interrupt occurs when the execution of a command written in the command code register is executed and a new command can be issued again if needed This interrupt has little practical significance if the PCLK is less than 50 Mhz and can be disabled in the normal working environments. A delay of 4 Bitclks is enough to ensure that command in the Command Code register is not overwritten.

30.4.4.4 Cmd_data_full_int

For a Read command like get_chip_id when the read data become available in the command data register this interrupt will be issued to the CPU. It takes 7 USB clock cycles to get the data in the Command data register after writing the command in the command code register.

30.4.4.5 End_packet_out_int

When a packet transfer to the CPU is fully over this interrupt is generated. The software can use this interrupt to stop the Read operation or it can track the transfer from the packet length information receiving from the Rx Packet Length register.

30.4.4.6 End_packet_in_int

When transfer reaches the IN transfer length programmed by the software this interrupt is generated showing that one packet of data has been transferred to RAM.

30.4.5 INTERRUPT HANDLING

When CPU gets FIQ interrupt it does not need to read the status register as there is only one source for it depending on the selection of the FIQ select register. In the service routine CPU has to clear the interrupt by writing ‘1’ into bit position ‘0’ of intr_clear_register but when it gets the IRQ interrupt it has to read the Interrupt status register and understand which interrupt bit is set. The service routine has to clear the corresponding interrupt. If it is an interrupt from USB core (Bit 1 to Bit 8 of Status register) the clear interrupt command to the USB core must also be executed.

30.4.6 ZERO OVERHEAD OPERATION

To read the data without software overhead the CPU has to rely on the end_of_packet interrupt. The CPU can go on reading the receive data register and the read operation is to be terminated when the end of packet interrupt occurs. Hence the software does not have to track how many bytes it transferred. Still the number of bytes information is needed to remove the garbage bytes read.

31 MULTIMEDIA CARD INTERFACE

31.1 Overview

The Multimedia Card Interface (MCI) is an advanced microcontroller bus architecture (AMBA) compliant peripheral.

The multimedia card system provides communications and data storage, and consists of:

- A multimedia card stack. This can consist of up to 30 cards on a single physical bus.
- A multimedia card controller: This is the multimedia card master, and provides an interface between the system bus and the multimedia card bus.

The multimedia cards are grouped into three types according to their function:

- Read Only Memory(ROM) cards, containing a preprogrammed data.
- Read/Write(R/W) cards, used for mass storage.
- Input/Output(I/O) cards, used for communication.

The multimedia card system transfers commands and data using three signal lines:

- CLK: One bit transferred on both command and data lines with each clock cycle. The clock frequency varies between 0 MHz and 20 MHz.
- CMD: Bidirectional command channel that initializes a card and transfers commands. CMD has two operational modes, first mode ‘open drain’ for initialization and second mode ‘push-pull’ for command transfer.
- DAT: Bidirectional data channel, operating in push-pull mode.

31.1.1 Choice of flash memory cards

There are two different types of FLASH memory: NAND FLASH and NOR FLASH. The two FLASH types lend themselves to different applications. Basically NOR FLASH is a replacement for EPROM. NAND FLASH is a magnetic media replacement, particularly suited to serial data.

Today’s FLASH cards are give in the table below.

Table 251 Available FLASH Cards.

FLASH CARD	VENDOR	VCC	CAPACITY	INTERFACE	SIZE
Solid State Floppy Disk Card (SSFDC) or SmartMedia Card	Toshiba & Samsung	2.7-3.6V and 5V	...32 MByte available 64 MByte Q2 1999	DOS file system ATA (22-pins)	37 x 45 x 0.76
Multi Media Card (MMC)	Hitachi Ltd. & Infineon Technologies (open standard)	2.7-3.6V	16 MByte available 32 MByte Q3 1999 64 MByte 2000 128 MByte 2001	SPI (7-pins)	32 x 24 x 1.40
Memory Stick (MS)	Sony (license needed)	2.7-3.6V	...8 MByte available 16 MByte Q2 1999	Serial (10-pins)	50 x 21.5 x 2.8
Compact Flash Card (CFC)	SanDisk Corp.	3.3V/5V tolerant	...64 MByte available	DOS file system ATA (50-pins)	42 x 36 x 3.3

The Pinning of the Multi Media Card is given in table 252

Table 252 Pinning of the Multi Media Card (7 pins)

PIN	SYMBOL	DESCRIPTION
1	NC	
2	CMD	Command/Response
3	VSS1	Supply voltage ground
4	VDD	Supply voltage
5	CLK	Clock
6	VSS2	Supply voltage ground
7	DAT	Data

31.2 Functional description

The MCI is an interface between the APB and multimedia cards. It consist of two parts:

- The MCI adapter block provides all functions specific to the multimedia card, such as the clock generation unit, power management control, command and data transfer.
- The APB interface accesses the MCI adapter registers, and generates interrupt and DMA request signals.
- Conformance to multimedia card specifications v2.11
- Use as a multimedia card bus, it can be connected to up to 30 cards.

See Fig.47 on page 372 for a simplified block diagram of the MCI.

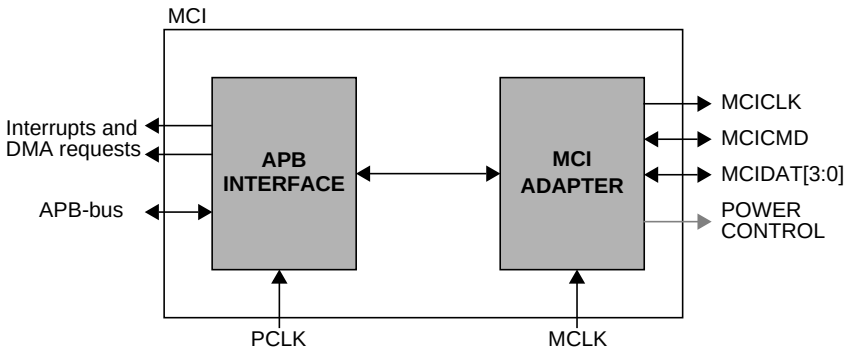


Fig. 47 MCI Block Diagram

The MCI uses two clock signals, each with a maximum frequency of 100 MHz. One or both clocks can be switched off for power saving:

- MCI adapter clock (MCLK).
- APB clock (PCLK).

The relation between MCLK and PCLK is defined below:

$$Fp \geq \frac{3}{8} Fm$$

where p = PCLK and m = MCLK

31.2.1 MCI ADAPTER

Fig.48 on page 373 shows a simplified block diagram of the MCI adapter.

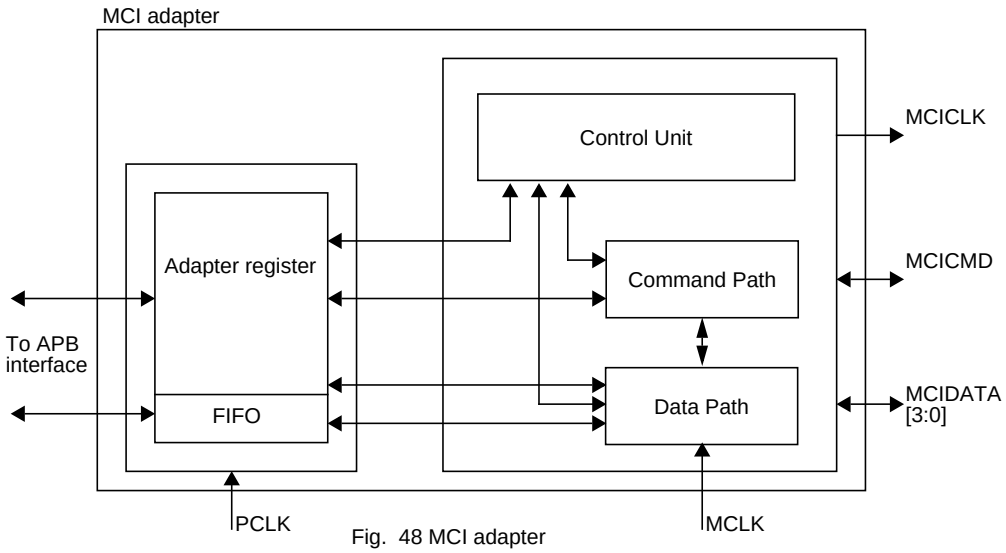


Fig. 48 MCI adapter

The MCI adapter is a multimedia memory card bus master that provides an interface to the multimedia card stack. It consist of five sub units:

- Adapter register block.
- Control unit.
- Command path.
- Data path.
- Data FIFO.

The adapter registers and FIFO use the APB clock domain. The control unit, command path and data path use the MCI adapter clock domain.

31.2.1.1 Adapter register block

The adapter register block contains all system registers. This block also generates the signals that clear the static flags in the multimedia card. The clear signals are generated when 1 is written into the corresponding bit location of the MCIClear register. The clear signal for flags generated in the MCLK domain is synchronized to that domain.

31.2.1.2 Control Unit

The control unit contains the power management functions and the card bus clock divider, Fig.49 on page 374 shows a block diagram of the control unit.

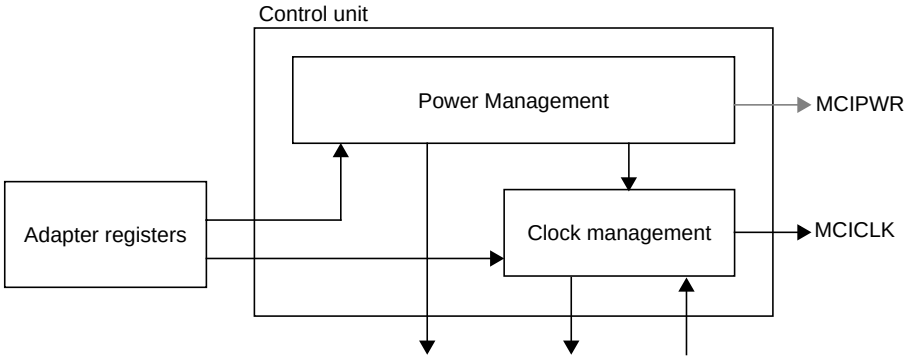


Fig. 49 Control unit

There are three power phases:

- power-off.
- power-up.
- power-on.

The power management logic controls an external power supply unit and disables the card bus output signals during the power-off or power-up phases.

The power-up phase is a transition phase between the power-off and power-on phases and allows an external power supply to reach card bus operating voltage. A device driver is used to ensure that the MCI remains in the power-up phase until the external power supply reaches the operating voltage.

The clock management logic generates and controls the MCICLK signal. The clock output can use either a clock divide or clock bypass. The clock output is inactive:

- After the MCI is reset.
- During the power-off or power-up phases.
- If the power saving mode is enabled and the card bus is in the IDLE state (eight clock periods after both the command and data path subunits enter the IDLE phase)

31.2.1.3 Command Path

The command path subunit sends commands to and receives responses from the cards, figure 50 shows a block diagram of the command path.

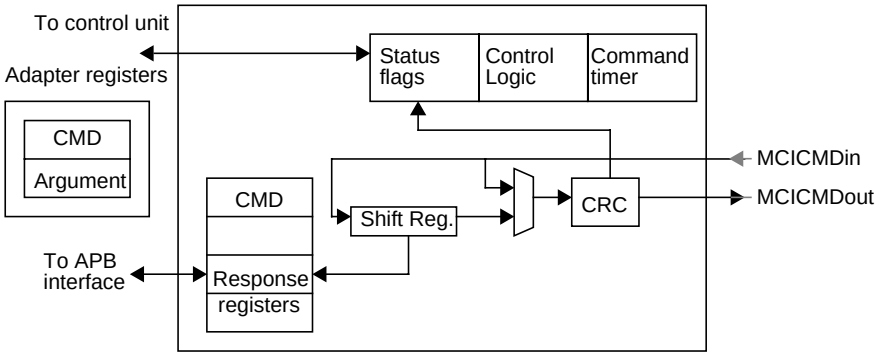


Fig. 50 Command path

When the command register is written to and the enable bit is set, command transfer starts. When the command has been sent, the Command Path State Machine sets the status flags and enters the IDLE state if response is not required. If a response is required, it waits for the response. When the response is received, the received CRC code and the internally generated code are compared and the appropriate status flags are set.

When the WAIT state is entered, the command timer starts running. If time-out is reached before the CPSM moves to the RECEIVE state, the time-out flag is set and the IDLE state is entered. The time-out period has a fixed value of 64 MCICLK clock periods. If the interrupt bit is set in the command register, the timer is disabled and the CPSM waits for an interrupt request from one of the cards.

If the pending bit is set in the command register, the CPSM enters the PEND state and waits for CmdPend signal from the data path subunit. When CmdPend is detected, the CPSM moves to the SEND state. This enables the data counter to trigger the stop command transmission.

The CPSM remains in the IDLE state for at least eight MCICLK clock periods to meet NCC and NRC timing constraints.

The command path operates in a half-duplex mode, so that commands and responses can be either sent or received. If the CPSM is not in the SEND state, the MCICMD output is in hi-z state. Data on the MCICMD signal is synchronous to the rising MCICLK edge.

All commands have a fixed length of 48 bits, the command format is shown in Table 5-1.

Table 253 Command Format

BIT POSITION	47	46	[45:40]	[39:8]	[7:1]	0
Width	1	1	6	32	7	1
Value	0	1	-	-	-	1
Description	start bit	transmission bit	command index	argument	CRC7	end bit

The MCI Adapter supports two response types, 48 and 136 bit long response both with CRC error checking. If the response does not contain CRC(CMD1 response), the device driver must ignore the CRC failed status.

Table 254 Short Response Format

BIT POSITION	47	46	[45:40]	[39:8]	[7:1]	0
Width	1	1	6	32	7	1
Value	0	0	-	-	-	1
Description	start bit	transmission bit	command index	card status	CRC7 or '1111111'	end bit

Table 255 Long Response Format

BIT POSITION	135	134	[133:128]	[127:1]	0
Width	1	1	6	127	1
Value	0	1	'111111'	-	1
Description	start bit	transmission bit	reserved	CID or CSD (including internal CRC7)	end bit

The command register contains the command index (six bits sent to a card) and the command type. These determine whether the command requires a response and whether the response is 48 or 136 bit long. The command path implements the status flags shown in table 256 on page 376.

Table 256 Command path status flags

FLAG	DESCRIPTION
CmdRespEnd	Set if response CRC is OK
CmdCrcFail	Set if response CRC fails
CmdSent	Set when command is sent
CmdTimeOut	Response timeout
CmdActive	Command transfer in progress

The CRC generator calculates the CRC checksum for all bits before the CRC code. This includes the start bit, transmitter bit, command index and command argument (or card status).The CRC checksum is calculated for the first 120 bits of CID or CSD for long response format. Note that the start bit, transmitter bit and the six reserved bits are not used in the CRC calculation.

The CRC checksum is a 7 bit value:

$$CRC[6:0] = Remainder[\ (M(x) * x^{7}) / G(x)\ ]$$
$$G(x) = x^{7} + x^{3} + 1$$
$$M(x) = (start\ bit) * x^{39} + \ + (last\ bit\ before\ CRC) * x^{0,}\ or$$
$$M(x) = (CID\ or\ CSD\ bit) * x^{119} + \ + (last\ bit\ before\ CRC) * x^{0}$$

31.2.1.4 Data Path

The data path subunit transfers data to and from cards.

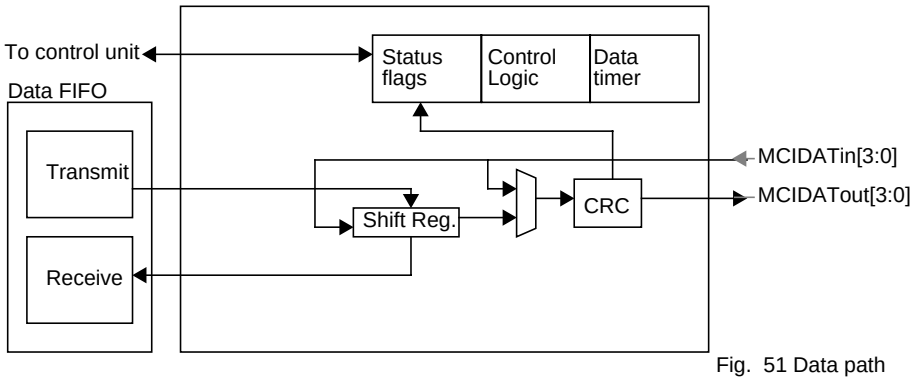


Fig. 51 Data path

The card data bus width can be programmed using the clock control register. If the wide bus mode is enabled, data is transferred at four bits per clock cycle over all four data signals(MCIDAT[3:0]). if the wide bus mode is not enabled, only one bit per clock cycle is transferred over MCIDAT0.

Depending on the transfer direction (send or receive), the Data Path State Machine moves to WAIT_S or WAIT_R state when it is enabled.

The DPSM moves to WAIT_S state. If there is data in the send FIFO, DPSM moves to SEND state and the data path subunit starts sending data to a card.

The DPSM moves to WAIT_R state and wait for a start bit. When it receives a start bit, the DPSM moves to RECEIVE state and the data path subunit starts receiving data from a card.

The DPSM operates at MCICLK frequency. Data on the card bus signals is synchronous to the rising edge of the MCICLK.

The DPSM has 6 states:

- IDLE: - The data path is inactive and the MCIDAT[3:0] outputs are in hi-z. When the data control register is written and enable bit is set, the DPSM loads the data counter with a new value and, depending on the data direction bit, moves to either the WAIT_S or WAIT_R state.
- WAIT_R: - If the data counter equals zero the DPSM moves to the IDLE state when the receive FIFO is empty. If the data counter is not zero, the DPSM waits for a start bit on MCIDAT. The DPSM moves to the RECEIVE state If it receives a start bit before a time-out and loads the data block counter. If it reaches a timeout before it detects a start bit or start bit error occurs, it moves to IDLE state and sets the timeout status flag.
- RECEIVE: - Serial data received from a card is packed in bytes and written to the data FIFO. Depending on the transfer mode bit in the data control register, the data transfer mode can be either block or stream: In the block mode, when the data block counter reaches zero, the DPSM waits until it receives the CRC code. If the received code matches the internally generated CRC code, the DPSM moves to WAIT_S state. if not, the CRC fail status flag is set and DPSM moves to the IDLE state. In the stream mode, the DPSM receives data while the data counter is not zero. When the counter is zero, the remaining data in the shift register is written to the data FIFO, and the DSPM moves to WAIT_R state.

  If a FIFO overrun error occurs, the DPSM sets the FIFO error flag and moves to WAIT_R state.
- WAIT_S: - The DPSM moves to the IDLE state if the data counter is zero, if not it waits until the data FIFO empty flag is de-asserted and moves to the SEND state.

  The DPSM remains in the WAIT_S state for at least two clock periods to meet Nwr timing constraints.

- SEND: - The DPSM starts sending data to a card. Depending on the transfer mode bit in the data control register, the data can be either block or stream: In block mode, when the data block counter reaches zero, the DPSM sends an internally generated CRC code and end bit and moves to the BUSY state. In stream mode, the DPSM sends data to a card while the enable bit is HIGH and the data counter is not zero. It then moves to the IDLE state.

  If the FIFO underrun error occurs, the DPSM sets the FIFO error flag and moves to the IDLE state.
- BUSY: - The DPSM waits for the CRC status flag, if it does not receive a positive CRC status, it moves to the IDLE state and sets the CRC fail status flag. If it receives a positive CRC status, it moves to WAIT_S state if MCIDAT0 is not low.

  If a timeout occurs while the DPSM is in the busy state, it sets the data timeout flag and moves to the IDLE state.

The data timer is enabled when the DPSM is in the WAIT_R or BUSY state and generates the data time-out error:

- When transmitting data, the timeout occurs if the DPSM stays in the BUSY state longer then the programmed timeout period.
- When receiving data, the timeout occurs if the end of the data is not true and if the DPSM stays in the WAIT_R state longer then the programmed timeout period

31.2.1.5 Data Counter

The data counter has two functions:

- To stop a data transfer when it reaches zero. This is the end of the data condition.
- To start transferring a pending command. This is used to send the stop command for a stream data transfer.

The data block counter determines the end of a data block. If the counter is zero, the end-of-data condition is true.

31.2.1.6 Bus Mode

In wide bus mode, all four data signals(MCIDAT[3:0]) are used to transfer data and the CRC code is calculated separately for each data signal. While transmitting data blocks to a card, only MCIDAT0 is used for the CRC token and busy signalling. The start bit must be transmitted on all four data signals at the same time (during the same clock period). If the start bit is not detected on all data signals on the same clock edge while receiving data, the DPSM sets the start bit error flag and moves to the IDLE state.

The data path also operates in half-duplex mode, where data is either sent to a card or received from a card. While not being transferred, MCIDAT[3:0] are in hi-z state. Data on these signals is synchronous to the rising edge of the clock period.

When selecting wide mode, both nMCIDAT0EN and nMCIDATEN outputs are driven low at the same time. If not, the MCIDAT[3:1] outputs are always in hi-z state and only the MCIDAT0 output is driven low when data is transmitted.

31.2.1.7 CRC Token Status

The CRC token status follows each write data block and determines whether a card has received the data block correctly. When the token has been received, the card asserts a busy signal by driving MCIDAT0 low. Table 257 shows the CRC token status values.

Table 257 CRC token status

TOKEN	DESCRIPTION
010	Card has received error-free data block
101	Card has detected a CRC error

31.2.1.8 Status Flags

Table 258 lists the data path status flags.

Table 258 Data path status flags

FLAG	DESCRIPTION
TxFifoFull	Transmit FIFO is full
TxFifoEmpty	Transmit FIFO is empty
TxFifoHalfEmpty	Transmit FIFO is half full
TxDataAvlbl	Transmit FIFO data available
TxUnderrun	Transmit FIFO underrun error
RxFifiFull	Receive FIFO is full
RxFifoEmpty	Receive FIFO is empty
RxFifoHalfEmpty	Receive FIFO is half full
RxDataAvlbl	Receive FIFO data available
RxOverrun	Receive FIFO underrun error
DataBlockEnd	Data block sent/received
StartBitErr	Start bit not detected on all data signals in wide bus mode
DataCrcFail	Data packet CRC failed
DataEnd	Data end
DataTimeOut	Data timeout
TxActive	Data transmission in progress
RxActive	Data reception in progress

31.2.1.9 CRC Generator

The CRC generator calculates the CRC checksum only for the data bits in a single block and is bypassed In data stream mode. The checksum is a 16-bit value:

$$CRC[15:0] = Remainder[\ (M(x) * x^{15}\ ) / G(x)\ ]$$
$$G(x) = x^{16} + x^{12} + x^{5} + 1$$
$$M(x) = (first\ data\ bit) * x^{n} + \ \ + (last\ data\ bit) * x^{0}$$

31.2.1.10 Data FIFO

The data FIFO subunit is a data buffer with transmit and receive logic. Fig. shows a block diagram of the FIFO.

The FIFO contains a 32-bit wide, 16 word deep data buffer and transmit and receive logic. Because the data FIFO operates in the APB clock domain (PCLK), all signals from the subunits in the MCI clock domain (MCLK) are re synchronised.

Depending on TxActive and RxActive, the FIFO can be disabled, transmit enabled or receive enabled. TxActive and RxActive are driven by the data path subunit and are mutually exclusive:

- The transmit FIFO refers to the transmit logic and data buffer when TxActive is asserted.
- The receive FIFO refers to the receive logic and data buffer when RxActive is asserted.

31.2.1.11 Transmit FIFO

Data is written to the transmit FIFO through the APB interface once the MCI is enabled for transmission. When the write signal is asserted, data can be written into the FIFO location specified by the current value of the data pointer. The pointer is incremented after every FIFO write.

The transmit FIFO contains a data output register. This holds the data word pointed to by the read pointer. When the data path subunit has loaded its shift register, the data path logic asserts TxRdPtrInc. This signal is synchronised with PCLK and increments the read pointer and drives new data on the TxRdData output.

If the transmit FIFO is disabled, all status flags are de-asserted and the read and write pointers are reset. The data path subunit asserts the TxActive signal when it is transmits data. Table lists the transmit FIFO status flags

The transmit FIFO generates 4 status flags:

- TxFifoFull: Set to high when all 16 transmit FIFO words contain valid data.
- TxFifoEmpty: Set to high when the transmit FIFO does not contain valid data.
- TxHalfEmpty: Set to high when 8 or more transmit FIFO words are empty. This flag can be used as a DMA request.
- TxDataAvlbl: Set to high when the transmit FIFO contains valid data. This flag is inverse of the TxFifoEmpty flag.
- TxUnderrun: Set to high when an underrun error occurs. This flag is cleared by writing to the MCIClear register.

31.2.1.12 Receive FIFO

When the data path subunit receives a word of data, it drives data on the write data bus and asserts the write enable signal. This signal is synchronised to the PCLK domain. The write pointer is incremented after the write is completed and the receive FIFO control logic asserts RxWrDone, that then de-asserts the write enable signal.

On the read side, the content of the FIFO word pointed to by the current value of the read pointer is driven on the read data bus. The read pointer is incremented when the APB interface asserts RxRdPrtInc.

If the receive FIFO is disabled, all status flags are de-asserted and the read and write pointers are reset. The data path subunit assert RxActive when it is receiving data.

The receive FIFO generates 4 status flags:

- RxFifoFull: Set to high when all 16 receive FIFO words contain valid data.
- RxFifoEmpty: Set to high when the receive FIFO does not contain valid data.
- RxHalfFull: Set to high when 8 or more receive FIFO words contain valid data. This flag can be used as a DMA request
- RxDataAvlbl: Set to high when the receive FIFO is not empty. This flag is the inverse of the RxFifoEmpty flag.
- RxOverrun: Set to high when an overrun error occurs. This flag is cleared by writing to the MCIClear register.

31.2.1.13 APB Interface

Figure shows a block diagram of the APB interface.

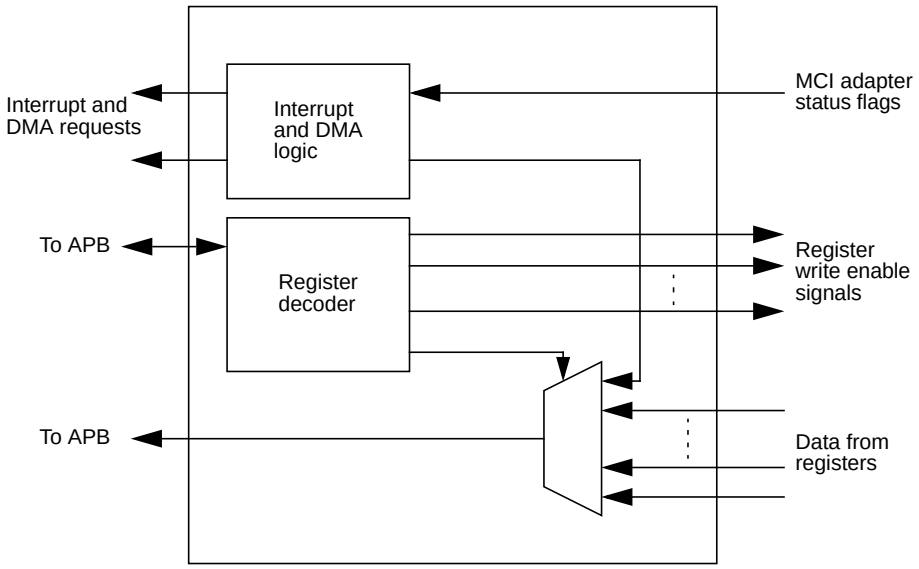


Fig. 52 APB Interface

The APB interface generates the interrupt and DMA requests and accesses the MCI adapter registers and the data FIFO. It consists of a data path, register decoder and interrupt/DMA logic.

31.2.1.14 Interrupt Logic

The interrupt logic generates two interrupt request signals, that are asserted when at least one of the selected status flags is high. A status flag generates the interrupt request if a corresponding mask flag is set. The interrupt request can be asserted even if PCLK is disabled.

A separate mask register is provided for each interrupt request signal.

31.2.1.15 DMA

The interface to the DMA controller includes the signals described in Table 259

Table 259 DMA controller interface signals

SIGNAL	TYPE	DESCRIPTION
DMASREQ	Single word DMA transfer request, asserted by MCI	For receive: Asserted if data counter is zero and receive FIFO contains more than one and fewer than eight words. For transmit: Asserted if fewer than eight and more than one word remain for transfer to FIFO
DMABREQ	Burst DMAtransfer request, asserted by MCI	For receive: Asserted if FIFO contains eight words and data counter is not zero, or if FIFO contains more than eight words. For transmit: Asserted if only eight words remain for transfer to FIFO
DMALSREQ	Last Single word DMA transfer request, asserted by MCI	For receive: Asserted if data counter is zero and FIFO contains only one word. For transmit: Asserted if only one word remains for transfer to FIFO
DMALBREQ	Last Burst DMA transfer request, asserted by MCI	For receive: Asserted if data counter is zero and FIFO contains eight words. For transmit: Asserted if only eight words remains for transfer to FIFO
DMACLR	DMA request clear, asserted by DMA controller to clear request signals	Asserted during transfer of last data in burst if DMA burst transfer is requested.

Because the four request signals are mutually exclusive, only one signal is asserted at a time. The signal remains asserted until DMACLR is asserted. After this, a request signal can be active again, depending on the conditions described in table 259 on page 382

31.3 Registers

The register offsets for the MCI Controller are defined in table 260 on page 383.

Table 260 MCI register map

ADDRESS SPACE	OFFSET	REGISTER NAME	R/W ACCESS	RESET
0x8980_0000 - 0x8980_0FFF	0x000	MCIPOWER	R/W	0x0
	0x004	MCICLOCK	R/W	0x0
	0x008	MCIARGUMENT	R/W	0x0
	0x00C	MCICOMMAND	R/W	0x0
	0x010	MCIRESPCMD	R	0x0
	0x014	MCIRESPONSE0	R	0x0
	0x018	MCIRESPONSE1	R	0x0
	0x01C	MCIRESPONSE2	R	
	0x020	MCIRESPONSE3	R	
	0x024	MCIDATATIMER	R/W	0x0
	0x028	MCIDATALENGTH	R/W	0x0
	0x02C	MCIDATACTRL	R/W	0x0
	0x030	MCIDATACNT	R	0x0
	0x034	MCISTATUS	R	0x0
	0x038	MCICLEAR	W	
	0x03C	MCIMASK0	R/W	0x0
	0x040	MCIMASK1	R/W	0x0
	0x044	MCISELECT	R/W	
	0x048	MCIFIFOCNT	R	0x0
	0x04C			
	|	Reserved		
	0x07C			
	0x080			
	|	MCIFIFO	R/W	0x0
	0x0BC			
	0xFE0	MCIPERIPHID0	R	0x00 00 00 80
	0xFE4	MCIPERIPHID1	R	0x00 00 00 11
	0xFE8	MCIPERIPHID2	R	0x00 00 00 04
	0xFEC	MCIPERIPHID3	R	0x0
	0xFF0	MCIPCELLID0	R	0x00 00 00 0D
	0xFF4	MCIPCELLID1	R	0x00 00 00 F0
	0xFF8	MCIPCELLID2	R	0x00 00 00 05
	0xFFC	MCIPCELLID3	R	0x00 00 00 B1

31.3.1 Power Control Register

The power control register controls an external power supply. Switching the power on and off adjusts the output voltage. Table 261 shows the bit assignment of power control register.

Table 261 MCI Power register

BIT	VARIABLE	DESCRIPTION	RESET	R/W
1 - 0	Ctrl	00 power-off 01 reserved 10 power-up 11 power-on	0	R/W
5 - 2	Voltage	Output voltage	0	R/W
6	OpenDrain	MCI CMD output control	0	R/W
7	Rod	Rod control	0	R/W
31 - 8	-	Reserved	-	-

When switching the external power supply on, the software first enters the power-up phase and waits until the supply output is stable before moving to the power-on phase. During the power-up phase, the MCIPWR signal is set HIGH. The card bus outlets are disabled during both phases.

The supply output voltage can be set using the voltage value on the MCIVDD outputs. Because the operating voltage range can be any value between 2.0 and 3.6 volts, the encoding of the voltage bits in the power control register application-specific.

31.3.2 Clock control register

The clock register is used to control the MCICLK output.

Table 262 MCIClock register

BIT	NAME	DESCRIPTION	RESET	R/W
7 - 0	ClkDiv	MCI bus clock period: MCLCLK freq = MCLK / [2x(ClkDiv+1)]	0	R/W
8	Enable	Enable MCI bus clock 0 clock disabled 1 clock enabled	0	R/W
9	PwrSave	Disable MCI clock output when bus is idle 0 always enabled 1 clock enabled when bus is active	0	R/W
10	Bypass	Enable bypass of the clock divide logic 0 disable bypass 1 enable bypass	0	R/W
11	WideBus	Enable wide bus mode 0 standard bus mode (only MCIDAT0 used) 1 Wide bus mode(MCIDAT3:0 used)	0	R/W
31 - 12	-	Reserved	-	-

While the MCI is in the Identification Mode, the MCICLK frequency must be less then 400 kHz. The clock frequency can be changed to maximum card bus frequency when relative card addresses are assigned to all cards.

After a data write, data cannot be written to this register for three MCLK clock periods plus two PCLK clock periods.

31.3.3 Argument Register

The argument register contains a 32 bit command argument, which is sent to a card as a part of a command message.

Table 263 MCI Argument register

BIT	NAME	DESCRIPTION	RESET	R/W
31 - 0	CmdArg	Command argument	0	R/W

If a command contains an argument, it must be loaded into the argument register before writing a command to the command register.

31.3.4 Command Register

The command register contains the command index and the command type bits:

- The command index is sent to a card as a part of a command massage.
- The command type bits controls the command path state machine (CPSM).

  Writing'1' to the enable bit starts the command send operation, while clearing the bit disables the CPSM.

Table 264 Command register

BIT	NAME	DESCRIPTION	RESET	R/W
5 - 0	CmdIndex	Command index	0	R/W
6	Response	If set CPSM waits for a response	0	R/W
7	LongRsp	If set CPSM receives a 136 bit long response	0	R/W
8	Interrupt	If set CPSM disables the command timer and waits for an interrupt request	0	R/W
9	Pending	If set CPSM waits for CmdPend signal before it starts sending a command	0	R/W
10	Enable	If set CPSM is enabled	0	R/W
31 - 11	-	Reserved	-	-

After a data write, data cannot be written to this register for three MCLK clock periods plus two PCLK clock periods.

Table 265 Command response type

RESPONSE	LONGRSP	DESCRIPTION
0	0	No response, expect CmdSent flag
0	1	No response, expect CmdSent flag
1	0	Short response, expect CmdRespEnd or CmdCrcFail flag
1	1	Long response, expect CmdRespEnd or CmdCrcFail flag

31.3.5 Response command register

This register contains the command index field of the last received command response.

Table 266 MCI RespCommand Register

BIT	NAME	DESCRIPTION	RESET	R/W
5 - 0	RespCmd	Response command index	0	R
31 - 6	-	Reserved	-	-

If the command response transmission does not contain the command index field (long response), the RespCmd field is unknown, although it must contain ’111111’ (the value of the reserved field from the response).

31.3.6 Response Registers

The response0-3 registers contain the status of a card, which is part of the received response.

Table 267 MCIResponse0 Register

BIT	NAME	DESCRIPTION	RESET	R/W
31 - 0	Status	Card status	0	R

The card status size can be 32 or 127 bits, depending on the response type (short/long).

Table 268 Response Registers

	SHORT RESPONSE	LONG RESPONSE
MCIResponse0	card status [31:0]	card status [127:96]
MCIResponse1	unused	card status [95: 64]
MCIResponse2	unused	card status [63:32]
MCIResponse3	unused	card status [31:1]

The most significant bit of the card status is received first. The MCIResponse3 register LSBit is always ‘0’.

31.3.7 Data Timer Register

The data timer register contains the data timeout period, in card bus clock periods.

Table 269 MCI DataTimer Register

BIT	NAME	DESCRIPTION	RESET	R/W
31 - 0	DataTime	Data timeout period	0	R/W

A counter loads the value from the data timer register and starts decrementing when the DPSM enters the WAIT_R or BUSY state. If the timer reaches 0 while the DPSM is in either of these states, timeout status flag is set.

A data transfer must be written to the data time register and the data length register before being written to the data control register.

31.3.8 Data Length Register

The data length register contains number of data bytes to be transferred. The value is loaded into the data counter when data transfer starts.

Table 270 MCI DataLength Register

BIT	NAME	DESCRIPTION	RESET	R/W
15 - 0	DataLength	Data length value.	0	R/W
31 - 16	-	reserved	-	-

For a block data transfer, the value in the data length register must be multiple of block size.

31.3.9 Data Control Register

The data control register controls the data path state machine (DPSM).

Table 271 Data Control Register

BIT	NAME	DESCRIPTION	RESET	R/W
0	Enable	Data transfer enabled	0	R/W
1	Direction	Data transfer direction 0 From the controller to a card 1 From a card to the controller	0	R/W
2	Mode	Data transfer mode 0 Block data transfer 1 Stream data transfer 6	0	R/W
3	-	Reserved	-	-
7 - 4	BlockSize	Data block length	0	R/W
31 - 8	-	Reserved	-	-

After a data write, data cannot be written to this register for three MCLK clock periods plus two PCLK clock periods.

Data transfer starts if ‘1’ is written to the enable bit. Depending on the direction bit, the DPSM moves to WAIT_S or WAIT_R state. It is not necessary to clear the enable bit after transfer.

Table 272 Data Block Length

BLOCKSIZE	BLOCK LENGTH
0	2^0 = 1 byte
1	2^1 = 2 bytes
...	-
11	2^{11} = 2048 bytes
12 - 15	Reserved

31.3.10 Data Counter

The data counter register loads the value from the data length register when the data path state machine (DPSM) moves from the IDLE state to the WAIT_R or WAIT_S state. As data is transferred the counter decrements the value until it reaches 0. The DPSM then moves to the IDLE state and the data status end flag is set.

Table 273 MCI DataCounter Register

BIT	NAME	DESCRIPTION	RESET	R/W
15 - 0	DataCount	Remaining data	0	R
31 - 16	-	reserved	-	-

31.3.11 Status Register

The status register is a read-only register. It contains two types of flags:

, static and dynamic. If static flags are asserted, they will remain asserted until software clears them by writing to the Clear register. Dynamic flags change their state according to the state of the underlying logic, i.e. FIFO full and empty flags will be asserted and de-asserted as FIFO is written to.

- Static[10:0] These remain asserted until they are cleared by writing to the Clear register.
- Dynamic[21:11] These change state depending on the state of the underlying logic (for example, FIFO full and empty flags are asserted and de-asserted as data while written to the FIFO).

Table 274 MCI Status Register

BIT	NAME	DESCRIPTION	RESET	R/W
0	CmdCrcFail	Command response received, CRC check failed	0	R
1	DataCrcFail	Data block is sent/received, CRC check failed	0	R
2	CmdTimeOut	Command response timeout	0	R
3	DataTimeOut	Data timeout	0	R
4	TxUnderrun	Transmit FIFO underrun error	0	R
5	RxOverrun	Receive FIFO overrun error	0	R
6	CmdRespEnd	Command response received, CRC check passed	0	R
7	CmdSent	Command sent (cmd does not require a response)	0	R
8	DataEnd	Data end, data counter is zero	0	R
9	StartBitErr	Start bit not detected on all data signals in wide bus mode	0	R
10	DataBlockEnd	Data block is sent/received, CRC check passed	0	R
11	CmdActive	Command transfer in progress Dynamic	0	R
12	TxActive	Data transmit in progress	0	R
13	RxActive	Data receive in progress	0	R
14	TxFifoHalfEmpty	Transmit FIFO is half empty	0	R
15	RxFifoHalfFull	Receive FIFO is half full	0	R
16	TxFifoFull	Transmit FIFO is full	0	R
17	RxFifoFull	Receive FIFO is full	0	R
18	TxFifoEmpty	Transmit FIFO is empty	0	R
19	RxFifoEmpty	Receive FIFO is empty	0	R
20	TxDataAvlbl	Transmit FIFO data available	0	R
21	RxDataAvlbl	Receive FIFO data available	0	R
31 - 22	-	reserved	0	R

31.3.12 Clear register

The clear register is a write-only register. The corresponding static status flag can be cleared by writing a ‘1’ to the corresponding bit in the register.

Table 275 MCI Clear register

BIT	NAME	DESCRIPTION	RESET	R/W
0	CmdCrcFailClr	Clears CmdCrcFail flag	0	W
1	DataCrcFailClr	Clears DataCrcFail flag	0	W
2	CmdTimeOutClr	Clears CmdTimeOut flag	0	W
3	DataTimeOutClr	Clears DataTimeOut flag	0	W
4	TxUnderrunClr	Clears TxUnderrun flag	0	W
5	RxOverrunClr	Clears RxOverrun flag	0	W
6	CmdRespEndClr	Clears CmdRespEnd flag	0	W
7	CmdSentClr	Clears CmdSent flag	0	W
8	DataEndClr	Clears DataEnd flag	0	W
9	StartBitErrClr	Clears StartBitErr flag	0	W
10	DataBlockEndClr	Clears DataBlockEnd flag	0	W
31 - 11	reserved		-	-

31.3.13 Interrupt Mask Registers

There are two interrupt mask register, Mask0-1, one for each interrupt request signal.

Table 276 MCI Mask Registers

BIT	NAME	DESCRIPTION	RESET	R/W
0	Mask0	Mask CmdCrcFail flag	0	R/W
1	Mask1	Mask DataCrcFail flag	0	R/W
2	Mask2	Mask CmdTimeOut flag	0	R/W
3	Mask3	Mask DataTimeOut flag	0	R/W
4	Mask4	Mask TxUnderrun flag	0	R/W
5	Mask5	Mask RxOverrun flag	0	R/W
6	Mask6	Mask CmdRespEnd flag	0	R/W
7	Mask7	Mask CmdSent flag	0	R/W
8	Mask8	Mask DataEnd flag	0	R/W
9	Mask9	Mask StartBitErr flag	0	R/W
10	Mask10	Mask DataBlockEnd flag	0	R/W
11	Mask11	Mask CmdActive flag	0	R/W
12	Mask12	Mask TxActive flag	0	R/W
13	Mask13	Mask RxActive flag	0	R/W
14	Mask14	Mask TxFifoHalfEmpty flag	0	R/W
15	Mask15	Mask RxFifoHalfFull flag	0	R/W
16	Mask16	Mask TxFifoFull flag	0	R/W
17	Mask17	Mask RxFifoFull flag	0	R/W
18	Mask18	Mask TxFifoEmpty flag	0	R/W
19	Mask19	Mask RxFifoEmpty flag	0	R/W
20	Mask20	Mask TxDataAvlbl flag	0	R/W
21	Mask21	Mask RxDataAvlbl flag	0	R/W
31 - 22	-	reserved	-	-

The interrupt mask registers determine which status flags generate an interrupt request by setting the corresponding bit to 1.

31.3.14 FIFO Counter Registers

The FIFOCnt register contains the remaining number of words to be written to or read from the FIFO. The FIFO counter loads the value from the data length register when the Enable bit is set in the data control register. If the data length is not word aligned (multiple of 4), remaining 1 to 3 bytes are regarded as a word.

Table 277 MCI FIFO Count Register

BIT	NAME	DESCRIPTION	RESET	R/W
14 - 0	DataCount	Remaining data	0	R
31 - 15	Reserved	-	-	-

31.3.15 Data FIFO Registers

The receive and transmit FIFOs can be read or write as 32-bit registers. The FIFOs contain 16 entries on 16 sequential addresses. This allows the CPU to use its load and store multiple operands to read/write to the FIFO.

Table 278 MCI FIFO Register

BIT	NAME	DESCRIPTION	RESET	R/W
31 - 0	Data	FIFO data	0	R/W

31.3.16 Peripheral Identification Registers

The PeriphID0-3 registers are four 8-bit registers, that span address locations 0xFE0 - 0xFEC. The registers can conceptually be treated as a single 32-bit registers. The read-only registers provide the following options of the peripheral:

- Part number [11:0] This is used to identify the peripheral. The three digit product code 180 is used for the MCI.
- Designer [19:12] This is the identification of the designer. 0x41 ARM Ltd.
- Revision number [23:20] This is the revision number of the peripheral. The revision number starts from 0.
- Configuration This is the configuration option for the peripheral. The configuration value is 0.

While designing a systems memory map it must be kept in mind that the register has 4KB-memory footprint.

The 4-bit revision number is implemented by instantiating a component called RevisionAnd four times with its inputs tied off as appropriate and the output sent to the read multiplexer.

All memory assesses to the peripheral identification registers must be 32-bit, using the LDR and STR instructions.

Table 279 PeriphID0 register

BIT	NAME	DESCRIPTION	RESET	R/W
7 - 0	Partnumber0	These bits read back as 0x80	0x80	R
31 - 8	Reserved	-	-	-

The PeriphID0 register is hard coded and the fields within the registers determine the reset value.

Table 280 PeriphID1 register

BIT	NAME	DESCRIPTION	RESET	R/W
3 - 0	Partnumber1	These bits read back as 0x1	1	R
7 - 4	Designer0	These bits read back as 0x1	1	R
31 - 8	Reserved	-	-	-

The PeriphID1 register is hard coded and the fields within the registers determine the reset value.

Table 281 PeriphID2 register

BIT	NAME	DESCRIPTION	RESET	R/W
3 - 0	Designer1	These bits read back as 0x4	4	R
7 - 4	Revision	These bits read back as 0x0	0	R
31 - 8	Reserved	-	-	-

The PeriphID2 register is hard coded and the fields within the registers determine the reset value.

Table 282 PeriphID3 register

BIT	NAME	DESCRIPTION	RESET	R/W
7 - 0	Configuration	These bits read back as 0x0	0	R
31 - 8	Reserved	-	-	-

The PeriphID3 register is hard coded and the fields within the registers determine the reset value.

31.3.17 Cell Identification Registers

The CellD0-3 registers are four 8-bit registers, that span address locations 0xFF0 - 0xFFC. The read-only registers can conceptually be treated as a single 32-bit registers. The registers is used as a standard cross-peripheral identification system.

Table 283 CellID0 register

BIT	NAME	DESCRIPTION	RESET	R/W
7 - 0	CellID0	These bits read back as 0x0D	0x0D	R
31 - 8	Reserved	-	-	-

The CellID0 register is hard coded and the fields within the registers determine the reset value.

Table 284 CellID1 register

BIT	NAME	DESCRIPTION	RESET	R/W
7 - 0	CellID1	These bits read back as 0xF0	0xF0	R
31 - 8	Reserved	-	-	-

The CellID1 register is hard coded and the fields within the registers determine the reset value.

Table 285 CellID2 register

BIT	NAME	DESCRIPTION	RESET	R/W
7 - 0	CellID2	These bits read back as 0x05	0x05	R
31 - 8	Reserved	-	-	-

The CellID0 register is hard coded and the fields within the registers determine the reset value.

Table 286 CellID3 register

BIT	NAME	DESCRIPTION	RESET	R/W
7 - 0	CellID3	These bits read back as 0xB1	0xB1	R
31 - 8	Reserved	-	-	-

The CellID0 register is hard coded and the fields within the registers determine the reset value.

32 AUDIO DSP SUBSYSTEM

32.1 Introduction

The audio dsp subsystem consists of the Epics7B DSP subsystem and the audio interfaces to the outside world which are an ADC input, a DAC output, I2S input and output and an SPIDF input.

The Epics7B DPS subsystem consists of the Epics7B DSP, DIO registers, memory, memory controller (sometimes confusingly referred to as ‘DMA’ controller) and interrupt controller.

For readability we will simplify some terms:

- “audio dsp subsystem” becomes “audio subsystem”
- “Epics7B DSP subsystem” becomes “dsp subsystem”
- “Epics7B DSP” becomes “Epics”

All audio interfaces have connections to the Epics and to the system bus.

The audio subsystem as a whole has two bus interfaces. One VPB slave interface and one AHB slave interface. Via the VPB interface we can reach the dsp subsystem’s DIO registers, the audio interfaces and the audio subsystem’s configuration registers. Via the AHB interface we can reach the memories and registers of the dsp subsystem.

32.2 Overview

The audio subsystem consist of the following blocks:

Table 287 Subblocks of the audio subsystem with their HDL module names

SUBBLOCK
ADC
ahb interface
bitslicer
configuration registers
dac
decimator latch
decimator (DEC)
digital audio input (DAI)
digital audio output (DAO)
dsp subsystem
edge detector
headphone amplifier
interpolator latch
interpolator (IPOL)
low noise amplifier (LNA)
nsof counter
programmable gain (PG)
simple audio input (SAI)
simple audio output (SAO)
single to differential converter (SD)
spdifin decoder
timestamp counter

Below follows a short description of each block.

- DAI1/2

  This part is an IIS input. It generates a parallel data signal for left and right and a newsam signal. The newsam signal is a strobe/latch signal that indicates that a new audio sample is available on the data lines, by going high for one clock cycle on the same clock edge as the new data becomes available. Depending on the selection of the input the format will be IIS, LSB justified or MSB justified.

- DAO

  This part will generate an IIS output data stream and depending on the setting it will be a IIS, LSB justified or MSB justified output format.

- SPDIFIN

  This part is an SPDIF receiver. In series a second DAI24 module is connected to convert the I2S output stream from the SPDIF receiver to a 24 bit parallel format.

  The status bits can be read by the ARM processor via the VPB interface from the audio configuration registers or by the Epics via its user defined control registers.

- ADC

  The audio subsystem has two16 bit adc's (for left and right). The outputstream of the adc is 128 times oversampled. Therefore there is a decimator connected in series to each adc.

- DAC

  The audio subsystem has two 16 bit dac's (for left and right). The inputstream of the dac is128 times oversampled. Therefore there is an interpolator connected in series to each dac.

- SAO

  This is a serial audio output interface from the VPB. It can be used to stream audio data from the VPB to the Epics or to the DAO or DAC audio outputs. The audio subsystem contains two of them.

- SAI

  This is a serial audio input interface to the VPB. It can be used to stream audio data to the VPB from the Epics or from the DAI or ADC or SPDIF audio inputs. The audio subsystem contains three of them.

- EDGE DETECTOR

  This module creates a NEWSAM signal (which is a latch enable signal) for the DAO from the DAO_WS. This word select is generated by the CGU.

- NSOF counter

  The NSOF counter contains another edge detector that generates a pulse from the usb frame signal. The frame signal is either a clock in case of the ip9021 usb 2.0 core or a toggle in case of the fs22 usb 1.1 core. The NSOF counter can be programmed to do either positive edge or dual edge detection, via the bit POS_EDGE_ONLY.

- Timestamp counter

  This is just a simple 24 bit wrap round counter on the dsp clock. The value of this counter is latched by a number of DIO registers. Each latch enable of these DIO registers is connected to a newsample signal of one of the audio sources. This way each audio source can have its own timestamp. In addition there is also a provision for the output audio samples.

- DSP subsystem

  The Epics7B DSP subsystem consists of the Epics7B DSP, DIO registers, memory, memory controller (sometimes confusingly referred to as 'DMA' controller and interrupt controller.

  The dsp subsystem is delivered with embedded DIO registers. All sources are fed to the DSP core via IO-addresses. The DSP will read these addresses and process the data. After processing, the data is sent back to a DSP IO-address which will go to a IIS output generator or to the CPU via the SAI block.

  The newsample signal of each audio source is connected to a user flag of the epics. Compared to the epics7a solution, the user flags(1) can be seen as a dedicated i-flag(2) per audio stream. Because the epics7b can detect new samples

per audio stream, it is now possible to do sample rate conversion from an arbitrary input sample frequency to an arbitrary output sample frequency. This also makes the generation of an asf pulse superfluous.

If the DSP is not used for audio-processing, the dsp subsystem can be put in idle mode and woken up on sample basis to process a sample and power-down again. The dsp subsystem can also be bypassed entirely.

- Configuration register block

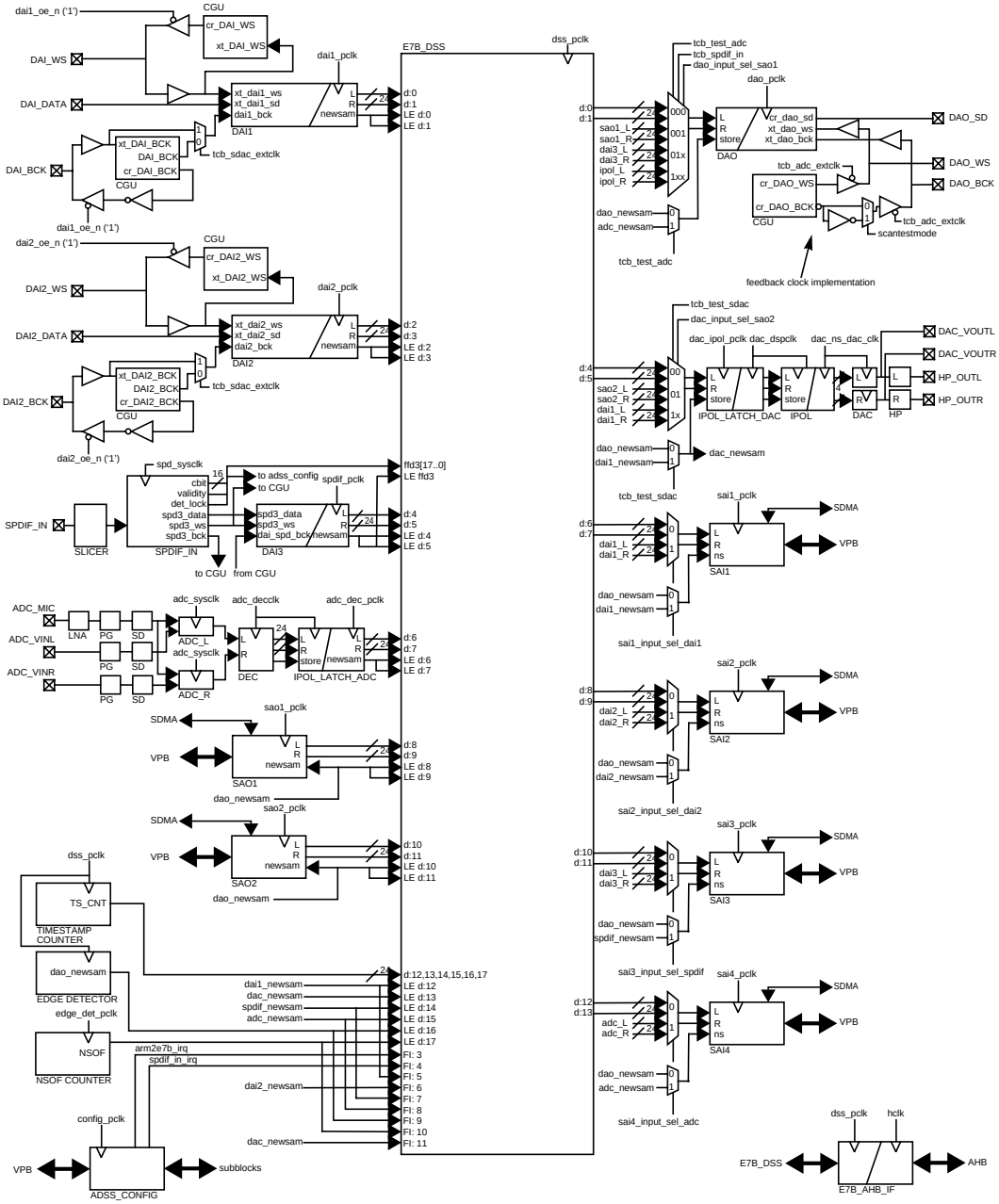
  This block offers a vpb slave interface to various configuration registers.

- AHB interface

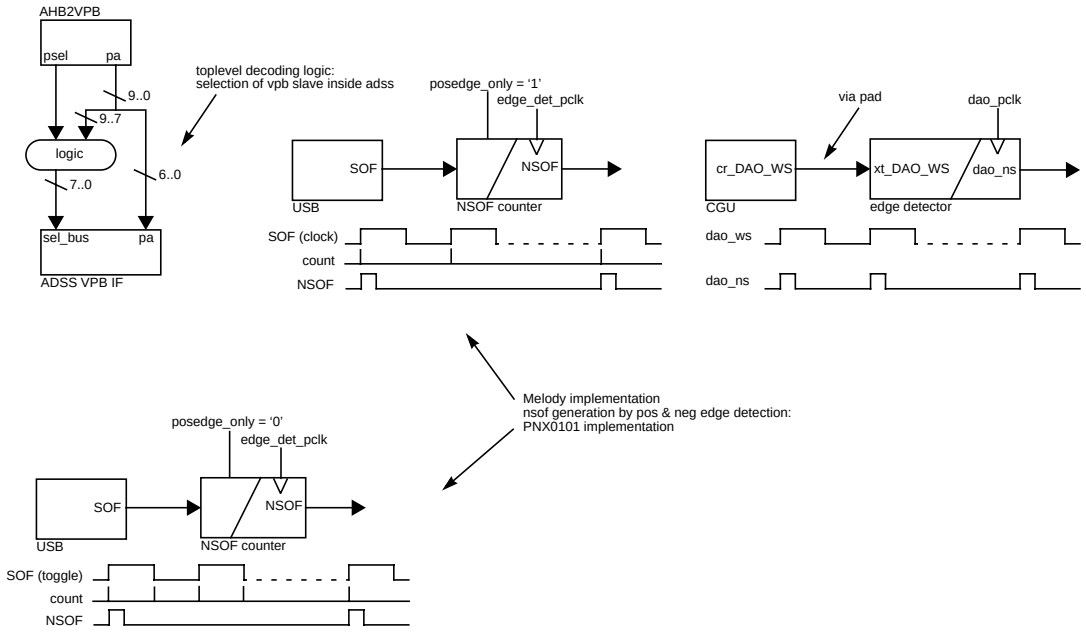
  This block offers an ahb slave interface to the dsp susbsystem.

(1) A user flag is a user defined IO flag of the Epics with a dedicated input and output pin on the Epics module boundary.
(2) The i-flag is removed on the Epics7B and was used on the Epics7A to indicate the presence of a new sample. On the Epics7B one or more user flags can be usd for this purpose.

32.3 Top level interconnect of the Audio DSP Subsystem



Audio DSP Subsystem - connectivity overview



Audio DSP Subsystem - additional details on top level interconnect

32.4 Functional Description

32.4.1 CLOCKS

The dsp subsystem is running on one single clock frequency that is independent of the other systemclocks. There is a clock domain crossing in the AHB2VPB bridge and there is a clock domain crossing in the AHB interface of the E7B_DSS.

The audio subsystem assumes to have only one output sample clock, which is generated by the audio pll. All other audio interfaces are assumed to be locked to this frequency or that sample rate conversion will take place by the Epics.

The clocks can be configured via the CGU registers. See the applicable section on how to control the clocks. In the table below is a complete list of the clocks of the audio subsystem.

All pclk's are basically the same clock originating from one and the same fracdiv. This means they can be considered as one and the same clock domain and no special hardware design measures for clock domain crossing are necesarry. For example, the newsam signal from DAI is one clock pulse high but this clock pulse is still safely sampled by the dsp subsystem. The only difference between the clocks is that they can be enabled seperately. This means each block that has it's own clock (pclk) can be brought into power down state seperately.

An advantage of having the pclks on the same clock domain as the vpb clock is that the audio interfaces do not need to have a clock domain crossing. Basically the clock domain crossing from the ARM (subsystem) to the audio subsystem is shifted outside the audio subsystem into the ahb2vpb bridge. In other words, no part of the audio subsystem, except parts of the ahb and vpb bus interfaces, is running on the ahb clock domain.

This implies that any other signal than bus signals crossing the border between the audio subsystem and the rest of the PNX0101 is crossing a clock domain. This specifically concerns the request and acknowledge signals to the simple dma controller and the interrupt request lines.

Table 288 Clocks of the audio subsystem

CLOCK NAME	BASE	DEFAULT FREQ [KHZ]
adc_decclk	CLK1024FS_BASE	5644.8
adc_sysclk	CLK1024FS_BASE	5644.8
ahb_clk	SYS_BASE	64000
config_pclk	AHB0_VPB3_ASYNC_PCLK_BASE	48000
cr_DAI_BCK	CLK1024FS_BASE	5644.8
cr_DAI_WS	CLK1024FS_BASE	44.1
cr_DAO_BCK	CLK1024FS_BASE	5644.8
cr_DAO_WS	CLK1024FS_BASE	44.1
dac_dsp_clk	CLK1024FS_BASE	5644.8
dac_ns_dac_clk	CLK1024FS_BASE	5644.8
DAI_BCK	DAI_BCK_BASE	external source
dai_pclk	AHB0_VPB3_ASYNC_PCLK_BASE	48000
dai_spd_bck	SPD3_BCK_BASE	external source
dao_pclk	AHB0_VPB3_ASYNC_PCLK_BASE	48000
dss_pclk	AHB0_VPB3_ASYNC_PCLK_BASE	48000
adc_dec_pclk	AHB0_VPB3_ASYNC_PCLK_BASE	48000
dac_ipol_pclk	AHB0_VPB3_ASYNC_PCLK_BASE	48000
edge_det_pclk	AHB0_VPB3_ASYNC_PCLK_BASE	48000
sai1_pclk	AHB0_VPB3_ASYNC_PCLK_BASE	48000
sai2_pclk	AHB0_VPB3_ASYNC_PCLK_BASE	48000
sai3_pclk	AHB0_VPB3_ASYNC_PCLK_BASE	48000
sai4_pclk	AHB0_VPB3_ASYNC_PCLK_BASE	48000
sao1_pclk	AHB0_VPB3_ASYNC_PCLK_BASE	48000
sao2_pclk	AHB0_VPB3_ASYNC_PCLK_BASE	48000
spdif_pclk	AHB0_VPB3_ASYNC_PCLK_BASE	48000
spd_sysclk	SYS_BASE	48000
(dai2_pclk)	AHB0_VPB3_ASYNC_PCLK_BASE	48000

One clock is not listed and this is the DAO_CLK, which is a 256*fs output of the audio pll going directly to a pin. This clock is not used in the audio subsystem. The frequency fractional division value of this clock can be configured indepently from the other clocks that are derived from the audio pll.

32.4.2 RESETS

All listed resets are controllable by software. The ahb reset ‘hreset’ (see the table below), will reset the entire ahb including the connected blocks, thus also the main cpu, resulting in a hangup. The second reset will reset the entire vpb3

domain (which only has the audio subsystem as a connected block). The rest of the resets are block resets only. Most blocks have their own reset, apart from the blocks that are on the same clockdomain.

Table 289 Available resets of the audio subsystem

RESET	CLOCK DOMAIN	CONNECTED BLOCKS
ahbif_rst_n	dss_pclk	e7b_ahb_if; e7b_dss; nsof; ts_cnt
dai_rst_n	dai_pclk	dai
dai2_rst_n	dai2_pclk	dai2
dao_rst_n	dao_pclk	dao
dec_rst_n	adc_decclk	decimator
edet_rst_n	edge_det_pclk	edge detector
hreset	hclk	e7b_ahb_if
int_rst_n	dac_dsp_clk	interpolator
pnres	config_pclk	audioss_config
sai1_rst_n	sai1_pclk	sai1
sai2_rst_n	sai2_pclk	sai2
sai3_rst_n	sai3_pclk	sai3
sai4_rst_n	sai4_pclk	sai4
sao1_rst_n	sao1_pclk	sao1
sao2_rst_n	sao2_pclk	sao2
spdif_rst_n	spdif_pclk	spdifin

32.4.3 DMA interface

Each fifo of the vpb audio interfaces (SAI & SAO) has a dedicated handshake to the simple dma (SDMA) controller. This is implemented by request outputs and clear inputs. See the table below.

Table 290 Simple DMA interface of the audio subsystem: regular req/ack method

DMA REQ LINE	DMA ACK LINE
req_left_sai1	clr_left_sai1
req_left_sai2	clr_left_sai2
req_left_sai3	clr_left_sai3
req_left_sai4	clr_left_sai4
req_left_sao1	clr_left_sao1
req_left_sao2	clr_left_sao2
req_right_sai1	clr_right_sai1
req_right_sai2	clr_right_sai2
req_right_sai3	clr_right_sai3
req_right_sai4	clr_right_sai4
req_right_sao1	clr_right_sao1
req_right_sao2	clr_right_sao2

In addition to the regular dma handshake of the vpb audio interfaces the epics has a special type of handshake to the simple dma controller. Instead of the regular request/acknowledge type of handshake a channel enable type of handshake is implemented by using an input and an output user flag (FI:2 and FO:1, see also section 32.5.3) of the dsp

subsystem. This dma channel needs to be configured for memory to memory copy so that this channel does not respond to the regular req and ack signals. With the channel enable handshake the Epics can trigger data transfers to and from system memory. For further details on this type of control refer to the section on the sdma in this document and the document Data Transfers from/to E7B, v0.2, by John van Tol, 5 september 2003.

No special measures are taken for clock domain crossing since all handshakes are a request/acknowledge type of handshake.

Table 291 Simple DMA interface of the audio subsystem: channel enable method

DMA LINE	CLOCK DOMAIN BOUNDARY
sdma_enable_ch_e7b	dss_pclk -> ahb0_clk
sdma_enable_ack_e7b	ahb0_clk -> dss_pclk

32.4.4 INTERRUPTS

Below an overview of all interrupts of the audio subsystem. See also section 32.5.3 for the mapping to the epics user flags of the e7b2arm_irq_1 and e7b2arm_irq_2.

The interrupts on the sai/sao blocks provide a backup for the sdma based transfer. These interrupts can be irq, since even at a worst case sample rate of 96 kHz and a fifo depth of 1 sample and assuming the ARM running on 48 MHz, we have 500 cycles for interrupt handling which should be sufficient.

The interrupt from the epics is choosen to be irq instead of fiq, taking into account that the epics is doing block based processing (it could ask eg. the ARM to place a new mp3 frame into memory). That way the fiq is reserved for sources that really need it.

The spdif_in_irq is generated by the audioss_config block which is able to detect a change in the spdif channel status bits. This interrupt is routed to the system interrupt controller (IP_1106) as well as to the dsp subsystem irq input (FI:4). Note that the DSP subsystem has its own integrated interrupt controller that is different from the system interrupt controller.

Table 292 Interrupt requests from the audio subsystem to the system interrupt controller

IRQ
e7b2arm_irq_1
e7b2arm_irq_2
sai1_irq
sai2_irq
sai3_irq
sai4_irq
sao1_irq
sao2_irq
spdif_in_irq

32.4.5 BYPASS MODES OF THE DSP SUBSYSTEM

The bypass modes listed below can be configured by activating the appropriate mux-select signal. These are bits of the ‘mux_settings’ register of the audioss_config block. See the section about the audioss_config block for a register map description.

Table 293 Bypass selection of the dsp subsystem

SELECTION SIGNAL	FROM	TO
dac_input_sel_sao2	SAO2	DAC
dao_input_sel_sao1	SAO1	DAO
sai1_input_sel_dai1	DAI	SAI1
sai2_input_sel_dai2	DAI	SAI2
sai3_input_sel_spdif	SPDIFIN	SAI3
sai4_input_sel_adc	ADC	SAI4
tcb_spdif_in	SPDIFIN	DAO
tcb_test_adc	ADC	DAO
tcb_test_sdac	DAI	DAC

32.4.6 DAI

The DAI can be used in slave and master mode. In slave mode an external IIS source generates the bitclock and in master mode the PNX0101 IC generates the bitclock.

In slave mode the bitclock arrives on pad DAI_BCK and is led to the CGU input xt_DAI_BCK. This input should be switched directly to the CGU output DAI_BCK which delivers the bit clock for the DAI block. The bitclock needs to pass through the cgu because during scantest mode the testclock needs to be inserted on all clocks, so also on the bitclock. In slave mode the audio pll needs to lock on the incoming source. This can best be done on the bitclock or on the word select. The bitclock is the preferred source because of it's higher frequency. The pl550 has problems with locking on frequencies below 100 kHz. If the ratio between the bitclock and the sample frequency is not known, the source word select can be used. The digital audio source will put out the data and the word select on the negative edge of the bitclock and these will be sampled by the DAI block on the positive edge of the bitclock.

In master mode the audio pll is set on the desired frequency by making use of the oscillator input as a frequency reference. The CGU then generates both the word select and the bitclock of the IIS input stream. In that case the pads DAI_WS and DAI_BCK need to be set to output by resetting the dai1_oe_n signal. This can be done from the audioss_config block. The bitclock is inverted in master mode, because the word select needs to be generated on the negative clock edge.

During test and evaluation of the sdac the bitclock needs to come directly from the pad and not through the CGU, because the CGU cannot be configured to switch it's output DAI_BCK to xt_DAI_BCK: no software is running.

32.4.7 DAO

The DAO can only be used in master mode. For this reason the output enable of the DAO_WS and DAO_BCK pads is always active in functional mode. The pads can only be switched to input mode by the tcb_adc_extclk, because the CGU cannot be programmed to generate the word select and bitclock during testmode.

The bitclock generated by the CGU is inverted with respect to the word select, such that word select changes on a negative edge of the bitclock.

In functional mode the dao gets its input from either the dsp subsystem or the sao1. In either case the newsam (or 'store') signal is derived from cr_DAO_WS from the CGU. The frequency of this signal is set by programming the audio pll to 1024 times this frequency and setting the corresponding fracdiv to 1024, which will generate a cr_DAO_WS with the desired frequency. The word select is then going to the edge detector, which gives a pulse on the rising edge of the word select. This pulse is synchronized to the dss_pclk and is one dss_pclk clock period high. This is necessary because both input sources (epics and sao1) are running on the vpb synchronized clocks dss_pclk and sao1_pclk. For the same reason the sao and the dsp subsystem are also connected to dao_newsam. The software running on the epics should synchronize it's output samples to FI:9, which is connected to dao_newsam.

32.4.8 SPDIF INPUT

The spdif input hardware consists of a series connection of the bitslicer, which is an analog module, the spdif decoder and a DAI block. This DAI block is exactly the same hardware as the other DAI block that is connected directly to the IIS input pads. The only difference between those DAI blocks is that the input format of the spdif DAI block is fixed in hardware to accept the spd3 format only.

The spdif decoder is running on a dedicated clock called spd_sysclk, which should lie between 36 and 69 MHz. In this clockdomain a signal spd3_bck is generated, which is treated by the DAI block as a bitclock. This bitclock is again routed via the cgu to be able to insert the testclock during testmode. The spdifin decoder puts out the data on the negative edge of spd3_bck. The DAI will latch the data on the positive edge of the bitclock. This guarantees reliable data transfer even though the clock is delayed by the path through the CGU.

The word select from the spdif in decoder is routed to the cgu. This makes it possible to lock the audio pll to the incoming spdif stream.

The bitslicer supply can be configured independently by a bit in the audioss_config block.

32.4.9 ADC

The ADC circuitry consists of two identical adc's, one for the left and one for the right channel. Each adc has two inputs so source selection can be performed: a line input or a microphone input is sampled. The microphone and line inputs have a programmable gain (PG) amplifier which can tune the signal from 0 to 24 dB. Additionally the microphone input has a low noise amplifier (LNA) with 30 dB gain. The output of the PG is fed to a single to differential converter (SD) and then fed to the ADC. The output of the ADC is a bitstream at 128*fs, which is the frequency set by adc_sysclk. The decimator (DEC) converts this bitstream to 24 bit parallel format clocked on the sample rate. The decimator contains a ROM for the filter coefficients and a RAM. The data generated by the decimator needs to be synchronized to the dss_pclk, which is done by the interpolator_latch_adc block.

32.4.10 DAC

The DAC circuitry receives 24 bit parallel data from either the dsp subsystem or the sao2. This data is resynchronized to dac_dspclk that runs on 128*fs by the interpolator_latch_dac. The resynchronized data is then upsampled to 128*fs by the interpolator. The interpolator also contains a noise shaper, which runs on either 128*fs or 256*fs and is clocked by dac_ns_dac_clk. For optimal signal to noise ratio the noise shaper (and dac) should run on 256*fs in case of audio sample frequencies below 32 kHz. The noise shaper output and dac input consists of 4 bit samples (quantization noise is shifted outside of the audible frequency band). The DAC output is fed to the line outputs and to a headphone driver (HP) that drives the headphone outputs.

32.4.11 SAO

The SAO is used to generate an audio stream from the vpb interface. From the vpb a 4 sample fifo can be filled, which is then clocked out on the sample frequency set by the audio pll (by dao_newsam - see section 32.4.7). There are two SAO interfaces, which can be used to stream data to the DAO or the SDAC respectively.

The SAO can stream it's data also to the dsp subsystem, to enable sound post processing by the epics.

32.4.12 SAI

The SAI is used to stream audio data to the vpb bus. This is done by storing the 24 bit audio stream in a 4 sample fifo, which can then be read from the vpb interface. The audio source can be one of the audio interfaces or the dsp subsystem. The audio interfaces are the DAI, the SPDIF input and the ADC. In that case the data is clocked into the fifo by a strobe signal from the audio interface (dai_newsam, spdif_newsam or adc_newsam respectively). Each audio interface has it's own SAI.

When the audio source is the dsp subsystem, the samples are clocked into the fifo by the audio pll (by dao_newsam - see section 32.4.7). In that case the epics should synchronize it's output data with FI:9.

32.4.13 TIMESTAMP COUNTER

The timestamp counter output is hardwired to 6 dio input registers. Each input register will be latched by another strobe signal. These strobe signals are generated by the audio interfaces DAI, SPDIF, ADC, USB (NSOF - see next section), DAO and DAC. This way each sample of each audio source and sink can be labelled with a timestamp. The timestamp increases by one every dss_pclk tic, and will wrap-round at value 2^24-1.

32.4.14 NSOF COUNTER

The NSOF is intended to indicate the presence or need of new audio data in case USB streaming audio is used as an input or output. The NSOF generates a pulse per a programmable number of USB frames. This number of frames can be programmed with the audioss_config block. (Refer for a more detailed description to USB Streaming Audio, SSA1 platform, by J.Ph van Tol, 13 may 2002)

The NSOF generates just one pulse on dss_pclk. Meaning we need an edge sensitive interrupt mechanism to detect this. This can easily be implemented by putting the corresponding epics user flag FI:10 into stretch mode.

32.5 Software Interface Specification

32.5.1 Memory of the dsp subsystem

Both the ARM and the Epics can access all memories of the DSP subsystem. The Epics can also trigger data transfers to and from system memory: the ARM must program the SDMA and the Epics will enable this channel when needed.

The memory configuration of the dsp subsystem is specified in the table below. In fact the delivery of the dsp subsystem contains larger memories, so that a wrapper has been used around the truely instantiated memories to mimic the interface of the larger memories.

Table 294 Memory map of the DSP subsystem

ARM ADDRESS	DSP ADDRESS	NAME	SIZE (WORDS)	WIDTH
0x106F.FFF8 - 0x106F.FFFC	-	DMA access registers	-	32 bit
-	-	not used	-	-
0x106F.FC00 - 0x106F.FC48	0xFF00 - 0xFF11	DIO registers	-	24 bit
-	-	not used	-	-
0x106A.0000 - 0x106A.1FFC	0x8000 - 0x87FF	PRAM	2k	32 bit
0x1068.0000 - 0x1068.DFFC	0x0000 - 0x37FF	PROM	14k	32 bit
-	0x0000 - 0x00FF	BIOSROM	256	32 bit
0x1066.0000 - 0x1066.0FFC	0x8000 - 0x83FF	YRAM	1k	12 bit
0x1064.0000 - 0x1064.8FFC	0x0000 - 0x23FF	YROM	9k	12 bit
0x1063.FF00 - 0x1063.FFFC	0xFFC0 - 0xFFFF	DSP control registers	-	24 bit
0x1060.0000 - 0x1060.9FFC	0x0000 - 0x27FF	XRAM	10k	24 bit

Input register FFD3 is used to read the spdif channel status bits (FFD3[17..2]) and the spdif validity (FFD3[1]) and lock (FFD3[0]) signals. This is a hardwired connection from the SPDIF block. The latch enable of this register is tied to logic_1.

Output registers FFDF and FFDE are used by the Epics as IRQ reason registers to indicate the reason of the interrupt it gives to the ARM via FO16 (IRQ1) and FO17 (IRQ2). This does not imply any wiring, it is just an agreement to use these registers to indicate the reason.

Table 295 DSP CONTROL registers

REGISTER NAME	BASE	WIDTH	COMMENT
DSP program counter	0x0FFFF	16	
DSP status 1	0X0FFFE	24	
DSP status 2	0X0FFFD	5	
DSP interrupt stack	0X0FFFC	16	
DSP IO configuration 1	0X0FFFB	18	
DSP IO configuration 2	0X0FFFA	18	
INTC polarity	0X0FFF9	18	
INTC mode	0X0FFF8	18	
INTC mask	0X0FFF7	18	
INTC status	0X0FFF6	-	
INTC test	0X0FFF5	18	
INTC SW Clear	0X0FFF4	18	
IRQ userflag	0X0FFF3	18	
DMAC IRQ counter	0X0FFF2	16	
Output register	0X0FFDF	24	BIOS ROM version 0x101
Output register	0X0FFDE	24	
Output register	0X0FFDD	24	
Output register	0X0FFDC	24	
Output register	0X0FFDB	24	
Output register	0X0FFDA	24	
Output register	0X0FFD9	24	
Output register	0X0FFD8	24	
Output register	0X0FFD7	24	
Output register	0X0FFD6	24	
Output register	0X0FFD5	24	
Output register	0X0FFD4	24	
Input register	0X0FFD3	24	SPDIF (see text)
Input register	0X0FFD2	24	
Input register	0X0FFD1	24	
Input register	0X0FFD0	24	
Input register	0X0FFCF	24	
Input register	0X0FFCE	24	
Input register	0X0FFCD	24	
Input register	0X0FFCC	24	

32.5.2 Mapping of the DIO registers

The interface from the audio peripherals to the dsp subsystem consists of 18 registers of 24 bits. See the two tables below for the details.

Table 296 DIO input register connections

DIO ADDRESS	DIO NAME	COMMENT
I0	DAI L	
I1	DAI R	
I2	DAI2 L	
I3	DAI2 R	
I4	SPDIFin L	
I5	SPDIFin R	
I6	ADC L	
I7	ADC R	
I8	SAO1 L	sao1 also connected to dao
I9	SAO1 R	
I10	SAO2 L	sao2 also connected to dac
I11	SAO2 R	
I12	TSTAMP DAI	
I13	TSTAMP DAC	redundant with tstamp of DAI or DAO
I14	TSTAMP SPDIF	
I15	TSTAMP ADC	
I16	TSTAMP DAO	
I17	TSTAMP USB	

Table 297 DIO output register connections

DIO ADDRESS	DIO NAME	COMMENT
O0	DAO L	dao1 also connected to sao1
O1	DAO R	
O2	future expansion	
O3	future expansion	
O4	DAC L	dac also connected to sao2
O5	DAC R	
O6	SAI1 L	sai1 also connected to dai1
O7	SAI1 R	
O8	SAI2 L	sai2 also connected to dai2
O9	SAI2 R	
O10	SAI3 L	sai3 also connected to spdif
O11	SAI3 R	
O12	SAI4 L	sai4 also connected to adc
O13	SAI4 R	
O14 - O17	future expansion	

32.5.3 User Flag Mapping

The tables below show how the user flags of the dsp subsystem are connected inside the audio subsystem. The user flags have an input (FI) and output (FO). The input user flags can be configured as either a normal flag or as an interrupt. In the latter case the flag is routed to an interrupt input of the interrupt controller inside the dsp subsystem, which will then activate the irq pin of the epics.

The indication ‘not connected’ means that the flags should be left not connected, because their functionality is blocked by the current configuration.

The indication ‘future expansion’ means that the flags are currently not connected but may be connected to additional hardware in future.

Table 298 User flag mapping: Input flags / Interrupt requests

FI / IRQ	FROM	COMMENT
0	future expansion	
1	not connected	
2	sdma_enable_ack_e7b	acknowledge from sdma channel enable
3	e7b_irq_req	this is an irq from the ARM via the audioss_config of the audio subsystem
4	spdif_in_irq	this is an irq from SPDIFin via the audioss_config of the audio subsystem
5	newsam DAI	
6	newsam DAI2	
7	newsam SPDIF	
8	newsam ADC	
9	newsam DAO	
10	NSOF	
11	newsam DAC	redundant with newsamp of DAI or DAO
12	FI12	to pin - used for evaluation purposes
13	FI13	to pin - used for evaluation purposes
14 - 17	not connected	

Table 299 User flag mapping: Ouput flags / Interrupt acknowledges

FO / IRQ_ACK	TO	COMMENT
0	future expansion	
1	sdma_enable_ch_e7b	
2	not connected	
3	e7b_irq_ack	connected to audioss_config to clear interrupt from arm
4 - 11	not connected	not connected because corresponding FI can be used as IRQ
12 - 13	not connected	
14	FO14	to pin - used for evaluation purposes
15	FO15	to pin - used for evaluation purposes
16	e7b2arm_irq_1	IRQ1 to system interrupt controller
17	e7b2arm_irq_2	IRQ2 to system interrupt controller

32.5.4 PROGRAMMER'S GUIDE

The main idea behind the architecture of the audio subsystem is that we have an audio output running on a frequency that is dictated by the audio pll. With this setup it is possible to:

- let the audio pll lock on an input source and slave the whole system to this source.
- set a fixed frequency on the audio pll, by using the oscillator and let both audio source and sink slave to this.
- set a fixed frequency on the audio pll and still slave the input devices to an external audio source - sample rate conversion is now needed and can be performed by the Epics.

In the Epics7B dsp subsystem "stretch mode" is an option for all user flags and is needed to reliably detect a pulse if a polling loop is used. Without the stretch the software detection cannot not be done reliably, because in a polling loop there is a moment in time when a branch is executed and thus the input flag status cannot be checked.

When a user flag FI is used as an irq source, the corresponding FO will be reserved for the irq-ack from the embedded interrupt controller in the dsp subsystem and cannot be assigned another function.

The Epics can give an interrupt to the ARM via FOx to the system interrupt controller(1). There is no way the interrupt handler, which is running on the ARM processor, can clear this interrupt: this can only be done by a software module running on the Epics. Since the system interrupt controller can handle only level sensitive interrupts, the ARM interrupt handler must signal to the Epics that the interrupt should be cleared. This signal could be an interrupt from the ARM to the Epics. The ARM can indicate the interrupt reason in the appropriate e7b_dss control register, and it can generate the interrupt itself by writing to the appropriate register in audio_creg.

To be able to perform software sample rate conversion from each source to each sink without too much hardware settings, each source and sink have a dedicated timestamp register in the dsp subsystem plus a dedicated user flag input.

When an external source (DAI or SPDIF) dictates the sample frequency, the audio pll can be locked on this source so no sample rate conversion is necesarry.

When streaming audio from the ARM system to the audio subsystem via the VPB interface (SAO) to the DAO or DAC, the whole audio sample chain must run on the audio pll sample frequency, because the newsample signal of the SAO is connected to the audio pll output (dao_newsam) only. This means that when sample rate conversion is needed on prerecorded data (eg. an mp3 or pcm file recorded on 32 kHz, that needs to be mixed with a 44.1 kHz IIS source and output to a 44.1 kHz IIS sink), the epics needs to do this by (block based) data retrieval via the AHB interface.

(1) To be implemented: the Epics can also give an interrupt to the ARM via FOx to the event router. The ARM interrupt controller can now clear the interrupt. It is up to the software designer to decide whether to use a direct irq or an irq via the event router. In the first case the ARM interrupt handler needs to wait until the Epics clears the interrupt. In the second case the ARM needs some time to find out which event caused the interrupt.

The word select signals of DAI and DAO are derived from the same fracdiv (at least when the cgu is configured in default chip mode) which means that when PNX0101 is IIS master both source and sink have the same sample frequency.

When no sample rate conversion is performed by the Epics, but still some kind of sound processing is done by the Epics, it is necesarry to trigger the latching of source data by an input flag and to trigger the generation of a new sample pair by a dao_newsam (FI:9). An additional problem will be caused by clock jitter. Clock jitter will cause slight phase differences between the source trigger and sink trigger. Clock jitter can occur when eg. the audio pll locks to an external IIS source and we put out data on the DAO or the DAC. This is possible because dai_bck generated by the CGU is not derived from the audio pll but simply passed thru from the input and dao_bck is derived from the audio pll. The result is that dai_newsam and dao_newsam can jitter with respect to each other, although they are frequency locked. It is now possible that the Epics is offered a new sample pair while still waiting for a trigger to output the result from the previous pair, or that the Epics is asked to produce a new sample pair while not having received yet an input sample pair. This should be resolved in software by a short input or output fifo.

33 AUDIO CONFIG

33.1 Overview

33.1.1 FEATURES

- DAI1 and DAO1 I2S input/output format settings
- Status of SPDIF IN module
- SPDIF IN interrupt request to ARM interrupt controller, with enabling, status and clear functionality
- SDAC control and status registers
- SADC control and status registers
- Interrupt request to EPICS7B, with automatically clearing register
- E7B AHB Interface settings
- N-SOF Counter setting for USB audio streaming

33.2 Architecture

33.2.1 HARDWARE INTERFACE AND SIGNAL DESCRIPTION

The Audio Config Subsystem interface is an external interface from the multi-layer APB to the AudioConfig module. Fig.55 on page 409 shows the Audio Config Toplevel block diagram. Table 318 describes the pin connection of the interface.

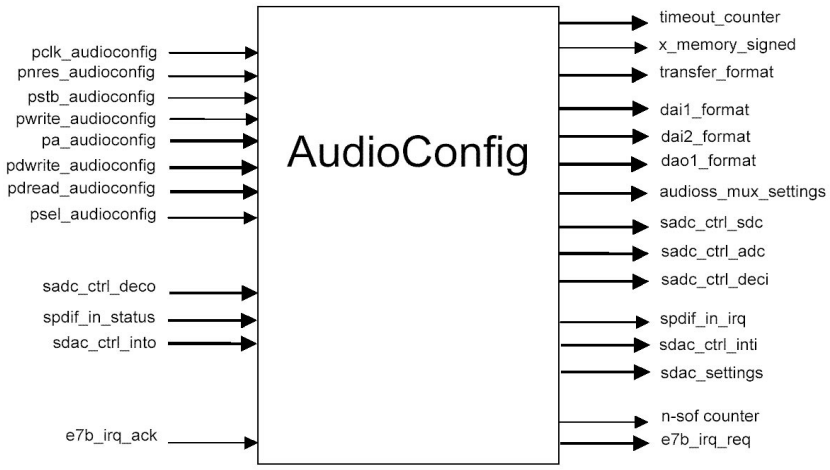


Fig. 55 Audio Config Toplevel

33.2.2 Audio Sub System Config Memory Map

The Audio Sub System Config memory map contians all defined Audio control bits. In Table 300 the register map overview is depicted.

Table 300 Memory Map

ADDRESS SPACE	OFFSET	REGISTER NAME	R/W ACCESS	RESET
0x80200380	0x00	IIS_FORMAT_SETTINGS	R/W	0xDB
	0x04	AUDIOSS_MUX_SETTINGS	R/W	0x180
	0x08	SPDIF_IN_STATUS	R	0x0
	0x0C	SPDIF_IN_IRQ_EN	R/W	0x0
	0x10	SPDIF_IN_IRQ_STATUS	R	0x0
	0x14	SPDIF_IN_IRQ_CLEAR	W	-
	0x18	SDAC_CTRL_INTI	R/W	0x0
	0x1C	SDAC_CTRL_INTO	R	0x0
	0x20	SDAC_SETTINGS	R/W	0x1000
	0x24	SADC_CTRL_SDC	R/W	0x0
	0x28	SADC_CTRL_ADC	R/W	0x0
	0x2C	SADC_CTRL_DECI	R/W	0x0
	0x30	SADC_CTRL_DECO	R	0x0
	0x34	E7B_INTERRUPT_REQ	R/W	0x0
	0x38	E7B_AHB_SETTINGS	R/W	0x0
	0x3C	N_SOF_COUNTER	R/W	0x0

33.2.3 IIS_FORMAT_SETTINGS

IIS format of the data-stream, which is processed, see Table 301.

Table 301 IIS_FORMAT_SETTINGS

BIT	VARIABLE	DESCRIPTION	RESET	R/W
2 - 0	DAI1_FORMAT	DAI1 IIS1 input format	0x3	R/W
5 - 3	DAI2_FORMAT	DAI2 IIS1 input format	0x3	R/W
8 - 6	DAO1_FORMAT	DAO1 IIS1 output format	0x3	R/W
31 - 9		Reserved		

Table 302 DAI1 - DAI2 IIS input format

FORMAT			OUTPUT
BIT 2	BIT 1	BIT 0	
0	0	0	Undefined
0	0	1	Undefined
0	1	0	Undefined
0	1	1	PHILIPS IIS (Standard IIS)
1	0	0	LSB justified 16 bits
1	0	1	LSB justified 18 bits
1	1	0	LSB justified 20 bits
1	1	1	LSB justified 24 bits

Table 303 DAO1 IIS output format

FORMAT			OUTPUT
BIT 2	BIT 1	BIT 0	
0	0	0	Undefined
0	0	1	Undefined
0	1	0	Undefined
0	1	1	PHILIPS IIS (Standard IIS)
1	0	0	LSB justified 16 bits
1	0	1	LSB justified 18 bits
1	1	0	LSB justified 20 bits
1	1	1	LSB justified 24 bits

33.2.4 AUDIOSS MUX SETTINGS

The audio mux settings register can select digital audio input to digital audio output.

Table 304 AUDIO MUX SETTINGS

BIT	VARIABLE	DESCRIPTION	R/W	RESET
0	dao_input_sel_sao1	Bypass selection: 0: E7B to DAO 1: SAO1 to DAO	R/W	0x0
1	dac_input_sel_sao2	Bypass selection: 0: E7B to DAC 1: SAO2 to DAC	R/W	0x0
2	sai1_input_sel_dai1	Bypass selection: 0: E7B to SAI1 1: DAI1 to SAI1	R/W	0x0
3	sai2_input_sel_dai2	Bypass selection: 0: E7B to SAI2 1: DAI2 to SAI2	R/W	0x0
4	sai3_input_sel_spdif	Bypass selection: 0: E7B to SAI3 1: SPDIF to SAI3	R/W	0x0

BIT	VARIABLE	DESCRIPTION	R/W	RESET
5	sai4_input_sel_adc	Bypass selection: 0: E7B to SAI4 1: ADC to SAI4	R/W	0x0
6	pos_edge_only	NSOF Counter: 0: Negative and positive edge detection 1: Possitive edge detection only	R/W	0x0
7	dai1_oe_n	output enable of DAI_BCK and DAI_WS pads of dai1	R/W	0x1
8	dai2_oe_n	output enable of DAI_BCK and DAI_WS pads of dai2	R/W	0x1
9	spdif_pon	0: Power off SPDIFin 1: Power on SPDIFin	R/W	0x0
31- 10		Reserved		

33.2.5 SPDIF_IN_STATUS

For the supported channel status bits see Table 305.

Table 305 SPDIF_IN_STATUS

BIT	VARIABLE	DESCRIPTION	R/W
0	spdif_lock	SPDIF lock status	R
1	spdif_validity	SPDIF audio validity	R
2	spdif_non_pcm	SPDIF channel status bit 1	R
3	spdif_pre_emph	SPDIF channel status bit 3	R
7 - 4	spdif_samp_freq	SPDIF channel status bits 24 till 27	R
9 - 8	spdif_accuracy	SPDIF channel status bits 28 till 29	R
13-10	spdif_wordlength	SPDIF channel status bits 32 till 35	R
17-14	spdif_orig_fs	SPDIF channel status bits 36 till 39	R
31- 18		Reserved	

spdif_lock: SPDIF lock status bit

0: no-lock

1: locked

spdif_validity: SPDIF audio validity bit

0: valid audio

1: non-valid audio

spdif_pcm: SPDIF channel status bit 1

0: PCM audio

1: non-PCM audio

spdif_pre_emph: SPDIF channel status bit 3

0: no pre-emphasis

1: with 50/15 us pre-emphasis

spdif_sam_freq: SPDIF channel status bits 24 till 27: Sample frequency

Table 306 SPDIF channel status bits 24 till 27: Sample frequency

BIT 7	BIT 6	BIT 5	BIT 4	SAMPLE FREQUENCY
0	0	0	0	44.1 kHz
0	0	0	1	Not indicated
0	0	1	0	48 kHz
0	0	1	1	32 kHz
0	1	0	0	22.05 kHz
0	1	0	1	Undefined
0	1	1	0	24 kHz
0	1	1	1	Undefined
1	0	0	0	88.2 kHz
1	0	0	1	Undefined
1	0	1	0	96 kHz
1	0	1	1	Undefined
1	1	0	0	176.4 kHz
1	1	0	1	Undefined
1	1	1	0	192 kHz
1	1	1	1	Undefined

spdif_accuracy: SPDIF channel status bits 28 till 29: Clock accuracy

Table 307 SPDIF channel status bits 28 till 29: Clock accuracy

BIT 9	BIT 8	CLOCK ACCURACY
0	0	Level II (1000 ppm)
0	1	Level I (50 ppm)
1	0	Level III (12.5%)
1	1	Sample rate differs from frame rate

spdif_wordlength SPDIF channel status bits 32 till 35: Word length indication

Table 308 SPDIF channel status bits 32 till 35: Word length indication

BIT 13	BIT 12	BIT 11	BIT 10	WORD LENGTH INDICATION
0	0	0	0	Not indicated, max. 20 bits
0	0	0	1	Not indicated, max. 24 bits
0	0	1	0	16 bits
0	0	1	1	20 bits
0	1	0	0	18 bits
0	1	0	1	22 bits
0	1	1	0	Undefined
0	1	1	1	Undefined
1	0	0	0	19 bits
1	0	0	1	23 bits
1	0	1	0	20 bits
1	0	1	1	24 bits
1	1	0	0	17 bits
1	1	0	1	21 bits
1	1	1	0	Undefined
1	1	1	1	Undefined

spdif_orig_fs: SPDIF channel status bits 36 till 39: Original sample frequency

Table 309 SPDIF channel status bits 36 till 39: Original sample frequency

BIT 17	BIT 16	BIT 15	BIT 14	SAMPLE FREQUENCY
0	0	0	0	Not indicated
0	0	0	1	192 kHz
0	0	1	0	12 kHz
0	0	1	1	176.4 kHz
0	1	0	0	Undefined
0	1	0	1	96 kHz
0	1	1	0	8 kHz
0	1	1	1	88.2 kHz
1	0	0	0	16 kHz
1	0	0	1	24 kHz
1	0	1	0	11.025 kHz
1	0	1	1	22.05 kHz
1	1	0	0	32 kHz
1	1	0	1	48 kHz
1	1	1	0	Undefined
1	1	1	1	44.1 kHz

33.2.6 SPDIF_IN_IRQ_EN

This register contains a variable to enable/disable the interrupt request generation to the ARM interrupt controller.

Table 310 SPDIF_IN_IRQ_EN

BIT	VARIABLE	DESCRIPTION	RESET	R/W
0	SPDIF_IN_IRQ_EN	SPDIF IN interrupt request enable	0x0	R/W
31 - 1		Reserved		

SPDIF_IN_IRQ_EN: Interrupt enable

0: Disable SPDIF_IN_IRQ_EN interrupt request

1: Enable SPDIF_IN_IRQ_

33.2.7 SPDIF_IN_IRQ_STATUS

This register contains a status variable that indicates the presence of the interrupt condition. It is read-only. Interrupt (SPDIF_IN_IRQ) will be pending when SPDIF_IN_STATUS register value is changed with respect to previous SPDIF_IN_STATUS register value.

Table 311 SPDIF_IN_IRQ_STATUS

BIT	VARIABLE	DESCRIPTION	RESET	R/W
0	SPDIF_IN_IRQ_STATUS	SPDIF IN interrupt request status	0x0	R
31 - 1		Reserved		

SPDIF_IN_IRQ_STATUS: Interrupt status

0: Non interrupt pending

1: Interrupt pending

33.2.8 SPDIF_IN_IRQ_CLEAR

A write action to this address location allows clearing the SPDIF_IN_IRQ

Table 312 SPDIF_IN_IRQ_CLEAR

BIT	VARIABLE	DESCRIPTION	RESET	R/W
0	SPDIF_IN_IRQ_CLEAR	SPDIF IN interrupt request status	0x0	W
31 - 1		Reserved		

SPDIF_IN_IRQ_CLEAR: Interrupt clear

0: No effect

1: Clear SPDIF_IN_IRQ

33.2.9 SDAC_CTRL_INTI

The interpolation filter is controlled by the control bus sdac_ctrl_inti[29:0].

Table 313 SDAC_CTRL_INTI

BIT	VARIABLE	VALUE	DESCRIPTION	R/W
7 - 0	SDAC_CTRL_INTI[7:0]	0000.0000	0 dB gain right channel	R/W
		0000.0001	-0.25 dB gain right channel	R/W
		:	:	R/W
		1100.0111	-49.75 dB gain right channel	R/W
		1100.1000	-50 dB gain right channel	R/W
		1100.1100	-53 dB gain right channel	R/W
		1101.0000	-56 dB gain right channel	R/W
		:	:	R/W
		1110.1100	-76.3 dB gain right channel	R/W
		1111.0000	-80.8 dB gain right channel	R/W
		1111.0100	-84.3 dB gain right channel	R/W
		1111.1000	-90.3 dB gain right channel	R/W
		1111.1100	-∞ dB gain right channel (mute)	R/W
15 - 8	SDAC_CTRL_INTI[15:8]	0000.0000	0 dB gain left channel	R/W
		0000.0001	-0.25 dB gain left channel	R/W
		:		R/W
		1100.0111	-49.75 dB gain left channel	R/W
		1100.1000	-50 dB gain left channel	R/W
		1100.1100	-53 dB gain left channel	R/W
		1101.0000	-56 dB gain left channel	R/W
		:		R/W
		1110.1100	-76.3 dB gain left channel	R/W
		1111.0000	-80.8 dB gain left channel	R/W
		1111.0100	-84.3 dB gain left channel	R/W
		1111.1000	-90.3 dB gain left channel	R/W
		1111.1100	-∞ dB gain left channel (mute)	R/W
18 - 16	SDAC_CTRL_INTI[18:16]	000	No digital de-emphasis	R/W
		001	Digital de-emphasis for fs = 32 kHz	R/W
		010	Digital de-emphasis for fs = 44.1 kHz	R/W
		011	Digital de-emphasis for fs = 48 kHz	R/W
		100	Digital de-emphasis for fs = 96kHz	R/W
		101 - 111	No digital de-emphasis	R/W
19	SDAC_CTRL_INTI[19]	0	No muting or de-mute with raised cosine roll-off	R/W
		1	Soft muting with raised cosine roll-off	R/W
21 - 20	SDAC_CTRL_INTI[21:20]	00	Normal speed mode: 8 kHz < fs < 96 kHz	R/W
		01/10	Double speed mode: 96 kHz < fs < 200 kHz	R/W
		11	DSD mode	R/W
23 - 22	SDAC_CTRL_INTI[23:22]	00/01/11	2fs to 8fs filter coefficients for slow roll-off	R/W
		10	2fs to 8fs filter coefficients for sharp roll-off	R/W

BIT	VARIABLE	VALUE	DESCRIPTION	R/W
24	SDAC_CTRL_INTI[24]	0	power down/up slope: 65536 * DSP clock cycles (11.6 ms @ 128fs, fs = 44.1 kHz	R/W
		1	power down/up slope: 131072 * DSP clock cycles (23.2 ms @ 128fs, fs = 44.1 kHz	R/W
25	SDAC_CTRL_INTI[25]	0	power down sequence disabled, i.e. DC ramp up followed by raised cosine	R/W
		1	power down sequence enabled, i.e. raised cosine attenuation followed by DC ramp down	R/W
26	SDAC_CTRL_INTI[26]	0	Normal operation	R/W
		1	Invert left and right channel output data polarity	R/W
28 - 27	SDAC_CTRL_INTI[28:27]	00	set_sd_value; 3200 fs samples	R/W
		01	set_sd_value; 4800 fs samples	R/W
		10	set_sd_value; 9600 fs samples	R/W
		11	set_sd_value; 19200 fs samples	R/W
29	SDAC_CTRL_INTI[29]	0	Silence detection disabled	R/W
		1	Silence detection enabled	R/W
31 -30			Reserved	

33.2.10 SDAC_CTRL_INTO

The interpolation filter has a control output bus, sdac_ctrl_into[11:0] comprising different indication bits.

Table 314 SDAC_CTRL_INTO

BIT	VARIABLE	DESCRIPTION	RESET	R/W
11 - 0	SDAC_CTRL_INTO[11:0]	The interpolation filter - control output bus	-	-
31 - 12		Reserved		

SDAC_CTRL_INTO[11] Dynamic

Pulse with frequency 1fs, independent of speed mode settings.

This signal is HIGH for one system clock period

SDAC_CTRL_INTO[10:4] Dynamic

Program counter that runs from 0 to 127 on the system clock

SDAC_CTRL_INTO[3]

0: No silence detected left channel

1: Silence detected left channel

SDAC_CTRL_INTO[2]

0: No silence detected right channel

1: Silence detected right channel

SDAC_CTRL_INTO[1]

0: No power down or power down in progress

1: Left and right in power down

SDAC_CTRL_INTO[0]

0: No mute or de-mute in progress

1: Left and right muted

33.2.11 SDAC_SETTINGS

An overview of the SDAC settings register is depicted in Table 315.

Table 315 SDAC_SETTINGS

BIT	VARIABLE	DESCRIPTION	RESET	R/W
7 - 0		Reserved	0x0	R/W
8	SDAC_PON_INP_RIGHT_DYN	Right channel dynamic power down input	0x0	R/W
9	SDAC_PON_INP_LEFT_DYN	Left channel dynamic power down input	0x0	R/W
12 - 10	SDAC_CTRL_DAC	Left and right channel DAC control input bus	0x0	R/W
13	HP_PON_C	Head Phone Power on Common	0x0	R/W
14	HP_PON_L	Head Phone Power on Left	0x0	R/W
15	HP_PON_R	Head Phone Power on Right	0x0	R/W
17 - 16	HP_SET_LIMITER_C	Head Phone Set limiter Centre	0x0	R/W
19 - 18	HP_SET_LIMITER_L	Head Phone Set Limiter Left	0x0	R/W
21 - 20	HP_SET_LIMITER_R	Head Phone Set Limiter Right	0x0	R/W
31 - 22		Reserved	0x0	R/W

The definition of the C18DA16b44k control bus is summarized in Table 316.

Table 316 SDAC_CTRL_DAC

BIT	VARIABLE	VALUE	DESCRIPTION	RESET	R/W
10	SDAC_CTRL_DAC[0]	0	Uni-directional DWA algorithm applied (Left channel)	-	R/W
		1	Bi-directional DWA algorithm applied (Left channel)	-	
11	SDAC_CTRL_DAC[0]	0	Uni-directional DWA algorithm applied (Right channel)	-	R/W
		1	Bi-directional DWA algorithm applied (Right channel)	-	
12	SDAC_CTRL_DAC[2]	0	The sel_two pin of the C18DA16b44k is 0, is used when the parallel input data is offset binary	-	R/W
		1	The sel_two pin of the C18DA16b44k is 1, is used when the parallel input data is 2 s complement (must be used in any case!!!)	-	R/W

33.2.12 SADC_CTRL_SDC

The left and right single-to-differential converters can be powered down separately.

Table 317 SADC_CTRL_SDC

BIT	VARIABLE	DESCRIPTION	RESET	R/W
0	SADC_CTRL_SDC_R	Right single-to-differential converter powerdown	0x0	R/W
1	SADC_CTRL_SDC_L	Left single-to-differential converter powerdown	0x0	R/W
2	SADC_CTRL_SDC_MIC	MIC single-to-differential converter powerdown	0x0	R/W
6 - 3	SADC_CTRL_PGA_R	PGA Control Right	0x0	R/W
7	SADC_CTRL_PGA_PD_R	PGA Power down Right	0x0	R/W
11 - 8	SADC_CTRL_PGA_L	PGA Control Left	0x0	R/W
12	SADC_CTRL_PGA_PD_L	PGA Power down Left	0x0	R/W
16 - 13	SADC_CTRL_PGA_MIC	PGA Control MIC	0x0	R/W
17	SADC_CTRL_PGA_MIC_PD	PGA Power down MIC	0x0	R/W
18	LNA_PDNLNA	Power down LNA	0x0	R/W
31 - 19		Reserved	0x0	R/W

SADC_CTRL_SDC_R single-to-differential converter

0: Normal operation

1: Power down for channel right C18SD

SADC_CTRL_SDC_L single-to-differential converter

0: Normal operation

1: Power down for channel left C18SD

SADC_CTRL_SDC_MIC single-to-differential converter

0: Normal operation

1: Power down for channel MIC

SADC_CTRL_PGA_R/SADC_CTRL_PGA_L/SADC_CTRL_PGA_MIC, see Table 318

Table 318 Gain Settings for PGA Control right/PGA Control left/PGA Control MIC

CTRL3	CTRL2	CTRL1	CTRL0	GAIN
0	0	0	0	0 dB
0	0	0	1	3 dB
0	0	1	1	6 dB
0	1	0	0	9 dB
0	1	0	1	12 dB
0	1	1	0	15 dB
0	1	1	1	21 dB
1	X	X	X	24 dB

SADC_CTRL_PGA_PD_R:

0: Normal operation

1: Power down for channel right

SADC_CTRL_PGA_PD_L:

0: Normal operation

1: Power down for channel left

SADC_CTRL_PGA_PD_R:

0: Normal operation

1: Power down for channel MIC

33.2.13 SADC_CTRL_ADC

Analog-to-Digital converter control input bus, left and right channel

Table 319 SADC_CTRL_ADC

BIT	VARIABLE	VALUE	DESCRIPTION	RESET	R/W
0	SADC_CTRL_ADC_R[0]	0	Analog output signal C18SD MIC to C18AD16b44k_R, input2	0	R/W
		1	Analog output signal C18SD right to C18AD16b44k_R, input1	-	
1	SADC_CTRL_ADC_R[1]	0	No dither applied to the C18AD16b44k_R	0	R/W
		1	Dither applied to the C18AD16b44k_R	-	
2	SADC_CTRL_ADC_R[2]	0	Audio mode of the C18AD16b44k_R	0	R/W
		1	Radio mode of the C18AD16b44k_R	-	R/W
3	SADC_CTRL_ADC_R[3]	0	Normal operation of the C18AD16b44k_R	0	R/W
		1	Power down of the C18AD16b44k_R	-	R/W
4	SADC_CTRL_ADC_L[0]	0	Analog output signal C18SD MIC to C18AD16b44k_L, input2	0	R/W
		1	Analog output signal C18SD left to C18AD16b44k_L, input1	-	R/W
5	SADC_CTRL_ADC_L[1]	0	No dither applied to the C18AD16b44k_L	0	R/W
		1	Dither applied to the C18AD16b44k_L	-	R/W
6	SADC_CTRL_ADC_L[2]	0	Audio mode of the C18AD16b44k_L	0	R/W
		1	Radio mode of the C18AD16b44k_L	-	R/W
7	SADC_CTRL_ADC_L[3]	0	Normal operation of the C18AD16b44k_L	0	R/W
		1	Power down of the C18AD16b44k_L	-	R/W

33.2.14 SADC_CTRL_DECI

See Table 320, for the decimator control registers.

Table 320 SADC_CTRL_DECI

BIT	VARIABLE	VALUE	DESCRIPTION	R/W
7 - 0	SADC_CTRL_DECI[7:0]	0111.1111	+24.0 dB gain right channel	R/W
		:	:	R/W
		0011.0001	+24.0 dB gain right channel	R/W
		0011.0000	+24.0 dB gain right channel	R/W
		0010.1111	+23.5 dB gain right channel	R/W
		:	:	R/W
		0000.0010	+1.0 dB gain right channel	R/W
		0000.0001	+0.5 dB gain right channel	R/W
		0000.0000	0.0 dB gain right channel	R/W
		1111.1111	-0.5 dB gain right channel	R/W
		1111.1110	-1.0 dB gain right channel	R/W
		:	:	R/W
		1000.0010	-63.0 dB gain right channel	R/W
		1000.0001	-63.5 dB gain right channel	R/W
		1000.0000	-∞ dB gain right channel (mute)	R/W
15 - 8	SADC_CTRL_DECI[15:8]	0111.1111	+24.0 dB gain left channel	R/W
		:	:	R/W
		0011.0001	+24.0 dB gain left channel	R/W
		0011.0000	+24.0 dB gain left channel	R/W
		0010.1111	+23.5 dB gain left channel	R/W
		:	:	R/W
		0000.0010	+1.0 dB gain left channel	R/W
		0000.0001	+0.5 dB gain left channel	R/W
		0000.0000	0.0 dB gain left channel	R/W
		1111.1111	-0.5 dB gain left channel	R/W
		1111.1110	-1.0 dB gain left channel	R/W
		:	:	R/W
		1000.0010	-63.0 dB gain left channel	R/W
		1000.0001	-63.5 dB gain left channel	R/W
		1000.0000	-∞ dB gain left channel (mute)	R/W
16	SADC_CTRL_DECI[16]	0	Normal operation	R/W
		1	RAM/ROM tri-state	R/W
17	SADC_CTRL_DECI[17]	0	Normal operation	R/W
		1	Polarity inversion left and right channels R/W	R/W
18	SADC_CTRL_DECI[18]	0	Normal operation	R/W
		1	Mute left and right channels	R/W
19	SADC_CTRL_DECI[19]	0	Disable output DC blocking filter	R/W
		1	Enable output DC blocking filter	R/W
20	SADC_CTRL_DECI[20]	0	Disable input DC blocking filter	R/W
		1	Enable input DC blocking filter	R/W

BIT	VARIABLE	VALUE	DESCRIPTION	R/W
21	SADC_CTRL_DECI[21]	0	Disable dB linear after reset	R/W
		1	Enable dB linear after reset	R/W
22	SADC_CTRL_DECI[22]	0	Disable timer after reset	R/W
		1	Enable timer after reset	R/W
23	AGC_EN	0	Disable AGC	R/W
		1	Enable AGC	R/W
25 - 24	AGC_LEVEL	00	AGC Level 0 - AGC Target Level = -5.5 dBFS	R/W
		01	AGC Level 1 - AGC Target Level = -8.0 dBFS	R/W
		10	AGC Level 2 - AGC Target Level = -11.5 dBFS	R/W
		11	AGC Level 3 - AGC Target Level = -14.0 dBFS	R/W
28 - 26	AGC_TIM	-	See Table 321	R/W
31 - 29			Reserved	

Table 321 AGC Time constant settings

AGC_TIME2	AGC_TIME1	AGC_TIME0	AGC SETTING			
			44.1KHZ SAMPLING		8KHZ SAMPLING	
			Attack time(ms)	Decay time(ms)	Attack time(ms)	Decay time(ms)
0	0	0	11	100	61	551
0	0	1	16	100	88.2	551
0	1	0	11	200	61	1102
0	1	1	16	200	88.2	1102
1	0	0	21	200	116	1102
1	0	1	11	400	61	2205
1	1	0	16	400	88.2	2205
1	1	1	21	400	116	2205

33.2.15 SADC_CTRL_DECO

Decimator control output bus.

Table 322 SADC_CTRL_DECO

BIT	VARIABLE	VALUE	DESCRIPTION	R/W
0	SADC_CTRL_DECO[0]	0	No mute or mute/de-mute in progress	R
		1	Left and right channels muted	R
1	SADC_CTRL_DECO[1]	0	No ctrl_deci[1]	R
		1	Overflow in left or right channel	R
2	SADC_AGC_STATUS	0	AGC Status	R
		1	AGC Status	R
31 - 29			Reserved	

33.2.16 E7B_INTERRUPT_REQ

Irq3 of the EPICS7b interrupt controller is connected to this register. When setting this register, interrupt Irq3 of the EPICS7B interrupt controller will be set. E7B_INTERRUPT REQ register is automatically cleared.

Table 323 E7B_INTERRUPT_REQ

BIT	VARIABLE	DESCRIPTION	RESET	R/W
0	E7B_INTERRUPT REQ	EPICS7B interrupt request	0x0	R/W
31 - 1		Reserved		

E7B_INTERRUPT_REQ

0: No request to E7B

1: Interrupt request to E7B

33.2.17 E7B_AHB_SETTINGS

The settings for the EPICS7B AHB interface must be done with E7B_AHB_SETTINGS register.

Table 324 E7B_AHB_SETTINGS

BIT	VARIABLE	DESCRIPTION	RESET	R/W
0	x_memory_signed	Reading X-memory data signed	0x0	R/W
2 - 1	transfer format	X-memory write/read transfer	0x0	R/W
9 - 3	timeout_counter	AHB respond error after reaching of value timeout counter	0x0	R/W
31 - 10		Reserved		

x_memory_signed Select signed/unsigned when reading X-memory

0: unsigned - data bit 24 till 31 always zero

1: signed - data 24 till 31 depends on sign bit 23 of X-memory

transfer format Select data transfer

00: 32 bits <-> 24 bits with loss of data (R/W)

01: 32 bits -> 24 bits without loss of data (W)

10: 32 bits <-> 2 x 16 bits (R/W)

11: Reserved

33.2.18 N-SOF COUNTER

Table 325 N_SOF_COUNTER

BIT	VARIABLE	DESCRIPTION	RESET	R/W
7 - 0	N_SOF_COUNTER	Divider value of the SOF signal	0x0	R/W
31 - 8		Reserved		

N_SOF_COUNTER Divider value

0: Output N_SOF_COUNTER is zero

1 255: OUTPUT N_SOF_COUNTER = SOF signal divided by programmed value.

34 EPICS7B DSP SUBSYSTEM

34.1 Introduction

EPICS7B is a programmable Digital Signal Processor which can be used for digital audio, telecom and speech processing. An EPICS7B instance can be integrated as an embedded DSP core with application-specific sub-circuits and I/O devices, like A/D and D/A convertors, on a single chip.

This chapter shall deal with the explanation of the EPICS7B DSP Sub-System architecture. For details regarding the AHB interface of EPICS7B, refer to chapter on EPICS7B AHB interface.

34.2 EPICS7B DSS architecture

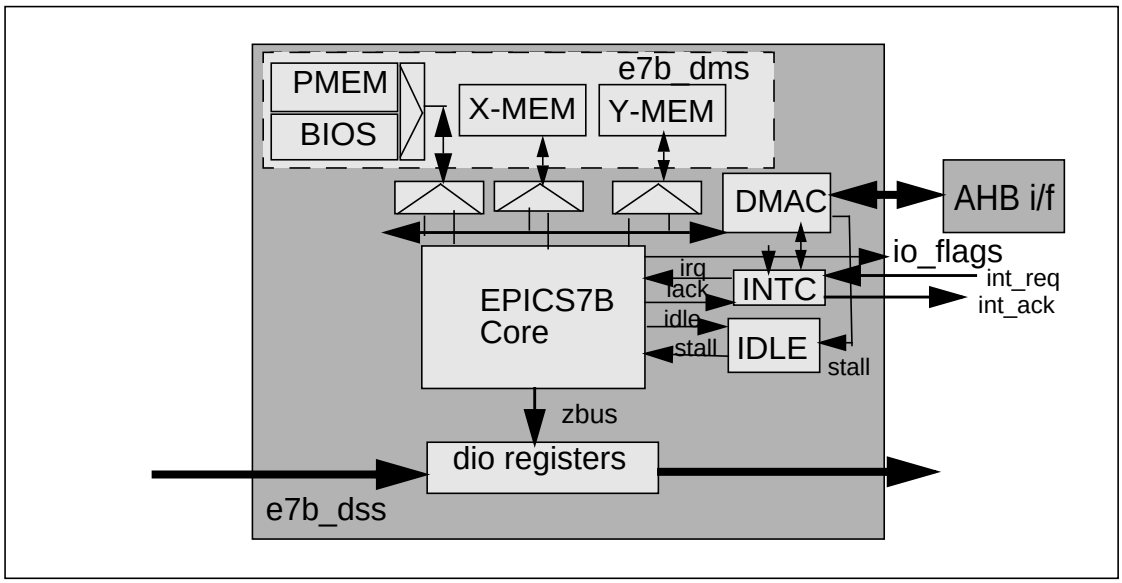


Fig. 56 Block diagram of EPICS7B DSS architecture

The EPICS7B sub-system consists of various sub-blocks that together execute the software application running on it.

At the centre of the processing logic is the EPICS7B core, that contains the various arithmetic and logic calculation units, the memory address and data calculation units and corresponding memory interfaces, program counter and interrupt request handling units, status registers, etc. For details regarding the functionality and programming of EPICS7B core, refer to the EPICS7B Programmer's Guide[1].

The DMA controller is meant for external access to all memories (P, X and Y) and to internal and external EPICS7B DIO and Control registers. This access can be done by the ARM system. There is only one DMA channel available per EPICS7B.

The DMA access registers can be accessed only via DMA. These registers control the reset, idle, bios selection and feeding of external instruction(s) to core via DMA.The DMA access registers used inside each EPICS7B DSS instance and their default values are depicted in Table 326.

Via the IDLE module, the clock going into EPICS7B core can be shut-off, thereby stalling the core. This can be possible through the IDLE software instruction, via DMA or by activating an external stall signal from the EPICS7B DMA access register 0x3FFFF(3).

The EPICS7B memories are embedded inside the Subsystem architecture. The DSP Memory System block contains the P, X, Y and BIOS memories, as well as the required interfaces towards and from them. The memories can be accessed by the EPICS7B core or directly by the system via DMA access.

Table 326 P, X, Y and BIOS memories for EPICS7B

NAME	DESCRIPTION
P-ROM (14k of 32 bits)	Program ROM for EPICS7B
P-RAM (2k of 32 bits)	Program RAM for
X-RAM (10k of 24 bits)	Data RAM for EPICS7B
Y-ROM (9k of 12 bits)	Coefficient ROM for EPICS7B
Y-RAM (1k of 12 bits)	Coefficient RAM for EPICS7B
BIOS (256 of 32 bits)	BIOS ROM for EPICS7B

The EPICS7B DSS architecture contains Control registers that can be accessed by the system embedding the DSP via DMA as well as by the DSP core itself. These registers serve to program the DSS configuration with respect to memory-map spacing, interrupt behaviour and also to provide information to the external system regarding the status of the DSP. The Control registers are X-memory mapped, occupying address space from 0xFFC0 till 0xFFFF. A few of the control registers are existing inside the Core itself, namely the Program Counter (0xFFFF) Status Register 1 (0xFFFE), Status Register 2 (0xFFFD) and the Interrupt Service Routine return address register (0xFFFC).

The Program Counter, Status Register 1 and Status Register 2 are read only registers both by the Core and DMA.

The ISR stack register can be read and written only by the Core, and not via the DMA channel, which can only read it. Control register 0xFFF6 depicting the interrupt status is also read only for both Core and DMA. All other control registers are both read and write accessible by the Core as well as via DMA channel.

The EPICS7B DSS architecture also contains interface to Data Input and Output registers through which data exchanges take place between the EPICS7B Subsystem and various audio interfaces like IIS, A/D and D/A convertors and SPDIFin.

34.3 Interrupt controller

The interrupt controller has the function to connect multiple sources to the single IRQ line of the DSP core. It takes care of the source IRQ polarity, level/edge sensitivity and the separate enabling/disabling of the individual IRQ sources. The control register 0x0FFF3 defines the set visible as input flags seen by the Core, and the output flags seen by the system integrating the EPICS7B.

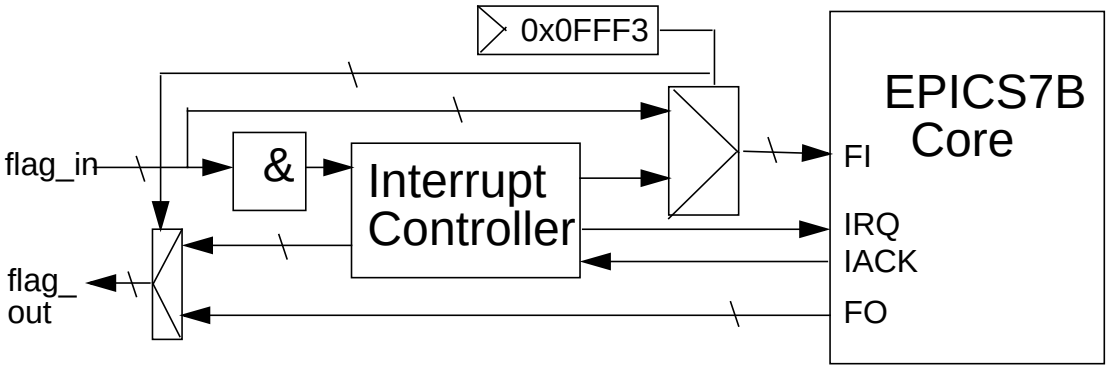


Fig. 57 Block diagram of EPICS7B flag interface

When the jump to the ISR is executed the instruction in the fetching phase (level 0) of the pipeline will be flushed and replaced by the "goint" (0xFC060000) instruction. The program counter value belonging to the flushed instruction will be saved in a specific stack register. This stack register is X-memory mapped to address 0xFFFC.

Upon reaching the ISR it should be determined in software what caused the interrupt. In case multiple interrupt sources are connected to the external interrupt controller some prioritization might be necessary. Therefore interrupt status information can be passed through from the INTC block to the DSP core via the user-FLAG inputs. The ISR needs an interrupt handler algorithm that sorts out the priority when multiple interrupts occur at the same moment. Special attention should also be paid to the acknowledgement towards the individual interrupt sources. The user-FLAG outputs (when put in stretch-mode) of the DSP core can be used to pass this acknowledge information to the sources. Prioritization and acknowledge handling should all be covered in software by the ISR.

The Epics7B only supports one level of interrupts, meaning that no nesting is possible. Although the DSS can support multiple interrupt sources the DSP core will ignore other sources while it is executing the ISR. After the RTI-instruction is executed and the DSP returned to normal operation new (or stalled) interrupts can be taken into account.

The functioning of the interrupt control block in the DSP system is configurable through various control registers (see section on control registers for default values and explanation). The following list summarizes which control registers can be connected to the INTC block:

1. int_mask_in: masks (‘1’-value) the corresponding interrupt channel or enables (‘0’-value) the channel.

2. int_type_sel_in: Selects whether the corresponding channel will detect edge-sensitive interrupts (‘1’-value) or level-sensitive interrupts (‘0’-value).

3. int_cond_sel_in: Selects depending on the interrupt-type whether the interrupt is active at a low-level respectively falling-edge (‘0’-value), or at a high-level respectively rising-edge (‘1’-value).

4. int_swclr_reg_in: (optional, to disable connect to ’0’) This register can alternatively be used to give an acknowledge signal. This option could be useful when all user-FLAGS of the DSP core are unavailable for the INTC block. On implementation of this register one should be advised to make this write-only register self-resetting in order to avoid to do two register moves (one to set and one to clear one-bit). The INTC hardware will only react on a rising edge of a bit from this register. NOTE: either this register or the "int_ack_vec_in" input signal should be used to pass acknowledge signals from the DSP core to the INTC block. The input that is not used should be connected to a fixed value.

5. int_test_in: This register contains two more bits as the number of interrupt channels. The MSbit is used to put the INTC block in test mode. In test-mode (’1’-value) the INTC-block reads its interrupt values from the least significant bits of this register. In normal operation (’0’-value) the INTC block selects the interrupt-input port. The "MSbit-1" is used to disable the IRQ output signal of the INTC block. The masked interrupt signal (which normally goes to the FLAG input of the EPICS7B core) is not disable by the signal.

Some special attention should be paid to the following points:

§ The interrupt controller is connected to the DSP clock and is thus operating in the DSP clock domain. The interrupt peripherals may all be operating in different clock domains.

§ In order to detect an interrupt the peripheral should, independent whether the interrupt channel is configured as level- or edge-sensitive interrupt, assert the interrupt pulse for at least one DSP clock cycle.

§ For correct handling of level-sensitive interrupts, the peripheral should maintain the interrupt asserted until it receives an acknowledge. When it received the acknowledge, it should clear as fast as possible the interrupt to free the interrupt handling for new interrupts.

§ A level-sensitive interrupt source can also be detected with the edge-sensitive detection logic.

§ An edge-sensitive interrupt source can launch a second interrupt to the controller as soon as it receives an acknowledge of the first interrupt request. This second interrupt will be detected and serviced once the current interrupt has been serviced and new interrupts with higher priority have been handled.

§ It is advised to the software designer to acknowledge an interrupt as soon as possible in the interrupt service routine. Usually the acknowledge will immediately be executed after the priority handling of the incoming interrupts. This will give the interrupt peripheral the maximum of time to clear its interrupt (especially true for the level-sensitive sources).

§ The acknowledge signal lasts minimally 3 DSP clock cycles in the ’worst case’.(RTI, INSTRUCTION1, INSTRUCTION2).

34.4 DSS Internal Control Registers

Table 327 DSS Control register map overview

OFFSET	REGISTER NAME	R/W/ RW	RESET VALUE
0x3FFFF	DSP Control Register	RW	0x1
0x3FFFE	DSP Epics Instruction Register	RW	0xFC000000
0x0FFFF	DSP Program Counter Register	R	-
0x0FFFE	DSP Status Register 1	R	-
0x0FFFD	DSP Status Register 2	R	-
0x0FFFC	DSP Interrupt Stack Register	RW	-
0x0FFFB	DSP Configuration Register 1	RW	0x03C002
0x0FFFA	DSP Configuration Register 2	RW	0x000FFD
0x0FFF9	DSP INTC Polarity (Pos/Neg)	RW	0x03FFFF
0x0FFF8	DSP INTC Type (Level/Edge)	RW	0x03FFFF
0x0FFF7	DSP INTC Mask (Enable)	RW	0x03FFFF
0x0FFF6	DSP INTC Status	R	-
0x0FFF5	DSP INTC Test	RW	0x000000
0x0FFF4	Reserved	-	-
0x0FFF3	DSP IRQ_UserFlag	RW	0x000001
0x0FFF2	DSP DMAC IRQ counter value	RW	0x000000
0x0FFF1- 0x0FFE0	Reserved	-	-
0x0FFDF	Output register 0x0FFDF	RW	0x000000
0x0FFDE	Output register 0x0FFDE	RW	0x000000
0x0FFDD	Output register 0x0FFDD	RW	0x000000
0x0FFDC	Output register 0x0FFDC	RW	0x000000
0x0FFDB	Output register 0x0FFDB	RW	0x000000
0x0FFDA	Output register 0x0FFDA	RW	0x000000
0x0FFD9	Output register 0x0FFD9	RW	0x000000
0x0FFD8	Output register 0x0FFD8	RW	0x000000
0x0FFD7	Output register 0x0FFD7	RW	0x000000
0x0FFD6	Output register 0x0FFD6	RW	0x000000
0x0FFD5	Output register 0x0FFD5	RW	0x000000
0x0FFD4	Output register 0x0FFD4	RW	0x000000
0x0FFD3	Input register 0x0FFD3	RW	0x000000
0x0FFD2	Input register 0x0FFD2	RW	0x000000
0x0FFD1	Input register 0x0FFD1	RW	0x000000
0x0FFD0	Input register 0x0FFD0	RW	0x000000
0x0FFCF	Input register 0x0FFCF	RW	0x000000
0x0FFCE	Input register 0x0FFCE	RW	0x000000
0x0FFCD	Input register 0x0FFCD	RW	0x000000
0x0FFCC	Input register 0x0FFCC	RW	0x000000

Table 328 DSP Control Register (0x3FFFF)

BITS	VARIABLE	DESCRIPTION	R/W	RESET VALUE
0	DSP_bios_rom_sel	Select between bios_rom and PRAM. Only used for evaluation purposes. 0 =>PRAM 1 =>BIOS ROM	R/W	1
1	DSP_dsp_reset	Reset DSP PC, SR and internal interrupt FSM 0 => DSP is running 1 => Reset	R/W	0
2	DSP_eir_en	Enable EIR register as DSP instruction. 0 => Disabled 1=> Enabled	R/W	0
3	DSP_dsp_idle	0 => DSP is running 1 => DSP in IDLE mode, no code execution.	R/W	0
31..4	Reserved	Reserved	-	-

Table 329 DSP Epics Instruction Register (0x3FFFE)

BITS	VARIABLE	DESCRIPTION	R/W	RESET VALUE
31..0	EIR	Goto mode instruction	R/W	0xF4000000

Table 330 DSP Program Counter Register (0xFFFF)

BITS	VARIABLE	DESCRIPTION	R/W	RESET VALUE
15..0	PC-register	Program counter of DSP	R	-
23..16	Reserved	Reserved	-	-

Table 331 DSP Status Register 1 (0xFFFE)

BITS	VARIABLE	DESCRIPTION	R/W	RESET VALUE
23..0	SR_register 1	Status register 1 of DSP	R	-

Table 332 DSP Status Register 2 (0xFFFD)

BITS	VARIABLE	DESCRIPTION	R/W	RESET VALUE
4..0	SR_register 2	Status register 2 of DSP	R	-
23..5	Reserved	Reserved	-	-

Table 333 DSP Interrupt Stack Register (0xFFFC)

BITS	VARIABLE	DESCRIPTION	R/W	RESET VALUE
15..0	RTI_stack	Interrupt stack register to save PC at ISR	R/W	-
23..16	Reserved	Reserved	-	-

Table 334 DSP Configuration Register 1 (0xFFFB)

BITS	VARIABLE	DESCRIPTION	R/W	RESET VALUE
17..0	IO-config	IO-configuration bits for DSP. These bits only have a function if the corresponding bit in register 0xFFFA is set to 0. 0 => input 1 => output	R/W	0x03C002
23..18	Reserved	Reserved	-	-

Table 335 DSP Configuration Register 2 (0xFFFA)

BITS	VARIABLE	DESCRIPTION	R/W	RESET VALUE
17..0	Mode-config	Switch the IO-flags into stretch mode (1) or IO mode (0)	R/W	0x000FFD
23..18	Reserved	Reserved	-	-

Table 336 DSP INTC Polarity Register (0xFFF9)

BITS	VARIABLE	DESCRIPTION	R/W	RESET VALUE
17..0	Int_pol	Interrupt polarity. 0 => low level or falling edge 1 => high level or rising edge	R/W	0x03FFFF
23..18	Reserved	Reserved	-	-

Table 337 DSP INTC Type Register (0xFFF8)

BITS	VARIABLE	DESCRIPTION	R/W	RESET VALUE
17..0	Int_mode	Interrupt type 0 =>level sensitive 1 =>edge sensitive	R/W	0x03FFFF
23..18	Reserved	Reserved	-	-

Table 338 DSP INTC Mask Register (0xFFF7)

BITS	VARIABLE	DESCRIPTION	R/W	RESET VALUE
17..0	Int_mask	Interrupt disable 0 =>enabled 1 =>masked	R/W	0x03FFFF
23..18	Reserved	Reserved	-	-

Table 339 DSP INTC Status Register (0xFFF6)

BITS	VARIABLE	DESCRIPTION	R/W	RESET VALUE
17..0	Int_status	Non masked interrupt status information	R	-
23..18	Reserved	Reserved	-	-

Table 340 DSP INTC Test Register (0xFFF5)

BITS	VARIABLE	DESCRIPTION	R/W	RESET VALUE
0	Int_disable	Disable interrupt request to DSP core 0 => enable 1 => disable	R/W	0
1	Int_test_en	Enable interrupt test register 0 => disable 1 => enable	R/W	0
19..2	Int_test	Interrupt request input data	R/W	0x00000
23..20	Reserved	Reserved	-	-

Table 341 DSP IRQ versus User Flag Selection Register (0xFFF3)

BITS	VARIABLE	DESCRIPTION	R/W	RESET VALUE
17..0	IRQ_user_flag	Selection if an input is used either as interrupt or as user flag. 0 => user flag 1 => interrupt	R/W	0x000001
23..18	Not used	Not used	-	-

Table 342 DSP DMAC IRQ Counter Register (0xFFF2)

BITS	VARIABLE	DESCRIPTION	R/W	RESET VALUE
15..0	DMA_irq_count	Counter value to generate a DMA interrupt to the core.	R/W	0x0000
23..16	Not used	Not used	-	-

35 E7B_AHB_INTERFACE

35.1 Overview

35.1.1 FEATURES

- Read/write accesses from AHB master to EPICS7B: X-, Y-, P-memory and internal DSS registers, like DIO- and hardware-registers.
- Support memory single access and memory burst accesses.
- Supported different data transfers for X-memory like, with data loss, without data loss, 32 bits to 16 bits and 32 bits to 8 bits.
- Timeout Counter, adjustable from 16 to 64 wait states.
- Minimum latency of 3 wait states for read/write accesses (single/burst).

35.2 Architecture

35.2.1 Hardware Interface and Signal Description

The EPICS7B AHB interface is an external interface from the multi-layer AHB to the audio DSS. An AHB master device can transfer data from/to the audio DSS memories and from/to internal DSS registers. Table 362 describes the signal connection of the interface.

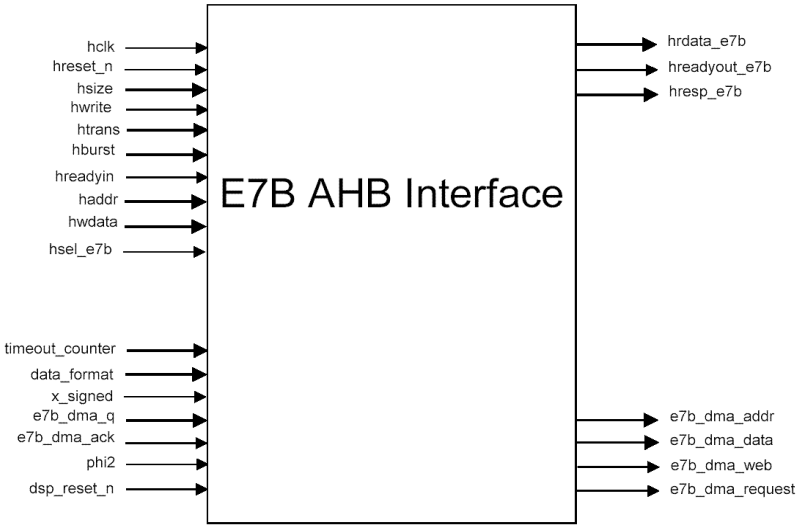


Fig. 58 E7B AHB Interface Toplevel

35.2.2 E7B AHB Interface Memory Map

The memory map of the DSP subsystem is shown in Table 343.

Table 343 Memory map of the DSP subsystem

ARM ADDRESS	DSP ADDRESS	NAME	SIZE (WORDS)	WIDTH
0x106F.FFF8 - 0x106F.FFFC	-	DMA access registers	-	32 bit
	-	not used	-	-
0x106F.FC00 - 0x106F.FC48	0xFF00 - 0xFF11	DIO registers	-	24 bit
	-	not used	-	-
0x106A.0000 - 0x106A.1FFC	0x8000 - 0x87FF	PRAM	2k	32 bit
0x1068.0000 - 0x1068.DFFC	0x0000 - 0x37FF	PROM	14k	32 bit
-	0x0000 - 0x00FF	BIOSROM	256	32 bit
0x1066.0000 - 0x1066.0FFC	0x8000 - 0x83FF	YRAM	1k	12 bit
0x1064.0000 - 0x1064.8FFC	0x0000 - 0x23FF	YROM	9k	12 bit
0x1063.FF00 - 0x1063.FFFC	0xFFC0 - 0xFFFF	DSP control registers	-	24 bit
0x1060.0000 - 0x1060.9FFC	0x0000 - 0x27FF	XRAM	10k	24 bit

35.2.3 MEMORY TRANSFERS

The Memory Access State will be described in the following paragraphs in more detail. There are 3 memory transfer schedules for the X-memory. The schedules are:

- Data transfers with loss of data (R/W)
- Data transfers without loss of data (W)
- 32 bits data transfers to 2 x 16 bits data transfers (R/W)

For Y-memory transfer schedule 1 will be used, so 2 till 3 are not applicable.

35.2.3.1 Data transfers with loss of data (R/W)

First schedule is a transfer from 32 bits to 24 bits with loss of 8 bits for X-memory, DIO registers, EPICS control registers and 20 bits for Y-memory. There is no data loss for P-memory and DMA access registers because it is 32 bits.

For X-memory reading there are two modes within this data transfer schedule, namely unsigned and signed. Selecting unsigned, x_signed is low, then the bits e7b_ahb_data(31 DOWNTO 24) are always zero. Selecting signed, x_signed is high, the value of bits 31 DOWNTO 24 depends on bit 23 of the Xmemory. If bit 23 is a one (negative value) then bit 31 DOWNTO 24 are also a one and vice versa.

Example:

Reading 0x8----- from X-memory and signed bit is set then the data at the AHB is 0xFF8----- The AHB master peripheral is LSB aligned and the EPICS7B is MSB aligned, the alignment is shown in Table 344 .

Table 344 Data alignment with loss of data

MEMORY/REGISTERS	AHB MASTER DATA ALIGNMENT	E7B DATA ALIGNMENT
X-memory	==== ==== M=== ==== ==== ==== ==== ===L	M=== ==== ==== ==== ==== ===L
Y-memory	==== ==== ==== ==== ==== M=== ==== ===L	M=== ==== ===L
P-memory	M=== ==== ==== ==== ==== ==== ==== ===L	M=== ==== ==== ==== ==== ==== ==== ===L
EPICS Control Registers	==== ==== M=== ==== ==== ==== ==== ===L	M=== ==== ==== ==== ==== ===L
DMA access registers	M=== ==== ==== ==== ==== ==== ==== ===L	M=== ==== ==== ==== ==== ==== ==== ===L

M = Most significant Bit, L = Least Significant Bit

35.2.3.2 Data transfers without loss of data (W)

A data transfer from 32 bits AHB peripheral to a 24 bits X-memory location can be done without loosing data. The data format has to be programmed as 01 and is only applicable for writing the Xmemory. The data flow is depicted in Figure 9.

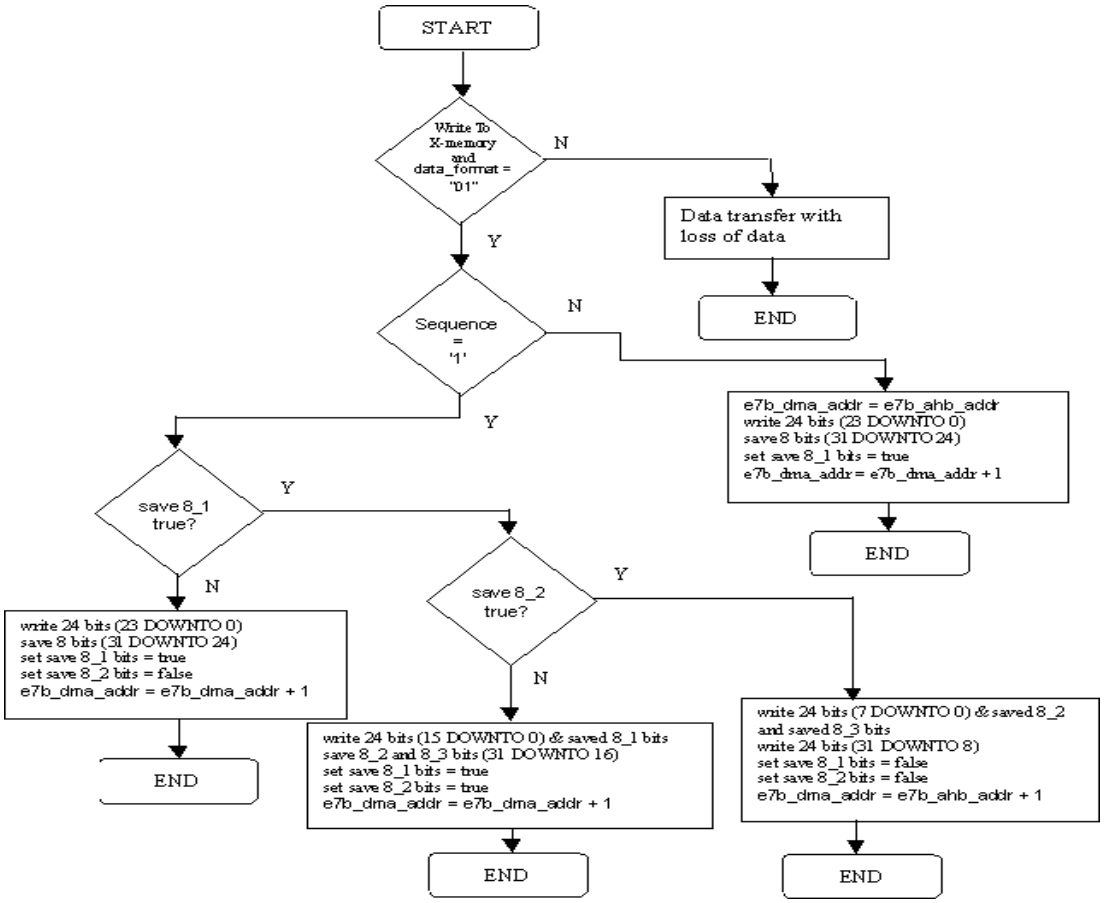


Fig. 59 Flow diagram: data transfers without loss of data

The flow start with checking if the data_format is programmed as 01 and also the data transfer must be an X-memory write, if not, a data transfer with loss will be executed.

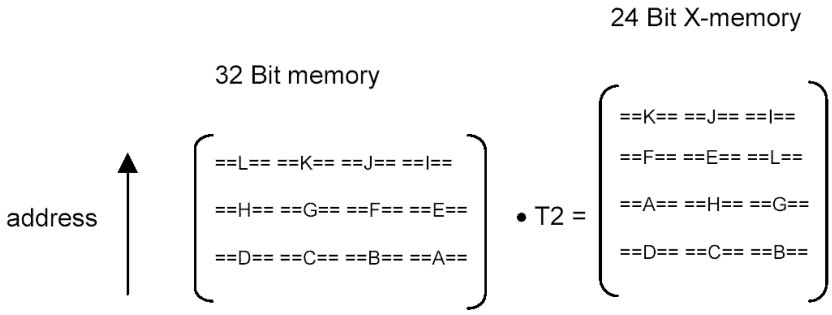
Comply with this request a 32 bits -> 24 bits without loss of data will be started. If the e7b_addr_value of the previous write transfer is equal to e7b_ahb_addr minus one then the sequence is one.

If the sequence is zero a new data transfer sequence is started and a memory access will be done to the X-memory(e7b_ahb_addr(31 DOWNTO 8). Also the 8 bits from e7b_ahb_data(7 DOWNTO 0) will be saved in a register and the save 8_1 bits is set to true. The last step is to increase the e7b_dma_addr with one.

If next write access is a sequence one, then e7b_ahb_data(31 DOWNTO 16) and save register 8_1 will be written to the X-memory. The 16 bits from e7b_ahb_data(15 DOWNTO 0) will be saved in registers save 8_2 and save 8_3, and set save 8_2 bits to true. Also in this process the e7b_dma_addr is increased with one.

The next data transfers are two memory accesses, first e7b_ahb_data(31 DOWNTO 24) and save register 8_2 and 8_3, then increase e7b_dma_addr and write for a second time(e7b_ahb_data(23 DOWNTO 0)) and increase once again the e7b_dma_addr.

The next data transfers are two memory accesses, first e7b_ahb_data(31 DOWNTO 24) and save register 8_2 and 8_3, then increase e7b_dma_addr and write for a second time(e7b_ahb_data(23 DOWNTO 0)) and increase once again the e7b_dma_addr.



For calculation of the number of wait states the following formula can be used.

Equation 3

$$\#\text{NumberOfWaitStates} = 2 + \left(5 \cdot \frac{HCLK}{PHI2}\right)$$

35.2.3.3 32 bits data transfers to 2 x 16 bits data transfers (R/W)

The transfer is done in the DSP clock domain, every 32 bits transfer, reading or writing, will split in two E7B DMA transfers. The memory accesses will be done in dma burst mode.

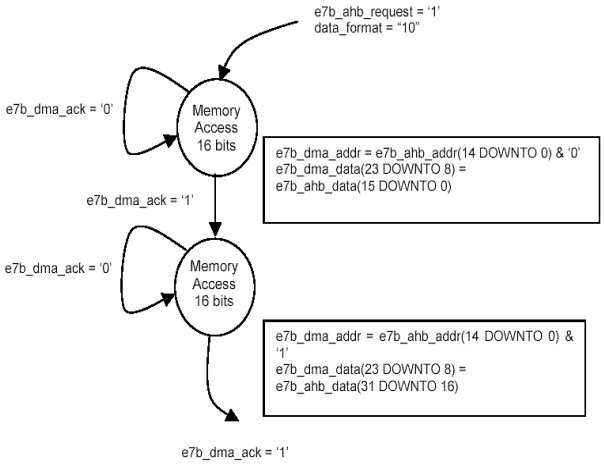


Fig. 61 2x16 bits data transfer

Data Alignment 32 <-> 2 x 16

MEMORY/REGISTER	AHB MASTER PERIPHERAL ALIGNMENT	DSP ALIGNMENT
X-memory	31=== ==== ==== ==== 15=== ==== ==== ===0 R L	23=== ==== ==== ===8 0000 0000 L 23=== ==== ==== ===8 0000 0000 R

For calculation of the number of wait states the following formula can be used.

$$\#\text{NumberOfWaitStates} = 2 + \left(5 \cdot \frac{HCLK}{PHI2}\right)$$

36 SIMPLE AUDIO INPUT

36.1 Overview

The Simple Audio Intput, or SAI, is a small APB slave to process one channel of left and right audio from an audio-input interface. (I2S, SADC, SPDIF-in, EPICS,..).

It contains a 4 sample FIFO for both left and right channel, for flexibility reasons.

The SAI has ARM flow-control using only single requests for AHb data transfers using a ARM compatible DMA controller like the SDMA.

36.2 Functional Description

36.2.1 INTERFACE

- APB interface for the CPU sending control and receive audio samples.
- Small in size; Uses only minimum amount of buffers while guaranteeing non-interrupted streaming in combiation with the SDMA controller.
- Supports single 16 bit transfers from the left or right FIFO.
- Supports single 24 bit transfers from the left or right FIFO.
- Supports 32-bit interleaved transfers, with the bottom 16 bits being left, and the top 16 bits being right audio.
- Supports double 16-bit samples, combined in a 32-bit word (2 left or 2 right samples) to reduce busload
- Contains maskable interrupts for audio status.
- Uses ARM flow control, single request only.

36.3 SDMA flow control

The best method for transporting audio-samples from the audio-sub_units to memory is via SDMA flow-control. The SDMA is a small AHB DMA controller, which respond to ARM flow-control signals.

If the SAI has one sample or more availble, it will output a request to the SDMA controler. The SDMA perform a single read, and acklowledges this with the assertion of ‘CLR’. Then the ‘SREQ’ is deasserted.
If the SAI has more samles availbable, req will be made active as soon as clr is de-asserted.
For timing information, please refer to the ARM PL080 specification.

Special about SAI DMA flow control: The SAI can receive the ‘DMA_CLR’ signal from a different asynchronous clock domain

The DMA request is based on the HALF_FIFO flag, which indicates the SAI has 2 or more samples in its FIFO.

Recommendations for the SDMA settings:

To reduce system requirements and interrupt handling for the CPU the following options can be used in the SDMA:

- Use comapnion-tables OR use ‘cicular buffer’ setting.
  Companion tables would require the double amount of channels, where at the end of the transfer inside a channel, a ‘companion’ table would automatically be enabled. This would give the CPU plenty of time to respond, and to reprogram the old DMA channel.
  By using the ‘circular buffer’ option, the same channel would be used over and over again, where at the end of the channel, it will return to its start address and continue.

36.4 Registers

Table 345SAI register map

ADDRESS SPACE	OFFSET	REGISTER NAME	R/W ACCESS	RESET
<system specific>	0x00	AUDIO_IN_LEFT_16BIT	R	0x0
	0x04	AUDIO_IN_RIGHT_16BIT	R	0x0
	0x08	AUDIO_IN_LEFT_24_BIT	R	0x0
	0x0C	AUDIO_IN_RIGHT_24_BIT	R	0x0
	0x10	IRQ_STATUS	R/W	0x0
	0x14	IRQ_MASK	R/W	0x7f
	0x18	<reserved>	-	0x0
	0x20 - 0x3C	AUDIO_IN_LEFT_32_BIT	R	0x0
	0x40 - 0x5C	AUDIO_IN_RIGHT_32_BIT	R	0x0
	0x60 - 0x7C	AUDIO_IN_INTERLEAVED	R	0x0

AUDIO_IN_LEFT_16BIT

Reading from this register reads the 16-bit representation of the left audio from the left FIFO. 8 LSB audio bits (zeros) are discarded from the internal 24 bits value.
When reading from this register, Bits [31 - 16] are all ‘0’’

AUDIO_IN_RIGHT_16_BIT

Reading from this register reads the 16-bit representation of the right audio from the right FIFO. 8 LSB audio bits (zeros) are discarded from the internal 24 bits value.
When reading from this register, Bits [31 - 16] are all ‘0’.

AUDIO_IN_LEFT_24_BIT

Reading fromthis register reads the pure 24-bit representation of the left audio from the left FIFO.
When reading from this register, bits [31-24] are all ‘0’.

AUIDIO_IN_RIGHT_24_BIT

Reading fromthis register reads the pure 24-bit representation of the rightaudio from the left FIFO.
When reading from this register, bits [31-24] are all ‘0’.

IRQ_STATUS

This register contains status information about the FIFO’s

Table 346IRQ_STATUS_CLEAR register

BIT	LABEL	RESET	R/W
0	FIFO RIGHT underrun	0	R/W
2	FIFO LEFT underrun	0	R/W
2	FIFO RIGHT overrun	0	R/W
3	FIFO LEFT overrun	0	R/W
4	FIFO LEFT full	0	R/W
5	FIFO LEFT half_full	0	R/W
6	FIFO LEFT NOT empty	0	R/W
7	FIFO RIGHT full	0	R/W
8	FIFO RIGHT half_empty	0	R/W
9	FIFO RIGHT NOT empty	0	R/W

Interrupt clearing: Writing to the Status register(any value) clears the underrun and overrun bits.
The ‘not empty’, ‘half_full’ and ‘full’ bits are direct response signals from the FIFO. Reading from the FIFO will automacally ‘clear’ a situation.

IRQ_MASK

Table 347IRQ_MASK register

BIT	LABEL	RESET	R/W
0	IRQ from status bit 0 is masked	1	R/W
1	IRQ from status bit 1 is masked	1	R/W
2	IRQ from status bit 2 is masked	1	R/W
3	IRQ from status bit 3 is masked	1	R/W
4	IRQ from status bit 4 is masked	1	R/W
5	IRQ from status bit 5 is masked	1	R/W
6	IRQ from status bit 6 is masked	1	R/W
7	IRQ from status bit 7 is masked	1	R/W
8	IRQ from status bit 8 is masked	1	R/W
9	IRQ from status bit 9 is masked	1	R/W

AUDIO_IN_LEFT_32_BIT

Reading from this register reads 2 samples from the left FIFO. The LSB 16 bits represents the first sample.

AUDIO_IN_RIGHT_32_BIT

Reading from this register reads 2 samples from the right FIFO. The LSB 16 bits represents the first sample.

AUDIO_IN_INTERLEAVED
reading from this register reads both left and right audio as interleaved.
Left = Bits [15-0]
Right = Bits [31-16]

36.5 FIFO and IRQ behaviour

The FIFO’s are shifted by one position as soon as the NEXT_SAMPLE signal arrives.
If the FIFO’s are empty, and data is read by a busmaster, an underrun is generated, which can be read and cleared inside

the status register.
If the FIFO is full, and a ‘next_sample’ is received intrenally, an overrun is generated which can be read and cleared inside the status register.

When a single sample is read, the FIFO is shifted by one position
When two sampels are read, in the case of reading from register AUDIO_IN_LEFT_32_BIT and AUDIO_IN_RIGHT_32_BIT, the FIFO’s are shifted by 2 positions.
When reading from the ‘AUDIO_IN_INTERLEAVED’ register, both left and right FIFO will shift by one position.

The current FIFO is 4 samples long, each sample being 24 bits wide.

37 SIMPLE AUDIO OUTPUT

37.1 Overview

The Simple Audio Output, or SAO, is a small APB slave to process one channel of left and right audio to a audio-output interface. (I2S, SDAC, SPDIF, EPICS,..).

It contains a 4 sample FIFO for both left and right channel, for flexibility reasons.

The SAO makes use of ARM flow control for transfering data directly from memory via the SDMA controller

37.2 Functional Description

37.2.1 INTERFACE

- APB interface for sending control and send audio samples.
- Small in size; Uses only minimum amount of buffers while guaranteeing non-interrupted streaming in combiation with the SDMA controller.
- Supports single 16 bit transfers to the left or right FIFO.
- Supports single 24 bit transfers to the left or right FIFO.
- Supports 32-bit interleaved transfers, with the bottom 16 bits being left, and the top 16 bits being right audio.
- Supports double 16-bit samples, combined in a 32-bit word (2 left or 2 right samples) to reduce busload
- Contains maskable interrupts for audio status.

37.3 SDMA flow control

The best method for transporting audio-samples to the audio-sub_units is via SDMA flow-control. The SDMA is a small AHB DMA controller, which respond to ARM DMA flow control.

See the ARM PL080 spec for timing information.

Special about the SAO:
The ‘DMA_CLR’ signal may be origination from a different asynchronous clock domain
The REQ will only be de-asserted when the data has been received. Not sooner, even is ‘DMA_CLR’ is active. This means there is no requirement for the SAO to be behind a write-buffer-disabled slave.

Recommendations for the SDMA settings.

To reduce system requirements and interrupt handling for the CPU the following options can be used in the SDMA:

- Use companion-tables OR use ‘cicular buffer’ setting.
  Copy tables would require the double amount of tables, where at the end of the transfer inside a table, a ‘companion’ table would automatically be enabled. This would give the CPU plenty of time to respond, and to reprogram the old table.
  By using the ‘circular buffer’ option, the same table would be used over and over again, where at the end of the table, it will return to its start address and continue. The CPU can be interrupted in the middle of the table, and wt the end of the table. Using ‘circular buffer’ is preferred over companion tables, since this method requires just one table to setup

37.4 Registers

Table 348SAO register map

ADDRESS SPACE	OFFSET	REGISTER NAME	R/W ACCESS	RESET
<system specific>	0x00	AUDIO_OUT_LEFT_16BIT	W	0x0
	0x04	AUDIO_OUT_RIGHT_16BIT	W	0x0
	0x08	AUDIO_OUT_LEFT_24_BIT	W	0x0
	0x0C	AUDIO_OUT_RIGHT_24_BIT	W	0x0
	0x10	IRQ_STATUS	R/W	0x0
	0x14	IRQ_MASK	R/W	0x7f
	0x18	<reserved>	R	0x0
	0x1C	<reserved>	R	0x0
	0x20 - 0x3C	AUDIO_OUT_LEFT_32_BIT	W	0x0
	0x40 - 0x5C	AUDIO_OUT_RIGHT_32_BIT	W	0x0
	0x60 - 0x7C	AUDIO_OUT_INTERLEAVED	W	0x0

AUDIO_OUT_LEFT_16BIT

Writing to this register writes the 16-bit representation of the left audio into the left FIFO. 8 LSB audio bits (zeros) are added to the sample to make it 24 bits wide internally
When writing to this register, Bits [31 - 16] are ignored.
The FIFO_LEFT_counter will shift by one sample after writing.

AUDIO_OUT_RIGHT_16_BIT

Writing to this register writes the 16-bit representation of the right audio into the right FIFO. 8 LSB audio bits are added to the sample to make it 24 bits wide.
When writing to this register, Bits [31 - 16] are ignored
The FIFO_RIGHT_counter will shift by one sample after writing.

AUDIO_OUT_LEFT_24_BIT

writing to this register writes the pure 24-bit representation of the left audio into the left FIFO.
When writing to this register, bits [31-24] are ignored.
The FIFO_LEFT_counter will shift by one sample after writing.

AUIDIO_OUT_RIGHT_24_BIT

writing to this register writes the pure 24-bit representation of the right audio into the right FIFO.
When writing to this register, bits [31-24] are ignored.
The FIFO_RIGHT_counter will shift by one sample after writing.

IRQ_STATUS

This register contains status information about the FIFO’s

Table 349IRQ_STATUS register

BIT	LABEL	RESET	R/W
0	FIFO right underrun	0	R/W
1	FIFO left underrun	0	R/W
2	FIFO right overrun	0	R/W
3	FIFO left overrun	0	R/W
4	FIFO left full	0	R/W
5	FIFO left half_empty	0	R/W
6	FIFO left empty	0	R/W
7	FIFO right full	0	R/W
8	FIFO right half_empty	0	R/W
9	FIFO right empty	0	R/W

Interrupt clearing: Writing to the Status register clears the overrun and underrun bits.
The empty, half-empty and full bits are cleared automatically when FIFO is filled again. These signals represent the current status of the FIFO.

IRQ_MASK

Table 350IRQ_MASK register

BIT	LABEL	RESET	R/W
0	IRQ from status bit 0 is masked	1	R/W
1	IRQ from status bit 1 is masked	1	R/W
2	IRQ from status bit 2 is masked	1	R/W
3	IRQ from status bit 3 is masked	1	R/W
4	IRQ from status bit 4 is masked	1	R/W
5	IRQ from status bit 5 is masked	1	R/W
6	IRQ from status bit 6 is masked	1	R/W
7	IRQ from status bit 7 is masked	1	R/W
8	IRQ from status bit 8 is masked	1	R/W
9	IRQ from status bit 9 is masked	1	R/W

AUDIO_OUT_LEFT_32_BIT
Writing to this register writes 2 samples to the left FIFO. The LSB 16 bits represents the first sample.
The FIFO_LEFT counter is increased by 2

AUDIO_OUT_RIGHT_32_BIT
Writing to this register writes 2 samples to the right FIFO. The LSB 16 bits are the first sample
Te FIFO_RIGHT counter is incremented by 2

AUDIO_OUT_INTERLEAVED
Writing to this register writes both left and right audio as interleaved.
Left = Bits [15-0]
Right = Bits [31-16]
The FIFO_LEFT counter is incremented by 1, AND the FIFO_RIGHT counter is incremented by 1.

37.5 FIFO and IRQ behaviour

The FIFO's are shifted by one position as soon as the NEXT_SAMPLE signal arrives.
If the FIFO's are full, and data is written by a busmaster, an overrun is generated, which can be read and cleared inside the status register.
If the FIFO is empty, and a 'next_sample' is requested intrenally, an underrun is generated which can be read and cleared inside the status register.

When a single sample is written, the FIFO is shifted by one position
When two samples are written, in the case of writing to register AUDIO_LEFT_32_BIT and AUDIO_RIGHT_32_BIT, the FIFO's are shifted by 2 positions.
When writing to the 'AUDIO_OUT_INTERLEAVED' register, both left and right FIFO will shift by one position.

The current FIFO is 4 samples long, each sample being 24 bits wide.

38 LNA/PGA/SDC/ADC/DEC

38.1 ADC analog front-end

The analog frond-end of the ADC consists of one stereo ADC with a selector in front of it. Using this selector one can either select the microphone input with the microphone amplifier (LNA) with a fixed 30dB gain or the line input. The microphone input as well as the line inputs have a Programmable Gain Amplifier (PGA) that allows gain control from 0dB to 24dB in steps of 3dB.

Note: the input impedence of the PGA (line in) is 12KΩ, for the LNA this is 5KΩ.

38.1.1 APPLICATIONS AND POWER DOWN MODES

The following power down modes and/or functional modes are supported:

- power down mode in which the current consumption is VERY LOW = only leakage currents
  Note: in this mode there is NO reference voltage at the line input.
- line in mode, in which the PGA can be used
- microphone mode, in which the rest of the non-used PGA's and ADC's are powered down.
  Note: In this mode the mono microphone signal can be sent to both left and right input of the decimation filter. This is done with a separate MUX in front of the decimation input. This MUX is controlled by the SEL_MIC bit in the L3/I 2 C control interface.
- mixed PGA and LNA mode = one line-in and one microphone input.

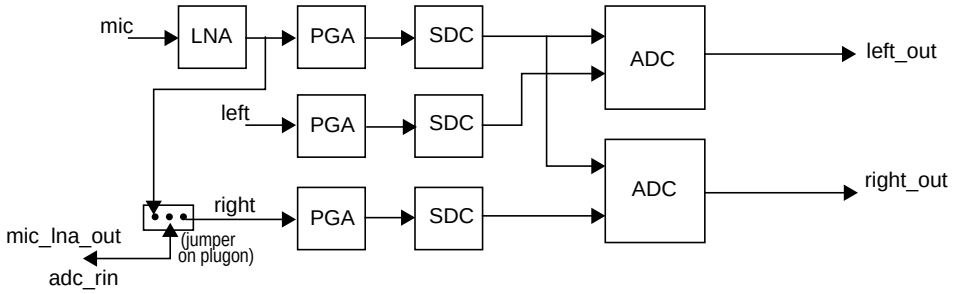


Fig. 62 analog front end

38.1.2 LNA

LNA, a Low Noise microphone Amplifier with nominal gain of 30dB. For more information see the "Shrinkability Test Report" of the Low noise amplifier (LNA_VGA_des_shrink_final_1p0.pdf).

38.1.3 PGA

The input signal is amplified with a gain set by 4 control bits *ctrl3..ctrl0*. The resulting signal will be available at V_{out} . The control bit *ctrl3* = 1 sets the gain to 24 dB independent of the other ctrl bits. *Ctrl3* = 0 allows for other (lower) gain settings. The C18PG is based on an inverting amplifier architecture. The feedback resistance exists of a resistor string. By switching between different resistors with the use of a 4-bit digital decoder the gain of the amplifier can be modified. The gain can be set in steps of 3 dB up to 24 dB. The C18PG is designed to handle a nominal $1V_{rms}$ input level. A systematic gain of -1.94 dB is added (20*log 0.8) to accommodate the $800mV_{rms}$ input level of a Single-to-Differential converter that is normally connected to the C18PG output. The *pd* signal is controlled by the digital core of the system chip.

Table 351 PGA Gain settings

CTRL3	CTRL2	CTRL1	CTRL0	GAIN
0	0	0	0	0 dB
0	0	0	1	3 dB
0	0	1	0	6 dB
0	0	1	1	9 dB
0	1	0	0	12 dB
0	1	0	1	15 dB
0	1	1	0	18 dB
0	1	1	1	21 dB
1	x	x	x	24 dB

38.1.4 APPLICATIONS WITH 2VRMS INPUT

For the line-in it is preferable to have 0dB and 6dB gain setting in order to be able to apply both 1Vrms and 2Vrms (using series resistance). For this purpose a PGA is used which has 0dB to 24dB gain with 3dB steps.

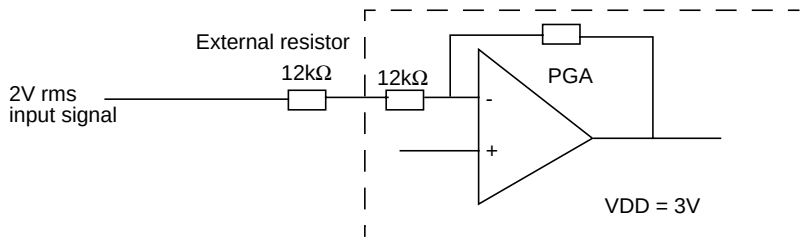


Fig. 63 Schematic of ADC front-end with PGA (line-in input)

In applications in which a 2 V (RMS) input signal is used, a 12kW resistor must be used in series with the input of the ADC. This forms a voltage divider together with the internal ADC resistor and ensures that only 1 V (RMS) maximum is input to the IC. Using this application for a 2 V (RMS) input signal, the switch must be set to 0 dB. When a 1 V (RMS) input signal is input to the ADC in the same application, the gain switch must be set to 6 dB.

An overview of the maximum input voltages allowed against the presence of an external resistor and the setting of the gain switch is given in Table 2; the power supply voltage is assumed to be 3 V.

Table 352 Application modes using input gain stage

RESISTOR (12KΩ)	INPUT GAIN SWITCH	MAXIMUM INPUT VOLTAGE
Present	0 dB	2 V (RMS)
Present	6 dB	1 V (RMS)
Absent	0 dB	1 V (RMS)
Absent	6 dB	0.5 V (RMS)

38.1.5 SDC

The C18SD is a Single-to-Differential Converter that consists of an inverting amplifier and a filter network (required filter for a switched-cap Analog-to-Digital Converter such as the C18AD16b44k). The input is dc-coupled, which means that decoupling must be done in front of this module in case the input signal has a different common mode level then the C18SD. For optional biasing conditions, the C18SD requires a sourcing bias current (into an NMOS transistor) that is preferably proportional to the analog supply voltage.

38.2 Decimation filter (ADC)

The decimation from 128fs is performed in two stages. The first stage realizes sin(x)/x characteristics with decimation factor of 16. The second stage consists of 3 half-band filters, each decimating by a factor of 2. The filter characteristics are shown in Table 353.

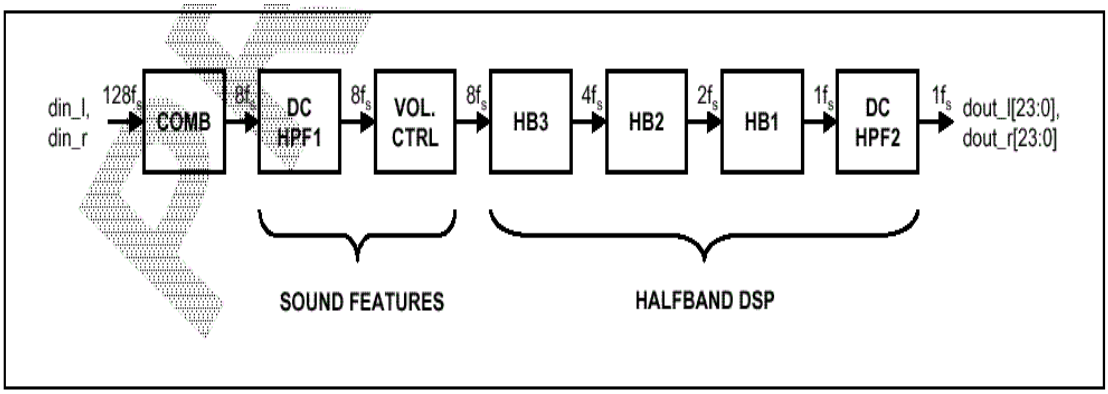


Fig. 64 decimator data path

Table 353 Decimation filter characteristics

ITEM	CONDITION	VALUE (dB)
Pass-band ripple	0 to $0.45f_s$	0.01
Stop band	$>0.55f_s$	-70
Dynamic range	0 to $0.45f_s$	>135
digital output level	at 0dB input analog	-1.5

Please also consult the **AudioSS_creg** section on controlling the next features.

38.2.1 VOLUME CONTROL

The decimator is equipped with a digital volume control. This volume control is separate for left and right and can be set via the SADC_CTRL_DECI register. The range is from +24dB downto -63dB and mute in steps of 0.5dB.

38.2.2 DC BLOCKING FILTER

Two optional 1st order Infinite Impulse Response (IIR) high-pass filters are provided to remove unwanted DC components from the input (DC-offset, DC-dither) and/or volume control output to avoid clipping when using large gain settings. These filters may be bypassed by setting the en_dcfilti (SADC_CTRL_DECI[20]) and/or en_dcfilto (SADC_CTRL_DECI[19]) pins LOW, which is necessary e.g. when fast settling of the decimator is required.

On recovery from power-down or after a reset, the parallel output data on pins dout_l[23:0] and dout_r[23:0] is held at LOW level until valid data is available from the decimation filter. This time depends on which of the DC-blocking filters is selected and if the enable pin of the delay timer is on (en_delay_dblin = SADC_CTRL_DECI[21]):

- *en_delay_dblin* off:
  t = 0
- DC filter1 is off, DC filter 2 is off and *en_delay_dblin* is on:
  $t = 44/f_s$; t = 1 ms at f_s = 44.1kHz
- DC filter1 is on, DC filter 2 is off and *en_delay_dblin* is on:
  t = 17066/fs; t = 387 ms at fs = 44.1kHz
- DC filter 2 is on and *en_delay_dblin* is on:
  t = 67473/fs; t = 1.53 s at fs = 44.1kHz

38.2.3 SOFT START-UP AFTER RESET

After a reset of the decimation filter and if en_dblin is HIGH (SADC_CTRL_DECI[21]), the output gain of the decimator is increased from mute to -63.5 dB and at a rate of 0.5 dB per f_s period to 0 dB (dB linear) to avoid harsh audible plops. The time required for a complete soft start-up if en_dblin is HIGH is 128 f_s periods. This time is without the time required if en_delay_dblin is HIGH (SADC_CTRL_DECI[22])(see paragraph 38.2.2). e.g. if en_dblin and en_delay_dblin are HIGH, en_dcfilti and en_dcfilto are low the total time required is (44 +128) f_s periods (see table Table 354). The decimator soft start-up function is illustrated in Fig. 65. For readability, the parallel output data is shown in its analog representation.

Table 354required time after reset

EN_DELAY_DBLIN	EN_DBLIN	EN_DCFILTI	EN_DCFILTO	REQUIRED TIME
0	0	x	x	0 s
0	1	x	x	128 f_s periods
1	0	0	0	44 f_s periods
1	0	1	0	17066 f_s periods
1	0	x	1	67473 f_s periods
1	1	0	0	(44+128) f_s periods
1	1	1	0	(17066+128) f_s periods
1	1	x	1	(67473+128) f_s periods

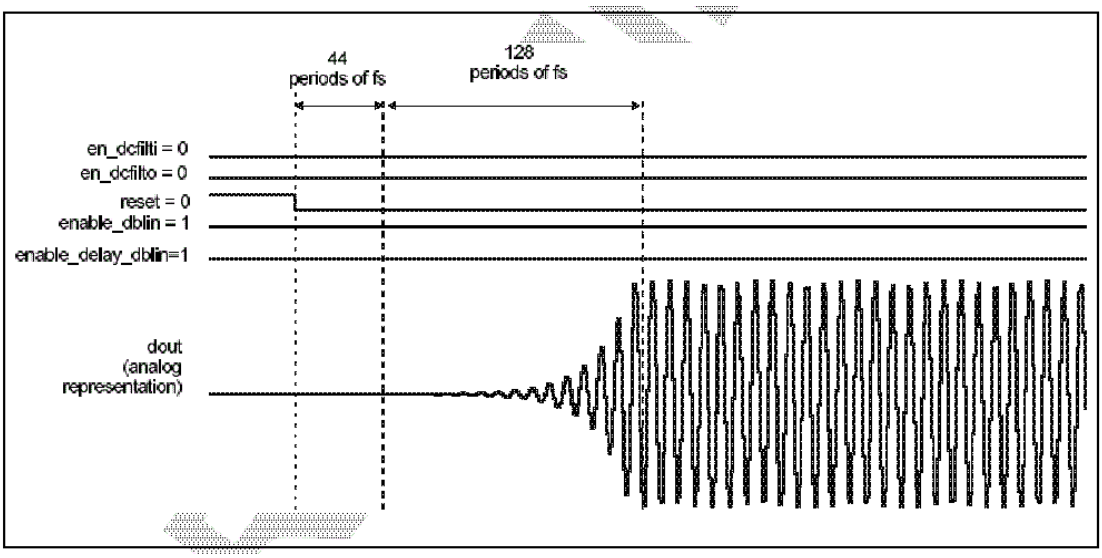


Fig. 65 soft startup function

38.2.4 SIGNAL POLARITY

The polarity of the output signal is controlled by the en_pol_inv pin (SADC_CTRL_DECI[17]). When this pin is enabled, the polarity of the output data is inverted.

38.2.5 MUTE

When the left and right channel of the decimator are muted (en_mute = HIGH), the gain in the decimator is linear decreased to -63.5 dB with a final step to mute at a rate of 0.5 dB per fs period (dB linear). This is done to avoid harsh audible plops. The time required for a complete mute depends on the initial gain setting. Maximum required time is 256 fs periods. When a complete mute is achieved for both left and right channels, the pin mute_state is made HIGH (SADC_CTRL_DECO[0]). When the channels are de-muted (en_mute = LOW) the gain of the decimator is increased at the same rate until the programmed gain setting is achieved.

38.2.6 OVERFLOW DETECTION

In common practice the output signal is used to indicate whenever the output data, in either the left or right channel, is larger than -1.16 dB of the maximum possible digital swing. When this condition is detected the overflow bit (SADC_CTRL_DECO[1]) is forced HIGH for at least $512f_S$ cycles (11.6 ms at f_S = 44.1 kHz) allowing even a slow micro-controller to poll this event. This time-out is reset for each infringement.

38.2.7 AGC FUNCTION

The decimation filter is equipped with an Automatic gain Control block (AGC). This function is intended, when enabled, to keep the output signal at a constant level.

The AGC can be used for microphone applications in which the distance to the microphone is not always the same.

The AGC can be enabled via the control register (SADC_CTRL_DECI[23]). In that case it bypasses the digital volume control. Via the same register also some other settings of the AGC like the attack/decay settings and the target level settings can be done.

Note: The DC filter in front of the decimation filter must be enabled when AGC is in operation, otherwise the output will be disturbed by the DC offset added in the ADC.

38.2.7.1 AGC ENABLE

A 1 bit value to enable/disable the AGC.

Table 355AGC enable control

AGC_EN	AGC FUNCTION
0	off (default), manual gain control via the Left/Right decimator volume control
1	enabled, with manual microphone gain settings via PGA

38.2.7.2 AGC LEVEL

A 2 bit value to set the AGC target level

Table 356AGC target level settings

AGC_LEVEL1	AGC_LEVEL0	AGC TARGET LEVEL VALUE (DB FS)
0	0	-5.5
0	1	-8.0
1	0	-11.5
1	1	-14.0

38.2.7.3 AGC TIME CONSTANTS

A 3 bit value to set the AGC time constants, being the attack and decay time constants. The given constants are for 44.1 kHz and 8kHz sampling frequencies, and must be scaled down or up according to the sampling frequency used.

Table 357AGC Time constant settings

AGC_TIME2	AGC_TIME1	AGC_TIME0	AGC SETTING			
			44.1KHZ SAMPLING		8KHZ SAMPLING	
			Attack time(ms)	Decay time(ms)	Attack time(ms)	Decay time(ms)
0	0	0	11	100	61	551
0	0	1	16	100	88.2	551
0	1	0	11	200	61	1102
0	1	1	16	200	88.2	1102
1	0	0	21	200	116	1102
1	0	1	11	400	61	2205
1	1	0	16	400	88.2	2205
1	1	1	21	400	116	2205

39 STEREO DIGITAL-TO-ANALOG CONVERTER AND HEADPHONE

39.1 *General Description*

The SDAC module is a stereo Digital-to-Analog Converter with interpolation filters and noise shaper for low frequency applications such as audio and TV-audio. In this note the analog and digital part is described. The digital part consists of an interpolation filter that increases the sample rate from 1fs to 128fs and a third order noise shaper that runs on 128fs or 256fs. The analog part consists of two single ended SDAC modules for the left and right channels. For the sub circuit the process option of thick gate oxide is used allowing a supply voltage of 3.3V for the analog part. To reduce leakage the high Vt option is used for the Data Weighted Averaging part of the SDAC.

The inputs to the are two 24-bit parallel input words, left and right, and a synchronisation signal din_valid at fs (fs, the sample rate, is typical 44.1 kHz). The output is a stereo analog signal (vout_linel and vout_liner).

39.2 *Features*

- 24-bit data path with 16-bit coefficients
- Full FIR filter implementation for all of the upsampling filter
- Digital dB-linear volume control in 0.25 dB steps
- Digital de-emphasis for 32 kHz, 44.1 kHz, 48 kHz and 96 kHz
- Selection for the 2f s to 8f s upsampling filter characteristics (sharp/slow-roll-off)
- Support for 2f s and 8f s input signals:

  -1fs with full feature support, being deemphasis, master volume control and soft mute

  -2fs input with master volume and mute support: required for double speed mode

  -8fs input no features supported. This is intended for DSD support = grabbing data at 8fs from an external DSD unit
- Soft mute with a raised cosine function
- Controlled power down sequence comprising a raised cosine mute function followed by a DC ramp down to zero to avoid audible plops or clicks
- Integrated digital silence detection for left and right with selectable silence detection time
- Polarity control
- Simple switched resistors architecture
- Data Weighted Averaging technique reducing distortion
- Large Supply voltage range (0.8 .. 3.6V)

39.3 Block Diagram

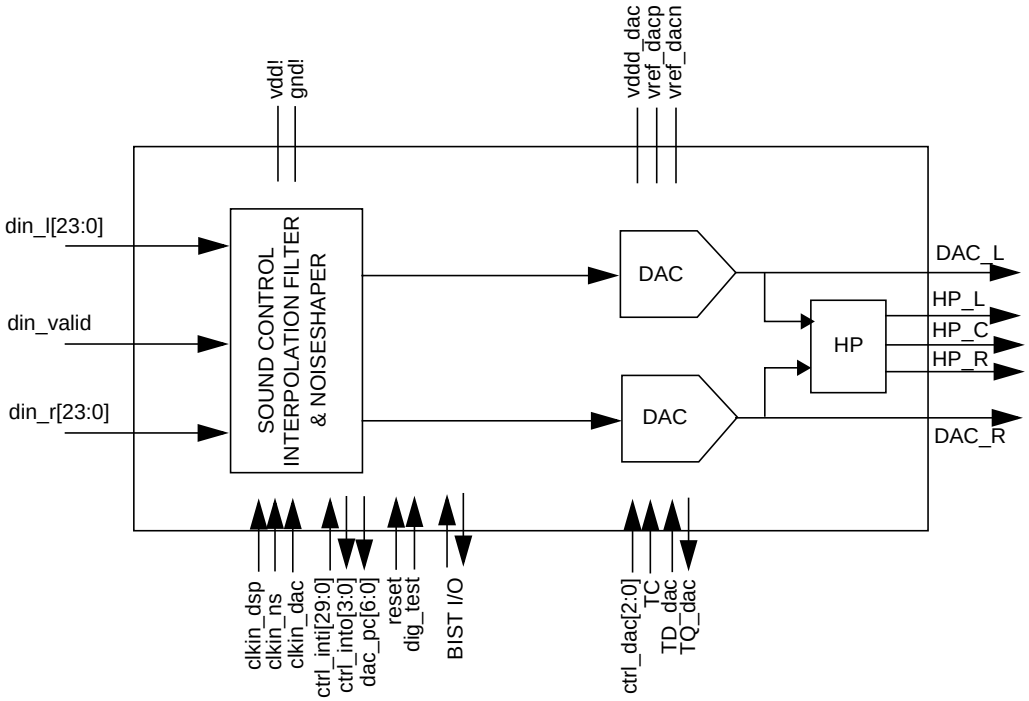


Fig. 66 Block Diagram of the SDAC

39.4 *Functional Description*

The SDAC comprises the following functions:

- Sound feature processor
- Digital upsampling filter
- Noise shaper
- DAC (two instances of the SDAC module)

39.4.1 SOUND FEATURE PROCESSOR

Digital de-emphasis can be set by a 3-bit control bus (ctrl_inti[18:16]) for the range of sample frequencies avail-able (32 kHz, 44.1 kHz, 48 kHz and 96 kHz). The de-emphasis filters are only in the signal path for normal speed mode (data input at 1fs). See table 4 for the exact settings.

39.4.2 DIGITAL VOLUME CONTROL

In the interpolation filter a three wise linear digital volume control is provided with a range from 0 dB to -89 dB and -∞ dB. Down to the attenuation of -50 dB the step size equals 0.25 dB, from -50 dB to -83 dB it equals 3 dB and the last step to -89 dB is one of 6 dB. The attenuation for the left channel is controlled by ctrl_inti[15:8]. the attenuation for the right channel is controlled by ctrl_inti[7:0] (see table 4 for more details).

39.4.3 MUTE

When the left and right channels of the interpolator are muted (ctrl_inti[19] = HIGH), the gain in the interpolator is decreased to -∞ dB conform a raised cosine function to avoid harsh audible plops (soft mute). This mute function is completed after a period of 128 samples in normal mode i.e. 2.9 ms at s = 44.1 kHz. When a complete mute is achieved for both left and right channels, the pin ctrl_into[0] is made HIGH. The interpolator mute function is illustrated in Fig.67.

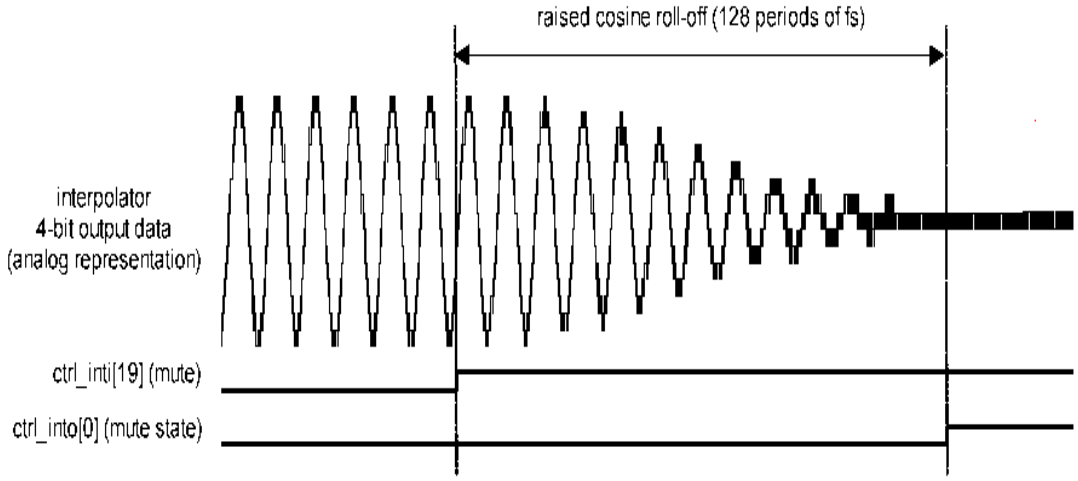


Fig. 67 Interpolator mute signals

39.4.4 POWER DOWN

When the interpolator is powered down (ctrl_inti[25] = HIGH), the gain in the interpolator is decreased to -∞ dB conform a raised cosine function to avoid harsh audible plops (soft mute). This is followed by a DC ramp down to zero output data (hex. code 000000). The slope of this DC ramp can be set by ctrl_inti[24] to either 512 (default) or 1024 fs periods. The power up follows the reverse procedure, i.e. a DC ramp up to mid scale plus DC dither ($2^{-6} + 2^{-10} + 2^{-17}$) followed by a gain increase conform a raised cosine function. Total time required for a full power up/down equals 128 (raised cosine function) plus 512 (DC ramp up/down) make 640 fs periods or 14.5 ms for fs = 44.1 kHz. Total time required for a full power down/up cycle. The power-up and power-down function is illustrated in Fig.68.

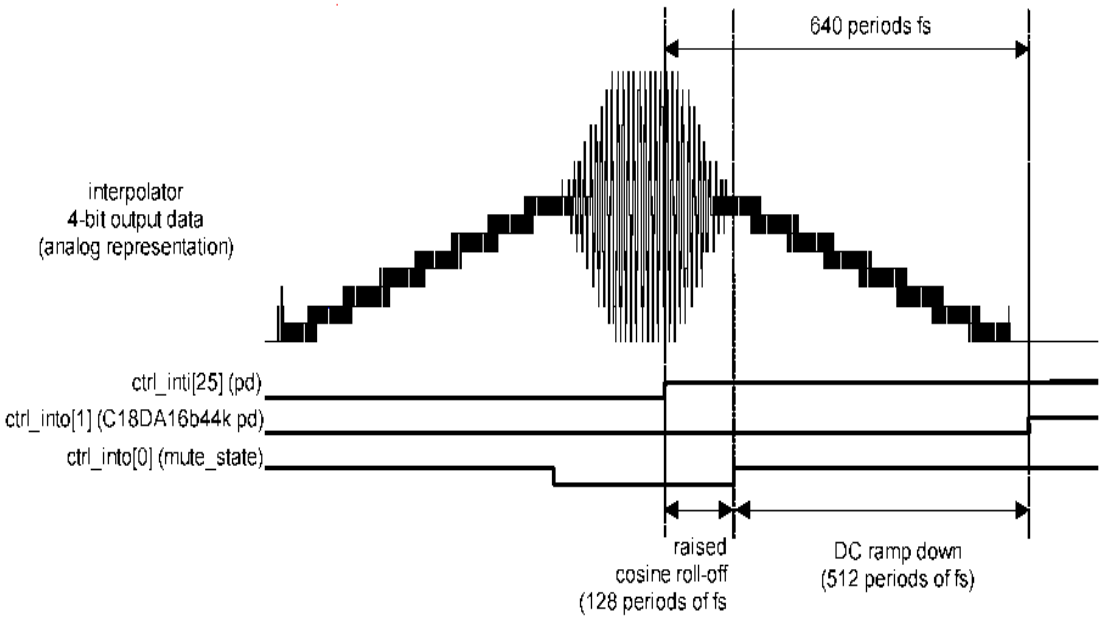


Fig. 68 Interpolator power up and power down sequence

39.4.5 SILENCE DETECTION

The C18INT features a silence detection circuit that counts the number of digital input samples equal to zero. It is enabled by the control pin ctrl_inti[30]. The number of zero samples before signalling 'silence detected' (ctrl_into[3] for left channel and ctrl_into[2] for right channel) can be set by ctrl_inti[29:28]. Note that this feature is not used to control the SDAC, it is simply a feature that can be used in the system where the SDAC is used.

39.4.6 POLARITY CONTROL

The stereo output signal polarity of the C18INT can be changed by setting the ctrl_inti[26] HIGH. Note that this single control bit affects both channels.

39.4.7 DIGITAL UPSAMPLING FILTER

The data path of the C18INT is depicted in Fig.69.

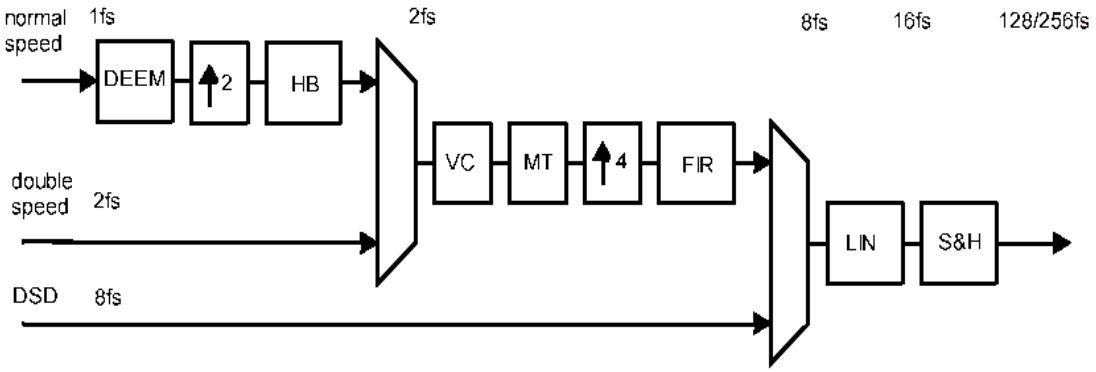


Fig. 69 Interpolator datapath

The interpolation from 1f s to 128f s is realized in four stages:

1. The first stage is a 99 taps half band filter (HB) which increases the sample rate from 1fs to 2fs and has a steep transition band to correct for the missing inherent filter function of the DAC used (SDAC).

2. The second stage is a 31 tap FIR filter which increases the data rate from 2fs to 8fs, scales the signal and compensates for the roll-off caused by the Sample & Hold (S&H) function prior to the noise shaper. For this filter 3 sets of coefficients can be chosen realizing 3 different transfer characteristics.

3. The third stage is a simple hardware linear interpolator (LIN) function that increases the sample rate from 8fs to 16fs and removes the 8fs component in the output spectrum. The main reason for upsampling to 16fs is the fact that the analog DAC (SDAC) only has a first order roll-off function.

4. The fourth and last stage is a sample and hold function (S&H) increasing the sample rate from 16fs to a selectable 128fs or 256fs, depending on the actual input data rate. For input sample rates between 8 kHz and 32 kHz the noise shaper (and DAC!!!) must run on 256fs instead of the typical 128fs to avoid a significant noise increase in the audible frequency band of [0 .. 20 kHz].

The SDAC has 3 modes of operation which are set by the ctrl_inti[[21:20] control input bits:

1. Normal 1f s input mode used for input data rates between 8 kHz and 96 kHz using sharp filter roll-off. De-emphasis (DEEM), volume control (VC) and mute (MT) functions are all available in this mode.

2. 2fs input mode which may be used as: - double speed input when the data rate is between 96 kHz and 200 kHz - a means to get slow roll-off by skipping the first half band filter (HB) Note that in this mode the de-emphasis (DEEM) is not available.

3. 8fs or DSD input mode, in which case the input is obtained form an external DSD block. De-emphasis (DEEM), volume control (VC) and mute (MT) features are unavailable in this mode.

The different modes are further explained in Table 358.

Table 358 Speed mode, filter and clock settings

APPLICATION MODE	CTRL_INTI [21:20] (SPEED MODE)	CTRL_INTI [23:22] (FIR FILTER)	CLKIN_DSP	CLKIN_NS
normal speed mode: low sampling freq. mode (all filters used)	00	00/01/11	128fs	256fs
normal speed mode: normal sampling freq. mode (all filters used)	00	00/01/11	128fs	128fs
double speed mode: slow roll-off (HB skipped)	01/10	00/01/11	128fs	128fs
double speed mode: sharp roll-off (HB skipped)	01/10	10	128fs	128fs
DSD mode, f_s = 44.1 kHz	11	XX	128fs	128fs

The influence of filter settings (ctrl_inti[23:22]) and speed mode settings (ctrl_inti[21:20]) are shown in Table 359.

Table 359 Speed mode and filter settings dependent filter characteristics

CTRL_INTI [21:20] (SPEED MODE)	CTRL_INTI[23:22]	ROLL-OFF	FILTER IN USE	PASS BAND	STOP BAND
00	00/11	sharp	HB, FIR (1fs to 8fs)	<0.4535fs	>0.5465fs
	01	slow1	FIR (2fs to 8fs)	<0.2268fs	>0.7619fs
	10	slow2	FIR (2fs to 8fs)	<0.2268fs	>0.6094fs
01	00/11	not supported	(DSP frequency too high)		
	01	slow1	FIR (2fs to 8fs)	<0.2268fs	>0.7619fs
	10	slow2	FIR (2fs to 8fs)	<0.2268fs	>0.6094fs
11	XX	defined in	external DSD processing	block	

39.4.8 NOISE SHAPER

The 3rd-order noise shaper operates at either 128fs or 256fs depending on the mode of operation defined by ctrl_inti[21:20] and ctrl_inti[23:22]. It shifts in-band quantization noise to frequencies well above the audio band. This noise shaping technique enables high signal-to-noise ratios to be achieved at low frequencies. The noise shaper output is converted into an analog signal using a 4-bit Switched Resistor Digital-to-Analog Converter (SDAC).

39.4.9 DAC (SDAC)

The 4-bit stereo DAC is based on a switched resistor architecture which is merely a controlled voltage divider between the positive and negative reference supplies vref_dacp and vref_dacn. The 4-bit input data from the noise shaper is first decoded to a 15 level thermometer code controlling the 15 taps of the converter. Added to the decoding is a selectable Data Weighted Averaging technique (DWA) which guarantees that there is no correlation between the input signal and the resistors used for that input signal.

After decoding and Data Weighted Averaging the buffers connect the resistors to either the vref_dacp or vref_dacn pin. In doing this the reference voltage will be divided depending on the input signal. The result is an analog output voltage with a rail-to-rail maximum output swing. The output impedance of this DAC is approx. 1 kΩ. By applying an external capacitor of 3.3 nF to the line output pin (vout_linel or vout_liner) a low pass post filter is introduced with a -3 dB roll off at 48 kHz (dimensioned for fs = 44.1 kHz). This will thus reduce the 3rd order noise shaped output spectrum of the DAC

to a noise spectrum increasing with 2nd order. Note that the value of this capacitor depends on the actual sample frequency used.

39.4.10 Data Weighting Averaging

The SDAC features two Data Weighted Averaging (DWA) algorithms which can be selected in depend-ently for the left (ctrl_dac[1]) and right (ctrl_dac[0]) channels. By setting these bits to logic '0' the uni-directional DWA algorithm is choosen which is best suited for good SNR figures. By setting these bits to logic '1' the bi-directional DWA algorithm is choosen which is best for low distortion.

39.4.11 Input data format

The data format of the input signal to the SDAC can be set to either 2's complement (ctrl_dac[2] = '1') or offset binary (ctrl_dac[2] = '0'). It should be noted however that the C18INT only works properly with the 2's complement input data format, so ctrl_dac[2] should always be tied to a logic '1'.

39.4.12 Control input bus C18INT

The interpolation filter is controlled by the control bus sdac_ctrl_inti[29:0]. Please consult the **AudioSS_creg** section for the definitions of this bus.

39.4.13 Control output bus C18INT

The interpolation filter has an control output bus sdac_ctrl_into comprising different indication bits which can be used as input control signals to the customer's system. Please consult the **AudioSS_creg** section for the definition of this bus.

39.4.14 Control input bus SDAC

Please consult the **AudioSS_creg** section for the definition of the SDAC control bus. You can also find more control bits for HP in there**.**

39.4.15 Headphone driver

The headphone driver can deliver 22mW (at 3.0V power supply) into16Ω load.

The headphone driver does NOT need external dc decoupling capacitors because it can be dc-coupled w.r.t. a special headphone output reference voltage. This saves two external capacitors (which is quite useful in a portable device).

Important note:
From the schematic in Fig.66 can be concluded that changes in the load on the DAC outputs influence the output of the headphone. This is because the headphone inputs are directly connected to the DAC outputs.

39.5 Specifications

General characteristics and specifications of the SDAC module are shown in Table 386 and Table 360.

This module may be irreparably damaged if taken outside the specified absolute maximum rating range:

- Digital input signals: -0.5 to 2.5 Vdc
- Digital supply voltage -0.5 to 2.5 Vdc
- Analog supply voltage -0.5 to 4.6 Vdc
- Storage temperature: -65 Deg C to +150 Deg C
- Max. junction temperature: +150 Deg C

Table 360 Specification of the SDAC

PARAMETER	SYMBOL	MIN.	TYP.	MAX.	UNIT
Operation conditions					
Junction temperature	T_{junc}	-40	25	125	C
Supplies					
DAC positive reference voltage	$V_{REFP(DAC)}$	3.0	3.3	3.6	V
DAC negative reference voltage	$V_{REFN(DAC)}$		0		V
DAC analog power supply	$V_{DDA(DAC)}$		$V_{REFP(DAC)}$		V
DAC digital power supply	$V_{DDD(DAC)}$	$V_{REFP(DAC)}$ - 0.25	3.3	3.6	V
Digital positive power supply	V_{DDD}	1.65	1.8	1.95	V
DAC positive reference current (stereo)	$I_{REFP(DAC)}$		1.64		mA
DAC analog supply current (stereo)	$I_{DDA(DAC)}$		140		uA
Digital supply current	I_{DDD}		1.62		mA
General					
Sampling frequency	F_s	8	44.1	96	kHz
Oversampling ratio	OSR		128		
Digital filter					
Interpolation filter characteristics for normal speed mode (ctrl_inti[21:20] = ‘00’)					
Pass band ripple, 0 .. 0.4535f_s			0.02		dB
Stop band, > 0.5465f_s			-72		dB
Gain			-1.1		dB
Dynamic range, 0 .. 0.4535f_s			> 143		dB
SDAC: at typical 3V power supply					
Output voltage; digital i/p level = 0 dBFS	V_{OUT}		1.094		V_{RMS}
Common mode output voltage	V_{CM}		$1/2(V_{REFP(DAC)} + V_{REFN(DAC)})$		V
Output voltage during Power down	V_{OUT}		$V_{REFN(DAC)}$		V
Output resistance	R_{OUT}	0.7	1	1.3	kΩ
Load resistance	R_L	10			kΩ
Resistance between $V_{REFP(DAC)}$ and $V_{REFN(DAC)}$	R_{INT}		4.025		kΩ
(THD+N)/S @ 0 dBFS; f_{in} = 1 kHz; single ended, uni-directional DWA	(THD+N)/S		-76		dB
(THD+N)/S @ 60 dBFS; f_{in} = 1 kHz; single ended, uni-directional DWA; A-weighted	DR		-50		dB(A)
Signal to Noise Ratio, uni-directional DWA; A-weighted	SNR		110		dB(A)
(THD+N)/S @ 0 dBFS; f_{in} = 1 kHz; single ended, bi-directional DWA	(THD+N)/S		-89		dB
(THD+N)/S @ -60 dBFS; f_{in} = 1 kHz; single ended, bi-direction- al DWA; A-weighted	DR		-45		dB(A)

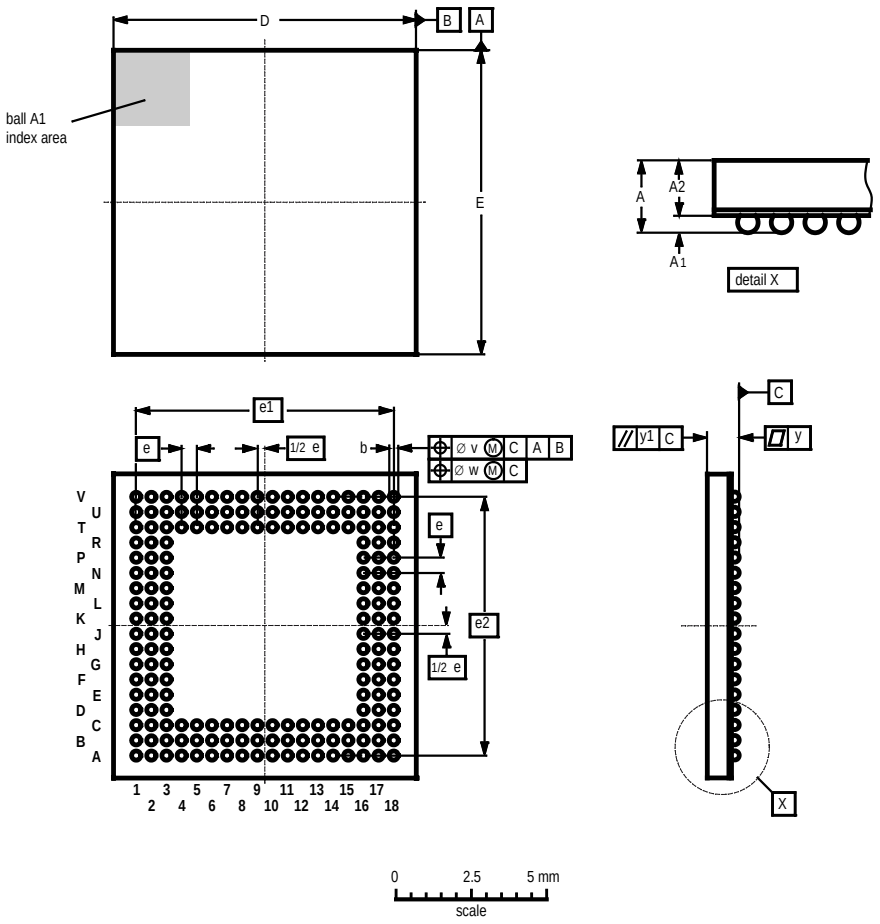
PARAMETER	SYMBOL	MIN.	TYP.	MAX.	UNIT
Signal to Noise Ratio, bi-directional DWA; A-weighted	SNR		105		dB(A)
Channel separation	α_{CS}		100		dB
Data input setup time	$t_{D;S}$	5.4			ns
Data input hold time	$t_{D;H}$	0.6			ns
Data propagation delay (falling edge *clkin_dac* to output voltage)	$t_{P;D}$		4.0		ns

Table 361 Logic level definition for digital inputs/outputs

PARAMETER	SYMBOL	MIN.	TYP.	MAX.	UNIT
Low level input voltage	V_{IL}			$0.2V_{DDD}$	V
High level input voltage	V_{IH}	$0.8V_{DDD}$			V
Low level output voltage	V_{OL}			0.4	V
High level output voltage	V_{OH}	$0.8V_{DDD}$			V

40 PACKAGE OUTLINE

TFBGA180: plastic thin fine-pitch ball grid array package; 180 balls; body 10 x 10 x 0.8 mm **SOT640-1**



DIMENSIONS (mm are the original dimensions)

UNIT	A max.	A_1	A_2	b	D	E	e	e1	e2	v	w	y	y1
mm	1.11	0.31 0.19	0.84 0.76	0.39 0.29	10.1 9.9	10.1 9.9	0.5	8.5	8.5	0.1	0.15	0.12	0.1

OUTLINE VERSION	REFERENCES				EUROPEAN PROJECTION	ISSUE DATE
	IEC	JEDEC	EIAJ			
SOT640-1		MO-195				~~00-10-16~~ 01-06-07

41 ABBREVIATIONS

Table 362 Used abbreviations in this document.

ABBREVIATION	EXPLANATION
Hardware related	
ADPCM	Adaptive Delta Pulse Code Modulation
AHB	Advanced High-performance Bus
APB	Advanced Peripheral Bus
ARM	Advanced RISC Machines
ATX	Analog Transeiver
BCR	Base Control Register
CGU	Clock Generation Unit
DAI	Digital Audio Input
DAO	Digital Audio Output
DEC	Decimator
DSP	Digital Signal Processor
DSS	DSP SubSystem
ECP	Extended Capability Port
EJTAG	Enhanced JTAG
EPICS	Economic Parameterised Integrated CoreS
FI	Input User Flag of Epics
FO	Output User Flag of Epics
IIC	Inter IC Communication
IIS	Inter IC Sound
INT	Interpolator
INTC	Interrupt Controller
JTAG	Joint Test Action Group
LCD	Liquid Crystal Display
MCI	Multimedia Card Interface
MPMC	Multi Purpose Memory Controller
NSOF	Nth Start Of USB Frame
PCR	Power Control Register
PLL	Phase Locked Loop
POR	Power On Reset
PSR	Power Status Register
RISC	Reduced Instruction Set Computer
RNG	Random Number Generator
SAI	Simple Audio Input
SAO	Simple Audio Output
SDMA	Simple Direct Memory Access
SDRAM	Synchronous Dynamic RAM
SPDIF	Sony Philips Digital Input Format
UART	Universal Asynchronous Receiver Transmitter

ABBREVIATION	EXPLANATION
Software related	
AAC	Advanced Audio Compression
AES	Advanced Encryption Standard
DES	Data Encryption Standard
IPSEC	Internet Protocol Security
MD5	Message Digest 5
MP3	MPEG 1 Audio Layer 3
MPEG	Moving Picture Export Group
RC2/RC6	Rivest's Cipher
RSA	Rivest, Shamir and Adelman
SDMI	Secure Digital Music Initiative
SHA-1	Secure Hash Algorithm 1
SSC	SinuSoidal Coding
S/MIME	Secure/ Multipurpose Internet Mail Extensions
WMA	Windows Media Audio

National Semiconductor

April 2003

LM2703
Micropower Step-up DC/DC Converter with 350mA Peak Current Limit

General Description

The LM2703 is a micropower step-up DC/DC in a small 5-lead SOT-23 package. A current limited, fixed off-time control scheme conserves operating current resulting in high efficiency over a wide range of load conditions. The 21V switch allows for output voltages as high as 20V. The low 400ns off-time permits the use of tiny, low profile inductors and capacitors to minimize footprint and cost in space-conscious portable applications. The LM2703 is ideal for LCD panels requiring low current and high efficiency as well as white LED applications for cellular phone back-lighting. The LM2703 can drive up to 4 white LEDs from a single Li-Ion battery.

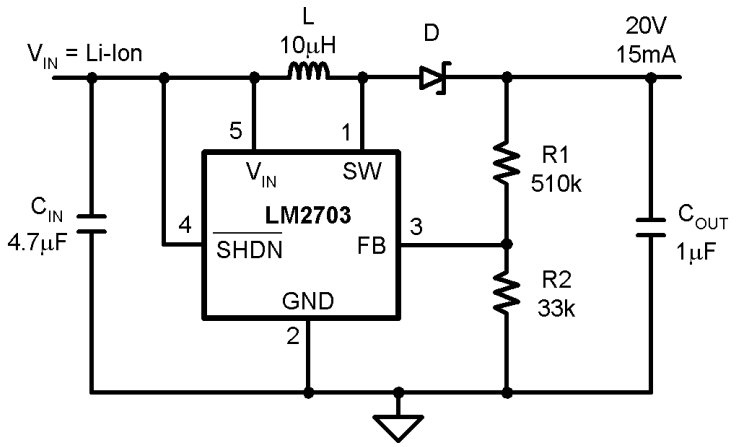
Features

- 350mA, 0.7Ω, internal switch
- Uses small surface mount components
- Adjustable output voltage up to 20V
- 2.2V to 7V input range
- Input undervoltage lockout
- 0.01µA shutdown current
- Small 5-Lead SOT-23 package

Applications

- LCD Bias Supplies
- White LED Back-Lighting
- Handheld Devices
- Digital Cameras
- Portable Applications

Typical Application Circuit



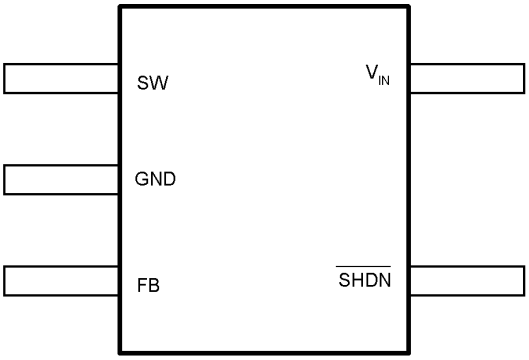
C_{IN}: Taiyo Yuden Ceramic
C_{OUT}: Taiyo Yuden Ceramic
L: Coilcraft DT1608C-103
D: Motorola MBRM130LT3

20030601

FIGURE 1. Typical 20V Application

Connection Diagram

Top View



SOT23-5

T_{Jmax} = 125˚C, θ_{JA} = 220˚C/W (Note 2)

Ordering Information

Order Number	Package Type	NSC Package Drawing	Top Mark	Supplied As
LM2703MF-ADJ	SOT23-5	MA05B	S48B	1000 Units, Tape and Reel
LM2703MFX-ADJ	SOT23-5	MA05B	S48B	3000 Units, Tape and Reel

Pin Description/Functions

Pin	Name	Function
1	SW	Power Switch input.
2	GND	Ground.
3	FB	Output voltage feedback input.
4	$\overline{SHDN}$	Shutdown control input, active low.
5	V_{IN}	Analog and Power input.

SW(Pin 1): Switch Pin. This is the drain of the internal NMOS power switch. Minimize the metal trace area connected to this pin to minimize EMI.

GND(Pin 2): Ground Pin. Tie directly to ground plane.

FB(Pin 3): Feedback Pin. Set the output voltage by selecting values for R1 and R2 using:

$$R1 = R2\left(\frac{V_{OUT}}{1.237V} - 1\right)$$

Connect the ground of the feedback network to an AGND plane which should be tied directly to the GND pin.

$\overline{SHDN}$(Pin 4): Shutdown Pin. The shutdown pin is an active low control. Tie this pin above 1.1V to enable the device. Tie this pin below 0.3V to turn off the device.

V_{IN}(Pin 5): Input Supply Pin. Bypass this pin with a capacitor as close to the device as possible.

Absolute Maximum Ratings (Note 1)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

V_{IN}	7.5V
SW Voltage	21V
FB Voltage	2V
$\overline{SHDN}$ Voltage	7.5V
Maximum Junction Temp. T_J (Note 2)	150˚C
Lead Temperature (Soldering 10 sec.)	300˚C
Vapor Phase (60 sec.)	215˚C
Infrared (15 sec.)	220˚C
ESD Ratings (Note 3)	
Human Body Model	2kV
Machine Model (Note 4)	200V

Operating Conditions

Junction Temperature (Note 5)	−40˚C to +125˚C
Supply Voltage	2.2V to 7V
SW Voltage Max.	20.5V

Electrical Characteristics

Specifications in standard type face are for T_J = 25˚C and those in **boldface type** apply over the full **Operating Temperature Range** (T_J = −40˚C to +125˚C). Unless otherwise specified. V_{IN} =2.2V.

Symbol	Parameter	Conditions	Min (Note 5)	Typ (Note 6)	Max (Note 5)	Units
I_Q	Device Disabled	FB = 1.3V		40	**70**	µA
	Device Enabled	FB = 1.2V		235	**300**	
	Shutdown	$\overline{SHDN}$ = 0V		0.01	2.5	
V_{FB}	FeedbackTrip Point		**1.189**	1.237	**1.269**	V
I_{CL}	Switch Current Limit		275 **260**	350	400 **400**	mA
I_B	FB Pin Bias Current	FB = 1.23V (Note 7)		30	**120**	nA
V_{IN}	Input Voltage Range		**2.2**		**7.0**	V
R_{DSON}	Switch R_{DSON}			0.7	**1.6**	Ω
T_{OFF}	Switch Off Time			400		ns
I_{SD}	$\overline{SHDN}$ Pin Current	$\overline{SHDN}$ = V_{IN}, T_J = 25˚C		0	80	nA
		$\overline{SHDN}$ = V_{IN}, T_J = 125˚C		15		
		$\overline{SHDN}$ = GND		0		
I_L	Switch Leakage Current	V_{SW} = 20V		0.05	5	µA
UVP	Input Undervoltage Lockout	ON/OFF Threshold		1.8		V
V_{FB} Hysteresis	Feedback Hysteresis			8		mV
$\overline{SHDN}$ Threshold	$\overline{SHDN}$ low			0.7	**0.3**	V
	$\overline{SHDN}$ High		**1.1**	0.7		
θ_{JA}	Thermal Resistance			220		˚C/W

Note 1: Absolute maximum ratings are limits beyond which damage to the device may occur. Operating Ratings are conditions for which the device is intended to be functional, but device parameter specifications may not be guaranteed. For guaranteed specifications and test conditions, see the Electrical Characteristics.

Note 2: The maximum allowable power dissipation is a function of the maximum junction temperature, T_J(MAX), the junction-to-ambient thermal resistance, θ_{JA}, and the ambient temperature, T_A. See the Electrical Characteristics table for the thermal resistance. The maximum allowable power dissipation at any ambient temperature is calculated using: P_D (MAX) = ($T_{J(MAX)}$ − T_A)/θ_{JA}. Exceeding the maximum allowable power dissipation will cause excessive die temperature.

Note 3: The human body model is a 100 pF capacitor discharged through a 1.5 kΩ resistor into each pin. The machine model is a 200 pF capacitor discharged directly into each pin.

Note 4: ESD susceptibility using the machine model is 150V for SW pin.

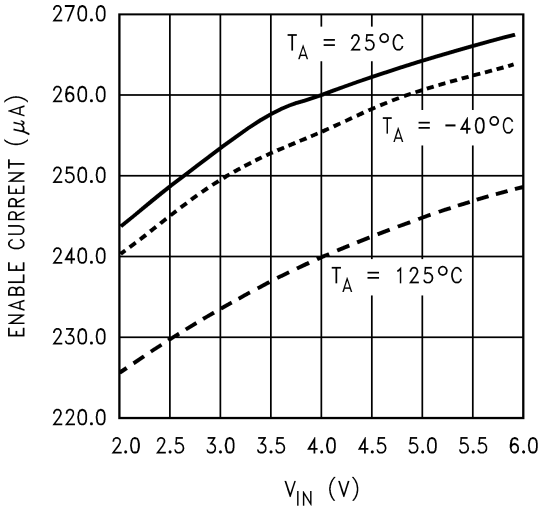
Note 5: All limits guaranteed at room temperature (standard typeface) and at temperature extremes (bold typeface). All room temperature limits are 100% production tested or guaranteed through statistical analysis. All limits at temperature extremes are guaranteed via correlation using standard Statistical Quality Control (SQC) methods. All limits are used to calculate Average Outgoing Quality Level (AOQL).

Note 6: Typical numbers are at 25˚C and represent the most likely norm.

Note 7: Feedback current flows into the pin.

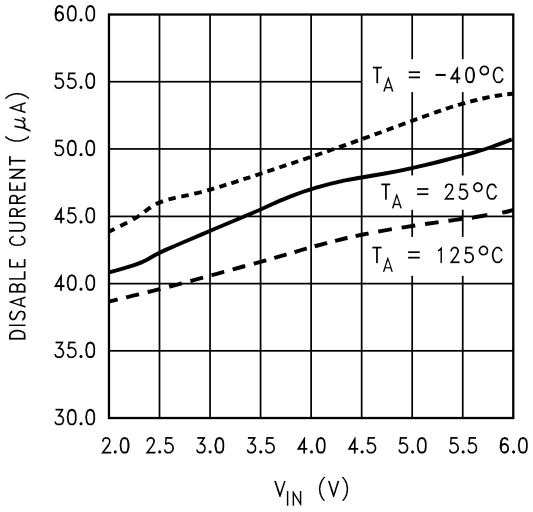
Typical Performance Characteristics

Enable Current vs V_{IN} (Part Switching)



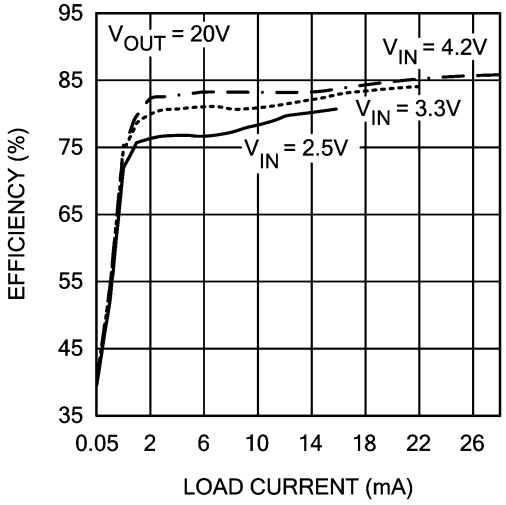
20030605

Disable Current vs V_{IN} (Part Not Switching)



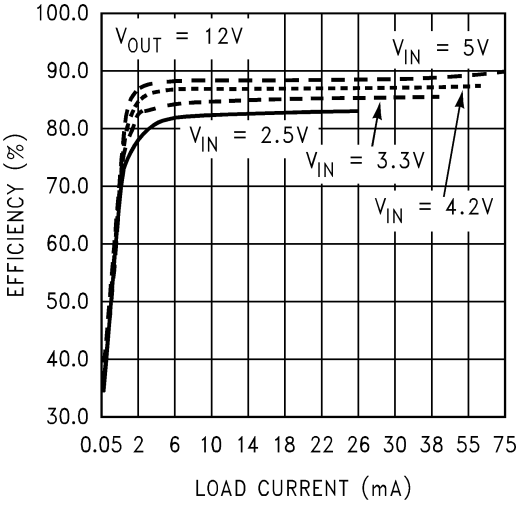
20030606

Efficiency vs Load Current



20030610

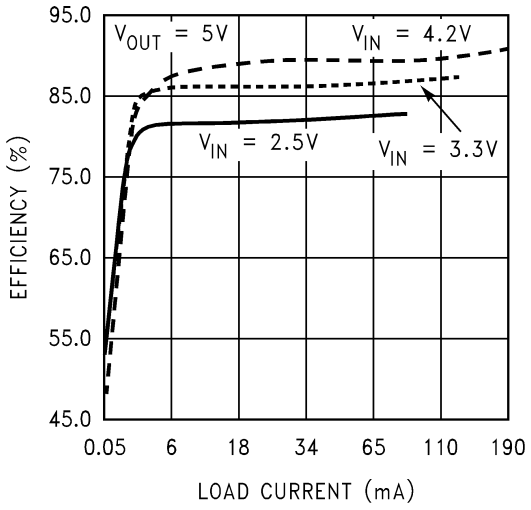
Efficiency vs Load Current



20030611

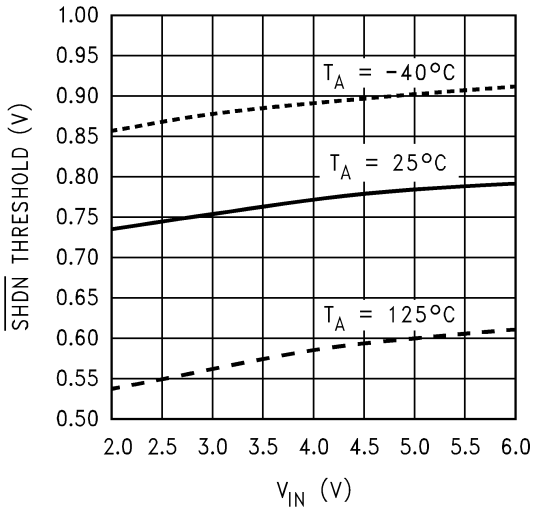
Typical Performance Characteristics (Continued)

Efficiency vs Load Current



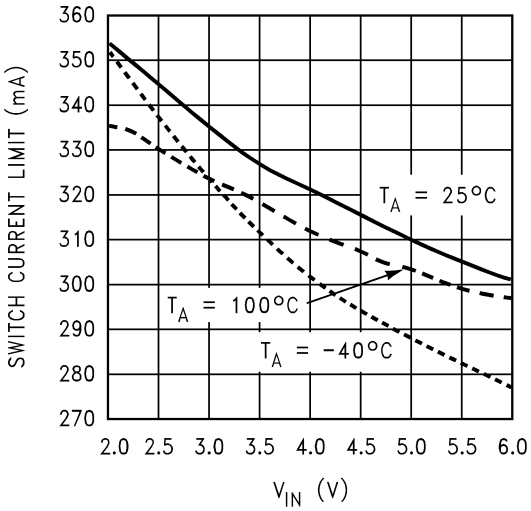
20030612

$\overline{SHDN}$ Threshold vs V_{IN}



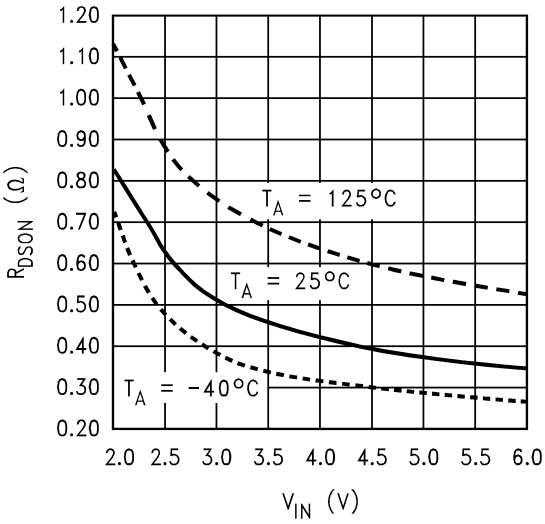
20030613

Switch Current Limit vs V_{IN}



20030614

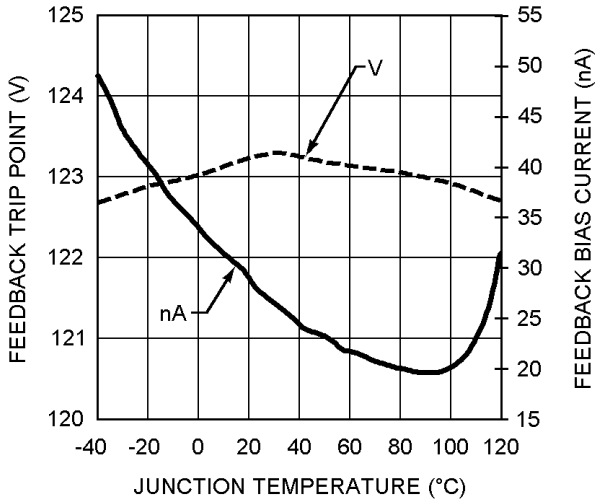
Switch R_{DSON} vs V_{IN}



20030615

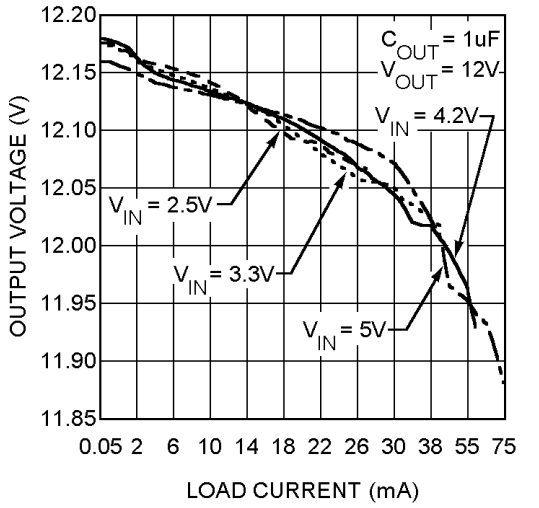
Typical Performance Characteristics (Continued)

FB Trip Point and FB Pin Current vs Temperature



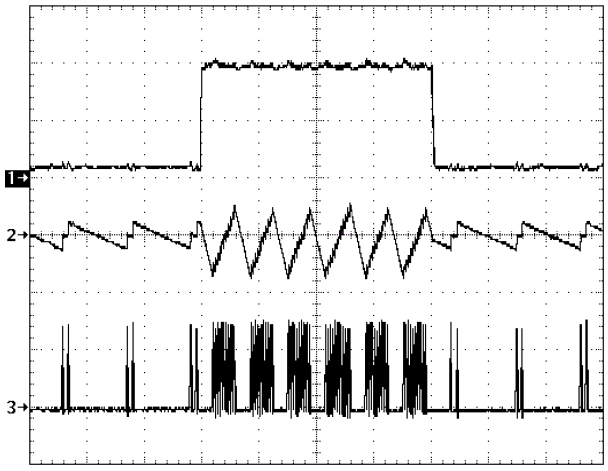
20030623

Output Voltage vs Load Current



20030622

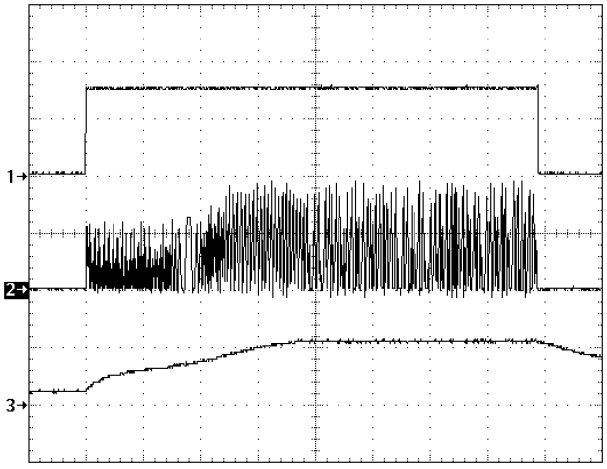
Step Response



20030616

V_{OUT} = 20V, V_{IN} = 2.5V
1) Load, 1mA to 10mA to 1mA, DC
2) V_{OUT}, 200mV/div, AC
3) I_L, 200mA/div, DC
T = 50µs/div

Start-Up/Shutdown



20030617

V_{OUT} = 20V, V_{IN} = 2.5V
1) $\overline{SHDN}$, 1V/div, DC
2) I_L, 200mA/div, DC
3) V_{OUT}, 20V/div, DC
T = 400µs/div
R_L = 1.8kΩ

Operation

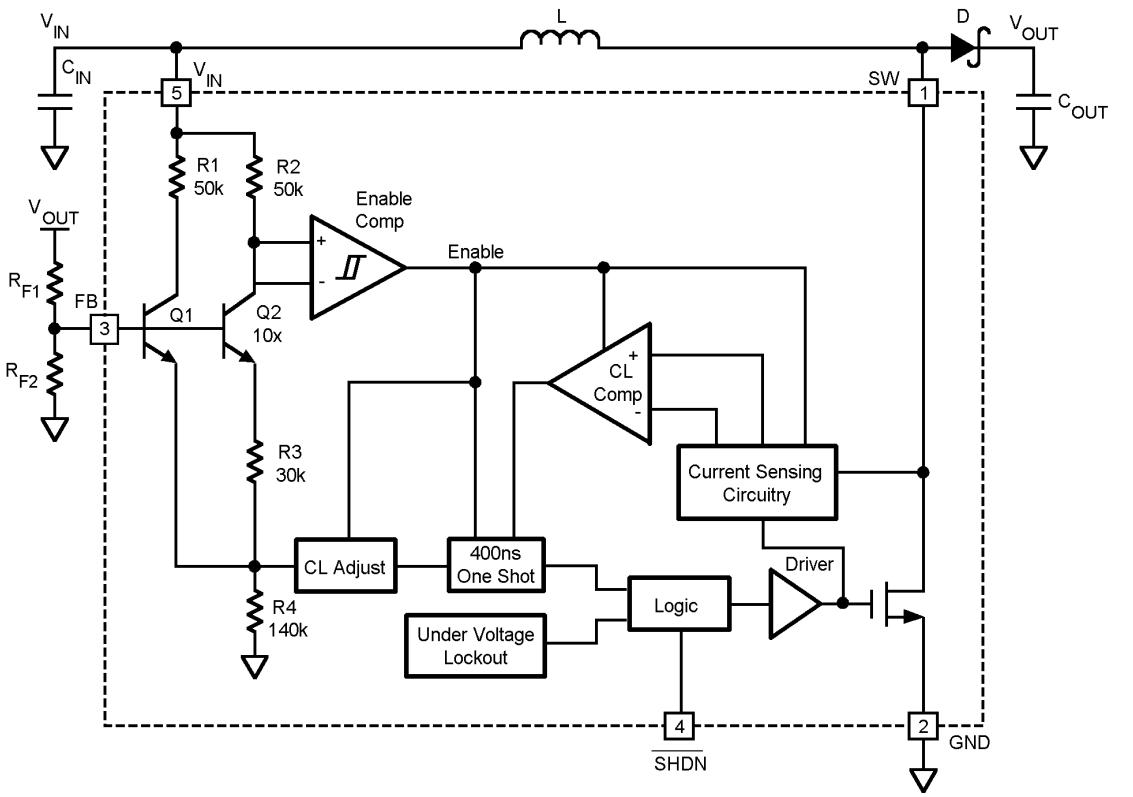
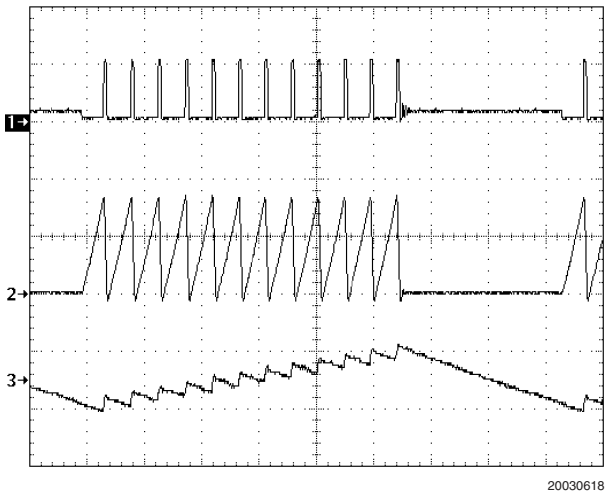


FIGURE 2. LM2703 Block Diagram



V_{OUT} = 20V, V_{IN} = 2.5V
1) V_{SW}, 20V/div, DC
2) Inductor Current, 200mA/div, DC
3) V_{OUT}, 200mV/div, AC
T = 4µs/div

FIGURE 3. Typical Switching Waveform

Operation (Continued)

The LM2703 features a constant off-time control scheme. Operation can be best understood by referring to *Figure 2* and *Figure 3*. Transistors Q1 and Q2 and resistors R3 and R4 of *Figure 2* form a bandgap reference used to control the output voltage. When the voltage at the FB pin is less than 1.237V, the Enable Comp in *Figure 2* enables the device and the NMOS switch is turned on pulling the SW pin to ground. When the NMOS switch is on, current begins to flow through inductor L while the load current is supplied by the output capacitor C_{OUT}. Once the current in the inductor reaches the current limit, the CL Comp trips and the 400ns One Shot turns off the NMOS switch.The SW voltage will then rise to the output voltage plus a diode drop and the inductor current will begin to decrease as shown in *Figure 3*. During this time the energy stored in the inductor is transferred to C_{OUT} and the load. After the 400ns off-time the NMOS switch is turned on and energy is stored in the inductor again. This energy transfer from the inductor to the output causes a stepping effect in the output ripple as shown in *Figure 3*.

This cycle is continued until the voltage at FB reaches 1.237V. When FB reaches this voltage, the enable comparator then disables the device turning off the NMOS switch and reducing the Iq of the device to 40uA. The load current is then supplied solely by C_{OUT} indicated by the gradually decreasing slope at the output as shown in *Figure 3*. When the FB pin drops slightly below 1.237V, the enable comparator enables the device and begins the cycle described previously. The $\overline{SHDN}$ pin can be used to turn off the LM2703 and reduce the I_q to 0.01µA. In shutdown mode the output voltage will be a diode drop lower than the input voltage.

Application Information

INDUCTOR SELECTION

The appropriate inductor for a given application is calculated using the following equation:

$$L = \left(\frac{V_{OUT} - V_{IN(min)} + V_D}{I_{CL}} \right) T_{OFF}$$

where V_D is the schottky diode voltage, I_{CL} is the switch current limit found in the *Typical Performance Characteristics* section, and T_{OFF} is the switch off time. When using this equation be sure to use the minimum input voltage for the application, such as for battery powered applications. For the LM2703 constant-off time control scheme, the NMOS power switch is turned off when the current limit is reached. There is approximately a 200ns delay from the time the current limit is reached in the NMOS power switch and when the internal logic actually turns off the switch. During this 200ns delay, the peak inductor current will increase. This increase in inductor current demands a larger saturation current rating for the inductor. This saturation current can be approximated by the following equation:

$$I_{PK} = I_{CL} + \left(\frac{V_{IN(max)}}{L} \right) 200ns$$

Choosing inductors with low ESR decrease power losses and increase efficiency.

Care should be taken when choosing an inductor. For applications that require an input voltage that approaches the output voltage, such as when converting a Li-Ion battery voltage to 5V, the 400ns off time may not be enough time to discharge the energy in the inductor and transfer the energy to the output capacitor and load. This can cause a ramping effect in the inductor current waveform and an increased ripple on the output voltage. Using a smaller inductor will cause the I_{PK} to increase and will increase the output voltage ripple further. This can be solved by adding a 4.7pF capacitor across the R_{F1} feedback resistor (*Figure 2*) and slightly increasing the output capacitor. A smaller inductor can then be used to ensure proper discharge in the 400ns off time.

DIODE SELECTION

To maintain high efficiency, the average current rating of the schottky diode should be larger than the peak inductor current, I_{PK}. Schottky diodes with a low forward drop and fast switching speeds are ideal for increasing efficiency in portable applications. Choose a reverse breakdown of the schottky diode larger than the output voltage.

CAPACITOR SELECTION

Choose low ESR capacitors for the output to minimize output voltage ripple. Multilayer ceramic capacitors are the best choice. For most applications, a 1µF ceramic capacitor is sufficient. For some applications a reduction in output voltage ripple can be achieved by increasing the output capacitor.

Local bypassing for the input is needed on the LM2703. Multilayer ceramic capacitors are a good choice for this as well. A 4.7µF capacitor is sufficient for most applications. For additional bypassing, a 100nF ceramic capacitor can be used to shunt high frequency ripple on the input.

LAYOUT CONSIDERATIONS

The input bypass capacitor C_{IN}, as shown in *Figure 1*, must be placed close to the IC. This will reduce copper trace resistance which effects input voltage ripple of the IC. For additional input voltage filtering, a 100nF bypass capacitor can be placed in parallel with C_{IN} to shunt any high frequency noise to ground. The output capacitor, C_{OUT}, should also be placed close to the IC. Any copper trace connections for the Cout capacitor can increase the series resistance, which directly effects output voltage ripple. The feedback network, resistors R1 and R2, should be kept close to the FB pin to minimize copper trace connections that can inject noise into the system. The ground connection for the feedback resistor network should connect directly to an analog ground plane. The analog ground plane should tie directly to the GND pin. If no analog ground plane is available, the ground connection for the feedback network should tie directly to the GND pin. Trace connections made to the inductor and schottky diode should be minimized to reduce power dissipation and increase overall efficiency.

Application Information (Continued)

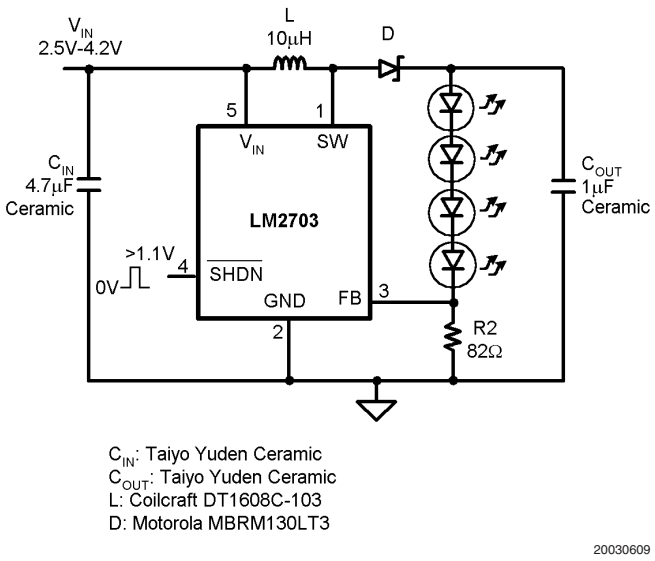


FIGURE 4. White LED Application

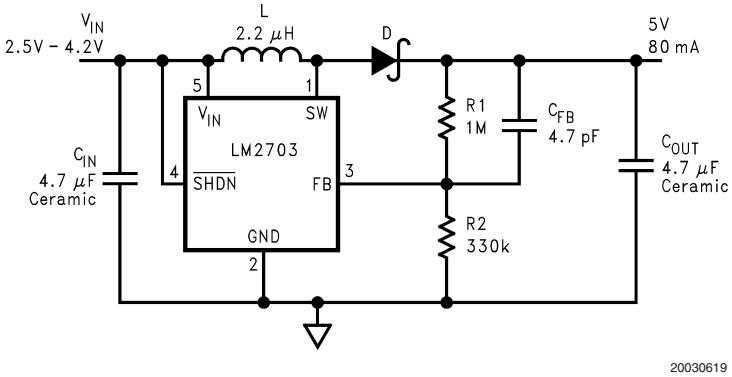


FIGURE 5. Li-Ion 5V Application

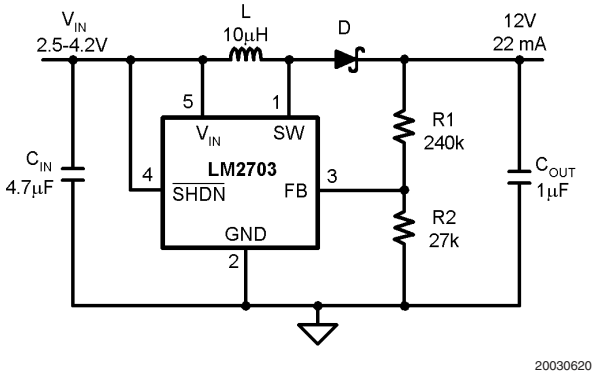


FIGURE 6. Li-Ion 12V Application

Application Information (Continued)

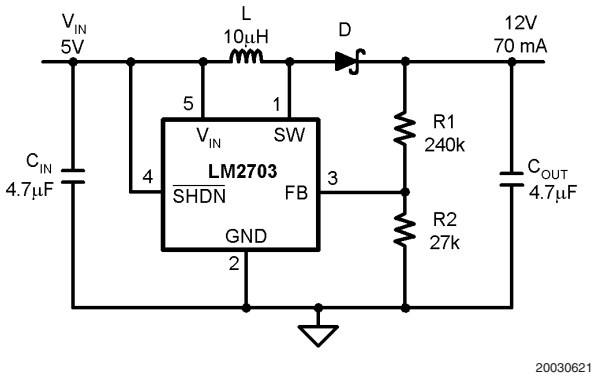
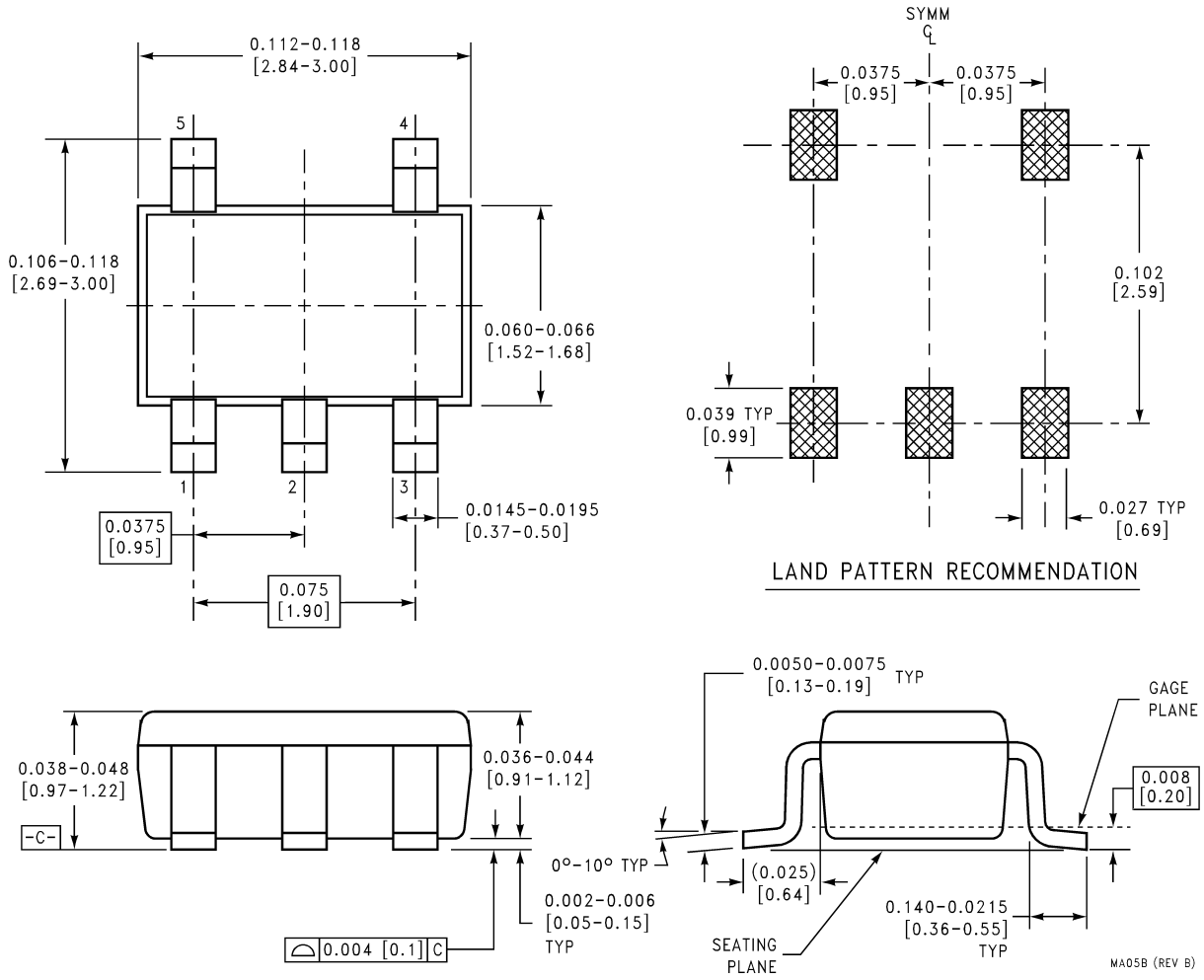


FIGURE 7. 5V to 12V Application

Physical Dimensions inches (millimeters) unless otherwise noted



5-Lead Small Outline Package (M5)
For Ordering, Refer to Ordering Information Table
NS Package Number MA05B

LIFE SUPPORT POLICY

NATIONAL'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF THE PRESIDENT AND GENERAL COUNSEL OF NATIONAL SEMICONDUCTOR CORPORATION. As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

National Semiconductor
Americas Customer
Support Center
Email: new.feedback@nsc.com
Tel: 1-800-272-9959

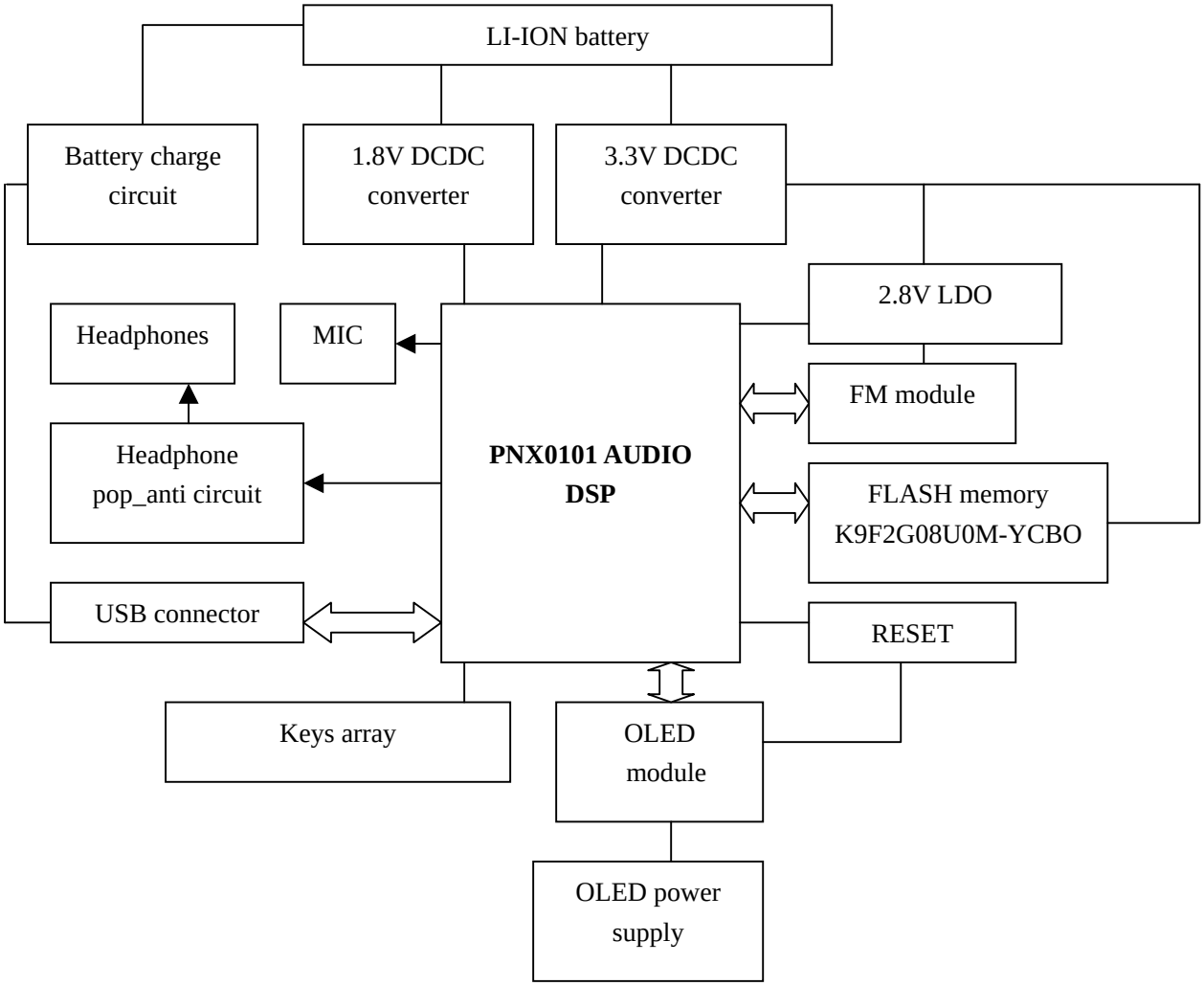
www.national.com

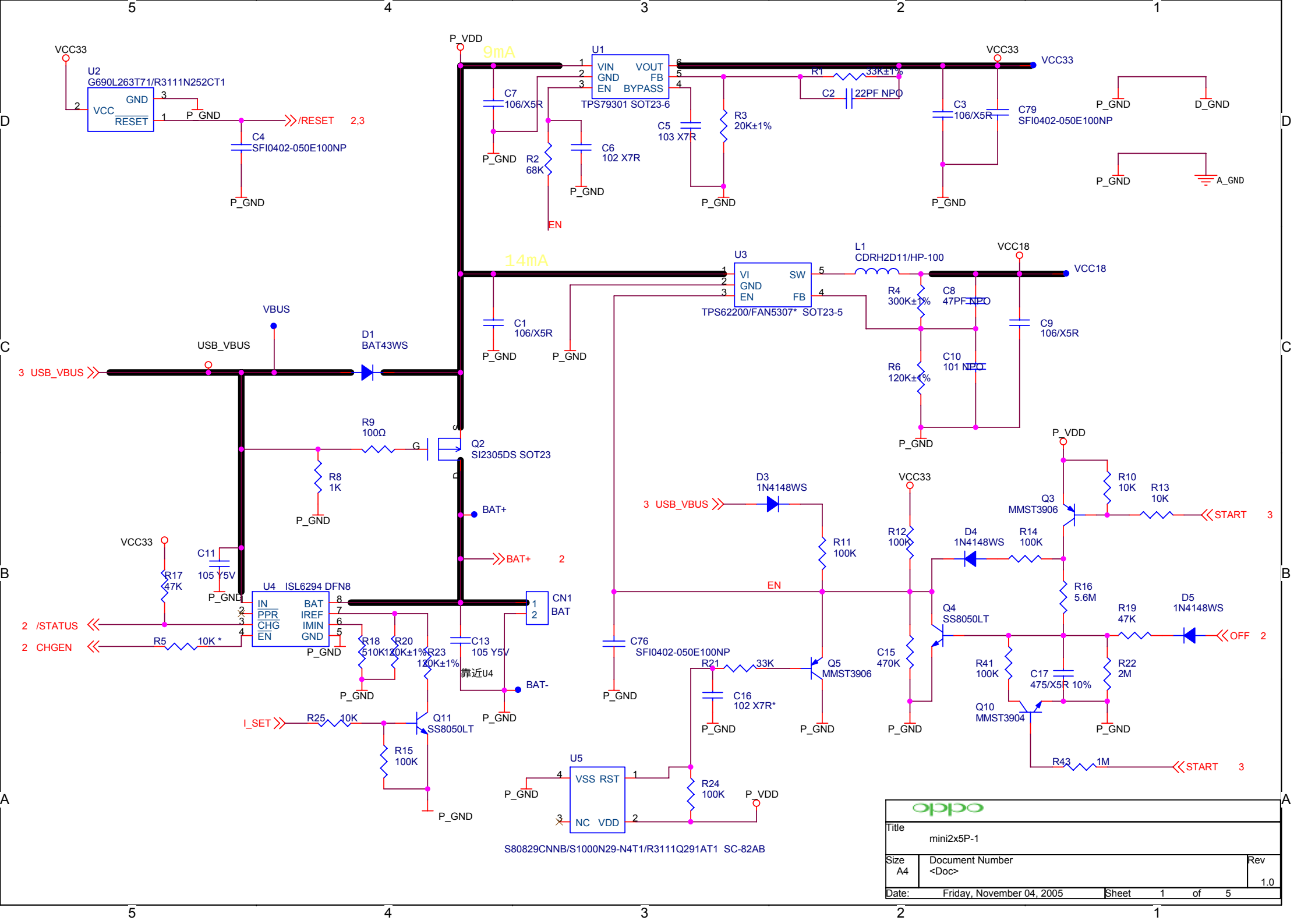
National Semiconductor
Europe Customer Support Center
Fax: +49 (0) 180-530 85 86
Email: europe.support@nsc.com
Deutsch Tel: +49 (0) 69 9508 6208
English Tel: +44 (0) 870 24 0 2171
Français Tel: +33 (0) 1 41 91 8790

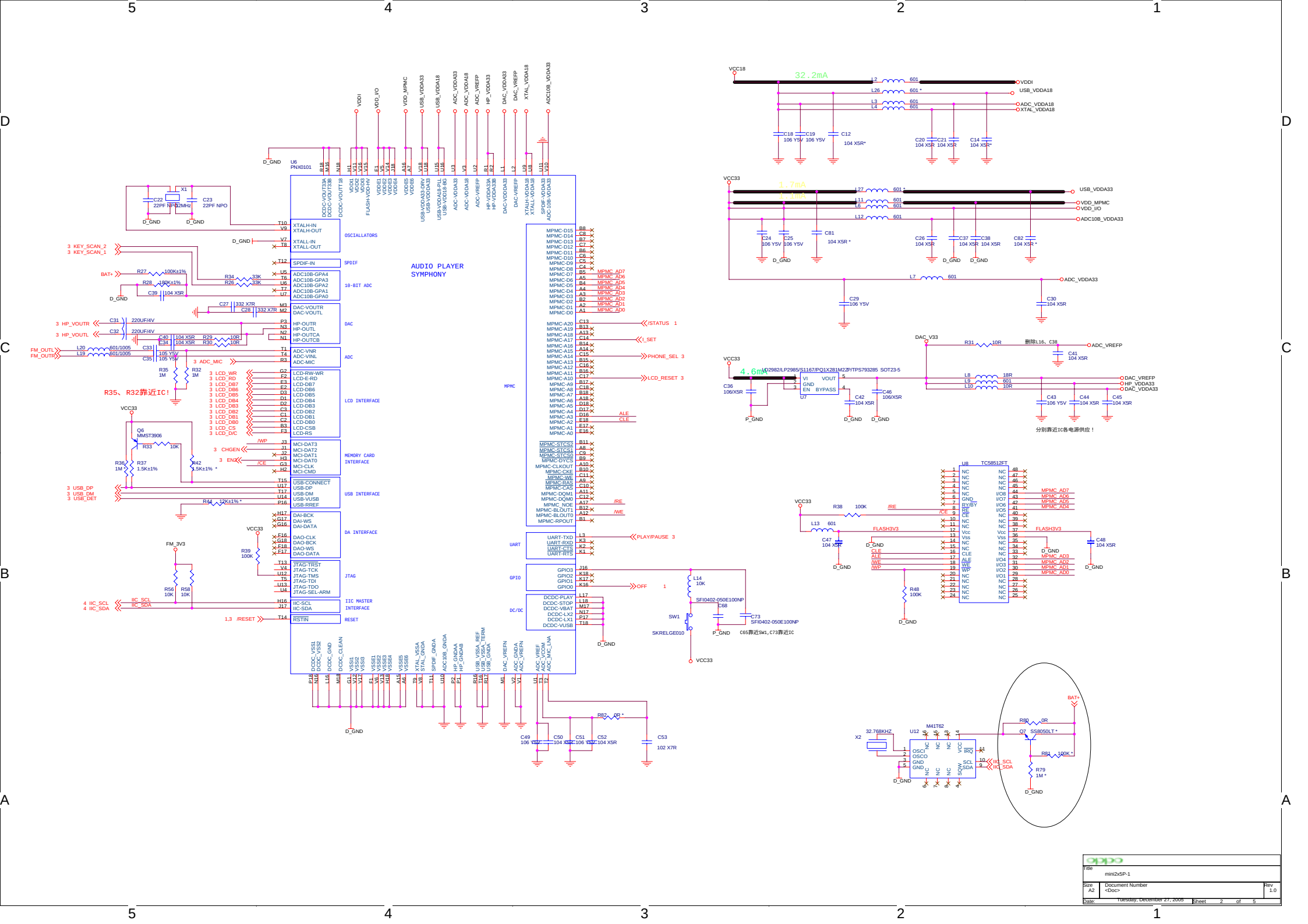
National Semiconductor
Asia Pacific Customer
Support Center
Fax: +65-6250 4466
Email: ap.support@nsc.com
Tel: +65-6254 4466

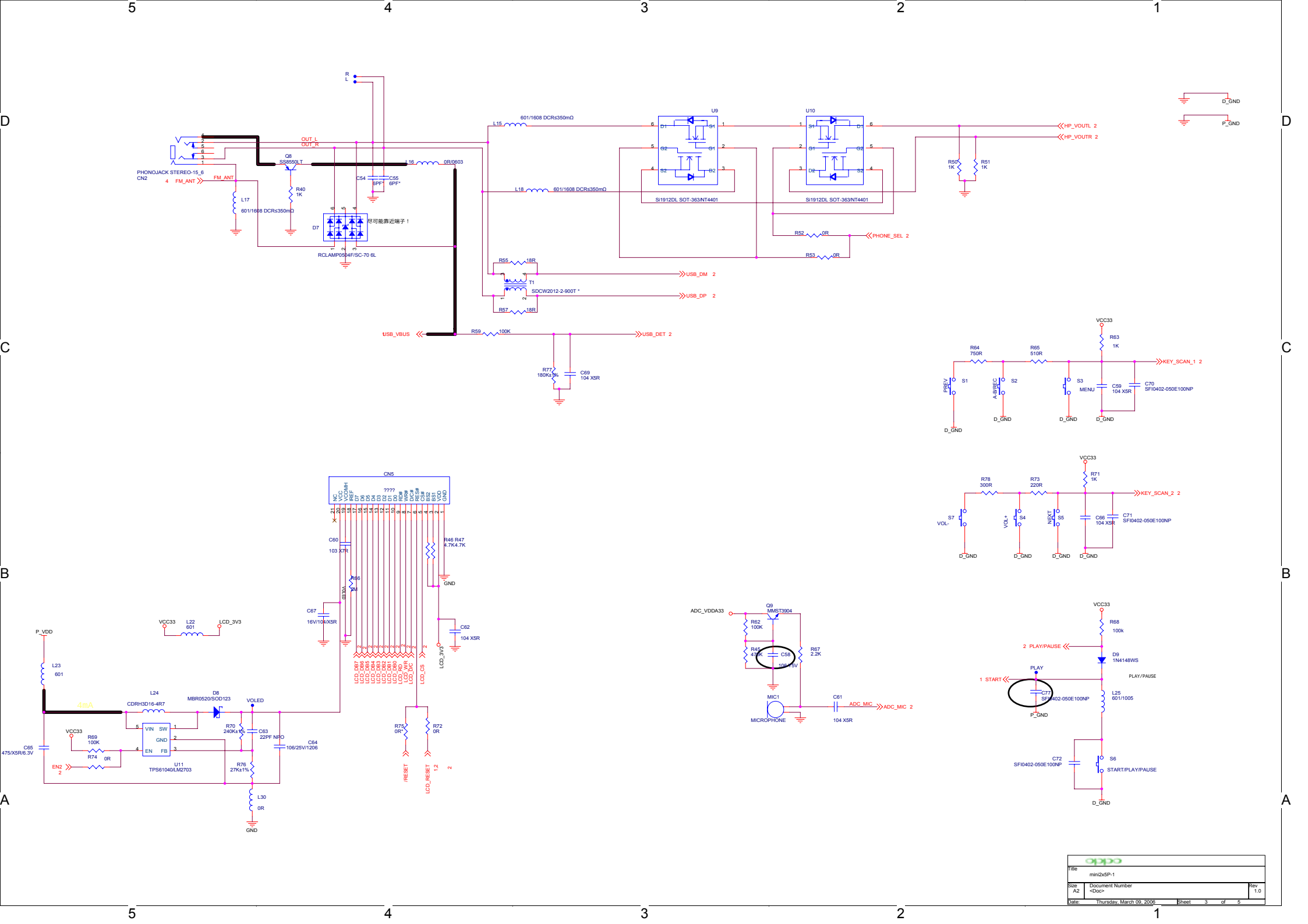
National Semiconductor
Japan Customer Support Center
Fax: 81-3-5639-7507
Email: jpn.feedback@nsc.com
Tel: 81-3-5639-7560

X5 mini block diagram




VCC33
U2
G690L263T71/R3111N252CT1
GND
VCC
RESET
P_GND
/RESET 2,3
C4
SFI0402-050E100NP
P_VDD
9mA
U1
VIN VOUT
GND FB
EN BYPASS
TPS79301 SOT23-6
C7
106/X5R
R2
68K
C6
102 X7R
C5
103 X7R
R3
20K±1%
R1
33K±1%
C2
22PF NPO
C3
106/X5R
C79
SFI0402-050E100NP
D_GND
A_GND
EN
14mA
U3
VI SW
GND
EN FB
TPS62200/FAN5307* SOT23-5
L1
CDRH2D11/HP-100
VCC18
R4
300K±1%
C8
47PF NPO
C9
106/X5R
C10
101 NPO
R6
120K±1%
C1
106/X5R
VBUS
USB_VBUS
D1
BAT43WS
3 USB_VBUS
R9
100Ω
Q2
SI2305DS SOT23
R8
1K
BAT+
BAT+ 2
VCC33
C11
105 Y5V
R17
47K
U4 ISL6294 DFN8
IN BAT
PPR IREF
CHG IMIN
EN GND
2 /STATUS
2 CHGEN
R5
10K *
R18
510K
R20
120K±1%
R23
120K±1%
C13
105 Y5V
靠近U4
CN1
BAT
BAT-
I_SET
R25
10K
Q11
SS8050LT
R15
100K
D3
1N4148WS
3 USB_VBUS
R11
100K
EN
C76
SFI0402-050E100NP
R21
33K
Q5
MMST3906
C16
102 X7R*
U5
VSS RST
NC VDD
R24
100K
S80829CNNB/S1000N29-N4T1/R3111Q291AT1 SC-82AB
R12
100K
D4
1N4148WS
R14
100K
C15
470K
Q3
MMST3906
R10
10K
R13
10K
START 3
R16
5.6M
Q4
SS8050LT
R19
47K
D5
1N4148WS
OFF 2
R41
100K
C17
475/X5R 10%
R22
2M
Q10
MMST3904
R43
1M
START 3
oppo
Title
mini2x5P-1
Size
A4
Document Number
<Doc>
Rev
1.0
Date:
Friday, November 04, 2005
Sheet 1 of 5

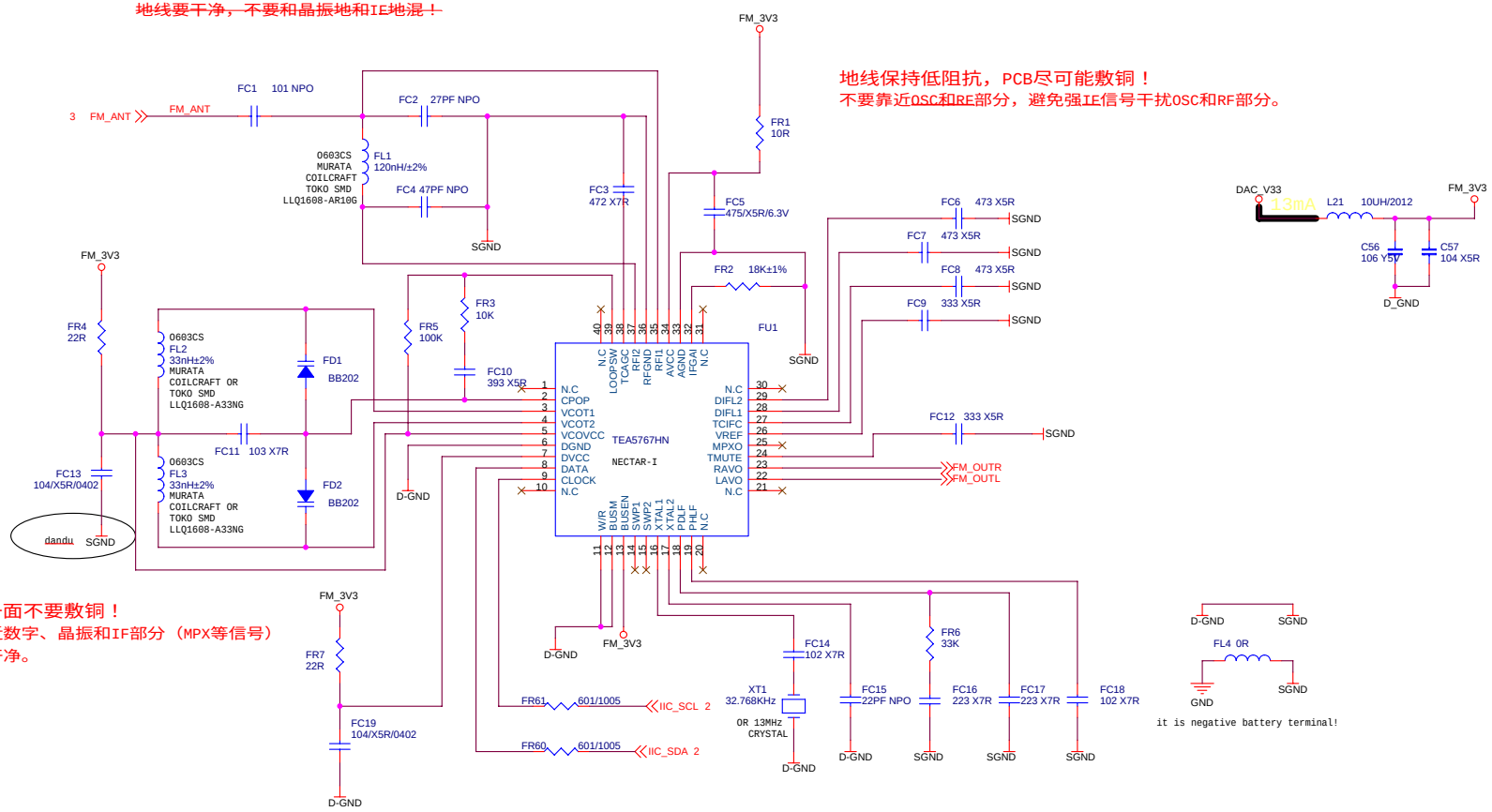

AUDIO PLAYER
SYMPHONY
U6
PNX0101
OSCILLATORS
SPDIF
10-BIT ADC
DAC
ADC
LCD INTERFACE
MEMORY CARD
INTERFACE
USB INTERFACE
DA INTERFACE
JTAG
IIC MASTER
INTERFACE
RESET
MPMC
UART
GPIO
DC/DC
R35、R32靠近IC!
分别靠近IC各电源供应!
C65靠近SW1,C73靠近IC
U8
TC58512FT
U12
M41T62
U7
LD2982/LP2985/S1167/PQ1X281M22P/TPS793285 SOT23-5
Title
mini2x5P-1
Size
A2
Document Number
<Doc>
Rev
1.0
Date:
Tuesday, December 27, 2005
Sheet
2
of
5



Title	mini2x5P-1
Size	A2
Document Number	<Doc>
Rev	1.0
Date:	Thursday, March 09, 2006
Sheet	3 of 5

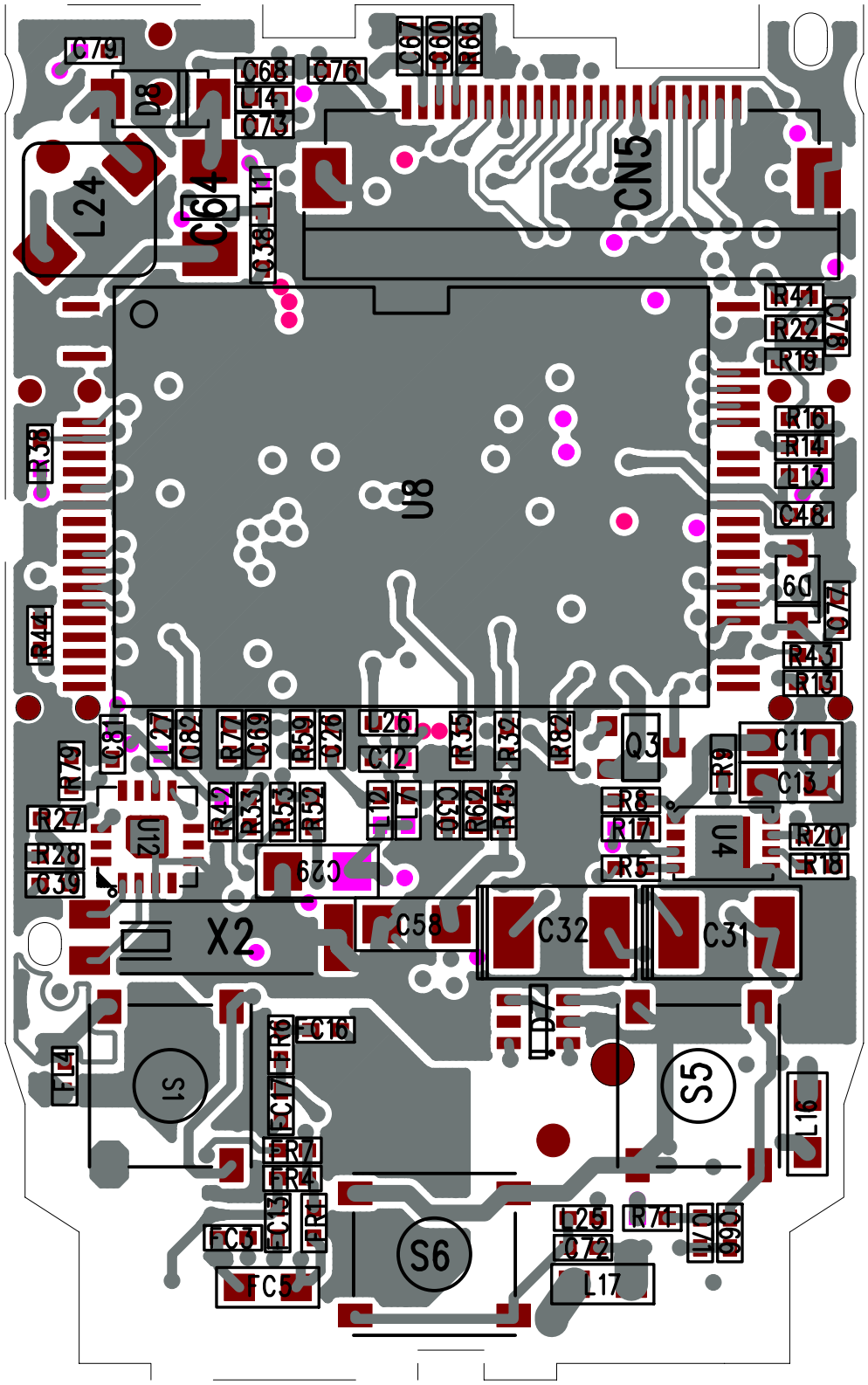
PCB另一面不要敷铜！
不要靠近晶振和IF部分（MPX等信号）
~~地线要干净，不要和晶振地和IF地混！~~

地线保持低阻抗，PCB尽可能敷铜！
不要靠近OSC和RF部分，避免强IF信号干扰OSC和RF部分。

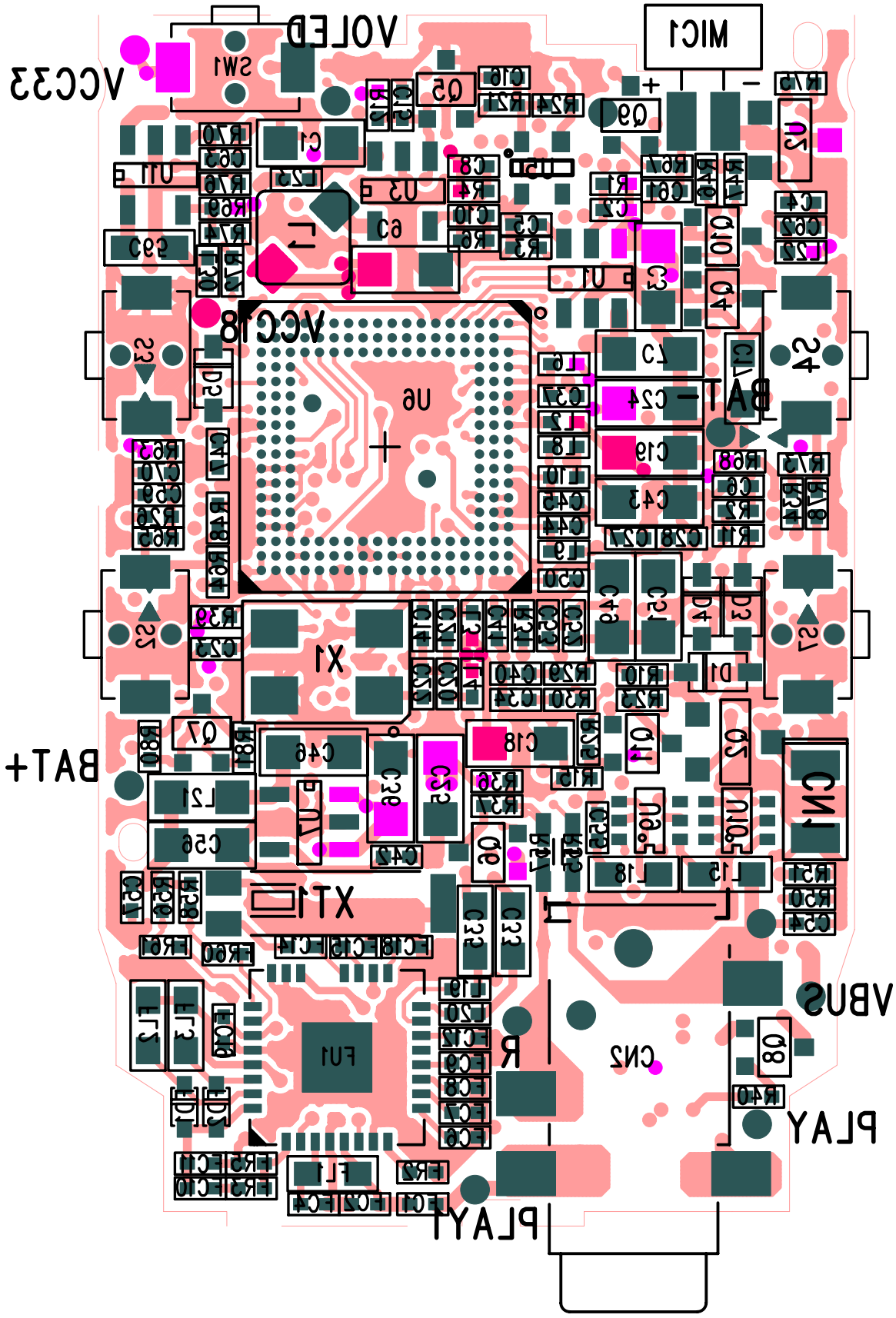


PCB另一面不要敷铜！
不要靠近数字、晶振和IF部分（MPX等信号）
地线要干净。

Title	mini2x5P-1	
Size B	Document Number <Doc>	Rev 1.0
Date: Friday, November 04, 2005	Sheet 4 of 5	



2MINIX5P-1.pcb - Thu Mar 09 09:46:29 2006


VOLED
VCC33
MIC1
VCC18
U6
BAT
X1
XT1
BAT+
CN1
VBUS
PLAY
CN2
PLAY1

MAINBOARD

MATERIAL CODE	MATERIAL NAME	SPECIFICATIONS	LOCATION
0090324	SMD RESISTOR	1/16W 0O± 5% 0402	FL4,L30,R52,R53,R74,R72
0090326	SMD RESISTOR	1/16W 10O± 5% 0402	FR1,L10,R29,R30,R31
0090644	SMD RESISTOR	1/16W 18O± 5% 0402	R55,R57,L8
0090447	SMD RESISTOR	1/16W 22O± 5% 0402	FR7,FR4
0090339	SMD RESISTOR	1/16W 100O± 5% 0402	R9
0090346	SMD RESISTOR	1/16W 220O± 5% 0402	R73
0090349	SMD RESISTOR	1/16W 300O ± 5% 0402	R78
0090355	SMD RESISTOR	1/16W 510O± 5% 0402	R65
0090359	SMD RESISTOR	1/16W 750O± 5% 0402	R64
0090362	SMD RESISTOR	1/16W 1K± 5% 0402	R8,R40,R50,R51,R63,R71
0090369	SMD RESISTOR	1/16W 2.2K± 5% 0402	R67
0090377	SMD RESISTOR	1/16W 4.7K± 5% 0402	R46,R47
0090385	SMD RESISTOR	1/16W 10K± 5% 0402	FR3,R10,R13,R25,R56,R58,L14,R33
0090390	SMD RESISTOR	1/16W 18K± 5% 0402	FR2
0090396	SMD RESISTOR	1/16W 33K± 5% 0402	FR6,R21,R26,R34
0090400	SMD RESISTOR	1/16W 47K± 5% 0402	R19,R17
0090404	SMD RESISTOR	1/16W 68K± 5% 0402	R2
0090408	SMD RESISTOR	1/16W 100K± 5% 0402	FR5,R11,R12,R14,R15,R24,R38,R39,R41,R48,R59,R62,R68,R69
0090425	SMD RESISTOR	1/16W 470K± 5% 0402	C15,R45
0090433	SMD RESISTOR	1/16W 1MO± 5% 0402	R35,R32,R43
0090436	SMD RESISTOR	1/16W 2MO± 5% 0402	R22,R66
0090443	SMD RESISTOR	1/16W 5.6MO± 5% 0402	R16
0090639	PRECISION SMD RESISTOR	1/16W 1.5K± 1% 0402	R37
0090672	PRECISION SMD RESISTOR	1/16W 20K ± 1% 0402	R3
0090671	PRECISION SMD RESISTOR	1/16W 33K ± 1% 0402	R1
0090485	PRECISION SMD RESISTOR	1/16W 27K± 1% 0402	R76
0090509	PRECISION SMD RESISTOR	1/16W 100K± 1% 0402	R27
0090681	PRECISION SMD RESISTOR	1/16W 120K± 1% 0402	R6,R20,R23
0090645	PRECISION SMD RESISTOR	1/16W 180K± 1% 0402	R28,R77
0090682	PRECISION SMD RESISTOR	1/16W 300K± 1% 0402	R4
0090687	PRECISION SMD RESISTOR	1/16W 240K± 1% 0402	R70
0090667	PRECISION SMD RESISTOR	1/16W 510K ± 1% 0402	R18

1030029	SMD PRESS SENSITIVITY RESISTOR	SFI0402-050E100NP	C4,C68,C70,C71,C72,C73,C76,C77,C79
0310416	SMD CAPACITOR	50V 22P± 5% NPO 0402	C2,C22,C23,C63,FC15
0310418	SMD CAPACITOR	50V 27P± 5% NPO 0402	FC2
0310424	SMD CAPACITOR	50V 47P± 5% NPO 0402	C8,FC4
0310432	SMD CAPACITOR	50V 101± 5% NPO 0402	C10,FC1
0310704	SMD CAPACITOR	25V 102± 10% X7R 0402	C6,C53,FC14,FC18
0310705	SMD CAPACITOR	25V 332± 10% X7R 0402	C27,C28
0310706	SMD CAPACITOR	25V 472± 10% X7R 0402	FC3
0310453	SMD CAPACITOR	25V 103± 10% X7R 0402	C5,C60,FC11
0310455	SMD CAPACITOR	16V 223± 10% X7R 0402	FC16,FC17
0310710	SMD CAPACITOR	16V 333± 10% X5R 0402	FC12,FC9
0310711	SMD CAPACITOR	10V 393± 10% X5R 0402	FC10
0310712	SMD CAPACITOR	10V 473± 10% X5R 0402	FC6,FC7,FC8
0310480	SMD CAPACITOR	10V 104± 10% X5R 0402	C20,C21,C26,C30,C34,C37,C38,C39,C40,C41,C42,C44,C45,C47,C48,C50,C52,C57,C59,C62,C66,C61,C69,FC13,FC19
0310753	SMD CAPACITOR	16V 104 ± 20% X5R 0402	C67
0310703	SMD CAPACITOR	10V 105± 20% Y5V 0603	C11,C13,C33,C35
0310674	SMD CAPACITOR	6.3V 475± 10% X5R 0603	C17
0310717	SMD CAPACITOR	6.3V 475± 20% X5R 0603	C65,FC5
0310486	SMD CAPACITOR	6.3V 106± 20% X5R 0805	C1,C3,C7,C9,C36,C46
0310701	SMD CAPACITOR	10V 106± 20% Y5V 0805	C18,C19,C24,C25,C29,C43,C49,C51,C56,C58
0310389	SMD CAPACITOR	10V 106 +80%-20% Y5V 0805	C18,C19,C24,C25,C29,C43,C49,C51,C56,C58
0310752	SMD CAPACITOR	25V 106± 10% X5R 1206	C64
0310736	SMD TANTALUM CAPACITOR	4V 220uF± 20% 3528(B)	C31,C32
0390142	SMD MAGNETIC BEADS	FCM1608-601T02	L15,L17,L18
0090001	SMD RESISTOR	1/16W 0O ± 5% 0603	L16
0390388	SMD MAGNETIC BEADS	600O/100MHZ± 25% 1005	L2,L3,L4,L6,L7,L9,L11,L12,L13,L22,L23,L19, L20, L25,FR60,FR61
0390044	SMD INDUCTOR	10UH ± 10% 2012	L21
0390397	SMD CORES INDUCTOR	10uH ± 20% CDRH2D11/HP	L1
0390221	SMD COIL THREAD INDUCTOR	33nH± 2% 1608	FL2,FL3
0390398	SMD COIL THREAD INDUCTOR	120nH± 2% 1608	FL1
0390384	SMD CORES INDUCTOR	4.7uH ± 20% CDRH3D16-4R7	L24
0390387	SMD CORES INDUCTOR	4.7uH ± 30% CDRH3D16/HP	L24
1090080	ESD ELEMENT	RCLAMP0504F SC70-6L	D7

0700154	SMD TRIODE	1N4148WS SOD-323	D3,D4,D5,D9
0680077	SMD SCHOTTKY DIODE	MBR0520 SOD123	D8
0680074	SMD SCHOTTKY DIODE	BAT43WS SOD-323	D1
0700115	SMD TRANSFIGURATION DIODE	BB202	FD1,FD2
0780298	SMD TRIODE	MMST3904 SOD-323	Q9,Q10
0780299	SMD TRIODE	SS8050LT SOD-323	Q4,Q11
0780293	SMD TRIODE	MMST3906 SOD-323	Q3,Q5,Q6
0780300	SMD TRIODE	SS8550LT SOD-323	Q8
0790041	SMD FIELD EFFECT TRANSISTOR	SI2305DS SOT-23	Q2
0790068	SMD FIELD EFFECT TRANSISTOR	SI1912 SOT363	U9,U10
0790070	SMD FIELD EFFECT TRANSISTOR	NTJD4401N SOT363	U9,U10
0882475	IC	TPS79301 SOT23-6	U1
0882668	IC	TPS79333DBVR SOT23-5	U1
0882388	IC	TEA5767HN HVQFN	FU1
0882481	IC	G690L263T71 SOT23-3	U2
0882476	IC	TPS62200 SOT23-5	U3
0882851	IC	ISL6294IRZ DFN	U4
0882524	IC	S80829CNNB SC-82	U5
0882403	IC	PNX0101ET/N302 TFBGA	U6
0882480	IC	PQ1X281M2ZP SOT23-5	U7
0882629	IC	TPS793285DBVR SOT23-5	U7
0882324	IC	K9F2G08U0M-YCBO TSOP	U8
0882576	IC	K9F2G08U0M-PIB0 TSOP	U8
0882565	IC	LM2703 SOT-23-5	U11
1340099	SMD LIGHT TOUCH SWITCH	SKRELGE010	SW1,S2,S3,S4,S7
1340128	LIGHT TOUCH SWITCH	TS-1186	S1,S5,S6
1140082	MICROPHONE	44dB ± 3dB 4× 1.2 WITH NEEDLE	MIC1
1980074	SMD EARPHONE SOCKET	2SJ-0386-001	CN2
1940282	CABLE SOCKET	21P 0.5mm SMD,SUBMIT MEET WITH CLASP	CN5
1632822	PCB	2MINIX5P-1	
0960279	SMD CRYSTAL OSCILLATOR	32.768KHz ± 20ppm SSPT6 12.5P	XT1
0960284	SMD CRYSTAL OSCILLATOR	12MHz± 30ppm 5032/4 20P	X1

BATTERY PROTECT BOARD

MATERIAL CODE	MATERIAL NAME	SPECIFICATIONS	LOCATION
0090223	SMD RESISTOR	1/16W 2K ± 5%	R2
0090011	SMD RESISTOR	1/16W 470O ± 5% 0603	R1
0310207	SMD CAPACITOR	50V104 ± 20% 0603	C1,C2,C3
0310543	SMD CAPACITOR	50V 104 ± 10% X7R 0603	C1,C2,C3
0882570	IC	S-8261AANMD-G2N-T2 SOT23-6	U1
0790065	SMD FIELD EFFECT TRANSISTOR	ECH8601 TSSOP	U2
0790090	SMDFIELD EFFECT TRANSISTOR	ECH8601R TSSOP	U2
1632263	PCB	EX9-0	

LITHIUM BATTERY CORD UNIT

MATERIAL CODE	MATERIAL NAME	SPECIFICATIONS
1510166	RECHARGEABLE BATTERY	200mAH 3.7V LITHIUM 26× 24× 4.4
5446654	PCB SEMI-FINISHED PRODUCT	SEX9-0 X9
2110508	LEAD	28# 30mm RED
2110167	LEAD	28# 30mm BLACK

ASSEMBLY

3004186	SURFACE CASING	X5 MINIBLACK
3012082	BOTTOM CASING	X5 MINI RED
3071861	NEXT/PREV BUTTON	MINI X5 RED
3071862	PLAY BUTTON	MINI X5 RED
3071863	VOLUME（+）	MINI X5 RED
3071864	VOLUME（-）	MINI X5 RED
3071865	MANU BUTTON	MINI X5 RED
3071866	RECORD BUTTON	MINI X5 RED
3071919	GLASS	F-620A(TH)RED WHITE
4000660	SELF-TAPPING SCREW	CB1.4× 5 WHITE NICKEL
5234448	PORON SOFT SPACER	25× 1.5× 0.5 SINGLE-FACED WITH GLUE IN REAR SIDE
5234449	PORON SOFT SPACER	21× 1× 0.2 SINGLE-FACED WITH GLUE IN REAR SIDE
5234450	PORON SOFT SPACER	15× 1.5× 0.2 SINGLE-FACED WITH GLUE IN REAR SIDE
5234542	PORON SOFT SPACER	24× 17× 0.5 DOUBLE-FACED WITH GLUE IN REAR SIDE
5234543	SOFT SPONGE SPACER	13× 13× 1.5 DOUBLE-FACED,SOFT
5234541	PVC PIECE	28× 22× 0.2 DOUBLE-FACED WITH GLUE IN REAR SIDE
1632699	FPC SOCKET	S4MINIX5P-A
1210248	OLED MODULE	RGS080960390W006
5462157	LITHIUM BATTEMETAL OXIDE FILM RESISTOR CORD UNIT	200mAH 3.7V LINE LONG30mm
5447795	PCB SEMI-FINISHED PRODUCT	2MINIX5P-1 X5 MINI(RU)

PACKAGE MATERIAL

1150063	EARPHONE	OPPO/Sennheiser MX400 WITH EARPHONE SET
1150067	EARPHONE	Sennheiser MX400II WITH EARPHONE SET
5471439	CHARGER	@5V 300mA WITH F3.5 EARPHONE HEAD,RUSSIA CE
2180083	USB CORD	1.2m DOUBLE HEAD(USB MALE/F3.5)2#
5070981	GLUE BAG FOR ENVIRONMENTAL PROTECTION (WITH HOLE)	145× 65× 0.05 PE

5234457	PROTECT FILM	MINI X5
5234662	BOLT SPOUT PLUG	F2.65× 3 RED RUBBER
5080026	GLUE SET	X5 MINI(SILICA GEL)MILK
5070954	GLUE BAG FOR ENVIRONMENTAL PROTECTION (WITH HOLE)	95× 65× 0.05 PE
5456532	MP3 PLAYER	X5 MINI(RU)(256M)RED BLACK
5210465	WARRANTY CARD	MP3 120× 80 RUSSIA
5194440	USER MANUAL	X5 mini(RU)ENGLISH/RUSSIA
2400253	SUITABLE SOFTWARE DISC (8cm)	MP3(RU) philips
5180806	SEALING LABELL OF GIFT BOX	X17(RU) F25
5013410	INNER PAPER CSE	154× 61× 125 MINI X5
5013592	GIFT BOX	X5 MINI(RU)
5002904	CARTON BOX	33× 32.5× 15.5cm3X5 MINI(RU)
5002906	CARTON BOX	63× 33.5× 34.5cm3X5 MINI(RU)
5156120	MODEL LABELL FOR GIFE BOX	X5 MINI(RU) REDBLACK
5156124	PAPER BOX WITH MODEL STICKER	X5 MINI(RU) REDBLACK65× 38
5142663	SN LABELL	MP3(RUSSIA BBK) WITH BAR CODE NUMBER
5180754	COLOR LABELL	RED BLACK (COLOR) F12
5071141	LIFTING ROPE	MP3(OPPO)GIRTH 80cm DOUBLE HANG HEAD CLASSICALITY BLACK
SN LASEL		
5142067	SN LABELL	RUSSIA WITHOUT BAR CODE NUMBER
SUPPLEMENT MODULE		
5233646	RESIST HIGH TEMPERATURE INSULATING TAPE	LENGTH:33m WIDTH:6mm
5120754	GLUEWATER	LOCTITE 3517
5120761	AID WELDING GREASE	MC40