

HMS30C7202

Highly-integrated MPU
(ARM Based 32-Bit Microprocessor)

Datasheet

Version 1.4

www.DataSheet.in

Hynix Semiconductor Inc.



HMS30C7202

www.DataSheet.in

Copyright. 2002 Hynix Semiconductor Inc.

ALL RIGHTS RESERVED. No part of this publication may be copied in any form, by photocopy, microfilm, retrieval system, or by any other means now known or hereafter invented without the prior written permission of Hynix Semiconductor Inc.

Hynix Semiconductor Inc.

#1, Hyangjeong-dong, Heungduk-gu, Cheongju-si, Chungcheonbuk-do, Republic of Korea

Homepage: www.hynix.com

Technical Support Homepage: www.softonchip.com

H.Q. of Hynix Semiconductor Inc. Marketing Site

Telephone: 82-(0)43-270-4070

Facsimile: 82-(0)43-270-4099

Telephone: 82-(0)43-270-4085

Facsimile: 82-(0)43-270-4099

Sales in Korea

Telephone: 82-(0)2-3459-3843

Facsimile: 82-(0)2-3459-3945

World Wide Sales Network**U.S.A.**

Telephone: 1-408-232-8757

Facsimile: 1-408-232-8135

Taiwan

Telephone: 886-(0)2-2500-8357

Facsimile: 886-(0)2-2509-8977

Hong Kong

Telephone: 852-2971-1640

Facsimile: 852-2971-1622

HMS30C7202 Datasheet, ver1.4

May 12, 2004

Proprietary Notice

Hynix logo is trademark of Hynix Semiconductor Inc.

Neither the whole nor any part of the information contained in, or the product described in, this document may be adapted or reproduced in any material from excepts with the prior permission of the copyright holder.

The product described in this document is subject to continuous developments and improvements. All particulars of the product and its use contained in this document are given by Hynix in good faith. However, all warranties implied or expressed, including but not limited to implied warranties or merchantability, or fitness for purpose, are excluded.

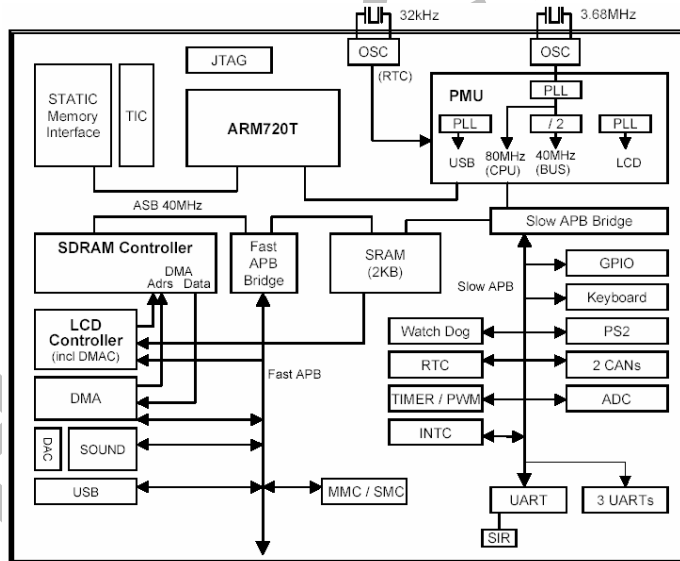
This document is intended only to assist the reader in the use of the product. Hynix Semiconductor Inc. shall not be liable for any loss or damage arising from the use of any information in this document, or any error or omission in such information, or any incorrect use of the product. Hynix Semiconductor Inc. may make changes to specification and product description at any time without notice.

Change Log

| Issue | Date | By | Change |
|-------|------------|-----------|---|
| A-01 | 2002/08/27 | Kisun Kim | The First Draft |
| A-02 | 2002/08/28 | Kisun Kim | PMU Freq. Range / QFP Footprint / BGA Pin Diagram |
| A-03 | 2002/10/01 | Kisun Kim | PKG Soldering Condition / Harry's Update |
| A-04 | 2002/10/05 | Kisun Kim | CAN Interrupt Desc. / Interrupt controller FIQ Desc. |
| A-05 | 2002/10/14 | Kisun Kim | Harry's Review (Chapter 1~5) |
| A-06 | 2002/12/28 | Kisun Kim | Harry's Review (Chapter 6~11) |
| A-07 | 2003/01/09 | Kisun Kim | SMI example / SMC / USB |
| A-08 | 2003/02/26 | Injae Koo | DC Electrical Characteristics / RTC / USB |
| A-10 | 2003/03/03 | Injae Koo | The First Release (Version 1.0) |
| A-11 | 2003/03/28 | Injae Koo | LCD / CAN / Overview / DC E. Char. / GPIO (Version 1.1) |
| A-12 | 2003/06/03 | Injae Koo | DMA / DC Char. / GPIO / Delete the I2S (Version 1.2) |
| A-13 | 2003/09/15 | Injae Koo | ERRATA(version 1.0) is incorporated into this version |
| A-14 | 2004/05/12 | Injae Koo | ADC/GPIO/SDRAMC/MMC/LCD/AC Characteristics (SMI) |

FEATURES

- 32-bit ARM7TDMI RISC static CMOS CPU core : Running up to 70 MHz
- 8Kbytes combined instruction/data cache
- Memory management unit
- Supports Little Endian operating system
- 2Kbytes SRAM for internal buffer memory
- On-chip peripherals with individual power-down:
 - Multi-channel DMA
 - 4 Timer Channels with Watch Dog Timer
 - Intelligent Interrupt Controller
 - Memory controller for ROM, Flash, SRAM, SDRAM
 - Power management unit
 - LCD Controller for mono/color STN and TFT LCD
 - Real-time clock (32.768kHz oscillator)
 - Infrared communications (SIR support)
 - 4 UARTs (16C550 compatible)
 - PS/2 External Keyboard / Mouse interface
 - 2 Pulse-Width-Modulated (PWM) interface
 - Matrix Keyboard control interface (8*8)
 - GPIO
 - MMC / SMC Card interface
 - 2 Controller Area Network (CAN)
 - USB (slave)
 - On-chip ADC and interface module (Battery Check, Audio In, Touch Panel)
 - On-chip DAC and interface module (8 Bit Stereo Audio Output)
 - 3 PLLs


Figure A. Functional Block Diagram

- JTAG debug interface and boundary scan
- 0.25um Low Power CMOS Process
- 2.5V internal / 3.3V IO supply voltage
- 256-pin MQFP / FBGA package
- Low power consumption

OVERVIEW

The HMS30C7202 is a highly integrated low power microprocessor for personal digital assistants, and other applications described below. The device incorporates an ARM720T CPU and system interface logic to interface with various types of devices. HMS30C7202 is a highly modular design based on the AMBA bus architecture between CPU and internal modules.

The on-chip peripherals include LCD controller with DMA support for external SDRAM memory, analog functions such as ADC, DAC, and PLLs. Intelligent interrupt controller and internal 2Kbytes SRAM can support an efficient interrupt service execution. The HMS30C7202 also supports voice recording, sound playback and a touch panel interface. UART, USB, PS2 and CAN provide serial communication channels for external systems. The power management features result in very low power consumption. The HMS30C7202 provides an excellent solution for personal digital assistants (PDAs), and data terminal running the Microsoft Windows CE operating system. Other applications include smart phones, Internet appliances, telematic systems and embedded computer.

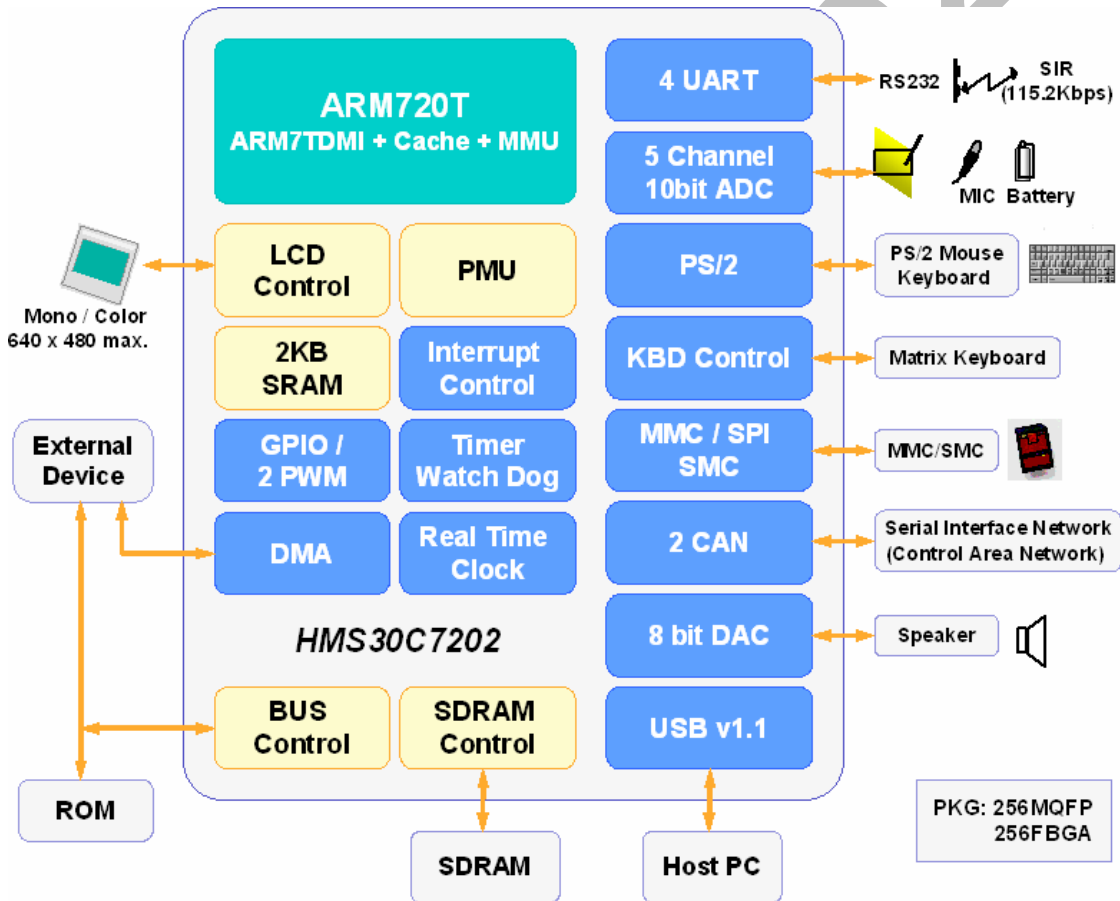


Figure B. System Configuration

TABLE OF CONTENTS

| | | |
|----------|---|-----------|
| 1 | ARCHITECTURAL OVERVIEW | 9 |
| 1.1 | PROCESSOR..... | 9 |
| 1.2 | VIDEO | 9 |
| 1.3 | MEMORY | 9 |
| 1.4 | INTERNAL BUS STRUCTURE..... | 9 |
| 1.4.1 | ASB..... | 9 |
| 1.4.2 | Video bus | 9 |
| 1.4.3 | APB | 9 |
| 1.5 | SDRAM CONTROLLER..... | 10 |
| 1.6 | PERIPHERAL DMA..... | 10 |
| 1.6.1 | Overview..... | 10 |
| 1.6.2 | Transfer sizes..... | 10 |
| 1.6.3 | Fly-by | 10 |
| 1.6.4 | Timing..... | 11 |
| 1.6.5 | Sound output..... | 11 |
| 1.7 | PERIPHERALS | 11 |
| 1.8 | POWER MANAGEMENT | 11 |
| 1.8.1 | Clock gating | 12 |
| 1.8.2 | PMU..... | 12 |
| 1.9 | TEST AND DEBUG | 12 |
| 2 | PIN DESCRIPTION | 13 |
| 2.1 | 256-PIN DIAGRAM..... | 13 |
| 2.1.1 | MQFP Type | 13 |
| 2.1.2 | FBGA Type | 15 |
| 2.2 | PIN DESCRIPTIONS | 17 |
| 2.2.1 | External Signal Functions | 17 |
| 2.2.2 | Multiple Function Pins | 20 |
| 2.2.2.1 | PORT A | 20 |
| 2.2.2.2 | PORT B | 20 |
| 2.2.2.3 | PORT C | 21 |
| 2.2.2.4 | PORT D | 21 |
| 2.2.2.5 | PORT E..... | 22 |
| 2.2.2.6 | USB Transceiver Test & Analog Test | 23 |
| 2.2.2.7 | DMA..... | 23 |
| 2.2.2.8 | Inverter Chain..... | 23 |
| 3 | ARM720T MACROCELL..... | 24 |
| 3.1 | ARM720T MACROCELL..... | 24 |
| 4 | MEMORY MAP | 25 |
| 5 | PMU & PLL..... | 27 |
| 5.1 | BLOCK FUNCTIONS | 27 |
| 5.2 | POWER MANAGEMENT | 28 |
| 5.2.1 | State Diagram..... | 28 |
| 5.2.2 | Power management states | 28 |
| 5.2.3 | Wake-up Debounce and Interrupt..... | 29 |
| 5.3 | REGISTERS..... | 30 |
| 5.3.1 | PMU Mode Register (PMUMODE) | 30 |
| 5.3.2 | PMU ID Register (PMUID)..... | 30 |
| 5.3.3 | PMU Reset /PLL Status Register (PMUSTAT)..... | 30 |
| 5.3.4 | PMU Clock Control Register (PMUCLK) | 32 |
| 5.3.5 | PMU Debounce Counter Test Register (PMUDBCT)..... | 33 |
| 5.3.6 | PMU PLL Test Register (PMUPLLTR)..... | 33 |
| 5.4 | TIMINGS | 34 |

| | | |
|----------|---|-----------|
| 5.4.1 | Reset Sequences of Power On Reset..... | 34 |
| 5.4.2 | Software Generated Warm Reset | 35 |
| 5.4.3 | An Externally generated Warm Reset | 35 |
| 6 | SDRAM CONTROLLER..... | 37 |
| 6.1 | SUPPORTED MEMORY DEVICES | 37 |
| 6.2 | REGISTERS | 38 |
| 6.2.1 | SDRAM Controller Configuration Register (SDCON)..... | 38 |
| 6.2.2 | SDRAM Controller Refresh Timer Register (SDREF)..... | 40 |
| 6.2.3 | SDRAM Controller Write buffer flush timer Register (SDWBF)..... | 40 |
| 6.2.4 | SDRAM Controller Wait Driver Register (SDWAIT)..... | 40 |
| 6.3 | POWER-UP INITIALIZATION OF THE SDRAMs..... | 40 |
| 6.4 | SDRAM MEMORY MAP | 41 |
| 6.5 | AMBA ACCESSES AND ARBITRATION | 42 |
| 6.6 | MERGING WRITE BUFFER | 42 |
| 7 | STATIC MEMORY INTERFACE..... | 44 |
| 7.1 | EXTERNAL SIGNALS..... | 44 |
| 7.2 | FUNCTIONAL DESCRIPTION | 44 |
| 7.2.1 | Memory bank select..... | 44 |
| 7.2.2 | Access sequencing | 44 |
| 7.2.3 | Wait states generation | 45 |
| 7.2.4 | Burst read control..... | 45 |
| 7.2.5 | Byte lane write control | 45 |
| 7.3 | REGISTERS | 46 |
| 7.3.1 | MEM Configuration Register | 46 |
| 7.4 | EXAMPLES OF THE SMI READ, WRITE WAIT TIMING DIAGRAM | 47 |
| 7.4.1 | Read normal wait (Non-Sequential mode)..... | 47 |
| 7.4.2 | Read normal wait (Sequential mode) | 48 |
| 7.4.3 | Read burst wait (Sequential mode)..... | 49 |
| 7.4.4 | Write normal wait (Sequential mode)..... | 50 |
| 7.5 | INTERNAL SRAM..... | 51 |
| 7.5.1 | Remapping Enable Register | 51 |
| 7.5.2 | Remap Source Address Register | 51 |
| 8 | LCD CONTROLLER..... | 52 |
| 8.1 | VIDEO OPERATION | 52 |
| 8.1.1 | LCD datapath..... | 53 |
| 8.1.1.1 | Palette RAM & 16bpp mode | 53 |
| 8.1.2 | Color/Grayscale Dithering..... | 55 |
| 8.1.3 | How to order the bit on LD[7:0] output..... | 55 |
| 8.1.4 | TFT mode | 56 |
| 8.2 | REGISTERS | 56 |
| 8.2.1 | LCD Power Control | 56 |
| 8.2.2 | LCD Controller Status/Mask and Interrupt Registers | 57 |
| 8.2.3 | LCD DMA Base Address Register..... | 58 |
| 8.2.4 | LCD DMA Channel Current Address Register | 58 |
| 8.2.5 | LCD Timing 0 Register..... | 58 |
| 8.2.6 | LCD Timing 1 Register..... | 59 |
| 8.2.7 | LCD Timing 2 Register..... | 60 |
| 8.2.8 | LCD Test Register..... | 61 |
| 8.2.9 | Grayscale Test Registers..... | 61 |
| 8.2.10 | LCD Palette registers..... | 62 |
| 8.3 | TIMINGS | 63 |
| 9 | FAST AMBA PERIPHERALS..... | 64 |
| 9.1 | DMA CONTROLLER..... | 64 |
| 9.1.1 | External Signals | 64 |
| 9.1.2 | Registers..... | 64 |
| 9.1.2.1 | ADR0 | 65 |

| | | |
|----------|---|----|
| 9.1.2.2 | ASR | 65 |
| 9.1.2.3 | TNR0 | 65 |
| 9.1.2.4 | TSR | 65 |
| 9.1.2.5 | CCR0 | 65 |
| 9.1.2.6 | ADR1 | 66 |
| 9.1.2.7 | TNR1 | 66 |
| 9.1.2.8 | CCR1 | 66 |
| 9.1.2.9 | ADR2 | 66 |
| 9.1.2.10 | TNR2 | 66 |
| 9.1.2.11 | CCR2 | 67 |
| 9.1.2.12 | FLAGR | 67 |
| 9.1.2.13 | DMAOR | 67 |
| 9.1.3 | <i>DMAC operation</i> | 68 |
| 9.2 | MMC/ SPI CONTROLLER | 69 |
| 9.2.1 | <i>External Signals</i> | 69 |
| 9.2.2 | <i>Registers (SPI Mode)</i> | 69 |
| 9.2.2.1 | SPIMMC Control Register (SPICR) | 69 |
| 9.2.2.2 | SPIMMC Status Register (SPISR) | 70 |
| 9.2.2.3 | SPIMMC XCH Counter Register (XHCNT) | 70 |
| 9.2.2.4 | SPIMMC TX Data Buffer Register (TXBUFF) | 70 |
| 9.2.2.5 | SPIMMC RX Data Buffer Register (RXBUFF) | 70 |
| 9.2.2.6 | SPIMMC Reset Register (ResetReg) | 71 |
| 9.2.3 | <i>Timings</i> | 71 |
| 9.2.4 | <i>SPI Operation for MMC</i> | 72 |
| 9.2.5 | <i>Multimedia Card Host Controller</i> | 73 |
| 9.2.6 | <i>Registers</i> | 73 |
| 9.2.6.1 | MMC Mode Register | 73 |
| 9.2.6.2 | MMC Operation Register | 74 |
| 9.2.6.3 | MMC Status Register | 74 |
| 9.2.6.4 | MMC Interrupt Enable Register | 75 |
| 9.2.6.5 | MMC Block Size Register | 76 |
| 9.2.6.6 | MMC Block Number Register | 76 |
| 9.2.6.7 | MMC Time Period Register | 76 |
| 9.2.6.8 | MMC Command Buffer Register | 76 |
| 9.2.6.9 | MMC Argument Buffer Register | 76 |
| 9.2.6.10 | MMC Response Buffer Register | 77 |
| 9.2.6.11 | MMC Data Buffer Register | 77 |
| 9.2.6.12 | MMC Ready Timeout Register | 77 |
| 9.2.7 | <i>Basic Operation in MMC Mode</i> | 77 |
| 9.2.7.1 | Write Operation | 78 |
| 9.2.7.2 | Read Operation | 78 |
| 9.3 | SMC CONTROLLER | 79 |
| 9.3.1 | <i>External Signals</i> | 79 |
| 9.3.2 | <i>Registers</i> | 79 |
| 9.3.2.1 | SMC Command Register (SMCCMD) | 79 |
| 9.3.2.2 | SMC Address Register (SMCADR) | 80 |
| 9.3.2.3 | SMC Data Write Register (SMCDATW) | 80 |
| 9.3.2.4 | SMC Data Read Register (SMCDATR) | 81 |
| 9.3.2.5 | SMC Configuration Register (SMCCONF) | 81 |
| 9.3.2.6 | SMC Timing Parameter Register (SMCTIME) | 82 |
| 9.3.2.7 | SMC Status Register (SMCSTAT) | 82 |
| 9.4 | SOUND INTERFACE | 84 |
| 9.4.1 | <i>External Signals</i> | 84 |
| 9.4.2 | <i>Registers</i> | 84 |
| 9.4.2.1 | SCONT | 84 |
| 9.4.2.2 | SDADR | 85 |
| 9.5 | USB SLAVE INTERFACE | 86 |
| 9.5.1 | <i>Block Diagram</i> | 87 |
| 9.5.2 | <i>Theory of Operation</i> | 87 |
| 9.5.3 | <i>Endpoint FIFOs (Rx, Tx)</i> | 90 |

| | | |
|-----------|--|-----------|
| 9.5.4 | External Signals | 90 |
| 9.5.5 | Registers..... | 90 |
| 9.5.5.1 | GCTRL..... | 90 |
| 9.5.5.2 | EPCTRL..... | 90 |
| 9.5.5.3 | INTMASK..... | 91 |
| 9.5.5.4 | INTSTAT..... | 91 |
| 9.5.5.5 | PWR..... | 92 |
| 9.5.5.6 | DEVID..... | 92 |
| 9.5.5.7 | DEVCLASS..... | 92 |
| 9.5.5.8 | INTCLASS..... | 92 |
| 9.5.5.9 | SETUP0 / SETUP1..... | 94 |
| 9.5.5.10 | ENDP0RD..... | 94 |
| 9.5.5.11 | ENDP0WT..... | 94 |
| 9.5.5.12 | ENDP1RD..... | 94 |
| 9.5.5.13 | ENDP2WT..... | 94 |
| 10 | SLOW AMBA PERIPHERALS..... | 95 |
| 10.1 | ADC INTERFACE CONTROLLER..... | 95 |
| 10.1.1 | External Signals..... | 95 |
| 10.1.2 | Registers..... | 95 |
| 10.1.2.1 | ADC Control Register (ADCCR)..... | 96 |
| 10.1.2.2 | ADC Touch Panel Control Register (ADCTPCR)..... | 96 |
| 10.1.2.3 | ADC Battery check Control Register (ADCBACR)..... | 97 |
| 10.1.2.4 | ADC Sound Control Register (ADCSDCR)..... | 97 |
| 10.1.2.5 | ADC Interrupt Status Register (ADCISR)..... | 97 |
| 10.1.2.6 | ADC Tip Down Control Status Register (ADCTDCSR)..... | 98 |
| 10.1.2.7 | ADC Direct Control Register (ADCDIRCR)..... | 98 |
| 10.1.2.8 | ADC Direct Data Read Register (ADCDIRDATA)..... | 98 |
| 10.1.2.9 | ADC 1 ST Touch Panel Data register..... | 99 |
| 10.1.2.10 | ADC 2 ND Touch Panel Data Register..... | 99 |
| 10.1.2.11 | ADC Main Battery Data Register (ADCMBDATA)..... | 100 |
| 10.1.2.12 | ADC Backup Battery Data Register (ADCBBDATA)..... | 100 |
| 10.1.2.13 | ADC Sound Data Register (ADCSDATA0 – ADCSDATA7)..... | 100 |
| 10.2 | CAN INTERFACE..... | 102 |
| 10.2.1 | Block Diagram..... | 103 |
| 10.2.2 | Register Map..... | 103 |
| 10.2.3 | Registers..... | 104 |
| 10.2.3.1 | CAN Control Register..... | 104 |
| 10.2.3.2 | CAN Status Register..... | 105 |
| 10.2.3.3 | CAN Error Counting Register..... | 105 |
| 10.2.3.4 | CAN Bit Timing Register..... | 106 |
| 10.2.3.5 | CAN Interrupt Register..... | 106 |
| 10.2.3.6 | CAN Test Register..... | 106 |
| 10.2.3.7 | CAN BRP Extension Register..... | 107 |
| 10.2.3.8 | CAN Enable Register..... | 107 |
| 10.2.3.9 | Interface X Command Request Register..... | 107 |
| 10.2.3.10 | Interface X Command Mask Register..... | 107 |
| 10.2.3.11 | Interface X Mask 1 Register..... | 108 |
| 10.2.3.12 | Interface X Mask 2 Register..... | 108 |
| 10.2.3.13 | Interface X Arbitration 1 Register..... | 109 |
| 10.2.3.14 | Interface X Arbitration 2 Register..... | 109 |
| 10.2.3.15 | Interface X Message Control Register..... | 109 |
| 10.2.3.16 | Interface X Data A1 Register..... | 110 |
| 10.2.3.17 | Interface X Data A2 Register..... | 110 |
| 10.2.3.18 | Interface X Data B1 Register..... | 110 |
| 10.2.3.19 | Interface X Data B2 Register..... | 111 |
| 10.2.3.20 | Transmission Request 1 Register..... | 111 |
| 10.2.3.21 | Transmission Request 2 Register..... | 111 |
| 10.2.3.22 | New Data 1 Register..... | 111 |
| 10.2.3.23 | New Data 2 Register..... | 111 |

| | | |
|-----------|--|-----|
| 10.2.3.24 | Interrupt Pending 1 Register | 112 |
| 10.2.3.25 | Interrupt Pending 2 Register | 112 |
| 10.2.3.26 | Message Valid 1 Register..... | 112 |
| 10.2.3.27 | Message Valid 2 Register..... | 112 |
| 10.3 | GPIO | 114 |
| 10.3.1 | External Signals..... | 114 |
| 10.3.2 | Registers..... | 114 |
| 10.3.2.1 | ADATA | 115 |
| 10.3.2.2 | ADIR | 115 |
| 10.3.2.3 | AMASK..... | 116 |
| 10.3.2.4 | ASTAT | 116 |
| 10.3.2.5 | AEDGE | 116 |
| 10.3.2.6 | ACLR | 116 |
| 10.3.2.7 | APOL..... | 116 |
| 10.3.2.8 | GPIO PORT A Enable Register | 116 |
| 10.3.2.9 | BDATA | 117 |
| 10.3.2.10 | BDIR | 117 |
| 10.3.2.11 | BMASK..... | 117 |
| 10.3.2.12 | BSTAT | 117 |
| 10.3.2.13 | BEDGE..... | 117 |
| 10.3.2.14 | BCLK | 117 |
| 10.3.2.15 | BPOL..... | 117 |
| 10.3.2.16 | GPIO PORT B Enable Register | 117 |
| 10.3.2.17 | CDATA | 117 |
| 10.3.2.18 | CDIR | 117 |
| 10.3.2.19 | CMASK..... | 118 |
| 10.3.2.20 | CBSTAT | 118 |
| 10.3.2.21 | CEEDGE..... | 118 |
| 10.3.2.22 | CCLK | 118 |
| 10.3.2.23 | CPOL..... | 118 |
| 10.3.2.24 | GPIO PORT C Enable Register | 118 |
| 10.3.2.25 | DDATA..... | 118 |
| 10.3.2.26 | DDIR..... | 118 |
| 10.3.2.27 | DMASK..... | 118 |
| 10.3.2.28 | DBSTAT | 118 |
| 10.3.2.29 | DEEDGE | 118 |
| 10.3.2.30 | DCLK | 118 |
| 10.3.2.31 | DPOL..... | 118 |
| 10.3.2.32 | GPIO PORT D Enable Register | 118 |
| 10.3.2.33 | EDATA..... | 119 |
| 10.3.2.34 | EDIR..... | 119 |
| 10.3.2.35 | EMASK..... | 119 |
| 10.3.2.36 | EBSTAT..... | 119 |
| 10.3.2.37 | EEDGE..... | 119 |
| 10.3.2.38 | ECLK..... | 119 |
| 10.3.2.39 | EPOL..... | 119 |
| 10.3.2.40 | GPIO PORT E Enable Register | 119 |
| 10.3.2.41 | Tic Test mode Register(TICTMDR)..... | 119 |
| 10.3.2.42 | PORTA Multi-function Select register(AMULSEL)..... | 120 |
| 10.3.2.43 | SWAP Pin Configuration Register(SWAP)..... | 120 |
| 10.3.3 | GPIO Interrupt..... | 120 |
| 10.3.4 | GPIO Rise/Fall Time..... | 121 |
| 10.4 | INTERRUPT CONTROLLER..... | 122 |
| 10.4.1 | Block diagram | 122 |
| 10.4.2 | Registers..... | 122 |
| 10.4.2.1 | Interrupt Enable Register (IER)..... | 123 |
| 10.4.2.2 | Interrupt Status Register (ISR)..... | 124 |
| 10.4.2.3 | IRQ Vector Register (IVR) | 125 |
| 10.4.2.4 | Source Vector Register (SVR0 to SVR30)..... | 125 |
| 10.4.2.5 | Interrupt ID Register (IDR)..... | 125 |

| | | |
|-----------|--|-----|
| 10.4.2.6 | Priority Set Register (PSR0 to PSR7)..... | 125 |
| 10.5 | MATRIX KEYBOARD INTERFACE CONTROLLER | 127 |
| 10.5.1 | External Signals | 127 |
| 10.5.2 | Registers..... | 127 |
| 10.5.2.1 | Keyboard Configuration Register (KBCR) | 128 |
| 10.5.2.2 | Keyboard Scanout Register(KBSC) | 128 |
| 10.5.2.3 | Keyboard Test Register (KBTR) | 129 |
| 10.5.2.4 | Keyboard Value Register (KVR0)..... | 129 |
| 10.5.2.5 | Keyboard Value Register (KVR1)..... | 129 |
| 10.5.2.6 | Keyboard Status Register (KBSR) | 129 |
| 10.6 | PS/2 INTERFACE CONTROLLER..... | 131 |
| 10.6.1 | External Signals | 131 |
| 10.6.2 | Registers..... | 131 |
| 10.6.2.1 | PSDATA | 131 |
| 10.6.2.2 | PSSTAT | 132 |
| 10.6.2.3 | PSCONF..... | 132 |
| 10.6.2.4 | PSINTR..... | 133 |
| 10.6.2.5 | PSTDLO..... | 133 |
| 10.6.2.6 | PSTPRI..... | 133 |
| 10.6.2.7 | PSTXMT..... | 134 |
| 10.6.2.8 | PSTREC..... | 134 |
| 10.6.2.9 | PSPWDN..... | 135 |
| 10.6.3 | Application Notes | 135 |
| 10.7 | RTC..... | 136 |
| 10.7.1 | External Signals | 137 |
| 10.7.2 | Functional Description..... | 137 |
| 10.7.3 | Registers..... | 137 |
| 10.7.3.1 | RTC Data Register (RTCDR)..... | 137 |
| 10.7.3.2 | RTC Match Register (RTCMR)..... | 138 |
| 10.7.3.3 | RTC Status Register (RTCS)..... | 138 |
| 10.7.3.4 | RTC Control Register (RTCCR)..... | 138 |
| 10.8 | TIMER..... | 139 |
| 10.8.1 | External Signals | 139 |
| 10.8.2 | Registers..... | 139 |
| 10.8.2.1 | Timer [0,1,2] Base Register (T[0,1,2]BASE) | 139 |
| 10.8.2.2 | Timer [0,1,2] Count Register (T[0,1,2]COUNT)..... | 140 |
| 10.8.2.3 | Timer [0,1,2] Control Register (T[0,1,2]CTRL)..... | 140 |
| 10.8.2.4 | Timer Top-level Control Register (TOPCTRL)..... | 140 |
| 10.8.2.5 | Timer Status Register (TOPSTAT) | 141 |
| 10.8.2.6 | Timer Lower 32-bit Count Register of 64-bit Counter (T64LOW) | 141 |
| 10.8.2.7 | Timer Upper 32-bit Count Register of 64-bit Counter (T64HIGH)..... | 141 |
| 10.8.2.8 | Timer 64-bit Counter Control Register (T64CTRL)..... | 141 |
| 10.8.2.9 | Timer 64-bit Counter Test Register (T64TR) | 141 |
| 10.8.2.10 | Timer Lower 32-bit Base Register of 64-bit Counter (T64LBASE)..... | 142 |
| 10.8.2.11 | Timer Upper 32-bit Base Register of 64-bit Counter (T64HBASE)..... | 142 |
| 10.8.2.12 | PWM Channel [0,1] Count Register (P[0,1]COUNT)..... | 142 |
| 10.8.2.13 | PWM Channel [0,1] Width Register (P[0,1]WIDTH) | 143 |
| 10.8.2.14 | PWM Channel [0,1] Period Register (P[0,1]PERIOD) | 143 |
| 10.8.2.15 | PWM Channel [0,1] Control Register (P[0,1]CTRL)..... | 143 |
| 10.8.2.16 | PWM Channel[0,1] Test Register(P[0,1]PWMTR) | 143 |
| 10.9 | UART/SIR..... | 144 |
| 10.9.1 | External Signals | 144 |
| 10.9.2 | Registers..... | 145 |
| 10.9.2.1 | RBR/THR/DLL..... | 146 |
| 10.9.2.2 | IER/DLM | 146 |
| 10.9.2.3 | IIR/FCR..... | 146 |
| 10.9.2.4 | LCR..... | 148 |
| 10.9.2.5 | MCR..... | 149 |
| 10.9.2.6 | LSR | 150 |
| 10.9.2.7 | MSR | 151 |

| | | |
|-----------|---|------------|
| 10.9.2.8 | SCR | 152 |
| 10.9.2.9 | UartEn | 152 |
| 10.9.3 | FIFO Interrupt Mode Operation | 152 |
| 10.10 | WATCHDOG TIMER | 154 |
| 10.10.1 | Watchdog Timer Operation | 154 |
| 10.10.1.1 | The Watchdog Timer Mode | 154 |
| 10.10.1.2 | The Interval Timer Mode | 154 |
| 10.10.1.3 | Timing of setting the overflow flag | 155 |
| 10.10.1.4 | Timing of clearing the overflow flag | 155 |
| 10.10.2 | Registers | 155 |
| 10.10.2.1 | WDT Control Register (WDTCTRL) | 155 |
| 10.10.2.2 | WDT Status Register (WDTSTAT) | 156 |
| 10.10.2.3 | WDT Counter (WDCNT) | 156 |
| 10.10.3 | Examples of Register Setting | 157 |
| 10.10.3.1 | Interval Timer Mode | 157 |
| 10.10.3.2 | Watchdog Timer Mode with Internal Reset Disable | 157 |
| 10.10.3.3 | Watchdog Timer Mode with Manual Reset | 158 |
| 11 | DEBUG AND TEST INTERFACE | 159 |
| 11.1 | OVERVIEW | 159 |
| 11.2 | SOFTWARE DEVELOPMENT DEBUG AND TEST INTERFACE | 159 |
| 11.3 | TEST ACCESS PORT AND BOUNDARY-SCAN | 159 |
| 11.3.1 | Reset | 160 |
| 11.3.2 | Pull up Resistors | 160 |
| 11.3.3 | Instruction Register | 161 |
| 11.3.4 | Public Instructions | 161 |
| 11.3.5 | Test Data Registers | 163 |
| 11.3.6 | Boundary Scan Interface Signals | 164 |
| 11.4 | PRODUCTION TEST FEATURES | 172 |
| 12 | ELECTRICAL CHARACTERISTICS | 173 |
| 12.1 | ABSOLUTE MAXIMUM RATINGS | 173 |
| 12.2 | DC CHARACTERISTICS | 174 |
| 12.3 | A/D CONVERTER ELECTRICAL CHARACTERISTICS | 175 |
| 12.4 | D/A CONVERTER ELECTRICAL CHARACTERISTICS | 176 |
| 12.5 | AC CHARACTERISTICS | 177 |
| 12.5.1 | Static Memory Interface | 177 |
| 12.5.1.1 | READ Access Timing (Single Mode) | 177 |
| 12.5.1.2 | READ Access Timing (Burst Mode) | 178 |
| 12.5.1.3 | WRITE Access Timing | 179 |
| 12.5.2 | SDRAM Interface | 180 |
| 12.5.3 | LCD Interface | 181 |
| 12.5.4 | UART(Universal Asynchronous Receiver Transmitter) | 183 |
| 12.6 | PACKAGE | 184 |
| 12.6.1 | Recommended Soldering Conditions | 184 |
| 12.6.1.1 | MQFP(Metric Quad Flat Pack) Type | 184 |
| 12.6.1.2 | FBGA(Chip Array Ball Grid Array) Type | 184 |
| 12.6.2 | Pictures of Package Marking | 185 |
| 13 | APPENDIX | 186 |
| 13.1 | DEEP-SLEEP, WAKE-UP ISSUES OF HMS30C7202 PMU | 186 |
| 13.1.1 | Wake-up | 186 |
| 13.1.2 | Deep-sleep | 186 |

LIST OF FIGURES

| | |
|---|-----|
| Figure 5-1 PMU Power Management State Diagram..... | 28 |
| Figure 5-2 PMU Cold Reset Event..... | 34 |
| Figure 5-3 PMU Software Generated Warm Reset..... | 35 |
| Figure 5-4 PMU An Externally Generated Warm Reset..... | 36 |
| Figure 6-1 SDRAM Controller Software Example and Memory Operation Diagram..... | 39 |
| Figure 8-1 Video System Block Diagram..... | 52 |
| Figure 8-2 5:6:5 Combination of 16bpp Data..... | 53 |
| Figure 8-3 Palette RAM Entries for 5:6:5 Combination..... | 54 |
| Figure 8-4 Sample Code for 5:6:5 Palette Generation..... | 54 |
| Figure 8-5 LCD Palette Word Bit Field for STN mode..... | 62 |
| Figure 8-6 LCD Palette Word Bit Field for TFT mode..... | 62 |
| Figure 8-7 Example Mono STN LCD Panel Signal Waveforms..... | 63 |
| Figure 8-8 Example TFT Signal Waveforms, Start of Frame..... | 63 |
| Figure 8-9 Example TFT Signal Waveforms, End of Last Line..... | 63 |
| Figure 9-1 USB Block Diagram..... | 87 |
| Figure 9-2 USB Serial Interface Engine..... | 88 |
| Figure 9-3 USB Device Interface Device Controller..... | 89 |
| Figure 10-1 Typical CAN Network..... | 102 |
| Figure 10-2 Block Diagram of the CAN..... | 103 |
| Figure 10-3 Interrupt controller block diagram..... | 122 |
| Figure 10-4 A flow chart of the keyboard controller..... | 127 |
| Figure 10-5 PS/2 Controller Transmitting Data Timing Diagram..... | 133 |
| Figure 10-6 PS/2 Controller Receiving Data Timing Diagram..... | 134 |
| Figure 10-7 RTC Connection..... | 136 |
| Figure 10-8 RTC Block Diagram..... | 137 |
| Figure 10-9 WDT Operation in the Watchdog Timer mode..... | 154 |
| Figure 10-10 WDT Operation in the Interval Timer mode..... | 155 |
| Figure 10-11 Interrupt Clear in the interval timer mode..... | 157 |
| Figure 10-12 Interrupt Clear in the watchdog timer mode with reset disable..... | 158 |
| Figure 10-13 Interrupt Clear in the watchdog timer mode with manual reset..... | 158 |

LIST OF TABLES

| | |
|---|-----|
| Table 2-1 Pin Signal Type Definition..... | 17 |
| Table 2-2 External Signal Functions..... | 19 |
| Table 4-1 Top-level address map..... | 25 |
| Table 4-2 Peripherals Base Addresses..... | 26 |
| Table 5-1 PMU Register Summary..... | 30 |
| Table 5-2 PMU Bit Settings for a cold Reset Event within PMUSTAT Register..... | 35 |
| Table 5-3 PMU Bit Settings for a Software Generated Warm Reset within PMUSTAT Register..... | 35 |
| Table 5-4 PMU Bit Settings for a Warm Reset within PMUSTAT Register..... | 36 |
| Table 6-1 SDRAM Controller Register Summary..... | 38 |
| Table 6-2 SDRAM Row/Column Address Map..... | 41 |
| Table 6-3 SDRAM Device Selection..... | 42 |
| Table 7-1 Static Memory Controller Register Summary..... | 46 |
| Table 8-1 LCD Color grayscale intensities and modulation rates..... | 55 |
| Table 8-2 How to order the bit on LD[7:0] in 8-bit color STN mode..... | 56 |
| Table 8-3 LCD Controller Register Summary..... | 56 |
| Table 9-1 DMA Controller Register Summary..... | 65 |
| Table 10-1 ADC Controller Register Summary..... | 95 |
| Table 10-2 Interrupt controller Configuration..... | 122 |
| Table 10-3 Interrupt controller Register Summary..... | 123 |
| Table 10-4 Matrix Keyboard Interface Controller Register Summary..... | 128 |
| Table 10-5 PS/2 Controller Register Summary..... | 131 |
| Table 10-6 Non-AMBA Signals within RTC Core Block..... | 136 |
| Table 10-7 RTC Register Summary..... | 137 |
| Table 10-8 Timer Register Summary..... | 139 |
| Table 10-9 UART/SIR Register Summary..... | 146 |
| Table 10-10 Baud Rate with Decimal Divisor at 3.6864MHz Crystal Frequency..... | 149 |
| Table 10-11 Watchdog Timer Register Summary..... | 155 |

1 ARCHITECTURAL OVERVIEW

1.1 Processor

The ARM720T core incorporates an 8K unified write-through cache, and an 8 data entry, 4-address entry write buffer. It also incorporates an MMU with a 64 entry TLB, and WinCE enhancements.

1.2 Video

The integrated LCD controller can control STN displays and TFT displays, up to 640x480 (VGA) resolution and 16bit color. On mono displays it can directly generate 16 gray scales.

1.3 Memory

HMS30C7202 incorporates two independent memory controllers. A high-speed 16-bit wide interface connects directly to one or two 16, 64, 128 or 256MBit SDRAM devices, supporting DRAM memory sizes in the range 2 to 64MB. A separate 32-bit data path interfaces to ROM or Flash devices. Burst mode ROMs are supported, for increased performance, allowing operating system code to be executed directly from ROM. Since the ROM and SDRAM interfaces are independent, the processor core can execute ROM code simultaneously with video DMA access to the SDRAM, thus increasing total effective memory bandwidth, and hence overall performance.

1.4 Internal Bus Structure

The HMS30C7202 internal bus organization is based upon the AMBA standard, but with some minor modifications to the peripheral buses (the APBs). There are three main buses in the HMS30C7202:

1. The main system bus (the ASB) to which the CPU and memory controllers are connected
2. The fast APB to which high-bandwidth peripherals are connected
3. The slow APB (to which timers, the UART and other low-bandwidth peripherals are connected)

There is also a separate video DMA bus.

1.4.1 ASB

The ASB is designed to allow the ARM continuous access to both, the ROM and the SDRAM interface. The SDRAM controller straddles both the ASB and the video DMA bus so the LCD can access the SDRAM controller simultaneously with activity on the ASB. This means that the ARM can read code from ROM, or access a peripheral, without being interrupted by video DMA.

The HMS30C7202 uses a modified arbiter to control mastership on the main ASB bus. The arbiter only arbitrates on quad-word boundaries, or when the bus is idle. This is to get the best performance with the ARM720T, which uses a quad-word cache line, and also to get the best performance from the SDRAM, which uses a burst size of eight half-words per access. By arbitrating only when the bus is idle or on quad-word boundaries ($A[3:2] = 11$), it ensures that cache line fills are not broken up, hence SDRAM bursts are not broken up.

The SDRAM controller controls video ASB arbitration. This is explained in 6.5 Arbitration on page 39.

1.4.2 Video bus

The video bus connects the LCD controller with the SDRAM controller. Data transfers are DMA controlled. The video bus consists of an address bus, data bus and control signals to/from the SDRAM controller. The LCD registers are programmed through the fast APB. The SDRAM controller arbitrates between ASB, VGA access requests. Video always has higher priority than ASB access requests. The splitting ASB/video bus allows slow ASB device accesses SDRAM without blocking video DMA.

1.4.3 APB

There are two APB buses, the fast and slow APB bus. The fast APB bus operates at the speed of the ASB, and hosts the USB interface, the sound output interface, the LCD registers, etc. These are the high performance peripherals, which are generally DMA targets. The slow APB peripherals generally operate at the UART crystal clock frequency of 3.6864MHz, though register access via the APB is at ASB speed.

The slow APB peripherals do not support DMA transfers. This arrangement of running most of the peripherals at a slower clock, and reducing the load on the faster bus, results in significantly reduced power consumption. Both APB buses connect to the main ASB bus via bridges. The slow APB bridge takes care of all resynchronization, handing over data and control signals between the ASB and UART clock domains in a safe and reliable manner.

The fast APB Bridge is modified from the normal AMBA Bridge, to allow DMA access to fast APB peripherals. Additional signals from the DMA controller to the APB bridge request select and acknowledge DMA transfers to and from DMA-aware peripherals.

1.5 SDRAM Controller

The SDRAM controller is a key part of the HMS30C7202 architecture. The SDRAM controller has two data ports - one for video DMA and one for the main ASB - and interfaces to 16-bit wide SDRAMs. One to four 16, 64, 128, or 256 Mbit x16-bit devices are supported, giving a memory size ranging from 2 to 64 Mbytes.

The main ASB and video DMA buses are independent, and operate concurrently. The video bus has always higher priority than the main bus.

The video interface consists of address, data and control signals. The video access burst size is fixed to 16 words. The address is non-incrementing for words within a burst (as the SDRAM controller only makes use of the first address for each burst request).

1.6 Peripheral DMA

1.6.1 Overview

HMS30C7202 incorporates a four-channel, general-purpose DMA controller that operates on the ASB. The DMA controller is an AMBA compliant ASB bus master with a higher arbitration priority than the ARM processor, to ensure low DMA latency. Since, however, the main ASB bus always has lower priority access to the SDRAM controller than the video bus, it will always get lower priority access to SDRAM than the LCD.

1.6.2 Transfer sizes

A device that uses the peripheral DMA is the Sound output. The sound output data rate is 88.2KB/sec. To ensure reasonable usage of SDRAM, APB and ASB bandwidth, the transfer sizes to the sound controller is a single word.

The SDRAM controller does a complete quad-word access for every SDRAM access. The maximum SDRAM bandwidth taken by sound device running concurrently is 0.75%.

DMA accesses to Sound blocks are fully AMBA compliant, meaning that a word transfer takes two bus cycles.

1.6.3 Fly-by

The DMA controller is tightly coupled to the fast APB Bridge. In order for the DMA Controller to start a transfer, it must first receive a DMA data request from one of the peripherals; it will then request mastership of the ASB. Once granted, the DMA Controller will retain mastership of the ASB until the requested DMA transaction is completed, which ensures correct data in the DMA peripherals (i.e. the ARM core cannot modify data while a DMA transfer is in progress).

The DMA transfer request is monitored by the Fast APB bridge, which performs the correspondent APB transfer by inverting the read/write line with respect to the ASB and generates a PWRITE signal on the APB. The DMA transfer is acknowledged on the APB by asserting a PSELDMA signal for the given peripheral. The data is timed by PSTB as on a normal APB transfer. The APB address PA is not used for DMA transfers.

The APB bridge receives two signals from the DMA controller called CHAN [1:0], which tells it which DMA channel (peripheral) the DMA access is for. All other information comes from monitoring the ASB bus signals. For example, the direction of transfer comes from BWRITE (the sense is inverted to get the APB signal), and

when the SDRAM transfer completes, comes from the bridge monitoring the BWAIT ASB signal.

1.6.4 Timing

This is detailed in Chapter 9, Fast AMBA Peripherals.

1.6.5 Sound output

In the HMS30C7202, the sound peripheral is connected to the fast APB bus and supported by the DMA controller. (Note that this is compatible with some operating systems, which require DMA-support sound hardware.)

1.7 Peripherals

Universal Serial Bus (USB) device controller

The USB device controller is used to transfer data from/to host system like PC in high-speed (12Mbits/s) mode. No external USB transceiver is necessary.

PS/2 Interface

The PS/2 port can be used with keyboard, mice or other PS/2 compliant devices. In PS/2 mode the pins are open-drain I/Os, as GPIOs they have normal characteristics.

Universal Asynchronous Receiver and Transmitter (UART)

Four UART ports are implemented. One of them supports full modem interface signals. Some pins are used as GPIO or matrix keyboard pins when not used for UART.

IrDA

IrDA uses UART1 for its SIR transfer in 115 Kbit/s speed. The pins are used as GPIO or matrix keyboard pins when not used for IrDA.

Controller Area Network (CAN)

The two CAN ports are used. The pins are used as GPIO when not used for CAN.

Multimedia Card (MMC), Solid State Floppy Disk Card (SSFDC)

MMC or SSFDC memory card can be used as storage device. The pins are used as GPIO when not used for MMC or SSFDC.

Pulse-Width-Modulated (PWM) Interface

Two PWM output signals are generated. The pins are used as GPIO when not used for PWM.

Matrix Keyboard Interface

Matrix keyboard interface supports up to 64 keys. The pins are used as GPIO when not used for matrix keyboards.

General Purpose DMA Channel

One DMA channel is provided for external device that needs DMA access. The pins are used as GPIO when not used for DMA.

DAC

On chip DAC provides 8-bit audio stereo sound.

ADC

A 5 channel ADC is implemented for touch panel, audio input and monitoring of two voltages. No external transistor switch is necessary for touch panel operation.

PLL

CPU, video and USB clocks are generated by three PLL with 3.6864 MHz input clock.

1.8 Power management

The HMS30C7202 incorporates advanced power management functions, allowing the whole device to be put into a standby mode, when only the real time clock runs. The SDRAM is put into low-power self-refresh mode to preserve its contents. The HMS30C7202 may be forced out of this state by either a real-time clock wake-up interrupt, a user wake-up event (which would generally be a user pressing the "on" key) or by the UART ring-indicate input. The power management unit (PMU) controls the safe exit from standby mode to operational mode, ensuring that SDRAM contents are preserved. In addition, halt and slow modes allow the processor to be halted or run at reduced speed to reduce power consumption. The processor can be quickly brought out of the halted state by a peripheral interrupt. The advanced power management unit controls all this functionality. In addition, individual devices and peripherals may be powered down when they are not in use. The

HMS30C7202 is designed for battery-powered portable applications and incorporates innovative design features in the bus structure and the PMU to reduce power consumption. The slow APB bus allows peripherals to be clocked slowly hence reducing power consumption. The use of three buses reduces the number of nodes that are toggled during a data access, and thereby further reducing power consumption. In addition, clocks to peripherals that are not active can also be gated.

1.8.1 Clock gating

The high performance peripherals, such as the SDRAM controller and the LCD controller, run most of the time at high frequencies and careful design, including the use of clock gating, has minimized their power consumption. Any peripherals can be powered down completely when not in use.

1.8.2 PMU

The Power Management Unit (PMU) is used to control the overall state the system is in. The system can be in one of five states:

Run

The system is running normally. All clocks are running (except where gated locally), and the SDRAM controller is performing normal refresh.

Slow

The system operates normally, except the ARM is placed into Fast Bus mode, and hence is clocked at half its normal rate.

Idle

In this mode, the PMU becomes the bus master until there is an interrupt for the CPU, or the peripheral DMA controller requests mastership of the bus.

Sleep

The SDRAM is placed into self-refresh mode, and internal clocks are gated off. This mode can only be entered from Idle mode (that is, the PMU must be ASB master before this mode can be entered). The PMU must get bus mastership to ensure that the system is stopped in a safe state and not, for example, halfway through an SDRAM write. Usually this state is only to be entered briefly, on the way to entering deep sleep mode.

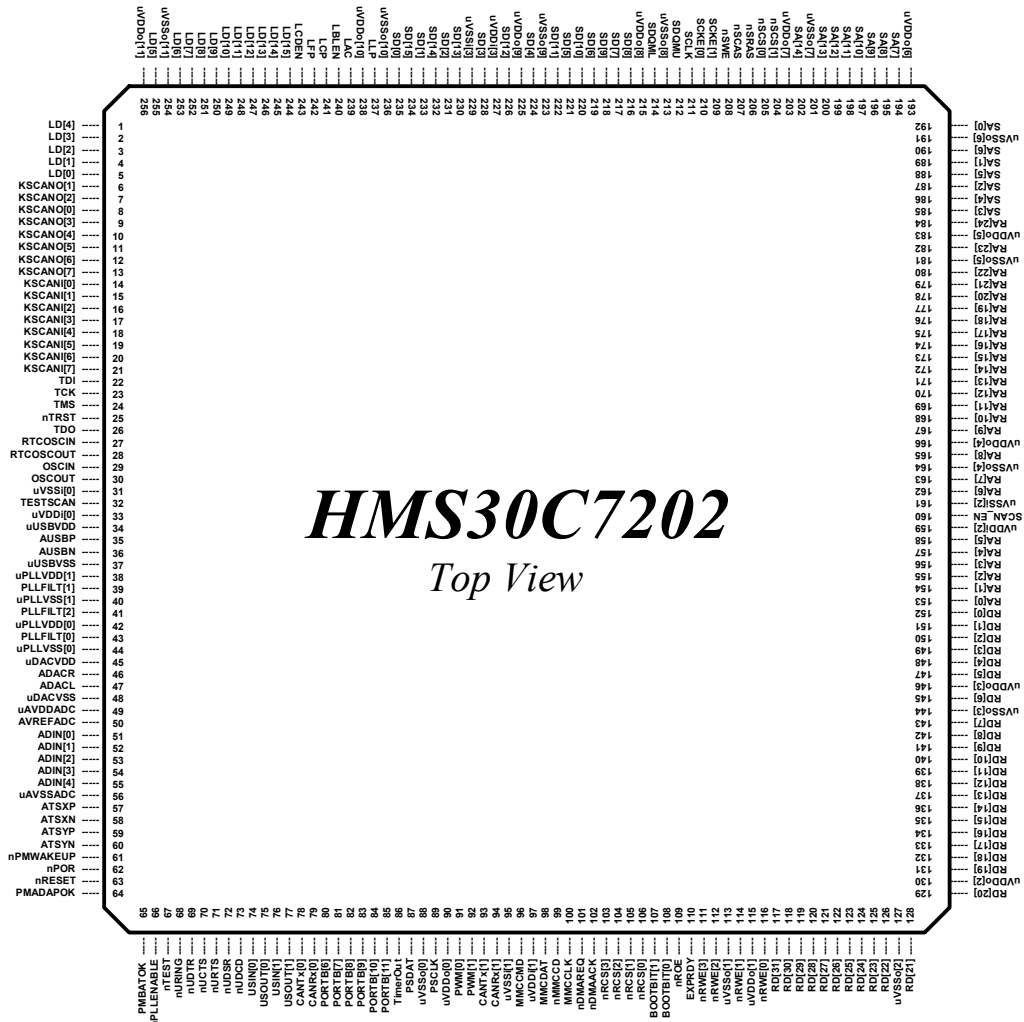
Deep Sleep

In deep sleep mode, the 3.6864MHz oscillator and the PLLs are disabled. This is the lowest power state available. Only the 32kHz oscillator runs. The real time clock and wakeup sections of the PMU are operated from this clock. Everything else is powered down, and SDRAM is in self-refresh mode. This is the normal system "off" mode. Sleep and Deep Sleep modes are exited either by a user wake-up event (generally pressing the "On" key), an RTC wake-up alarm, a device reset request, or by a modem ring indicate event. These interrupt sources go directly to the PMU. In addition, the modem ring indicate signal also goes to the normal interrupt controller to signal an interrupt if there is a ring indicate event in a non-sleep mode.

1.9 Test and debug

The HMS30C7202 incorporates the ARM standard test interface controller (TIC) allowing 32-bit parallel test vectors to be passed onto the internal bus. This allows access to the ARM720T macro-cell core, and also to memory mapped devices and peripherals within the HMS30C7202. In addition, the ARM720T includes support for the ARM debug architecture (Embedded ICE), which makes use of a JTAG boundary scan port to support debug of code on the embedded processor. The same boundary scan port is also used to support a normal pad-ring boundary scan for board level test applications.]

2 PIN DESCRIPTION
2.1 256-Pin Diagram



2.1.1 MQFP Type

| Lead Count | Body Size | Body Thickness | Lead Pitch | Lead Form | Standoff |
|------------|-----------|----------------|------------|-----------|----------|
| 256 | 28.0X28.0 | 3.37 | .40 | 1.30 | .13 |

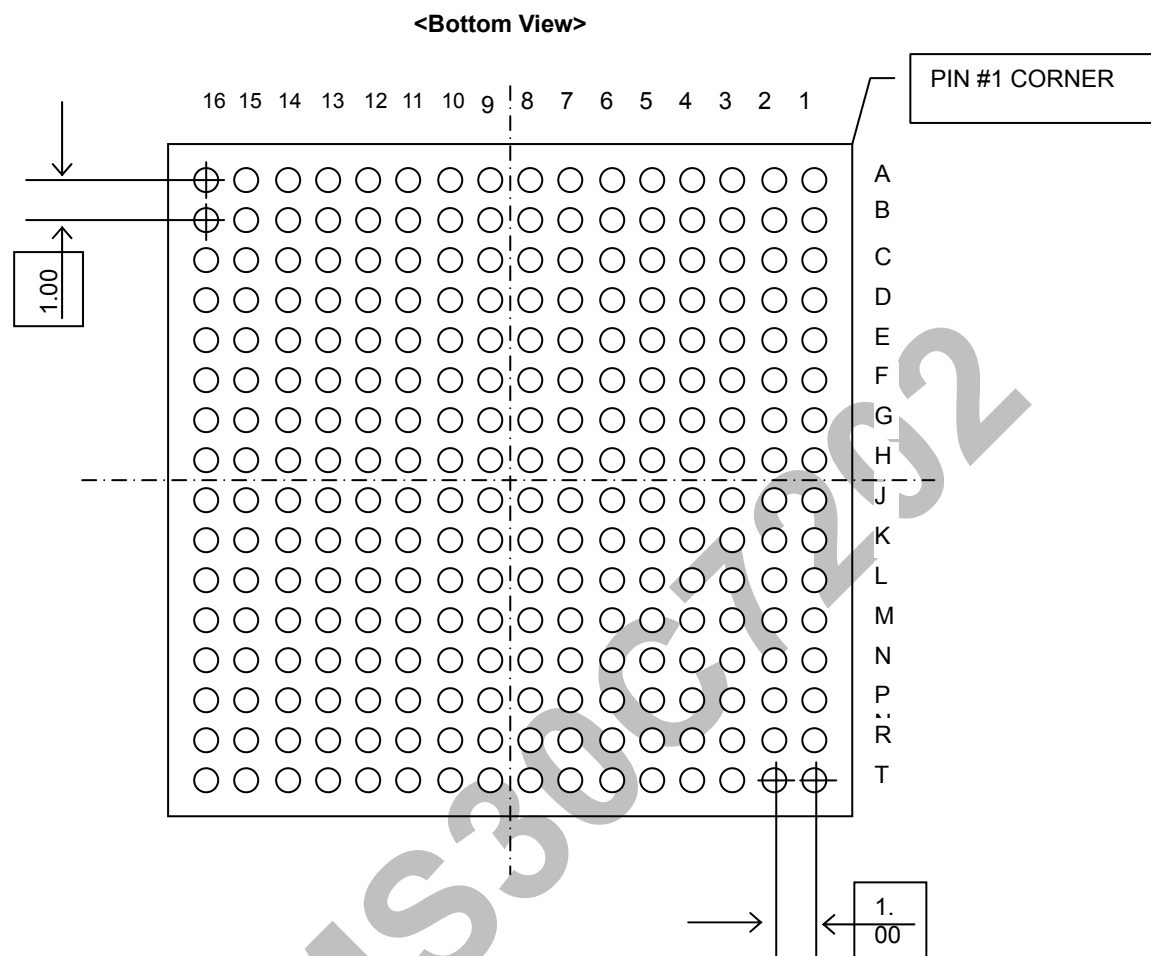
Note : All dimensions in mm.

| Pin No. | PAD Name | Pin No. | PAD Name | Pin No. | PAD Name | Pin No. | PAD Name |
|---------|-----------|---------|-----------|---------|----------|---------|----------|
| 1 | LD[4] | 65 | PMBATOK | 129 | RD[20] | 193 | uVDDo6 |
| 2 | LD[3] | 66 | nPLENABLE | 130 | uVDDo2 | 194 | SA[7] |
| 3 | LD[2] | 67 | nTEST | 131 | RD[19] | 195 | SA[8] |
| 4 | LD[1] | 68 | nURING | 132 | RD[18] | 196 | SA[9] |
| 5 | LD[0] | 69 | nUDTR | 133 | RD[17] | 197 | SA[10] |
| 6 | KSCANO[1] | 70 | nUCTS | 134 | RD[16] | 198 | SA[11] |
| 7 | KSCANO[2] | 71 | nURTS | 135 | RD[15] | 199 | SA[12] |

Figure t Clear in the watchdog timer mode with manual reset Pin Location and Signal

| | | | | | | | |
|----|------------|-----|------------|-----|---------|-----|---------|
| 8 | KSCANO[0] | 72 | nUDSR | 136 | RD[14] | 200 | SA[13] |
| 9 | KSCANO[3] | 73 | nUDCD | 137 | RD[13] | 201 | uVSSo7 |
| 10 | KSCANO[4] | 74 | USIN[0] | 138 | RD[12] | 202 | SA[14] |
| 11 | KSCANO[5] | 75 | USOUT[0] | 139 | RD[11] | 203 | uVDDo7 |
| 12 | KSCANO[6] | 76 | USIN[1] | 140 | RD[10] | 204 | nSCS[1] |
| 13 | KSCANO[7] | 77 | USOUT[1] | 141 | RD[9] | 205 | nSCS[0] |
| 14 | KSCANI[0] | 78 | CANTx[0] | 142 | RD[8] | 206 | nSRAS |
| 15 | KSCANI[1] | 79 | CANRx[0] | 143 | RD[7] | 207 | nSCAS |
| 16 | KSCANI[2] | 80 | PORTB[6] | 144 | uVSSo3 | 208 | nSWE |
| 17 | KSCANI[3] | 81 | PORTB[7] | 145 | RD[6] | 209 | SCKE[1] |
| 18 | KSCANI[4] | 82 | PORTB[8] | 146 | uVDDo3 | 210 | SCKE[0] |
| 19 | KSCANI[5] | 83 | PORTB[9] | 147 | RD[5] | 211 | SCLK |
| 20 | KSCANI[6] | 84 | PORTB[10] | 148 | RD[4] | 212 | SDQMU |
| 21 | KSCANI[7] | 85 | PORTB[11] | 149 | RD[3] | 213 | uVSSo8 |
| 22 | TDI | 86 | TimerOut | 150 | RD[2] | 214 | SDQML |
| 23 | TCK | 87 | PSDAT | 151 | RD[1] | 215 | uVDDo8 |
| 24 | TMS | 88 | uVSSo0 | 152 | RD[0] | 216 | SD[8] |
| 25 | nTRST | 89 | PSCLK | 153 | RA[0] | 217 | SD[7] |
| 26 | TDO | 90 | uVDDo0 | 154 | RA[1] | 218 | SD[9] |
| 27 | RTCOSCIN | 91 | PWM[0] | 155 | RA[2] | 219 | SD[6] |
| 28 | RTCOSCOUT | 92 | PWM[1] | 156 | RA[3] | 220 | SD[10] |
| 29 | OSCIN | 93 | CANTx[1] | 157 | RA[4] | 221 | SD[5] |
| 30 | OSCOUT | 94 | CANRx[1] | 158 | RA[5] | 222 | SD[11] |
| 31 | uVSSi0 | 95 | uVSSi1 | 159 | uVDDi2 | 223 | uVSSo9 |
| 32 | TESTSCAN | 96 | MMCCMD | 160 | SCAN_EN | 224 | SD[4] |
| 33 | uVDDi0 | 97 | uVDDi1 | 161 | uVSSi2 | 225 | uVDDo9 |
| 34 | AVDDUSB | 98 | MMCDAT | 162 | RA[6] | 226 | SD[12] |
| 35 | AUSBP | 99 | nMMCCD | 163 | RA[7] | 227 | uVDDi3 |
| 36 | AUSBN | 100 | MMCCLK | 164 | uVSSo4 | 228 | SD[3] |
| 37 | AVSSUSB | 101 | nDMAREQ | 165 | RA[8] | 229 | uVSSi3 |
| 38 | PLLVD[1] | 102 | nDMAACK | 166 | uVDDo4 | 230 | SD[13] |
| 39 | PLLFILT[1] | 103 | nRCS[3] | 167 | RA[9] | 231 | SD[2] |
| 40 | PLLVD[2] | 104 | nRCS[2] | 168 | RA[10] | 232 | SD[14] |
| 41 | PLLFILT[2] | 105 | nRCS[1] | 169 | RA[11] | 233 | SD[1] |
| 42 | PLLVD[0] | 106 | nRCS[0] | 170 | RA[12] | 234 | SD[15] |
| 43 | PLLFILT[0] | 107 | BOOTBIT[1] | 171 | RA[13] | 235 | SD[0] |
| 44 | PLLVD[0] | 108 | BOOTBIT[0] | 172 | RA[14] | 236 | uVSSo10 |
| 45 | AVDDDAC | 109 | nROE | 173 | RA[15] | 237 | LLP |
| 46 | ADACR | 110 | EXPRDY | 174 | RA[16] | 238 | uVDDo10 |
| 47 | ADACL | 111 | nRWE[3] | 175 | RA[17] | 239 | LAC |
| 48 | AVSSDAC | 112 | nRWE[2] | 176 | RA[18] | 240 | LBLEN |
| 49 | AVDDADC | 113 | uVSSo1 | 177 | RA[19] | 241 | LCP |
| 50 | AVREFADC | 114 | nRWE[1] | 178 | RA[20] | 242 | LFP |
| 51 | ADIN[0] | 115 | uVDDo1 | 179 | RA[21] | 243 | LCDEN |
| 52 | ADIN[1] | 116 | nRWE[0] | 180 | RA[22] | 244 | LD[15] |
| 53 | ADIN[2] | 117 | RD[31] | 181 | uVSSo5 | 245 | LD[14] |
| 54 | ADIN[3] | 118 | RD[30] | 182 | RA[23] | 246 | LD[13] |
| 55 | ADIN[4] | 119 | RD[29] | 183 | uVDDo5 | 247 | LD[12] |
| 56 | AVSSADC | 120 | RD[28] | 184 | RA[24] | 248 | LD[11] |
| 57 | ATSXP | 121 | RD[27] | 185 | SA[3] | 249 | LD[10] |
| 58 | ATSXN | 122 | RD[26] | 186 | SA[4] | 250 | LD[9] |
| 59 | ATSYP | 123 | RD[25] | 187 | SA[2] | 251 | LD[8] |
| 60 | ATSYN | 124 | RD[24] | 188 | SA[5] | 252 | LD[7] |
| 61 | nPMWAKEUP | 125 | RD[23] | 189 | SA[1] | 253 | LD[6] |
| 62 | nPOR | 126 | RD[22] | 190 | SA[6] | 254 | uVSSo11 |
| 63 | nRESET | 127 | uVSSo2 | 191 | uVSSo6 | 255 | LD[5] |
| 64 | PMADAPOK | 128 | RD[21] | 192 | SA[0] | 256 | uVDDo11 |

2.1.2 FBGA Type



| Body Size | Ball Count | Signal I/O | Package Height | Row Array | Ball Matrix | Ball Pitch |
|-----------|------------|------------|----------------|------------|-------------|------------|
| 17.0X17.0 | 256 | 256 | 1.40 | Full Array | 16X16 | 1.00 |

Note : All dimensions in mm.

| Pin No. | PAD Name | Pin No. | PAD Name | Pin No. | PAD Name | Pin No. | PAD Name |
|---------|-----------|---------|------------|---------|----------|---------|----------|
| B1 | LD[4] | T2 | PMBATOK | R16 | RD[20] | A15 | uVDDo6 |
| C2 | LD[3] | R3 | nPLLENABLE | P15 | uVDDo2 | B14 | SA[7] |
| C1 | LD[2] | T3 | nTEST | P16 | RD[19] | A14 | SA[8] |
| D3 | LD[1] | P4 | nURING | N14 | RD[18] | C13 | SA[9] |
| D1 | LD[0] | T4 | nUDTR | N16 | RD[17] | A13 | SA[10] |
| D2 | KSCANO[1] | R4 | nUCTS | N15 | RD[16] | B13 | SA[11] |
| E4 | KSCANO[2] | N5 | nURTS | M13 | RD[15] | D12 | SA[12] |
| E1 | KSCANO[0] | T5 | nUDSR | M16 | RD[14] | A12 | SA[13] |
| E3 | KSCANO[3] | P5 | nUDCD | M14 | RD[13] | C12 | uVSSo7 |
| E2 | KSCANO[4] | R5 | USIN[0] | M15 | RD[12] | B12 | SA[14] |
| F5 | KSCANO[5] | M6 | USOUT[0] | L12 | RD[11] | E11 | uVDDo7 |
| F4 | KSCANO[6] | N6 | USIN[1] | L13 | RD[10] | D11 | nSCS[1] |
| F1 | KSCANO[7] | T6 | USOUT[1] | L16 | RD[9] | A11 | nSCS[0] |
| F3 | KSCANI[0] | P6 | CANTx[0] | L14 | RD[8] | C11 | nSRAS |

| | | | | | | | |
|----|------------|-----|------------|-----|---------|-----|---------|
| F2 | KSCANI[1] | R6 | CANRx[0] | L15 | RD[7] | B11 | nSCAS |
| G6 | KSCANI[2] | L7 | PORTB[6] | K11 | uVSSo3 | F10 | nSWE |
| G5 | KSCANI[3] | M7 | PORTB[7] | K12 | RD[6] | E10 | SCKE[1] |
| G4 | KSCANI[4] | N7 | PORTB[8] | K13 | uVDDo3 | D10 | SCKE[0] |
| G1 | KSCANI[5] | T7 | PORTB[9] | K16 | RD[5] | A10 | SCLK |
| G3 | KSCANI[6] | P7 | PORTB[10] | K14 | RD[4] | C10 | SDQMU |
| G2 | KSCANI[7] | R7 | PORTB[11] | K15 | RD[3] | B10 | uVSSo8 |
| H7 | TDI | K8 | TimerOut | J10 | RD[2] | G9 | SDQML |
| H6 | TCK | L8 | PSDAT | J11 | RD[1] | F9 | uVDDo8 |
| H5 | TMS | M8 | uVSSo0 | J12 | RD[0] | E9 | SD[8] |
| H4 | nTRST | N8 | PSCLK | J13 | RA[0] | D9 | SD[7] |
| H1 | TDO | T8 | uVDDo0 | J16 | RA[1] | A9 | SD[9] |
| H3 | RTCOSCIN | P8 | PWM[0] | J14 | RA[2] | C9 | SD[6] |
| H2 | RTCOSCOUT | R8 | PWM[1] | J15 | RA[3] | B9 | SD[10] |
| J7 | OSCIN | K9 | CANTx[1] | H10 | RA[4] | G8 | SD[5] |
| J6 | OSCOUT | L9 | CANRx[1] | H11 | RA[5] | F8 | SD[11] |
| J5 | uVSSI0 | M9 | uVSSI1 | H12 | uVDDi2 | E8 | uVSSo9 |
| J4 | TESTSCAN | N9 | MMCCMD | H13 | SCAN_EN | D8 | SD[4] |
| J2 | uVDDi0 | R9 | uVDDi1 | H15 | uVSSI2 | B8 | uVDDo9 |
| J3 | AVDDUSB | P9 | MMCDAT | H14 | RA[6] | C8 | SD[12] |
| J1 | AUSBP | T9 | nMMCCD | H16 | RA[7] | A8 | uVDDi3 |
| J8 | AUSBN | J9 | MMCKLK | H9 | uVSSo4 | H8 | SD[3] |
| K5 | AVSSUSB | M10 | nDMAREQ | G12 | RA[8] | E7 | uVSSI3 |
| K4 | PLLVD[1] | N10 | nDMAACK | G13 | uVDDo4 | D7 | SD[13] |
| K2 | PLLFILT[1] | R10 | nRCS[3] | G15 | RA[9] | B7 | SD[2] |
| K3 | PLLVS[1] | P10 | nRCS[2] | G14 | RA[10] | C7 | SD[14] |
| K6 | PLLFILT[2] | L10 | nRCS[1] | G11 | RA[11] | F7 | SD[1] |
| K7 | PLLVD[0] | K10 | nRCS[0] | G10 | RA[12] | G7 | SD[15] |
| K1 | PLLFILT[0] | T10 | BOOTBIT[1] | G16 | RA[13] | A7 | SD[0] |
| L6 | PLLVS[0] | L11 | BOOTBIT[0] | F11 | RA[14] | F6 | uVSSo10 |
| L2 | AVDDDAC | R11 | nROE | F15 | RA[15] | B6 | LLP |
| L4 | ADACR | N11 | EXPRDY | F13 | RA[16] | D6 | uVDDo10 |
| L3 | ADACL | P11 | nRWE[3] | F14 | RA[17] | C6 | LAC |
| L5 | AVSSDAC | M11 | nRWE[2] | F12 | RA[18] | E6 | LBLEN |
| L1 | AVDDADC | T11 | uVSSo1 | F16 | RA[19] | A6 | LCP |
| M4 | AVREFADC | N12 | nRWE[1] | E13 | RA[20] | D5 | LFP |
| M2 | ADIN[0] | R12 | uVDDo1 | E15 | RA[21] | B5 | LCDEN |
| M5 | ADIN[1] | M12 | nRWE[0] | E12 | RA[22] | E5 | LD[15] |
| M3 | ADIN[2] | P12 | RD[31] | E14 | uVSSo5 | C5 | LD[14] |
| M1 | ADIN[3] | T12 | RD[30] | E16 | RA[23] | A5 | LD[13] |
| N4 | ADIN[4] | N13 | RD[29] | D13 | uVDDo5 | D4 | LD[12] |
| N2 | AVSSADC | R13 | RD[28] | D15 | RA[24] | B4 | LD[11] |
| N3 | ATXSP | P13 | RD[27] | D14 | SA[3] | C4 | LD[10] |
| N1 | ATXSN | T13 | RD[26] | D16 | SA[4] | A4 | LD[9] |
| P2 | ATXSP | R14 | RD[25] | C15 | SA[2] | B3 | LD[8] |
| P1 | ATXSN | T14 | RD[24] | C16 | SA[5] | A3 | LD[7] |
| P3 | nPMWAKEUP | P14 | RD[23] | C14 | SA[1] | C3 | LD[6] |
| R1 | nPOR | T15 | RD[22] | B16 | SA[6] | A2 | uVSSo11 |
| R2 | nRESET | R15 | uVSSo2 | B15 | uVSSo6 | B2 | LD[5] |
| T1 | PMADAPOK | T16 | RD[21] | A16 | SA[0] | A1 | uVDDo11 |

2.2 Pin Descriptions

Table 2-2 describes the function of all the external signals to the HMS30C7202.

| Type | Description | Type | Description |
|------|--|------|---------------------------------------|
| O | Output | OA | Analog Output |
| I | Input | IA | Analog Input |
| IO | Input/Output | IOA | Analog Input/Output |
| IS | Input with Schmitt level input threshold | P | Power input |
| U | Suffix to indicate integral pull-up | D | Suffix to indicate integral pull-down |
| m | Suffix to multiple function pin | | |

Table 2-1 Pin Signal Type Definition

2.2.1 External Signal Functions

| Function | Signal Name | Signal Type | Description |
|-------------------------|--------------|-------------------|--|
| LCD | LD[15:0] | Om | LCD data bus. Allow 5:6:5 TFT, color (using [7:0]) or mono, using [3:0] or [7:0] |
| | LCP | O | LCD clock pulse |
| | LLP | O | LCD line pulse (Hsync for TFT) |
| | LFP | O | LCD frame pulse (Vsync for TFT) |
| | LAC | O | LCD AC bias (clock enable for TFT) |
| | LCDEN | O | Display enable signal for LCD. Enables high voltage to LCD |
| | LBLEN | Om | LCD backlight enable |
| Static Memory Interface | RA[24:0] | O | ROM address bus |
| | RD[31:0] | IOm | ROM data bus |
| | nRCS[3:0] | Om | ROM chip select outputs |
| | nROE | O | ROM output enable signal |
| | nRWE[3:0] | Om | ROM write enable signals |
| | EXPRDY | I | Wait from external I/O |
| | BOOTBIT[1:0] | I | 8/16/32 bit ROM selection |
| SDRAM Interface | SCLK | O | SDRAM clock output |
| | SCKE[1:0] | O | SDRAM clock enable output |
| | nSRAS | O | SDRAM RAS output |
| | nSCAS | O | SDRAM CAS output |
| | nSWE | O | SDRAM write enable output |
| | nSCS[1:0] | O | SDRAM chip select outputs |
| | SDQML | O | SDRAM lower data byte enable |
| | SDQMU | O | SDRAM upper data byte enable |
| | SD[15:0] | IO | SDRAM data bus |
| SA[14:0] | O | SDRAM address bus | |
| DMA Interface | nDMAREQ | Im | DMA request input (active Low) |
| | nDMAACK | Om | DMA acknowledge output |
| UART | nUDCD0 | Im | UART data carrier detect input |
| | nUDSR0 | Im | UART data set ready input |
| | nUCTS0 | Im | UART clear to send input |
| | USIN[3:0] | Im | UART serial data inputs |
| | USOUT[3:0] | Om | UART serial data outputs |
| | nUDTR0 | Om | UART data terminal ready |
| | nURTS0 | Om | UART request to send |
| | nURING0 | Im | UART ring input signal (wake-up signal to PMU) |
| IrDA | IRDIN1 | Im | IrDA infra-red data input |
| | IRDOUT1 | Om | IrDA infra-red data output |
| USB | AUSBP | AIO | USB positive signal |
| | AUSBN | AIO | USB negative signal |

| Function | Signal Name | Signal Type | Description |
|-------------------|--------------|-------------|---|
| | AVDDUSB | P | USB analog Vdd |
| | AVSSUSB | P | USB analog Vss |
| PWM | PWM[1:0] | Om | Pulse width modulation output |
| | TIMEROUT | Om | Timer output |
| CAN | CANTX[1:0] | Om | Controlled Area Network data output |
| | CANRX[1:0] | Im | Controlled Area Network data input |
| Matrix Keyboard | KSCANO[7:0] | Om | Matrix keyboard scan outputs |
| | KSCANI[7:0] | Im | Matrix keyboard scan inputs |
| PS/2 Interface | PS2D | ODm | PS2 data signal |
| | PS2CK | ODm | PS2 clock signal |
| | SSDO | Om | MMC card controller data output |
| MMC | SSDI | Im | MMC card controller data input |
| | SSCLK | Om | MMC card controller clock output |
| | nSSCS | Om | MMC card controller chip select |
| | SMD[7:0] | IOm | Smart Media Card (SSFDC) data signals |
| | nSMWP | Om | Smart Media Card (SSFDC) write protect |
| | nSMWE | Om | Smart Media Card (SSFDC) write enable |
| | SMALE | Om | Smart Media Card (SSFDC) address latch enable |
| SSFDC (SmartCard) | SMCLE | Om | Smart Media Card (SSFDC) command latch enable |
| | nSMCD | Im | Smart Media Card (SSFDC) card detection signal |
| | nSMCE | Om | Smart Media Card (SSFDC) chip enable |
| | nSMRE | Om | Smart Media Card (SSFDC) read enable |
| | nSMRB | Im | Smart Media Card (SSFDC) READY/nBUSY signal |
| | ATSXP | IO | Touch screen switch X high drive |
| | ATSXN | O | Touch screen switch X low drive |
| ADC | ATSYP | IO | Touch screen switch Y high drive |
| | ATSYN | O | Touch screen switch Y low drive |
| | ADIN[4:0] | AI | ADC inputs for MIC, battery, touch |
| | AVDDADC | P | ADC analog Vdd |
| | AVSSADC | P | ADC analog Vss |
| | AVREFADC | AI | ADC reference voltage |
| | AVDDDAC | P | DAC analog Vdd |
| DAC | AVSSDAC | P | DAC analog Vss |
| | ADACR | AO | Sound DAC output (Right channel) |
| | ADACL | AO | Sound DAC output (Left channel) |
| | PLLVDD[1:0] | P | PLL analog Vdd |
| PLL | PLLVSS[1:0] | P | PLL analog Vss |
| | PLLFILT[2:0] | AI | External PLL loop filter input pins (1 per PLL) |
| | PORTA[15:0] | IOm | General purpose input/output signals |
| | PORTB[11:0] | IOm | General purpose input/output signals |
| GPIO | PORTC[10:0] | IOm | General purpose input/output signals |
| | PORTD[8:0] | IOm | General purpose input/output signals |
| | PORTE[24:0] | IOm | General purpose input/output signals |
| | nPOR | IS | Power on reset input. Schmitt level input with pullup |
| System | nPMWAKEUP | IS | Wake-up "on-key" input. Low causes PMU to exit standby state. |
| | nRESET | IO | Reset input (also driven out in POR, until the PLL is locked) |
| | PMADAPOK | I | Adapter power OK |
| | PMBATOK | I | Main battery OK |
| | RTCOSCIN | I | RTC oscillator input |
| Oscillator | RTCOSCOUT | O | RTC oscillator output |
| | OSCIN | I | Main oscillator input |
| | OSCOUT | O | Main oscillator output |
| Digital Power/ | VDDCore[3:0] | P | Core Vdd supply (2.5V) |
| | VSSCore[3:0] | P | Core Vss supply |

| Function | Signal Name | Signal Type | Description |
|----------|-------------|-------------|---|
| Ground | VDD[11:0] | P | IO Vdd supply (3.3V) |
| | VSS[11:0] | P | IO Vss supply |
| JTAG | TCK | Iu | JTAG boundary scan and debug test clock |
| | nTRST | Id | JTAG boundary scan and debug test reset |
| | TMS | Iu | JTAG boundary scan and debug test mode select |
| | TDI | Iu | JTAG boundary scan and debug test data input |
| | TDO | O | JTAG boundary scan and debug test data output |
| Test | nPLEENABLE | Id | Low to enable PLL. High to bypass PLL with clock from OSCIN |
| | TESTSCAN | Id | Scan Test Mode Enable |
| | SCAN_EN | Id | Scan Chain Activated |
| | nTEST | Iu | Test mode select |

Table 2-2 External Signal Functions

HMS30C7202

2.2.2 Multiple Function Pins

2.2.2.1 PORT A

Data Input/Output

| Primary (nTEST nPLLENABLE) & ~AEN* & ~AMULSEL** | | GPIO Enable (nTEST nPLLENABLE) & AEN & ~AMULSEL | | MultiFunction Enable (nTEST nPLLENABLE) & ~AEN & AMULSEL | | BOTH Enable (nTEST nPLLENABLE) & AEN & AMULSEL | | Analog Test (~nTEST & ~nPLLENABLE) | |
|---|---------|--|---------|--|---------|---|---------|---------------------------------------|--------------|
| I | O | I | O | I | O | I | O | I | O |
| | KSCAN00 | PORTA0 | PORTA0 | | PORTA0 | PORTA0 | PORTA0 | TPLL3FREQSEL[0] | |
| | KSCAN01 | PORTA1 | PORTA1 | | PORTA1 | PORTA1 | PORTA1 | TPLL3FREQSEL[1] | |
| | KSCAN02 | PORTA2 | PORTA2 | | PORTA2 | PORTA2 | PORTA2 | TPLL3FREQSEL[2] | |
| | KSCAN03 | PORTA3 | PORTA3 | | | PORTA3 | PORTA3 | TPLL3FREQSEL[3] | |
| | KSCAN04 | PORTA4 | PORTA4 | | | PORTA4 | PORTA4 | TPLL3FREQSEL[4] | |
| | KSCAN05 | PORTA5 | PORTA5 | USIN2 | | PORTA5 | PORTA5 | TPLL3FREQSEL[5] | |
| | KSCAN06 | PORTA6 | PORTA6 | | USOUT2 | PORTA6 | PORTA6 | TPLL3PWDN | |
| | KSCAN07 | PORTA7 | PORTA7 | | IRDOUT | PORTA7 | PORTA7 | | TPLL3CLKOut |
| KSCAN10 | | PORTA8 | PORTA8 | | PORTA8 | PORTA8 | PORTA8 | | TPLL3CLKQOut |
| KSCAN11 | | PORTA9 | PORTA9 | | PORTA9 | PORTA9 | PORTA9 | | TPLL3LOCKOut |
| KSCAN12 | | PORTA10 | PORTA10 | | PORTA10 | PORTA10 | PORTA10 | TAIOSTOP | |
| KSCAN13 | | PORTA11 | PORTA11 | | | PORTA11 | PORTA11 | TACH[0] | |
| KSCAN14 | | PORTA12 | PORTA12 | | | PORTA12 | PORTA12 | TACH[1] | |
| KSCAN15 | | PORTA13 | PORTA13 | USIN3 | | PORTA13 | PORTA13 | TACH[2] | |
| KSCAN16 | | PORTA14 | PORTA14 | | USOUT3 | PORTA14 | PORTA14 | TACH[3] | |
| KSCAN17 | | PORTA15 | PORTA15 | IRDIN | | PORTA15 | PORTA15 | TACH[4] | |

* AEN : GPIO PORT A Enable Register (0x8002.301C).

** AMULSEL : GPIO PORT A Multi-Function Select Register (0x8002.30A4).

2.2.2.2 PORT B

Data Input/Output

| Primary nTEST & ~nPLLENABLE & ~BEN* | | GPIO Enable nTEST & ~nPLLENABLE & BEN | | Normal Bypass nTEST & nPLLENABLE | | Normal TEST ~nTEST & nPLLENABLE & ~BEN | | UART TEST ~nTEST & nPLLENABLE & BEN | | Analog Test ~nTEST & ~nPLLENABLE | |
|--|---------|--|---------|--|-------|---|------|--|-------|--|--------|
| I | O | I | O | I | O | I | O | I | O | I | O |
| nURING | | PORTB0 | PORTB0 | nURING | | TBCLK | | nURING | | | |
| | nUDTR | PORTB1 | PORTB1 | | nUDTR | TBCCLK | | | nUDTR | | |
| nUCTS | | PORTB2 | PORTB2 | nUCTS | | | | nUCTS | | TACK | |
| | nURTS | PORTB3 | PORTB3 | | nURTS | | | | nURTS | | TAD[9] |
| nUDSR | | PORTB4 | PORTB4 | nUDSR | | | | nUDSR | | | TAD[8] |
| nUDCD | | PORTB5 | PORTB5 | nUDCD | | | | nUDCD | | | TAD[7] |
| PORTB6 | PORTB6 | PORTB6 | PORTB6 | TBFCLK | | TBFCLK | | | | | |
| PORTB7 | PORTB7 | PORTB7 | PORTB7 | TBQFCLK | | TBQFCLK | | | | | |
| PORTB8 | PORTB8 | PORTB8 | PORTB8 | TBCLK | | TBCLK | | | | | |
| PORTB9 | PORTB9 | PORTB9 | PORTB9 | | | | TACK | | TACK | | TACK |
| PORTB10 | PORTB10 | PORTB10 | PORTB10 | TBCLK | | TREQB | | TREQB | | TREQB | |
| PORTB11 | PORTB11 | PORTB11 | PORTB11 | TBCLK | | TREQA | | TREQA | | TREQA | |

* BEN : GPIO PORT B Enable Register (0x8002.303C).

2.2.2.3 PORT C

Data Input/Output

| Primary (nTEST nPllenable) & ~CEN* | | GPIO Enable (nTEST nPllenable) & CEN | | Analog Test ~nTEST & ~nPllenable | |
|---|-----------------------|--|---------|--|--------|
| I | O | I | O | I | O |
| | TIMEROUT | PORTC0 | PORTC0 | | TAD[2] |
| | CANTX0 | PORTC1 | PORTC1 | | TAD[4] |
| CANRX0 | | PORTC2 | PORTC2 | | TAD[3] |
| PSDAT | PSDAT | PORTC3 | PORTC3 | | TAD[1] |
| PSCLK | PSCLK | PORTC4 | PORTC4 | | TAD[0] |
| | PWM0 | PORTC5 | PORTC5 | TDIOSTOP | |
| | PWM1 | PORTC6 | PORTC6 | TDLEFT | |
| nDMAREQ | | PORTC7 | PORTC7 | TDD[2] | |
| | nDMAACK | PORTC8 | PORTC8 | TDD[1] | |
| | nRCS2 / [nRCS2dma] | PORTC9 | PORTC9 | | |
| | nRCS3 | PORTC10 | PORTC10 | TDD[0] | |

* CEN : GPIO PORT C Enable Register (0x8002.305C).

2.2.2.4 PORT D

Data Input/Output

| Primary (nTEST nPllenable) & ~DEN* | | GPIO Enable (nTEST nPllenable) & DEN | | Analog Test ~nTEST & ~nPllenable | |
|---|-------|---|--------|-------------------------------------|---|
| I | O | I | O | I | O |
| | LD8 | PORTD0 | PORTD0 | TPLL1PWDN | |
| | LD9 | PORTD1 | PORTD1 | TPLL1FREQSEL[0] | |
| | LD10 | PORTD2 | PORTD2 | TPLL1FREQSEL[1] | |
| | LD11 | PORTD3 | PORTD3 | TPLL1FREQSEL[2] | |
| | LD12 | PORTD4 | PORTD4 | TPLL1FREQSEL[3] | |
| | LD13 | PORTD5 | PORTD5 | TPLL1FREQSEL[4] | |
| | LD14 | PORTD6 | PORTD6 | TPLL1FREQSEL[5] | |
| | LD15 | PORTD7 | PORTD7 | TPLL1PCLKIn | |
| | LBLEN | PORTD8 | PORTD8 | | |

● DEN : GPIO PORT D Enable Register (0x8002.307C).

**2.2.2.5 PORTE
Data Input/Output**

| Primary (nTEST & ~HalfWordSel & ~EEN*1) | | GPIO Enable (nTEST & EEN) | | MultiFunction 1 (nTEST & HalfWordSel*3 & ~EEN & ~SWAP*2) | | MultiFunction 2 (nTEST & HalfWordSel & ~EEN & SWAP) | | Test Mode (~nTEST) | | Analog Test (~nTEST & ~nPLEENABLE) | |
|--|------------------|------------------------------|---------|--|----------|---|----------|-----------------------|---------|--|---|
| I | O | I | O | I | O | I | O | I | O | I | O |
| RD16 | RD16 | PORTE0 | PORTE0 | | nUSBOE | SMD7 | SMD7 | RD16 | RD16 | | |
| RD17 | RD17 | PORTE1 | PORTE1 | | UVPO | SMD6 | SMD6 | RD17 | RD17 | | |
| RD18 | RD18 | PORTE2 | PORTE2 | | UVMO | SMD5 | SMD5 | RD18 | RD18 | | |
| RD19 | RD19 | PORTE3 | PORTE3 | | USUSPEND | SMD4 | SMD4 | RD19 | RD19 | | |
| RD20 | RD20 | PORTE4 | PORTE4 | URCVIN | | SMD3 | SMD3 | RD20 | RD20 | | |
| RD21 | RD21 | PORTE5 | PORTE5 | UVM | | SMD2 | SMD2 | RD21 | RD21 | | |
| RD22 | RD22 | PORTE6 | PORTE6 | UVP | | SMD1 | SMD1 | RD22 | RD22 | | |
| RD23 | RD23 | PORTE7 | PORTE7 | SMD7 | SMD7 | SMD0 | SMD0 | RD23 | RD23 | | |
| RD24 | RD24 | PORTE8 | PORTE8 | SMD6 | SMD6 | | nSMWP | RD24 | RD24 | | |
| RD25 | RD25 | PORTE9 | PORTE9 | SMD5 | SMD5 | | nSMWE | RD25 | RD25 | | |
| RD26 | RD26 | PORTE10 | PORTE10 | SMD4 | SMD4 | | SMALE | RD26 | RD26 | | |
| RD27 | RD27 | PORTE11 | PORTE11 | SMD3 | SMD3 | | nSMRE | RD27 | RD27 | | |
| RD28 | RD28 | PORTE12 | PORTE12 | SMD2 | SMD2 | | nSMCE | RD28 | RD28 | | |
| RD29 | RD29 | PORTE13 | PORTE13 | SMD1 | SMD1 | nSMCD | | RD29 | RD29 | | |
| RD30 | RD30 | PORTE14 | PORTE14 | SMD0 | SMD0 | | SMCLE | RD30 | RD30 | | |
| RD31 | RD31 | PORTE15 | PORTE15 | | nSMWP | nSMRB | | RD31 | RD31 | | |
| | nRW2 | PORTE16 | PORTE16 | | nSMWE | CANRX1 | | | PORTE16 | | |
| | nRW3 | PORTE17 | PORTE17 | | SMALE | | CANTX1 | | PORTE17 | | |
| MMCCMD / SSDI | MMCCMD/ ZERO | PORTE18 | PORTE18 | | nSMRE | | nUSBOE | | PORTE18 | TDD[6] | |
| MMCDAT | MMCDAT / SSDO | PORTE19 | PORTE19 | | nSMCE | | UVPO | | PORTE19 | TDD[5] | |
| nMMCCD | ZERO/ nSSCS | PORTE20 | PORTE20 | nSMCD | | | UVMO | | PORTE20 | TDD[4] | |
| | MMCLK / SSCLK | PORTE21 | PORTE21 | | SMCLE | | USUSPEND | | PORTE21 | TDD[3] | |
| CANRX1 | | PORTE22 | PORTE22 | nSMRB | | URCVIN | | | PORTE22 | TDD[7] | |
| | CANTX1 | PORTE23 | PORTE23 | | PORTE23 | UVM | | | PORTE23 | TDRIGHT | |
| | RA24 | PORTE24 | PORTE24 | | RA24 | UVP | | | RA24 | | |

*1 EEN : GPIO PORT E Enable Register (0x8002.309C).

*2 SWAP : SWAP Pin Configuration Register (0x8002.30A8).

*3 When HalfWordSel is enable, MultiFunction 1 or 2 is usable instead of Primary RD16~31. To enable HalfWordSel , you should set bottom bits[1:0] of SMI Registers(MEMCFG0~3 on the Table 7-1) to [01 or 10 or 11].

Note : A 32 bit access is not possible without RD16~RD31.

So User should make program to disable PORTE for 32bit access time.

We are not guarantee that the program is alternated 32bit access(RD0~31) with PORTE.

2.2.2.6 USB Transceiver Test & Analog Test
Data Input/Output

| nTEST & | | ~nTEST & | | | |
|---------|-----|-----------------------|--------|------------------------|---------|
| Primary | | ~LCDEn & ~USBTransSel | | & ~LCDEn & USBTransSel | |
| I | O | I | O | I | O |
| | LD0 | TCANCK | | TnUSBOE | |
| | LD1 | TCANSM | | TUVPO | |
| | LD2 | TCANSI | | TUVMO | |
| | LD3 | | TCANSO | TUSUSPEND | |
| | LD4 | | | | TURCVIN |
| | LD5 | | | | TUVM |
| | LD6 | | | | TUVP |

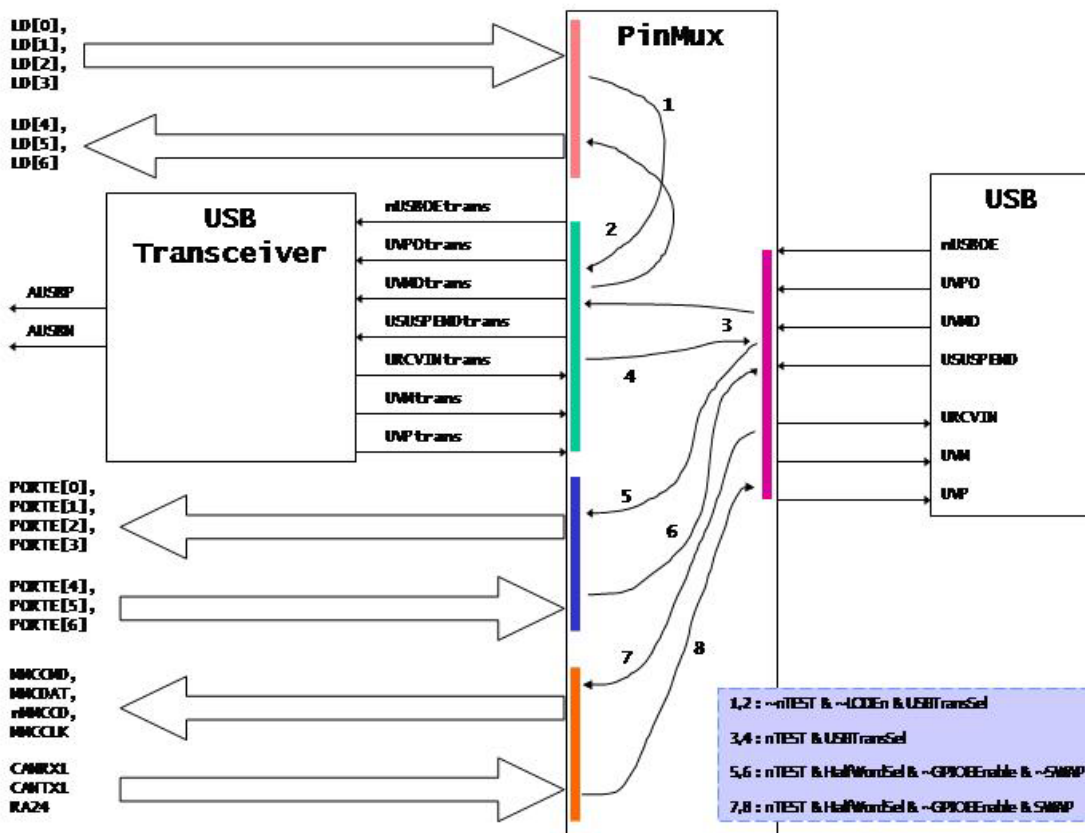


Figure USB Transceiver Test Scheme VER1.5

2.2.2.7 DMA
Data Input/Output

| nTEST & | | | |
|---------|-------|----------|----------|
| nDMAACK | | ~nDMAACK | |
| I | O | I | O |
| | nROE | | nROEdma |
| | nRWE0 | | nRWE0dma |

2.2.2.8 Inverter Chain

When nTESTANA == 0, BOOTBIT1 → nRWE1 (total 50ns delay expected)

3 ARM720T MACROCELL

3.1 ARM720T Macrocell

For details of the ARM720T, please refer to the *ARM720T Data Sheet* (DDI 0087).

HMS30C7202

4 MEMORY MAP

There are five main memory map divisions, outlined in Table 4-1 Top-level address map

| Base Address (Byte) | Base Address (Hex) | Size | Description |
|---------------------|--------------------|-----------|----------------------------|
| 0 Mbyte | 0x0000.0000 | 32Mbytes | ROM chip select 0 |
| 64 Mbytes | 0x0400.0000 | 32Mbytes | ROM chip select 1 |
| 128 Mbytes | 0x0800.0000 | 32Mbytes | ROM chip select 2 |
| 192 Mbytes | 0x0C00.0000 | 32Mbytes | ROM chip select 3 |
| 256 Mbytes | 0x1000.0000 | 256Mbytes | Reserved |
| 512 Mbytes | 0x2000.0000 | 512Mbytes | Reserved |
| 1024 Mbytes | 0x4000.0000 | 32Mbytes | SDRAM chip select 0 |
| 1056 Mbytes | 0x4200.0000 | 32Mbytes | SDRAM chip select 1 |
| 1088 Mbytes | 0x4400.0000 | | SDRAM mode register chip 0 |
| 1120 Mbytes | 0x4600.0000 | | SDRAM mode register chip 1 |
| 1152 Mbytes | 0x4800.0000 | 896Mbytes | Reserved |
| 2048 Mbytes | 0x8000.0000 | 336Kbytes | Peripherals |

Table 4-1 Top-level address map

The ROM has an address space of 256Mbytes that is split equally between four external ROM chip select. Actual address range for each chip select is 32Mbytes with 25 external address signals.

There is a maximum of 64Mbytes of SDRAM space. Reading from the address space(over 0x4400.0000) above the SDRAM address space(0x4000.0000~0x43ff.ffff) sets the mode registers in the SDRAM (**To set the SDRAM mode register, read operation from the ranges of SDRAM mode register is needed. For more information, refer 6.3.**).

The peripheral address space is subdivided into three main areas: those on the ASB, the fast APB and the slow APB. The base address for the peripherals is given in Table 3-2: Peripherals base addresses.

| Function | Base Address (Hex) | Name | Description |
|----------------------|--------------------|--------------|----------------------|
| ASB Peripherals | 0x7F00.0000 | IntSRAM Base | Internal SRAM |
| | 0x7F00.0800 | Reserved | ~0x7FFF.FFFF |
| | 0x8000.0000 | SDRAMC Base | SDRAM Controller |
| | 0x8000.1000 | PMU Base | PMU/PLL |
| | 0x8000.2000 | Reserved | |
| | 0x8000.3000 | BUSC Base | Bus controller |
| | 0x8000.4000 | DMAC Base | DMAC |
| | 0x8000.5000 | Reserved | ~0x8000.FFFF |
| Fast APB Peripherals | 0x8001.0000 | LCD | LCD |
| | 0x8001.1000 | Reserved | |
| | 0x8001.2000 | USB Base | USB |
| | 0x8001.3000 | Sound Base | SOUND |
| | 0x8001.4000 | Reserved | |
| | 0x8001.5000 | MMC Base | MMC/ SPI |
| | 0x8001.6000 | SMC Base | SMC |
| Slow APB Peripherals | 0x8001.7000 | Reserved | ~0x8001.FFFF |
| | 0x8002.0000 | U0 Base | UART 0 |
| | 0x8002.1000 | U1 Base | UART 1 (support SIR) |
| | 0x8002.2000 | KBD Base | KBD |
| | 0x8002.3000 | GPIO Base | GPIO |
| | 0x8002.4000 | INTC Base | INTC |
| | 0x8002.5000 | Timer Base | TIMER |
| | 0x8002.6000 | Reserved | ~0x8002.7FFF |
| | 0x8002.8000 | RTC Base | RTC |
| | 0x8002.9000 | ADC Base | ADC |
| | 0x8002.A000 | Reserved | |
| 0x8002.B000 | WDT Base | WDT | |

| Function | Base Address (Hex) | Name | Description |
|-----------------|---------------------------|-------------|--------------------|
| | 0x8002.C000 | PS2 Base | PS2 |
| | 0x8002.D000 | U2 Base | UART2 |
| | 0x8002.E000 | U3 Base | UART3 |
| | 0x8002.F000 | CAN0 Base | CAN0 |
| | 0x8003.0000 | CAN1 Base | CAN1 |
| | 0x8003.1000 | Reserved | ~0x8004.FFFF |

Table 4-2 Peripherals Base Addresses

HMS30C7202

5 PMU & PLL

The HMS30C7202 is designed primarily for HPC and other portable computing applications. Therefore there are 4 operating modes to reduce power consumption and extend battery life.

- RUN - normal operation (used for CPU-intensive tasks)
- SLOW - half-speed operation used when the application interacts with a user (e.g. word processing)
- IDLE - where the CPU operation is halted but peripherals operation continue (such as screen refresh, or serial communications)
- SLEEP & DEEP SLEEP - This mode will be perceived as 'OFF' by the user, but the SDRAM contents is maintained and only the real-time clock is running.

The transition between these modes is controlled by the PMU (see also 7.3 Power management states, page 7-5). The PMU is an ASB slave unit to allow the CPU to write to its control registers, and is an ASB master unit to provide the mechanism for stopping the ARM core's internal clock.

5.1 Block Functions

CLOCK generator

The CLOCK generator module controls the PLLs and gating clocks while the PLL outputs are unknown and to ensure that clocks are available during test modes and during RESET sequences.

FCLK (ARM Processor and SDRAM controller clock)

Derived from PLL3, programmable between 49.7664 MHz and 82.944 MHz by a 6-bit register (default frequency is 70.0416 MHz).

There are two methods for updating frequency, depending upon the state of bit 6 of the Clock Control register ClkCtl (see ClkCtl register on page 7-11). If bit 6 is set, then any data written to bits [5:0] of the ClkCtl register are immediately transferred to the pins of PLL3, thus causing the loop to unlock and to mute FCLK. This is only a safe mode of operation if PLL3 frequency and mark-space ratio is guaranteed to be within limits immediately after the Lock Detect signal has become active. If bit 6 is NOT set, then the HMS30C7202 must enter DEEP sleep mode before bits [5:0] of the Clock Control register are transferred to PLL3.

To switch between the two frequencies when bit 6 is not set:

- Software writes the new value into the ClkCtl register
- Set a Real Time Clock Alarm to wake the HMS30C7202 in 2 seconds
- Enter DEEP SLEEP Mode by writing to the PMUMode Register
- The HMS30C7202 will power up with PLL3 running at the new frequency

BCLK

Bus Clock is generated by the PMU by dividing FCLK by 2.

VCLK

VCLK is generated by PLL1 and clocks the LCD controller. The frequency is selectable between 24.8832MHz or 41.472MHz (default is 30.4128 MHz). The VCLK PLL is disabled when on BnRES is active or when the PMU is put into DEEP SLEEP mode. On exit from either of these conditions, the VCLK PLL must be re-enabled by software.

Changing Frequency:

1. Software must first disable the VCLK pll, by writing a '0' to the PLL1Enable bit of the ClkCtl register.
2. Write the new value to the PLL1Freq bit.
3. Re-enable the VCLK pll by writing 1 to the PLL1Enable bit.

CCLK

CCLK is generated by PLL2 and clocks the CAN and the USB block - Nominally 48MHz. The CCLK PLL is disabled when BnRES active or when the PMU is put into DEEP SLEEP mode. On exit from either of these conditions, the CCLK PLL must be re-enabled by software.

PMU state machine

The state machine handles the transition between the power management states described below. The CPU

can write to the PMU mode registers (which is what would typically happens when a user switches off the device) and the state machine will proceed to the commanded state.

5.2 Power management

5.2.1 State Diagram

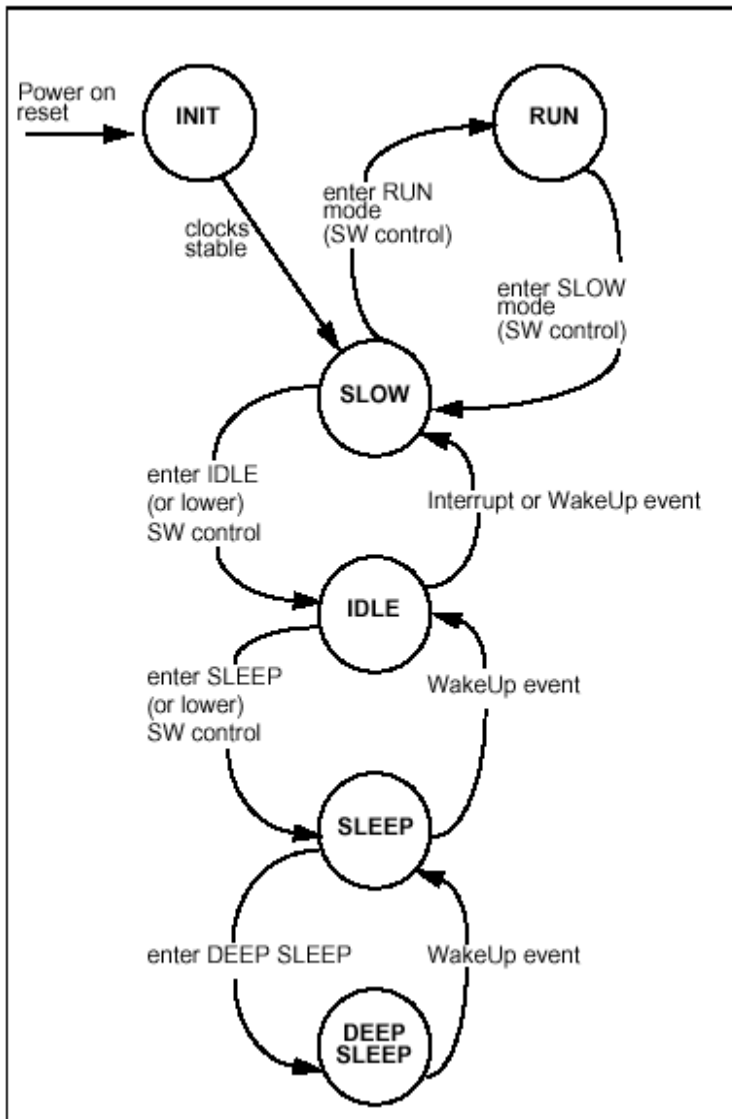


Figure 5-1 PMU Power Management State Diagram

5.2.2 Power management states

RUN

The system is running normally. All Clocks running (except where gated locally). The SDRAM controller is performing normal refresh.

SLOW

The CPU is switched into FastBus mode, and hence runs at the BCLK rate (half the FCLK rate). This is the default mode after exiting SLEEP Mode.

IDLE

In this mode, the PMU becomes the bus master until there is either a fast or normal interrupt for the CPU, or the peripheral DMA controller requests master-ship of the bus.

This will cause the clocks in the CPU to stop when it attempts an ASB access. This mode can be initiated by writing the PMU_IDLE value to the PMU Mode Register (in RUN or SLOW mode), or by a WakeUp signal while the CPU is in SLEEP or DEEP SLEEP mode.

SLEEP

In this mode, the SDRAM is put into self-refresh mode, and internal clocks are gated off. This mode can only be entered from IDLE mode (the PMU bus master must have mastership of the ASB before this mode can be entered). The PMU must be bus master to ensure that the system is stopped in a safe state, and is not half way through a SDRAM write (for example). Both the Video and Communication clocks should be disabled before entering this state.

Usually this state would only be entered briefly, on the way to entering DEEP SLEEP mode.

DEEP SLEEP

In DEEP SLEEP mode, the 3.6864MHz oscillator and the PLL are disabled. This is the lowest power state available. Only the 32 kHz oscillator runs, driving the real time clock and the PMU. Clocked circuitry in the PMU runs at 4kHz (i.e. the RTC clock divided by 8). Everything else is powered down, and SDRAM is in self refresh mode. This is the normal system "off" mode.

SLEEP and DEEP SLEEP modes are exited either by a user wake-up event (generally pressing the "On" key), or by an RTC wake-up alarm, or by a modem ring indicate event. These interrupt sources go directly to the PMU.

5.2.3 Wake-up Debounce and Interrupt

The Wake-up events are debounced as follows:

Each of the event signals which are liable to noise (nRESET, RTC, nPMWAKEUP, and Modem Ring Indicator, Power Adapter Condition) is re-timed to a 250 Hz clock derived from the low power (4 kHz) clock. After filtering to a quarter of 250 Hz, each event has an associated 'sticky' register bit. nPMWAKEUP is an external input, which may be typically connected to an "ON" key.

A 'sticky' bit is a register bit that is set by the incoming event, but is only reset by the CPU. Thus should a PLL drop out of lock momentarily (for example) the CPU will be informed of the event, even if the PLL has regained lock by the time the CPU can read its associated register bit.

The nPMWAKEUP, Modem, Real Time Clock, HotSync(GPIOB[10]) and Power Adapter condition inputs are combined to form the PMU Interrupt. Each of these four interrupt sources can wake up from deep-sleep mode individually and all wake-up operation can not mask able. But when wake-up occur, user can mask interrupt signal to inform interrupt controller.

To make use of the nPMWAKEUP Interrupt, (for example) controlling software will need to complete the following tasks:

- Enable the nPMWAKEUP interrupt bit, by writing 1 to bit[11] of the Reset / Status register (PMUSTAT register).
- Once an interrupt has occurred, read the RESET / Status register to identify the source(s) of interrupt. In the case of a nPMWAKEUP event, the register will return 0x10.
- Clear the appropriate 'sticky' bit by writing a 1 to the appropriate location (in the nPMWAKEUP case, this will be 0x10.).

But Even though the nPMWAKEUP interrupt mask bit is masked, by writing 0 to bit[11] of the Reset Status register, chip shall wake-up with nPMWAKEUP signal.

PORTB[10] (HotSync) Wake-up Sequence

The HotSync interrupt is OR gated with nPMWAKEUP to support additional wake up sources.

HotSync input signal can be used as a wake up source; they are enabled using the Interrupt MASK Register. After wake up, s/w should program the PORTB Interrupt Mask Register and/or the PMU ResetStatus Register.

One other possible application is to use the nDCD signal, from the UART interface, as a wake up source, by connecting nDCD to a PORTB input. In Deep Sleep mode, nDCD can wake up the system by generating a PORTB interrupt request to the PMU block. The PMU state machine then returns the system to the operational mode.

5.3 Registers

| Address | Name | Width | Default | Description |
|-------------|----------|-------|---------|-------------------------------|
| 0x8000.1000 | PMUMODE | 4 | | PMU Mode Register |
| 0x8000.1010 | PMUID | 32 | | PMU ID Register |
| 0x8000.1020 | PMUSTAT | 17 | | PMU Reset/PLL Status Register |
| 0x8000.1028 | PMUCLK | 16 | 0x1B | PMU Clock Control Register |
| 0x8000.1030 | PMUDBCT | 9 | | PMU Debounce Test Register |
| 0x8000.1038 | PUMPLLTR | 21 | | PMU PLL Test Register |

Table 5-1 PMU Register Summary

5.3.1 PMU Mode Register (PMUMODE)

This read/write register is to change from RUN mode or SLOW mode into a different mode. The encoding is shown below, in PMU Mode encoding. The register can only be accessed in RUN mode or SLOW mode (these are the only modes in which the processor is active). Therefore, the processor will never be able to read values for modes other than mode 0x00 and mode 0x 01. A test controller may read other values as long as clocks are enabled with bit 8 of the PMU Debounce Counter Test Register. For more information, please refer 5.3.6.

| | | | | 0x80001000 | | | |
|------|------|--|--------------------------|------------|---|--|--|
| 31 | ... | 3 | 2 | 1 | 0 | | |
| | | WAKEUP | MODE SEL | | | | |
| Bits | Type | Function | | | | | |
| 31:4 | - | Reserved | | | | | |
| 3 | R/W | Writing a `1` to this bit allows PMU to exit DEEP SLEEP mode when pins PMBATOK and PMADAPOK are both low. Writing a `0` to this bit prevents the PMU from leaving DEEP SLEEP mode when PMBATOK and PMADAPOK are both low | | | | | |
| 2:0 | R/W | Value | PMU Mode encoding | | | | |
| | | 0x04 | Initialization mode | | | | |
| | | 0x01 | RUN mode | | | | |
| | | 0x00 | SLOW mode | | | | |
| | | 0x02 | IDLE mode | | | | |
| | | 0x03 | SLEEP mode | | | | |
| | | 0x07 | DEEP SLEEP mode | | | | |

Note: All other values in the above table are undefined.

5.3.2 PMU ID Register (PMUID)

This read-only register returns a unique chip revision ID. Revision 0 of the HMS30C7202 device (the first revision) will return the constant value 0x00720200.

| | | | | | | | |
|----|-----|------------|--|------------|--|---|--|
| | | | | 0x80001010 | | | |
| 31 | ... | | | | | 0 | |
| | | 0x00720200 | | | | | |

5.3.3 PMU Reset/PLL Status Register (PMUSTAT)

This read/write register provides status information on power on reset and the PLL status. The allocation is shown in following two tables: ResetStatus Register Bits. The bits in this register are `sticky` bits. For a definition of a sticky bit, please refer to 5.2.3 Wake-up Debounce and Interrupt. Generally, this register will be

read each time the ARM exits reset mode, so that the ARM can identify what event has caused it to exit from reset mode.

0x80001020

| | | | | | | | |
|-------------------|-----------------|-----------------|------------------|----------------|-------------------|--------------|--------------------|
| | | | | | | | 16 |
| | | | | | | | WARM RESET |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| HOTSYNC INTR | ADAPTOR INTR | RTC INTR | MRING INTR | WAKEUP INTR | HOTSYNC STATUS | WDT RST | WARM RST STATUS |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ADAPTOR STATUS | RTC STATUS | MRING STATUS | WAKEUP STATUS | PLL3 LOCK | PLL2 LOCK | PLL1 LOCK | POR STATUS |

| Bits | Type | Function |
|-------|------|--|
| 31:17 | - | Reserved |
| 16 | W | Warm RESET. Writing a `1' causes nRESET to be asserted. Writing `0' has no effect. |
| 15 | R/W | HOTSYNC interrupt Mask. When reads, 0 = Disable Hotsync interrupt from External pin. 1 = Enable Hotsync interrupt from External pin. When writes to these bits, PMU Interrupts will be enabling. `1' enables interrupts to the CPU, `0' masks such activity. Should the enable bit be set to one when one of the debounced event signals is set, then an interrupt WILL be generated (i.e. the interrupt is level sensitive, not edge sensitive). |
| 14 | R/W | No External Power Interrupt Mask. When reads, 0 = Disable PMU interrupt from PMADAPOK LOW. 1 = Enable PMU interrupt from PMADAPOK LOW. |
| 13 | R/W | RTCEvt Interrupt Mask. When reads, 0 = Disable PMU interrupt from RTC 1 = Enable PMU interrupt from RTC |
| 12 | R/W | RIEvt Interrupt MASK PMU Interrupt Request / Clear When reads, 0 = Disable PMU interrupt from MRING 1 = Enable PMU interrupt from MRING |
| 11 | R/W | OnEvt Interrupt MASK PMU Interrupt Enable When reads, 0 = Disable PMU interrupt from nPMWAKEUP 1 = Enable PMU interrupt from nPMWAKEUP |
| 10 | R/w | HOTSYNC Event When reads, 0 = Not Hot Sync state; 1 = Hot Sync status When writes, HotSync Interrupt Clear. Writing a `1' to this bit clears the event bit |
| 9 | R/w | WDTEvt: Watch Dog Reset (Warm reset) When reads, 0 = No Watch dog Timer event occurred 1 = A Watch dog timer event has occurred since last cleared When writes, Watch dog Reset Clear. Writing a `1' to this bit clears the event bit |
| 8 | R/w | RESETEvt: Warm RESET Event (debounced) When reads, 0 = No Warm RESET event has occurred 1 = A Warm RESET event has occurred since last cleared When writes, Warm Reset Clear. Writing a `1' to this bit clears the event bit. |
| 7 | R/w | PowerFailEvt: ADPATOR NOT OK (debounced) When reads, 0 = No Power Fail event since last cleared 1 = A Power Fail event has occurred since last cleared When writes, Power Fail Interrupt Clear. Writing a `1' to this bit clears a pending interrupt bit. |
| 6 | R/w | RTCEvt When reads, 0 = No Real Time Clock (RTC) calendar wake-up event since last cleared 1 = Real Time Clock (RTC) calendar wake-up event since last cleared When writes, RTC Interrupt Clear. Writing a `1' to this bit clears a pending interrupt bit. |
| 5 | R/w | RIEvt (debounced) When reads, 0 = No Modem Ring Indicate wake-up event since last cleared 1 = Modem Ring Indicate wake-up event since last cleared When writes, RI Interrupt Clear. Writing a `1' to this bit clears a pending interrupt bit. |

| | | |
|---|-----|---|
| 4 | R/w | OnEvt (debounced) When reads, 0 = No On key event since last cleared; 1 = On key event since last cleared When writes, OnEvt Interrupt Clear. Writing a `1` to this bit clears a pending interrupt bit. |
| 3 | R/w | PLLLock3 When reads, 0 = System PLL has been locked since last cleared 1 = System PLL has fallen out of lock since last cleared When writes, writing a `1` to this bit causes the PLL3 Unlock event flag to be cleared. |
| 2 | R/w | PLLLock2 When reads, 0 = Comms PLL has been locked since last cleared 1 = Comms PLL has fallen out of lock since last cleared When writes, writing a `1` to this bit causes the PLL2 Unlock event flag to be cleared. |
| 1 | R/w | PLLLock1 When reads, 0 = LCD PLL has been locked since last cleared 1 = LCD PLL has fallen out of lock since last cleared When writes, writing a `1` to this bit causes the PLL1 Unlock event flag to be cleared. |
| 0 | R/w | PORStatus When reads, 0 = No POR since last cleared; 1 = POR since last cleared When writes, writing a `1` to this bit causes the nPOR event flag to be cleared. |

5.3.4 PMU Clock Control Register (PMUCLK)

This register is used to control the frequency of PLL3, the system clock PLL and PLL1, the LCD clock. Six bits are defined which control the frequency of FCLK, and a further bit is used to control the frequency of PLL1, the LCD clock. The Default (Power on Reset) value for this register is 0x2126.

| | | | | | | | |
|-------------|------------------|-----------|----|----|----|---|------------|
| | | | | | | | 0x80001028 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| PLL2 ENABLE | PLL1 ENABLE | PLL1 FREQ | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PLL3 MUTE | PLL3 FREQ UPDATE | PLL3 FREQ | | | | | |

| Bits | Type | Function | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|-------|-------------|---|-----------------------|-----------|-------|-----------|------|-------------|------|-------------|------|-------------|------|-----------------------|------|-------------|------|-------------|------|-------------|------|-------------|------|-------------|------|-------------|------|-------------|------|-------------|------|-------------|------|-------------|
| 31:16 | - | Reserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 15 | R/W | Set for PLL2 enable. Output will be gated until PLL2 Lock Detect (LD) is received. Reset for disable PLL2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 14 | R/W | Set for PLL1 enable. Output will be gated until PLL1 Lock Detect (LD) is received. Reset for disable PLL1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 13:8 | R/W | Same with bit [5:0]. But output clock frequency will be half of PLL3 – default 30.4128 MHz | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 7 | R/W | Reset: PLL3 is muted when Lock detect = 0 (default) Set: PLL3 only muted after nPOR or nRESET. Subsequent unlock condition does not mute the clock. Allows dynamic changes to the clock frequency without halting execution. Care: this only will be legal if PLL3 is under-damped (i.e. will not exhibit overshoot in its lock behavior). | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 6 | R/W | Reset: PLL3 frequency control frequency is only updated when PMU exits DEEP SLEEP mode (default) Set: PLL3 frequency control frequency is updated instantaneously | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 5:0 | R/W | <table border="1"> <thead> <tr> <th>Value</th> <th>Frequency</th> <th>Value</th> <th>Frequency</th> </tr> </thead> <tbody> <tr> <td>0x1B</td> <td>49.7664 MHz</td> <td>0x25</td> <td>68.1984 MHz</td> </tr> <tr> <td>0x1C</td> <td>51.6096 MHz</td> <td>0x26</td> <td>70.0416 MHz - default</td> </tr> <tr> <td>0x1D</td> <td>53.4528 MHz</td> <td>0x27</td> <td>71.8848 MHz</td> </tr> <tr> <td>0x1E</td> <td>55.2960 MHz</td> <td>0x28</td> <td>73.7280 MHz</td> </tr> <tr> <td>0x1F</td> <td>57.1392 MHz</td> <td>0x29</td> <td>75.5712 MHz</td> </tr> <tr> <td>0x20</td> <td>58.9824 MHz</td> <td>0x2A</td> <td>77.4144 MHz</td> </tr> <tr> <td>0x21</td> <td>60.8256 MHz</td> <td>0x2B</td> <td>79.2576 MHz</td> </tr> </tbody> </table> | Value | Frequency | Value | Frequency | 0x1B | 49.7664 MHz | 0x25 | 68.1984 MHz | 0x1C | 51.6096 MHz | 0x26 | 70.0416 MHz - default | 0x1D | 53.4528 MHz | 0x27 | 71.8848 MHz | 0x1E | 55.2960 MHz | 0x28 | 73.7280 MHz | 0x1F | 57.1392 MHz | 0x29 | 75.5712 MHz | 0x20 | 58.9824 MHz | 0x2A | 77.4144 MHz | 0x21 | 60.8256 MHz | 0x2B | 79.2576 MHz |
| Value | Frequency | Value | Frequency | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x1B | 49.7664 MHz | 0x25 | 68.1984 MHz | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x1C | 51.6096 MHz | 0x26 | 70.0416 MHz - default | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x1D | 53.4528 MHz | 0x27 | 71.8848 MHz | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x1E | 55.2960 MHz | 0x28 | 73.7280 MHz | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x1F | 57.1392 MHz | 0x29 | 75.5712 MHz | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x20 | 58.9824 MHz | 0x2A | 77.4144 MHz | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x21 | 60.8256 MHz | 0x2B | 79.2576 MHz | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

| | | | |
|------|-------------|--------------|-------------|
| 0x22 | 62.6688 MHz | 0x2c | 81.1008 MHz |
| 0x23 | 64.5120 MHz | 0x2D | 82.9440 MHz |
| 0x24 | 66.3552 MHz | Other values | Reserved |

IF BIT 6 is `0`

When the CPU writes to bits 5:0 of this register, these bits are stored in a temporary buffer, which is not transferred to the PLL until the next time the PLL lock signal becomes inactive. This means that for a new value to take effect, it is necessary for the device to enter DEEP SLEEP mode first.

IF BIT 6 is `1`

The first effect that writing a new value to bits [5:0] will have is that PLL3 will go out of lock, and the Clock control circuit will immediately inhibit FCLK and BCLK, without first verifying that SDRAM operations have completed.

5.3.5 PMU Debounce Counter Test Register (PMUDBCT)

| | | | 0x80001030 | |
|------|------|---|---------------------|--|
| Bits | Type | Function | | |
| | | Read | Write | |
| 31:9 | - | Reserved | | |
| 8 | W | Reset: Normal operation Set: Forces FCLK and BCLK to be active in all PMU states (test purposes only) | | |
| 7:6 | - | Reserved | | |
| 5 | R | Selected debounce counter bits | | |
| 4 | R/W | Reserved Reset: normal operation Set: disables Bus Request from the PMU to allow CPU to read state machine for test purposes during PMU IDLE state. | | |
| 3 | R/W | Prescaler bits Reset: nTEST takes value from input pin Set: forces local test mode | | |
| 2:0 | R/W | Select Debounce counter for | | |
| | | Value | Function | |
| | | 0x0 | nPMWAKEUP | |
| | | 0x1 | RING event | |
| | | 0x3 | Power Adapter event | |
| | | 0x4 | Warm Reset | |

In order that the debounce counters (which would normally be clocked at 4 kHz) may be independently exercised and observed, the counters may be triggered and observed using the above registers. **These registers are for testing only and are not required in normal use.**

5.3.6 PMU PLL Test Register (PMUPLLTR)

| | | | | | | | | 0x80001038 |
|----------------|--------|----------|-------------------|----------------|--|----|--------------|------------|
| 31 | ... | 21 | 20 | 19 | 18 | 17 | 16 | |
| Reserved | | | Select LCLK, CCLK | Select BCLK | Select PLL Test 01(PLL1), 10(PLL2), 11(PLL3) | | PLL TEST MUX | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | |
| PMUTEST | PWRDN1 | PWRDN2 | PWRDN3 | PLL1 Frequency | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| PLL3 Frequency | | | | | | | | |
| Bits | Type | Function | | | | | | |
| 31:21 | - | Reserved | | | | | | |
| 20 | | | | | | | | |
| 19 | | | | | | | | |

| |
|-------|
| 18:17 |
| 16 |
| 15 |
| 14 |
| 13 |
| 12 |
| 11:6 |
| 5:0 |

5.4 Timings

5.4.1 Reset Sequences of Power On Reset

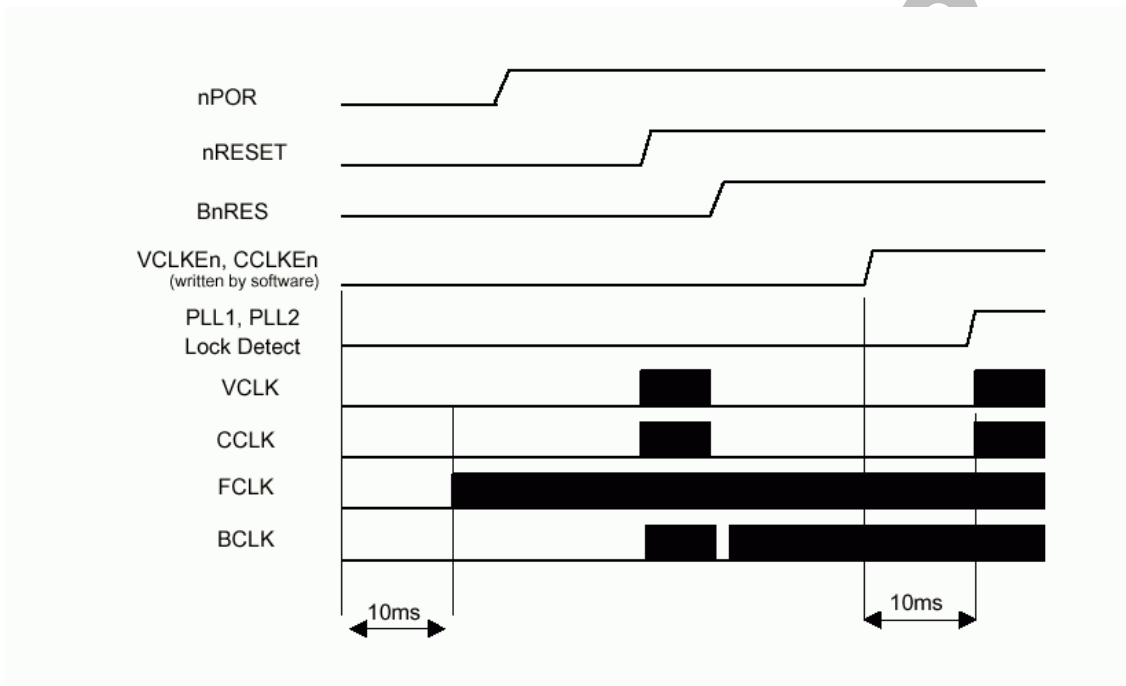


Figure 5-2 PMU Cold Reset Event

In the event of removal and re-application of all power to the HMS30C7202, the following sequence may be typical:

- nPOR input is active. All internal registers are reset to their default values. The PMU drives nRESETout LOW to reset any off-chip peripheral devices.
- BnRES becomes active on exit from the nPOR condition. Clocks are enabled temporarily to allow synchronous resets to operate.
- The default frequency of FCLK on exit from nPOR will be 70.0416 MHz.
- When FCLK is stable, the CPU clock is released. If the CPU were to read the RESET/Status register at this time, it will return 0x10f as a initial value.
- If you are to clear these flag bits, write 0x10f to the RESET register. (Refer 5.3.4 PMU Reset/PLL Status Register).
- The CPU writes 0x20 to the clock control register, which will set a FCLK speed of 58.9824MHz. The new clock frequency, however, is not adopted until the
- PMU has entered and left DEEP SLEEP mode.
- The CPU sets a RTC timer alarm to expire in approximately 2 seconds
- The CPU sets DEEP SLEEP into the PMU Mode Register
- The PMU state machine will enter DEEP SLEEP mode (via the intermediate states shown in Figure 5-1: Power Management State Diagram).
- When the RTC timer alarm is activated, the PMU automatically wakes up into SLOW mode, but with the new FCLK

frequency of 58.9824Mhz.

- The CPU may write 0xE120 to the Clock Control register, which enables CCLK and VCLK, and retains the new FCLK frequency.

| Bit | Meaning |
|------------|-----------------------------------|
| Bit 0 set: | Power On Reset event has occurred |
| Bit 1 set: | PLL1 has been `unlocked' |
| Bit 2 set: | PLL2 has been `unlocked' |
| Bit 3 set: | PLL3 has been `unlocked' |

Table 5-2 PMU Bit Settings for a cold Reset Event within PMUSTAT Register

5.4.2 Software Generated Warm Reset

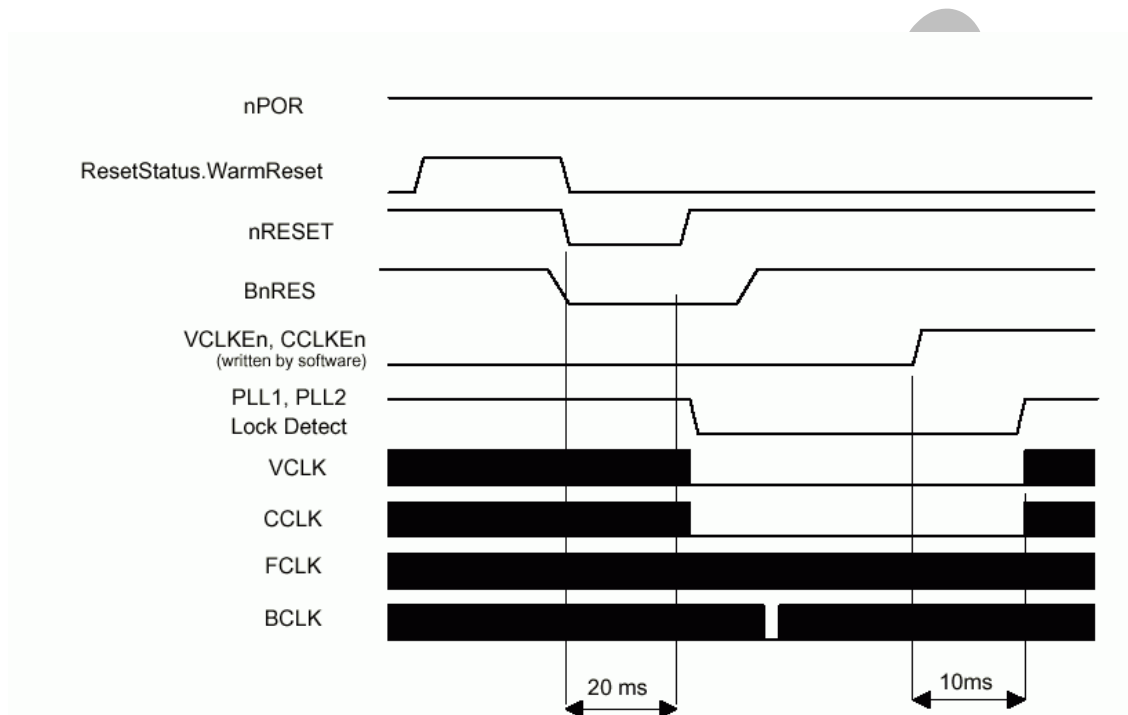


Figure 5-3 PMU Software Generated Warm Reset

The CPU writes `1' to the WarmReset bit of PMUSTAT register. The PMU drives nRESET low. The internal chip reset, BnRES is driven low. The PMU detects that the bi-directional nRESET pin is low. nRESET is filtered by a de-bounce circuit. Note that this means that nRESET will remain low for a minimum of 16ms. BnRES becomes active once the de-bounced nRESET goes high once more, which disables PLL1 and PLL2. The CPU may read the PMUSTAT register, which will return 0x106:

| Bit | Meaning |
|------------|-----------------------------|
| Bit 1 set: | PLL1 has been `unlocked' |
| Bit 2 set: | PLL2 has been `unlocked' |
| Bit 8 set: | A RESET event has occurred. |

Table 5-3 PMU Bit Settings for a Software Generated Warm Reset within PMUSTAT Register

5.4.3 An Externally generated Warm Reset

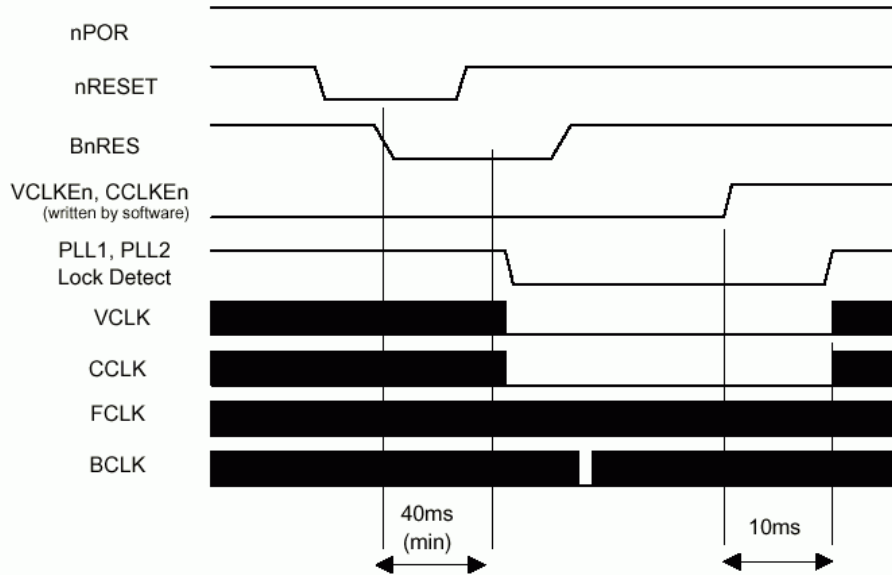


Figure 5-4 PMU An Externally Generated Warm Reset

nRESET is driven to '0' by external hardware. The nRESET input is filtered by a de-bounce circuit. Note that this means that nRESET must remain low for a minimum of 40ms. BnRES (the on-chip reset signal) becomes active as soon as nRESET is low, and high once the de-bounced nRESET goes high once more. BnRES disables PLL1 and PLL2. The CPU may read the RESET register, which will return 0x106:

| Bit | Meaning |
|------------|-----------------------------|
| Bit 1 set: | PLL1 has been 'unlocked' |
| Bit 2 set: | PLL2 has been 'unlocked' |
| Bit 8 set: | A RESET event has occurred. |

Table 5-4 PMU Bit Settings for a Warm Reset within PMUSTAT Register

Note

The internal chip reset, BnRES, remains active for 20ms after an externally generated nRESET. External devices should not assume that the HMS30C7202 is in an active state during this period.

6 SDRAM CONTROLLER

The SDRAM controller operates at the full CPU core frequency (FCLK = SCLK) and is connected to the core via the ASB bus. Internally the SDRAM controller arbitrates between access requests from the main AMBA bus, and the video bus.

It can control up to two SDRAMs of 16Mx16 density maximum. To reduce the system power consumption it can power down these individually using the Clock Enable (CKE). When the MCU is in standby mode the SDRAMs are powered down into self-refresh mode.

SDRAMs achieve the highest throughput when accessed sequentially – like video data. However accesses from the core are less regular. The SDRAM controller uses access predictability to maximize the memory interface bandwidth by having access to the LCD address buses.

Video accesses to the SDRAM occur in fixed-burst lengths of 16 words; At each Video access, SDRAM controller issues 4 consecutive "Read" commands of which burst length is 8 half-word. So, If you want to get the successive 16 words, the start address of SDRAM read must be arranged to 4-Word(8-HalfWord) boundary - The start address of SDRAM must be 0xXXXX_XXX0.

ARM and DMA controller accesses occur in a fixed-burst length of four words. If the requested accesses are shorter than four words, then the extra data is ignored. In Addition, ARM/DMA Access SDRAM Controller discards the data of which the address is not sequentially increased. For example, If ARM do the 4-Word "ldm(load Block data)" of which start address is 0x4000_0004, the Address output from SDRAM Controller to SDRAM is start from 2 (just 4bits from LBS). SDRAM do the 8-HalfWord Burst Read and it's address sequence is 2-3-4-5-6-7-0-1. In that case, SDRAM Controller discards data from address 0,1 and just get the 6-HalfWord Data(Address from 2 to 7). After that, SDRAM Controller issue the "Read" Command again of which Start address to SDRAM is 8 and gets the 2-HalfWord data(data from SDRAM address 8,9).

FEATURES

- 16 Bits wide external bus interface (two access requires for each word)
- Supports 16/64/128/256Mbit device
- Supports 2~64 Mbytes in up to two devices (the size of each memory device may be different)
- Programmable CAS latency
- Supports 2/4 banks with page lengths of 256 or 512 half words
- Programmable Auto Refresh Timer
- Support low power mode when IDLE (each device's CKE is disable individually).
- Support External Device interface with DMA channel 2.

6.1 Supported Memory Devices

2-64Mbytes of SDRAM are supported with any combination of one or two 16/64/128/256Mbit devices. Each device is mapped to a 32 Mbyte address space. The MMU (memory management unit) maps different device combinations (e.g. 16- and 64Mbit devices) into a continuous address space for the ARM core. Note that 16Mbit devices appear eight times, and 64Mbit devices appear twice in the memory map.

| Total Memory | 16Mbit devices | 64Mbit devices | 128Mbit devices | 256Mbit devices |
|--------------|----------------|----------------|-----------------|-----------------|
| 2Mbyte | 1 | - | - | - |
| 4Mbyte | 2 | - | - | - |
| 8Mbyte | - | 1 | - | - |
| 16Mbyte | - | 2 | 1 | - |
| 32Mbyte | - | - | 2 | 1 |
| 64Mbyte | - | - | - | 2 |

Note

The MMU (memory management unit) must be programmed according to the actual memory configuration (combination of 16/64/128/256 Mbit SDRAMs).

The SDRAM controller allows up to four memory banks to be open simultaneously. The open banks may exist in different physical SDRAM devices.

6.2 Registers

The SDRAM controller has four registers: the configuration, refresh timer, the Write Buffer Flush timer and wait driver. The configuration register's main function is to specify the number of SDRAMs connected, and whether they are 2- or 4-bank devices. The refresh timer gives the number of BCLK ticks that need to be counted in-between each refresh period. The Write Buffer Flush timer is used to set the number of BCLK ticks since the last write operation, before the write buffer's contents are transferred to SDRAM. The wait driver is used to set wait delay for external slow device.

| Address | Name | Width | Default | Description |
|-------------|--------|-------|------------|-------------------------------|
| 0x8000.0000 | SDCON | 32 | 0x00700000 | Configuration register |
| 0x8000.0004 | SDREF | 16 | 0x0080 | Refresh timer |
| 0x8000.0008 | SDWBF | 3 | 0x1 | Write back buffer flush timer |
| 0x8000.000C | SDWAIT | 4 | 0x1 | Wait driver register |

Table 6-1 SDRAM Controller Register Summary

In addition to the SDRAM control registers, the ARM may access the SDRAM mode registers by writing to a 64MByte address space referenced from the SDRAM mode register base address. Writing to the SDRAM mode registers is discussed further in 오류! 참조 원본을 찾을 수 없습니다. 오류! 참조 원본을 찾을 수 없습니다.

6.2.1 SDRAM Controller Configuration Register (SDCON)

| 31 | 30 | ... | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | ... | 7 | 6 | ... | 3 | 2 | ... | 0x8000.0000 |
|-------|------|---|----|----|----|----|----|----|----|----|-----|----|----|-----|----|----|-----|-------------|
| S1 | S0 | - | W | R | A | C1 | C0 | D | C | B | - | E1 | B1 | - | E0 | B0 | - | |
| Bits | Type | Function | | | | | | | | | | | | | | | | |
| 31:30 | R | SDRAM controller Status 11:Reserved 10:Self refresh 01:Busy 00:Idle | | | | | | | | | | | | | | | | |
| 24 | R/W | Wait driver enable bit for test purpose | | | | | | | | | | | | | | | | |
| 23 | R/W | Normal SDRAM controller refresh enable 1 = the SDRAM controller provides refresh control 0 = the SDRAM controller does not provide refresh | | | | | | | | | | | | | | | | |
| 22 | R/W | Auto pre-charge on ASB accesses 1 = auto pre-charge (default) 0 = no auto pre-charge If auto pre-charge is enabled, SDRAM controller issues "Read/Write with Auto Pre-charge" command instead of normal "Read/Write" command. So, SDRAM controller generates "Active" command before each Read/Write operation. If auto-pre-charge is disabled, SDRAM controller uses normal "Read/Write" command and SDRAM page that is accessed before remains active. So, SDRAM Controller automatically issues "Pre-charge" command only in the case that One SDRAM page is active and there is need to read/write the other page address in the same bank. You had better disable auto pre-charge bit, if a number of SDRAM accesses occur in the same page boundary - You can perform SDRAM "Read/Write" command fastly without "Pre-charge" & "Active" command. | | | | | | | | | | | | | | | | |
| 21:20 | R/W | 11:CAS latency3 10:CAS latency2 01:CAS latency1 00:Reserved | | | | | | | | | | | | | | | | |
| 19 | R/W | SDRAM bus tri-state control 0 = the controller drives the last data onto the SDRAM data bus (default) 1 = the SDRAM bus is tri-stated except during writes | | | | | | | | | | | | | | | | |

| | | |
|----|-----|--|
| | | This bit should be cleared before the IC enters a low power mode. Driving the data lines avoids floating inputs that could increase device power consumption. During normal operation the D bit should be set, to avoid data bus drive conflicts with SDRAM. |
| 18 | R/W | SDRAM clock enable control 0 = the clock of IDLE devices are disabled to save power (default) 1 = all clock enables are driven HIGH continuously |
| 17 | R/W | Write buffer enable Value = 1 if the write buffer is enabled Value = 0 if the write buffer is disabled |
| 7 | R/W | 1 = a device is present at address range 32-64Mbyte 0 = no device present at address range 32-64Mbyte The bit E is used to control the auto-refresh |
| 6 | R/W | Specifies the number of banks of the SDRAM at address range 32-64Mbyte 1 = the SDRAM is a four-bank device 0 = the SDRAM is a two-bank device |
| 3 | R/W | 1 = a device is present at address range 0-32MByte 0 = no device present at address range 0-32Mbyte The bit E is used to control the auto-refresh |
| 2 | R/W | Specifies the number of banks of the SDRAM at address range 0-32Mbyte 1 = the SDRAM is a four-bank device 0 = the SDRAM is a two-bank device |

The SDRAM controller powers-up with E[1:0]=00 and R=0. This indicates that the memory interface is IDLE. Next, the software should set at least one E bit to 1 with the R bit 0. This will cause both devices to be precharged (if present). The next operation in the initialization sequence is to auto-refresh the SDRAMs. Note that the number of refresh operations required is device-dependent. Set R=1 and E[1:0]=00 to start the auto-refresh process. Software will have to ensure that the prescribed number of refresh cycles is completed before moving on to the next step. The final step in the sequence is to set R=1 and to set the E bits corresponding to the populated slots. This will put the SDRAM controller (and the SDRAMs) in their normal operational mode. After that SDRAM mode register (in the SDRAM, not SDCON) must be initialized as to Write Burst Mode = "Programmed Burst Length", Burst Type = "Sequential", Burst Length = "8".

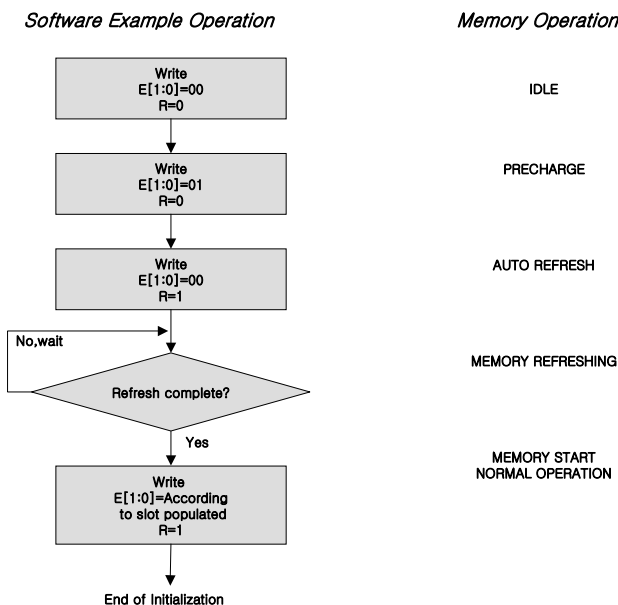


Figure 6-1 SDRAM Controller Software Example and Memory Operation Diagram

6.2.2 SDRAM Controller Refresh Timer Register (SDREF)

| | | | | | |
|----------|--|--|--------|--|-------------|
| | | | | | 0x8000.0004 |
| - | | | 15 - 0 | | |
| Reserved | | | SDREF | | |

| Bits | Type | Function |
|------|------|--|
| 15:0 | R/W | <p>A 16-bit read/write register that is programmed with the number of BCLK ticks that should be counted between SDRAM refresh cycles. For example, for the common refresh period of 16μs, and a BCLK frequency of 50MHz, the following value should be programmed into it:</p> $16 \times 10^{-6} * 50 \times 10^6 = 800$ <p>The refresh timer defaults to a value of 128, which for a 16μs refresh period assumes a worst case (i.e. slowest) clock rate of:</p> $128 / (16 \times 10^{-6}) = 8 \text{ MHz}$ <p>The refresh register should be programmed as early as possible in the system start-up procedure, and in the first few cycles if the system clock is less than 8MHz.</p> |

6.2.3 SDRAM Controller Write buffer flush timer Register (SDWBF)

| | | | | | |
|----------|--|--|-------|--|-------------|
| | | | | | 0x8000.0008 |
| - | | | 2 - 0 | | |
| Reserved | | | SDWBF | | |

| Bits | Type | Function | | | | | | | | | | | | | | | | | | |
|-------------|------------------------------|---|-------------|------------------------------|-----|-----|-----|----|-----|----|-----|----|-----|---|-----|---|-----|---|-----|-------------------|
| 2:0 | R/W | <p>A 3-bit read/write register that sets the time-out value for flushing the quad word merging write buffer. The times are given in the following table.</p> <table border="1" style="margin-left: 20px; border-collapse: collapse;"> <thead> <tr> <th>Timer value</th> <th>BCLK ticks between time-outs</th> </tr> </thead> <tbody> <tr><td>111</td><td>128</td></tr> <tr><td>110</td><td>64</td></tr> <tr><td>101</td><td>32</td></tr> <tr><td>100</td><td>16</td></tr> <tr><td>011</td><td>8</td></tr> <tr><td>010</td><td>4</td></tr> <tr><td>001</td><td>2</td></tr> <tr><td>000</td><td>Time-out disabled</td></tr> </tbody> </table> | Timer value | BCLK ticks between time-outs | 111 | 128 | 110 | 64 | 101 | 32 | 100 | 16 | 011 | 8 | 010 | 4 | 001 | 2 | 000 | Time-out disabled |
| Timer value | BCLK ticks between time-outs | | | | | | | | | | | | | | | | | | | |
| 111 | 128 | | | | | | | | | | | | | | | | | | | |
| 110 | 64 | | | | | | | | | | | | | | | | | | | |
| 101 | 32 | | | | | | | | | | | | | | | | | | | |
| 100 | 16 | | | | | | | | | | | | | | | | | | | |
| 011 | 8 | | | | | | | | | | | | | | | | | | | |
| 010 | 4 | | | | | | | | | | | | | | | | | | | |
| 001 | 2 | | | | | | | | | | | | | | | | | | | |
| 000 | Time-out disabled | | | | | | | | | | | | | | | | | | | |

6.2.4 SDRAM Controller Wait Driver Register (SDWAIT)

| | | | | | |
|----------|--|--|--------|--|-------------|
| | | | | | 0x8000.000C |
| - | | | 3 - 0 | | |
| Reserved | | | SDWAIT | | |

| Bits | Type | Function |
|------|------|--|
| 3:0 | R/W | <p>This value specifies the waited delay time (BCLK cycles) of the BWAIT signal of the system bus (AMBA ASB); default value is 1. This register affects only the external device with DMA channel-2 operation and does not affect channel-0 and channel-1. During access to the external device with DMA channel-2, Write-Back buffer is always enable even if SDCON (SDRAM Controller Configuration Register)'s W bit (Write-Back buffer enable) is reset (disabling the operation of Write-Back Buffer).</p> |

6.3 Power-up Initialization of the SDRAMs

The SDRAMs are initialized by applying power, waiting a prescribed amount of settling time (typically 100 μ s), performing at least 2 auto-refresh cycles and then writing to the SDRAM mode register. The exact sequence is

SDRAM device-dependent.

The settling time is referenced from when the SDRAM CLK starts. The processor should wait for the settling time before enabling the SDRAM controller refreshes, by setting the R bit in the SDRAM control register. The SDRAM controller automatically provides an auto refresh cycle for every refresh period programmed into the Refresh Timer when the R bit is set. The processor must wait for sufficient time to allow the manufacturer's specified number of auto-refresh cycles before writing to the SDRAM's mode register.

The SDRAM's mode register is written to via its address pins (A[14:0]). Hence, when the processor wishes to write to the mode register, it should read from the binary address (AMBA address bits [24:9]), which gives the binary pattern on A[14:0] which is to be written. The mode register of each of the SDRAMs may be written to by reading from a 64Mbyte address space from the SDRAM mode register base address. The correspondence between the AMBA address bits and the SDRAM address lines (A[14:0]) is given in the Row address mapping of 오류! 참조 원본을 찾을 수 없습니다.. Bits [25] of the AMBA address bus select the device to be initialized.

The SDRAM must be initialized to have the same CAS latency as is programmed into C[1:0] bits of the SDRAM control register, and always to have a burst length of 8.

6.4 SDRAM Memory Map

The SDRAM controller can interface with up to two SDRAMs of 1Mx16, 4Mx16, 8Mx16 or 16Mx16 density. The SDRAMs may be organized in either two or four banks. The controller can address 64Mbyte, subdivided into two 32Mbyte blocks, one for each SDRAMs.

The mapping of the AMBA address bus to the SDRAM row and column addresses is given in 오류! 참조 원본을 찾을 수 없습니다.. The first row of the diagram indicates the SDRAM Controller Address output (SA[14:0]) and the SDRAM address bit (BS1, BS0,A12~A0); If you use 64Mbit SDRAM, you should connect A11~A0 to SA[11:0] and BS0~1 to SA[13:12].

The remaining numbers indicate the AMBA address bits MBA[24:1].

| SDRAM ADDR | SA[14] A12 | SA[13] BS0 | SA[12] BS1 | SA[11] A11 | SA[10] A10 | SA[9] A9 | SA[8] A8 | SA[7] A7 | SA[6] A6 | SA[5] A5 | SA[4] A4 | SA[3] A3 | SA[2] A2 | SA[1] A1 | SA[0] A0 |
|-------------|------------|------------|------------|------------|------------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|
| Row 16Mbit | 24 | 10* | 9* | 22 | 20* | Note 1 | 19* | 18* | 17* | 16* | 15* | 14* | 13* | 12* | 11* |
| Col 16Mbit | 24 | 10* | 9* | Note1 | 20 | Note 1 | 23 | 8* | 7* | 6* | 5* | 4* | 3* | 2* | Note2 |
| Row 64Mbit | 24 | 10* | 9* | 22* | 20* | 21* | 19* | 18* | 17* | 16* | 15* | 14* | 13* | 12* | 11* |
| Col 64Mbit | 24 | 10* | 9* | 22 | 20 | 21 | 23 | 8* | 7* | 6* | 5* | 4* | 3* | 2* | Note2 |
| Row 128Mbit | 24 | 10* | 9* | 22* | 20* | 21* | 19* | 18* | 18* | 16* | 15* | 14* | 13* | 12* | 11* |
| Col 128Mbit | 24 | 10* | 9* | 22 | 20 | 21 | 23* | 8* | 7* | 6* | 5* | 4* | 3* | 2* | Note2 |
| Row 256Mbit | 24* | 10* | 9* | 22* | 20* | 21* | 19* | 18* | 18* | 16* | 15* | 14* | 13* | 12* | 11* |
| Col 256Mbit | 24 | 10* | 9* | 22 | 20 | 21 | 23* | 8* | 7* | 6* | 5* | 4* | 3* | 2* | Note2 |
| Mode Write | 24* | 10* | 9* | 22* | 20* | 21* | 19* | 18* | 17* | 16* | 15* | 14* | 13* | 12* | 11* |
| Summary | 24 | 10 | 9 | 22 | 20 | 21 | 19/23 | 18/8 | 17/7 | 16/6 | 15/5 | 14/4 | 13/3 | 12/2 | 11* |

Table 6-2 SDRAM Row/Column Address Map

Notes

(1) For the 16Mbit device, SDRAM address line A11 should be connected to the HMS30C7202 pin SA[13](BS0), and the SDRAM address line A9 should be connected to the HMS30C7202 pin SA[12](BS1). The HMS30C7202 address lines SA[11] and SA[9] should not be connected.

(2) Since all burst accesses commence on a word boundary, and SDRAM addresses are non-incrementing (the address incremented is internal to the device), column address zero will always be driven to logic '0'.

* An asterisk denotes the address lines that are used by the SDRAM.

The start address of each SDRAM is fixed to a 32Mbyte boundary. The memory management unit will be used

to map the actual banks that exist into contiguous memory as seen by the ARM. Bits [25] of the AMBA address bus select the device to be initialized, as described in 오류! 참조 원본을 찾을 수 없습니다..

| A25 | Device selected |
|-----|-----------------|
| 0 | Device 0 |
| 1 | Device 1 |

Table 6-3 SDRAM Device Selection

6.5 AMBA Accesses and Arbitration

The SDRAM controller bridges both the AMBA Main and Video buses. On the Main bus, the SDRAM appears as a normal slave device. On the Video DMA bus, the SDRAM controller integrates the functions of the bus arbiter and address decoder. Writes from the main bus may be merged in the quad word merging write buffer. A Main/Video arbiter according to the following sequence arbitrates access requests from either the Main or Video buses:

- Highest Priority: LCD Refresh request
- Lowest Priority: Main bus peripheral (PMU, ARM, DMA)--order determined by Main bus arbiter.

Video SDRAM accesses always occur in bursts of 16 words. Once a burst has started, the SDRAM controller provides data without wait states. Video data is only read from SDRAM, no write path is supported.

If a refresh cycle is requested, then it will have lower priority than the Video bus, but will be higher than any other accesses from the Main bus. Assuming a worst-case BCLK frequency of 8MHz, the maximum, worst-case latency that the arbitration scheme enforces is 11.5us before a refresh cycle can take place. This is comfortably within the 16us limit. Note that the 2 external SDRAM devices are refreshed on 2 consecutive clock cycles to reduce the peak current demand on the power source.

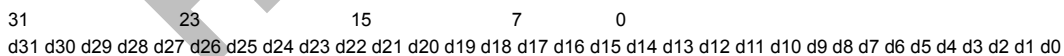
The arbitration of the Main bus is left to the Main bus arbiter. Data transfers requested from the Main bus always occur as a burst of eight half-word accesses to SDRAM. The Main bus arbiter cannot break into access requests from the Main bus. In the case where fewer than four words are actually requested by the Main bus peripheral, the excess data from the SDRAM is ignored by the SDRAM controller in the case of read operations, or masked in the case of writes. In the case where more than four words are actually requested by the Main bus peripheral, the SDRAM controller asserts BLAST to force the ASB decoder to break the burst.

In the case of word/half-word/byte misalignment to a quad word boundary (when any of address bits [3:0] are non-zero at the start of the transfer), BLAST is asserted at the next quad word boundary (bits 3, 2, 1 and 0 properly set 1 for each type) to force the ASB decoder to break the burst.

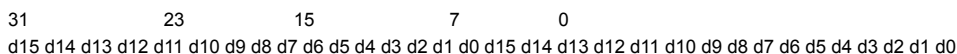
Sequential half word (or byte) reads are supported and the controller asserting BLAST at quad word boundary.

In the case of byte or half word reads, data is replicated across the whole of the ASB data bus.

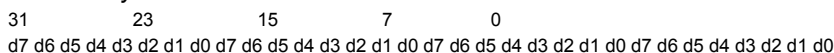
Data bus for word access:



Data bus for half word access:



Data bus for byte access:



6.6 Merging Write Buffer

An eight word merging Write-Buffer is implemented in the SDRAM controller to improve write performance. The write buffer can be disabled, but its operation is completely transparent to the programmer. The eight words of the buffer are split into two quad words, the same size as all data transactions to the SDRAMs. The

split into two quad words allows one quad word to be written to at the same time as the contents of the other are being transferred to SDRAM. The quad word buffer currently being written to may be accessed with non-contiguous word, half word or byte writes, which will be merged into a single quad word. The buffered quad word will be transferred to the SDRAM when:

- There is a write to an SDRAM address outside the current quad word being merged into
- There is a read to the address of the quad word being merged into
- There is a time-out on the write back timer

The two quad-words that make up the write buffer operate in "ping-pong" fashion, whereby one is initially designated the buffer for writes to go into, and the other is the buffer for write backs. When one of the three events that can cause a write-back occurs, the functions of the two buffers are swapped. Thus the buffer containing data to be written back becomes the buffer that is currently writing back, and the buffer that was the write-back buffer becomes the buffer being written to.

In the case of a write-back initiated by a read from the same address as the data in the merge buffer, the quad word in the buffer is written to SDRAM, and then the read occurs from SDRAM. The write before read is essential, because not all of the quad word in the buffer may have been updated, so its contents need to be merged with the SDRAM contents to fill any gaps where the buffer was not updated. The write buffer flush timer forces a write back to occur after a programmable amount of time. Every time a write into the buffer occurs, the counter is re-loaded with the programmed time-out value, and starts to counts down. If a time-out occurs, then data in the write buffer is written to SDRAM.

HMS30C7202

7 STATIC MEMORY INTERFACE

The Static Memory Controller (SMI) interfaces the AMBA Advanced System Bus (ASB) to the External Bus Interface (EBI). It controls four separate memory or expansion banks. Each bank is 32MB in size and can be programmed individually to support:

- 8-, 16- or 32-bit wide, little-endian memory
- Variable wait states (up to 16)
- Burst mode read access

Burst mode access allows fast sequential access within quad word boundaries. This can significantly improve bus bandwidth in reading from memory (that must support at least four word burst reads). In addition, bus transfers can be extended using the EXPRDY input signal.

7.1 External Signals

| Pin Name | Type | Description |
|----------------|------|---|
| EXPRDY | I | Expansion channel ready. When LOW, during phase one this signal will force the current memory transfer to be extended. |
| nRWE [3:0] | O | These signals are active LOW write enables for each of the memory byte lanes on the external bus. |
| nROE | O | This is the active LOW output enable for devices on the external bus. |
| nRCS [3:0] | O | Active LOW chip selects. |
| RA [24:0] | O | ROM Address Bus |
| RD [31:0] | I/O | ROM Data Bus |
| BOOTSBIT [1:0] | I | Configuration input. 00 - Select bank 0 as 32-bit memory 01 - Select bank 0 as 16-bit memory 10 - Select bank 0 as 8-bit memory 11 - Reserved |

7.2 Functional Description

The main functions of the Static Memory Controller (SMI) are :

- Memory bank select
- Access sequencing
- Wait states generation
- Burst read control
- Byte lane write control

These are described below

7.2.1 Memory bank select

| Start Address | Address (Hex) | Size | Description |
|---------------|---------------|----------|-------------------|
| 0 Mbytes | 0x0000.0000 | 32Mbytes | ROM chip select 0 |
| 64 Mbytes | 0x0400.0000 | 32Mbytes | ROM chip select 1 |
| 128 Mbytes | 0x0800.0000 | 32Mbytes | ROM chip select 2 |
| 192 Mbytes | 0x0C00.0000 | 32Mbytes | ROM chip select 3 |

7.2.2 Access sequencing

The bank configuration also determines the width of the external memory devices. When the external memory bus is narrower than the transfer initiated from the current master, the internal transfer will take several external bus transfers to complete. For example, in case that memory Bank0 is configured as 8-bit wide

memory and a 32-bit read is initiated the AMBA bus stalls while the SMI read four consecutive bytes from the memory. During these accesses the data path is controlled (in the EBI) to demultiplex the four bytes into one 32-bit word on the AMBA ASB bus.

7.2.3 Wait states generation

The Static Memory Controller supports wait states for read and write accesses. This is configurable between one and 16 wait states for standard memory access, and zero and 15 wait states for burst mode. The Static Memory Controller also allows transfers to be extended indefinitely, using the EXPRDY signal. To hold the current transfer, EXPRDY must be LOW on the falling edge of BCLK before the last cycle of the accesses. The transfer cannot complete until EXPRDY is HIGH for at least one cycle.

7.2.4 Burst read control

Up to four consecutive locations in 8-, 16- or 32-bit memories can be read in one burst. If the bus width of external memory is less than that of internal bus, you have to set the value of **BURST READ WAIT STATE** in 7.3.1 MEM Configuration Register more than 1 cycle for stable data transfers between them.

7.2.5 Byte lane write control

This controls nRWE [3:0] according to transfer width, BA [1:0] and the access sequencing. The table below shows nRWE coding case by little endian accessing to 32,16,8-bit external memory bus.

CASE1. ACCESS: Write, 32-Bit external bus

| BSIZE [1:0] | BA [1:0] | nRWE [3:0] |
|-------------|----------|------------|
| 10(WORD) | XX | 0000 |
| 01(HALF) | 1X | 0011 |
| | 0X | 1100 |
| 00(BYTE) | 11 | 0111 |
| | 10 | 1011 |
| | 01 | 1101 |
| | 00 | 1110 |

CASE2. ACCESS: Write, 16-Bit external bus

| BSIZE [1:0] | BA [1:0] | IA [1:0] ^{†1} | nRWE [3:0] |
|-------------|----------|------------------------|------------|
| 10(WORD) | XX | 1X | 1100 |
| | XX | 0X | 1100 |
| 01(HALF) | 1X | 1X | 1100 |
| | 0X | 0X | 1100 |
| 00(BYTE) | 11 | 1X | 1101 |
| | 10 | 1X | 1110 |
| | 01 | 0X | 1101 |
| | 00 | 0X | 1110 |

CASE3. ACCESS: Write, 8-Bit external bus

| BSIZE [1:0] | BA [1:0] | IA [1:0] ^{†1} | nRWE [3:0] |
|-------------|----------|------------------------|------------|
| 10(WORD) | XX | 11 | 1110 |
| | XX | 10 | 1110 |
| | XX | 01 | 1110 |
| | XX | 00 | 1110 |
| 01(HALF) | 1X | 11 | 1110 |
| | 1X | 10 | 1110 |

| | | | |
|----------|----|----|------|
| | 0X | 01 | 1110 |
| | 0X | 00 | 1110 |
| 00(BYTE) | 11 | 11 | 1110 |
| | 10 | 10 | 1110 |
| | 01 | 01 | 1110 |
| | 00 | 00 | 1110 |

Note *1 IA [1:0] : internal SMI address

7.3 Registers

| Address | Name | Width | Default | Description |
|-------------|---------|-------|---------|---------------------------------|
| 0x8000.3000 | MEMCFG0 | | 0x0 | Memory Configuration Register 0 |
| 0x8000.3004 | MEMCFG1 | | 0x0 | Memory Configuration Register 1 |
| 0x8000.3008 | MEMCFG2 | | 0x0 | Memory Configuration Register 2 |
| 0x8000.300C | MEMCFG3 | | 0x0 | Memory Configuration Register 3 |

Table 7-1 Static Memory Controller Register Summary

7.3.1 MEM Configuration Register

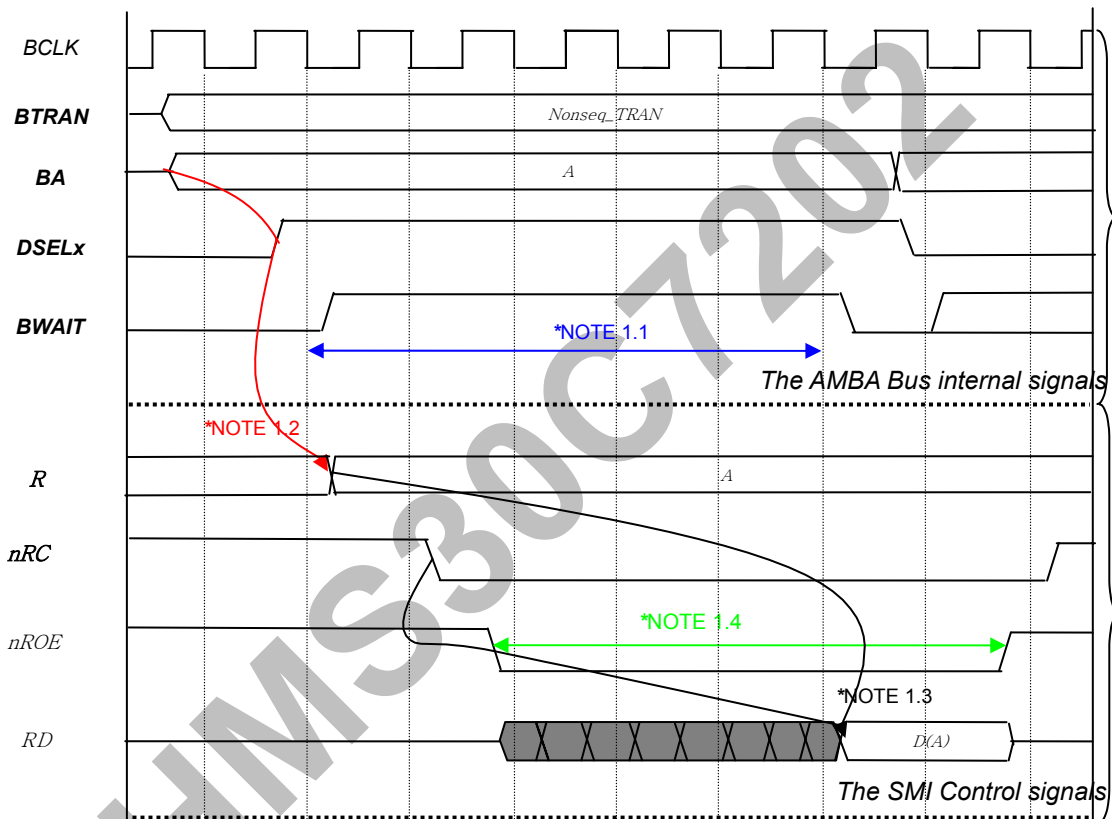
| | | | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-------|------|--|------------------------------------|-----------------------|---|---|---|--------------------------|---|---|---|-----------|---|---|
| | | | BUR EN | BURST READ WAIT STATE | | | | NORMAL ACCESS WAIT STATE | | | | MEM WIDTH | | |
| Bits | Type | Function | | | | | | | | | | | | |
| 31:12 | - | Reserved | | | | | | | | | | | | |
| 11 | R/W | Burst Enable. Setting this bit enables burst reads to take advantage of faster access times from memory devices that support burst mode. | | | | | | | | | | | | |
| 10:7 | R/W | Value | Number of Burst Read Wait State | | | | | | | | | | | |
| | | 1111 | 0 | | | | | | | | | | | |
| | | 1110 | 1 | | | | | | | | | | | |
| | | ... | ... | | | | | | | | | | | |
| | | 0001 | 14 | | | | | | | | | | | |
| | | 0000 | 15 (default) | | | | | | | | | | | |
| 6:3 | R/W | Value | Number of Normal Access Wait State | | | | | | | | | | | |
| | | 1111 | 1 | | | | | | | | | | | |
| | | 1110 | 2 | | | | | | | | | | | |
| | | ... | ... | | | | | | | | | | | |
| | | 0001 | 15 | | | | | | | | | | | |
| | | 0000 | 16 (default) | | | | | | | | | | | |
| 2 | - | Reserved | | | | | | | | | | | | |
| 1:0 | R/W | Value | Memory Width | | | | | | | | | | | |
| | | 11 | Reserved | | | | | | | | | | | |
| | | 10 | 8 bit memory access | | | | | | | | | | | |
| | | 01 | 16 bit memory access | | | | | | | | | | | |
| | | 00 | 32 bit memory access | | | | | | | | | | | |

7.4 Examples of the SMI Read, Write wait timing diagram

The following timing diagrams show sequential and non-sequential read and write accesses. For information on the AMBA bus internal signals refer to the AMBA specification (ARM IHI 0011A)

7.4.1 Read normal wait (Non-Sequential mode)

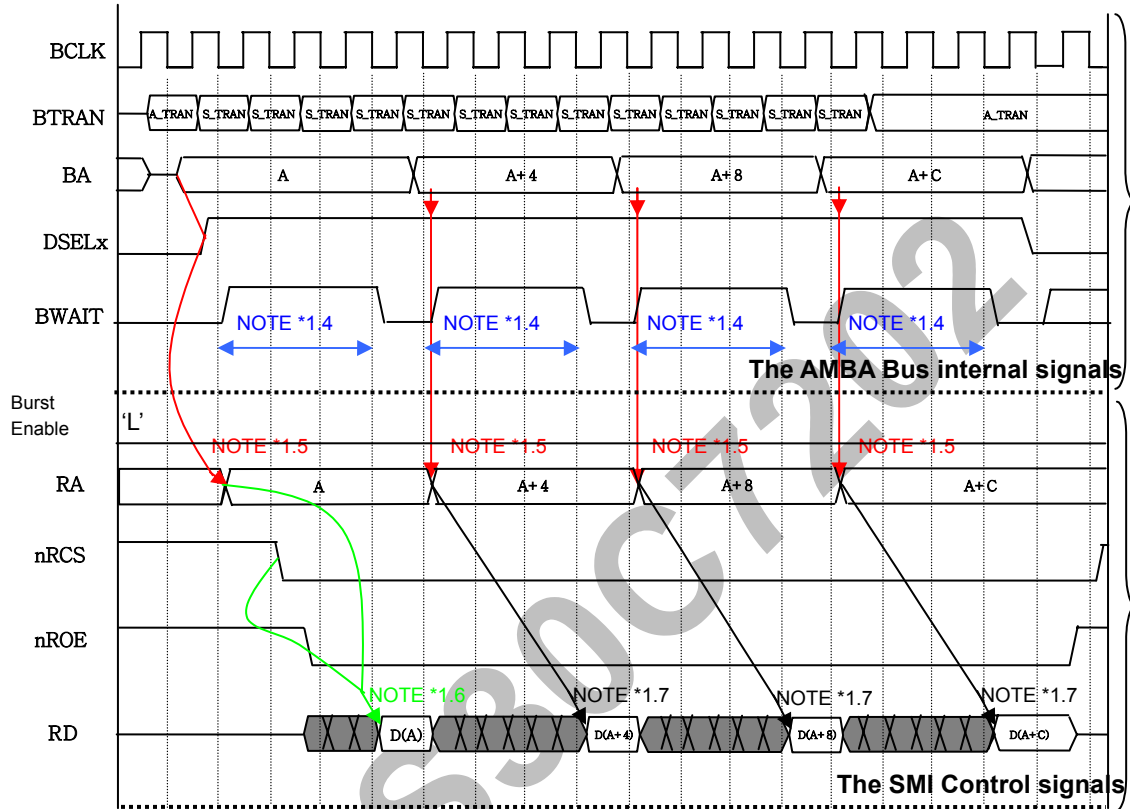
This timing diagram shows a non-sequential read accesses with 5 wait cycles (MEM config register = 0x058).



- *NOTE 1.1: BWAIT time = BCLK x 5 wait cycle
- *NOTE 1.2: Valid the SMI address latch on the ASB Bus address when BA and DSEL are valid condition.
- *NOTE 1.3: After generated SMI control signals and the end of 5wait cycles, external device read data is valid with SMI address (RA), nRCS, and nROE.
- *NOTE 1.4: External Memory access time. It is the same as Wait time (i.e. BWAIT cycle time = 5 wait cycle)

7.4.2 Read normal wait (Sequential mode)

This timing diagram shows a sequential read accesses with 3 wait cycles (MEM config register = 0x068)



*NOTE 1.4: BWAIT time = BCLK x 3 wait cycle (If MCR is set)

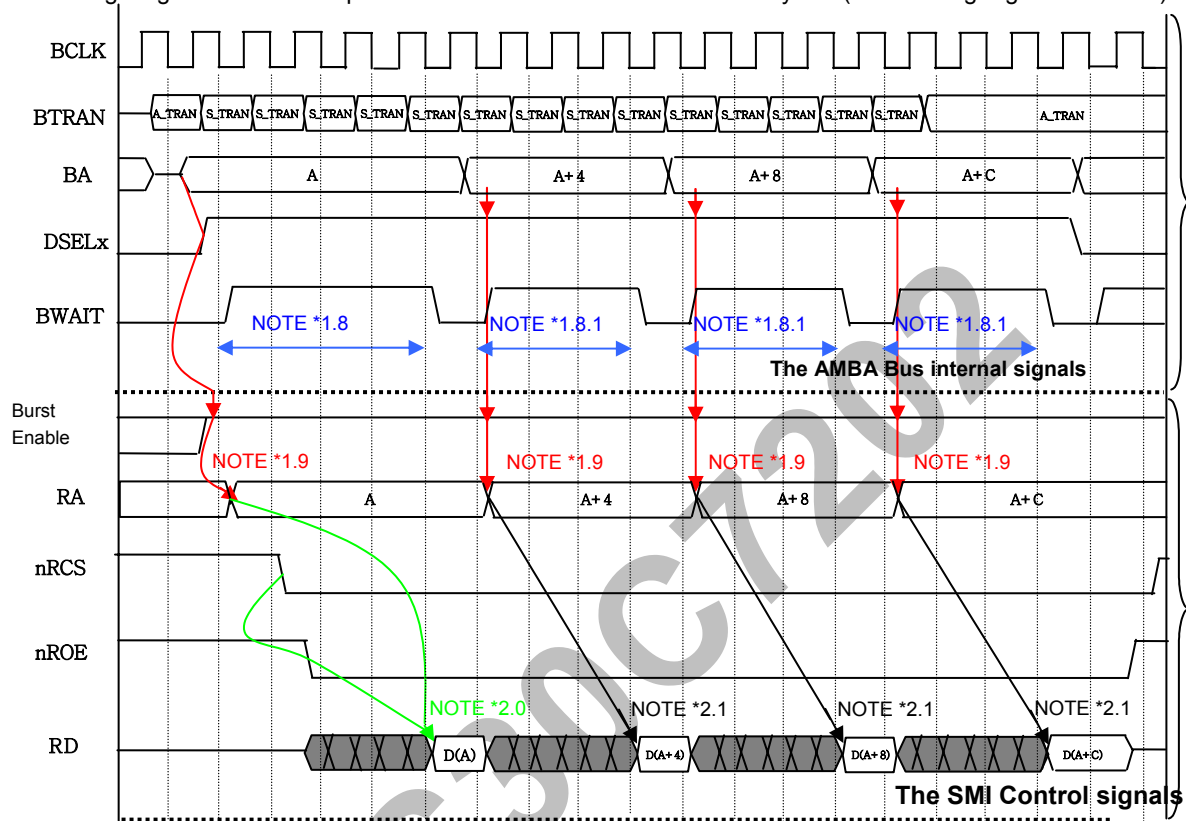
*NOTE 1.5: Valid the SMI address latch on the ASB Bus address when BA and DSEL are valid condition.

*NOTE 1.6: After generated SMI control signals, external device read data is valid with SMI address (RA), nRCS, and nROE.

*NOTE 1.7: The BTRAN is sequential transfer so the SMI control signal (nRCS, nROE) are not asserted any more, and then external device read data is valid with SMI address (RA).

7.4.3 Read burst wait (Sequential mode)

This timing diagram shows a sequential burst read accesses with 3 wait cycles (MEM config register = 0xE60)

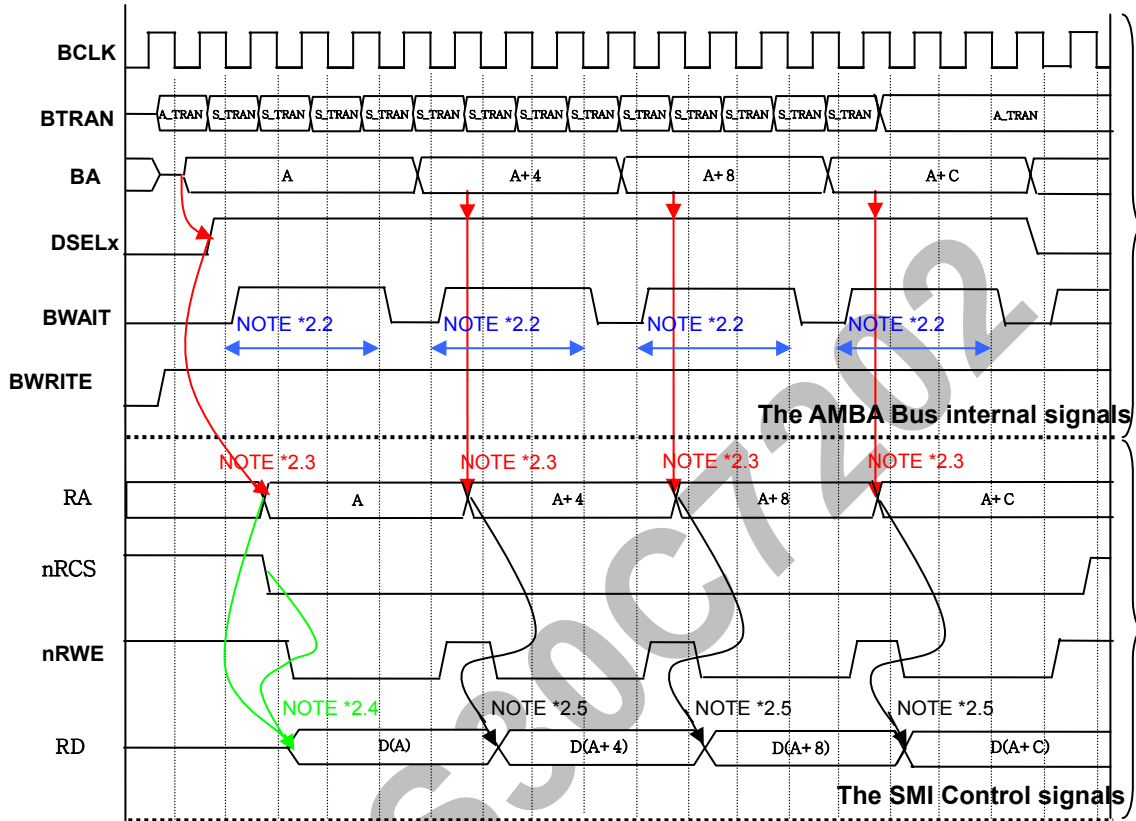


- *NOTE 1.8 : For the 1st read of a Burst read transfer the wait time is Normal Wait time (in this example 4 cycles).
- *NOTE 1.8.1: BWAIT time = BCLK x 3 wait cycle (If MCR is set)
- *NOTE 1.9: Valid the SMI address latch on the ASB Bus address when BA, DSEL, and BurstEnable are valid condition.
- *NOTE 2.0: After generated SMI control signals, external device read data is valid with SMI address (RA), nRCS, and nROE.
- *NOTE 2.1: The BTRAN is sequential transfer so the SMI control signal (nRCS, nROE) are not asserted any more, and then external device read data is valid with SMI address(RA).

Above the figure of burst wait signals; make sure that BurstEnable signal will be change (High to Low) at the BA is become to different value.

7.4.4 Write normal wait (Sequential mode)

This timing diagram shows a sequential write accesses with 3 wait cycles (MEM config register = 0x068).



*NOTE 2.2: BWAIT time = BCLK x 3 wait cycle (If MCR is set)

*NOTE 2.3: Valid the SMI address latch on the ASB Bus address when BA and DSEL are valid condition.

*NOTE 2.4: After generated SMI control signals, external device write data is valid with SMI address (RA), nRCS, and nRWE.

*NOTE 2.5: The BTRAN is Sequential transfer so nRCS external chip enable signal is not asserted, but nRWE external write enable signal asserted on the falling edge of BCLK.

7.5 Internal SRAM

7.5.1 Remapping Enable Register

HMS30C7202 allows the remapping of the internal SRAM block (Base address : 0x7F00.0000 - 2KB size) to enhance the performance.

| | | | | | | | | |
|------------|----|----|----|------------|----------|----|---------|-------------|
| | | | | | | | | 0x8000.1040 |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | |
| Reserved | | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | |
| Reserved | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | |
| Reserved | | | | Remap Size | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| Remap Size | | | | | Reserved | | RemapEn | |

| Bits | Type | Function |
|-------|------|---|
| 31:13 | - | Reserved |
| 12:3 | R/W | Remap Size (word Boundary) Caution : Max size of remapping is 0x7FF(2KB area). If remap size setting exceeds this value, the correct operation can not be guaranteed |
| 2:1 | | Reserved |
| 0 | R/W | 1 : Enable Remap 0 : Disable Remap |

7.5.2 Remap Source Address Register

| | | | | | | | | |
|----------------------|----|----|----|----|----|----------|----|-------------|
| | | | | | | | | 0x8000.1048 |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | |
| Reserved | | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | |
| Remap Source Address | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | |
| Remap Source Address | | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| Remap Source Address | | | | | | Reserved | | |

| Bits | Type | Function |
|-------|------|---|
| 31:25 | - | Reserved |
| 24:2 | R/W | Remap Source Address Start address of Remapping(Word Boundary) |
| 0:1 | | Reserved |

8 LCD CONTROLLER

FEATURES

- Single panel color and monochrome STN displays
- TFT color displays
- Resolution programmable up to 640x480
- Single panel mono STN displays with either 4- or 8-bit interfaces
- 15 gray-level mono support, 3375 color STN support
- 4bpp mono, 4 or 8bpp palletized color displays
- 16bpp color 'true-color' color displays(TFT)
- Programmable timing for different display panels
- 3 x 256 entry, 5-bit Red, Blue and 6-bit Green palette RAM in TFT mode
- 3 x 256 entry, 4-bit palette RAM in STN mode
- Patented grayscale algorithm
- Little-endian operation

Note

The controller does not support dual panel STN displays.
There is no hardware cursor support, since WinCE does not use a cursor.

8.1 Video operation

A block diagram of the video system is shown in Figure 8-1: Video System Block Diagram. The video system has a data path for STN LCD and for TFT LCDs.

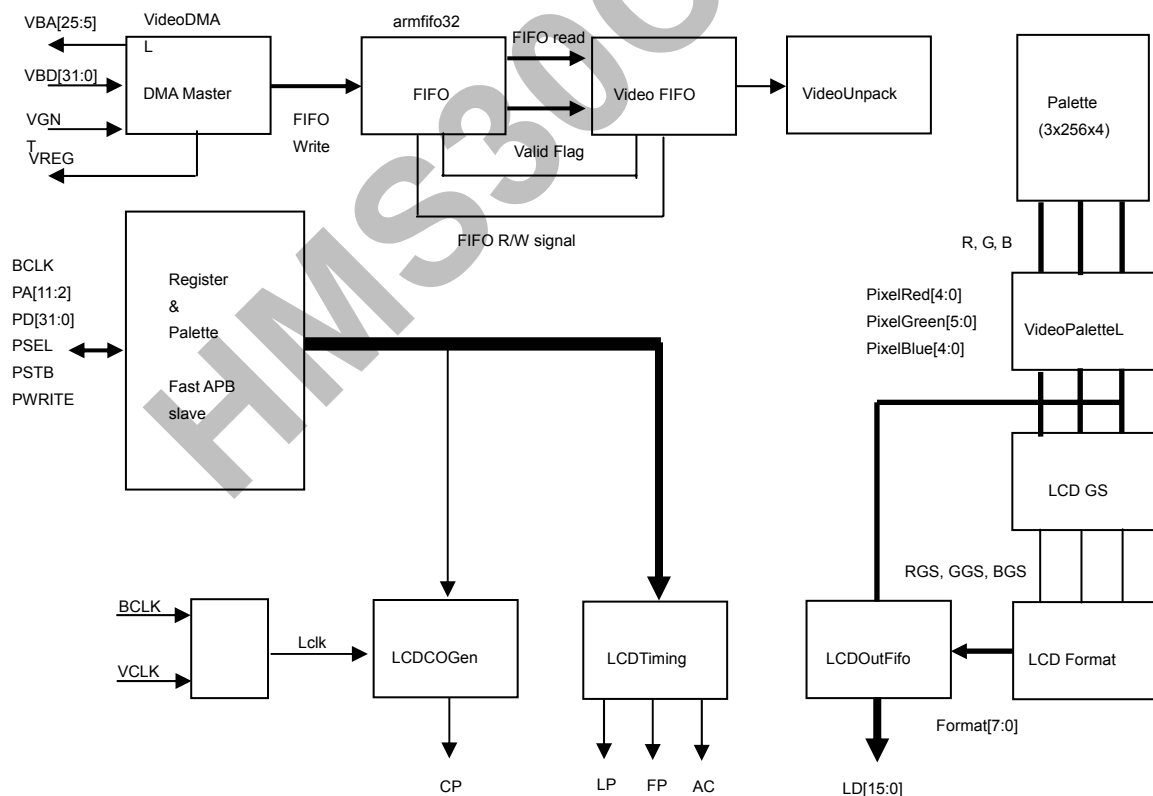


Figure 8-1 Video System Block Diagram

8.1.1 LCD datapath

In TFT mode the digital RGB data is directly available at output pins. However, in STN mode, the data must be gray scaled, and then formatted for the LCD panel. The grayscale block converts the 4 bit per color gun data into a single bit per gun, using a patented time/space dither algorithm. In mono mode, only the B gun data is used. The output of the grayscale is fed to the formatter, which formats the pixels in the correct order for the LCD panel type in use. (4 or 8 mono pixels per clock for mono panels, or 2 2/3pixels per clock for color data.) The output of the formatter in color mode is bursty, due to the 2 2/3pixels per clock that are output, so the formatter output goes to a small FIFO, which smoothes out this burstiness, before data is output to the LCD panel at a constant rate.

8.1.1.1 Palette RAM & 16bpp mode

Logical pixels are either 8 or 16 bits. In 8-bit mode, the logical pixel value is used to index into the three palette arrays to select the three color components of the physical pixel value. In 16-bit pseudo true-color mode, a patented technique is used to allow 2^{16} colors to be selected from 2^{24} possible colors. Separate color gun values are independently used to index into the three palette arrays, to select an 8-bit value for each of the color guns. By splitting the palette RAM into three separate RAM arrays, it allows 16bpp data to generate 8-bit color gun data. The method used is an ARM patented technique, where 16bpp data is split into three overlapping 8-bit fields that are used to index into the three RAM arrays. The red gun is indexed by bits 15:8 of the 16-bit pixel value, the blue gun is indexed by bits 7:0 of the pixel value, and the green gun is indexed by bits 11:4 of the pixel value. By programming the palette with the correct values, 5:5:5, 5:6:5, 4:8:4, and many other combinations of 16-bit data may be used. Thus:

8 bpp : 256 palette entries are used for each palette array. All three palette RAMs are indexed by pixel[7:0]

16 bpp : 256 palette entries are used for each palette array. Red array is indexed by pixel[15:8], green array is indexed by pixel[11:4], and blue array is indexed by pixel[7:0]

Figure 8-2 shows 5:6:5 combination. Least significant 3bits are don't cares for red index, most significant 3bits are don't cares for blue index. Bit0 and bit7 are don't cares for green index.

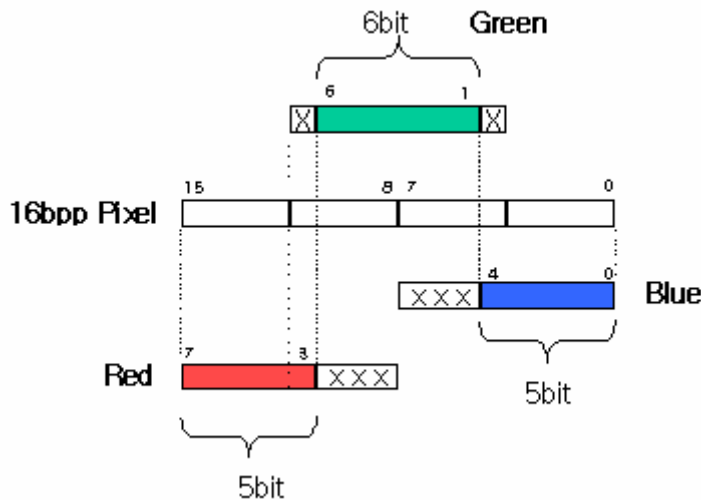


Figure 8-2 5:6:5 Combination of 16bpp Data

The effective 5, 6, and 5 bits are indexes to HMS30C7202 palette RAM for TFT mode. Figure 8-3 shows HMS30C7202 palette register mapping for 16bpp(5:6:5) representation.

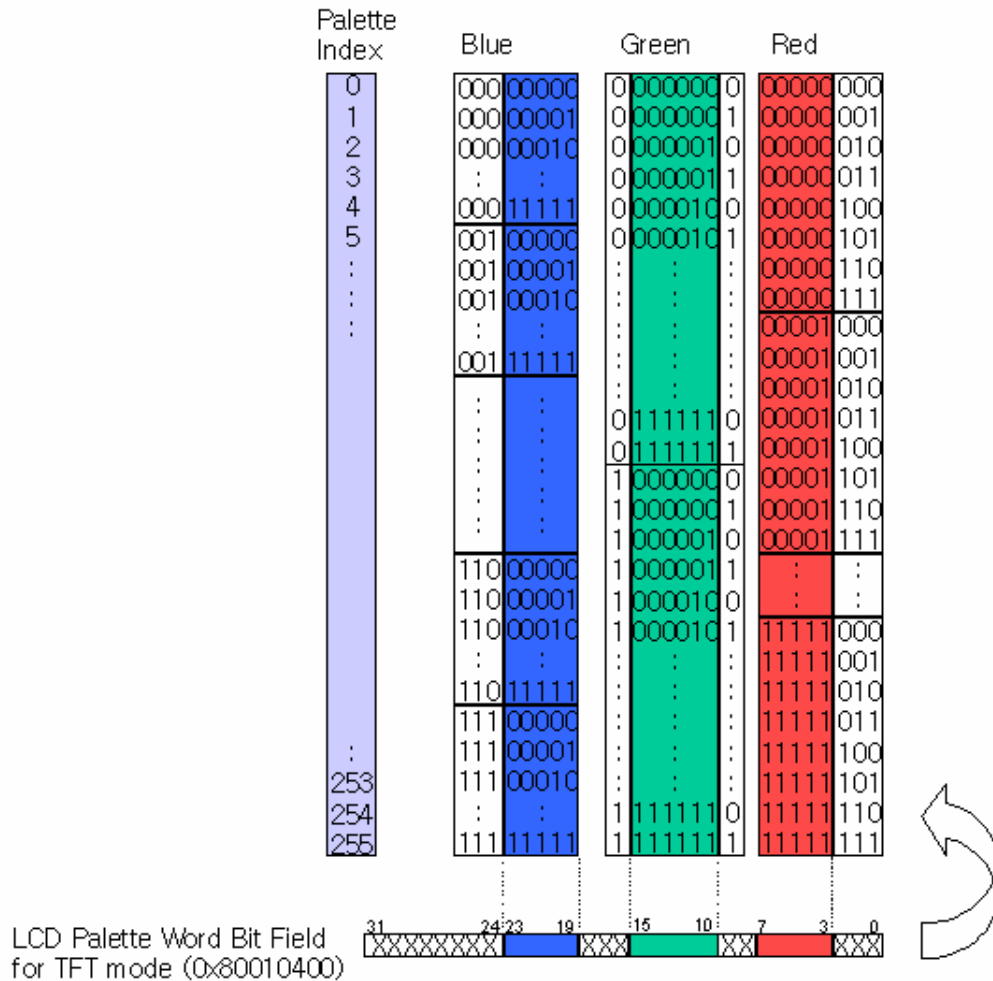


Figure 8-3 Palette RAM Entries for 5:6:5 Combination

To program palette RAM as in Figure 8-3, refer to the code in Figure 8-4.

```

unsigned long palette[256];

main ( )
{
    int i;

    for (i=0; i<256; i++)
    {
        // store 5 bits red, 6 bits green, and 5 bits blue
        palette[i] = ((i&0x1f) << 19) | ((i&0x7e) << 9) | ((i&0xf8));

        printf("%d, %02X, %02X, %02X\r\n", i, (i&0x1f) << 3, (i&0x7e) << 1, (i&0xf8) << 0);
    }
}
    
```

Figure 8-4 Sample Code for 5:6:5 Palette Generation

8.1.2 Color/Grayscale Dithering

Entries selected from the look-up palette are sent to the color/grayscale space/time base dither generator. Each 4-bit value is used to select one of 15 intensity levels.

Note that two of the 16 dither values are identical. The table below assumes that a pixel data input to the LCD panel is active HIGH. That is, a `1' in the pixel data stream will turn the pixel on, and a `0' will turn it off. If this is not the case, the intensity order will be reversed, with "0000" being the most intense color. This polarity is LCD panel dependent.

The gray/color intensity is controlled by turning individual pixels on and off at varying periodic rates. More intense grays/colors are produced by making the average time that the pixel is off longer than the average time that it is on. The proprietary dither algorithm is optimized to provide a range of intensity values that match the eye's visual perception of color/gray gradations, with smaller changes in intensity nearer to the mid-gray level, and greater nearer the black and the white levels. In color mode, red, green and blue components are gray-scaled simultaneously as if they were mono pixels. The duty cycle and resultant intensity level for all 15 color/grayscale levels is summarized in Table 8-1: Color/grayscale intensities and modulation rates.

| Dither Value (4 bit value from palette) | Intensity (0% is white) | Modulation Rate (ration of ON to ON+OFF pixels) |
|--|----------------------------|--|
| 1111 | 100.0 | 1 |
| 1110 | 100.0 | 1 |
| 1101 | 88.9 | 8/9 |
| 1100 | 80.0 | 4/5 |
| 1011 | 73.3 | 11/15 |
| 1010 | 66.6 | 6/9 |
| 1001 | 60.0 | 3/5 |
| 1000 | 55.6 | 5/9 |
| 0111 | 50.0 | 1/2 |
| 0110 | 44.4 | 4/9 |
| 0101 | 40.0 | 2/5 |
| 0100 | 33.3 | 3/9 |
| 0011 | 26.7 | 4/15 |
| 0010 | 20.0 | 1/5 |
| 0001 | 11.1 | 1/9 |
| 0000 | 0.0 | 0 |

Table 8-1 LCD Colorgrayscale intensities and modulation rates

8.1.3 How to order the bit on LD[7:0] output

In STN mode, the low order LD signals are the first pixels on the line, and the high order LD signals are later pixels on the line.

In color mode things are different once again. LD[7] is the red component of the first pixel on the line, and LD[6] is the green component of the pixel, and LD[5] the blue, with LD[4] being the red component of the next pixel. This pattern continues, with LD[0] being the green component of the third pixel, and LD[7] of the next clock being the blue component of the same pixel.

| LCD Pin | Time Sequence | | | | | | |
|---------|---------------|----|----|-----|-----|----|----|
| | → | | | | | | |
| LD[7] | R0 | B2 | G5 | R8 | ... | R0 | B2 |
| LD[6] | G0 | R3 | B5 | G8 | ... | G0 | R3 |
| LD[5] | B0 | G3 | R6 | B8 | ... | B0 | G3 |
| LD[4] | R1 | B3 | G6 | R9 | ... | R1 | B3 |
| LD[3] | G1 | R4 | B6 | G9 | ... | G1 | R4 |
| LD[2] | B1 | G4 | R7 | B9 | ... | B1 | G4 |
| LD[1] | R2 | B4 | G7 | R10 | ... | R2 | B4 |
| LD[0] | G2 | R5 | B7 | G10 | ... | G2 | R5 |

Table 8-2 How to order the bit on LD[7:0] in 8-bit color STN mode

8.1.4 TFT mode

When TFT display mode is enabled, the timing of the pixel, line and frame clocks as well as the AC-bias pin change. The pixel clock transitions continuously in this mode as long as the LCD is enabled. The AC-bias pin functions as an output enable. When it is HIGH, the display latches data from the LCD's pins using the pixel clock. The line clock pin is used as the horizontal synchronization signal (HSYNC), and the frame clock is used as the vertical synchronization signal (VSync). Pixel data is output one pixel per clock, rather than 4, 8 or 22/3pixels per clock, as it is in the passive LCD modes.

8.2 Registers

| Address | Name | Width | Default | Description |
|-----------------------------|--------------|-------|---------|--|
| 0x8001.0000 | LcdControl | | | LCD Control Register |
| 0x8001.0004 | LcdStatus | | | LCD Status Register |
| 0x8001.0008 | LcdStatusM | | | LCD Status Mask Register |
| 0x8001.000C | LcdInterrupt | | | LCD Interrupt Register |
| 0x8001.0010 | LcdDBAR | | | LCD DMA Channel Base Address Register |
| 0x8001.0014 | LcdDCAR | | | LCD DMA Channel Current Address Register |
| 0x8001.0020 | LcdTiming0 | | | LCD Timing 0 Register |
| 0x8001.0024 | LcdTiming1 | | | LCD Timing 1 Register |
| 0x8001.0028 | LcdTiming2 | | | LCD Timing 2 Register |
| 0x8001.0040 | LcdTest | | | LCD Test register |
| 0x8001.0044 | GsfState | | | Grayscale production test register |
| 0x8001.0048 | GsrState | | | Grayscale production test register |
| 0x8001.004C | GscState | | | Grayscale production test register |
| 0x8001.0400~ 0x8001.07FC | LcdPalette | | | LCD Palette programming registers |

Table 8-3 LCD Controller Register Summary

8.2.1 LCD Power Control

LCD displays require that the LCD is running before power is applied. For this reason, the LCD's power on control is not set to "1" unless both LcdEn and LcdPwr are set to "1". Note that most LCD displays require the LcdEn must be set to "1" approximately 20ms before LcdPwr is set to "1" for powering up. Likewise, LcdPwr is set to "0" 20ms before LcdEn is set to "0" for powering down.

| 0x80010000 | | | | | | |
|------------|--------|----------|-----|----------|----|---------|
| | | | | | | 24 |
| | | | | | | LDbusEn |
| 23 | 22 | 21 | | 19 | 18 | |
| LcdBLE | LcdPwr | LcdMono8 | | LcdVComp | | |
| | | | 12 | | | |
| | | | BGR | | | |

| | | | 4 | 3 | 2 | 1 | 0 | |
|-------|------|--|--------|-------|--------|-------|---|--|
| | | | LcdTFT | LcdBW | LcdBpp | LcdEn | | |
| Bits | Type | Function | | | | | | |
| 31:25 | - | Reserved | | | | | | |
| 24 | R/W | LD data bus Enable 0 – LD data bus disable (initial value) 1 – LD data bus Enable | | | | | | |
| 23 | R/W | Lcd Backlight enable This drives "0" or "1" out to the Lcd backlight enable pin | | | | | | |
| 22 | R/W | Lcd power enable 0 - Lcd is off 1 - Lcd is on when LcdEn=1 | | | | | | |
| 21 | R/W | Lcd monochrome data width 0 - 4 bits Lcd module 1 - 8 bits Lcd module | | | | | | |
| 20 | - | Reserved | | | | | | |
| 19:18 | R/W | Generate interrupt at: 00 - start of VSync 01 - start of BACK PORCH 10 - start of ACTIVE VIDEO 11 - start of FRONT PORCH | | | | | | |
| 17:13 | - | Reserved | | | | | | |
| 12 | R/W | 0 - RGB normal video output for LCD 1 - BGR red and blue swapped for LCD | | | | | | |
| 11:5 | - | Reserved | | | | | | |
| 4 | R/W | LCD TFT 0 - Passive or STN display operation enabled 1 - Active or TFT display operation enabled | | | | | | |
| 3 | R/W | LCD Monochrome 0 - Color operation enabled 1 - Monochrome operation only enabled | | | | | | |
| 2:1 | R/W | LCD Bits Per Pixel 00 - 4bpp 01 - 8bpp 10 - 16bpp 11 – Reserved | | | | | | |
| 0 | R/W | LCD Controller Enable 0 - LCD controller disabled 1 - LCD controller enabled | | | | | | |

8.2.2 LCD Controller Status/Mask and Interrupt Registers

The LCD controller status, mask and interrupt registers all have the same format. Each bit of the status register is a status bit that may generate an interrupt. The corresponding bits in the mask register mask the interrupt. The interrupt register is the logical AND of the status and mask registers, and the interrupt output from the LCD controller is the logical OR of the bits within the interrupt register.

The LCD controller status register contains bits that signal an under-run error for the FIFO, the DMA next base update ready status, and the DMA done status. Each of these hardware-detected events can generate an interrupt request to the interrupt controller.

| | | | 0x80010004 ~ 0x800100c | | | |
|------|------|---|------------------------|-------|-------|------|
| | | | 3 | 2 | 1 | 0 |
| | | | LDone | VComp | LNext | LFUF |
| Bits | Type | Function | | | | |
| 31:4 | - | Reserved | | | | |
| 3 | R | LCD Done frame status/mask/interrupt bit The LCD Frame Done (Done) is a read-only status bit that is set after the LCD has been | | | | |

| | | |
|---|-----|---|
| | | disabled (LcdEn = 0) and the frame that is current active finishes being output to the LCD's data pins. It is cleared by writing the base address (LcdDBAR) or enabling the LCD, or, by writing "1" to the LDone bit of the Status Register. When the LCD is disabled by clearing the LCD enable bit (LcdEn=0) in LcdControl, the LCD allows the current frame to complete before it is disabled. After the last set of pixels is clocked out onto the LCD's data pins by the pixel clock, the LCD is disabled and Done is set. |
| 2 | R/W | Vertical compare interrupt This bit is set when the Lcd timing generator reaches the vertical region programmed in the Video Control Register. This bit is "sticky", meaning it remains set until it is cleared by writing a "1" to this bit |
| 1 | R | LCD Next base address update status/mask/interrupt bit The LCD Next Frame (LNext) is a read-only status bit that is set after the contents of the LCD DMA base address register are transferred to the LCD DMA current address register at the start of frame, and it is cleared when the LCD DMA base address register is written. |
| 0 | R/W | FIFO underflow status/mask/interrupt bit The LCD FIFO underflow (LFUF) status bit is set when the LCD FIFO under-runs. The status bit is "sticky", meaning it remains set after the FIFO is no longer underrunning. The status bit is cleared by writing a `1' to this bit. |

8.2.3 LCD DMA Base Address Register

The LCD DMA base address register (LcdDBAR) is a read/write register used to specify the base address of the off-chip frame buffer for the LCD. Addresses programmed in the base address register must be aligned on sixteen-word boundaries, thus the least significant six bits (LcdDBAR [5:0]) must always be written with zeros. Only 26 bits of the register are valid (including the LS 6 bits which must be zero), because LCD DMA is only allowed from SDRAM.

The 26 bits address range allows the LCD DMA to access any address within the SDRAM. The upper address lines are not needed, because these are the address lines used to select which device is accessed, but the LCD always accesses SDRAM. The user must initialize the base address register before enabling the LCD, and may also write a new value to it while the LCD is enabled to allow a new frame buffer to be used for the next frame. The user can change the state of LcdDBAR while the LCD controller is active, after the Next Frame (Next) status bit is set within the LCD's status register that generates an interrupt request. This status bit indicates that the value in the base address pointer has been transferred to the current address pointer register and that it is safe to write a new base address value. This allows double-buffered video to be implemented if required.

0x80010010

| Bits | Type | Function |
|------|------|--|
| 31:6 | - | Reserved. Keep these bits zero |
| 25:6 | R/W | LcdDBAR : LCD DMA Channel Base Address Pointer 16-word aligned base address in SDRAM of the frame buffer within off-chip memory. |
| 5:0 | - | Reserved. Keep these bits zero |

8.2.4 LCD DMA Channel Current Address Register

This read-only register allows the processor to read the current value of the LCD DMA channel current address register. This is not something that would normally be done, but it allows additional test observability. Its value cannot be expected to be exact, it could change at an moment. However, its contents can be read to determine the approximate line that the LCD controller is currently displaying and driving out to the display

0x80010014

| Bits | Type | Function |
|------|------|---|
| 31:6 | - | Reserved. Keep these bits zero |
| 25:6 | R/W | LcdDCAR : LCD DMA Channel Current Address Pointer 16-word aligned current address pointer to data in SDRAM frame buffer currently being displayed |
| 5:0 | - | Reserved. Keep these bits zero |

8.2.5 LCD Timing 0 Register

LCD Timing 0 Register (LcdTiming0) controls horizontal LCD timing. See 8.6.2 Pixel Clock Divider (PCD) on

page 8-13 for a description of the terms "PixelClock" and "LcdClk"

0x80010020

| | | | | | | | | | | | | | | | |
|-----|----|----|----|----|----|----|----|-----|----|----|----|----|----|----|----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| HBP | | | | | | | | HFP | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | | |
| HSW | | | | | | | | PPL | | | | | | | |

| Bits | Type | Function |
|-------|------|---|
| 31:24 | R/W | Horizontal Back Porch The 8-bit Horizontal Back Porch (HBP) field is used to specify the number of pixel clock periods to insert at the beginning of each line or row of pixels. After the line clock for the previous line has been negated, the value in HBP is used to count the number of pixel clocks to wait before starting to output the first set of pixels in the next line. HBP generates a wait period ranging from 1-256 pixel clock cycles (Number of LcdClk clock periods to add to the beginning of a line transmission before the first set of pixels is output to the display minus 1). |
| 23:16 | R/W | HFP Horizontal Front Porch The 8-bit Horizontal Front Porch (HFP) field is used to specify the number of pixel clock periods to insert at the end of each line or row of pixels before pulsing the line clock pin. Once a complete line of pixels is transmitted to the LCD driver, the value in HFP is used to count the number of pixel clocks to wait before pulsing the line clock. HFP generates a wait period ranging from 1-256 pixel clock cycles. (Program to value required minus one). |
| 15:8 | R/W | Horizontal Sync Pulse Width The 6-bit horizontal sync pulse width (HSW) field is used to specify the pulse width of the line clock in passive mode, or horizontal synchronization pulse in active mode. Number of LcdClk clock periods to pulse the line clock at the end of each line minus 1 |
| 7:2 | R/W | The pixels-per-line (PPL) bit-field is used to specify the number of pixels in each line or row on the screen. PPL is a 6-bit value that represents between 16-1024 pixels per line. PPL is used to count the correct number of pixel clocks that must occur before the line clock can be pulsed. Program the value required divided by 16, minus 1. |
| 1:0 | - | Reserved |

8.2.6 LCD Timing 1 Register

LCD Timing 1 Register (LcdTiming1) controls LCD vertical timing parameters.

0x80010024

| | | | | | | | | | | | | | | | |
|-----|----|----|----|----|----|----|----|-----|----|----|----|----|----|----|----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| VBP | | | | | | | | VFP | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| VSW | | | | | | | | LPS | | | | | | | |

| Bits | Type | Function |
|-------|------|---|
| 31:24 | R/W | Vertical Back Porch The 8-bit Vertical Back Porch (VBP) field is used to specify the number of line clocks to insert at the beginning of each frame, i.e. number of inactive lines at the start of a frame, after VSync period. The VBP count starts just after the VSync signal for the previous frame has been negated for active mode, or the extra line clocks have been inserted as specified by the VSW bit-field in passive mode. After this has occurred, the value in VBP is used to count the number of line clock periods to insert before starting to output pixels in the next frame. VBP generates from 0-255 extra line clock cycles. This should be programmed to zero in passive mode, unless sensing LCD to VGA to share DMA data |
| 23:16 | R/W | Vertical Front Porch The 8-bit Vertical Front Porch (VFP) field is used to specify the number of line clocks to insert at the end of each frame, i.e. number of inactive lines at the end of frame, before VSync period. Once a complete frame of pixels is transmitted to the LCD display, the value in VFP is used to count the number of line clock periods to wait. After the count has elapsed the VSync (LcdFP) signal is pulsed in active mode, or extra line clocks are inserted as specified by the VSW bit-field in passive mode. VFP generates from 0-255 line clock cycles. This should be zero for passive display modes, unless synchronizing to the VGA to share data. |

| | | |
|-------|-----|--|
| 15:10 | R/W | Vertical Sync Pulse Width The 6-bit vertical sync pulse width (VSW) field is used to specify the pulse width of the vertical synchronization pulse in active mode, or is used to add extra dummy line clock delays between frames in passive mode. Should be small for passive LCD, but should be long enough to re-program the video palette under interrupt control, without writing the video palette at the same time as video is being displayed. The register is programmed with the number of lines of VSync minus one. |
| 9:0 | R/W | Lines Per Screen The Lines Per Screen (LPS) bit-field is used to specify the number of lines or rows per LCD panel being controlled. LPS is a 10-bit value that represents 1-1024 Lines Per Screen. The register is programmed with the number of lines per screen minus 1. |

8.2.7 LCD Timing 2 Register

LCD Timing 2 Register (LcdTiming2) controls various functions associated with the timing of the LCD controller. 0x80010028

| | | | | | | | | | | | | | | | |
|-----|-----|-----|-----|-------|-----|-----|----|----|----|----|-----|-----|----|----|----|
| | | | | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | | | | Skip4 | BCD | CPL | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SLV | IEO | IPC | IHS | IVS | ACB | | | | | | LCS | PCD | | | |

| Bits | Type | Function |
|-------|------|---|
| 31:28 | - | Reserved |
| 27 | R/W | Set this bit to "1" when running a color passive LCD with slave mode. This produces an irregular clock to the LCD, where every fourth clock pulse is suppressed (the clock stays LOW for one clock period). This is necessary because two-and-two-third pixels per clock, which are sent to the LCD, is not an integer multiple. This means that three clocks will be output every four-clock period. If PCD is zero, then eight pixels will be output every eight LcdClk periods, since the LCD CP clock will be half the frequency of LcdClk. |
| 26 | R/W | Bypass Pixel Clock Divider Setting this bit allows an undivided LCD clock to be output on LCD. This bit could only be set for TFT mode but not in normal cases. |
| 25:16 | R/W | Clocks Per Line This is the actual number of clocks output to the LCD panel each line, minus one. This must be programmed, in addition to the PPL field in the LCD Timing 0 Register. The number of clocks per line is the number of pixels per line divided by 1, 4, 8 or two-and-two-thirds for TFT mode, mono 4-bit mode, mono 8-bit, or color STN mode (22/3) respectively. |
| 15 | R/W | Slave mode Slave (or genlock) LCD to VGA video. The HSync and VSync are locked to the VGA timing generator. The LCD horizontal timing must be carefully programmed if sharing DMA data |
| 14 | R/W | Invert Output Enable The Invert Output Enable (IEO) bit is used to select the active and inactive state of the output enable signal in active display mode. In this mode, the AC-bias pin is used as an enable that signals the off-chip device when data is actively being driven out using the pixel clock. When IEO=0, the LcdAC pin is active HIGH. When IEO=1, the LcdAC pin is active LOW. In active display mode, data is driven onto the LCD's data lines on the programmed edge of LcdCP when LcdAC is in its active state. 0 - LcdAC pin is active HIGH in TFT mode 1 - LcdAC pin is active LOW in TFT mode |
| 13 | R/W | Invert Pixel Clock The Invert Pixel Clock (IPC) bit is used to select which edge of the pixel clock pixel data is driven out onto the LCD's data lines. When IPC=0, data is driven onto the LCD's data lines on the rising-edge of LcdCP. When IPC=1, data is driven onto the LCD's data lines on the falling-edge of LcdCP. 0 - Data is driven on the LCD's data lines on the rising-edge of LcdCP. 1 - Data is driven on the LCD's data lines on the falling-edge of LcdCP. |
| 12 | R/W | Invert Hsync The Invert HSync (IHS) bit is used to invert the polarity of the LcdLP signal. 0 - LcdLP pin is active HIGH and inactive LOW. |

| | | |
|------|-----|---|
| | | 1 - LcdLP pin is active LOW and inactive HIGH. |
| 11 | R/W | Invert Vsync The Invert VSync (IVS) bit is used to invert the polarity of the LcdFP signal. 0 - LcdFP pin is active HIGH and inactive LOW. 1 - LcdFP pin is active LOW and inactive HIGH. |
| 10:6 | R/W | AC Bias Pin Frequency The 5-bit AC-bias frequency (ACB) field is used to specify the number of line clock periods to count between each toggle of the AC-bias pin (LcdAC). This pin is used to periodically invert the polarity of the power supply to prevent DC charge build-up within the display. The value programmed is the number of lines between transitions, minus 1. Note The ACB bit field had no effect on LcdAC in active mode. The pixel clock transitions continuously in active mode and the AC Bias line is used as an output enable signal |
| 5 | R/W | LCD Clock source selection 0 - DMA bus clock (system bus clock) 1 - Video PLL clock (VCLK; in normal operation) |
| 4:0 | R/W | Pixel Clock Divisor Used to specify the frequency of the pixel clock based on the LCD clock (LcdCLK) frequency. Pixel clock frequency can range from LcdCLK/2 to LcdCLK/33, where LcdCLK is the clock selected by LCS. Pixel Clock Frequency = LcdCLK/(PCD+2). Note that in the case of the LCD, the pixel clock is not the frequency of some nominal clock rate that individual pixels are output to the LCD. It is the frequency of the LcdCP signal. In normal mono mode (4-bit interface), four pixels are output per LcdCP cycle, so the PixelClock is one quarter the nominal pixel rate. In the case of 8-bit interface mono, PixelClock is one-eighth the nominal pixel rate, since 8 pixels are output per LcdCP cycle. In the case of color, PixelClock is 0.375 times the nominal pixel rate, because 22/3 pixels are output per LcdCP cycle. If the LCD and VGA are operating concurrently, and sharing DMA data, then in color mode the pixel clock should normally be 3/8 the VGA clock. To achieve this, PCD should be programmed to the value 0 and the skip4 bit set to "1". The skip4 bit produces a null clock cycle (no high phase) every fourth clock cycle. |

8.2.8 LCD Test Register

The LCD test register contains bits that allow certain LCD signals to be output on the LCD pins for test purposes. This register should not normally be used. The register is reset to all zero, and this will result in normal operation.

| | | | | | | | | |
|-----|-----|-----|-----|-----|-----|--------|-----------|------------|
| | | | | | | | 8 | 0x80010040 |
| | | | | | | | TCOUNT | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| TCC | TLC | TCR | TLR | TCF | TRF | TLDATA | TEST MODE | |

| Bits | Type | Function |
|------|------|---|
| 31:9 | - | Reserved |
| 8 | R/W | Separates the 10-bit counter into nibbles for the test purpose |
| 7 | R/W | For production test of grayscale, never write a "1" to these registers in normal use. |
| 6 | R/W | For production test of grayscale, never write a "1" to these registers in normal use. |
| 5 | R/W | For production test of grayscale, never write a "1" to these registers in normal use. |
| 4 | R/W | For production test of grayscale, never write a "1" to these registers in normal use. |
| 3 | R/W | For production test of grayscale, never write a "1" to these registers in normal use. |
| 2 | R/W | For production test of grayscale, never write a "1" to these registers in normal use. |
| 1 | R/W | Walking one's pattern used in place of SDRAM data for the LCD controller |
| 0 | R/W | Test mode bit for grey-scaler |

8.2.9 Grayscale Test Registers

The registers GSFrame State, GSRow State and GS Column State are used for the purpose of production test

and **must not** be written to or read from in normal use.

0x80010044, 0x80010048, 0x8001004c

8.2.10 LCD Palette registers

The LCD palette registers are a set of 256 word-aligned registers that allow the LCD to be programmed. The format of the palette data is shown below. At the TFT mode, the palette RAM bit width will be increased as Figure 8-6.

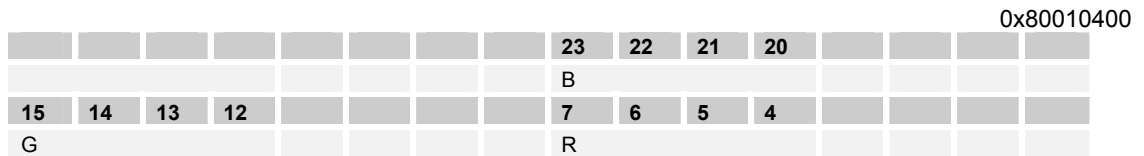


Figure 8-5 LCD Palette Word Bit Field for STN mode

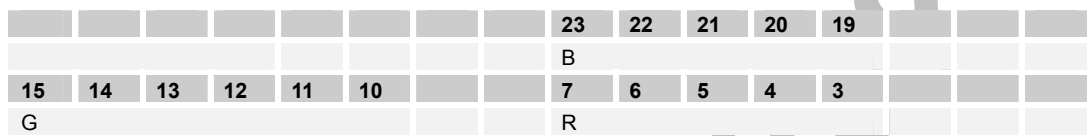


Figure 8-6 LCD Palette Word Bit Field for TFT mode

HMS30C7202

8.3 Timings

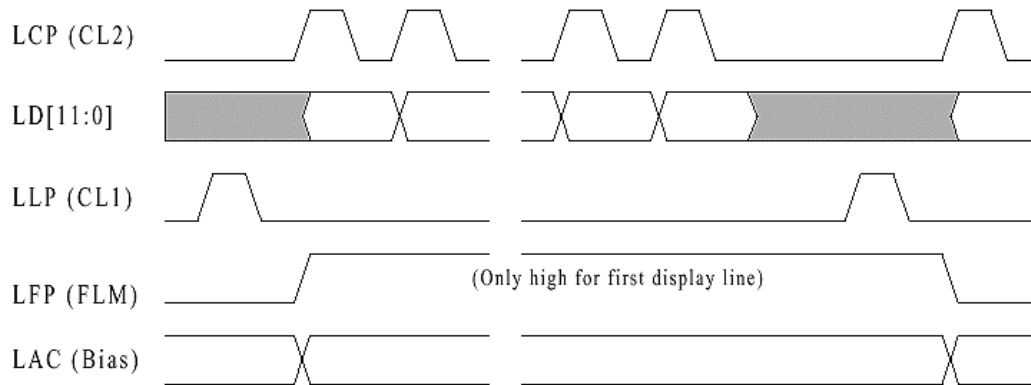


Figure 8-7 Example Mono STN LCD Panel Signal Waveforms

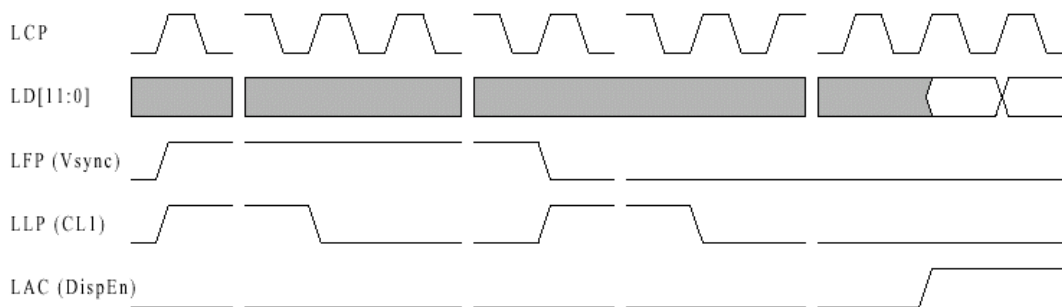


Figure 8-8 Example TFT Signal Waveforms, Start of Frame

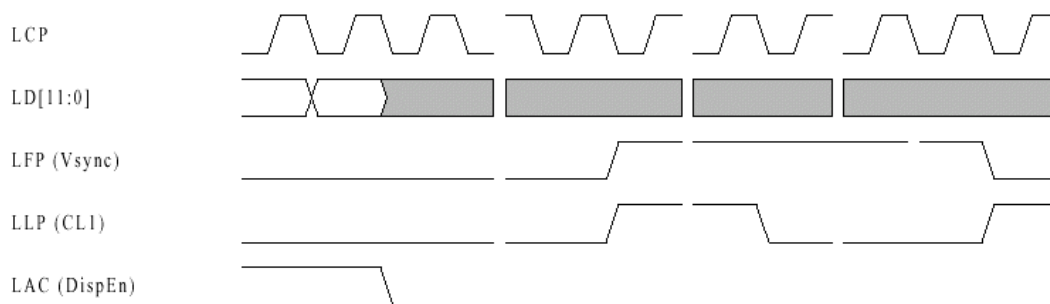


Figure 8-9 Example TFT Signal Waveforms, End of Last Line

9 FAST AMBA PERIPHERALS

9.1 DMA Controller

This chip includes a three-channel direct memory access controller (DMAC). High-speed transfers between peripheral devices and the SDRAM can be controlled by the DMAC instead of the CPU core. Transfers using addresses other than SDRAM will produce unpredictable results.

Features

- Three Channels.
- Max Transfer rate: 133MB/s.
- Max Buffer size: 16383.
- Address mode: Single(SDRAM) address is supported.
- Channel function: Transfer modes are different in each channel.
 - i. **Channel 0:** Dedicated to the sound interface controller. This channel has a source address reload function. The memory space of the sound I/O device consists of a double buffer. The sound interface uses exception bus mode and word access. The channel performs only DMA transfers for transmitting data (transfers from SDRAM to the sound interface).
 - ii. **Channel 1:** Dedicated to the SMC/MMC interface block. The channel uses exception bus mode and word access. It controls DMA transfers for both transmitting (from SDRAM) and receiving (to SDRAM). Word is the only supported transfer size. Correct DMA operation of this channel is guaranteed only if the SDRAM write buffer is enabled and LCD operation is disabled. Otherwise it will produce unpredictable results.
 - iii. **Channel 2:** Used by external IO device. The channel supports both exception and burst bus modes. Transfer sizes of byte, half word (16 bits) and word are all supported.
- Channel priority: Configured by register setting.
- Interrupt request: The DMAC interrupt request can be triggered by each channel whenever the DMA transfer is completed by buffer size. Since only one interrupt ID is assigned to the DMAC, the interrupt flag register (FLAGR) maintains the information on which DMA channel requested the interrupt.
- The channel 2 should not be enabled with either of the other channels at the same time.

9.1.1 External Signals

| Pin Name | Type | Description |
|----------|------|---|
| nDMAREQ | I | DMA request input signal from external device (level sensitive, active Low) |
| nDMAACK | O | DMA acknowledge output signal to external Device. |

9.1.2 Registers

| Address | Name | Width | Default | Description |
|-------------|------|-------|---------|--|
| 0x8000.4000 | ADR0 | 32 | 0x0 | Write: Start address of the first buffer of Channel 0 Read: Current address of the first buffer of Channel 0 |
| 0x8000.4004 | ASR | 32 | 0x0 | Write: Start address of the second buffer of Channel 0 Read: Current address of the second buffer of Channel 0 |
| 0x8000.4008 | TNR0 | 14 | 0x3FFF | Write: Size of the first buffer of Channel 0 (in words) Read: Number of words in the first buffer of Channel 0 which remain to be transferred |
| 0x8000.400C | TSR | 14 | 0x3FFF | Write: Size of the second buffer of Channel 0 (in words) Read: Number of words in the second buffer of Channel 0 which remain to be transferred |
| 0x8000.4010 | CCR0 | 4 | 0x0 | Channel 0 control |
| 0x8000.4014 | ADR1 | 32 | 0x0 | Write: Start address of Channel 1 buffer Read: Current address of Channel 1 buffer |
| 0x8000.4018 | TNR1 | 14 | 0x3FFF | Write: Size of Channel 1 buffer (in words) Read: Number of words in Channel 1 buffer which remain to be transferred |

| | | | | |
|-----------------------------|-------|----|--------|---|
| 0x8000.401C | CCR1 | 3 | 0x0 | Channel 1 control |
| 0x8000.4020 | ADR2 | 32 | 0x0 | Write: Start address of Channel 2 buffer Read: Current address of Channel 2 buffer |
| 0x8000.4024 | TNR2 | 14 | 0x3FFF | Write: Size of Channel 2 buffer (in unit of transfer size). Read: Number of data in Channel 2 buffer which remain to be transferred (in unit of transfer size) |
| 0x8000.4028 | CCR2 | 8 | 0x0 | Channel 2 control |
| 0x8000.4038~ 0x8000.4040 | - | - | - | Reserved |
| 0x8000.4044 | FLAGR | 5 | 0x0 | DMA interrupt flags |
| 0x8000.4048~ 0x8000.4050 | - | - | - | Reserved |
| 0x8000.4054 | DMAOR | 3 | 0x0 | Operation control of the DMAC |

Table 9-1 DMA Controller Register Summary

9.1.2.1 ADR0

| | | | | 0x8000.4000 | | |
|------|------|--|-----|-------------|---|---|
| 31 | 30 | 29 | ... | 2 | 1 | 0 |
| ADR0 | | | | | | |
| Bits | Type | Function | | | | |
| 31:0 | R/W | Write: Start address of the first buffer (Buffer 0) of Channel 0 (for the sound interface) Read: Current address of the first buffer of Channel 0 | | | | |

9.1.2.2 ASR

| | | | | 0x8000.4004 | | |
|------|------|--|-----|-------------|---|---|
| 31 | 30 | 29 | ... | 2 | 1 | 0 |
| ASR | | | | | | |
| Bits | Type | Function | | | | |
| 31:0 | R/W | Write: Start address of the second buffer (Buffer 1) of Channel 0 Read: Current address of the second buffer of Channel 0 | | | | |

9.1.2.3 TNR0

| | | | | 0x8000.4008 | | |
|----------|------|--|------|-------------|-----|-----|
| | | | 13 | 12 | ... | 1 0 |
| Reserved | | | TNR0 | | | |
| Bits | Type | Function | | | | |
| 13:0 | R/W | Write: Size of the first buffer of Channel 0 (in words, max. 16383) Read: Number of words in the first buffer of Channel 0 which remain to be transferred | | | | |

9.1.2.4 TSR

| | | | | 0x8000.400C | | |
|----------|------|--|-----|-------------|-----|-----|
| | | | 13 | 12 | ... | 1 0 |
| Reserved | | | TSR | | | |
| Bits | Type | Function | | | | |
| 13:0 | R/W | Write: Size of the second buffer of Channel 0 (in words, max. 16383) Read: Number of words in the second buffer of Channel 0 which remain to be transferred | | | | |

9.1.2.5 CCR0

| | | | | 0x8000.4010 | | |
|----------|------|----------|--------|-------------|--|--|
| | | 2 | 1 | 0 | | |
| Reserved | | MASK01 | MASK00 | DMEN0 | | |
| Bits | Type | Function | | | | |

| | | |
|---|-----|--|
| 2 | R/W | Buffer 1 transfer end interrupt mask bit of Channel 0 1 = Interrupt request is generated when the whole DMA transfer of Buffer 1 is completed. 0 = No Interrupt request is generated when the whole DMA transfer of Buffer 1 is completed. |
| 1 | R/W | Buffer 0 transfer end interrupt mask bit of Channel 0 1 = Interrupt request is generated when the whole DMA transfer of Buffer 0 is completed. 0 = No Interrupt request is generated when the whole DMA transfer of Buffer 0 is completed. |
| 0 | R/W | Channel 0 enable bit 1 = Channel 0 is enabled. 0 = Channel 0 is disabled. |

9.1.2.6 ADR1

| | | | | 0x8000.4014 | | |
|------|------|---|-----|-------------|---|---|
| 31 | 30 | 29 | ... | 2 | 1 | 0 |
| ADR1 | | | | | | |
| Bits | Type | Function | | | | |
| 31:0 | R/W | Write: Start address of Channel 1 buffer (for SMC/MMC) Read: Current address of Channel 1 buffer | | | | |

9.1.2.7 TNR1

| | | | | 0x8000.4018 | | |
|----------|------|--|-----|-------------|---|--|
| - | 13 | 12 | ... | 1 | 0 | |
| Reserved | | | | TNR1 | | |
| Bits | Type | Function | | | | |
| 13:0 | R/W | Write: Size of Channel 1 buffer (in words, max. 16383) Read: Number of words in Channel 1 buffer which remain to be transferred | | | | |

9.1.2.8 CCR1

| | | | | 0x8000.401C | | |
|----------|------|---|---|-------------|-------|-------|
| - | 2 | 1 | 0 | | | |
| Reserved | | | | MASK1 | MODE1 | DMEN1 |
| Bits | Type | Function | | | | |
| 2 | R/W | Transfer end interrupt mask bit of Channel 1 1 = Interrupt request is generated when the DMA transfer of the whole buffer is completed. 0 = No Interrupt request is generated when the DMA transfer of the whole buffer is completed. | | | | |
| 1 | R/W | Transfer direction 0 = Transfer from SDRAM to SMC/MMC 1 = Transfer from SMC/MMC to SDRAM | | | | |
| 0 | R/W | Channel 1 enable bit 1 = Channel 1 is enabled. 0 = Channel 1 is disabled. | | | | |

9.1.2.9 ADR2

| | | | | 0x8000.4020 | | |
|------|------|---|-----|-------------|---|---|
| 31 | 30 | 29 | ... | 2 | 1 | 0 |
| ADR2 | | | | | | |
| Bits | Type | Function | | | | |
| 31:0 | R/W | Write: Start address of Channel 2 buffer (for external I/O device) Read: Current address of Channel 2 buffer | | | | |

9.1.2.10 TNR2

| | | | | | | |
|----------|----|----|-----|-------------|---|--|
| | | | | 0x8000.4024 | | |
| - | 13 | 12 | ... | 1 | 0 | |
| Reserved | | | | TNR2 | | |

| Bits | Type | Function |
|------|------|--|
| 13:0 | R/W | Write: Size of Channel 2 buffer (in unit of transfer size, max. 16383) Read: Number of data in Channel 2 buffer which remain to be transferred (in unit of transfer size) |

9.1.2.11 CCR2

0x8000.4028

| | | | | | | | | | |
|----------|-----|-------|---|------|------|---|-------|-------|-------|
| - | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | ISA | BURST | | TYPE | SIZE | | MASK2 | MODE2 | DMEN2 |

| Bits | Type | Function |
|------|------|---|
| 8 | R/W | External bus type 0: not ISA type (Default) 1: ISA type |
| 7:6 | R/W | Burst length 11: 32 beats 10: 16 beats 01: 8 beats 00: 4 beats |
| 5 | R/W | Transfer type 0: exception mode 1: burst mode |
| 4:3 | R/W | Transfer size 11: reserved 10: word 01: half word 00: byte |
| 2 | R/W | Transfer end interrupt mask bit of Channel 2 1 = Interrupt request is generated when the DMA transfer of the whole buffer is completed. 0 = No Interrupt request is generated when the DMA transfer of the whole buffer is completed. |
| 1 | R/W | Transfer direction 0 = Transfer from SDRAM to external I/O 1 = Transfer from external I/O to SDRAM |
| 0 | R/W | Channel 2 enable bit 1 = Channel 2 is enabled. 0 = Channel 2 is disabled. |

* Note: The burst mode must be used in the external bus type of ISA

9.1.2.12 FLAGR

0x8000.4044

| | | | | |
|----------|-------|-------|--------|--------|
| - | 3 | 2 | 1 | 0 |
| Reserved | FLAG2 | FLAG1 | FLAG01 | FLAG00 |

| Bits | Type | Function |
|------|------|---|
| 3 | R/W | Interrupt flag of Channel 2 Set when the whole transfer of Channel 2 buffer is completed. If MASK2 (Bit 2 of CCR2) is set, there is an interrupt request. |
| 2 | R/W | Interrupt flag of Channel 1 Set when the whole transfer of Channel 1 buffer is completed. If MASK1 (Bit 2 of CCR1) is set, there is an interrupt request. |
| 1 | R/W | Interrupt flag of the first buffer of Channel 0 Set when the whole transfer of the first buffer of Channel 0 is completed. If MASK01 (Bit 2 of CCR0) is set, there is an interrupt request. |
| 0 | R/W | Interrupt flag of the second buffer of Channel 0 Set when the whole transfer of the second buffer of Channel 0 is completed. If MASK00 (Bit 1 of CCR0) is set, there is an interrupt request. |

Note: Each flag bit is cleared by writing '1' to its bit position.

9.1.2.13 DMAOR

0x8000.4054

| | | | |
|----------|------|---|-------|
| - | 2 | 1 | 0 |
| Reserved | PRMD | | DMAEN |

| Bits | Type | Function |
|------|------|---|
| 2:1 | R/W | Defines the channel priorities in case of simultaneous transfer requests for multiple channels. 11: ch0 > ch1 > ch2 10: ch2 > ch1 > ch0 01: ch1 > ch0 > ch2 00: ch1 > ch2 > ch0 (initial value) |

| | | |
|---|-----|--|
| 0 | R/W | DMA operation enable bit 1 = DMA operation is enabled. 0 = DMA operation is disabled. A specific DMA channel is enabled when both of this bit and the corresponding channel enable bit (DMENx) are set. |
|---|-----|--|

9.1.3 DMAC operation

For correct DMA operation, the DMA address register (ADR_x or ASR), DMA buffer size register (TNR_x), DMA channel control register (CCR_x), and DMA operation register (DMAOR) must be set properly. Then the DMAC performs DMA data transfers as follows.

- The DMAC checks if the corresponding channel enable bit (DMEN_x, Bit 0 of CCR_x) and the DMAEN (Bit 0 of DMAOR) are enabled.
- When there is a transfer request from internal or external I/O and the DMA transfer in the corresponding channel is enabled, the DMAC initiates DMA data transfers according to the bus size, transfer direction and bus mode.
- The DMAC ends data transfers and sets the corresponding interrupt flag (FLAG_x of FLAGR) when the whole buffer is transferred (when the internal count value equals TNR_x or TSR). If the interrupt mask bit of the channel is set (and the DMA interrupt is enabled in the interrupt controller), a DMA transfer end interrupt request is sent to the CPU core.

DMA Channel Priority

When the DMAC receives simultaneous DMA transfer requests, the channel with the higher priority is served first. The channel priorities are programmable in the DMAOR register.

DMA bus mode

Burst mode (for Channel 2)

Once the bus mastership is obtained, the transfer is performed continuously by the burst length (BURST, Bit 7 of CCR2) as long as nDMAREQ pin is driven high. Then the bus mastership is given to the CPU.

Exception mode (cycle-steal mode)

In the exception mode, the bus mastership is given to the CPU core whenever one transfer is completed

DMA transfer request

The DMA transfer request should be disabled by I/O device module.

9.2 MMC/ SPI Controller

The SPI is a high-speed synchronous serial port. This chapter describes the SPI communication with a MMC device.

The communication between CP (master) and MMC is controlled by the CP. The data transmission starts when the CS (chip-select) goes LOW and ends when the CS goes HIGH.

SPI-MMC messages are built from command, response and data-block tokens. Every command, response and data block is built with one byte (8-bit). Generally every MMC token transferred on the data signal is protected by CRC bits. But MMC offers also a non-protected mode that allows a system, built with reliable data links to exclude the hardware or firmware required for CRC generation and verification.

In the non-protected mode, the CRC bits of the command, response and data tokens are still required in the tokens; they are, however, defined as "don't care" for the transmitters and are ignored by the receivers.

MMC is initialized in the non-protected mode. The CP can turn this option on and off using the CRCONOFF command (CMD39). We assume that CRC is processed by software.

9.2.1 External Signals

| Pin Name | Type | Description |
|----------|------|----------------------------------|
| SSDO | O | MMC card controller data output |
| SSDI | I | MMC card controller data input |
| SSCLK | O | MMC card controller clock output |
| nSSCS | O | MMC card controller chip select |

9.2.2 Registers (SPI Mode)

| Address | Name | Width | Default | Description |
|-------------|----------|-------|---------|---------------------------|
| 0x8001.5000 | SPICR | 0x20 | 0x20 | SPI control register |
| 0x8001.5004 | SPISR | 0x0 | 0x0 | SPI status register |
| 0x8001.5008 | XCHCNT | 0x0 | 0x0 | Number of exchange data |
| 0x8001.500C | TXBUFF | 0x0 | 0x0 | TX data buffer (8*8 bits) |
| 0x8001.5010 | RXBUFF | 0x0 | 0x0 | RX data buffer (8*8 bits) |
| 0x8001.5014 | TestReg1 | 0x0 | 0x0 | Test register 1 |
| 0x8001.5018 | TestReg2 | 0x0 | 0x0 | Test register 2 |
| 0x8001.501C | ResetReg | 0x0 | 0x0 | SPI reset register |
| 0x8001.5024 | TicReg | 0x0 | 0x0 | Tic register |

Table 9-3 SPIMMC Controller Register Summary

9.2.2.1 SPIMMC Control Register (SPICR)

| | | | | | | | 0x8001.5000 |
|------|----------|--|---------|----------|------|-------|-------------|
| | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | DataRate | CS | XCHMode | TestMode | LOOP | SPIEN | XCH |
| Bits | Type | Function | | | | | |
| 7 | - | Reserved | | | | | |
| 6 | R/W | This bit sets the baud rate (SPICLK) 0 : SPICLK=BCLK/2 1 : SPICLK=BCLK/4 | | | | | |
| 5 | R/W | This bit is the Chip select signal. To communicate with external devices (MMC), CP asserts 0 in this bit. 0 = CP can exchange data with external device (MMC) 1 = CP cannot exchange data with external device (MMC) | | | | | |
| 4 | R/W | This bit determines the direction of transfer 0 = CP have valid data to send to MMC (send mode) 1 = CP have valid data to receive from MMC (receive mode) | | | | | |
| 3 | R/W | 0 = Normal operation 1 = the SPI-MMC block is in TIC mode. In this mode the Clock source is not BCLK/2 but TCLK that is made in the block. | | | | | |

| | | |
|---|-----|--|
| 2 | R/W | 0 = Normal operation 1 = The SPI-MMC block is in loopback mode. In the loopback mode the transmitter output is internally connected to the receiver input. MISO is internally connected to MOSI. |
| 1 | R/W | 0 = SPI master disable (reduce power consumption) 1 = SPI master enable The SPI must be enabled before initiating an exchange and should be disabled after the exchange is complete to reduce the power consumption. |
| 0 | R/W | This bit triggers the state machine to generate clocks at the selected bit rate. 1 = Initiate exchange 0 = No exchange occurs |

9.2.2.2 SPIMMC Status Register (SPISR)

0x8001.5004

| | | | | | | | | | |
|------|---------|--------|--|--|--|--|--|--|--|
| 7 | 6 | 5 | | | | | | | |
| TXET | XCHDONE | RXFULL | | | | | | | |

| Bits | Type | Function |
|------|------|--|
| 7 | R | When the TX data buffer is empty this bit is set and a serial peripheral interrupt is generated. The bit is reset by reading the SPISR. |
| 6 | R | When the exchange is completed between CP and MMC this bit is set and a serial peripheral interrupt is generated. The bit is reset by reading the SPISR. |
| 5 | R | When the RX data buffer is full this bit is set and a serial peripheral interrupt is generated. The bit is reset by reading the SPISR. |
| 4:0 | - | Reserved |

9.2.2.3 SPIMMC XCH Counter Register (XCHCNT)

0x8001.5008

| | | | | | | | | | |
|-------------|---|---|---|---|---|---|---|---|---|
| 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| XCH COUNTER | | | | | | | | | |

| Bits | Type | Function |
|------|------|--|
| 9:0 | R/W | Number of bytes to be exchanged between CP and SPI |

9.2.2.4 SPIMMC TX Data Buffer Register (TXBUFF)

0x8001.500C

This 8-bit register is the entry point of the TX FIFO. When CP writes an 8-bit data to this register, the SPI-MMC block shifts the content of the TX FIFO and appends the new data to the FIFO.

| | | | | | | | |
|---------------------|---|---|---|---|---|---|---|
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TX FIFO ENTRY POINT | | | | | | | |

| Bits | Type | Function |
|------|------|-----------------------|
| 7:0 | W | TX FIFO's Entry Point |

9.2.2.5 SPIMMC RX Data Buffer Register (RXBUFF)

0x8001.5010

This register is the access point of the RX FIFO. When CP reads one data item from this register, the SPI-MMC block shifts the RX FIFO so that the next data item becomes available at this location.

| | | | | | | | |
|----------------------|---|---|---|---|---|---|---|
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RX FIFO ACCESS POINT | | | | | | | |

| Bits | Type | Function |
|------|------|------------------------|
| 7:0 | R | RX FIFO's Access Point |

9.2.2.6 SPIMMC Reset Register (ResetReg)

| | | | | | | | |
|--|--|--|--|--|--|--|-------------|
| | | | | | | | 0x8001.501C |
| | | | | | | | 0 |
| | | | | | | | RESET |

| Bits | Type | Function |
|------|------|---|
| 7:1 | - | Reserved |
| 7:0 | R/W | When CP writes 0 to this location, all registers and counters of the SPI-MMC block are cleared. |

9.2.3 Timings

All timing diagrams use the following schematics and abbreviations.

| Name | Description | Name | Description |
|------|--------------------------|----------|----------------|
| H | Signal is HIGH (logic 1) | Busy | Busy token |
| L | Signal is LOW (logic 0) | Command | Command token |
| X | Don't care | Response | Response token |
| Z | High Impedance State | DataBlk | Data token |
| * | Repeater | | |

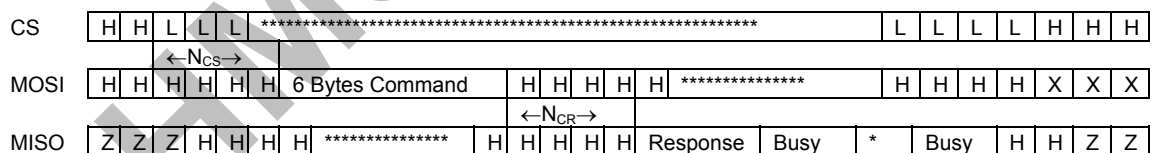
All timing values are defined as outlined below.

Command/Response

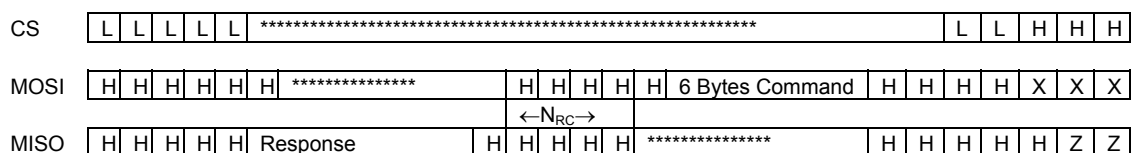
Host command to card response: card is ready



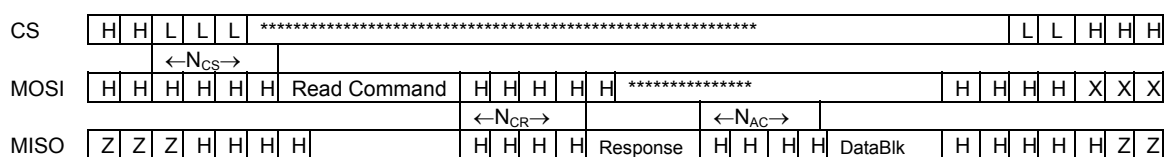
Host command to card response: card is busy



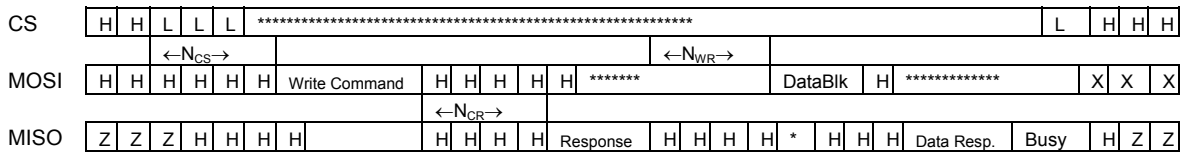
Card response to host command:



Data read



Data write



Timing constants definitions

| Name | Minimum | Maximum | Unit |
|-----------------|---------|---------|----------------|
| N _{CS} | 0 | - | 8 Clock Cycles |
| N _{CR} | 1 | 2 | 8 Clock Cycles |
| N _{RC} | 1 | - | 8 Clock Cycles |
| N _{AC} | 1 | - | 8 Clock Cycles |
| N _{WR} | 1 | - | 8 Clock Cycles |

9.2.4 SPI Operation for MMC

After CP writes a sequence of data to the TX FIFO, the content of the FIFO is loaded into the TX shift register and shifted out serially one byte at a time. When all elements in the TX FIFO are transferred to the TX shift register, the SPI-MMC issues an interrupt to CP, which may fill the TX FIFO for further data transfer. Serial input data is shifted into the RX shift register. After 8 bits are shifted in, the content of the RX shift register is copied into the RX FIFO. When the RX FIFO is full, the SPI-MMC issues an interrupt to CP through the SPIIRQ signal. CP reads the content of the RX FIFO in an interrupt service routine. The timing and control block produces all necessary control signals of the SPI-MMC block including SPICLK. The frequency of SPICLK signal is programmable. SPI-MMC transfer's protocol is command and response. Whenever CP sends a command to MMC (via SPI), MMC sends CP (via SPI) a response. The length of the response depends on the command – e.g. there are 1-, 6-, and 17-byte responses. There is only 6 bytes in command.

Consider the sequence of operations that occur in a read transfer.

1. CP sends a reset signal to the SPI-MMC block. In other word, CP writes "0" to bit in the ResetReg register. The signal is used to clear counters inside the block. Before new exchange begins and the content of XHCOUNTER is changed, and transmit mode is changed (XCHMODE BIT in the SPICR), CP must send a reset signal to the SPI-MMC block.
2. First, CP set up the SPICR register. In this example, XCHMODE is send mode.
3. CP writes number to send into XHCOUNTER register.
4. CP writes "Data read command (CMD17)" into the TX FIFO.
5. CP asserts CS signal. In other words, CP write 0 to CS bit in the SPICR.
6. CP sends a start signal to SPI-MMC. In other word, CP set XCH bit in the SPICR.
7. The SPI-MMC block sends out 6 bytes of command data from TX FIFO through TX shift register.
8. The SPI-MMC block issues the interrupt after it send all data in TX FIFO.
9. The CP reads the SPISR register in The SPI-MMC block and disable start signal (reset XCH bit). In other words, CP writes the SPICR register.
10. CP sends a reset signal to the SPI-MMC block. In other word, CP writes 0 to bit in the ResetReg register. The signal is used to clear counters inside the block. Before new exchange begins and the content of XHCOUNTER is changed, and transmit mode is changed (XCHMODE BIT in the SPICR), CP must send a reset signal to the SPI-MMC block.
11. CP changes transmit mode (XCHMODE is receive mode).
12. The CP writes number to be received into XHCOUNTER register.
13. CP sends a start signal to SPI-MMC (set XCH bit).
14. Then SPI-MMC controller receives response from MMC.
15. After SPI-MMC receives 1 byte (for CMD17 command), it sets XCH DONE status bits and it issues an interrupt to a CP.
16. The CP reads the SPISR register in the SPI-MMC block and disable start signal (reset XCH bit). In other words, CP writes the SPICR register.
17. The CP reads data RX FIFO.

18. After CP takes this response data and examine it, CP act as response data. If there is no error indication in response, CP informs SPI-MMC block that MMC sends data to it.
19. CP sends a reset signal to the SPI-MMC block. In other words, CP writes 0 to bit in the Reset register. The signal is used to clear counters inside the block. Before new exchange begins and the content of XCHCOUNTER is changed, and transmit mode is changed (XCHMODE BIT in the SPICR), CP must send a reset signal to the SPI-MMC block.
20. The CP writes number to be received into XCHCOUNTER register.
21. CP sends a start signal to SPI-MMC (set XCH bit).
22. The SPI-MMC block receives data from MMC (for example, data length is from 4 byte to 515 byte).
23. If SPI-MMC receives data like RX FIFO size, SPI-MMC block sets the "RX FIFO full" status bit and issues an interrupt to CP. At this time SPICLK disable start signal for prevention of RX FIFO overrun. If CP takes all data in RX FIFO, CP sends a start signal and receives response to remain. Repeat it.
24. After SPI-MMC block receive all data from MMC, it sets the XCH DONE status bit and issues an interrupt to CP.
25. The CP reads the SPISR register in the SPI-MMC block and disable start signal (reset XCH bit). In other words, CP writes the SPICR register.
26. After CP takes last data from RX FIFO, CP de-asserts CS signal.

9.2.5 Multimedia Card Host Controller

This document will describe the basic operation about the MMC Host controller for the ARM7202. This controller operates in MMC mode to communicate with Multimedia Card.

9.2.6 Registers

The MMC host controller has 12 registers. Following table shows the register map and its reset value.

| Address | Name | Width | Default | Description |
|-------------|--------------------|-------|---------|-------------------------------|
| 0x8001.5040 | mmcModeReg | 9 | | MMC Mode Register |
| 0x8001.5044 | mmcOperationReg | 9 | | MMC Operation Register |
| 0x8001.5048 | mmcStatusReg | 15 | | MMC Status Register |
| 0x8001.504C | mmclntrEnReg | 7 | | MMC interrupt Enable Register |
| 0x8001.5050 | mmcBlockSizeReg | 11 | | MMC Block Size Register |
| 0x8001.5054 | mmcBlockNumberReg | 16 | | MMC Block Number Register |
| 0x8001.5058 | mmcTimePeriodReg | 24 | | MMC Time Period Register |
| 0x8001.505C | mmcCMDBufferReg | 6 | | MMC Command Buffer Register |
| 0x8001.5060 | mmcARGBufferReg | 32 | | MMC ARG Buffer Register |
| 0x8001.5064 | mmcRESPBufferReg | 32 | | MMC RESP Buffer Register |
| 0x8001.5068 | mmcDATABufferReg | 32 | | MMC Data Buffer Register |
| 0x8001.507C | mmcReadyTimeoutReg | 24 | | MMC Ready Timeout Register |

Table 9-4 MMC Host Controller Register Summary

9.2.6.1 MMC Mode Register

| | | | | | | | |
|-------------|---------|--------|-----------|---------|-------|----------|--------|
| 0x8001.5040 | 8 | 7 | 6 | 5 : 3 | 2 | 1 | 0 |
| | IntrReq | DmaReq | SoftReset | ClkRate | DmaEn | Reserved | Enable |

| Bits | Type | Function |
|------|------|--|
| 8 | R | Interrupt Request Signal. |
| 7 | R | DMA Request Signal. |
| 6 | R/W | Software Reset. |
| 5:3 | R/W | Clock Rate Divisor Value. BCLK is 50MHz. |

MMCCLK speed will be one of these values according to divisor value.

- 0 for 25MHz (1/2 BCLK)
- 1 for 12.5MHz (1/4 BCLK)
- 2 for 6.25MHz (1/8 BCLK)
- 3 for 3.125MHz (1/16 BCLK)

| | | |
|---|-----|---------------------------------|
| | | 4 for 1.5625MHz (1/32 BCLK) |
| | | 5 for 0.78125MHz (1/64 BCLK) |
| | | 6 for 0.390625MHz (1/128 BCLK) |
| | | 7 for 0.1953125MHz (1/256 BCLK) |
| 2 | R/W | DMA Enable. |
| 1 | - | Reserved |
| 0 | R/W | MMC Enable. |

Table 9-5 MMC Mode Register

MMC Controller can be reset by the two methods. First is the system reset. In this case, most registers are initialized to the default value. But two registers (response FIFO and data FIFO) are not initialized. Second is the software reset. It is accomplished by writing the 7th bit of MMC mode control register with 1. Its effect is same with the first. Following table shows the MMC mode control register.

This controller sends the DMA request signal in two cases (Rx & Tx). And if you want to use DMA, you must set the DmaEn bit of MMC Mode register. For Rx, when the number of data in the FIFO is more then zero, it generates the request signal. For Tx, when the number of data in the FIFO are less then eight, it generate the request signal.

Operation frequency can be controlled by setting the ClkRate bit of MMC Mode register. Divisor controls the rate of MMC clock (MMCCLK). Assume that BCLK has 50MHz frequency.

9.2.6.2 MMC Operation Register

| | | | | | | | |
|-------------|-------------|---|--------|------------|----------------|-------|-------------|
| | | | | | | | 0x8001.5044 |
| 8 | 7 | 6 | 5 | 4:3 | 2 | 1 | 0 |
| BusyCheck | StreamEn | WriteEn | DataEn | RespFormat | Initialization | ClkEn | StartEn |
| Bits | Type | Function | | | | | |
| 8 | R/W | Current command needs the busy check after command operation. | | | | | |
| 7 | R/W | Define stream mode(1 = stream mode, 0 = block mode) | | | | | |
| 6 | R/W | 1 = write, 0 = read. default is read | | | | | |
| 5 | R/W | Indicate that current command contains the data operation | | | | | |
| 4:3 | R/W | Response format (No response, R1, R2, and R3) 0 for No response 1 for format R1 2 for format R2 3 for format R3 | | | | | |
| 2 | R/W | Add the 128 clocks before sending the command | | | | | |
| 1 | R/W | Enable the clock | | | | | |
| 0 | R/W/C | Start the mmc operation | | | | | |

Table 9-6 MMC Operation Register

All Multimedia Cards require at least 74 clock cycles prior to starting bus communication. and the clock frequency must be less then the Open-Drain frequency($F_{od}=0.5\text{Mhz}$). Therefore the host controller must do these during power-on.

For generating 74 clock cycles, set initialization bit of MMC Operation register. If initialization bit is set, then the controller will send additional 128 clocks before send start bit. Although this bit is zero, the controller sends 16 clocks before the start bit for safe operation. And add 8 clocks after the stop bit.

MMC has the four types of the response (No response, R1, R2, and R3). And each format is similar to the command format. But you need not know what they shape. You just only need to know the length of response to be stored after the response end. R1 and R3 have one word. And R2 has four words. Its contents are different according to the each command. You must analysis this content according to the each command after operation. And the response format can be specified by the RespFormat bits of the operation control register.

9.2.6.3 MMC Status Register

| | | | | | | | |
|---|------------|---------|--------------|------------|------------|-------------|-------------|
| | | | | | | | 0x8001.5048 |
| | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| | Detected_n | DetIntr | ReadyTimeout | RespCrcErr | DataCrcErr | RespTimeout | DataTimeout |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

| CardBusy | DataOperEnd | DataTransEnd | CmdRespEnd | ClkOnv | RxFifoFull | TxFifoEmpty | RxFifoReady |
|----------|-------------|---|------------|--------|------------|-------------|-------------|
| Bits | Type | Function | | | | | |
| 14 | R | card detection status | | | | | |
| 13 | R/WC | card detection interrupt | | | | | |
| 12 | R/WC | ready timeout error status | | | | | |
| 11 | R/WC | response data CRC error | | | | | |
| 10 | R/WC | Rx or Tx data CRC error | | | | | |
| 9 | R/WC | response timeout error | | | | | |
| 8 | R/WC | data timeout error | | | | | |
| 7 | R | card busy status | | | | | |
| 6 | R/WC | MMC operation status | | | | | |
| 5 | R/WC | data transfer status for Rx and Tx | | | | | |
| 4 | R/WC | command response end status | | | | | |
| 3 | R | clock status | | | | | |
| 2 | R | Rx FIFO is full | | | | | |
| 1 | R | Tx FIFO is empty | | | | | |
| 0 | R | Rx FIFO contains more than the one word | | | | | |

Table 9-7 MMC Status Register

WC : To clear these bit, you need to write any dummy value to these register.

9.2.6.4 MMC Interrupt Enable Register

| | | | | | | | 0x8001.504C |
|------------|------------|---|-----------|------------------|----------------|--------------|-------------|
| 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| mmcDetIntr | croErrIntr | timeoutIntr | OpEndIntr | dataTranfEndIntr | CmdRespEndIntr | dataFifoIntr | |
| Bits | Type | Function | | | | | |
| 6 | R/W | Card detection interrupt (insertion, remove). This interrupt can be used to check the card insertion and removal | | | | | |
| 5 | R/W | CRC errors interrupt (response CRC error, data CRC error). This interrupt is generated in two cases (response CRC error, data CRC error). If CRC interrupt is generated, MMC host controller will stop the current operation. | | | | | |
| 4 | R/W | Timeout interrupt (Response timeout, data timeout, and ready timeout). MMC host controller generates three types of timeout interrupt (response, data, and busy). These three timeout values are specified in MMC Time Period register and MMC Ready Timeout register. ISR can check the each timeout interrupt by reading MMC Status register. | | | | | |
| 3 | R/W | Operation end interrupt. This interrupt is generated when all operation is finished. Before the start operation, you need to set MMC Operation register. This register contains information about the operation. If an operation does not need the response and data, this interrupt is generated after the end of command transfer. If an operation just needs response, it is generated when MMC host controller receives the response. If an operation needs the data operation, it is generated when MMC host controller finish all operation including busy checking. | | | | | |
| 2 | R/W | Data transfer end interrupt. This interrupt is generated when MMC Host controller receives or sends the data specified by MMC Block Size register and MMC Block Number register. In most case, Multimedia Card goes into the ready state to write internal buffer data into flash memory. So after data transfer, Multimedia Card can be ready state for some time. This interrupt can be used to inform the data transfer end without busy check. | | | | | |
| 1 | R/W | Command response ends interrupt. This interrupt is generated when MMC Host controller finishes receiving of command response. For data read operation, because the response and data is transmitted currently, you can use this interrupt to check the data FIFO and the response FIFO independently. | | | | | |
| 0 | R/W | Data fifo interrupt (Rx fifo full, Tx fifo empty). This interrupt is generated in two cases (Rx FIFO full, Tx FIFO empty). You can use this interrupt to check the FIFO status during the read and write operation. And by reading the status register, you can know which interrupt is taken. | | | | | |

Table 9-8 MMC Interrupt Enable Register

MMC Host controller has the seven interrupt sources (Rx/Tx Fifo interrupt, command response interrupt, data transfer end interrupt, MMC operation interrupt, timeout interrupt, CRC error interrupt and MMC detection interrupt). Setting the each interrupt enable bit of MMC Interrupt Enable register can enable each interrupt.

We can consider the card detection in the two cases.

Firstly, In case that MMC Host controller is not enabled, clock is not supplied into the controller. So the detection logic can operate without the clock. To detect the MMC without clock, the detection signal is passed into the interrupt request directly. If the card detection interrupt is enabled, interrupt signal will be passed into the interrupt controller. Secondly, If MMC host controller is enabled; it means the detection logic now operates with clock. In this case, MMC Host controller detects both card insertion and card removal. When this interrupt is generated, you can detect if current interrupt is the card insertion or removal by reading the Detected_n bit of MMC Status register. If the value is zero, it indicates card insertion. If not, it notifies card removal.

9.2.6.5 MMC Block Size Register

| | | |
|-------------------|-------------|---|
| 10 | ... | 0x8001.5050 |
| Max. Block Length | | |
| <hr/> | | |
| Bits | Type | Function |
| 10:0 | R/W | Maximum Block Length Definition up to 2048 bytes. |

Table 9-9 MMC Block Size Register

9.2.6.6 MMC Block Number Register

| | | |
|----------------------|-------------|---|
| 15 | ... | 0x8001.5054 |
| Max. Number of Block | | |
| <hr/> | | |
| Bits | Type | Function |
| 15:0 | R/W | Maximum Number of Block Transfer Definition up to 64K blocks. |

Table 9-10 MMC Block Number Register

9.2.6.7 MMC Time Period Register

| | | | | | | |
|-------------|-------------|-------------------------|-------------|-----|---|-------------|
| 23 | ... | 16 | 15 | ... | 0 | 0x8001.5058 |
| RespTimeout | | | DataTimeout | | | |
| <hr/> | | | | | | |
| Bits | Type | Function | | | | |
| 23:16 | R/W | Response Timeout Period | | | | |
| 15:0 | R/W | Data Timeout Period | | | | |

Table 9-11 MMC Time Period Register

9.2.6.8 MMC Command Buffer Register

| | | |
|----------------|-------------|-----------------|
| 5 | ... | 0x8001.505C |
| Command Buffer | | |
| <hr/> | | |
| Bits | Type | Function |
| 5:0 | R/W | Command Buffer |

Table 9-12 MMC Command Buffer Register

9.2.6.9 MMC Argument Buffer Register

| | | |
|-----------------|-------------|-----------------|
| 31 | ... | 0x8001.5060 |
| Argument Buffer | | |
| <hr/> | | |
| Bits | Type | Function |
| 31:0 | R/W | Argument Buffer |

Table 9-3-6-9 MMC Argument Buffer Register

9.2.6.10 MMC Response Buffer Register

| | | | |
|-----------------|-------------|-----------------|-------------|
| 31 | ... | 0 | 0x8001.5064 |
| Response Buffer | | | |
| <hr/> | | | |
| Bits | Type | Function | |
| 31:0 | RO | Response Buffer | |

Table 9-13 MMC Response Buffer Register

This controller has two FIFO, which are response and data FIFO. Each has 4-word depths and 8-word depths. And Both FIFOs are cleared at the start of the command. If there were some data before starting, incorrect data will be transmitted, so you have to confirm that the FIFO is empty to writing any value into the status register. There is no way to write the MMC Resp Buffer directly. This Register can be written only when the Response from MMC Card is received.

For data FIFO, it used two modes. If the current operation is read, it will be used the Rx FIFO, if not, the Tx FIFO.

9.2.6.11 MMC Data Buffer Register

| | | | |
|-------------|-------------|-----------------|-------------|
| 31 | ... | 0 | 0x8001.5068 |
| Data Buffer | | | |
| <hr/> | | | |
| Bits | Type | Function | |
| 31:0 | R/W | Data Buffer | |

Table 9-14 MMC Data Buffer Register

9.2.6.12 MMC Ready Timeout Register

| | | | |
|--------------|-------------|----------------------|-------------|
| 23 | ... | 0 | 0x8001.507C |
| ReadyTimeout | | | |
| <hr/> | | | |
| Bits | Type | Function | |
| 23:0 | R/W | Ready Timeout Period | |

Table 9-15 MMC Ready Timeout Register

9.2.7 Basic Operation in MMC Mode

MMC command format consists of six parts. Four parts (start bit, transmitter bit, CRC, and stop bit) are automatically generated by MMC Host controller. For remain two parts (command index, argument), you must inform the MMC Host controller by setting registers properly.

After power-on, all Multimedia Cards need at least 74 clock cycles prior to starting the operation. It can be achieved by setting the third bit (Initialization) of the MMC Operation register. If set, MMC Host controller sends 128 clock cycles prior to sending start bit.

In case of data operation, you need to define the type of operation.

For example, at the case of block write, both DataEn and WriteEn are '1'. To enable stream read, both DataEn and StreamEn are '1' while WriteEn is '0'.

For some command, it needs the busy check after the command end. In this case, busy check bit of MMC Operation register is must be set (For more details, refer to "Multimedia Card Product Manual").

Finally, to initiate operation, write '1' to StartEn and ClkEn. Then MMC Host controller starts to send command to Multimedia Cards. And StartEn bit is cleared automatically when the MMC Host controller finishes current operation. ClkEn bit makes MMC clock to be enabled. In this case, the MMC clock (MMCCLK) is generated during the operation. If this bit is zero, clock to operate control block is not generated.

You can check the end of response or operation from MMC Status register. This register also contains lots of useful information about what MMC Host controller is doing.

If the current operation does not contain data operation, you just need to poll CmdRespEnd or DataOperEnd bit. But if not, you need to have another step prior to starting. If current command requires multiple block operation, you must inform the block length and the number of block to be transferred to MMC Host controller. These controllers (mmcBlockSizeReg, mmcBlockNumberReg) can specify up to 2048 bytes for the length of block, and 64K blocks for the number of block.

Following shows the procedure for the write and read operation.

9.2.7.1 Write Operation

MMC Host controller starts the sending of data at the end of response end. And if the controller does not receive response during a specified period, it will generate response timeout error. Anyway, after the response end, if Tx FIFO is not ready (FIFO is empty), waits until the data is ready. The data transfer can be done by the three methods (polling, interrupt, and DMA (Direct Memory Access)). Polling method checks the Tx FIFO empty bit of the status register and if it is empty, you can write less than the eight word. And again wait until Tx FIFO is empty. Repeat this procedure until the operation end bit is set. Interrupt method uses the Tx FIFO empty interrupt. For every interrupt, you can write the eight words. And exit the ISR (Interrupt Service Routine). And then wait for the next interrupt. DMA method uses not ARM720T core but the DMA controller, so you must program the DMA controller before start operation. MMC Host controller must set the DmaEn bit of MMC Mode register. DMA request signal is generated whenever the number of data in Tx FIFO is less than equal to seven.

9.2.7.2 Read Operation

Multimedia Card can send both response and data currently after receiving the command from the host controller. So, MMC Host controller waits the response and data at the same time. Therefore, you must check response and data concurrently. Or you can use the response end interrupt. But in most case, after the response end, you can start read data from the Rx FIFO. This is reason why the Rx FIFO sizes with the eight words (32*8 cycles), so to fill it, the controller needs the 256 cycles. The data transfer also can be done by the three methods like the write operation. Polling method checks the Rx FIFO ready bit of the status register. This bit is activated when Rx FIFO has more than two word data. So you just read two times when you check this bit is set. Interrupt method uses the Rx FIFO full interrupt. Because the Rx FIFO has eight word depths, whenever interrupt is called, ISR reads the eight words form the Rx FIFO. In the case of the Rx FIFO full, you don't worry about it because the MMC Host controller stops the output clock not to loss on the bus. In DMA mode, the DMA request is generated when the Rx FIFO has one more words.

* Note: Errata sheet (version 1.0) includes contents of the lower subject
[Subject] A way for using both MMC and SPI mode on a system board

9.3 SMC Controller

This SmartMedia™ Card Controller is an Advanced Microcontroller Bus Architecture (AMBA) compliant System-on-a-Chip peripheral providing an interface to industry-standard SmartMedia™ Flash Memory Card. A channel has 8 control signal outputs and 8 bits of bi-directional data ports.

FEATURES

- One 3.3V SmartMedia support
- 4MB to 128MB media (both Flash and Mask ROM type)
- Interrupt mode support when erase/write operation is finished
- Unique ID SmartMedia support
- Multi-page DMA access
- Marginal timing operation settable.

9.3.1 External Signals

| Pin Name | Type | Description |
|-----------|------|---|
| SMD [7:0] | I/O | Smart Media Card (SSFDC) 8bit data signals |
| nSMWP | O | Smart Media Card (SSFDC) write protect |
| nSMWE | O | Smart Media Card (SSFDC) write enable |
| SMALE | O | Smart Media Card (SSFDC) address latch enable |
| SMCLE | O | Smart Media Card (SSFDC) command latch enable |
| nSMCD | I | Smart Media Card (SSFDC) card detection signal |
| nSMCE | O | Smart Media Card (SSFDC) chip enable |
| nSMRE | O | Smart Media Card (SSFDC) read enable |
| nSMRB | I | Smart Media Card (SSFDC) READY/nBUSY signal. This is open-drain output so it requires a pull-up resistor. |

9.3.2 Registers

| Address | Name | Width | Default | Description |
|-------------|---------|-------|---------|---|
| 0x8001.6000 | SMCCMD | 32 | 0x0 | SmartMedia Card Command register |
| 0x8001.6004 | SMCADR | 27 | 0x0 | SmartMedia Card Address register |
| 0x8001.6008 | SMGDATW | 32 | 0x0 | Data written to SmartMedia Card |
| 0x8001.600C | SMCDATR | 32 | 0x0 | Data received from SmartMedia Card |
| 0x8001.6010 | SMCCONF | 8 | 0x0 | SmartMedia Card controller configuration register |
| 0x8001.6014 | SMCTIME | 20 | 0x0 | Timing parameter register |
| 0x8001.601C | SMCSTAT | 32 | 0x0 | SmartMedia Card controller status register |

Table 9-16 SmartMedia Controller Register Summary

9.3.2.1 SMC Command Register (SMCCMD)

| | | | | | | | | | | | | | | 0x8001.6000 | | | |
|------------------|------|---|----|----|----|----|----|------------------|----|----|----|----|----|-------------|----|--|--|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | | |
| Hidden Command 0 | | | | | | | | Hidden Command 1 | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | |
| Main Command | | | | | | | | Second Command | | | | | | | | | |
| Bits | Type | Function | | | | | | | | | | | | | | | |
| 31:24 | R/W | Hidden Command 0. This Unique ID feature will be available to 128Mb NAND Flash and upward density products to prevent illegal copy of music files. Unique ID is put into redundant block of SmartMedia. Use this hidden command to access redundant block that cannot be accessed with open command, This byte filed is ignored when user block is accessed. For more information, refer to SmartMedia Maker's datasheet. | | | | | | | | | | | | | | | |
| 23:16 | R/W | Hidden Command 1. Read ID command returns whether the SmartMedia card supports unique ID or not. Hidden 2 step command for Samsung is 30h-65h and for Toshiba is 5Ah- | | | | | | | | | | | | | | | |

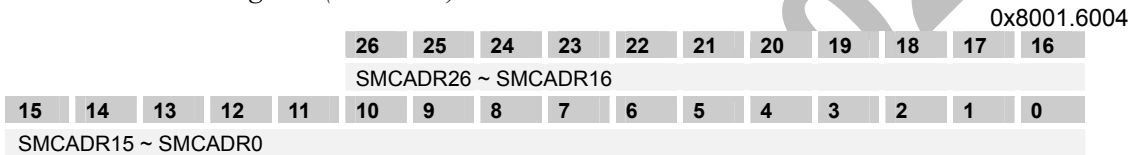
B5h. To return back to user block after accessing redundant block area, Reset command (FFh) should be carried out.

15:8 R/W There are 9 commands to operate SmartMedia card. This controller supports only parts of them (bold type). Set 1ST command into this byte field except writing to SmartMedia. For write operation, set this byte field to Serial Data Input (80h) and set Second Command byte field to Page Program (10h).

| Function | 1 ST cycle | 2 ND cycle | Function | 1 ST cycle | 2 ND cycle |
|--------------------------|-----------------------|-----------------------|---------------------|-----------------------|-----------------------|
| Serial Data Input | 80h | | Page Program | | 10h |
| Read 0 | 00h | | Block Erase | 60h | D0h |
| Read 1 | 01h | | Status Read | 70h | |
| Read 2 | 50h | | ID Read | 90h | |
| Reset | FFh | | | | |

7:0 R/W Set 2ND command here

9.3.2.2 SMC Address Register (SMCADR)

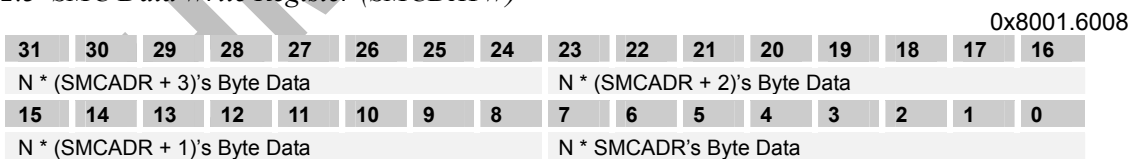


| Bits | Type | Function |
|------|------|----------|
|------|------|----------|

26:0 R/W SMC Address. SMC controller begins to operate after writing an address to SMCADR. Hence a valid command must be set to SMCCMD before writing to SMCADR. However, reset and status read commands activate SMC controller after writing to SMCCMD because they do not require an address.
Following table shows valid address range according to SmartMedia card size.

| Model | Valid Page Address |
|--------|--------------------|
| 4 MB | SMCADR0 ~ SMCADR21 |
| 8 MB | SMCADR0 ~ SMCADR22 |
| 16 MB | SMCADR0 ~ SMCADR23 |
| 32 MB | SMCADR0 ~ SMCADR24 |
| 64 MB | SMCADR0 ~ SMCADR25 |
| 128 MB | SMCADR0 ~ SMCADR26 |

9.3.2.3 SMC Data Write Register (SMCDATW)



| Bits | Type | Function |
|------|------|----------|
|------|------|----------|

31:0 R/W Four byte data written to this register will be sent to SmartMedia. SMC controller receives a 32bit data from host controller or DMA controller. Then It starts to transmit from least significant byte to most significant byte, one byte at a time. This SMC controller writes a whole page at a single write transaction, so it requires 132 times consecutive writing (528 = 512+16 bytes). A page program process is as follows:
1. Set SMCCMD to xxxx8010h (Sequential Data Input + Page Program), SMCADR to desired target page address space, and then write first 4 byte data onto SMCDATW. If DMA mode enabled, DMA interrupt will be repeated until it writes 528 byte data to SmartMedia. In normal mode, interrupt will be generated every 4 bytes write.
2. At the end of sequential data input, SmartMedia goes into page program mode by transmitting the second command to SmartMedia. Usually page program takes long time, no polling status register is recommended. SMC controller automatically generates write finish

interrupt when SmartMedia comes back to ready mode.

9.3.2.4 SMC Data Read Register (SMCDATR)

0x8001.600C

| | | | | | | | | | | | | | | | |
|------------------------------|----|----|----|----|----|----|----|------------------------------|----|----|----|----|----|----|----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| N * (SMCADR + 3)'s Byte Data | | | | | | | | N * (SMCADR + 2)'s Byte Data | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| N * (SMCADR + 1)'s Byte Data | | | | | | | | N * SMCADR's Byte Data | | | | | | | |

| Bits | Type | Function |
|------|------|--|
| 31:0 | R | <p>Four byte data read from SmartMedia is stored in this register. SMC controller receives a byte data from SmartMedia and stores it into 4 byte internal buffer to create 32bit data. First read byte data is stored at least significant byte and fourth byte data is stored at most significant byte of buffer. Host controller or DMA controller read this register to get 4 byte data at a time. This SMC controller reads a whole page at a single read transaction, so it requires 132 times consecutive reading. A page reading process is as follows:</p> <ol style="list-style-type: none"> 1. Set SMCCMD to xxxx00yyh (xxxx can be unique ID if redundant area accessed, yy is don't care. Only 00h command is valid. No 01h or 50h command supported) and then set SMCADR to target page address. 2. SMC controller will access SmartMedia with given command and address. 3. Interrupt (or DMA interrupt according to interrupt mode setting) will be generated after first four byte read. Like writing process, reading process reads a whole 528 byte in a page at a single transaction, so interrupt will be 132 times. <p>Against to write operation, there is no read finish interrupt because we can count the number of read transfers in software or can get the total access word size from BYTE COUNT of SMCSTAT.</p> |

9.3.2.5 SMC Configuration Register (SMCCONF)

0x8001.6010

| | | | | | | | |
|--------------|-------------|------------|--------------|---------|--------|--------------|-----------------|
| 31 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| POWER ENABLE | SAFE MARGIN | SMC ENABLE | CONT PAGE EN | INTR EN | DMA EN | UNIQUE ID EN | BIG CARD ENABLE |

| Bits | Type | Function |
|------|------|--|
| 31 | R/W | Power on bit. To activate SMC controller, set this bit. Reset will fall the controller into the deep sleep mode. |
| 30:7 | - | Reserved. Keep these bits to zero. |
| 6 | R/W | Safe margin enable bit. In normal mode, chip select signal changes simultaneously with read enable and write enable signals. But when this bit set, the duration of read and write enable signal applied to SmartMedia is reduced by 1 automatically. By enabling this, the rising edge of read and write enable signal will be earlier than the rising edge of chip enable, which guarantees latching data safely. |
| 5 | R/W | SMC controller enable bit. Reset this bit will make SMC controller stay in standby mode. No interrupt generated, no action occurred. |
| 4 | R/W | Continuous page read enable. If this bit set, then multi-page can be accessed in a single command and address setting. Usually DMA controller accesses multiple pages with a start address and a predefined size. Setting DMA access size in SMCTIME and enabling this bit will automatically read or write SmartMedia with DMA mode. |
| 3 | R/W | Interrupt enable. After reading a word or before writing a word, the interrupt bit of SMCSTAT will be set and interrupt will occur if INTR EN is enabled. If this bit is disabled, software must poll the interrupt flag of SMCSTAT to know the occurrence of an interrupt. After writing a whole page (or pages when CONT PAGE EN is enabled) to SmartMedia, write finish interrupt will also be generated to notice that the SmartMedia complete the write operation successfully. |
| 2 | R/W | DMA enable. If set, all interrupt during read or write data will be sent to DMA controller. However, write finish interrupt is a still normal interrupt. To minimize CPU burden and to maximize BUS utilization, enabling both interrupt and DMA mode together is recommended. |
| 1 | R/W | Redundant page enable. When use SmartMedia with unique ID and want to access redundant page area, set high. This bit cannot be cleared automatically, so in order to read |

| | | |
|---|-----|---|
| | | open page area clear this bit and set a reset command to SMCCMD. |
| 0 | R/W | Larger than 32MB SmartMedia support enable. When using 64MB or 128MB SmartMedia, set this bit high. |

9.3.2.6 SMC Timing Parameter Register (SMCTIME)

0x8001.6014

| | | | | | | | | | | | | | | | |
|--------------|----|----|----|--------------|----|----|----|--------------|----|----|----|----|----|----|----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| DMA SIZE | | | | WAIT COUNTER | | | | BYTE COUNTER | | | | | | | |
| | | | | | | 9 | 8 | | | | | | 2 | 1 | 0 |
| HIGH COUNTER | | | | | | | | LOW COUNTER | | | | | | | |

| Bits | Type | Function |
|-------|------|--|
| 31:28 | R/W | Multi-page DMA size bit. Maximum 15 pages are accessible at a time. 0000 = not defined. 0001 = 1 page 0010 = 2 pages ... 1111 = 15 pages |
| 27:24 | R/W | Wait counter maximum limit value. Waiting time delay between address latch and write data in page program mode or between address latch and read data in read ID mode and read status register is determined by this register. 0000 = 1 BCLK width 0001 = 2 BCLK width ... 1111 = 16 BCLK width |
| 23 | - | Reserved |
| 22:16 | R/W | Should set these bits as 0x7F to access full 512 bytes page at one access command (read or program). |
| 15:10 | - | Reserved |
| 9:8 | R/W | High pulse width value of read enable and write enable signal. The width must satisfy the AC characteristics of SmartMedia to guarantee correct transfer of data. With Safety Margin enable, width will be decreased by one. 00 = 1 BCLK width (0 BCLK with safety margin enable. Don't make this case) 01 = 2 BCLK width (1 BCLK with safety margin enable) 10 = 3 BCLK width (2 BCLK with safety margin enable) 11 = 4 BCLK width (3 BCLK with safety margin enable) |
| 7:3 | - | Reserved |
| 2:0 | R/W | Low pulse width value of read enable and write enable signal. The width must satisfy the AC characteristics of SmartMedia to guarantee correct transfer of data. With Safety Margin enable, width will be decreased by one. 000 = 1 BCLK width (0 BCLK with safety margin enable, Don't make this case) 001 = 2 BCLK width (1 BCLK with safety margin enable) ... 111 = 8 BCLK width (7 BCLK with safety margin enable) |

9.3.2.7 SMC Status Register (SMCSTAT)

0x8001.601C

| | | | | | | | |
|--|------------|-------|-------|-------|-------|-------|-------|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| CD INTR | nSMCE | SMCLE | SMALE | nSMWE | nSMRE | nSMWP | SMR/B |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| CURRENT COMMAND/CARD DETECT NOTIFICATION | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| EXTRA AREA | BYTE COUNT | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

| INTERNAL STATE | | | CARD DETECT | IRQ | DRQ | BUSY |
|----------------|------|---|----------------|-----|-----|------|
| Bits | Type | Function | | | | |
| 31 | R | Card Detect Interrupt. When card inserted or removed, card detect interrupt will be generated. In the interrupt service routine, look at this bit to identify interrupt type. | | | | |
| 30:24 | R | Current status of output signals. | | | | |
| 23:16 | R | Current active command. If in card detect interrupt, this byte shows 0xCD. | | | | |
| 15 | R | Set when extra area of a page is accessed. | | | | |
| 14:8 | R | Current address of a page in word units. | | | | |
| 7:4 | R | Shows internal state machine's state. | | | | |
| 3 | R | Set when SMC enable and SMC card inserted. It will be zero when card removed. | | | | |
| 2 | R | Interrupt flag | | | | |
| 1 | R | DMA interrupt flag | | | | |
| 0 | R | Reset shows SMC is in idle mode. Set means SMC in working mode. | | | | |

HMS30C7202

9.4 Sound Interface

The Sound Control Unit (SCU) is an interface block to transfer sound data to external speakers. The SCU is an interface block used to send data to the external speaker through the internal 8-bit DA converter. It can process 44.1/22.05/11.025/8KHz sampled 8-bit mono or 16-bit stereo sound data.

This unit has a 32-bit register to receive sound data from the CPU through DMA or interrupt mode. This unit requests the DMA or interrupt controller every 32-bit processing time, which depends on the sampling frequency. It has two separate signals for DAC that indicate the direction of data for the stereo sound. Either higher or lower byte of 16-bit stereo sound data can be played through the left or right speaker by programming the control register. During mono playback, this unit sends the same data for the left and right channels.

There are two test registers. Both these registers should be cleared during normal operation. TICCLK port is also assigned for production test only.

Features

- Sound playback
- Supports programmable sampling rate
- 32-bit internal data register for DMA
- Auto DMA request
- 8-bit resolution DAC control
- Supports non-overlapping left/right signal for DAC
- Supports test mode

9.4.1 External Signals

| Pin Name | Type | Description |
|----------|------|----------------------------|
| ADACR | O | Sound DAC output for Right |
| ADACL | O | Sound DAC output for Left |

9.4.2 Registers

| Address | Name | Width | Default | Description |
|-------------|-------|-------|---------|------------------|
| 0x8001.3000 | SCONT | 8 | 0x0 | Control register |
| 0x8001.3004 | SDADR | 32 | 0x0 | Data register |

Table 9-17 Sound Controller Register Summary

9.4.2.1 SCONT

| | | 0x8001.3000 | | | | | | | |
|----------|------|---|-----|-----|-----|----|------|---|-----|
| | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | MONO | DMA | POR | DAC | RL | SAMP | | INT |
| Bits | Type | Function | | | | | | | |
| 7 | R/W | 0 – stereo 1 – mono | | | | | | | |
| 6* | R/W | DMA request masking bit 0 - masking 1 – unmasking | | | | | | | |
| 5 | R/W | This bit should be cleared to minimize power consumption when not in use. 0 - power down mode 1 - normal mode | | | | | | | |
| 4 | R/W | DAC operation enable/disable. During disabled, DAC is in power save mode. 0 - DAC disable 1 - DAC enable | | | | | | | |
| 3 | R/W | When cleared, lower byte data goes to left speaker. (ADACL pin) 0 - lower byte data goes to ADACL pin 1 - lower byte data goes to ADACR pin | | | | | | | |
| 2:1 | R/W | Programmable sampling rate | | | | | | | |

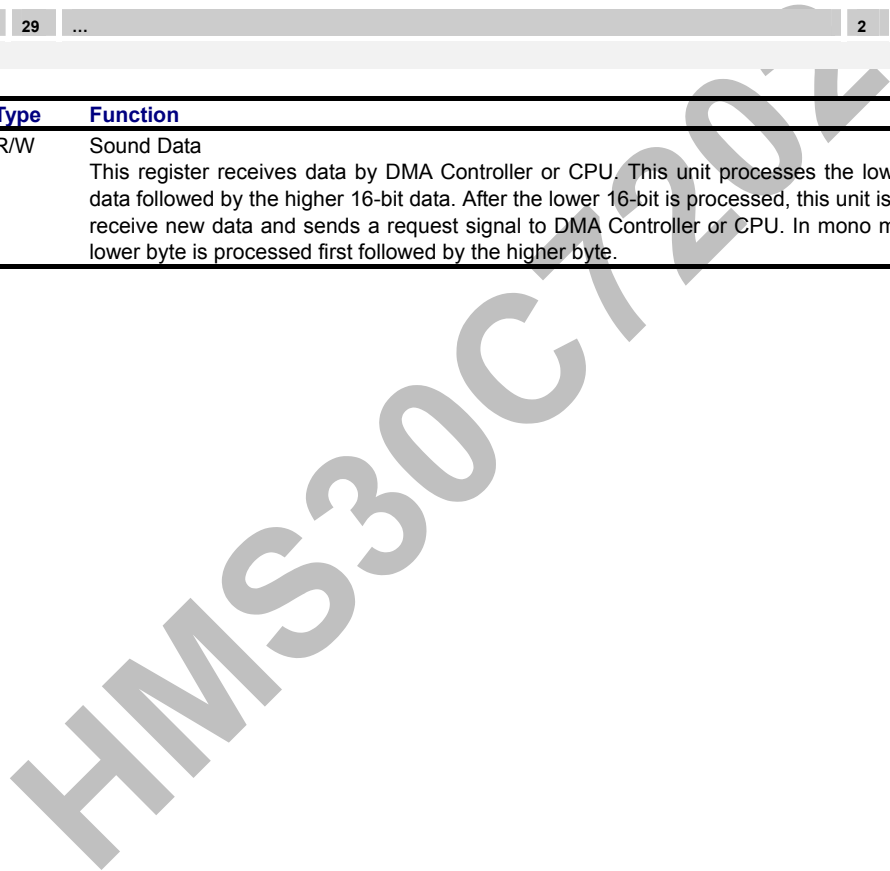
| | | |
|----|-----|---|
| | | 00 - 11.025KHz |
| | | 01 - 22.05KHz |
| | | 10 - 44.1KHz |
| | | 11 - 8KHz |
| 0* | R/W | 0 Interrupt request masking bit 0 - masking 1 - unmasking |

Note Those bits marked with an asterisk should not be enabled simultaneously during normal operation. (The programmer can select only one--either Interrupt or DMA mode.)

9.4.2.2 SDADR

This register can be programmed after setting Bit 5 of the SCONT register.

| | | | | 0x8001.3004 | | |
|-------|------|---|-----|-------------|---|---|
| 31 | 30 | 29 | ... | 2 | 1 | 0 |
| SDADR | | | | | | |
| Bits | Type | Function | | | | |
| 32 | R/W | Sound Data This register receives data by DMA Controller or CPU. This unit processes the lower 16-bit data followed by the higher 16-bit data. After the lower 16-bit is processed, this unit is ready to receive new data and sends a request signal to DMA Controller or CPU. In mono mode, the lower byte is processed first followed by the higher byte. | | | | |



9.5 USB Slave Interface

This section describes the implementation-specific options of USB protocol for a device controller. It is assumed that the user has knowledge of the USB standard. This USB Device Controller (USBDC) is chapter 9 (of USB specification) compliant, and supports standard device requests issued by the host. The user should refer to the Universal Serial Bus Specification revision 1.1 for a full understanding of the USB protocol and its operation. (The USB specification 1.1 can be accessed via the World Wide Web at: <http://www.usb.org>). The USBDC is a universal serial bus device controller (slave, not hub or host controller) which supports three endpoints and can operate half-duplex at a baud rate of 12 Mbps. Endpoint 0, by default is only used to communicate control transactions to configure the USBDC after it is reset or physically connected to an active USB host or hub. Endpoint 0's responsibilities include connection, address assignment, endpoint configuration and bus numeration.

The connected host that can get a device descriptor stored in USBDC's internal ROM via endpoint 0 configures the USBDC. The USBDC uses two separate 32 x 8 bit FIFO to buffer receiving and transmitting data to/from the host. The external pins dedicated to this interface are UVPO, UVP, UVMO, UVM, URCVIN, nUSBOE and USUSPEND. These signals should be connected to USB transceiver such as PDIUSBP11 provided by Philip Semiconductor. Refer to data sheet PDIUSBP11). The CPU can access the USBDC using Interrupt controller, by setting the control register appropriately. This section also defines the interface of USBDC and CPU.

* Notice: Don't use this USB device function with a LS device (like a USB mouse) in a same HUB.

FEATURES

- Full universal serial bus specification 1.1 compliant.
- Receiver and Transceiver have 32 bytes FIFO individually (this supports maximum data packet size of bulk transfer).
- Internal automatic FIFO control logic. (According to FIFO status, the USBDC generates Interrupt service request signals to the CPU)
- Supports high-speed USB transfer (12Mbps).
- There are two endpoint of transmitter and receiver respectively, totally three endpoints including endpoint 0 that has responsibility of the device configuration.
- CPU can access the internal USB configuration ROM storing the device descriptor for Hand-held PC (HPC) by setting the predefined control register bit.
- USB protocol and device enumeration is performed by internal state-machine in the USBDC.
- The USBDC only supports bulk transfer of 4-transfer type supported by USB for data transfer.
- Endpoint FIFO (Tx, Rx) has the control logic preventing FIFO overrun and under run error.

Note Product ID: 7210 Vendor ID: 05b4 * can be modified

Reference document - Hms30c7210_UsbDownLoad_V1.3.2Guide_with_Errata.pdf

[Location: <http://www.hynix.com> -SP-MCU-ARM Core Based-HMS30C7210 Reference Design Kit-Miscellany]

9.5.1 Block Diagram

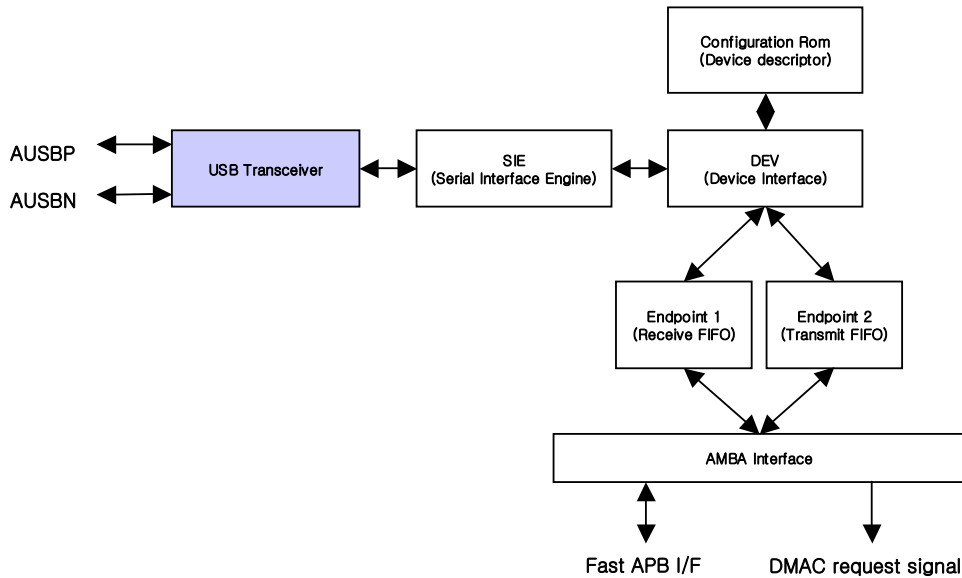


Figure 9-1 USB Block Diagram

The USB, Figure 9-2: USBD Block Diagram comprises the Serial Interface Engine (SIE) and Device Interface (DEV). The SIE connects to the USB through a bus transceiver, and performs NRZI conversion, bit un-stuffing, CRC checking, packet decoding and serial to parallel conversion of the incoming data stream. In outgoing data, it does the reverse, that is, parallel to serial of outgoing data stream and packetizing the data, CRC generation, bit stuffing and NRZI generation.

The DEV provides the interface between the SIE and the device's endpoint FIFO, ROM storing the device descriptor. The DEV handles the USB protocol, interpreting the incoming tokens and packets and collecting and sending the outgoing data packets and handshakes. The endpoints FIFO (RX, TX) give the information of their status (full/ empty) to the AMBA interface and AMBA I/F enable the CPU to access the FIFO's status register and the device descriptor stored in ROM. The AMBA interface generates a FIFO read/write strobe without FIFO's errors, based on APB signal timing. In case of data transmitting through TX FIFO (when USB generates an OUT token, AMBA I/F generates Interrupt to CPU), the user should set the transmitting enable bit in the control register. If the error of FIFO (Rx: overrun, TX: under-run) occurs, the AMBA I/F cannot generate FIFO read/ write.

9.5.2 Theory of Operation

The Hynix USB Core enables a designer to connect virtually any device requiring incoming or outgoing PC data to the Universal Serial Bus. As illustrated in Figure 9-2: USBD Block Diagram, the USB core comprises two parts, the SIE and DEV. The SIE connects to the Universal Serial Bus via a bus transceiver. The interface between the SIE and the DEV is a byte-oriented interface that exchanges various types of data packets between two blocks.

Serial Interface Engine

The SIE converts the bit-serial, NRZI encoded and bit-stuffed data stream of the USB into a byte and packet oriented data stream required by the DEV. As shown in Figure 9-3: Hynix Serial Interface Engine, it comprises seven blocks: Digital Phase Lock Loop, Input NRZI decode and bit-unstuff, Packet Decoder, Packet Encoder, Output bit stuff and NRZI encode, Counters, and the CRC Generation & Checking block. Each of the blocks is described in the following sections.

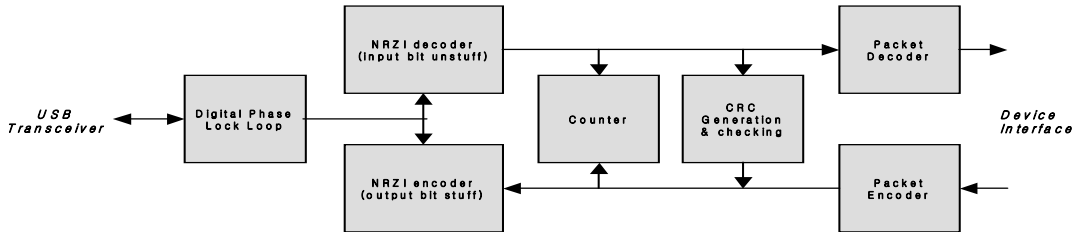


Figure 9-2 USB Serial Interface Engine

Digital Phase Lock Loop

The Digital Phase Lock Loop module takes the incoming data signals from the USB, synchronizes them to the 48MHz input clock, and then looks for USB data transitions. Based on these transitions, the module creates a divide-by-4 clock called the usbclock. Data is then output from this module synchronous to the usbclock.

Input NRZI decode and bit-unstuff

The Input NRZI decodes and bit-unstuff module extracts the NRZI encoded data from the incoming USB data. Transitions on the input serial stream indicate a 0, while no transition indicates a 1. Six ones in a row cause the transmitter to insert a 0 to force a transition, therefore any detected zero bit that occurs after six ones is thrown out.

Packet Decoder

The Packet Decoder module receives incoming data bits and decodes them to detect packet information. It checks that the PID (Packet ID) is valid and was sent without error.

After decoding the PID, the remainder of the packet is split into the address, endpoint, and CRC5 fields, if present. The CRC Checker is notified to verify the data using the incoming CRC5 field. If the packet is a data packet, the data is collected into bytes and passed on with an associated valid bit. Table 9-6: Supported PID Types shows the PID Types that are decoded (marked as either Receive or Both). At the end of the packet, either the packetok or packetnotok signal is asserted. Packetnotok is asserted if any error condition arose (bad valid bit, bit-stuff, bad PID, wrong length of a field, CRC error, etc.).

| PID Type | Value | Send/Receive | PID Type | Value | Send/Receive |
|----------|---------|--------------|----------|---------|--------------|
| OUT | 4'b0001 | Receive | DATA1 | 4'b1011 | Both |
| IN | 4'b1001 | Receive | ACK | 4'b0010 | Both |
| SOF | 4'b1101 | Receive | NAK | 4'b1010 | Send |
| SETUP | 4'b0000 | Receive | STALL | 4'b1110 | Send |
| DATA0 | 4'b0011 | Both | PRE | 4'b1100 | Receive |

Table 9-18 USB Supported PID Types

Packet Encoder

The Packet Encoder creates outgoing packets based on signals from the DEV. Table 9-6: Supported PID Types shows the PID Types that can be encoded (marked as Send or Both). For each packet type, if the associated signal sends type is received from the DEV, the packet is created and sent. Upon completion of the packet, packettypesent is asserted to inform the DEV of the successful transmission. The Packet Encoder creates the outgoing PID, grabs the data from the DEV a byte at a time, signals the CRC Generator to create the CRC16 across the data field, and then sends the CRC16 data. The serial bits are sent to the Output bit stuff and NRZI encoder.

Output bit stuff and NRZI encoder

The Output bit stuff and NRZI encoder takes the outgoing serial stream from the Packet Encoder, inserts stuff bits (a zero is inserted after six consecutive ones), and then encodes the data using the NRZI encoding scheme (zeroes cause a transition, ones leave the output unchanged).

Counter block

The Counter block tracks the incoming data stream in order to detect the following conditions: reset, suspend, and turnaround. It also signals to the transmit logic (Output NRZI and bit stuff) when the bus is idle so transmission can begin.

Generation and Checking block

The Generation and Checking block checks incoming CRC5 and CRC16 data fields, and generates CRC16 across outgoing data fields. It uses the CRC polynomial and remainder specified in the USB Specification Version 1.1.

Device Interface

The DEV shown in Figure 9-4: Device Interface works at the packet and byte level to connect a number of endpoints to the SIE. It understands the USB protocol for incoming and outgoing packets, so it knows when to grab data and how to correctly respond to incoming packets. A large portion of the DEV is devoted to the setup, configuration, and control features of the USB. As shown in Figure 9-4: Device Interface the DEV is divided into three blocks: Device Controller, Device ROM, and Start of Frame. The three blocks are described in the following sections.

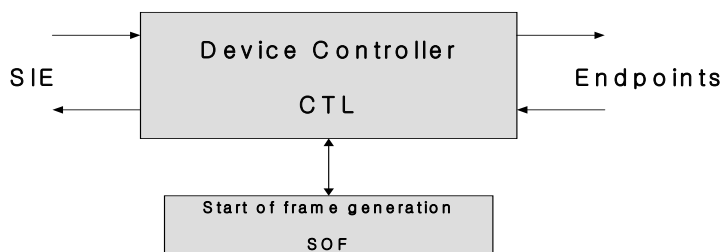


Figure 9-3 USB Device Interface Device Controller

Device Controller

The Device Controller contains a state machine that understands the USB protocol. The (SIE) provides the Device Controller with the type of packet, address value, endpoint value, and data stream for each incoming packet. The Device Controller then checks to see if the packet is targeted to the device by comparing the address/endpoint values with internal registers that were loaded with address and endpoint values during the USB enumeration process. Assuming the address/endpoint is a match, the Device Controller then interprets the packet. Data is passed on to the endpoint for all packets except SETUP packets, which are handled specially. Data toggle bits (DATA0 and DATA1 as defined by the USB spec) are maintained by the Device Controller. For IN data packets (device to host) the Device Controller sends either the maximum number of bytes in a packet or the number of bytes available from the endpoint. All packets are acknowledged as per the spec. For SETUP packets, the incoming data is extracted into the relevant internal fields, and then the appropriate action is carried out. Table 9-7: Supported Setup Requests lists the types of setup operations that are supported.

| Setup Request | Value | Supported | Setup Request | Value | Supported |
|----------------|-------|-----------------------------|-------------------|-------|---------------|
| Get Status | 0 | Device, Interface, Endpoint | Get Configuration | 8 | Device |
| Clear Feature | 1 | Endpoints Only | Set Configuration | 9 | Device |
| Set Feature | 3 | Not supported | Get Interface | 10 | Not supported |
| Set Address | 5 | Device | Set Interface | 11 | Not supported |
| Get Descriptor | 6 | Device | Synch Frame | 12 | Not supported |
| Set Descriptor | 7 | Not supported | | | |

Table 9-19 USB Supported Setup Requests

Start of Frame

The Start of Frame logic generates a pulse whenever either the incoming Start of Frame (SOF) packet arrives or approximately 1 ms after it the last one arrived. This allows an isochronous endpoint to stay in sync even if the SOF packet has been garbled.

9.5.3 Endpoint FIFOs (Rx, Tx)

Each endpoint FIFO has the specific number of FIFO depth according to data transfer rate. In case of maximum packet size for bulk transfer is 32 bytes that is supported in USB. Each FIFO generates data ready signals (means FIFO not full or FIFO not empty) to AMBA IF. It contains the control logic for transferring 4 bytes at a read/write strobe generated by AMBA to obtain better efficiency of AMBA bus.

9.5.4 External Signals

| Pin Name | Type | Description |
|----------|------|-------------------------------|
| USBP | I/O | USB transceiver signal for P+ |
| USBN | I/O | USB transceiver signal for N+ |

9.5.5 Registers

| Address | Name | Width | Default | Description |
|-------------|----------|-------|------------|------------------------------------|
| 0x8005.1000 | GCTRL | 4 | 0x0 | USB Global Configuration Register |
| 0x8005.1004 | EPCTRL | 21 | 0x0 | Endpoint Control Register |
| 0x8005.1008 | INTMASK | 10 | 0x3ff | Interrupt Mask Register |
| 0x8005.100C | INTSTAT | 20 | 0x0 | Interrupt Status Register |
| 0x8005.1010 | PWR | 4 | 0x0 | Power Control Register |
| 0x8005.1018 | DEVID | 32 | 0x721005b4 | Device ID Register |
| 0x8005.101C | DEVCLASS | 32 | 0xfffff | Device Class Register |
| 0x8005.1020 | INTCLASS | 32 | 0xfffff | Interface Class Register |
| 0x8005.1024 | SETUP0 | 32 | - | SETUP Device Request Lower Address |
| 0x8005.1028 | SETUP1 | 32 | - | SETUP Device Request Upper Address |
| 0x8005.102C | ENDP0RD | 32 | - | ENDPOINT0 Read Address |
| 0x8005.1030 | ENDP0WT | 32 | - | ENDPOINT0 WRITE Address |
| 0x8005.1034 | ENDP1RD | 32 | - | ENDPOINT1 READ Address |
| 0x8005.1038 | ENDP2WT | 32 | - | ENDPOINT2 WRITE Address |

Table 9-20 USB Slave interface Register Summary

9.5.5.1 GCTRL

| | | 0x8005.1000 | | | |
|----------|------|---|-------|--------|--------|
| 31 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | TRANSel | WBack | Resume | DMADis |
| Bits | Type | Function | | | |
| 3 | R/W | USB Transceiver power-down mode selection. When this bit is high, SUSPEND signal of internal USB transceiver is forced to go high immediately. This is for power-down scheme of that transceiver when USB function is NOT used. It is recommended that this value keeps zero while USB normal operation | | | |
| 2 | R/W | HMS30C7210 does not supports Write-Back clear mode for Interrupt Status Register. This bit must be set to '0'. | | | |
| 1 | R/W | This Enables Remote Resume Capabilities. When This Bit Set, USB Drives remote resume signaling. Should be cleared to stop resume | | | |
| 0 | R | DMA Disable bit. HMS30C7210 does not support DMA, so value of this bit (logic 1) is not changeable | | | |

9.5.5.2 EPCTRL

| 0x8005.1004 | | | | | | | | | | |
|-------------|----|------|------|------|-------|-------|------|------|------|----|
| 31 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 |
| Reserved | | CLR2 | CLR1 | CLR0 | E2TXB | E2SND | E2NK | E2ST | E2En | |
| 11 | 10 | 9 | 8 | 7 | 4 | 3 | 2 | 1 | 0 | |

E1RCV E1NK E1ST E1En E0TXB E0NK E0ST E0TR E0En

| Bits | Type | Function |
|-------|------|--|
| 20 | R/W | Clear Endpoint2 FIFO Pointer(Auto cleared by Hardware). |
| 19 | R/W | Clear Endpoint1 FIFO Pointer(Auto cleared by Hardware). |
| 18 | R/W | Clear Endpoint0 FIFO Pointer(Auto cleared by Hardware). |
| 17~16 | R/W | USB Can Transmit NON Maximum sized Packet. This Field contains the residue byte which should be transmitted. |
| 15 | R/W | This Bit enables NON Maximum sized Packet transfer. After NON maximum sized packet transfer, this bit is auto cleared and return to Maximum Packet size transfer mode. |
| 14 | R/W | When This Bit is Set, and Endpoint2 is not enabled, USB should send NAK Handshake |
| 13 | R/W | When This Bit is Set, and Endpoint2 is not enabled, USB should send STALL Handshake |
| 12 | R/W | Enable Endpoint2 as IN Endpoint |
| 11 | R/W | This bit must be zero. So only maximum packet size RX transfer mode is supported. This means RX (HOST OUT) data packet size is fixed to 32 bytes only. |
| 10 | R/W | When This Bit is Set, and Endpoint1 is not enabled, USB should send NAK Handshake |
| 9 | R/W | When This Bit is Set, and Endpoint1 is not enabled, USB should send STALL Handshake |
| 8 | R/W | Enable Endpoint1 as OUT Endpoint |
| 7~4 | R/W | This Bit Stores the Byte Count which should be transmitted to HOST when IN token is received (Exception :: When This bit is 0, 8 Byte are transferred) |
| 3 | R/W | When This Bit is Set, and Endpoint0 is not enabled, USB should send NAK Handshake |
| 2 | R/W | When This Bit is Set, and Endpoint0 is not enabled, USB should send STALL Handshake |
| 1 | R/W | When this Bit1, Endpoint0 is configured to IN endpoint. (others OUT endpoint) |
| 0 | R/W | Enable Endpoint0 |

9.5.5.3 INTMASK

0x8005.1008

| | | | | | | | | | | | |
|----------|-------|-----|-------|------|------|------|------|------|------|-----|---|
| 31 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | E0STL | SUS | RESET | E2EM | E1OV | E1FU | E0EM | E0OV | E0FU | SET | |

| Bits | Type | Function |
|------|------|--|
| 9 | R/W | Mask Endpoint0 Stall Interrupt |
| 8 | R/W | Mask SUSPEND Interrupt |
| 7 | R/W | Mask USB Cable RESET Interrupt |
| 6 | R/W | Mask Endpoint2 Empty Interrupt |
| 5 | R/W | Mask Endpoint1 Overrun Interrupt (May not be used) |
| 4 | R/W | Mask Endpoint1 Full Interrupt |
| 3 | R/W | Mask Endpoint0 Empty Interrupt |
| 2 | R/W | Mask Endpoint0 Overrun Interrupt (May not be used) |
| 1 | R/W | Mask Endpoint0 Full Interrupt |
| 0 | R/W | Mask Endpoint0 Setup Token Received Interrupt |

9.5.5.4 INTSTAT

0x8005.100C

| | | | | | | | | | |
|----------|-----|-----------|------|-----------|------|------|------|------|-----|
| 31 | 20 | 19 | 14 | 13 | 0 | | | | |
| Reserved | | EP1RXBYTE | | EP0RXBYTE | | | | | |
| 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| E0STL | SUS | RESET | E2EM | E1OV | E1FU | E0EM | E0OV | E0FU | SET |

| Bits | Type | Function |
|-------|------|--|
| 19~14 | R/W | Currently Remained Byte In Endpoint1 Receive FIFO which should be read by HOST |
| 13~10 | R/W | Currently Remained Byte in Endpoint0 Receive FIFO which should be read by HOST |
| 9 | R/W | Endpoint0 Stall Interrupt |
| 8 | R/W | SUSPEND Interrupt |
| 7 | R/W | USB Cable RESET Interrupt |
| 6 | R/W | Endpoint2 Empty Interrupt |

| | | |
|---|-----|---|
| 5 | R/W | Endpoint1 Overrun Interrupt (May not be used) |
| 4 | R/W | Endpoint1 Full Interrupt |
| 3 | R/W | Endpoint0 Empty Interrupt |
| 2 | R/W | Endpoint0 Overrun Interrupt (May not be used) |
| 1 | R/W | Endpoint0 Full Interrupt |
| 0 | R/W | Endpoint0 Setup Token Received Interrupt |

9.5.5.5 PWR

| | | | | 0x8005.1010 | |
|-----------|----------|--|----------|-------------|----------|
| 31 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | EnBCLK | SWUPDATE | PwrMD | |
| Bits | Type | Function | | | |
| 3 | R/W | Enable BCLK to USB FIFO Block. . | | | |
| 2 | R/W | USB Core Power Mode Update Mode, When This Bit 1, Only software can update USB Core Power Mode. But this bit 0, USB core automatically update its power status according to cable state | | | |
| 1 ~ 0 | R/W | USB Power Mode | | | |
| 0 | | 00 : Full Power Down -> Usb core can't detect any cable activity 01 : Power Power Down -> Usb can detect any cable activity but core doesn't operate normally 10 : Full Power Operation Mode | | | |

9.5.5.6 DEVID

| | | | | 0x8005.1018 | |
|------|------|---|--|-------------|--|
| Bits | Type | Function | | | |
| 31:0 | R/W | USB Core Can Change Device ID Field by writing Appropriate Device ID Value to This Register | | | |

9.5.5.7 DEVCLASS

| | | | | 0x8005.101C | |
|------|------|--|--|-------------|--|
| Bits | Type | Function | | | |
| 23:0 | R/W | USB Core Can Change Device Class Field by writing Appropriate Device ID Value to This Register | | | |

9.5.5.8 INTCLASS

| | | | | 0x8005.1020 | |
|------|------|---|--|-------------|--|
| Bits | Type | Function | | | |
| 23:0 | R/W | USB Core Can Change Interface Class Field by writing Appropriate Device ID Value to This Register | | | |

- While USB device configuration process, HOST requests Descriptors.
This USB block has a hard-wired descriptor ROM, but there are 3 fields (whole 10 bytes size) user adjustable.

[DEVICE DESCRIPTOR]

- see USB spec. 1.1 (9.6 Standard USB Descriptor Definitions) for more detail

| OFFSET (BYTE) | INITIAL VALUE | DESCRIPTION | ADJUSTABLE |
|---------------|---------------|--------------------------|------------|
| h00 | h12 | length | |
| h01 | h01 | DEVICE | |
| h02 | h00 | spec version 1.00 | |
| h03 | h01 | spec version | |
| h04 | hFF | device class | YES |
| h05 | hFF | device sub-class | YES |
| h06 | hFF | vendor specific protocol | YES |
| h07 | h08 | max packet size | |

| | | | |
|-----|-----|-------------------------------|-----|
| h08 | hB4 | vendor id | YES |
| h09 | h05 | vendor id (05b4) for HME | YES |
| h0a | h02 | product id | YES |
| h0b | h72 | product id (7210) for HME7210 | YES |
| h0c | h01 | device release # | |
| h0d | h00 | device release # | |
| h0e | h00 | manufacturer index string | |
| h0f | h00 | product index string | |
| h10 | h00 | serial number index string | |
| h11 | h01 | number of configurations | |

* DEVID register has 32-bit width and it covers vendor id to product id (offset from h08 to h0b): DEVID [31:24] – h0b, DEVID [23:16] – h0a, DEVID [15:8] – h09, DEVID [7:0] – h08

* DEVCLASS register has 24-bit width and it covers device class to vendor specific protocol (offset from h04 to h06): DEVCLASS [23:16] – h06, DEVCLASS [15:8] – h05, DEVCLASS [7:0] – h04

[CONFIGURATION DESCRIPTOR]

| OFFSET (BYTE) | INITIAL VALUE | DESCRIPTION | ADJUSTABLE |
|---------------|---------------|---|------------|
| h00 | h09 | Length of this descriptor | |
| h01 | h02 | CONFIGURATION (2) | |
| h02 | h20 | Total length includes endpoint descriptors | |
| h03 | h00 | Total length high byte | |
| h04 | h01 | Number of interfaces | |
| h05 | h01 | Configuration value for this one | |
| h06 | h00 | Configuration - string | |
| h07 | h80 | Attributes - bus powered, no wakeup | |
| h08 | h32 | Max power - 100 ma is 50 (32 hex) | |
| h09 | h09 | Length of the interface descriptor | |
| h0a | h04 | INTERFACE (4) | |
| h0b | h00 | Zero based index of this interface | |
| h0c | h00 | Alternate setting value (?) | |
| h0d | h02 | Number of endpoints (not counting 0) | |
| h0e | hFF | Interface class, ff is vendor specific | YES |
| h0f | hFF | Interface sub-class | YES |
| h10 | hFF | Interface protocol | YES |
| h11 | h00 | Index to string descriptor for this interface | |
| h12 | h07 | Length of this endpoint descriptor | |
| h13 | h05 | ENDPOINT (5) | |
| h14 | h01 | Endpoint direction (00 is out) and address | |
| h15 | h02 | Transfer type – h02 = BULK | |
| h16 | h20 | Max packet size - low : 32 byte | |
| h17 | h00 | Max packet size - high | |
| h18 | h00 | Polling interval in milliseconds (1 for iso) | |
| h19 | h07 | Length of this endpoint descriptor | |
| h1a | h05 | ENDPOINT (5) | |
| h1b | h82 | Endpoint direction (80 is in) and address | |
| h1c | h02 | Transfer type – h02 = BULK | |
| h1d | h20 | Max packet size - low : 32 byte | |
| h1e | h00 | Max packet size - high | |
| h1f | h00 | Polling interval in milliseconds (1 for iso) | |

* see USB spec. 1.1 (9.6 Standard USB Descriptor Definitions) for more detail

* The descriptor has 4 parts : Configuration, Interface, Endpoint1, Endpoint2 (doubled lines)

[STRING DESCRIPTOR]

| OFFSET | INITIAL VALUE | DESCRIPTION | ADJUSTABLE |
|--------|---------------|-----------------|------------|
| h0 | h02 | size in bytes | |
| h1 | h03 | STRING type (3) | |

* This index zero string descriptor means a kind of look up table. As there is no other string descriptor and as there is no further information in this descriptor, USB block does not support strings. (All string index fields are filled with zero)

9.5.5.9 SETUP0 / SETUP1

0x8005.1024 / 0x8005.1028

| Bits | Type | Function |
|------|------|--|
| 31:0 | R/W | USB Core can accept vendor specific protocol command using Endpoint0. This Register contains previously received Setup Device Request Value (64-bit Wide, half in each Register) |

- Below is Request format from HOST when configuration.

[Standard Device Request Format]

| bmRequestType | bRequest | wValue | wIndex | wLength |
|---------------|----------|--------|--------|---------|
| Byte 0 | Byte 1 | Byte 2 | Byte 3 | Byte 4 |
| | | Byte 5 | Byte 6 | Byte 7 |

When HOST sends request to USB device, this USB block handles a few requests by SIE (Serial Interface Engine).

This is the condition of requests which this USB SIE can handle.

- Request Type must be Standard (b00): see USB spec. 9.3 Table 9-2 'Format of Setup Data' for more detail. Offset 0 (bmRequestType field) D[6:5] (Type) ; 00 – Standard, 01 Class, 10 – Vendor, 11 – reserved.
- Request must be one of these: GET_DESCRIPTOR, SET_ADDRESS, SET_INTERFACE, SET_CONFIGURATION, GET_INTERFACE, GET_CONFIGURATION and GET_STATUS.

So for requests other than above, HMS30C7210 USB sets 9.5.5.4 INTSTAT [0] and it means HOST sent Setup Request that USB SIE cannot handle by itself and these 9.5.5.9 SETUP0 and 9.5.5.10 SETUP1 register hold Device Request Data (8 bytes : 64 bit described above). This function is to handle standard requests that SIE cannot handle and to handle vendor specific requests.

* Note: 9.5.5.4 INTSTAT [0] bit will not go 'high' in case of Setup request if SIE can handle that request by itself.

9.5.5.10 ENDP0RD

0x8005.102C

| Bits | Type | Function |
|------|------|---------------------------|
| 31:0 | R/W | Each Endpoint 0 FIFO Read |

9.5.5.11 ENDP0WT

0x8005.1030

| Bits | Type | Function |
|------|------|----------------------------|
| 31:0 | R/W | Each Endpoint 0 FIFO Write |

9.5.5.12 ENDP1RD

0x8005.1034

| Bits | Type | Function |
|------|------|---------------------------|
| 31:0 | R/W | Each Endpoint 1 FIFO Read |

9.5.5.13 ENDP2WT

0x8005.1038

| Bits | Type | Function |
|------|------|----------------------------|
| 31:0 | R/W | Each Endpoint 2 FIFO Write |

10 SLOW AMBA PERIPHERALS

10.1 ADC Interface Controller

HMS30C7210 has internal ADC and ADC interface logic for analog applications of touch panel interface, two 8-bit battery check, and one 8-bit sound sampling. If user doesn't need these applications or want to use for other functions, there's a direct ADC control register available.

FEATURES

- 5-channel 10-bit ADC embedded
- 4-sample data per one sampling point of touch panel (use 2 channels, X and Y, 10-bit)
- Main and backup battery check function (use 2 channels, 8-bit resolution)
- Eight 32-byte sound data buffer (8-word buffer, 8-bit sound data)
- Manual and Auto ADC power down mode

10.1.1 External Signals

| Pin Name | Type | Description |
|----------|--------------|---------------------------------|
| ADIN[0] | Analog input | Touch Panel X-axis signal input |
| ADIN[1] | Analog input | Touch Panel Y-axis signal input |
| ADIN[2] | Analog input | Main Battery value input |
| ADIN[3] | Analog input | Backup Battery value input |
| ADIN[4] | Analog input | Sound input |

10.1.2 Registers

| Address | Name | Width | Default | Description |
|-------------|------------|-------|---------|------------------------------------|
| 0x8002.9000 | ADCCR | | 0x80 | ADC Control Register |
| 0x8002.9004 | ADCTPCR | | 0x0 | Touch panel control register |
| 0x8002.9008 | ADCBACR | | 0x0 | Battery check Control Register |
| 0x8002.900C | ADCSDCR | | 0x0 | Sound Data Control Register |
| 0x8002.9010 | ADCISR | | 0x0 | ADC Interrupt Status Register |
| 0x8002.901C | ADCTDCSR | | 0x0X | Tip Down Control/Status Register |
| 0x8002.9020 | ADCDIRCR | | | ADC Direct Control Register |
| 0x8002.9024 | ADCDIRDATA | | | ADC Direct Data read register |
| 0x8002.9030 | ADCTPXDR0 | | | Touch Panel X Data register 0 |
| 0x8002.9034 | ADCTPXDR1 | | | Touch Panel X Data register 1 |
| 0x8002.9038 | ADCTPYDR0 | | | Touch Panel Y Data register 0 |
| 0x8002.903C | ADCTPYDR1 | | | Touch Panel Y Data register 1 |
| 0x8002.9040 | ADCTPXDR2 | | | Touch Panel X Data register 2 |
| 0x8002.9044 | ADCTPXDR3 | | | Touch Panel X Data register 3 |
| 0x8002.9048 | ADCTPYDR2 | | | Touch Panel Y Data register 2 |
| 0x8002.904C | ADCTPYDR3 | | | Touch Panel Y Data register 3 |
| 0x8002.9050 | ADCMBDATA | | | Main Battery check Data Register |
| 0x8002.9054 | ADCBBDATA | | | Backup Battery check Data Register |
| 0x8002.9060 | ADCSDATA0 | | | Sound Data Register |
| 0x8002.9064 | ADCSDATA1 | | | Sound Data Register |
| 0x8002.9068 | ADCSDATA2 | | | Sound Data Register |
| 0x8002.906C | ADCSDATA3 | | | Sound Data Register |
| 0x8002.9070 | ADCSDATA4 | | | Sound Data Register |
| 0x8002.9074 | ADCSDATA5 | | | Sound Data Register |
| 0x8002.9078 | ADCSDATA6 | | | Sound Data Register |
| 0x8002.907C | ADCSDATA7 | | | Sound Data Register |

Table 10-1 ADC Controller Register Summary

10.1.2.1 ADC Control Register (ADCCR)

User can set ADCPD to save power consumption by ADC. But ADC needs 10-40 ms to self calibrate for normal operation. DIRECTC bit can be used for direct accessing from CPU to ADC without interface function logic. All direct control signals are describe in ADCDIRCR register field. Basically ADC core converts Analog data to Digital data continuously in every 16 ADC operation-clocks. ADC operation clock is “aclk” (3.6864MHz) called as “PCLK” in SLOW APB

WAIT bit field select conversion time of ADC because in certain case interface logic can read wrong or unstable value from ADC. SOP bit can be used for one-shot operation to save power. When this bit is set and all ADC functions are disabled then interface logic strobe “power down” signal to ADC core. LONGCAL signal selects self-calibration time. Initially this bit set as “0” it means short calibration time (about 10 ms). But if first a couple of data were wrong value, user should select long calibration time (about 40 ms).

0x8002.9000

| | | | | | | | |
|-------|---------|--|--|------|---|-----|---------|
| 7 | 6 | | | 3 | 2 | 1 | 0 |
| ADCPD | DIRECTC | | | WAIT | | SOP | LONGCAL |

| Bits | Type | Function |
|------|------|---|
| 7 | R/W | ADC power down bit. Write “1” to go ADC power save mode. This bit blocks the clock to ADC, so ADC consumes no power when this bit is set. But after release this bit, ADC need 10 ~ 40 ms calibration time to normal operation. |
| 6 | R/W | If this bit was set, CPU access directly ADC through DIRCR and directly read ADC result value through DIRDATA register. |
| 5:4 | - | Reserved |
| 3:2 | R/W | Select ADC conversion wait time. It is for capture timing of the data from ADC to internal register. 00: no wait (read after 16 cycles, default wait time) 01: 2 clock wait (read after 18 cycles) 10: 4 clock wait (read after 20 cycles) |
| 1 | R/W | Self Operate Power down bit. When this bit is set, AIOSTOP bit will strobe high when no ADC functions are enabled. |
| 0 | R/W | Long calibration time. The default ADC calibration time is 10 ms but when needed ADC can be calibrated during 40ms with this bit. Short calibration time need 96 cycles of 8 kHz OCLK or 128 cycles of 11 kHz OCLK and the long time need 384 cycles of 8 kHz or 512 cycles of 11 kHz OCLK. OCLK is determined from SRATE bit of ADCSDCR. |

| ADCCR. LONGCAL bit | ADCSCR. SRATE bit | Calibration Time (the number of OCLK cycles) |
|-----------------------|----------------------|---|
| 0 | 0 | 96 |
| 0 | 1 | 128 |
| 1 | 0 | 383 |
| 1 | 1 | 511 |

10.1.2.2 ADC Touch Panel Control Register (ADCTPCR)

This register control functions related with touch panel interface. HMS30C7210 supports only external drive for touch panel, so prudent setting of this register is needed.

0x8002.9004

| | | | | | | | |
|------|---------|---------|--------|---------|-------|-------|---|
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TPEN | TINTMSK | SWBYPSS | SWINVT | INTTDEN | SSHOT | TRATE | |

| Bits | Type | Function |
|------|------|--|
| 7 | R/W | Touch panel read enable bit. Write “1” to enable touch panel function. |
| 6 | R/W | Touch panel read interrupt mask bit. Write “1” to enable touch panel interrupt. |
| 5 | R/W | Touch panel drive signal bypass bit for external drive circuit. You must set this bit to bypass switching signals to external pins such as SW_XP, SW_XN, SW_YP and SW_YN. |

| | | |
|-----|-----|--|
| 4 | R/W | Touch panel drive signal inversion bit. for flexibility |
| 3 | R/W | Internal tip-down detection logic enable bit. You must write "0" to disable this function. |
| 2 | R/W | Single touch panel read operation. Normally, touch panel data read twice. But this bit is set, touch panel data read once for a point and save power to read touch panel. |
| 1:0 | R/W | Select touch panel data sampling rate. It depends on basic operation clock of ADC interface(sound sampling rate). 11: 400 or 550 samples / sec 10: 200 or 275 samples / sec 01: 100 or 138 samples / sec 00: 50 or 69 samples / sec |

10.1.2.3 ADC Battery check Control Register (ADCBACR)

This registers controls battery check operation.

| | | | | | | | 0x8002.9008 |
|------|---------|--|--|------|---------|--|-------------|
| 7 | 6 | | | 3 | 2 | | |
| MBEN | MINTMSK | | | BBEN | BINTMSK | | |
| Bits | Type | Function | | | | | |
| 7 | R/W | Main battery check enable Write "1" to enable Four 8-bit battery check data recorded in ADCMBDATA register | | | | | |
| 6 | R/W | Main battery check interrupt mask bit Write "1" to enable | | | | | |
| 5:4 | - | Reserved | | | | | |
| 3 | R/W | Backup battery check enable Write "1" to enable Four 8-bit battery check data recorded in ADCBBDATA register | | | | | |
| 2 | R/W | Backup battery check interrupt mask bit Write "1" to enable | | | | | |
| 1:0 | - | Reserved | | | | | |

10.1.2.4 ADC Sound Control Register (ADCSDCR)

This registers controls sound sampling function. SRATE bit control base clock of ADC interface logic.

| | | | | | | | 0x8002.900C |
|-------|---------|--|--|--|--|-------|-------------|
| 7 | 6 | | | | | 0 | |
| SNDEN | SINTMSK | | | | | SRATE | |
| Bits | Type | Function | | | | | |
| 7 | R/W | Sound data capture enable bit Write "1" to enable | | | | | |
| 6 | R/W | Sound data interrupt mask bit Write "1" to enable | | | | | |
| 5:1 | - | Reserved | | | | | |
| 0 | R/W | Sound data sampling rate selection bit. This bit affects to all sampling rates of touch panel and battery operations. 0: 8 kHz sound sampling 1: 11.025 kHz sound sampling 8/11KHz is derived from ack (3.6864MHz) called as "PCLK" in SLOW APB. | | | | | |

10.1.2.5 ADC Interrupt Status Register (ADCISR)

Read only valid but write "1" to clear all interrupt value

| | | | | | | | |
|-------|-------|-------|-------|--|--|-------|-------------|
| | | | | | | | 0x8002.9010 |
| 7 | 6 | 5 | 4 | | | 1 | 0 |
| INTTP | INTMB | INTBB | INTSD | | | INTTD | INTTU |

| Bits | Type | Function |
|------|------|---|
| 7 | R/W | Touch panel data interrupt. Write "1" here to clear this interrupt. |
| 6 | R/W | Main battery checks interrupt. Write "1" here to clear this interrupt. |
| 5 | R/W | Backup battery check interrupt. Write "1" to clear this interrupt. |
| 4 | R/W | Sound data interrupt. It will be generated when all the 8 sound registers are full. Write "1" here to clear this interrupt. |
| 3:2 | - | Reserved |
| 1 | R/W | Tip Down interrupt. Write "1" here to clear this interrupt. |
| 0 | R/W | Tip Up interrupt. Write "1" here to clear this interrupt. |

10.1.2.6 ADC Tip Down Control Status Register (ADCTDCSR)

| 7 | 6 | 5 | 4 | 3 | 1 | 0 |
|------|-------|------|-------|-------|------|------|
| TDEN | TDMSK | TUEN | TUMSK | TPSEL | TP_X | TP_Y |

0x8002.901C

| Bits | Type | Function |
|------|------|---|
| 7 | R/W | Touch panel tip-down detection logic enable Write "1" to enable this function |
| 6 | R/W | Touch panel tip-down interrupt mask bit Write "1" to enable interrupt |
| 5 | R/W | Touch panel tip-up detection enable. When this bit is set, once in every 20 OCLK cycles, monitor touch panel status periodically. |
| 4 | R/W | Touch panel tip-up interrupt mask bit. |
| 3 | R/W | Select Tip Down/Up monitoring channel (0:X, 1:Y) |
| 2 | - | Reserved |
| 1 | R/W | X axis Tip status monitor bit (read only bit) |
| 0 | R/W | Y axis Tip status monitor bit (read only bit) |

10.1.2.7 ADC Direct Control Register (ADCDIRCR)

ADC I/F has the Direct Data Read Function. When DIRECTC bit in ADCCR register is set high, CPU can access directly A/D Converter through this register and can read conversion data of A/D Converter through DIRDATA register.

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---------|---|---|-----|---|---|---|---|
| AIOSTOP | | | ACH | | | | |

0x8002.9020

| Bits | Type | Function |
|------|------|---|
| 7 | R/W | AIOSTOP bit value to access ADC directly |
| 6:5 | - | Reserved |
| 4:0 | R/W | ADC channel selection bits to control ADC directly 00001: select channel 0 (touch panel X) 00010: select channel 1 (touch panel Y) 00100: select channel 2 (Main battery) 01000: select channel 3 (Backup battery) 10000: select channel 4 (Sound input) |

10.1.2.8 ADC Direct Data Read Register (ADCDIRDATA)

Register can be used to read data from ADC.

| 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---------|---|---|---|---|---|---|---|---|---|
| AD Data | | | | | | | | | |

0x8002.9024

| Bits | Type | Function |
|------|------|---------------------------|
| 9:0 | R | 10-bit AD conversion data |

10.1.2.9 ADC 1ST Touch Panel Data register

0x8002.9030 – 0x8002.903C

| | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | | | | | | | | | | | | | | | |

XDATA1: ADCTPXDR0, XDATA3: ADCTPXDR1
YDATA1: ADCTPYDR0, YDATA3: ADCTPYDR1

XDATA0: ADCTPXDR0, XDATA2: ADCTPXDR1
YDATA0: ADCTPYDR0, YDATA2: ADCTPYDR1

ADCTPXDR0: 0x80029030

| Bits | Type | Function |
|-------|------|---|
| 31:26 | - | Reserved |
| 25:16 | R | Touch panel X data 10-bit, 2/4 of the first sample cycle (XDATA1) |
| 15:10 | - | Reserved |
| 9:0 | R | Touch panel X data 10-bit, 1/4 of the first sample cycle (XDATA0) |

ADCTPXDR1: 0x80029034

| Bits | Type | Function |
|-------|------|---|
| 31:26 | - | Reserved |
| 25:16 | R | Touch panel X data 10-bit, 4/4 of the first sample cycle (XDATA3) |
| 15:10 | - | Reserved |
| 9:0 | R | Touch panel X data 10-bit, 3/4 of the first sample cycle (XDATA2) |

ADCTPYDR0: 0x80029038

| Bits | Type | Function |
|-------|------|---|
| 31:26 | - | Reserved |
| 25:16 | R | Touch panel Y data 10-bit, 2/4 of the first sample cycle (YDATA1) |
| 15:10 | - | Reserved |
| 9:0 | R | Touch panel Y data 10-bit, 1/4 of the first sample cycle (YDATA0) |

ADCTPYDR1: 0x8002903C

| Bits | Type | Function |
|-------|------|---|
| 31:26 | - | Reserved |
| 25:16 | R | Touch panel Y data 10-bit, 4/4 of the first sample cycle (YDATA3) |
| 15:10 | - | Reserved |
| 9:0 | R | Touch panel Y data 10-bit, 3/4 of the first sample cycle (YDATA2) |

10.1.2.10 ADC 2ND Touch Panel Data Register

0x8002.9040 – 0x8002.904C

| | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | | | | | | | | | | | | | | | |

XDATA5: ADCTPXDR2, XDATA7: ADCTPXDR3
YDATA5: ADCTPYDR2, YDATA7: ADCTPYDR3

XDATA5: ADCTPXDR2, XDATA6: ADCTPXDR3
YDATA5: ADCTPYDR2, YDATA6: ADCTPYDR3

ADCTPXDR2: 0x80029040

| Bits | Type | Function |
|-------|------|--|
| 31:26 | - | Reserved |
| 25:16 | R | Touch panel X data 10-bit, 2/4 of the second sample cycle (XDATA5) |
| 15:10 | - | Reserved |
| 9:0 | R | Touch panel X data 10-bit, 1/4 of the second sample cycle (XDATA4) |

ADCTPXDR3: 0x80029044

| Bits | Type | Function |
|-------|------|--|
| 31:26 | - | Reserved |
| 25:16 | R | Touch panel X data 10-bit, 4/4 of the second sample cycle (XDATA7) |
| 15:10 | - | Reserved |
| 9:0 | R | Touch panel X data 10-bit, 3/4 of the second sample cycle (XDATA6) |

ADCTPYDR2: 0x80029048

| Bits | Type | Function |
|-------|------|--|
| 31:26 | - | Reserved |
| 25:16 | R | Touch panel Y data 10-bit, 2/4 of the second sample cycle (YDATA5) |
| 15:10 | - | Reserved |
| 9:0 | R | Touch panel Y data 10-bit, 1/4 of the second sample cycle (YDATA4) |

ADCTPYDR3: 0x8002904C

| Bits | Type | Function |
|-------|------|--|
| 31:26 | - | Reserved |
| 25:16 | R | Touch panel Y data 10-bit, 4/4 of the second sample cycle (YDATA7) |
| 15:10 | - | Reserved |
| 9:0 | R | Touch panel Y data 10-bit, 3/4 of the second sample cycle (YDATA6) |

10.1.2.11 ADC Main Battery Data Register (ADCMBDATA)

| 0x8002.9050 | | | | | | | | | | | | | | | |
|-------------|------|--------------------------------|----|----|----|----|----|---------|----|----|----|----|----|----|----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| MBDATA3 | | | | | | | | MBDATA2 | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MBDATA1 | | | | | | | | MBDATA0 | | | | | | | |
| Bits | Type | Function | | | | | | | | | | | | | |
| 31:24 | R/W | Forth main battery check data | | | | | | | | | | | | | |
| 23:16 | R/W | Third main battery check data | | | | | | | | | | | | | |
| 15:8 | R/W | Second main battery check data | | | | | | | | | | | | | |
| 7:0 | R/W | First main battery check data | | | | | | | | | | | | | |

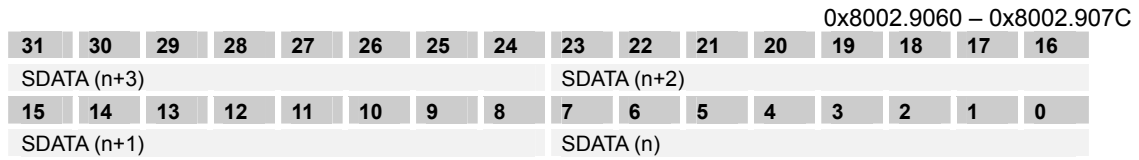
10.1.2.12 ADC Backup Battery Data Register (ADCBBDATA)

| 0x8002.9054 | | | | | | | | | | | | | | | |
|-------------|------|----------------------------------|----|----|----|----|----|---------|----|----|----|----|----|----|----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| BBDATA3 | | | | | | | | BBDATA2 | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| BBDATA1 | | | | | | | | BBDATA0 | | | | | | | |
| Bits | Type | Function | | | | | | | | | | | | | |
| 31:24 | R/W | Forth backup battery check data | | | | | | | | | | | | | |
| 23:16 | R/W | Third backup battery check data | | | | | | | | | | | | | |
| 15:8 | R/W | Second backup battery check data | | | | | | | | | | | | | |
| 7:0 | R/W | First backup battery check data | | | | | | | | | | | | | |

10.1.2.13 ADC Sound Data Register (ADCSDATA0 – ADCSDATA7)

HMS30C7202 has 8-word size sound register so it can contain 32 8-bit sound data. In ADC interface logic, there are 8-byte(2-word) temporal buffer for sound data and every 2-word write into SDATA0,1 / SDATA2,3 / SDATA4,5 / SDATA6,7 at a time (at end of every "all 8-byte temporal buffer full")

time). So, user has to read in 8 x (one sample period) second for getting valid ADCSDATA0,1(1st 2-word) after Sound interrupt.



| Bits | Type | Function |
|-------|------|--|
| 31:24 | R/W | (4n+3) TH Sound Data. (n = ADCSDATAn) |
| 23:16 | R/W | (4n+2) TH Sound Data. (n = ADCSDATAn) |
| 15:8 | R/W | (4n+1) TH Sound Data. (n = ADCSDATAn) |
| 7:0 | R/W | (4n) TH Sound Data. (n = ADCSDATAn) |

HMS30C7202

10.2 CAN Interface

The Controller Area Network (CAN) is a serial communication protocol that can be efficiently used in distributed real-time control with a very high level of security. Its domain of application ranges from high-speed networks to low cost multiplex wiring. Especially in automotive electronics, engine control units, antilock-break-systems, sensors, anti-skid-systems, etc. can be connected using a CAN with bitrates up to 1 Mbit/s. At the same time it is cost-effective to build into vehicle body electronics, e.g. lamp clusters, electric windows etc. to replace the wiring harness otherwise required.

The CAN used in HMS30C7202 performs communication according to the CAN protocol version 2.0. The register set of the CAN can be accessed directly by an ARM core via the module interface. These register are used to control/configure the CAN Core and the Message Handler and to access the Message RAM.

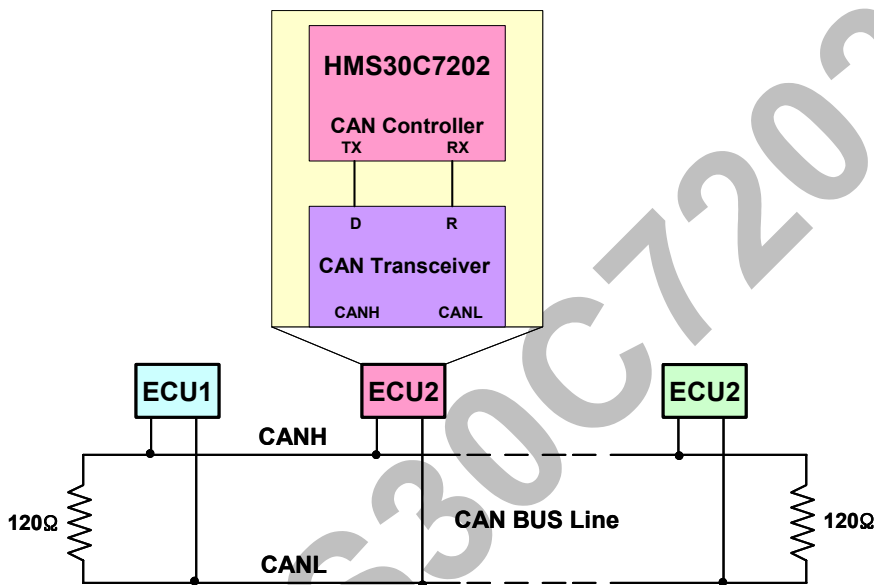


Figure 10-1 Typical CAN Network

FEATURES

- Supports CAN protocol version 2.0 part A and B
- Bit rates up to 1Mbit/s
- Disable Automatic Retransmission mode for Time Triggered CAN application
- 32 Message Objects
- Each Message Object has its own identifier mask
- Programmable FIFO mode
- Maskable interrupt
- Programmable loop-back mode for self test operation
- Two 16-bit module interface to the AMBA APB bus from ARM

10.2.1 Block Diagram

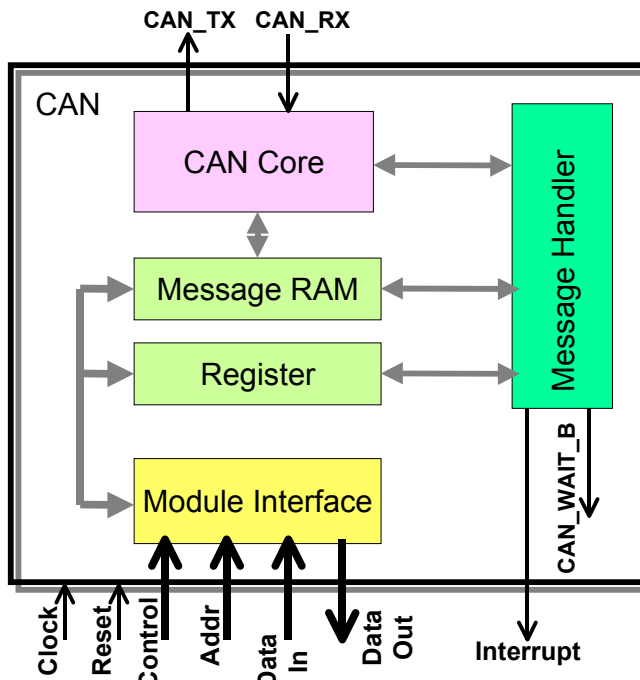


Figure 10-2 Block Diagram of the CAN

CAN CORE

CAN Protocol Controller and Rx/Tx Shift Register

Message RAM

Stores Message Objects and Identifier Masks

Registers

All registers used to control and to configure the CAN module

Message Handler

The State machine controls the data transfer between the Rx/Tx Shift Register of the CAN Core and the Message RAM as well as the generation of interrupts according to the programming of the Control and Configuration Registers

10.2.2 Register Map

The register map consists of :

- a set of Control, Configuration and Status Registers
- two CPU interface registers sets for access to the Message RAM
- 32 Message Objects located in the Message RAM

Base address of CAN0 : 0x8002.F000

Base address of CAN1 : 0x8003.0000

| Address | Name | Width | Default | Description |
|----------------|--------------|-------|---------|-----------------------------|
| CAN_BASE+0x000 | CANControl | 16h | 0x0001 | CAN Control Register |
| CAN_BASE+0x004 | CANStatus | 16 | 0x0000 | CAN Status Register |
| CAN_BASE+0x008 | CANError | 16 | 0x0000 | CAN Error Counting Register |
| CAN_BASE+0x00C | CANBitTimReg | 16 | 0x2301 | CAN Bit Timing Register |
| CAN_BASE+0x010 | CANIntReg | 16 | 0x0000 | CAN Interrupt Register |

| | | | | |
|----------------|------------|----|---------|-------------------------------------|
| CAN_BASE+0x014 | CANTestReg | 16 | 0x0000* | CAN Test Register |
| CAN_BASE+0x018 | CANBRPEExt | 16 | 0x0000 | CAN BRP Extension Register |
| CAN_BASE+0x01C | CANEnable | 16 | 0x0000 | CAN Enable Register |
| CAN_BASE+0x020 | IF1ComR | 16 | 0x0001 | Interface1 Command Request Register |
| CAN_BASE+0x024 | IF1ComM | 16 | 0x0000 | Interface1 Command Mask Register |
| CAN_BASE+0x028 | IF1Mask1 | 16 | 0xFFFF | Interface1 Mask1 Register |
| CAN_BASE+0x02C | IF1Mask2 | 16 | 0xFFFF | Interface1 Mask2 Register |
| CAN_BASE+0x030 | IF1Arb1 | 16 | 0x0000 | Interface1 Arbitration1 Register |
| CAN_BASE+0x034 | IF1Arb2 | 16 | 0x0000 | Interface1 Arbitration2 Register |
| CAN_BASE+0x038 | IF1Mcont | 16 | 0x0000 | Interface1 Message Control Register |
| CAN_BASE+0x03C | IF1DataA1 | 16 | 0x0000 | Interface1 Data A1 Register |
| CAN_BASE+0x040 | IF1DataA2 | 16 | 0x0000 | Interface1 Data A2 Register |
| CAN_BASE+0x044 | IF1DataB1 | 16 | 0x0000 | Interface1 Data B1 Register |
| CAN_BASE+0x048 | IF1DataB2 | 16 | 0x0000 | Interface1 Data B2 Register |
| CAN_BASE+0x080 | IF2ComR | 16 | 0x0001 | Interface2 Command Request Register |
| CAN_BASE+0x084 | IF2ComM | 16 | 0x0000 | Interface2 Command Mask Register |
| CAN_BASE+0x088 | IF2Mask1 | 16 | 0xFFFF | Interface2 Mask1 Register |
| CAN_BASE+0x08C | IF2Mask2 | 16 | 0xFFFF | Interface2 Mask2 Register |
| CAN_BASE+0x090 | IF2Arb1 | 16 | 0x0000 | Interface2 Arbitration1 Register |
| CAN_BASE+0x094 | IF2Arb2 | 16 | 0x0000 | Interface2 Arbitration2 Register |
| CAN_BASE+0x098 | IF2Mcont | 16 | 0x0000 | Interface2 Message Control Register |
| CAN_BASE+0x09C | IF2DataA1 | 16 | 0x0000 | Interface2 Data A1 Register |
| CAN_BASE+0x0A0 | IF2DataA2 | 16 | 0x0000 | Interface2 Data A2 Register |
| CAN_BASE+0x0A4 | IF2DataB1 | 16 | 0x0000 | Interface2 Data B1 Register |
| CAN_BASE+0x0A8 | IF2DataB2 | 16 | 0x0000 | Interface2 Data B2 Register |
| CAN_BASE+0x100 | TxRqst1 | 16 | 0x0000 | Transmission Request 1 |
| CAN_BASE+0x104 | TxRqst2 | 16 | 0x0000 | Transmission Request 2 |
| CAN_BASE+0x120 | NewDat1 | 16 | 0x0000 | New Data 1 |
| CAN_BASE+0x124 | NewDat2 | 16 | 0x0000 | New Data 2 |
| CAN_BASE+0x140 | IntPnd1 | 16 | 0x0000 | Interrupt Pending 1 |
| CAN_BASE+0x144 | IntPnd2 | 16 | 0x0000 | Interrupt Pending 2 |
| CAN_BASE+0x160 | MsgVal1 | 16 | 0x0000 | Message Validation 1 |
| CAN_BASE+0x164 | MsgVal2 | 16 | 0x0000 | Message Validation 2 |

(*: 0b0000.0000.r000.0000 : r = actual value at pin CAN_RX)

10.2.3 Registers

10.2.3.1 CAN Control Register

| | | | | | | | | |
|----------|-----|-----|----------|-----|-----|----|------|----------------|
| | | | | | | | | CAN_BASE+0x000 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | |
| Reserved | | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| Test | CCE | DAR | Reserved | EIE | SIE | IE | Init | |

| Bits | Type | Function |
|------|------|--|
| 7 | R/W | Test Mode Enable, '1': Test mode. '0': normal mode. |
| 6 | R/W | Configuration Change Enable '1': The CPU has write access to the Bit Timing Register (while Init = '1'). '0': The CPU has no write access to the Bit Timing Register. |
| 5 | R/W | Disable Automatic Retransmission '1': Automatic Retransmission disabled. '0': Automatic Retransmission of Distributed message enabled. |
| 4 | - | Reserved |
| 3 | R/W | Error Interrupt Enable '1': Enabled - a change in the BOff or Ewarn in the Status Register generates an interrupt. '0': Disabled – No Error Status Interrupt will be generated |
| 2 | R/W | Status-change Interrupt Enable '1': Enabled - successful completion of a message transfer or a detection of CAN bus error. |

| | | |
|---|-----|---|
| | | '0': Disabled – No Status Change Interrupt will be generated |
| 1 | R/W | Module Interrupt Enable '1': Enabled '0': Disabled |
| 0 | R/W | Internal Initialization Pending '1': Initialization is started. '0': Normal Operation |

10.2.3.2 CAN Status Register

| CAN_BASE+0x004 | | | | | | | |
|----------------|-------|---|------|------|-----|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Boff | Ewarn | EPass | RxOk | TxOk | LEC | | |
| Bits | Type | Function | | | | | |
| 7 | R | Bus Off Status '1': The CAN module in busoff status. '0': The CAN module is not busoff | | | | | |
| 6 | R | Error Warning Status '1': At least one of the error counters in the EML(Error Management Logic) has reached the error warning limit of 96. '0': Both error counters are below the error warning limit of 96 | | | | | |
| 5 | R | Error Passive '1': The CAN Core is in the error passive state as defined in the CAN Specification '0': The CAN Core is error active | | | | | |
| 4 | R | Received a Message Successfully '1': Since this bit was last reset(to zero) by the ARM Core, a message has been successfully received(independent of the result of acceptance filtering) '0': Since the ARM Core last reset this bit, no message has been successfully received. This bit is never reset by the CAN Core | | | | | |
| 3 | R | Transmitted a Message Successfully '1': Since this bit was last reset(to zero) by the ARM Core, a message has been successfully transmitted(error free and acknowledged by at least one other node). '0': Since the ARM Core last reset this bit, no message has been successfully transmitted. This bit is never reset by the CAN Core | | | | | |
| 2:0 | R | Last Error Code (See a CAN Specification for more information) "000" : No Error "001" : Stuff Error "010" : Form Error "011" : Acknowledgment Error "100" : Bit1 Error "101" : Bit0 Error "110" : CRC Error | | | | | |

(A read access to the Status Register clears the Status Interrupt.)

10.2.3.3 CAN Error Counting Register

| CAN_BASE+0x008 | | | | | | | |
|-----------------------|------|---|----|----|----|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Receive error passive | | Receive Error Count | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Transmit Error Count | | | | | | | |
| Bits | Type | Function | | | | | |
| 15 | R | Receive Error Passive '1': The receive error counter has reached the error passive level as defined in the CAN Specification '0': The receive error counter is below the error passive level. | | | | | |
| 14:8 | R | Receive Error Count | | | | | |

| | | |
|-----|---|--|
| | | Actual state of the Receive Error Counter Value : 0 ~ 127 |
| 7:0 | R | Transmit Error Count Actual state of the Transmit Error Counter Value : 0 ~ 255 |

10.2.3.4 CAN Bit Timing Register

| | | | | | | | | |
|----------|----|-----|----|-------|----|---|---|----------------|
| | | | | | | | | CAN_BASE+0x00C |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | |
| Reserved | | | | TSeg2 | | | | TSeg1 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| SWJ | | BRP | | | | | | |

| Bits | Type | Function |
|-------|------|---|
| 14:12 | WC | Timing Segment 2 The timing segment after the sample point, valid value for TSeg2 are [0...7]. The actual Interpretation by the hardware of this value is such that one more than the value programmed here is used. |
| 11:8 | WC | Timing Segment 1 The timing segment before the sample point, valid value for TSeg1 are [1...15]. The actual Interpretation by the hardware of this value is such that one more than the value programmed here is used. |
| 7:6 | WC | (Re)Synchronous Jump Width Valid values are 0~3. The actual Interpretation by the hardware of this value is such that one more than the value programmed here is used. |
| 5:0 | WC | Baud Rate Prescaler The value by which the oscillator frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quanta. Valid values for the Baud Rate Prescaler are 0~63. The actual Interpretation by the hardware of this value is such that one more than the value programmed here is used. |

(WC: Write access only if Configuration Change Enable)

10.2.3.5 CAN Interrupt Register

| | | | | | | | | |
|--------------|----|----|----|----|----|---|---|----------------|
| | | | | | | | | CAN_BASE+0x010 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | |
| IntId 15 – 8 | | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| IntId 7 – 0 | | | | | | | | |

| Bits | Type | Function |
|------|------|---|
| 15:0 | R | Interrupt Identifier 0x0000 : No Interrupt is pending. 0x0001 ~ 0x0020 : Number of Message Object which caused the interrupt. 0x0021 ~ 0x7FFF : Unused. 0x8000 : Status Interrupt. 0x8001 ~ 0xFFFF : Unused. |

10.2.3.6 CAN Test Register

| | | | | | | | | |
|----------|-----|-----|-----------|--------|-------|----------|---|----------------|
| | | | | | | | | CAN_BASE+0x014 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | |
| Reserved | | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| Rx | Tx1 | Tx0 | Loop Back | Silent | Basic | Reserved | | |

| Bits | Type | Function |
|------|------|--|
| 7 | R | Receive Monitors the actual value of the CAN_RX Pin. |
| 6:5 | WT | Control of CAN_TX pin "00" : Reset value, CAN_TX is controlled by the CAN Core. |

| | | |
|---|----|---|
| | | "01" : Sample Point can be monitored at CAN_TX pin. "10" : CAN_TX pin drives a dominant('0') value. "11" : CAN_TX pin drives a recessive('1') value. |
| 4 | WT | Loop Back Mode(Receiving its own transmission) '1' : Loop-back mode is enabled. '0' : Loop-back mode is disabled. |
| 3 | WT | Silent Mode(Never Send Dominant Bits) '1' : The module is in Silent Mode. '0' : Normal Operation. |
| 2 | WT | Basic Mode(Message-RAM is not available) '1' : IF1 registers are used as transmit buffers and IF2 registers are used as receive buffer. '0' : Regular mode. Message-RAM is used as transmit and receive buffer. |

(WT: Write access only if Test Mode enabled)

10.2.3.7 CAN BRP Extension Register

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | CAN_BASE+0x018 |
|----------|------|-------------------------------|----|------|----|---|---|----------------|
| Reserved | | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| Reserved | | | | BRPE | | | | |
| Bits | Type | Function | | | | | | |
| 3:0 | WC | Baud Rate Prescaler Extension | | | | | | |

10.2.3.8 CAN Enable Register

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | CAN_BASE+0x01C |
|----------|------|--|----|----|----|---|----|----------------|
| Reserved | | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| Reserved | | | | | | | En | |
| Bits | Type | Function | | | | | | |
| 0 | R/W | '1' : CAN module is enabled (Enable the clock "PCLK" and "BCLK"). '0' : Disable. | | | | | | |

10.2.3.9 Interface X Command Request Register

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | CAN_BASE+0x020 / 0x080 |
|----------|------|--|----|----|----|---|---|------------------------|
| Busy | | Reserved | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| Reserved | | Message Number | | | | | | |
| Bits | Type | Function | | | | | | |
| 15 | R/W | Bus Flag (Write access only when Busy = '0') '1' : set to one when writing to the IFx Command Request Register. '0' : reset to zero when read/write action has finished. | | | | | | |
| 5:0 | R/W | Message Number 1 to 32 A Message Object in the Message RAM is selected for data transfer. 33 to 63 Not a Valid Message Number. | | | | | | |

10.2.3.10 Interface X Command Mask Register

| | | | | | | | | |
|----------|----|----|----|----|----|---|---|------------------------|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | CAN_BASE+0x024 / 0x084 |
| Reserved | | | | | | | | |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-------|------|-----|---------|-----------|-------------------|-------|-------|
| WR/RD | Mask | Arb | Control | ClrIntPnd | TxRqst/ NewDat | DataA | DataB |

| Bits | Type | Function |
|------|------|--|
| 7 | R/W | Read/Write '1' : Write data from the selected Interface Registers to the Message Object addressed by the Command Request Register. '0' : Read data from the Message Object addressed by the Command Request Register into the selected Interface Register. |
| 6 | R/W | Access Interface X Mask Bits '1' : read/write Identifier Mask + Mdir + MXtd. '0' : Mask bits unchanged. |
| 5 | R/W | Access Interface X Arbitration '1' : read/write Identifier + Dir + Xtd + MsgVal. '0' : Arbitration bits unchanged. |
| 4 | R/W | Access Interface X Message Control Bits '1' : read/write control bits. '0' : Control Bits unchanged. |
| 3 | R/W | Clear(Reset) Interface X Clear Interrupt Pending '1' : clear IntPnd bit when reading the Message Object. '0' : IntPnd bit remains unchanged when reading the Message Object. |
| 2 | R/W | Access Transmission Request / New Data Bit WR/RD = Write '1' : set TxRqst bit. '0' : TxRqst bit unchanged. WR/RD = Read '1' : reset NewDat bit. '0' : NewDat bit unchanged. |
| 1 | R/W | Access Data Byte 0~3 '1' : read/write Data Byte 0~3. '0' : Data Byte 0~3 unchanged. |
| 0 | R/W | Access Data Byte 4~7 '1' : read/write Data Byte 4~7. '0' : Data Byte 4~7 unchanged. |

10.2.3.11 Interface X Mask 1 Register

| CAN_BASE+0x028 / 0x088 | | | | | | | |
|------------------------|-------|-------|-------|-------|-------|------|------|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Msk15 | Msk14 | Msk13 | Msk12 | Msk11 | Msk10 | Msk9 | Msk8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Msk7 | Msk6 | Msk5 | Msk4 | Msk3 | Msk2 | Msk2 | Msk0 |

SEE THE EXPLANATION OF 10.2.3.12

10.2.3.12 Interface X Mask 2 Register

| CAN_BASE+0x02C / 0x08C | | | | | | | |
|------------------------|-------|----------|-------|-------|-------|-------|-------|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MXtd | Mdir | Reserved | Msk28 | Msk27 | Msk26 | Msk25 | Msk24 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Msk23 | Msk22 | Msk21 | Msk20 | Msk19 | Msk18 | Msk17 | Msk16 |

MASK28-0 : Identifier Mask(Read/Write access)

Msk28-18 : Identifier Mask Standard Message

Msk28-0 : Identifier Mask Extended Message.

'1' : The corresponding identifier bit is used for acceptance filtering.

'0' : The corresponding bit in the identifier of the message object cannot inhibit the match in the acceptance filtering.

| Bits | Type | Function |
|------|------|----------|
|------|------|----------|

| | | |
|----|-----|--|
| 15 | R/W | Mask Extended Identifier '1' : The extended identifier bit(IDE) is used for acceptance filtering. '0' : The extended identifier bit(IDE) has no effect on the acceptance filtering |
| 14 | R/W | Mask Message Direction '1' : The message direction bit(RTR) is used for acceptance filtering. '0' : The message direction bit(RTR) has no effect on the acceptance filtering |

10.2.3.13 Interface X Arbitration 1 Register

| | | | | | | | |
|------------------------|------|------|------|------|------|-----|-----|
| CAN_BASE+0x030 / 0x090 | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| ID15 | ID14 | ID13 | ID12 | ID11 | ID10 | ID9 | ID8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ID7 | ID6 | ID5 | ID4 | ID3 | ID2 | ID1 | ID0 |

SEE THE EXPLANATION OF 10.2.3.14

10.2.3.14 Interface X Arbitration 2 Register

| | | | | | | | |
|------------------------|------|------|------|------|------|------|------|
| CAN_BASE+0x034 / 0x094 | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MsgVal | Xtd | Dir | ID28 | ID27 | ID26 | ID25 | ID24 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ID23 | ID22 | ID21 | ID20 | ID19 | ID18 | ID17 | ID16 |

ID28-0 : Identifier Message(Read/Write access)

ID28-18 : Identifier Standard Message

ID28-0 : Identifier Extended Message.

| Bits | Type | Function |
|------|------|---|
| 15 | R/W | Message Validation '1' : The Message Object is configured and should be considered by the Message Handler. '0' : The Message Object is ignored by the Message Handler. |
| 14 | R/W | Extended Identifier '1' : The extended Identifier(19 bit) will be used for this Message Object. '0' : The Standard Identifier(11 bit) will be used for this Message Object. |
| 13 | R/W | Message Direction '1' : Transmit '0' : Receive |

10.2.3.15 Interface X Message Control Register

| | | | | | | | |
|------------------------|----------|--------|-------|----------|------|-------|--------|
| CAN_BASE+0x038 / 0x098 | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| NewDat | MsgLst | IntPnd | UMask | TxIE | RxIE | RmtEn | TxRqst |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| EoB | Reserved | | | DLC(3-0) | | | |

| Bits | Type | Function |
|------|------|---|
| 15 | R/W | New Data '1' : The Message Handler or the CPU has written new data into the data portion of this Message Object. '0' : No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU. |
| 14 | R/W | Message Lost(Only valid for direction = receive) '1' : The Message Handler Stored a new message into this object when NewDat was still set, the CPU has lost a message. '0' : No message lost since last time this bit was reset by the CPU. |
| 13 | R/W | Interrupt Pending '1' : This message object has generated an interrupt. '0' : No interrupt was generated by this message object since last time the CPU has cleared this flag. |

| | | |
|-----|-----|---|
| 12 | R/W | Use Identifier Mask '1' : Use Identifier Mask. '0' : Identifier Mask ignored. |
| 11 | R/W | Transmit Interrupt Enable '1' : An interrupt is generated after a successful transmission of a frame. '0' : No interrupt is generated after a successful transmission of a frame. |
| 10 | R/W | Receive Interrupt Enable '1' : An interrupt is generated after a successful reception of a frame. '0' : No interrupt is generated after a successful reception of a frame. |
| 9 | R/W | Remote Enable '1' : At the reception of a Remote Frame, TxRqst is set. '0' : At the reception of a Remote Frame, TxRqst is left unchanged. |
| 8 | R/W | Transmit Request '1' : The transmit of this Message Object is requested and is not yet done. '0' : This Message Object is not waiting for transmission. |
| 7 | R/W | End of Buffer(For normal operation, this bit must be set to one) '1' : FIFO operation – Last Message Object of FIFO Buffer. '0' : FIFO operation – If MsgVal is set, this Message Object stores the Next Message. |
| 3:0 | R/W | Data Length Code Number of Data Bytes "0000" : 0 "0001" : 1 "0010" : 2 "0011" : 3 "0100" : 4 "0101" : 5 "0110" : 6 "0111" : 7 "1000" : 8 |

10.2.3.16 Interface X Data A1 Register

| | | | | | | | |
|------------------------|-------------|-------------------------------|----|----|----|---|---|
| CAN_BASE+0x03C / 0x09C | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Data1 | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Data0 | | | | | | | |
| Bits | Type | Function | | | | | |
| 15:8 | R/W | Data1 : Shift Register Byte 1 | | | | | |
| 7:0 | R/W | Data0 : Shift Register Byte 0 | | | | | |

10.2.3.17 Interface X Data A2 Register

| | | | | | | | |
|------------------------|-------------|-------------------------------|----|----|----|---|---|
| CAN_BASE+0x040 / 0x0A0 | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Data3 | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Data2 | | | | | | | |
| Bits | Type | Function | | | | | |
| 15:8 | R/W | Data3 : Shift Register Byte 3 | | | | | |
| 7:0 | R/W | Data2 : Shift Register Byte 2 | | | | | |

10.2.3.18 Interface X Data B1 Register

| | | | | | | | |
|------------------------|----|----|----|----|----|---|---|
| CAN_BASE+0x044 / 0x0A4 | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Data5 | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

Data4

| Bits | Type | Function |
|------|------|-------------------------------|
| 15:8 | R/W | Data5 : Shift Register Byte 5 |
| 7:0 | R/W | Data4 : Shift Register Byte 4 |

10.2.3.19 Interface X Data B2 Register

CAN_BASE+0x048 / 0x0A8

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|-------|------|-------------------------------|----|----|----|---|---|
| Data7 | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Data6 | | | | | | | |
| Bits | Type | Function | | | | | |
| 15:8 | R/W | Data7 : Shift Register Byte 7 | | | | | |
| 7:0 | R/W | Data6 : Shift Register Byte 6 | | | | | |

10.2.3.20 Transmission Request 1 Register

CAN_BASE+0x100

| | | | | | | | |
|---------------|----|----|----|----|----|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| TxRqst 16 – 9 | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TxRqst 8 – 1 | | | | | | | |

SEE THE EXPLANATION OF [10.2.3.21]

10.2.3.21 Transmission Request 2 Register

CAN_BASE+0x104

| | | | | | | | |
|----------------|----|----|----|----|----|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| TxRqst 32 – 25 | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TxRqst 24 – 17 | | | | | | | |

TxRqst 32 – 1 : Transmission Request Bits(Read-Only)

‘1’ : This transmission of this message object is requested and is not yet done.

‘0’ : This message object is not waiting for transmission.

10.2.3.22 New Data 1 Register

CAN_BASE+0x120

| | | | | | | | |
|-----------------|----|----|----|----|----|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| New Data 16 – 9 | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| New Data 8 – 1 | | | | | | | |

SEE THE EXPLANATION OF [10.2.3.23]

10.2.3.23 New Data 2 Register

CAN_BASE+0x124

| | | | | | | | |
|------------------|----|----|----|----|----|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| New Data 32 – 25 | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| New Data 24 – 17 | | | | | | | |

NewDat 32 – 1 : New Data Bits(Read-only)

‘1’ : The Message Handler or the CPU has written new data into the data portion of this Message Object.

‘0’ : No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the CPU.

10.2.3.24 Interrupt Pending 1 Register

| | | | | | | | | |
|---------------|----|----|----|----|----|---|---|----------------|
| | | | | | | | | CAN_BASE+0x140 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | |
| IntPnd 16 – 9 | | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| IntPnd 8 – 1 | | | | | | | | |

SEE THE EXPLANATION OF [10.2.3.25]

10.2.3.25 Interrupt Pending 2 Register

| | | | | | | | | |
|----------------|----|----|----|----|----|---|---|----------------|
| | | | | | | | | CAN_BASE+0x144 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | |
| IntPnd 32 – 25 | | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| IntPnd 24 – 17 | | | | | | | | |

IntPnd 32 – 1 : Interrupt Pending(Read-Only)

‘1’ : This Message Object has generated an interrupt.

‘0’ : No interrupt was generated by this message object since last time the CPU has cleared this flag.

10.2.3.26 Message Valid 1 Register

| | | | | | | | | |
|---------------|----|----|----|----|----|---|---|----------------|
| | | | | | | | | CAN_BASE+0x160 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | |
| MsgVal 16 – 9 | | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| MsgVal 8 – 1 | | | | | | | | |

SEE THE EXPLANATION OF [10.2.3.27]

10.2.3.27 Message Valid 2 Register

| | | | | | | | | |
|----------------|----|----|----|----|----|---|---|----------------|
| | | | | | | | | CAN_BASE+0x164 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | |
| MsgVal 32 – 25 | | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| MsgVal 24 – 17 | | | | | | | | |

MsgVal 32 – 1 : Message Validation(Read-Only)

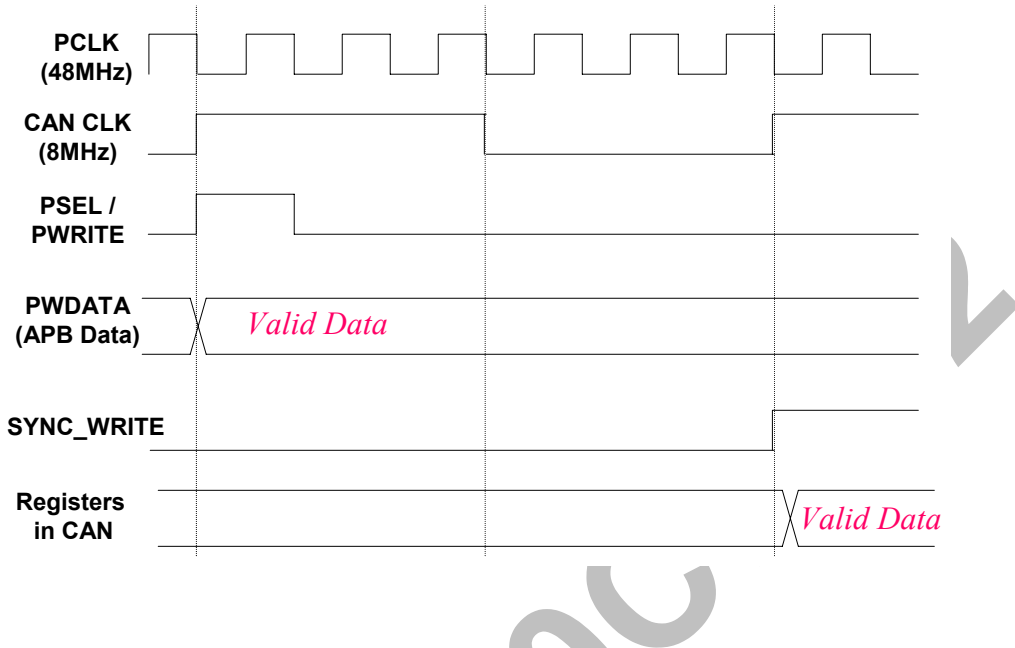
‘1’ : This Message Object is configured and should be considered by the Message Handler.

‘0’ : This Message Object is ignored by the Message Handler.

NOTE : CAN Data Read/Write Timing Diagram

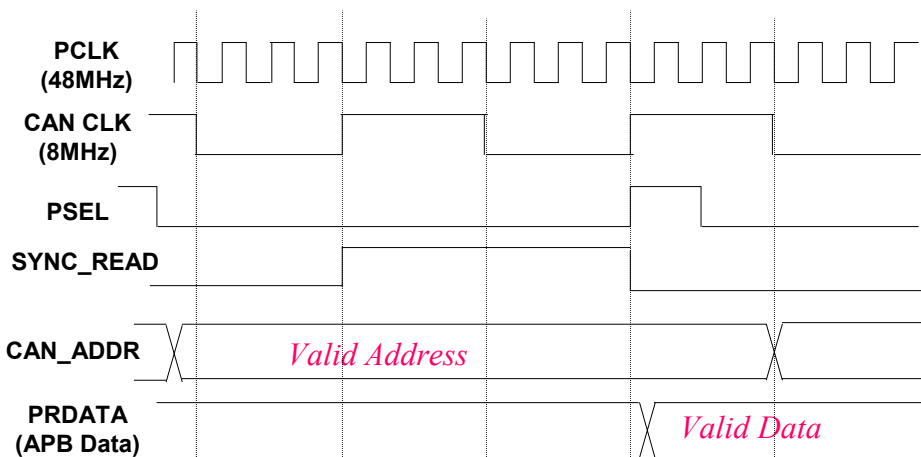
Data Write into CAN Block

- Two consecutive write accesses must have a minimum distance of 12 PCLK period(2 CAN clock).



Data Read from CAN Block

- A read access to CAN module with APB interface must be performed as "Double Read".
- Two consecutive read accesses must have a minimum distance of 13 PCLK period(2 CAN clock + 1 PCLK).



NOTE : For more information about CAN used in HMS30C7202 and its application, please refer "[CAN User Manual of HMS30C7202 for Software Engineer](#)" published by SP-SoC team in Hynix Semiconductor Inc. .

10.3 GPIO

This document describes the Programmable Input /Output module (PIO). This is an AMBA slave module that connects to the Advanced Peripheral Bus (APB). For more information about AMBA, please refer to the AMBA Specification (ARM IHI 0001).

The I/O status is not changed during “Sleep mode” or “Deep Sleep mode”.

10.3.1 External Signals

| Pin Name | Type | Description |
|--------------|------|--|
| KSCANI [7:0] | I/O | GPIO PORTA [15:8] |
| KSCANO [7:0] | I/O | GPIO PORTA [7:0] |
| PORTB [11:6] | I/O | GPIO PORTB [11:6] PORTB[11:10] : dedicated to the external interrupt of PMU |
| nUDCD0 | I/O | GPIO PORTB [5] |
| nUDSR0 | I/O | GPIO PORTB [4] |
| nURTS0 | I/O | GPIO PORTB [3] |
| nUCTS0 | I/O | GPIO PORTB [2] |
| nUDTR0 | I/O | GPIO PORTB [1] |
| nURING0 | I/O | GPIO PORTB [0] |
| nRCS3 | I/O | GPIO PORTC [10] |
| nRCS2 | I/O | GPIO PORTC [9] |
| nDMAACK | I/O | GPIO PORTC [8] |
| nDMAREQ | I/O | GPIO PORTC [7] |
| PWM1 | I/O | GPIO PORTC [6] |
| PWM0 | I/O | GPIO PORTC [5] |
| PS2CK | I/O | GPIO PORTC [4] |
| PS2D | I/O | GPIO PORTC [3] |
| CANRx0 | I/O | GPIO PORTC [2] |
| CANTx0 | I/O | GPIO PORTC [1] |
| TimerOut | I/O | GPIO PORTC [0] |
| LBLEN | I/O | GPIO PORTD [8] |
| LD [15:8] | I/O | GPIO PORTD [7:0] |
| RA [24] | I/O | GPIO PORTE [24] |
| CANTx1 | I/O | GPIO PORTE [23] |
| CANRx1 | I/O | GPIO PORTE [22] |
| MMCLK | I/O | GPIO PORTE [21] |
| MMCCD | I/O | GPIO PORTE [20] |
| MMCDAT | I/O | GPIO PORTE [19] |
| MMCCMD | I/O | GPIO PORTE [18] |
| nRW3 | I/O | GPIO PORTE [17] |
| nRW2 | I/O | GPIO PORTE [16] |
| RD [31:16] | I/O | GPIO PORTE [15:0] |

10.3.2 Registers

| Address | Name | Width | Default | Description |
|-------------|-------|-------|---------|--------------------------------------|
| 0x8002.3000 | ADATA | 16 | 0x0000 | GPIO PORTA Data register |
| 0x8002.3004 | ADIR | 16 | 0xFFFF | GPIO PORTA Data Direction register |
| 0x8002.3008 | AMASK | 16 | 0x0000 | GPIO PORTA Interrupt Mask register |
| 0x8002.300C | ASTAT | 16 | 0x0000 | GPIO PORTA Interrupt Status register |
| 0x8002.3010 | AEDGE | 16 | 0x0000 | GPIO PORTA Edge Mode register |
| 0x8002.3014 | ACLIR | 16 | 0x0000 | GPIO PORTA Clear register |
| 0x8002.3018 | APOL | 16 | 0x0000 | GPIO PORTA Polarity register |
| 0x8002.301C | AEN | 16 | 0x0000 | GPIO PORTA Enable register |
| 0x8002.3020 | BDATA | 12 | 0x0000 | GPIO PORTB Data register |
| 0x8002.3024 | BDIR | 12 | 0xFFFF | GPIO PORTB Data Direction register |

| | | | | |
|-------------|---------|----|-----------|---|
| 0x8002.3028 | BMASK | 12 | 0x000 | GPIO PORTB Interrupt Mask register |
| 0x8002.302C | BSTAT | 12 | 0x000 | GPIO PORTB Interrupt Status register |
| 0x8002.3030 | BEDGE | 12 | 0x000 | GPIO PORTB Edge Moderegister |
| 0x8002.3034 | BCLR | 12 | 0x000 | GPIO PORTB Clear register |
| 0x8002.3038 | BPOL | 12 | 0x000 | GPIO PORTB Polarity register |
| 0x8002.303C | BEN | 6 | 0x00 | GPIO PORTB Enable register |
| 0x8002.3040 | CDATA | 11 | 0x000 | GPIO PORTC Data register |
| 0x8002.3044 | CADIR | 11 | 0x7FF | GPIO PORTC Data Direction register |
| 0x8002.3048 | CMASK | 11 | 0x000 | GPIO PORTC Interrupt Mask register |
| 0x8002.304C | CSTAT | 11 | 0x000 | GPIO PORTC Interrupt Status register |
| 0x8002.3050 | CEDGE | 11 | 0x000 | GPIO PORTC Edge Mode register |
| 0x8002.3054 | CCLR | 11 | 0x000 | GPIO PORTC Clear register |
| 0x8002.3058 | CPOL | 11 | 0x000 | GPIO PORTC Polarity register |
| 0x8002.305C | CEN | 11 | 0x000 | GPIO PORTC Enable register |
| 0x8002.3060 | DDATA | 9 | 0x000 | GPIO PORTD Data register |
| 0x8002.3064 | DDIR | 9 | 0x1FF | GPIO PORTD Data Direction register |
| 0x8002.3068 | DMASK | 9 | 0x000 | GPIO PORTD Interrupt Mask register |
| 0x8002.306C | DSTAT | 9 | 0x000 | GPIO PORTD Interrupt Status register |
| 0x8002.3070 | DEEDGE | 9 | 0x000 | GPIO PORTD Edge Mode register |
| 0x8002.3074 | DCLR | 9 | 0x000 | GPIO PORTD Clear register |
| 0x8002.3078 | DPOL | 9 | 0x000 | GPIO PORTD Polarity register |
| 0x8002.307C | DEN | 9 | 0x000 | GPIO PORTD Enable register |
| 0x8002.3080 | EDATA | 25 | 0x0000000 | GPIO PORTE Data register |
| 0x8002.3084 | EDIR | 25 | 0x1FFFFFF | GPIO PORTE Data Direction register |
| 0x8002.3088 | EMASK | 25 | 0x0000000 | GPIO PORTE Interrupt Mask register |
| 0x8002.308C | ESTAT | 25 | 0x0000000 | GPIO PORTE Interrupt Status register |
| 0x8002.3090 | EEDGE | 25 | 0x0000000 | GPIO PORTE Edge Mode register |
| 0x8002.3094 | ECLR | 25 | 0x0000000 | GPIO PORTE Clear register |
| 0x8002.3098 | EPOL | 25 | 0x0000000 | GPIO PORTE Polarity register |
| 0x8002.309C | EEN | 25 | 0x0000000 | GPIO PORTE Enable register |
| 0x8002.30A0 | TICTMDR | 1 | 0x0 | GPIO Tic Test Mode register |
| 0x8002.30A4 | AMULSEL | 16 | 0x0000 | GPIO PORTA Multi-function Select register |
| 0x8002.30A8 | SWAP | 1 | 0x0 | SWAP Pin Configuration register |

10.3.2.1 ADATA

0x8002.3000

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--|------|---|----|----|----|---|---|---|---|---|---|---|---|---|---|
| ADATA, ADIR, AMASK, ASTAT, AEDGE, ACLR, APOL, AEN [15:0] | | | | | | | | | | | | | | | |
| Bits | Type | Function | | | | | | | | | | | | | |
| 16 | R/W | Values written to this register will be output on port [A,B,C,D,E] pins if the corresponding data direction bits are set Low (port output). Values read from this register reflect the external state of port [A,B,C,D,E] not necessarily the value written to it. All bits are cleared by a system reset. When the PIO pin is defined as input, this input can be an interrupt source with register setting. On reads, the Data Register contains the current status of correspondent port pins, whether they are configured as input or output. Writing to a Data Register only affects the pins that are configured as outputs. All PIO input pins can be used as interrupt source with enabled interrupt mask register bit. These interrupt sources can be selected as active HIGH/LOW, EDGE/LEVEL trigger mode. | | | | | | | | | | | | | |

10.3.2.2 ADIR

0x8002.3004

| Bits | Type | Function | | | | | | | | | | | | | |
|------|------|--|--|--|--|--|--|--|--|--|--|--|--|--|--|
| 16 | R/W | Bits set in this register will select the corresponding pin in port [A,B,C,D,E] to become an input, clearing a bit sets the pin to output. All bits are set by a system reset. | | | | | | | | | | | | | |

10.3.2.3 AMASK

0x8002.3008

| Bits | Type | Function |
|------|------|---|
| 16 | R/W | Bits set in this register will select the corresponding pin to become an interrupt source. All bits are cleared by a system reset. 0 = disable interrupt (default) 1 = enable interrupt |

10.3.2.4 ASTAT

0x8002.300C

| Bits | Type | Function |
|------|------|--|
| 16 | RO | All PIO signals can be used as interrupt sources according to the settings. Each port has the following registers and the interrupt signals to interrupt controller. Interrupt controller receives active HIGH, level mode interrupt sources only. But PIO block can receive not only active HIGH or active LOW, but also level or edge mode signals. Then it interprets and sends interrupt request to the interrupt controller. All bits can be controlled separately. Values in this 16-bit read-only register represents that the interrupt requests are pending on corresponding pins. All bits are cleared by a system reset. 0 = no interrupt request 1 = interrupt pending (masked interrupt is always 0) |

10.3.2.5 AEDGE

0x8002.3010

| Bits | Type | Function |
|------|------|---|
| 16 | R/W | Bits set in this 16-bit read/write register will select the corresponding pin to become an edge mode interrupt source. All bits are cleared by a system reset. 0 = level mode (default) 1 = edge mode |

10.3.2.6 ACLR

0x8002.3014

| Bits | Type | Function |
|------|------|--|
| 16 | WO | Bits set in this 16-bit write-only register will clear the stored interrupt request of corresponding bit in edge mode. All bits are automatically cleared after written. 0 = no action (default) 1 = clear interrupt source (self reset) |

10.3.2.7 APOL

0x8002.3018

| Bits | Type | Function |
|------|------|--|
| 16 | R/W | Bits set in this 16-bit read/write register will select the corresponding pin to become an active LOW mode interrupt source. All bits are cleared by a system reset. After accessing this register, the Edge Mode register should be cleared with the Clear register. 0 = active HIGH mode 1 = active LOW mode |

10.3.2.8 GPIO PORT A Enable Register

| | | | | | | | |
|---------|---------|---------|---------|---------|---------|--------|--------|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| PORTA15 | PORTA14 | PORTA13 | PORTA12 | PORTA11 | PORTA10 | PORTA9 | PORTA8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PORTA7 | PORTA6 | PORTA5 | PORTA4 | PORTA3 | PORTA2 | PORTA1 | PORTA0 |

0x8002.301C

| Bits | Type | Function |
|------|------|---|
| 15 | R/W | GPIO PORT A[15] Enable 1: PORT A[15] 0: KSCAN0[7] |

| | | | | |
|----|-----|------------------------|---------------|--------------|
| 14 | R/W | GPIO PORT A[14] Enable | 1: PORT A[14] | 0: KSCAN0[6] |
| 13 | R/W | GPIO PORT A[13] Enable | 1: PORT A[13] | 0: KSCAN0[5] |
| 12 | R/W | GPIO PORT A[12] Enable | 1: PORT A[12] | 0: KSCAN0[4] |
| 11 | R/W | GPIO PORT A[11] Enable | 1: PORT A[11] | 0: KSCAN0[3] |
| 10 | R/W | GPIO PORT A[10] Enable | 1: PORT A[10] | 0: KSCAN0[2] |
| 9 | R/W | GPIO PORT A[9] Enable | 1: PORT A[9] | 0: KSCAN0[1] |
| 8 | R/W | GPIO PORT A[8] Enable | 1: PORT A[8] | 0: KSCAN0[0] |
| 7 | R/W | GPIO PORT A[7] Enable | 1: PORT A[7] | 0: KSCANI[7] |
| 6 | R/W | GPIO PORT A[6] Enable | 1: PORT A[6] | 0: KSCANI[6] |
| 5 | R/W | GPIO PORT A[5] Enable | 1: PORT A[5] | 0: KSCANI[5] |
| 4 | R/W | GPIO PORT A[4] Enable | 1: PORT A[4] | 0: KSCANI[4] |
| 3 | R/W | GPIO PORT A[3] Enable | 1: PORT A[3] | 0: KSCANI[3] |
| 2 | R/W | GPIO PORT A[2] Enable | 1: PORT A[2] | 0: KSCANI[2] |
| 1 | R/W | GPIO PORT A[1] Enable | 1: PORT A[1] | 0: KSCANI[1] |
| 0 | R/W | GPIO PORT A[0] Enable | 1: PORT A[0] | 0: KSCANI[0] |

10.3.2.9 BDATA

| | | | | | | | | | | | | |
|---|----|---|---|---|---|---|---|---|---|---|---|-------------|
| 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 0x8002.3020 |
| BDATA, BDIR, BMASK, BSTAT, BEDGE, BCLR, BPOL, BEN | | | | | | | | | | | | |

10.3.2.10 BDIR

0x8002.3024

10.3.2.11 BMASK

0x8002.3028

10.3.2.12 BSTAT

0x8002.302C

10.3.2.13 BEDGE

0x8002.3030

10.3.2.14 BCLK

0x8002.3034

10.3.2.15 BPOL

0x8002.3038

10.3.2.16 GPIO PORT B Enable Register

| | | | | | | | | |
|----------|---|--------|--------|--------|--------|--------|--------|-------------|
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 0x8002.303C |
| Reserved | | PORTB5 | PORTB4 | PORTB3 | PORTB2 | PORTB1 | PORTB0 | |

| Bits | Type | Function |
|------|------|--|
| 5 | R/W | GPIO PORT B[5] Enable 1: PORT B[5] 0: nUDCD |
| 4 | R/W | GPIO PORT B[4] Enable 1: PORT B[4] 0: nUDSR |
| 3 | R/W | GPIO PORT B[3] Enable 1: PORT B[3] 0: nURTS |
| 2 | R/W | GPIO PORT B[2] Enable 1: PORT B[2] 0: nUCTS |
| 1 | R/W | GPIO PORT B[1] Enable 1: PORT B[1] 0: nUDTR |
| 0 | R/W | GPIO PORT B[0] Enable 1: PORT B[0] 0: nURING |

10.3.2.17 CDATA

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|-------------|
| 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 0x8002.3040 |
| CDATA, CDIR, CMASK, CSTAT, CEDGE, CCLR, CPOL, CEN | | | | | | | | | | | |

10.3.2.18 CDIR

0x8002.3044

| | | |
|-----------|-----------------------------|-------------|
| 10.3.2.19 | CMASK | 0x8002.3048 |
| 10.3.2.20 | CBSTAT | 0x8002.304C |
| 10.3.2.21 | CEEDGE | 0x8002.3050 |
| 10.3.2.22 | CCLK | 0x8002.3054 |
| 10.3.2.23 | CPOL | 0x8002.3058 |
| 10.3.2.24 | GPIO PORT C Enable Register | 0x8002.305C |

| | | | | | | | | |
|----------|--------|--------|--------|--------|--------|---------|--------|--------|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | |
| Reserved | | | | | | PORTC10 | PORTC9 | PORTC8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| PORTC7 | PORTC6 | PORTC5 | PORTC4 | PORTC3 | PORTC2 | PORTC1 | PORTC0 | |

0x8002.305C

| Bits | Type | Function |
|------|------|--|
| 10 | R/W | GPIO PORT C[10] Enable 1: PORT C[10] 0: nRCS3 |
| 9 | R/W | GPIO PORT C[9] Enable 1: PORT C[9] 0: nRCS2 |
| 8 | R/W | GPIO PORT C[8] Enable 1: PORT C[8] 0: nDMAACK |
| 7 | R/W | GPIO PORT C[7] Enable 1: PORT C[7] 0: nDMAREQ |
| 6 | R/W | GPIO PORT C[6] Enable 1: PORT C[6] 0: PWM1 |
| 5 | R/W | GPIO PORT C[5] Enable 1: PORT C[5] 0: PWM0 |
| 4 | R/W | GPIO PORT C[4] Enable 1: PORT C[4] 0: PS2CK |
| 3 | R/W | GPIO PORT C[3] Enable 1: PORT C[3] 0: PS2D |
| 2 | R/W | GPIO PORT C[2] Enable 1: PORT C[2] 0: CANRx0 |
| 1 | R/W | GPIO PORT C[1] Enable 1: PORT C[1] 0: CANTx0 |
| 0 | R/W | GPIO PORT C[0] Enable 1: PORT C[0] 0: TimerOut |

| | | |
|-----------|-------|-------------|
| 10.3.2.25 | DDATA | 0x8002.3060 |
|-----------|-------|-------------|

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DDATA, DDIR, DMASK, DSTAT, DEDGE, DCLR, DPOL, DEN | | | | | | | | |

| | | |
|-----------|-----------------------------|-------------|
| 10.3.2.26 | DDIR | 0x8002.3064 |
| 10.3.2.27 | DMASK | 0x8002.3068 |
| 10.3.2.28 | DBSTAT | 0x8002.306C |
| 10.3.2.29 | DEEDGE | 0x8002.3070 |
| 10.3.2.30 | DCLK | 0x8002.3074 |
| 10.3.2.31 | DPOL | 0x8002.3078 |
| 10.3.2.32 | GPIO PORT D Enable Register | 0x8002.3078 |

| | | | | | | | |
|----------|--------|--------|--------|--------|--------|--------|--------|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | | PORTD8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PORTD7 | PORTD6 | PORTD5 | PORTD4 | PORTD3 | PORTD2 | PORTD1 | PORTD0 |

0x8002.307C

| Bits | Type | Function |
|------|------|--|
| 8 | R/W | GPIO PORT D[8] Enable 1: PORT D[8] 0: LBEn |
| 7:0 | R/W | GPIO PORT D[7:0] Enable 0xFF: PORT D[7:0] 0x00: LD[15:8] |

10.3.2.33 EDATA

0x8002.3080

| 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | | | | | | | |
|--|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| EDATA, EDIR, EMASK, ESTAT, EEDGE, ECLR, EPOL, EEN [24:0] | | | | | | | | | | | | | | | |

10.3.2.34 EDIR

0x8002.3084

10.3.2.35 EMASK

0x8002.3088

10.3.2.36 ESTAT

0x8002.308C

10.3.2.37 EEDGE

0x8002.3090

10.3.2.38 ECLK

0x8002.3094

10.3.2.39 EPOL

0x8002.3098

10.3.2.40 GPIO PORT E Enable Register

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|----------|---------|---------|---------|---------|---------|---------|---------|
| Reserved | | | | | | | PORTE24 |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| PORTE23 | PORTE22 | PORTE21 | PORTE20 | PORTE19 | PORTE18 | PORTE17 | PORTE16 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| PORTE15 | PORTE14 | PORTE13 | PORTE12 | PORTE11 | PORTE10 | PORTE9 | PORTE8 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PORTE7 | PORTE6 | PORTE5 | PORTE4 | PORTE3 | PORTE2 | PORTE1 | PORTE0 |

0x8002.309C

| Bits | Type | Function |
|------|------|--|
| 24 | R/W | GPIO PORT E[24] Enable 1:PORT E[24] 0: RA[24] |
| 23 | R/W | GPIO PORT E[23] Enable 1:PORT E[23] 0: CANTx1 |
| 22 | R/W | GPIO PORT E[22] Enable 1:PORT E[22] 0: CANRx1 |
| 21 | R/W | GPIO PORT E[21] Enable 1:PORT E[21] 0: MMCLK |
| 20 | R/W | GPIO PORT E[20] Enable 1:PORT E[20] 0: MMCCD |
| 19 | R/W | GPIO PORT E[19] Enable 1:PORT E[19] 0: MMCDAT |
| 18 | R/W | GPIO PORT E[18] Enable 1:PORT E[18] 0: MMCCMD |
| 17 | R/W | GPIO PORT E[17] Enable 1:PORT E[17] 0: nRW3 |
| 16 | R/W | GPIO PORT E[16] Enable 1:PORT E[16] 0: nRW2 |
| 15:0 | R/W | GPIO PORT E[15] Enable 0xFFFF : PORT E[15:0] 0x0000: RD[31:16] |

10.3.2.41 Tic Test mode Register(TICTMDR)

0x8002.30A0

| | | | | | | | | |
|--|--|--|--|--|--|--|--|--------|
| | | | | | | | | 0 |
| | | | | | | | | TicSel |

| Bits | Type | Function |
|------|------|--|
| 0 | R/W | When TicSel is HIGH, there is 3 Port registers (B, D, F) access to check up special word. TicSelWR is enabling the TICTMDR and PSTB is clock signal. So TicSel data output is PD[0] bit. |

10.3.2.42 PORTA Multi-function Select register(AMULSEL)

0x8002.30A4

| | | | | | | | | | | | | | | | |
|---------|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| AMULSEL | | | | | | | | | | | | | | | |

| Bits | Type | Function |
|------|------|---|
| 15 | R/W | GPIO PORT A[15] Multi-function Select 1: IRIN 0: GPIO or Primary |
| 14 | R/W | GPIO PORT A[14] Multi-function Select 1: USOUT3 0: GPIO or Primary |
| 13 | R/W | GPIO PORT A[13] Multi-function Select 1: USIN3 0: GPIO or Primary |
| 12 | R/W | GPIO PORT A[12] Multi-function Select 1: ISECK 0: GPIO or Primary |
| 11 | R/W | GPIO PORT A[11] Multi-function Select 1: ISWS 0: GPIO or Primary |
| 10 | R/W | GPIO PORT A[10] Multi-function Select 1: PORT A[10] output 0: GPIO or Primary |
| 9 | R/W | GPIO PORT A[9] Multi-function Select 1: PORT A[9] output 0: GPIO or Primary |
| 8 | R/W | GPIO PORT A[8] Multi-function Select 1: PORT A[8] output 0: GPIO or Primary |
| 7 | R/W | GPIO PORT A[7] Multi-function Select 1: IROUT 0: GPIO or Primary |
| 6 | R/W | GPIO PORT A[6] Multi-function Select 1: USOUT2 0: GPIO or Primary |
| 5 | R/W | GPIO PORT A[5] Multi-function Select 1: USIN2 0: GPIO or Primary |
| 4 | R/W | GPIO PORT A[4] Multi-function Select 1: ISCLK 0: GPIO or Primary |
| 3 | R/W | GPIO PORT A[3] Multi-function Select 1: ISD 0: GPIO or Primary |
| 2 | R/W | GPIO PORT A[2] Multi-function Select 1: PORT A[2] output 0: GPIO or Primary |
| 1 | R/W | GPIO PORT A[1] Multi-function Select 1: PORT A[1] output 0: GPIO or Primary |
| 0 | R/W | GPIO PORT A[0] Multi-function Select 1: PORT A[0] output 0: GPIO or Primary |

10.3.2.43 SWAP Pin Configuration Register(SWAP)

0x8002.30A8

| | | | | | | | | | | | | | | | |
|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|------|
| | | | | | | | | | | | | | | | 0 |
| | | | | | | | | | | | | | | | SWAP |

| Bits | Type | Function |
|------|------|--|
| 0 | R/W | SWAP determines PORT E Pin configuration. When reset, USB transceiver signals, SMC and RA24 will be available. Otherwise, USB transceiver, SMC and CAN 1 will be available while RA 24 cannot be used so addressing space reduced by half. |

10.3.3 GPIO Interrupt

GPIO has 7 interrupt sources. Each port can be configured as 1 interrupt source except port B. To use a GPIO port as interrupt source, specify edge register polarity register according to interrupt type, for example, low level sensitive or rising edge sensitive, etc. then set mask register to enable interrupt. Port B has 3 interrupt sources, PORTB[11], PORTB[10] and PORTB[9:0]. PORTB[11] is assigned to make CPU go to deep sleep mode, PORTB[10] is to detect Hotsync. PORTB[9:0] is used as general GPIO interrupt source. So, following chart shows available GPIO interrupts.

| Interrupt Name | Configurable Bits |
|----------------|---------------------------------|
| GPIOINTR | PORTA[15:0] |
| GPIOB0INTR | PORTB[10], Hotsync Interrupt |
| GPIOB1INTR | PORTB[11], Deep Sleep Interrupt |
| GPIOBINTR | PORTB[9:0] |
| GPIOCINTR | PORTC[10:0] |
| GPIODINTR | PORTD[8:0] |
| GPIOEINTR | PORTE[24:0] |

10.3.4 GPIO Rise/Fall Time

Data output, unit : ns

| Port number | 50pF | | 100pF | | 150pF | |
|---|-------|--------|--------|--------|--------|--------|
| | Rise | Fall | Rise | Fall | Rise | Fall |
| A0~15, B0~11, C0~10, D0~8, E22~23 (*Group A) | 8.745 | 10.687 | 15.946 | 19.917 | 23.136 | 29.147 |
| E0~17,24 (Group B) | 6.098 | 5.693 | 10.896 | 10.317 | 15.696 | 14.927 |
| E18~21 (Group C) | 4.018 | 4.048 | 6.904 | 7.137 | 9.783 | 10.217 |

* It means the drive strength (Group A = 1, Group B = 2, Group C = 4)

HMS30C7202

10.4 Interrupt Controller

The HMS30C7202 has a fully programmable priority, individually maskable, vectored interrupt controller. This feature reduces the software overhead in handling interrupts. The Interrupt controller can trigger the Fast interrupt request (NFIQ) and the standard interrupt request (NIRQ) from any interrupt source (on-chip peripherals and GPIOs). The fully programmable priority encoder allows the user to define the priority of each interrupt source. External interrupt sources can be positive or negative edge triggered or high or low level sensitive, depending on the value programmed in the EDGE and POL registers (see GPIO registers).

| ID Code | Interrupt Source | ID Code | Interrupt Source |
|---------|--------------------------|---------|--|
| 00 | PMU | 10 | Timer1 or Timer2 or Timer3(64Bit) |
| 01 | DMA | 11 | Watchdog |
| 02 | LCD | 12 | CAN0 |
| 03 | Sound | 13 | CAN1 |
| 04 | Reserved | 14 | GPIOB0 (GPIOB [10]) |
| 05 | USB | 15 | GPIOB1 (GPIOB [11]) |
| 06 | MMC | 16 | GPIOA |
| 07 | RTC | 17 | GPIOB |
| 08 | UART0 | 18 | GPIOC |
| 09 | UART1 | 19 | GPIOD |
| 0A | UART2 | 1A | GPIOE |
| 0B | UART3 | 1B | ARM core (COMMRX debug only) |
| 0C | KBD (KeyBoard Interface) | 1C | ARM core (COMMTX debug only) |
| 0D | PS2 | 1D | SmartMedia Card |
| 0E | AIC | 1E | Software (auto generation by CPU register set) |
| 0F | Timer0 | | |

Table 10-2 Interrupt controller Configuration

Note The inputs GPIOB [10] and GPIOB [11] have internally a de-bouncing logic, which allows the direct connection to a button (e.g. for deep sleep and Hot Sync.).

10.4.1 Block diagram

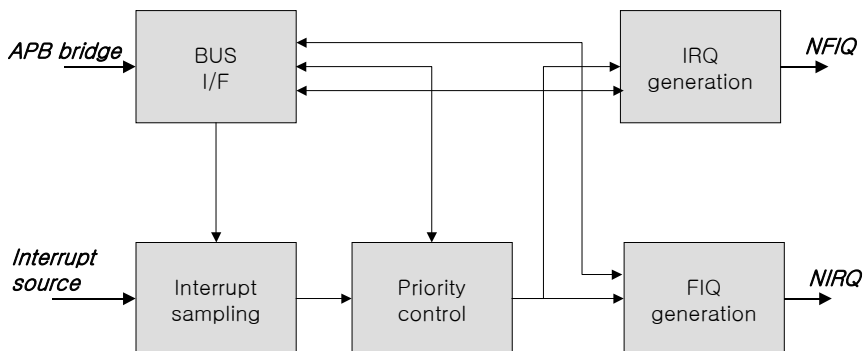


Figure 10-3 Interrupt controller block diagram

10.4.2 Registers

| Address | Name | Width | Default | Description |
|-------------|------|-------|------------|---------------------------|
| 0x8002.4000 | IER | 31 | 0x00000000 | Interrupt enable register |
| 0x8002.4004 | ISR | 31 | 0x00000000 | Interrupt status register |
| 0x8002.4008 | IVR | 32 | 0x00000000 | IRQ vector register |

| | | | | |
|-------------|-------|----|------------|---------------------------|
| 0x8002.4010 | SVR0 | 32 | 0x00000000 | Source vector register 0 |
| 0x8002.4014 | SVR1 | 32 | 0x00000000 | Source vector register 1 |
| 0x8002.4018 | SVR2 | 32 | 0x00000000 | Source vector register 2 |
| 0x8002.401C | SVR3 | 32 | 0x00000000 | Source vector register 3 |
| 0x8002.4020 | SVR4 | 32 | 0x00000000 | Source vector register 4 |
| 0x8002.4024 | SVR5 | 32 | 0x00000000 | Source vector register 5 |
| 0x8002.4028 | SVR6 | 32 | 0x00000000 | Source vector register 6 |
| 0x8002.402C | SVR7 | 32 | 0x00000000 | Source vector register 7 |
| 0x8002.4030 | SVR8 | 32 | 0x00000000 | Source vector register 8 |
| 0x8002.4034 | SVR9 | 32 | 0x00000000 | Source vector register 9 |
| 0x8002.4038 | SVR10 | 32 | 0x00000000 | Source vector register 10 |
| 0x8002.403C | SVR11 | 32 | 0x00000000 | Source vector register 11 |
| 0x8002.4040 | SVR12 | 32 | 0x00000000 | Source vector register 12 |
| 0x8002.4044 | SVR13 | 32 | 0x00000000 | Source vector register 13 |
| 0x8002.4048 | SVR14 | 32 | 0x00000000 | Source vector register 14 |
| 0x8002.404C | SVR15 | 32 | 0x00000000 | Source vector register 15 |
| 0x8002.4050 | SVR16 | 32 | 0x00000000 | Source vector register 16 |
| 0x8002.4054 | SVR17 | 32 | 0x00000000 | Source vector register 17 |
| 0x8002.4058 | SVR18 | 32 | 0x00000000 | Source vector register 18 |
| 0x8002.405C | SVR19 | 32 | 0x00000000 | Source vector register 19 |
| 0x8002.4060 | SVR20 | 32 | 0x00000000 | Source vector register 20 |
| 0x8002.4064 | SVR21 | 32 | 0x00000000 | Source vector register 21 |
| 0x8002.4068 | SVR22 | 32 | 0x00000000 | Source vector register 22 |
| 0x8002.406C | SVR23 | 32 | 0x00000000 | Source vector register 23 |
| 0x8002.4070 | SVR24 | 32 | 0x00000000 | Source vector register 24 |
| 0x8002.4074 | SVR25 | 32 | 0x00000000 | Source vector register 25 |
| 0x8002.4078 | SVR26 | 32 | 0x00000000 | Source vector register 26 |
| 0x8002.407C | SVR27 | 32 | 0x00000000 | Source vector register 27 |
| 0x8002.4080 | SVR28 | 32 | 0x00000000 | Source vector register 28 |
| 0x8002.4084 | SVR29 | 32 | 0x00000000 | Source vector register 29 |
| 0x8002.4088 | SVR30 | 32 | 0x00000000 | Source vector register 30 |
| 0x8002.4090 | IDR | 32 | 0x00001F1F | Interrupt ID register |
| 0x8002.4094 | PSR0 | 32 | 0x03020100 | Priority set register 0 |
| 0x8002.4098 | PSR1 | 32 | 0x07060504 | Priority set register 1 |
| 0x8002.409C | PSR2 | 32 | 0x0B0A0908 | Priority set register 2 |
| 0x8002.40A0 | PSR3 | 32 | 0x0F0E0D0C | Priority set register 3 |
| 0x8002.40A4 | PSR4 | 32 | 0x13121110 | Priority set register 4 |
| 0x8002.40A8 | PSR5 | 32 | 0x17161514 | Priority set register 5 |
| 0x8002.40AC | PSR6 | 32 | 0x1B1A1918 | Priority set register 6 |
| 0x8002.40B0 | PSR7 | 32 | 0x001E1D1C | Priority set register 7 |

Table 10-3 Interrupt controller Register Summary

10.4.2.1 Interrupt Enable Register (IER)

This register is used to enable/disable the interrupt request of interrupt sources.

0x8002.4000

| Bits | Type | Function |
|------|------|--|
| 31 | R/W | 0 : enable FIQ for priority 0 interrupts , 1 : disable FIQ (a priority 0 interrupt will trigger IRQ) |
| 30 | R/W | Software Interrupt |
| 29 | R/W | SmartMedia Card |
| 28 | R/W | ARM core (COMMTX: debug only) |
| 27 | R/W | ARM core (COMMRX: debug only) |
| 26 | R/W | GPIO port E |
| 25 | R/W | GPIO port D |
| 24 | R/W | GPIO port C |
| 23 | R/W | GPIO port B |
| 22 | R/W | GPIO port A |
| 21 | R/W | External Interrupt1 (GPIOB[11]) |
| 20 | R/W | External Interrupt0 (GPIOB[10]) |

| | | |
|----|-----|-----------------------------------|
| 19 | R/W | CAN1 |
| 18 | R/W | CAN0 |
| 17 | R/W | Watchdog timer |
| 16 | R/W | Timer1 or Timer2 or Timer3(64Bit) |
| 15 | R/W | Timer0 |
| 14 | R/W | AIC |
| 13 | R/W | PS2 |
| 12 | R/W | KBD (keyboard interface) |
| 11 | R/W | UART3 |
| 10 | R/W | UART2 |
| 9 | R/W | UART1 |
| 8 | R/W | UART0 |
| 7 | R/W | RTC |
| 6 | R/W | MMC |
| 5 | R/W | USB |
| 4 | R/W | Reserved |
| 3 | R/W | Sound |
| 2 | R/W | LCD |
| 1 | R/W | DMA |
| 0 | R/W | PMU |

Note

0: Disable interrupt / 1: Enable interrupt

The interrupt signals of Timer 1, 2, and 3 are merged into one interrupt source in Timer Block. So, you can use these ORed signal as one interrupt source.

10.4.2.2 Interrupt Status Register (ISR)

The IRQ Status register indicates whether or not the interrupt source has triggered an IRQ interrupt.

0x8002.4004

| Bits | Type | Function |
|------|------|-----------------------------------|
| 31 | R/O | Reserved |
| 30 | R/O | Software Interrupt |
| 29 | R/O | SmartMedia Card |
| 28 | R/O | ARM core (COMMTX: debug only) |
| 27 | R/O | ARM core (COMMRX: debug only) |
| 26 | R/O | GPIO port E |
| 25 | R/O | GPIO port D |
| 24 | R/O | GPIO port C |
| 23 | R/O | GPIO port B |
| 22 | R/O | GPIO port A |
| 21 | R/O | External Interrupt1 (GPIOB[11]) |
| 20 | R/O | External Interrupt0 (GPIOB[10]) |
| 19 | R/O | CAN1 |
| 18 | R/O | CAN0 |
| 17 | R/O | Watchdog timer |
| 16 | R/O | Timer1 or Timer2 or Timer3(64Bit) |
| 15 | R/O | Timer0 |
| 14 | R/O | AIC |
| 13 | R/O | PS2 |
| 12 | R/O | KBD (keyboard interface) |
| 11 | R/O | UART3 |
| 10 | R/O | UART2 |
| 9 | R/O | UART1 |
| 8 | R/O | UART0 |
| 7 | R/O | RTC |
| 6 | R/O | MMC |
| 5 | R/O | USB |
| 4 | R/O | Reserved |
| 3 | R/O | Sound |

| | | |
|---|-----|-----|
| 2 | R/O | LCD |
| 1 | R/O | DMA |
| 0 | R/O | PMU |

Note

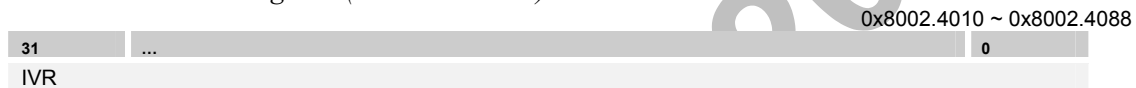
0: No interrupt requested (or interrupt source is disabled)
 1: Interrupt pending

10.4.2.3 IRQ Vector Register (IVR)



| Bits | Type | Function |
|------|------|---|
| 31:0 | R | The IRQ Vectored Register contains the vector programmed by the user in the Source Vector Register corresponding to the current interrupt. The Source Vector Register (0 to 31) is indexed using the ID number in the current interrupt ID register when the IRQ Vector Register is read. When there is no IRQ status, the IRQ Vector Register is set to 0. |

10.4.2.4 Source Vector Register (SVR0 to SVR30)



| Bits | Type | Function |
|------|------|--|
| 31:0 | R/W | The user may store in these registers the address of the corresponding handler for each interrupt source. This interrupt controller has 31-Source Vector Registers, which are corresponded to ID code. For example the Source Vector Register of the Interrupt by RTC is the SVR7 (Source Vector Register 7) |

10.4.2.5 Interrupt ID Register (IDR)

The Interrupt ID Register returns the current FIQ and IRQ interrupt source number.



| Bits | Type | Function |
|-------|------|----------|
| 31:13 | R | Reserved |
| 12:8 | R | FIQID |
| 7:5 | R | Reserved |
| 4:0 | R | IRQID |

10.4.2.6 Priority Set Register (PSR0 to PSR7)

The Priority Set Registers consist of 8 registers, representing 32 priority levels. Each interrupt source (see table 10-2) has its (unique) priority level. The FIQ interrupt source is defined in PSR0[7:0], e.g. if PSR0[7:0] = 0x09, UART 1 can trigger the FIQ interrupt.



| Register | Bits | Type | Initial ID value | Function |
|----------|-------|------|------------------|-----------------|
| PSR7 | 31:24 | R | 0x00 | Reserved |
| | 23:16 | R/W | 0x1E | IRQ priority 1E |
| | 15:8 | R/W | 0x1D | IRQ priority 1D |
| | 7:0 | R/W | 0x1C | IRQ priority 1C |
| PSR6 | 31:24 | R/W | 0x1B | IRQ priority 1B |
| | 23:16 | R/W | 0x1A | IRQ priority 1A |

| | | | | |
|------|-------|-----|------|---------------------------------------|
| | 15:8 | R/W | 0x19 | IRQ priority 19 |
| | 7:0 | R/W | 0x18 | IRQ priority 18 |
| PSR5 | 31:24 | R/W | 0x17 | IRQ priority 17 |
| | 23:16 | R/W | 0x16 | IRQ priority 16 |
| | 15:8 | R/W | 0x15 | IRQ priority 15 |
| | 7:0 | R/W | 0x14 | IRQ priority 14 |
| PSR4 | 31:24 | R/W | 0x13 | IRQ priority 13 |
| | 23:16 | R/W | 0x12 | IRQ priority 12 |
| | 15:8 | R/W | 0x11 | IRQ priority 11 |
| | 7:0 | R/W | 0x10 | IRQ priority 10 |
| PSR3 | 31:24 | R/W | 0x0F | IRQ priority F |
| | 23:16 | R/W | 0x0E | IRQ priority E |
| | 15:8 | R/W | 0x0D | IRQ priority D |
| | 7:0 | R/W | 0x0C | IRQ priority C |
| PSR2 | 31:24 | R/W | 0x0B | IRQ priority B |
| | 23:16 | R/W | 0x0A | IRQ priority A |
| | 15:8 | R/W | 0x09 | IRQ priority 9 |
| | 7:0 | R/W | 0x08 | IRQ priority 8 |
| PSR1 | 31:24 | R/W | 0x07 | IRQ priority 7 |
| | 23:16 | R/W | 0x06 | IRQ priority 6 |
| | 15:8 | R/W | 0x05 | IRQ priority 5 |
| | 7:0 | R/W | 0x04 | IRQ priority 4 |
| PSR0 | 31:24 | R/W | 0x03 | IRQ priority 3 |
| | 23:16 | R/W | 0x02 | IRQ priority 2 |
| | 15:8 | R/W | 0x01 | IRQ priority 1 |
| | 7:0 | R/W | 0x00 | IRQ priority 0 or FIQ source * |

Note

The Priority Level is to be defined as follows.

IRQ Priority 0 or **FIQ source** > IRQ Priority 1 > IRQ Priority 2 > . . . > IRQ Priority 1D> IRQ Priority 1E

* Disable Interrupt Type Bit(IER Bit31): **FIQ source** / Enable Interrupt Type Bit(IER Bit31) : IRQ priority 0

10.5 Matrix Keyboard Interface Controller

The Matrix keyboard interface controller is an AMBA slave module that connects to the Advanced Peripheral Bus (APB). For more information about AMBA, please refer to the AMBA Specification (ARM IHI 0001). The interface controller is designed to communicate with the external keyboard. The keyboard interface uses the pins KSCANI [7:0], KSCANO [7:0]. It is possible to select one of four scan clock modes.

FEATURES

- Four scanning modes
- 8x8 Matrix
- Byte key buffers

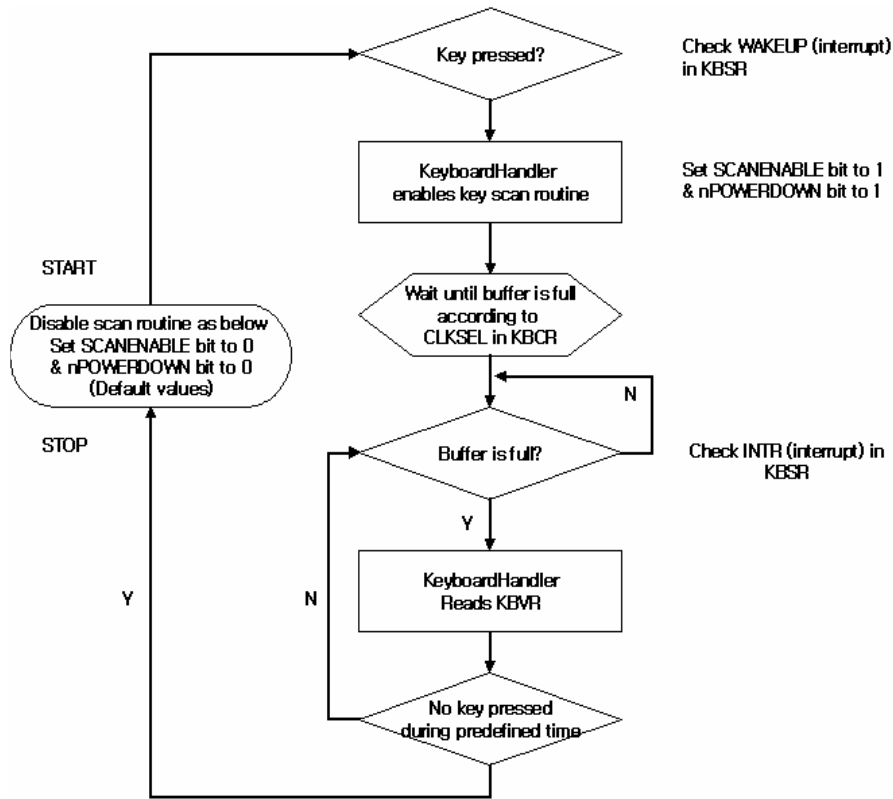


Figure10-4 A flow chart of the keyboard controller

10.5.1 External Signals

| Pin Name | Type | Description |
|--------------|------|---|
| KSCANO [7:0] | O | This assigns the x-axis' scan line. The value is changed periodically so as to cover every key matrix. During one keyboard scan, KSCANO [7:0] can have 8 different values. Active LOW signal. |
| KSCANI [7:0] | I | This indicates which key is pressed in the assigned scan line. Active LOW signal |

10.5.2 Registers

| Address | Name | Width | Default | Description |
|-------------|------|-------|---------|---------------------------------|
| 0x8002.2000 | KBCR | 8 | 0x0 | Keyboard Configuration register |
| 0x8002.2004 | KBSC | 8 | 0x0 | Keyboard Scanout register |

| | | | | |
|-------------|-------|----|-----|---------------------------|
| 0x8002.2008 | KBTR | 8 | 0x0 | Keyboard Test register |
| 0x8002.200C | KBVR0 | 32 | 0x0 | Keyboard value register 0 |
| 0x8002.2010 | KBVR1 | 32 | 0x0 | Keyboard value register 1 |
| 0x8002.2018 | KBSR | 1 | 0x0 | Keyboard status register |

Table 10-4 Matrix Keyboard Interface Controller Register Summary

10.5.2.1 Keyboard Configuration Register (KBCR)

| | | | 0x8002.2000 | | | |
|-------------|------|--|----------------------------------|--|---------|---|
| 7 | | | | 2 | 1 | 0 |
| SCAN ENABLE | | | | nPOWER DOWN | CLK SEL | |
| Bits | Type | Function | | | | |
| 7 | R/W | SCANENABLE bit. This starts or stops matrix keyboard scanning. To start keyboard input scanning, set the SCANENABLE bit and nPOWERDOWN bit of KBCR (Keyboard Configuration Register) and the CLK SEL bit of the KBCR. The key scan control signal is generated. Periodically, column scan code is saved in the 8byte key buffer. After the 8th column key data is stored, keyboard interrupt is generated to make the CPU read 8 scan values. The SCANENABLE bit and nPOWERDOWN bit are usually set or reset simultaneously. When all the column of keyboard has been scanned, an interrupt is generated, and, by interrogating the KBVR registers, software can determine which keys have been pressed. It is software's responsibility to debounce the key pressed information. Keyboard key press interrupts are generated in all PMU states except deep sleep. Start and stop scanning 0 = stop 1 = start | | | | |
| 6:3 | - | Reserved. Keep these bits to zero. | | | | |
| 2 | R/W | nPOWERDOWN bit. In the power down mode, no clock is inputted to this controller logic. 0 = power down mode, where clock is not operating 1 = normal mode, where clock is operating | | | | |
| 1:0 | R/W | CLKSEL bit. This controls the operating clock of scanning matrix keyboard. Base Scanning clock is generated using PCLK (3.6864MHz). | | | | |
| | | Value | Base Scanning Clock Rate | Scan Rate (8byte column buffer) | | |
| | | 00 | PCLK/2 (1.84MHz, test mode only) | 8861 times/sec | | |
| | | 01 | PCLK/128 (28KHz) | 138 times/sec | | |
| | | 10 | PCLK/256 (14KHz) | 69 times/sec | | |
| | | 11 | PCLK/512 (7KHz) | 34 times/sec | | |

10.5.2.2 Keyboard Scanout Register(KBSC)

| | | | 0x8002.2004 |
|------|---------|---|-------------|
| Bits | Initial | Function | |
| 7 | 0 | 0 = 1 st line will be scanned 1 = no scan | |
| 6 | 0 | 0 = 2 nd line will be scanned 1 = no scan | |
| 5 | 0 | 0 = 3 rd line will be scanned 1 = no scan | |
| 4 | 0 | 0 = 4 th line will be scanned 1 = no scan | |
| 3 | 0 | 0 = 5 th line will be scanned 1 = no scan | |
| 2 | 0 | 0 = 6 th line will be scanned 1 = no scan | |
| 1 | 0 | 0 = 7 th line will be scanned 1 = no scan | |

| | | |
|---|---|---|
| 0 | 0 | 0 = 8 th line will be scanned 1 = no scan |
|---|---|---|

10.5.2.3 Keyboard Test Register (KBTR)

0x8002.2008

| Bits | Initial | Function |
|------|---------|--|
| 7 | 1 | Indicates whether 1 st key in the selected scan column is pressed 0 = pressed, 1 = not pressed |
| 6 | 1 | Indicates whether 2 nd key in the selected scan column is pressed 0 = pressed, 1 = not pressed |
| 5 | 1 | Indicates whether 3 rd key in the selected scan column is pressed 0 = pressed, 1 = not pressed |
| 4 | 1 | Indicates whether 4 th key in the selected scan column is pressed 0 = pressed, 1 = not pressed |
| 3 | 1 | Indicates whether 5 th key in the selected scan column is pressed 0 = pressed, 1 = not pressed |
| 2 | 1 | Indicates whether 6 th key in the selected scan column is pressed 0 = pressed, 1 = not pressed |
| 1 | 1 | Indicates whether 7 th key in the selected scan column is pressed 0 = pressed, 1 = not pressed |
| 0 | 1 | Indicates whether 8 th key in the selected scan column is pressed 0 = pressed, 1 = not pressed |

10.5.2.4 Keyboard Value Register (KBVR0)

0x8002.200C

| | | | | | | | | | | | | | | | |
|-------------------------|----|----|----|----|----|----|----|-------------------------|----|----|----|----|----|----|----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| 1st column KSCANI [7:0] | | | | | | | | 2nd column KSCANI [7:0] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 3rd column KSCANI [7:0] | | | | | | | | 4th column KSCANI [7:0] | | | | | | | |

| Bits | Type | Function |
|-------|------|--|
| 31:24 | R | 1st column matrix keyboard scan input data. For example, if the value of KBVR0[32:24] is 00001100, the 5th and 6th keys are pressed and the others are released in 1st column. |
| 23:16 | R | 2nd column matrix keyboard scan input data |
| 15:8 | R | 3rd column matrix keyboard scan input data |
| 7:0 | R | 4th column matrix keyboard scan input data |

10.5.2.5 Keyboard Value Register (KBVR1)

0x8002.2010

| | | | | | | | | | | | | | | | |
|-------------------------|----|----|----|----|----|----|----|-------------------------|----|----|----|----|----|----|----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| 5th column KSCANI [7:0] | | | | | | | | 6th column KSCANI [7:0] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 7th column KSCANI [7:0] | | | | | | | | 8th column KSCANI [7:0] | | | | | | | |

| Bits | Type | Function |
|-------|------|--|
| 31:24 | R | 5th column matrix keyboard scan input data |
| 23:16 | R | 6th column matrix keyboard scan input data |
| 15:8 | R | 7th column matrix keyboard scan input data |
| 7:0 | R | 8th column matrix keyboard scan input data |

10.5.2.6 Keyboard Status Register (KBSR)

0x8002.2018

| | | | | | | | | | | | | | | | | | |
|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--------|------|
| | | | | | | | | | | | | | | | | 1 | 0 |
| | | | | | | | | | | | | | | | | WAKEUP | INTR |

| Bits | Type | Function |
|------|------|----------|
|------|------|----------|

| | | |
|-----|---|---|
| 7:2 | - | Reserved |
| 1 | R | The interrupt and the KBSR bit are cleared after the CPU reads KBSR. The WAKEUP bit is set if any key is pressed when SCANENABLE bit is inactive. Wake up state: 0 = no key pressed or scan enabled 1 = key pressed when scan disabled |
| 0 | R | Key bufferstate: 0 = key buffer is not full 1 = key buffer is full |

HMS30C7202

10.6 PS/2 Interface Controller

This PS/2 Controller is an Advanced Microcontroller Bus Architecture (AMBA) compliant System-on-a-Chip peripheral providing industry-standard PS/2 data transfer channel. A channel has two bi-directional signals that serve as direct interfaces to an external keyboard, mouse or any other PS/2-compatible pointing device. This is an AMBA slave module that connects to the Advanced Peripheral Bus (APB). For more information about AMBA, please refer to the AMBA Specification (ARM IHI 0001).

FEATURES

- AMBA compliant
- PS/2 compatible interface
- Half-duplex bi-directional synchronous serial interface using open-drain outputs for clock and data
- Enable/Disable channel
- Operation in polled or interrupt-driven mode
- Hardware support for PS/2 auxiliary device protocol
- Maskable transmit and receive interrupts
- Automatic odd parity generation and checking
- Optional software based PS/2 implementation
- Test Interface Controller compatible test registers and test modes

10.6.1 External Signals

| Pin Name | Type | Description |
|----------|------|--|
| PSCLK | I/O | PS/2 compatible clock signal pin. Pull-up this pad output (open-drain pad used.) |
| PSDAT | I/O | PS/2 compatible data signal pin. Also pull-up this pad (open-drain). |

10.6.2 Registers

| Address | Name | Width | Default | Description |
|-------------|--------|-------|---------|---|
| 0x8002.C000 | PSDATA | 8 | 00h | Transmit/Receive data register |
| 0x8002.C004 | PSSTAT | 7 | 00h | Internal status register |
| 0x8002.C008 | PSCONF | 6 | 00h | Configuration register |
| 0x8002.C00C | PSINTR | 5 | 00h | Interrupt/Error status and Interrupt ACK register |
| 0x8002.C010 | PSTDLO | 8 | 00h | Timing parameter register |
| 0x8002.C014 | PSTPRI | 8 | 00h | Timing parameter register |
| 0x8002.C018 | PSTXMT | 8 | 00h | Timing parameter register |
| 0x8002.C020 | PSTREC | 8 | 00h | Timing parameter register |
| 0x8002.C024 | PSTIC0 | 1 | | Test Register 0 |
| 0x8002.C024 | PSTIC1 | 8 | | Test Register 1 |
| 0x8002.C024 | PSTIC2 | 8 | | Test Register 2 |
| 0x8002.C024 | PSTIC3 | 8 | | Test Register 3 |
| 0x8002.C024 | PSTIC4 | 8 | | Test Register 4 |
| 0x8002.C024 | PSTIC5 | 8 | | Test Register 5 |
| 0x8002.C03C | PSPWDN | 1 | 00h | Power-down configuration register |

Table 10-5 PS/2 Controller Register Summary

NOTE: The initial value of registers may be not correct with the condition of testing environment. Above values are based on TIC test environment. With external model, some registers may have different value.

10.6.2.1 PSDATA

| 0x8002.C000 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-------------|-------------------------|--|---|---|---|---|---|---|
| | Transmit / Receive Data | | | | | | | |
| Bits | Type | Function | | | | | | |
| 7:0 | R/W | After wake up, PS/2 interface waits for one of two events: | | | | | | |

1. If data is written to the PSDATA register, a transmit sequence is initiated and the data is transmitted serially.
 2. If data signal is pulled low by the external devices and clock signal's negative edge is detected, a receive sequence begins and data is clocked into PSDATA register.
- At the end of transmission, transmit interrupt will occur. By reading PSSTAT status register will reveal the data is transmitted properly. Reading PSSTAT also de-asserts transmit interrupt request.
- PS/2 controller usually remains in receive data mode if no data is transmitting. The controller automatically receives data from external device and generates receive interrupt. By just reading PSDATA register the data will be acquired and the receive interrupt will be cleared.

10.6.2.2 PSSTAT

0x8002.C004

| | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--|--------|---------|--------|---------|---------|---------|----------|
| | PARITY | DATA IN | CLK IN | RX BUSY | RX FULL | TX BUSY | TX EMPTY |

| Bits | Type | Function |
|------|------|---|
| 7 | - | Reserved. Always Zero |
| 6 | R/O | The parity bit of the last received data byte |
| 5 | R/O | Double synchronized value of the current PSDAT being received/transmitted |
| 4 | R/O | Double synchronized value of the current PSCLK being received/transmitted |
| 3 | R/O | This bit indicates that the PS/2 controller is currently receiving data or not |
| 2 | R/O | This bit indicates that the a data is received and ready to be read |
| 1 | R/O | This bit indicates that the PS/2 controller is currently transmitting data or not |
| 0 | R/O | This bit indicates that the transmit register is empty and ready to transmit |

10.6.2.3 PSCONF

0x8002.C008

| | 6 | 5 | 4 | 3 | 2 | | 0 |
|--|-----|---------------|---------------|-----------|-----------|--|--------|
| | LCE | FORCE DAT LOW | FORCE CLK LOW | RX INTREN | TX INTREN | | ENABLE |

| Bits | Type | Function |
|------|------|--|
| 7 | - | Reserved |
| 6 | R/W | Line Control detection Enable bit. If set, PS/2 controller checks the line control bit from external device following by STOP bit. Otherwise PS/2 controller skips checking line control bit and proceeds to next operation. Default value is zero. Most PS/2 compatible device supports line control bit mechanism. But there are some devices that don't support line control bit. To handle such device, PS/2 controller can skip line control bit detection by resetting this bit. |
| 5 | R/W | When set, PSDAT output is forced LOW regardless of the current state of the PS/2 control logic. This mode can be used as manual communication with external device. |
| 4 | R/W | When set, PSCLK output is forced LOW regardless of the current state of the PS/2 control logic. |
| 3 | R/W | Enable receiver interrupt. To set means enable interrupt. Receiver interrupt is generated whenever PS/2 controller finishes receiving a byte data from external device. Except when transmit data, PS/2 controller goes in receive mode automatically. If receiver interrupt is disabled, PS/2 controller doesn't notify a data received. So polling PSINTR interrupt register is needed. |
| 2 | R/W | Enable transmitter interrupt. To set means enable interrupt. Transmitter interrupt is generated whenever PS/2 controller completes to transmit a byte data to external device. If transmitter interrupt is disabled then poll status register to know that the transmitting transaction is completed or poll interrupt register transmitter interrupt is generated. |
| 1 | - | Reserved |
| 0 | R/W | When reset, PS/2 controller is disabled and gets into deep sleep mode. When set, enabled. To activate PS/2 controller,, first set proper parameters of timing registers and then set this bit. As soon as this bit is enabled, PS/2 controller goes into receive mode by default. |

10.6.2.4PSINTR

0x8002.C00C

| | | | | | | | |
|--|--|--|---------------------|--------------------|-----------------|---------|---------|
| | | | 4 | 3 | 2 | 1 | 0 |
| | | | TRANSMIT TIMEOUT | RECEIVE TIMEOUT | PARITY ERROR | RX INTR | TX INTR |

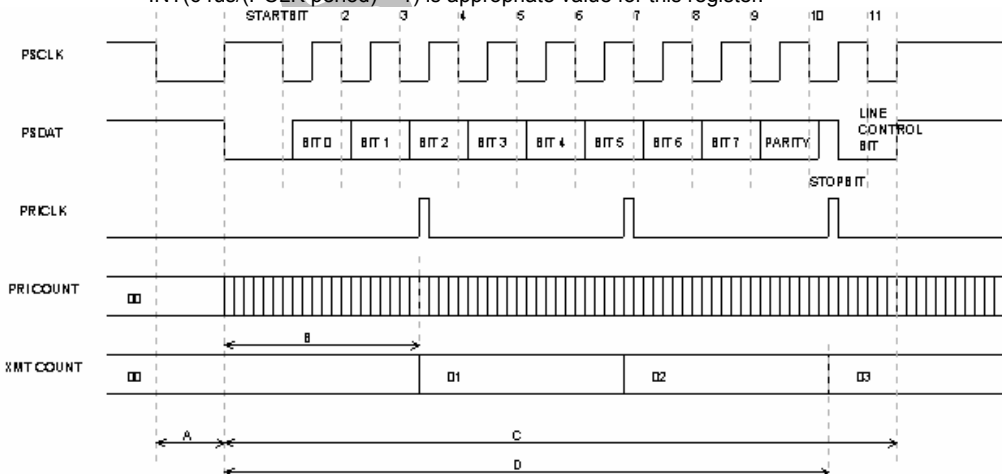
| Bits | Type | Function |
|------|------|--|
| 7:5 | - | Reserved |
| 4 | R/O | Set when PS/2 controller fails to send a complete byte data to external device in a given time. The time limit is defined in PSTXMT register. PS/2 controller doesn't try to re-transmit the data. Reset when PSSTAT register is read. |
| 3 | R/O | Set when a byte data was not constructed in a certain predefined time limit due to no more bit received or bit-rate is too slow. The time limit is defined in PSTREC register. PSDATA shows the incomplete data that has been received by that time. Reset as soon as the next byte data is arrived. |
| 2 | R/O | Set when the last received data has parity error. Cleared when the very next byte data is arrived. |
| 1 | R/O | Set when PS/2 controller receives a byte data from external device. Cleared when PSDATA register is read. When PSCONF.RXINTREN is reset, the only way to know that receiver interrupt is generated is to read this bit. |
| 0 | R/O | Set when PS/2 controller completes to transmit a byte data to external device. Cleared when PSSTAT register is read. When PSCONF.TXINTREN is reset, poll this bit to confirm that the transmission is completed. |

10.6.2.5PSTDLO

0x8002.C010

| | | | | | | | |
|--------|---|---|---|---|---|---|---|
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PSTDLO | | | | | | | |

| Bits | Type | Function |
|------|------|--|
| 7:0 | R/W | t_{PSTDLO} means the period that defines PCLK low period before initiates transmission (A in Figure 10-5 PS/2 Controller Transmitting Data Timing Diagram). Usually the value is 64us. To meet this condition, user must set this timing register properly. $INT(64us/(PCLK\ period) - 1)$ is appropriate value for this register. |



A: t_{PSTDLO} , B: t_{PSTPRI} , C: t_{XMT} , D: t_{PSTXMT}

Figure 10-5 PS/2 Controller Transmitting Data Timing Diagram

10.6.2.6 PSTPRI

0x8002.C014

| | | | | | | | |
|--------|---|---|---|---|---|---|---|
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PSTPRI | | | | | | | |

| Bits | Type | Function |
|------|------|--|
| 7:0 | R/W | Every timer in PS/2 controller is clocked by PRICLK except PRI COUNTER that generates PRICLK itself. The reason why uses PRICLK instead of PCLK is that PCLK is too fast so timing check counter requires more bits than slower clock rate. The period of PRICLK is determined by (PSTPRI+1) * that of PCLK. |

10.6.2.7 PSTXMT

0x8002.C018

| | | | | | | | |
|--------|---|---|---|---|---|---|---|
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PSTXMT | | | | | | | |

| Bits | Type | Function |
|------|------|---|
| 7:0 | R/W | This parameter determines the maximum transmission time. It is calculated as t_{PSTXMT} (D in Figure 10-5 PS/2 Controller Transmitting Data Timing Diagram) = (PSTXMT+1)* t_{PSTPRI} (B in Figure 10-5 PS/2 Controller Transmitting Data Timing Diagram). Error condition is when t_{XMT} (total transmission time, C in Figure 10-5 PS/2 Controller Transmitting Data Timing Diagram) exceeds t_{PSTXMT} . Typical value of max. t_{XMT} is 15ms. So adjust t_{PSTPRI} and t_{PSTXMT} to meet the condition. |

10.6.2.8 PSTREC

0x8002.C020

| | | | | | | | |
|--------|---|---|---|---|---|---|---|
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PSTREC | | | | | | | |

| Bits | Type | Function |
|------|------|---|
| 7:0 | R/W | This parameter determines the maximum data receiving time. It is calculated as t_{PSTREC} (B in Figure 10-6 PS/2 Controller Receiving Data Timing Diagram) = (PSTREC+1)* t_{PSTPRI} (A in Figure 10-6 PS/2 Controller Receiving Data Timing Diagram). Error condition is when t_{REC} (total receiving time, C in Figure 10-6 PS/2 Controller Receiving Data Timing Diagram) exceeds t_{PSTREC} . Typical value of max. t_{REC} is 15ms. So adjust t_{PSTPRI} and t_{PSTREC} to meet the condition. |

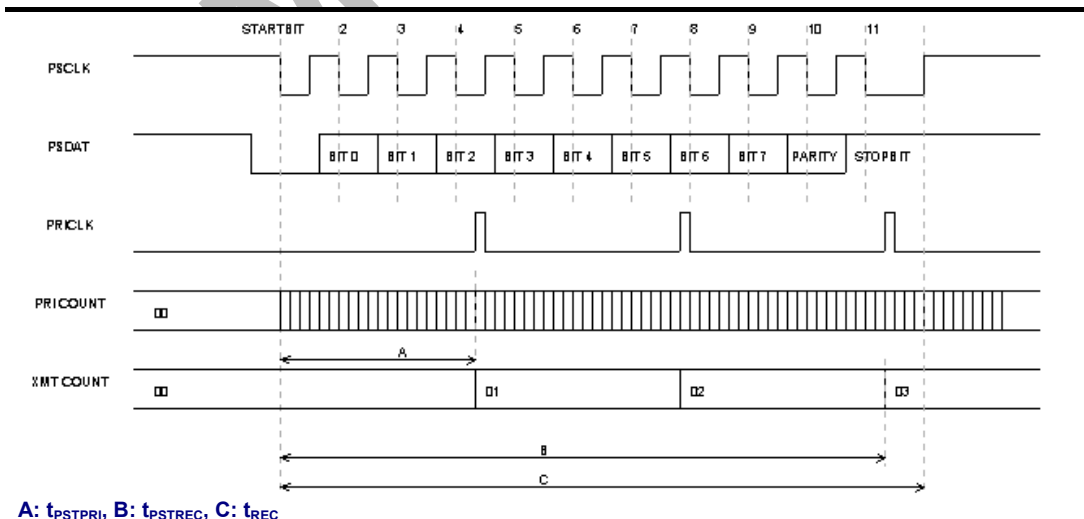


Figure 10-6 PS/2 Controller Receiving Data Timing Diagram

10.6.2.9 PSPWDN

| | | | | | | | |
|--|--|--|--|--|--|--|-------------|
| | | | | | | | 0x8002.C03C |
| | | | | | | | 0 |
| | | | | | | | PSPWDN |

| Bits | Type | Function |
|------|------|---|
| 7:1 | - | Reserved |
| 0 | R/W | Power Down disable. The initial value of power on reset is zero that means the PS/2 controller is in power down mode. To wake up PS/2 controller, set other timing registers then set this bit at last. User can put the PS/2 controller into power down mode by resetting this register at any time. |

10.6.3 Application Notes

- Use pull up resistors at the PSCLK and PSDAT pad output.
- For example, in order to set t_{PSTXMT} as 15ms, when PCLK speed is 3.6864MHz (271.3ns), see the procedure shown below.
 - i. First of all, total transmission time factor, $t_{XMT} = (PSTXMT+1) * t_{PSTPRI}$.
 - ii. So that equation is expanded as follows: $t_{XMT} = (PSTXMT+1) * \{(PSTPRI+1) * t_{PCLK}\}$.
 - iii. When t_{XMT} is 15ms and t_{PCLK} is 271.3ns, $(PSTXMT+1) * \{(PSTPRI+1) * 271.3ns\}$ is 55288.
 - iv. Due to both PSTXMT and PSTPRI is only 8-bit register, the values of these two register can hold only up to 256. So if we set (PSTPRI+1) to 256 then (PSTXMT+1) will be 216.
 - v. $PSTPRI = 255_{10} = FF_{16}$
 - vi. $PSTXMT = 215_{10} = D7_{16}$
- You can use the same flow to calculate t_{PSTREC} . Basically as the root, t_{PSTPRI} , is common with t_{PSTXMT} , the only factor you have to calculate is just PSTREC.

10.7 RTC

This module is a 32-bit counter clocked by a 32768Hz clock. This clock needs to be provided by the system, as there is no crystal inside the block. It also contains a 32-bit match register that can be programmed to generate an interrupt signal when the time in the RTC matches the specific value written to this register (alarm function - RTC event). The RTC has two event outputs, one which is synchronized to PCLK, RTCIRQ, and the second, URTCEV synchronized to the 32768Hz clock. RTCIRQ is connected to the system interrupt controller, and URTCEV is used by the PMU to provide a system alarm Wake up.

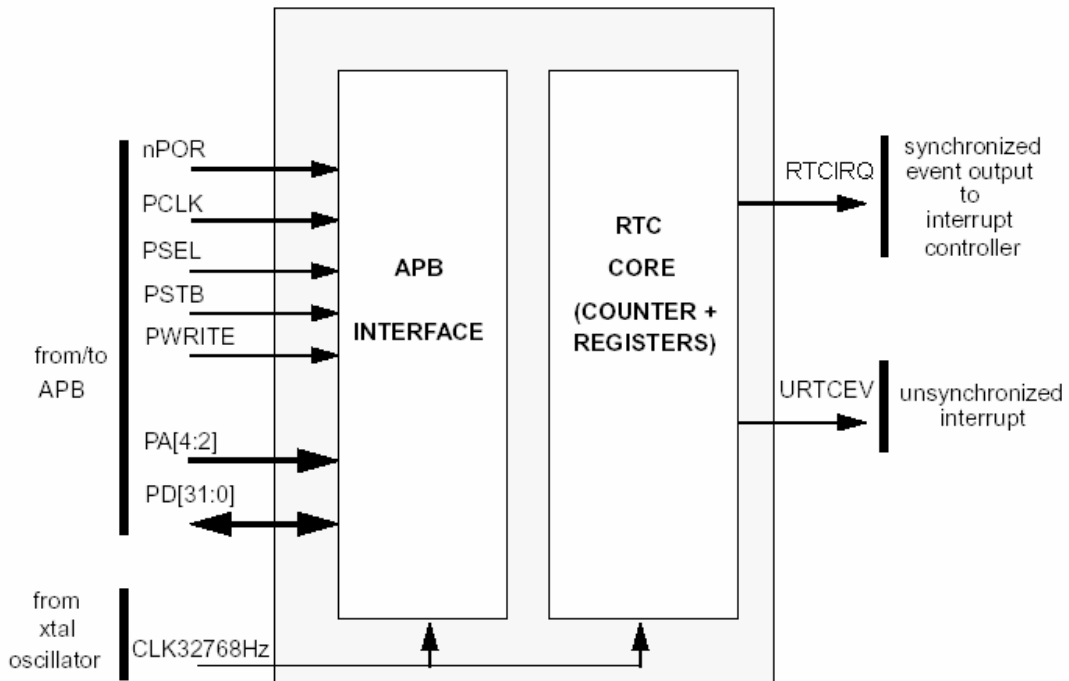


Figure 10-7 RTC Connection

As shown in Fig. 10-3, RTC module is connected to the APB. APB signals are refer to AMBA APB spec, and following table shows the non-AMBA signals from the RTC core block. The following table shows non-AMBA signals within RTC core block for more information about APB signals refer to the AMBA APB spec.

| NAME | Source/Destination | Description |
|----------|---------------------------|---|
| CLK32KHZ | Clock generator | 32768HZ clock input. This is the signal that clocks the counter during normal operation. |
| RTCIRQ | APB(Interrupt controller) | Interrupt signal to the interrupt module. When HIGH, this signal indicates a valid comparison between the counter value and the match register. It also indicates 1HZ interval with enable bit in control register. |
| URTCEV | ASB(PMU) | When HIGH, this signal indicates a valid comparison between the counter value and the match register. This signal is used to wake up the HMS30C7202 when it is in deep sleep mode. |

Table 10-6 Non-AMBA Signals within RTC Core Block

FEATURES

- Two type of Alarm function

10.7.1 External Signals

| Pin Name | Type | Description |
|-----------|------|----------------------------------|
| RTCOSCIN | I | RTC oscillator input. 32.768KHz |
| RTCOSCOUT | O | RTC oscillator output. 32.768KHz |

10.7.2 Functional Description

The counter is loaded by writing to the RTC data register. The counter will count up on each rising edge of the 1Hz clock and loops back to 0 when the maximum value(0xFFFFFFFF) is reached. At any moment the counter value can be obtained by reading the RTC data register.

The value of the match register can also be read at any time, and the read does not affect the counter value. The status of the interrupt signal is available in the status register. The status bit is set if a comparator match event has occurred or 1 second has elapsed. Reading from the status register will clear the status register.

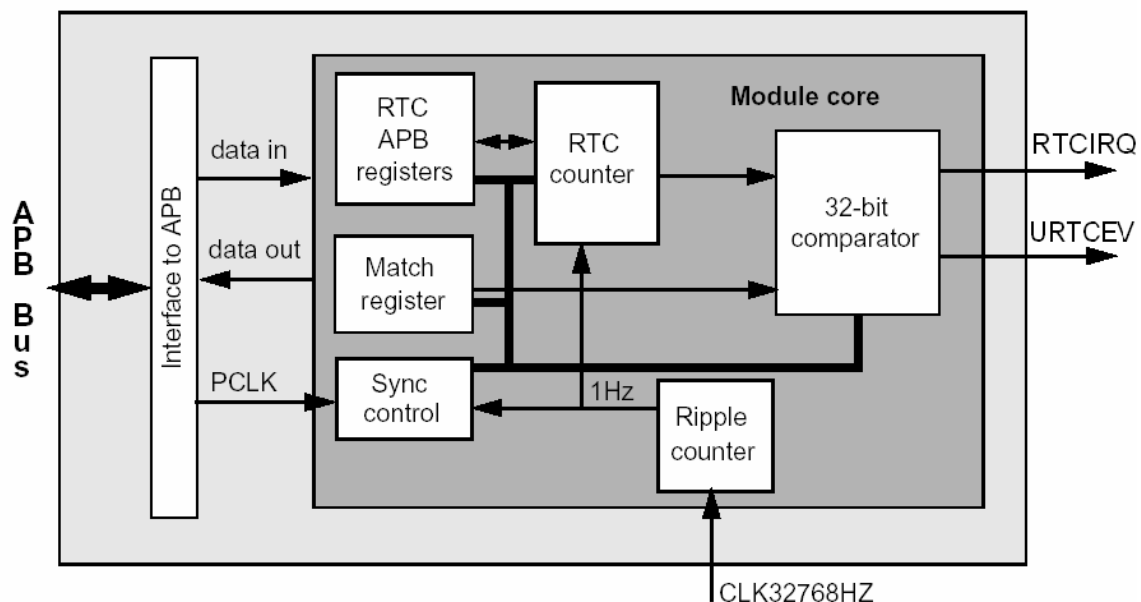


Figure 10-8 RTC Block Diagram

10.7.3 Registers

| Address | Name | Width | Default | Description |
|-------------|-------|-------|---------|----------------------|
| 0x8002.8000 | RTCDR | 32 | 0x0 | RTC Data Register |
| 0x8002.8004 | RTCMR | 32 | 0xF | RTC Match Register |
| 0x8002.8008 | RTCSR | 2 | 0x0 | RTC Status Register |
| 0x8002.8010 | RTCCR | 2 | 0x0 | RTC Control Register |

Table 10-7 RTC Register Summary

10.7.3.1 RTC Data Register (RTCDR)

| Bits | Type | Function |
|------|------|--|
| 31:0 | R/W | RTC Data register. Writing to this 32-bit register will load the counter. A read will give the |

current value of the counter. The counter is loaded by writing to the RTC data register. The counter will count up on each rising edge of the clock and loops back to 0 when the maximum value (0xFFFFFFFF) is reached. At any moment the counter value can be obtained by reading the RTC data register.

10.7.3.2 RTC Match Register (RTCMR)

0x8002.8004

| | | | | | | | | | | | | | | | |
|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RTCMR [31:16] | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RTCMR [15:0] | | | | | | | | | | | | | | | |

| Bits | Type | Function |
|------|------|---|
| 31:0 | R/W | RTC Match register. If this register's value is matched with current counter, an interrupt will be generated to implement alarm function. Writing to this 32-bit register will load the match register. This value can also be read back. |

10.7.3.3 RTC Status Register (RTCS)

0x8002.8008

| | | | | | | | | | | | | | |
|--|--|--|--|--|--|--|--|--|--|--|--|---------------|---------------|
| | | | | | | | | | | | | 1 | 0 |
| | | | | | | | | | | | | MATCH FLAG | 1 SEC FLAG |

| Bits | Type | Function |
|------|------|---|
| 7:2 | - | Reserved |
| 1 | R | Match event interrupt flag is set if the counter value equals to the content of match register, RTCMR. Reading from the status register will clear the status register. |
| 0 | R | When performing a read from this register the interrupt flag will be cleared. If 1 second has elapsed, this bit will be set. |

10.7.3.4 RTC Control Register (RTCCR)

0x8002.8010

| | | | | | | | | | | | | | |
|--|--|--|--|--|--|--|--|--|--|--|--|------------------|------------------|
| | | | | | | | | | | | | 1 | 0 |
| | | | | | | | | | | | | MATCH INTR EN | 1 SEC INTR EN |

| Bits | Type | Function |
|------|------|--|
| 7:2 | - | Reserved |
| 1 | R/W | Set this bit enables match event interrupt. |
| 0 | R/W | Set this bit enables 1 second event interrupt. |

10.8 TIMER

This module is a 32-bit counter clocked by a 3.6864MHz clock.

Timer is an AMBA slave module that connects to the Advanced Peripheral Bus (APB). For more information about AMBA, please refer to the AMBA Specification (ARM IHI 0001).

FEATURES

- 32-bit up ripple counter
- Auto repeat mode
- Count enable/disable
- Interrupt enable/disable
- 3-timer channel

10.8.1 External Signals

| Pin Name | Type | Description |
|-----------|------|-----------------------------|
| PWM [1:0] | O | PWM Output |
| TimerOut | O | Timer 1 output divided by 2 |

10.8.2 Registers

| Address | Name | Width | Default | Description |
|-------------|----------|-------|------------|---|
| 0x8002.5000 | T0BASE | 32 | 0xFFFFFFFF | Timer0 Base Register |
| 0x8002.5008 | T0COUNT | 32 | 0x0 | Timer0 Counter Register |
| 0x8002.5010 | T0CTRL | 3 | 0x0 | Timer0 Control Register |
| 0x8002.5020 | T1BASE | 32 | 0xFFFFFFFF | Timer1 Base Register |
| 0x8002.5028 | T1COUNT | 32 | 0x0 | Timer1 Counter Register |
| 0x8002.5030 | T1CTRL | 3 | 0x00 | Timer1 Control Register |
| 0x8002.5040 | T2BASE | 32 | 0xFFFFFFFF | Timer2 Base Register |
| 0x8002.5048 | T2COUNT | 32 | 0x0 | Timer2 Counter Register |
| 0x8002.5050 | T2CTRL | 3 | 0x0 | Timer2 Control Register |
| 0x8002.5060 | TOPCTRL | 32 | 0x9 | Top-level Control Register |
| 0x8002.5064 | TOPSTAT | 3 | 0x0 | Top-level Status Register |
| 0x8002.5080 | T64LOW | 32 | 0x0 | Lower 32-bit of 64-bit counter (Timer3) |
| 0x8002.5084 | T64HIGH | 32 | 0x0 | Upper 32-bit of 64-bit counter (Timer3) |
| 0x8002.5088 | T64CTRL | 2 | 0x0 | 64-bit Timer Control Register (Timer3) |
| 0x8002.508C | T64TR | 15 | 0x0 | 64-bit Timer Test Register (Timer3) |
| 0x8002.5094 | T64LBase | 32 | 0xFFFFFFFF | 64-bit Timer Lower Base (Timer3) |
| 0x8002.5098 | T64HBase | 32 | 0xFFFFFFFF | 64-bit Timer Higher Base (Timer3) |
| 0x8002.50A0 | P0COUNT | 16 | 0x0 | PWM channel 0 count register |
| 0x8002.50A4 | P0WIDTH | 16 | 0xFFFF | PWM channel 0 width register |
| 0x8002.50A8 | P0PERIOD | 16 | 0xFFFF | PWM channel 0 period register |
| 0x8002.50AC | P0CTRL | 5 | 0x0 | PWM channel 0 control register |
| 0x8002.50B0 | P0PWMTR | 4 | 0x0 | PWM channel 0 test register |
| 0x8002.50C0 | P1COUNT | 16 | 0x0 | PWM channel 1 count register |
| 0x8002.50C4 | P1WIDTH | 16 | 0xFFFF | PWM channel 1 width register |
| 0x8002.50C8 | P1PERIOD | 16 | 0xFFFF | PWM channel 1 period register |
| 0x8002.50CC | P1CTRL | 5 | 0x0 | PWM channel 1 control register |
| 0x8002.50D0 | P1PWMTR | 4 | 0x0 | PWM channel 1 test register |

Table 10-8 Timer Register Summary

10.8.2.1 Timer [0,1,2] Base Register (T[0,1,2]BASE)

| | | | | | | | | | | | | | | | |
|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x8002.5000 / 0x8002.5020 / 0x8002.5040 | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |

| T[0,1,2]BASE [31:16] | | | | | | | | | | | | | | | |
|----------------------|------|--|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| T[0,1,2]BASE [15:0] | | | | | | | | | | | | | | | |
| Bits | Type | Function | | | | | | | | | | | | | |
| 31:0 | R/W | Timer 0 (Timer 1, Timer 2) Base Register. 32-bit target count value (interval) is stored in here. The interrupt interval in repeat mode is (Base Register value + 1) clock periods. For example, if the Base Register is set to 0x3333, then the timer generates an interrupt request every 0x3333 + 1 clock cycles. | | | | | | | | | | | | | |

10.8.2.2 Timer [0,1,2] Count Register (T[0,1,2]COUNT)

| | | | | | | | | | | | | | | | 0x8002.5008 / 0x8002.5028 / 0x8002.5048 | | | |
|-----------------------|------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|---|--|--|--|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | | | |
| T[0,1,2]COUNT [31:16] | | | | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | | |
| T[0,1,2]COUNT [15:0] | | | | | | | | | | | | | | | | | | |
| Bits | Type | Function | | | | | | | | | | | | | | | | |
| 31:0 | R/W | 32bit up counter | | | | | | | | | | | | | | | | |

10.8.2.3 Timer [0,1,2] Control Register (T[0,1,2]CTRL)

| | | | | | | | 2 | 1 | 0 | 0x8002.5010 / 0x8002.5030 / 0x8002.5050 | | | |
|------|------|---|--|--|--|--|-------|-------------|--------------|---|--|--|--|
| | | | | | | | RESET | REPEAT MODE | COUNT ENABLE | | | | |
| Bits | Type | Function | | | | | | | | | | | |
| 7:3 | - | Reserved | | | | | | | | | | | |
| 2 | R/W | Set for reset counter register | | | | | | | | | | | |
| 1 | R/W | Set for count repeat mode | | | | | | | | | | | |
| 0 | R/W | Set to start count and reset to stop. For Timer 0, Timer 1, and Timer 2 in non-repeat mode, This bit will be cleared automatically whenever the counter reaches the target value. | | | | | | | | | | | |

10.8.2.4 Timer Top-level Control Register (TOPCTRL)

| | | | | | | | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 0x8002.5060 | |
|------|------|---|--|--|--|--|--------------|------------------|-----------------|------------|-----------------|-----------------|-----------------|-------------|--|
| | | | | | | | TIMER OUT EN | TIMER 64 INTR EN | TIMER 64 ENABLE | POWER DOWN | TIMER 2 INTR EN | TIMER 1 INTR EN | TIMER 0 INTR EN | | |
| Bits | Type | Function | | | | | | | | | | | | | |
| 7 | - | Reserved | | | | | | | | | | | | | |
| 6 | R/W | Timer 1 Output Enable. The interval of this output is 2 times of interrupt interval of Timer 1. 0 = disable, 1 = enable | | | | | | | | | | | | | |
| 5 | R/W | 64bit Timer Counter Overflow Interrupt Enable 0 = disable, 1 = enable | | | | | | | | | | | | | |
| 4 | R/W | 64bit Timer Enable. 0 = disable, 1 = enable | | | | | | | | | | | | | |
| 3 | R/W | Timer Controller POWER DOWN. 0 = Power Down mode, 1 = enable | | | | | | | | | | | | | |
| 2 | R/W | Timer 2 Interrupt Enable 0 = disable, 1 = enable | | | | | | | | | | | | | |
| 1 | R/W | Timer 1 Interrupt Enable 0 = disable, 1 = enable | | | | | | | | | | | | | |
| 0 | R/W | Timer 0 Interrupt Enable. If reset, no interrupt is generated at Timer 0. 0 = disable, 1 = enable | | | | | | | | | | | | | |

10.8.2.5 Timer Status Register (TOPSTAT)

0x8002.5064

| | | | | | | | |
|--|--|--|--|---------------|--------------|--------------|--------------|
| | | | | 3 | 2 | 1 | 0 |
| | | | | TIMER 64 INTR | TIMER 2 INTR | TIMER 1 INTR | TIMER 0 INTR |

| Bits | Type | Function |
|------|------|--------------------------------|
| 7:4 | - | Reserved |
| 3 | R | Timer 64 Interrupt Status Flag |
| 2 | R | Timer 2 Interrupt Status Flag |
| 1 | R | Timer 1 Interrupt Status Flag |
| 0 | R | Timer 0 Interrupt Status Flag |

10.8.2.6 Timer Lower 32-bit Count Register of 64-bit Counter (T64LOW)

0x8002.5080

| | | | | | | | | | | | | | | | |
|----------------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| T64LOW [31:16] | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| T64LOW [15:0] | | | | | | | | | | | | | | | |

| Bits | Type | Function |
|------|------|---|
| 31:0 | R/W | Lower 32bit count value of 64bit Timer (Timer3) |

10.8.2.7 Timer Upper 32-bit Count Register of 64-bit Counter (T64HIGH)

0x8002.5084

| | | | | | | | | | | | | | | | |
|-----------------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| T64HIGH [31:16] | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| T64HIGH [15:0] | | | | | | | | | | | | | | | |

| Bits | Type | Function |
|------|------|---|
| 31:0 | R/W | Upper 32bit count value of 64bit Timer (Timer3) |

10.8.2.8 Timer 64-bit Counter Control Register (T64CTRL)

0x8002.5088

| | | | | | | | | | | | | | | | | | | |
|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|----------|----------|--------------|
| | | | | | | | | | | | | | | | | 2 | 1 | 0 |
| | | | | | | | | | | | | | | | | RESET | | COUNT ENABLE |

| Bits | Type | Function |
|------|------|---|
| 7:3 | - | Reserved |
| 2 | R/W | Reset Timer 64 (Timer3). 0 = Keep Counting, 1 = Reset the counter register |
| 1 | | Reserved |
| 0 | R/W | Timer 64 (Timer3)Enable. 0 = Stop Counter, 1 = Start Counter |

10.8.2.9 Timer 64-bit Counter Test Register (T64TR)

0x8002.508C

| | | | | | | | | |
|----------|----------|-----------|-----------|-----------|-----------|-----------|----------|----------|
| | | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| | | Creg59 | Creg55 | Creg51 | Creg47 | Creg43 | Creg39 | Creg35 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| Creg31 | Creg27 | Creg23 | CReg19 | CReg15 | CReg11 | CReg7 | CReg3 | |

| Bits | Type | Function |
|------|------|----------|
|------|------|----------|

| | | |
|----|---|--|
| 14 | W | When TestReg[59] is HIGH, output is the same as CountCLK inversion. When TestReg[59] is LOW, output is the same as CountReg[59] |
| 13 | W | When TestReg[55] is HIGH, output is the same as CountCLK inversion. When TestReg[55] is LOW, output is the same as CountReg[55] |
| 12 | W | When TestReg[51] is HIGH, output is the same as CountCLK inversion. When TestReg[51] is LOW, output is the same as CountReg[51] |
| 11 | W | When TestReg[47] is HIGH, output is the same as CountCLK inversion. When TestReg[47] is LOW, output is the same as CountReg[47] |
| 10 | W | When TestReg[43] is HIGH, output is the same as CountCLK inversion. When TestReg[43] is LOW, output is the same as CountReg[43] |
| 9 | W | When TestReg[39] is HIGH, output is the same as CountCLK inversion. When TestReg[39] is LOW, output is the same as CountReg[39] |
| 8 | W | When TestReg[35] is HIGH, output is the same as CountCLK inversion. When TestReg[35] is LOW, output is the same as CountReg[35] |
| 7 | W | When TestReg[31] is HIGH, output is the same as CountCLK inversion. When TestReg[31] is LOW, output is the same as CountReg[31] |
| 6 | W | When TestReg[27] is HIGH, output is the same as CountCLK inversion. When TestReg[27] is LOW, output is the same as CountReg[27] |
| 5 | W | When TestReg[23] is HIGH, output is the same as CountCLK inversion. When TestReg[23] is LOW, output is the same as CountReg[23] |
| 4 | W | When TestReg[19] is HIGH, output is the same as CountCLK inversion. When TestReg[19] is LOW, output is the same as CountReg[19] |
| 3 | W | When TestReg[15] is HIGH, output is the same as CountCLK inversion. When TestReg[15] is LOW, output is the same as CountReg[15] |
| 2 | W | When TestReg[11] is HIGH, output is the same as CountCLK inversion. When TestReg[11] is LOW, output is the same as CountReg[11] |
| 1 | W | When TestReg[7] is HIGH, output is the same as CountCLK inversion. When TestReg[7] is LOW, output is the same as CountReg[7] |
| 0 | W | When TestReg[3] is HIGH, output is the same as CountCLK inversion. When TestReg[3] is LOW, output is the same as CountReg[3] |

10.8.2.10 Timer Lower 32-bit Base Register of 64-bit Counter (T64LBASE)

0x8002.5094

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|------------------|------|--|----|----|----|----|----|----|----|----|----|----|----|----|----|
| T64LBASE [31:16] | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| T64LBASE [15:0] | | | | | | | | | | | | | | | |
| Bits | Type | Function | | | | | | | | | | | | | |
| 31:0 | R/W | Lower 32bit base value of 64bit Timer (Timer3) | | | | | | | | | | | | | |

10.8.2.11 Timer Upper 32-bit Base Register of 64-bit Counter (T64HBASE)

0x8002.5098

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|------------------|------|--|----|----|----|----|----|----|----|----|----|----|----|----|----|
| T64HBASE [31:16] | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| T64HBASE [15:0] | | | | | | | | | | | | | | | |
| Bits | Type | Function | | | | | | | | | | | | | |
| 31:0 | R/W | Upper 32bit base value of 64bit Timer (Timer3) | | | | | | | | | | | | | |

10.8.2.12 PWM Channel [0,1] Count Register (P[0,1]COUNT)

0x8002.50A0 / 0x8002.50C0

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-------------|------|--------------------------|----|----|----|---|---|---|---|---|---|---|---|---|---|
| P[0,1]COUNT | | | | | | | | | | | | | | | |
| Bits | Type | Function | | | | | | | | | | | | | |
| 15:0 | R | PWM [0,1] Count Register | | | | | | | | | | | | | |

10.8.2.13 PWM Channel [0,1] Width Register (P[0,1]WIDTH)

0x8002.50A4 / 0x8002.50C4

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-------------|------|---|----|----|----|---|---|---|---|---|---|---|---|---|---|
| P[0,1]WIDTH | | | | | | | | | | | | | | | |
| Bits | Type | Function | | | | | | | | | | | | | |
| 15:0 | R/W | PWM [0,1] Width Register. Actual width of output is (P[0,1]WIDTH + 1) / PCLK. | | | | | | | | | | | | | |

10.8.2.14 PWM Channel [0,1] Period Register (P[0,1]PERIOD)

0x8002.50A8 / 0x8002.50C8

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------------|------|--|----|----|----|---|---|---|---|---|---|---|---|---|---|
| P[0,1]PERIOD | | | | | | | | | | | | | | | |
| Bits | Type | Function | | | | | | | | | | | | | |
| 15:0 | R/W | PWM [0,1] Period Register. Actual Period of output is (P[0,1]PERIOD + 1) / PCLK. | | | | | | | | | | | | | |

10.8.2.15 PWM Channel [0,1] Control Register (P[0,1]CTRL)

0x8002.50AC / 0x8002.50CC

| | | | 4 | 3 | 2 | 1 | 0 |
|------|------|--|---------|---------------|---------------|-------|-----------------|
| | | | CLK SEL | OUTPUT INVERT | OUTPUT ENABLE | RESET | PWM[0,1] ENABLE |
| Bits | Type | Function | | | | | |
| 7:5 | - | Reserved | | | | | |
| 4 | R/W | PWM [0,1] Source Clock Selection(PCLK) 0 = 3.6864MHz, 1 = 1.8432MHz | | | | | |
| 3 | R/W | PWM [0,1] Output Waveform Inverting 0 = non inverting, 1 = inverting | | | | | |
| 2 | R/W | PWM [0,1] Output Enable 0 = disable output driver, 1 = enable output driver | | | | | |
| 1 | R/W | PWM [0,1] Counter Reset 0 = keep count, 1 = reset counter register | | | | | |
| 0 | R/W | PWM [0,1] Counter Enable. 0 = stop counter, 1 = start counter | | | | | |

10.8.2.16 PWM Channel[0,1] Test Register(P[0,1]PWMTR)

0x8002.50B0 / 0x8002.50D0

| | | | 3 | 2 | 1 | 0 |
|------|------|--|----------|--------|-------|-------|
| | | | Reserved | Creg11 | Creg7 | Creg3 |
| Bits | Type | Function | | | | |
| 3 | | Reserved | | | | |
| 2 | W | When TestReg[11] is HIGH, output is the same as CountCLK inversion. When TestReg[11] is LOW, output is the same as CountReg[11] | | | | |
| 1 | W | When TestReg[7] is HIGH, output is the same as CountCLK inversion. When TestReg[7] is LOW, output is the same as CountReg[7] | | | | |
| 0 | W | When TestReg[3] is HIGH, output is the same as CountCLK inversion. When TestReg[3] is LOW, output is the same as CountReg[3] | | | | |

10.9 UART/SIR

The 16C550 is a Universal Asynchronous Receiver/Transmitter (UART), with FIFOs, and is functionally identical to the 16C450 on power-up (CHARACTER mode). The 16550 can be put into an alternate mode (FIFO mode) to relieve the CPU of excessive software overhead. In this mode internal FIFOs are activated, allowing 16 bytes plus 3 bit of error data per byte in the RCVR FIFO, to be stored in both receive and transmit modes. All the logic is on the chip to minimize the system overhead and to maximize efficiency.

The UART performs serial-to-parallel conversion on data characters received from a peripheral device or a MODEM, and parallel-to-serial conversion on data characters received from the CPU. The CPU can read the complete status of the UART at any time during the functional operation. Status information reported includes the type and condition of the transfer operations being performed by the UART, as well as any error conditions (parity, overrun, framing, or break interrupt).

The UART includes a programmable baud rate generator capable of dividing the timing reference clock input by divisors of 1 to $2^{16}-1$, and producing a 16x clock for driving the internal transmitter logic. Provisions are also included to use this 16x clock to drive the receiver logic.

The UART has complete MODEM-control capability, and a processor-interrupt system. Interrupts can be programmed to the user's requirements, minimizing the computing required to handle the communications link.

FEATURES

- Capable of running all existing 16C450 software.
- After reset, all registers are identical to the 16C450 register set.
- The FIFO mode transmitter and receiver are each buffered with 16 byte FIFOs to reduce the number of interrupts presented to the CPU.
- Add or delete standard asynchronous communication bits (start, stop and parity) to or from the serial data.
- Holding and shift registers in the 16C450 mode eliminate the need for precise synchronization between the CPU and serial data.
- Independently controlled transmit, receive, line status and data set interrupts.
- Programmable baud generator divides any input clock by 1 to 65535 and generates 16x clock
- Independent receiver clock input.
- MODEM control functions (CTS, RTS, DSR, DTR, RI and DCD).
- Fully programmable serial-interface characteristics:
 - 5-, 6-, 7- or 8-bit characters
 - Even, odd or no-parity bit generation and detection
 - 1-, 1.5- or 2-stop bit generation and detection
 - Baud generation (DC to 230k baud)
- False start bit detection.
- Complete status-reporting capabilities.
- Line breaks generation and detection.
- Internal diagnostic capabilities:
 - Loopback controls for communications link fault isolation
- Full prioritized interrupt system controls.

10.9.1 External Signals

| Pin Name | Type | Description |
|----------|------|--|
| nURING | I | <p>UART 0 ring input signal (wake-up signal to PMU).</p> <p>When LOW, this indicates that the MODEM or data set has received a telephone ring signal. The nURING signal is a MODEM status input whose condition can be tested by the CPU reading bit 6 (RI) of the MODEM Status Register. Bit 6 is the complement of the nURING signal. Bit 2 (TERI) of the MODEM Status Register indicates whether the nURING input signal has changed from a LOW to a HIGH state since the previous reading of the MODEM Status Register.</p> <p>Note: Whenever the RI bit of the MODEM Status Register changes from a HIGH to a LOW state, an interrupt is generated if the MODEM Status Interrupt is enabled. The nURING input from the external PAD is not provided. To use this signal, you should set up the UART control register of the AFE interface. For further information, refer to 13.9 Analog Front End, AFE (CODEC Interface) on</p> |

| | | page 13-56. |
|-----------|---|--|
| nUDTR | O | UART 0 data terminal ready. When LOW, this informs the MODEM or data set that the UART is ready to establish communication link. The nUDTR output signal can be set to an active LOW by programming bit 0 (DTR) of the MODEM Control Register to HIGH level. A Master Reset operation sets this signal to its inactive (HIGH) state. Loop mode operation holds this signal in its inactive state. |
| nUCTS | I | UART 0 clear to send input. When LOW, this indicates that the MODEM or data set is ready to exchange data. The nUCTS signal is a MODEM status input whose conditions can be tested by the CPU reading bit 4 (CTS) of the MODEM Status Register indicates whether the nUCTS input has changed state since the previous reading of the MODEM Status Register. nUCTS has no effect on the Transmitter. Note: Whenever the CTS bit of the MODEM Status Register changes its state, an interrupt is generated if the MODEM Status Interrupt is enabled. |
| nURTS | O | UART 0 request to send. When LOW, this informs the MODEM or data set that the UART is ready to exchange data. The nURTS output signal can be set to an active LOW by programming bit 1 (RTS) of the MODEM Control Register. A Master Reset operation sets this signal to its inactive (HIGH) state. Loop mode operation holds this signal in its inactive state. |
| nUDSR | I | UART 0 data set ready input. When LOW, this indicates that the MODEM or data set is ready to establish the communications link with the UART. The nUDSR signal is a MODEM status input whose conditions can be tested by the CPU reading bit 5 (DSR) of the MODEM Status Register. Bit 5 is the complement of the nUDSR signal. Bit 1(DDSR) of MODEM Status Register indicates whether the nUDSR input has changed state since the previous reading of the MODEM status register. Note: Whenever the DSR bit of the MODEM Status Register changes its state, an interrupt is generated if the MODEM Status Interrupt is enabled. |
| nUDCD | I | UART 0 data carrier detect input. When LOW, indicates that the data carrier has been detected by the MODEM data set. The signal is a MODEM status input whose condition can be tested by the CPU reading bit 7 (DCD) of the MODEM Status Register. Bit 7 is the complement of the signal. Bit 3 (DDCD) of the MODEM Status Register indicates whether the input has changed state since the previous reading of the MODEM Status Register. nUDCD has no effect on the receiver. Note: Whenever the DCD bit of the MODEM Status Register changes its state, an interrupt is generated if the MODEM Status Interrupt is enabled. |
| USIN [0] | I | UART 0 serial data inputs. Serial data input from the communications link (peripheral device, MODEM or data set). |
| USOUT [0] | O | UART 0 serial data outputs. Composite serial data output to the communications link (peripheral, MODEM or data set). The USOUT signal is set to the Marking (logic 1) state upon a Master Reset operation. |
| USIN [1] | I | UART 1 serial data inputs |
| USOUT [1] | O | UART 1 serial data outputs |
| USIN [2] | I | UART 2 serial data inputs (muxed with KSCAN05) |
| USOUT [2] | O | UART 2 serial data outputs (muxed with KSCAN06) |
| USIN [3] | I | UART 3 serial data inputs (muxed with KSCAN15) |
| USOUT [3] | O | UART 3 serial data outputs (muxed with KSCAN16) |

10.9.2 Registers

| Address | Name | Width | Default | Description |
|-------------|--------|-------|---------|--|
| 0x8002.0000 | U0Base | - | - | UART 0 Base |
| 0x8002.1000 | U1Base | - | - | UART 1 Base |
| 0x8002.D000 | U2Base | - | - | UART 2 Base |
| 0x8002.E000 | U3Base | - | - | UART 3 Base |
| UxBase+0x00 | RBR | 8 | 0x0 | Receiver Buffer Register (DLAB = 0, Read) |
| | THR | | | Transmitter Holding Register (DLAB = 0, Write) |

| | | | | |
|-------------|--------|-----------|------|--|
| | DLL | | | Divisor Latch Least Significant Byte (DLAB = 1) |
| UxBase+0x04 | IER | 8 | 0x0 | Interrupt Enable Register (DLAB = 0) |
| | DLM | | | Divisor Latch Most Significant Byte (DLAB = 1) |
| UxBase+0x08 | IIR | 8 | 0x1 | Interrupt Identification Register (Read) |
| | FCR | | 0x0 | FIFO Control Register (Write) |
| UxBase+0x0C | LCR | 8 | 0x0 | Line Control Register |
| UxBase+0x10 | MCR | 3 | 0x0 | Modem Control Register |
| UxBase+0x14 | LSR | 8 | 0x60 | Line Status Register |
| UxBase+0x18 | MSR | 8 | 0xX0 | Modem Status Register |
| UxBase+0x1C | SCR | 8 | 0x0 | Scratch Register |
| UxBase+0x30 | UartEN | 1 or 4 | 0x0 | UART Enable Register In Uart 1, this bit width is 4 (support SIR) |

Table 10-9 UART/SIR Register Summary

10.9.2.1 RBR/THR/DLL

| | | | | | | | UxBase+0x00 |
|--|------|--|---|---|---|---|-------------|
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Data Bit 7 ~ Data Bit 0 (RBR, THR; DLAB = 0) | | | | | | | |
| Bit 7 ~ Bit 0 (DLL; DLAB = 1) | | | | | | | |
| Bits | Type | Function | | | | | |
| 7:0 | R/W | When DLAB = 0, read this register represents RBR while writes does THR. When DLAB = 1, DLL will be read or written. | | | | | |

10.9.2.2 IER/DLM

This register enables the five types of UART interrupts. Each interrupt can individually activate the interrupt (INTUART) output signal. It is possible to totally disable the interrupt Enable Register (IER). Similarly, setting bits of the IER register to logic 1 enables the selected interrupt(s). Disabling an interrupt prevents it from being indicated as active in the IIR and from activating the INTUART output signal. All other system functions operate in their normal manner, including the setting of the Line Status and MODEM Status Registers. Table 13-6: Summary of registers on page 13-10 shows the contents of the IER. Details on each bit follow.

| | | | | | | | UxBase+0x04 |
|-------------------------------|------|---|---|---------|---------|---------------|---------------|
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0 | 0 | 0 | 0 | MS INTR | LS INTR | TX EMPTY INTR | DATA RDY INTR |
| Bit 7 ~ Bit 0 DLM; (DLAB = 1) | | | | | | | |
| Bits | Type | Function | | | | | |
| | | | | | | | DLM |
| 7 | R/W | 0 | | | | | |
| 6 | R/W | 0 | | | | | |
| 5 | R/W | 0 | | | | | |
| 4 | R/W | 0 | | | | | |
| 3 | R/W | Enables the MODEM Status Interrupt when set to logic 1. | | | | | |
| 2 | R/W | Enables the Receiver Line Status Interrupt when set to logic 1. | | | | | |
| 1 | R/W | Enables the Transmitter Holding Register Empty Interrupt when set to logic 1. | | | | | |
| 0 | R/W | Enables the Received Data Available Interrupt (and time-out interrupts in the FIFO mode) when set to logic 1. | | | | | |

10.9.2.3 IIR/FCR

| | | | | | | | |
|---|---|---|---|---|---|---|-------------|
| | | | | | | | UxBase+0x08 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

| | | | | | | |
|-----------------|---|---|---------|------------|------------|--------------|
| FIFO EN | 0 | 0 | INTR ID | | | INTR PENDING |
| RCVR TRIG LEVEL | - | - | - | XMIT RESET | RCVR RESET | FIFO EN |

Interrupt Identification Register

In order to provide minimum software overhead during data character transfers, the UART prioritizes interrupts into four levels and records these in the Interrupt Identification Register. The four levels of interrupt conditions are, in order of priority

1. Receiver Line Status
2. Received Data Ready
3. Transmitter Holding Register Empty
4. MODEM Status

When the CPU accesses the IIR, the UART freezes all interrupts and indicates the highest priority pending interrupt to the CPU. While this CPU access is occurring, the UART records new interrupts, but does not change its current indication until the access is complete.

| Bits | Type | Function |
|------|------|---|
| 7:6 | R | These two bits are set when FCR [0] = 1. |
| 5:4 | R | These two bits of the IIR are always logic 0 |
| 3:1 | R | These two bits of the IIR are used to identify the highest priority interrupt pending. In the 16C450 mode, IIR [3] is 0. In the FIFO mode, IIR [3] is set along with IIR [2] when a time-out interrupt is pending |

IIR [3:1] Interrupt Set and Reset Function

Priority Level
Interrupt Type
Interrupt Source
Interrupt Reset Control

000

-

None

None

-

011

Highest

Receiver Line Status

Overrun Error or Parity Error or Framing Error or Break Interrupt

Reading the Line Status Register

010

Second

Receiver Data Available

Receiver Data Available or Trigger Level Reached

Reading the Receiver Buffer Register or the FIFO drops below the trigger level

110

Second

Character Time-out Indication

No Characters have been removed from or input to the RCVR FIFO during the last 4 Character times and there is at least 1 Character in it during this time

Reading the Receiver Buffer Register

| | | |
|---|---|--|
| | | 001 Third Transmitter Holding Register Empty Transmitter Holding Register Empty Reading the IIR Register (if source of interrupt) or writing into the Transmitter Holding Register |
| | | 000 Fourth MODEM Status Clear to Send or Data Set Ready or Ring Indicator or Data Carrier Detect Reading the MODEM Status Register |
| 0 | R | This bit can be used in a prioritized interrupt environment to indicate whether an interrupt is pending. When bit 0 is logic 0, an interrupt is pending and the IIR contents may be used as a pointer to the appropriate interrupt service routine. When bit 0 is logic 1, no interrupt is pending |

FIFO Control Register

This is a write-only register at the same location as the IIR (the IIR is a read-only register). This register is used to enable the FIFOs, clear the FIFOs and set the RCVR FIFO trigger level.

| Bits | Type | Function |
|------|------|---|
| 7:6 | W | These two bits sets the trigger level for the RCVR FIFO interrupt |
| | | Value RCVR FIFO Trigger Level (Bytes) |
| | | 00 01 |
| | | 01 04 |
| | | 10 08 |
| | | 11 14 |
| 5:3 | - | Reserved |
| 2 | W | Writing 1 resets the transmitter FIFO counter logic to 0. The shift register is not cleared. The 1 that is written to this bit position is self-clearing |
| 1 | W | Writing 1 resets the receiver FIFO counter logic to 0. The shift register is not cleared. The 1 that is written to this bit position is self-clearing |
| 0 | W | Writing 1 enables both the XMIT and RCVR FIFOs. Resetting FCR0 will clear all bytes in both FIFOs. When changing from FIFO Mode to 16C450 Mode and vice versa, data is automatically cleared from the FIFOs. This bit must be a 1 when other FCR bits are written to or they will not be programmed |

10.9.2.4 LCR

The system programmer specifies the format of the asynchronous data communications exchange and set the Divisor Latch Access bit via the Line Control Register (LCR). The programmer can also read the contents of the Line Control Register. The read capability simplifies system programming and eliminates the need for separate storage in system memory of the line characteristics.

| UxBase+0x0C | | | | | | | |
|-------------|-----------|--|-------------|---------------|----------------|-------------|--------|
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DLAB | SET BREAK | STICK PARITY | EVEN PARITY | PARITY ENABLE | STOPBIT NUMBER | WORD SELECT | LENGTH |
| Bits | Type | Function | | | | | |
| 7 | | This bit is the Divisor Latch Access Bit (DLAB). It must be set HIGH (logic 1) to access the Divisor Latches of the Baud Generator during a Read or Write operation. It must be set LOW (logic 0) to access the Receiver Buffer, the Transmitter Holding Register or the Interrupt Enable Register | | | | | |
| 6 | | This bit is the Break Control bit. It causes a break condition to be transmitted to the receiving | | | | | |

| | | UART. When it is set to logic 1, the serial output (SOUT) is forced to the Spacing (logic 0) state. The break is disabled by setting logic 0. The Break Control bit acts only on SOUT and has no effect on the transmitter logic. Note: This feature enables the CPU to alert a terminal in a computer communications system. If the following sequence is followed, no erroneous or extraneous characters will be transmitted because of the break. | | | | | | | | | | |
|-------|------------------|---|-------|------------------|----|--------|----|--------|----|--------|----|--------|
| 5 | | This bit is the Stick Parity bit. When bits 3, 4 and 5 are logic 1 the Parity bit is transmitted and checked as logic 0. If bits 3 and 5 are 1 and bit 4 is logic 0 then the Parity bit is transmitted and checked as logic 1. If bit 5 is a logic 0 Stick Parity is disabled. | | | | | | | | | | |
| 4 | | This bit is the Even Parity Select bit. When bit 3 is logic 1 and bit 4 is logic 0, an odd number of logic 1s is transmitted or checked in the data word bits and Parity bit. When bit 3 is logic 1 and bit 4 is logic 1, an even number of logic 1s is transmitted or checked. | | | | | | | | | | |
| 3 | | This bit is the Parity Enable bit. When bit 3 is logic 1, a Parity bit is generated (transmit data) or checked (receive data) between the last data word bit and Stop bit of the serial data. (The Parity bit is used to produce an even or odd number of 1s when the data word bits and the Parity bit are summed). | | | | | | | | | | |
| 2 | | This bit specifies the number of Stop bits transmitted and received in each serial character. If bit 2 is logic 0, one Stop bit is generated in the transmitted data. If bit 2 is logic 1 when a 5-bit word length is selected via bits 0 and 1, one and a half Stop bits are generated. If bit 2 is a logic 1 when either a 6-, 7- or 8-bit word length is selected, two Stop bits are generated. The Receiver checks the first Stop-bit only, regardless of the number of Stop bits selected. | | | | | | | | | | |
| 1:0 | R/W | These two bits specify the number of bits in each transmitted and received serial character. The encoding of bits 0 and 1 is as follows: | | | | | | | | | | |
| | | <table border="1"> <thead> <tr> <th>Value</th> <th>Character Length</th> </tr> </thead> <tbody> <tr> <td>00</td> <td>5 Bits</td> </tr> <tr> <td>01</td> <td>6 Bits</td> </tr> <tr> <td>10</td> <td>7 Bits</td> </tr> <tr> <td>11</td> <td>8 Bits</td> </tr> </tbody> </table> | Value | Character Length | 00 | 5 Bits | 01 | 6 Bits | 10 | 7 Bits | 11 | 8 Bits |
| Value | Character Length | | | | | | | | | | | |
| 00 | 5 Bits | | | | | | | | | | | |
| 01 | 6 Bits | | | | | | | | | | | |
| 10 | 7 Bits | | | | | | | | | | | |
| 11 | 8 Bits | | | | | | | | | | | |

Programmable Baud Generator

The UART contains a programmable Baud Generator that is capable of taking any clock input from DC to 8.0MHz and dividing it by any divisor from 2 to $2^{16}-1$. 5.185 MHz(70MHz CPU Clock) is the highest input clock frequency recommended when the divisor=1. The output frequency of the Baud Generator is 16 x the Baud [divisor # = (frequency input) / (baud rate x 16)]. Two 8-bit latches store the divisor in a 16-bit binary format. These Divisor Latches must be loaded during initialization to ensure proper operation of the Baud Generator. Upon loading either of the Divisor Latches, a 16-bit Baud counter is immediately loaded.

Baud rate table below provides decimal divisors to use with a crystal frequency of 3.6864MHz. For baud rates of 38400 and below, the error obtained is minimal. The accuracy of the desired baud rate is dependent on the crystal frequency chosen. Using a divisor of zero is not recommended.

| Desired Baud Rate | Decimal Divisor (Used to generate 16 x Clock) | Percent Error Difference Between Desired and Actual |
|-------------------|--|---|
| 50 | 4608 | - |
| 110 | 2094 | 0.026 |
| 300 | 768 | - |
| 1200 | 192 | - |
| 2400 | 96 | - |
| 4800 | 48 | - |
| 9600 | 24 | - |
| 19200 | 12 | - |
| 38400 | 6 | - |
| 57600 | 4 | - |
| 115200 | 2 | - |

Table 10-10 Baud Rate with Decimal Divisor at 3.6864MHz Crystal Frequency

10.9.2.5 MCR

This register controls the interface with the MODEM or data set (or a peripheral device emulating a MODEM).

| | | | | | | | |
|-------------|---|---|------|---|---|-----|-----|
| UxBase+0x10 | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0 | 0 | 0 | LOOP | - | - | RTS | DTR |

| Bits | Type | Function |
|------|------|---|
| 7:5 | R | These bits are permanently set to logic 0 |
| 4 | | This bit provides a local loop back feature for diagnostic testing of the UART. When bit 4 is set to logic 1, the following occur: the transmitter Serial Output (SOUT) is set to the Marking (logic 1) state; the receiver Serial Input (SIN) is disconnected; the output of the Transmitter Shift Register is "looped back" into the Receiver Shift Register input; the four MODEM Control inputs (NCTS, NDSR, NDCD and NRI) are disconnected; and the two MODEM Control outputs (NDTR and NRTS) are internally connected to the four MODEM Control inputs, and the MODEM Control output pins are forced to their inactive state (HIGH). On the diagnostic mode, data that is transmitted is immediately received. This feature allows the processor to verify the transmit- and received-data paths of the UART. In the diagnostic mode, the receiver and transmitter interrupts are fully operational. Their sources are external to the part. The MODEM Control interrupts are also operational, but the interrupts sources are now the lower four bits of the MODEM Control Register instead of the four MODEM Control inputs. The interrupts are still controlled by the Interrupt Enable Register. |
| 3:2 | - | Reserved |
| 1 | | This bit controls the Request to Send (nURTS) output. Bit 1 affects the NRTS output in a manner identical to that described above for bit 0. |
| 0 | R/W | This bit controls the Data Terminal Ready (nUDTR) output. When bit is set to logic 1, the NDTR output is forced to logic 0. When bit 0 is reset to logic 0, the NDTR output is forced to logic 1. Note: The NDTR output of the UART may be applied to an EIA inverting line driver (such as the DS1488) to obtain the proper polarity input at the succeeding MODEM or data set. |

10.9.2.6 LSR

This register provides status information to the CPU concerning the data transfer.

| | | | | | | | |
|-------------|------|------|----|----|----|----|----|
| UxBase+0x14 | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| FIFO ERR | TEMT | THRE | BI | FE | PE | OE | DR |

| Bits | Type | Function |
|------|------|---|
| 7 | R | In the 16C450 mode this is always 0. In the FIFO mode LSR7 is set when there is at least one parity error, framing error or break indication in the FIFO. LSR7 is cleared when the CPU reads the LSR, if there are no subsequent errors in the FIFO. |
| 6 | R | This bit is the Transmitter Empty (TEMT) indicator. Bit 6 is set to a logic 1 whenever the Transmitter Holding Register (THR) and the Transmitter Shift Register (TSR) 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 one whenever the transmitter FIFO and register are both empty. |
| 5 | R | This bit is the Transmitter Holding Register Empty (THRE) indicator. Bit 5 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 the CPU when the Transmit Holding Register Empty Interrupt enable is set HIGH. The THRE bit is set to a logic 1 when a character is transferred from the Transmitter Holding Register into the Transmitter Shift Register. The bit is reset to logic 0 concurrently with the loading of the Transmitter Holding Register. In the FIFO mode this bit is set when the XMIT FIFO is empty; it is cleared when at least 1 byte is written to the XMIT FIFO. |
| 4 | R | This bit is the Break Interrupt (BI) indicator. Bit 4 is set to logic 1 whenever the received data input is held in the Spacing (logic 0) state for longer than a full word transmission time (that is, the total time of Start bit + data bits + Parity + Stop bits). The BI indicator is reset whenever the CPU reads the contents of the Line Status Register. In the FIFO mode this error is associated with the particular character in the FIFO it applies to. This error is revealed to the CPU when its associated character is at the top of the FIFO. When break occurs, only one zero character is loaded into the FIFO. The next character transfer is enabled after SIN goes |

| | | |
|---|---|---|
| | | to the marking state and receives the next valid start bit. |
| | | Note: Bits 1--4 are the error conditions that produce a Receiver Line Status interrupt whenever any of the corresponding conditions are detected and the interrupt is enabled. |
| 3 | R | This bit is the Framing Error (FE) indicator. Bit 3 indicates that the received character did not have a valid stop bit. Bit 3 is set to logic 1 whenever the Stop bit following the last data bit or parity bit is detected as a logic 0 bit (Spacing level). The FE indicator is reset whenever the CPU reads the contents of the Line Status Register. In the FIFO mode this error is associated with the particular character in the FIFO it applies to. This error is revealed to the CPU when its associated character is at the top of the FIFO. The UART will try to re-synchronize after a framing error. To do this it assumes that the framing error was due to the next start bit, so it samples this "start" bit twice and then takes in the "data". |
| 2 | R | This bit is the Parity Error (PE) indicator. Bit 2 indicates that the received data character does not have the correct even or odd parity, as selected by the even-parity-select bit. The PE bit is set to logic 1 upon detection of a parity error and is reset to logic 0 whenever the CPU reads the contents of the Line Status Register. In the FIFO mode, this error is associated with the particular character in the FIFO it applies to. This error is revealed to the CPU when its associated character is at the top of the FIFO. |
| 1 | R | This bit is the Overrun Error (OE) indicator. Bit 1 indicates that data in the Receiver Buffer Register was not read by the CPU before the next character was transferred into the Receiver Buffer Register, thereby destroying the previous character. The OE indicator is set to logic 1 upon detection of an overrun condition and reset whenever the CPU reads the contents of the Line Status Register. If the FIFO mode data continues to fill the FIFO beyond the trigger level, an overrun error will occur only after the FIFO is full and the next character has been completely received in the shift register. OE is indicated to the CPU as soon as it happens. The character in the shift register is overwritten, but it is not transferred to the FIFO. |
| 0 | R | This bit is the receiver Data Ready (DR) indicator. Bit 0 is set to logic 1 whenever a complete incoming character has been received and transferred into the Receiver Buffer Register or the FIFO. Bit 0 is reset to logic 0 by reading all of the data in the Receiver Buffer Register or the FIFO. |

Some bits in LSR are automatically cleared when CPU reads the LSR register, so interrupt handling routine should be written that if once reads LSR, then keep the value through entire the routine because second reading LSR returns just reset value.

10.9.2.7 MSR

This register provides the current state of the control lines from the MODEM (or peripheral device) to the CPU. In addition to this current-state information, four bits of the MODEM Status Register provide change information. These bits are set to logic 1 whenever a control input from the MODEM change state. They are reset to logic 0 whenever the CPU reads the MODEM Status Register.

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-----|----|-----|-----|------|------|------|------|
| DCD | RI | DSR | CTS | DDCD | TERI | DDSR | DCTS |

UxBASE+0x18

| Bits | Type | Function |
|------|------|---|
| 7 | | This bit is the complement of the Data Carrier Detect (nUDCD) input. If bit 4 of the MCR is set to a 1, this bit is equivalent to OUT2 in the MCR. |
| 6 | | This bit is the complement of the Ring Indicator (nURING) input. If bit 4 of the MCR is set to a 1, this bit is equivalent to OUT1 in the MCR. |
| 5 | | This bit is the complement of the Data Set Ready (nUDSR) input. If bit 4 of the MCR is set to a 1, this bit is equivalent to DTR in the MCR. |
| 4 | | This bit is the complement of the Clear to Send (nUCTS) input. If bit 4 (loop) of the MCR is set to a 1, this bit is equivalent to RTS in the MCR. |
| 3 | | This bit is the Delta Data Carrier Detect (nUDCD) indicator. Bit 3 indicates that the nUDCD input to the chip has changed state since the last time it was read by the CPU. Note: Whenever bit 0, 1, 2 or 3 is set to logic 1, a MODEM Status Interrupt is generated. |
| 2 | | This bit is the Trailing Edge of Ring Indicator (TERI) detector. Bit 2 indicates that the nURING input to the chip has changed from a LOW to a HIGH state. |
| 1 | | This bit is the Delta Data Set Ready (nUDSR) indicator. Bit 1 indicates that the nUDSR input |

| | | |
|---|-----|---|
| | | to the chip has changed state since the last time it was read by the CPU. |
| 0 | R/W | This bit is the Delta Clear to Send (nUCTS) indicator. Bit 0 indicates that the nUCTS input to the chip has changed state since the last time it was read by the CPU. |

10.9.2.8 SCR

This 8-bit Read/Write Register does not control the UART in any way. It is intended as a scratchpad register to be used by the programmer to hold data temporarily.

| | | | | | | | |
|-------------|-------------|------------------------|---|---|---|---|-------------|
| | | | | | | | UxBase+0x1C |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DATA | | | | | | | |
| Bits | Type | Function | | | | | |
| 7:0 | R/W | Temporary data storage | | | | | |

10.9.2.9 UartEn

| | | | | | | | UxBase+0x30 | | | | | | | | | | | | | | | | | | |
|---|------|--|--|---------------------------------------|---|----------------------------|-------------|------|------|----------|-----|---|----------|---|-----|--|---|-----|---|---|-----|--|---|-----|---|
| | | | | SIR Loop Back <i>Uart1 only</i> | Full Duplex Force <i>Uart1 only</i> | SIREN <i>Uart1 only</i> | UARTEN | | | | | | | | | | | | | | | | | | |
| <table border="1"> <thead> <tr> <th>Bits</th> <th>Type</th> <th>Function</th> </tr> </thead> <tbody> <tr> <td>7:4</td> <td>-</td> <td>Reserved</td> </tr> <tr> <td>3</td> <td>R/W</td> <td>SIR Loop-back Test (<i>Uart1 only</i>) 0 = SIR Loop-back Test disable 1 = SIR Loop-back Test enable.</td> </tr> <tr> <td>2</td> <td>R/W</td> <td>SIR Full-duplex Force (<i>Uart1 only</i>) 0 = Half Duplex. 1 = Full Duplex.</td> </tr> <tr> <td>1</td> <td>R/W</td> <td>SIR Enable (<i>Uart1 only</i>) 0 = SIR Mode disable 1 = SIR Mode enable (If you use SIR function, you must set this bit with UART En bit at the same time.)</td> </tr> <tr> <td>0</td> <td>R/W</td> <td>UART Enable. 0 = UART disable (Power-Down), UART Clock stop. 1 = UART enable.</td> </tr> </tbody> </table> | | | | | | | | Bits | Type | Function | 7:4 | - | Reserved | 3 | R/W | SIR Loop-back Test (<i>Uart1 only</i>) 0 = SIR Loop-back Test disable 1 = SIR Loop-back Test enable. | 2 | R/W | SIR Full-duplex Force (<i>Uart1 only</i>) 0 = Half Duplex. 1 = Full Duplex. | 1 | R/W | SIR Enable (<i>Uart1 only</i>) 0 = SIR Mode disable 1 = SIR Mode enable (If you use SIR function, you must set this bit with UART En bit at the same time.) | 0 | R/W | UART Enable. 0 = UART disable (Power-Down), UART Clock stop. 1 = UART enable. |
| Bits | Type | Function | | | | | | | | | | | | | | | | | | | | | | | |
| 7:4 | - | Reserved | | | | | | | | | | | | | | | | | | | | | | | |
| 3 | R/W | SIR Loop-back Test (<i>Uart1 only</i>) 0 = SIR Loop-back Test disable 1 = SIR Loop-back Test enable. | | | | | | | | | | | | | | | | | | | | | | | |
| 2 | R/W | SIR Full-duplex Force (<i>Uart1 only</i>) 0 = Half Duplex. 1 = Full Duplex. | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | R/W | SIR Enable (<i>Uart1 only</i>) 0 = SIR Mode disable 1 = SIR Mode enable (If you use SIR function, you must set this bit with UART En bit at the same time.) | | | | | | | | | | | | | | | | | | | | | | | |
| 0 | R/W | UART Enable. 0 = UART disable (Power-Down), UART Clock stop. 1 = UART enable. | | | | | | | | | | | | | | | | | | | | | | | |

10.9.3 FIFO Interrupt Mode Operation

When the RCVR FIFO and receiver interrupts are enabled (FCR 0 = 1, IER 0 = 1) RCVR interrupts occur as follows:

1. The received data available interrupt will be issued to the CPU when the FIFO has reached its programmed trigger level. It will be cleared as soon as the FIFO drops below its programmed trigger level.
2. The IIR receive data available indication also occurs when the FIFO trigger level is reached, and like the interrupt, it is cleared when the FIFO drops below the trigger level.
3. The receiver line status interrupt (IIR-06), as before, has higher priority than the received data available (IIR-04) interrupt.
4. The data ready bit (LSR 0) is set as soon as a character is transferred from the shift register to the RCVR FIFO. It is reset when the FIFO is empty.
- 5.

When RCVR FIFO and receiver interrupts are enabled, RCVR FIFO time-out interrupts occurs as follows:

1. A FIFO time-out interrupt occurs if the following conditions exist: at least one character is in the FIFO
- the most recent serial character received was longer than four continuous character times ago

(if two stop bits are programmed, the second one is included in this time delay)

- the most recent CPU read of the FIFO was longer than four continuous character times ago

This will cause a maximum character received to interrupt issued delay of 160 ms at 300 baud with a 12-bit character.

2. Character times are calculated by using the RCLK input, which is the internal signal of UART for a clock signal (this makes the delay proportional to the baud rate).

3. When a time-out interrupt has occurred, it is cleared and the timer is reset when the CPU reads one character from the RCVR FIFO.

4. When a time-out interrupt has not occurred the time-out timer is reset after a new character is received or after the CPU reads the RCVR FIFO.

When the XMIT FIFO and transmitter interrupts are enabled (FCR 0 = 1, IER 1 = 1), XMIT interrupts occurs as follows:

1. 1 The transmitter holding register interrupt (02) occurs when the XMIT FIFO is empty. It is cleared as soon as the transmitter holding register is written to (1 to 16 characters may be written to the XMIT FIFO while servicing this interrupt) or the IIR is read.

2. 2 The transmitter FIFO empty indications will be delayed 1 character time minus the last stop bit time whenever the following occurs: THRE = 1 and there has not been at least two bytes at the same time in the transmit FIFO since the last THRE = 1. The first transmitter interrupt affect changing FCRO will be immediate if it is enabled.

Character time-out and RCVR FIFO trigger level interrupts have the same priority as the current received data available interrupt; XMIT FIFO empty has the same priority as the current transmitter holding register empty interrupt.

10.10 Watchdog Timer

The watchdog timer (WDT) has a one-channel for monitoring system operations. If a system becomes uncontrolled and the timer counter overflows without being rewritten correctly by the CPU, a reset signal is output to PMU

When this watchdog function is not needed, the WDT can be used as an interval timer. In the interval timer operation, an interval timer interrupt is generated at each counter overflow.

FEATURES

- Watchdog timer mode and interval timer mode
- Interrupt signal INT_WDT to interrupt controller in the watchdog timer mode & interval timer mode
- Output signal MNRESET to PMU (Power Management Unit)
- Eight counter clock sources
- Selection whether to reset the chip internally or not
- Reset signal type: manual reset

10.10.1 Watchdog Timer Operation

10.10.1.1 The Watchdog Timer Mode

To use the WDT as a watchdog timer, set the MODESEL and TMEN bits of the WDTCTRL to 1. Software must prevent WDTCNT overflow by rewriting the WDTCNT value (normally by writing 0x00) before overflow occurs. If the WDTCNT fails to be rewritten and overflow due to a system crash or the like, INT_WDT signal and PORESET/MNRESET signal are output. The INT_WDT signal is not output if INTREN is disabled (INTREN = 0).

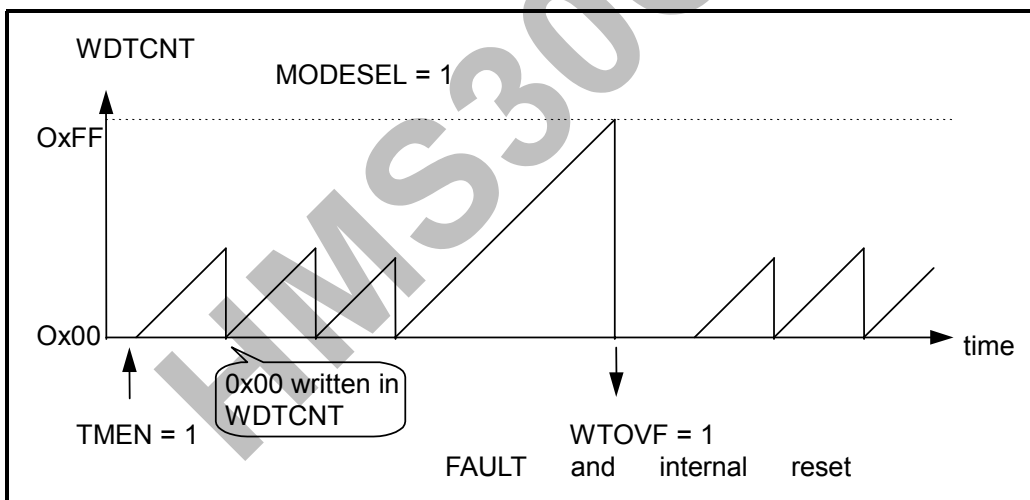


Figure 10-9 WDT Operation in the Watchdog Timer mode

If the RSTEN bit in the WDTCTRL is set to 1, a signal to reset the chip will be generated internally when WDTCNT overflows.

10.10.1.2 The Interval Timer Mode

To use the WDT as an interval timer, clear MODESEL in WDTCTRL to 0 and set TMEN to 1. A watchdog timer interrupt (INT_WDT) is generated each time the timer counter overflows. This function can be used to generate interval timer interrupts at regular intervals.

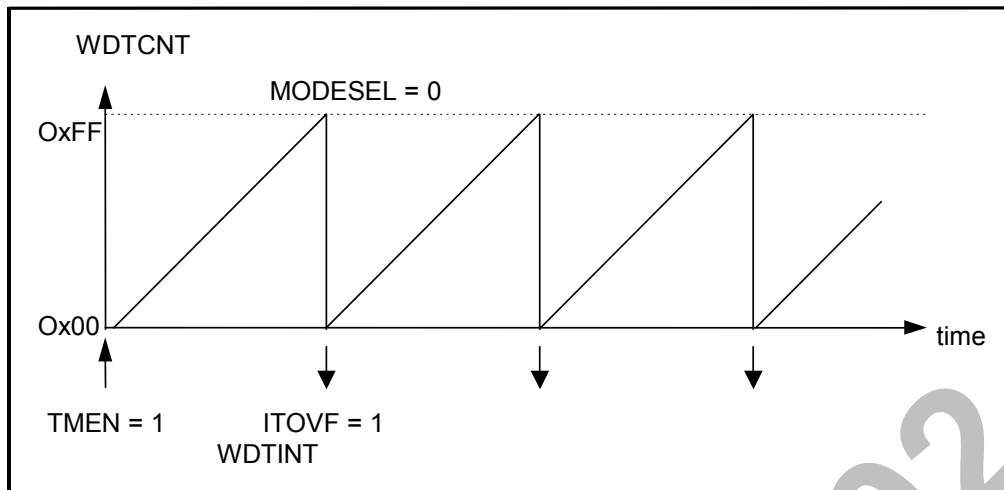


Figure 10-10 WDT Operation in the Interval Timer mode

10.10.1.3 Timing of setting the overflow flag

In the interval timer mode when the WDCNT overflows, the ITOVF flag is set to 1 and a watchdog timer interrupt (INT_WDT) is requested.

In the watchdog timer mode when the WDCNT overflows, the WTOVF bit of the WDTSTAT is set to 1 and a WDTOUT signal is output. When RSTEN bit is set to 1, WDCNT overflow enables an internal reset signal to be generated for the entire chip.

10.10.1.4 Timing of clearing the overflow flag

When the WDT Status Register (WDTSTAT) is read, the overflow flag is cleared.

10.10.2 Registers

| Address | Name | Width | Default | Description |
|-------------|---------|-------|---------|---------------------|
| 0x8002.B000 | WDTCTRL | 8 | 0x0 | Timer/Reset Control |
| 0x8002.B004 | WDTSTAT | 2 | 0x0 | Reset Status |
| 0x8002.B008 | WDCNT | 8 | | Timer Counter |

Table 10-11 Watchdog Timer Register Summary

10.10.2.1 WDT Control Register (WDTCTRL)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-------------|---------|--|-------|--------|----------------|---|---|
| INTREN | MODESEL | TMEN | RSTEN | RSTSEL | CLK SOURCE SEL | | |
| 0x8002.B000 | | | | | | | |
| Bits | Type | Function | | | | | |
| 7 | R/W | Enable or disable the interrupt request. 0 = disable 1 = enable | | | | | |
| 6 | R/W | Select whether to use the WDT as a watchdog timer or interval timer. 0 = interval timer mode 1 = watchdog timer mode | | | | | |
| 5 | R/W | Enable or disable the timer. 0 = disable 1 = enable | | | | | |
| 4 | R/W | Select whether to reset the chip internally or not if the TCNT overflows in the watchdog timer mode. 0 = disable | | | | | |

| | | |
|-----|-----|---|
| | | 1 = enable |
| 3 | R/W | Select the type of generated internal reset if the TCNT overflows in the watchdog timer mode. 1 = manual reset enable |
| 2:0 | R/W | The WDT has a clock generator which products eight counter clock sources. The clock signals are obtained by dividing the frequency of the system clock (B_CLK). |

VALUE
CLOCK SOURCE (SYSTEM CLOCK = 40 MHz)
OVERFLOW INTERVAL

- 000
The system clock is divided by 2
12.8 us
- 001
The system clock is divided by 8
51.2 us
- 010
The system clock is divided by 32
204.8 us
- 011
The system clock is divided by 64
409.6 us
- 100
The system clock is divided by 256
1.64 ms
- 101
The system clock is divided by 512
3.28 ms
- 110
The system clock is divided by 2048
13.11 ms
- 111
The system clock is divided by 8192
52.43 ms

10.10.2.2 WDT Status Register (WDTSTAT)

| | | | | | | | | | |
|--|--|--|--|--|--|--|----------|----------|-------------|
| | | | | | | | 1 | 0 | 0x8002.B004 |
| | | | | | | | ITOVF | WTOVF | |

| Bits | Type | Function |
|------|------|---|
| 7:2 | - | Reserved |
| 1 | R | Set when WDCNT has overflowed in the interval timer mode. |
| 0 | R | Set when WDCNT has overflowed in the watchdog timer mode. |

10.10.2.3 WDT Counter (WDCNT)

| | | | | | | | | |
|----------|----------|----------|----------|----------|----------|----------|----------|-------------|
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 0x8002.B008 |
| WDCNT | | | | | | | | |

| Bits | Type | Function |
|------|------|----------|
|------|------|----------|

| | | |
|-----|---|---|
| 7:0 | R | 8-bit up counter. When the timer is enabled, the timer counter starts counting pulse of the selected clock source. When the value of the WDCNT changes from 0xFF-0x00(overflows), a watchdog timer overflow signal is generated in the both timer modes. The WDCNT is initialized to 0x00 by a power-reset. |
|-----|---|---|

10.10.3 Examples of Register Setting

10.10.3.1 Interval Timer Mode

TCNT = 0x00 TRCR = 0xA0

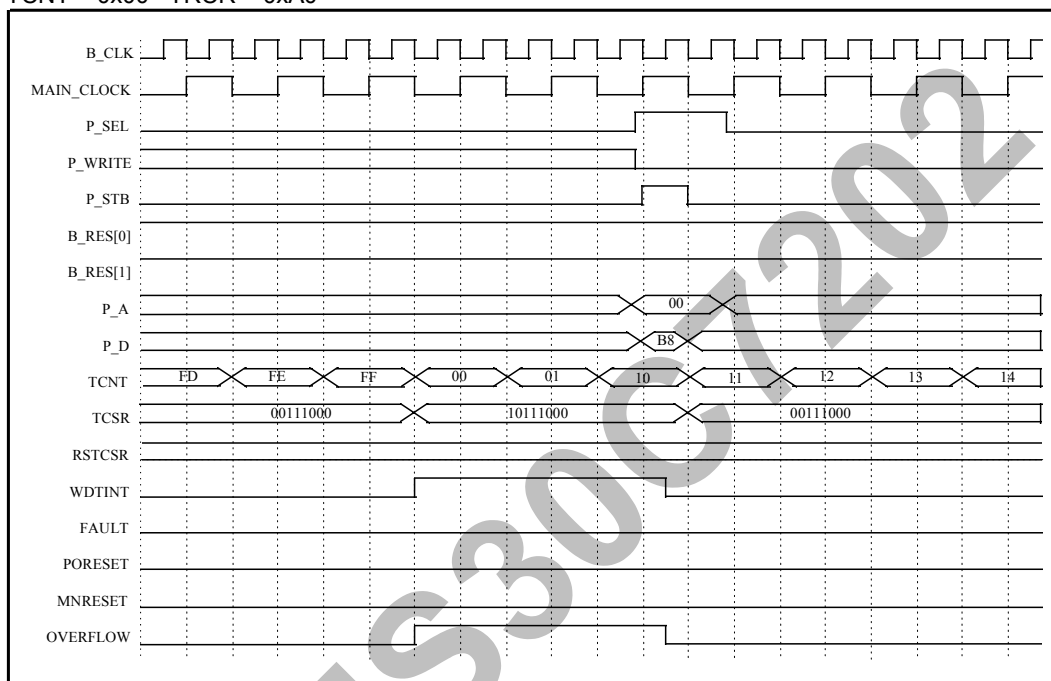


Figure 10-11 Interrupt Clear in the interval timer mode

10.10.3.2 Watchdog Timer Mode with Internal Reset Disable

TCNT = 0x00 (normally) TRCR = 0xE0

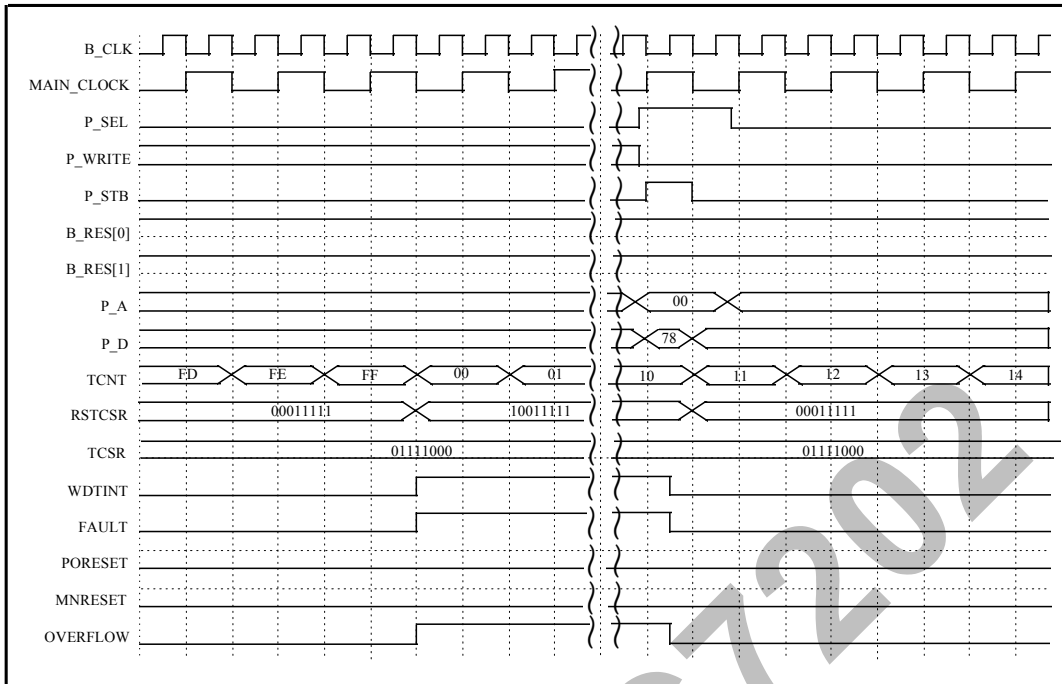


Figure 10-12 Interrupt Clear in the watchdog timer mode with reset disable

10.10.3.3 Watchdog Timer Mode with Manual Reset

TCNT = 0x00 TRCR = 0xF8

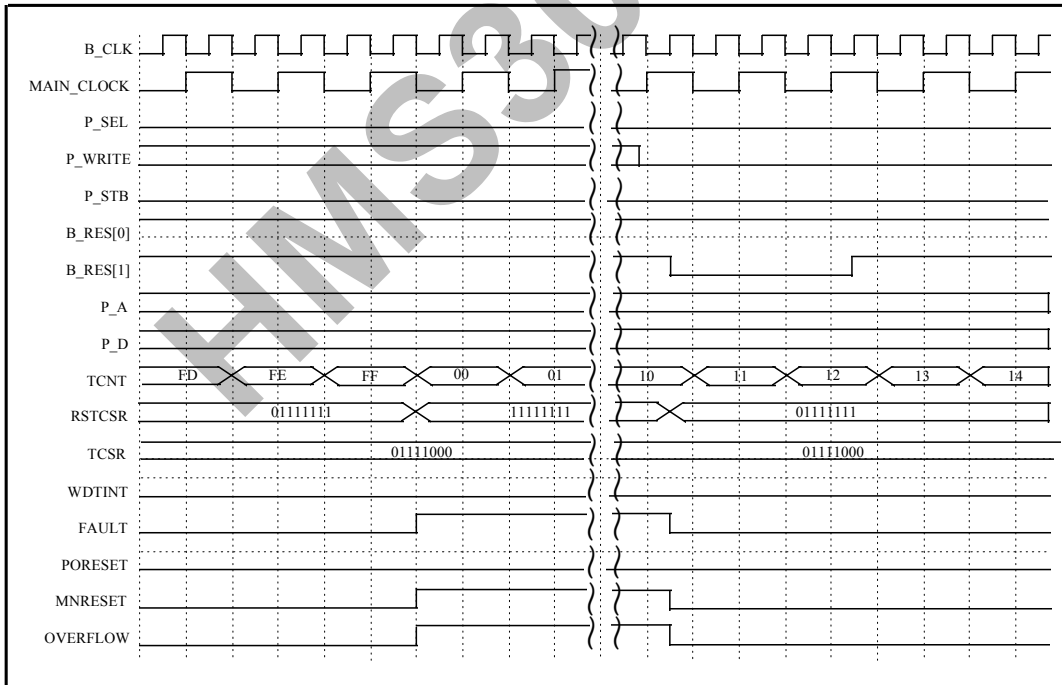


Figure 10-13 Interrupt Clear in the watchdog timer mode with manual reset

11 DEBUG AND TEST INTERFACE

11.1 Overview

The HMS30C7202 has built-in features that enable debug and test in a number of different contexts. Firstly, there are circuit structures to help with software development. Secondly, the device contains boundary scan cells for circuit board test. Finally, the device contains some special test modes that enable the generation production patterns for the device itself.

11.2 Software Development Debug and Test Interface

The ARM720T and Piccolo processors incorporated inside HMS30C7202 contain hardware extensions for advanced debugging features. These are intended to ease user development and debugging of application software, operating systems, and the hardware itself.

Full details of the debug interfaces and their programming can be found in *ARM720T Data Sheet* (ARM DDI-0087) and *Piccolo Data Sheet* (ARM DDI-0128). The MultiICE product enables the ARM720T and Piccolo macrocells to be debugged in one environment. Refer to *Guide to MultiICE* (ARM DUI-0048).

11.3 Test Access Port and Boundary-Scan

HMS30C7202 contains full boundary scan on its inputs and outputs to help with circuit board test. This supports both INTEST and EXTEST, allowing patterns to be applied serially to the HMS30C7202 when fixed in a board and for full circuit board connection respectively. The boundary-scan interface conforms to the IEEE Std. 1149.1- 1990, Standard Test Access Port and Boundary-Scan Architecture. (Please refer to this standard for an explanation of the terms used in this section and for a description of the TAP controller states.) The boundary-scan interface provides a means of testing the core of the device when it is fitted to a circuit board, and a means of driving and sampling all the external pins of the device irrespective of the core state. This latter function permits testing of both the device's electrical connections to the circuit board, and (in conjunction with other devices on the circuit board having a similar interface) testing the integrity of the circuit board connections between devices. The interface intercepts all external connections within the device, and each such "cell" is then connected together to form a serial register (the boundary scan register). The whole interface is controlled via 5 dedicated pins: **TDI**, **TMS**, **TCK**, **nTRST** and **TDO**. **Figure 11-1: Test Access Port (TAP) Controller State Transitions** shows the state transitions that occur in the TAP controller.

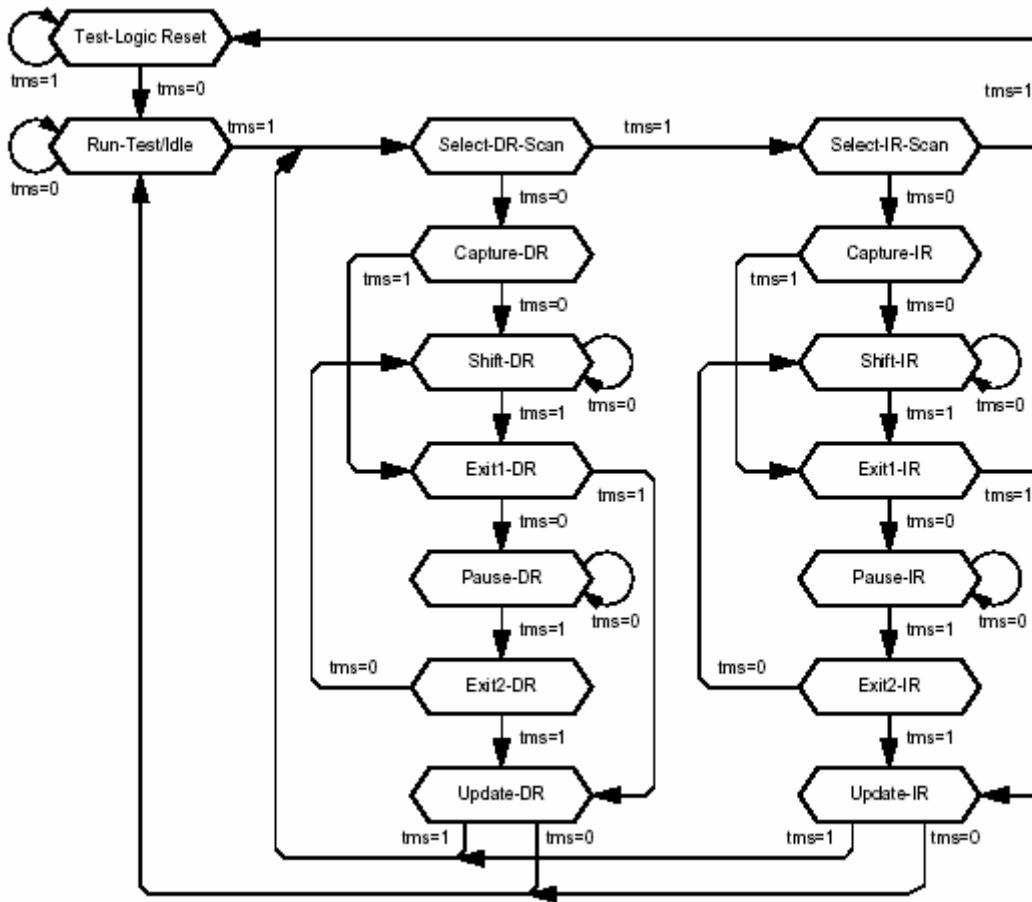


Figure 11-1: Test Access Port (TAP) Controller State Transitions

11.3.1 Reset

The boundary-scan interface includes a state-machine controller (the TAP controller). A pulldown resistor is included in the **nTRST** pad which holds the TAP controller state machine in a safe state after power up. In order to use the boundary scan interface, **nTRST** should be driven HIGH to take the TAP state machine out of reset.

The action of reset (either a pulse or a DC level) is as follows:

- System mode is selected (i.e. the boundary scan chain does NOT intercept any of the signals passing between the pads and the core).
- IDcode mode is selected. If **TCK** is pulsed, the contents of the ID register will be clocked out of **TDO**.

Note The TAP controller inside HMS30C7202 contains a scan chip register which is reset to the value **b0011** thus selecting the boundary scan chain. If this register is programmed to any value other than **b0011**, then it must be reprogrammed with **b0011** or a reset applied before boundary scan operation can be attempted.

11.3.2 Pull up Resistors

The IEEE 1149.1 standard requires pullup resistors in the input pins. However, to ensure safe operation an internal pulldown is present in the **nTRST** pin and therefore will have to be driven HIGH when using this interface.

| Pin Name | Internal Resistor |
|----------|-------------------|
| TCLK | Pullup |
| nTRST | Pulldown |
| TMS | Pullup |
| TDI | Pullup |

11.3.3 Instruction Register

The instruction register is 4 bits in length. There is no parity bit. The fixed value loaded into the instruction register during the CAPTURE-IR controller state is: 0001.

11.3.4 Public Instructions

The following public instructions are supported:

| Instruction | Binary Code |
|----------------|-------------|
| EXTEST | 0000 |
| SAMPLE/PRELOAD | 0011 |
| CLAMP | 0101 |
| HIGHZ | 0111 |
| CLAMPZ | 1001 |
| INTEST | 1100 |
| IDCODE | 1110 |
| BYPASS | 1111 |

In the descriptions that follow, **TDI** and **TMS** are sampled on the rising edge of **TCK** and all output transitions on **TDO** occur as a result of the falling edge of **TCK**.

EXTEST (0000)

The BS (boundary-scan) register is placed in test mode by the EXTEST instruction. The EXTEST instruction connects the BS register between **TDI** and **TDO**. When the instruction register is loaded with the EXTEST instruction, all the boundary-scan cells are placed in their test mode of operation.

In the CAPTURE-DR state, inputs from the system pins and outputs from the boundary-scan output cells to the system pins are captured by the boundary-scan cells. In the SHIFT-DR state, the previously captured test data is shifted out of the BS register via the **TDO** pin, whilst new test data is shifted in via the **TDI** pin to the BS register parallel input latch. In the UPDATE-DR state, the new test data is transferred into the BS register parallel output latch. Note that this data is applied immediately to the system logic and system pins. The first EXTEST vector should be clocked into the boundary-scan register, using the SAMPLE/PRELOAD instruction, prior to selecting EXTEST to ensure that known data is applied to the system logic.

SAMPLE/PRELOAD (0011)

The BS (boundary-scan) register is placed in normal (system) mode by the SAMPLE/PRELOAD instruction.

The SAMPLE/PRELOAD instruction connects the BS register between **TDI** and **TDO**.

When the instruction register is loaded with the SAMPLE/PRELOAD instruction, all the boundary-scan cells are placed in their normal system mode of operation.

In the CAPTURE-DR state, a snapshot of the signals at the boundary-scan cells is taken on the rising edge of **TCK**. Normal system operation is unaffected. In the SHIFT-DR state, the sampled test data is shifted out of the BS register via the **TDO** pin, whilst new data is shifted in via the **TDI** pin to preload the BS register parallel input latch. In the UPDATE-DR state, the preloaded data is transferred into the BS register parallel output latch. Note that this data is not applied to the system logic or system pins while the SAMPLE/PRELOAD

instruction is active. This instruction should be used to preload the boundary-scan register with known data prior to selecting the INTEST or EXTEST instructions.

CLAMP (0101)

The CLAMP instruction connects a 1 bit shift register (the BYPASS register) between **TDI** and **TDO**. When the CLAMP instruction is loaded into the instruction register, the state of all output signals is defined by the values previously loaded into the boundary-scan register. A guarding pattern should be pre-loaded into the boundary-scan register using the SAMPLE/PRELOAD instruction prior to selecting the CLAMP instruction. In the CAPTURE-DR state, a logic 0 is captured by the bypass register. In the SHIFT-DR state, test data is shifted into the bypass register via **TDI** and out via **TDO** after a delay of one **TCK** cycle. Note that the first bit shifted out will be a zero. The bypass register is not affected in the UPDATE-DR state.

HIGHZ (0111)

The HIGHZ instruction connects a 1 bit shift register (the BYPASS register) between **TDI** and **TDO**. When the HIGHZ instruction is loaded into the instruction register, all outputs are placed in an inactive drive state. In the CAPTURE-DR state, a logic 0 is captured by the bypass register. In the SHIFT-DR state, test data is shifted into the bypass register via **TDI** and out via **TDO** after a delay of one **TCK** cycle. Note that the first bit shifted out will be a zero. The bypass register is not affected in the UPDATE-DR state.

CLAMPZ (1001)

The CLAMPZ instruction connects a 1 bit shift register (the BYPASS register) between **TDI** and **TDO**. When the CLAMPZ instruction is loaded into the instruction register, all outputs are placed in an inactive drive state, but the data supplied to the disabled output drivers is derived from the boundary-scan cells. The purpose of this instruction is to ensure, during production testing, that each output driver can be disabled when its data input is either a 0 or a 1. A guarding pattern (specified for this device at the end of this section) should be pre-loaded into the boundary-scan register using the SAMPLE/PRELOAD instruction prior to selecting the CLAMPZ instruction. In the CAPTURE-DR state, a logic 0 is captured by the bypass register. In the SHIFT-DR state, test data is shifted into the bypass register via **TDI** and out via **TDO** after a delay of one **TCK** cycle. Note that the first bit shifted out will be a zero. The bypass register is not affected in the UPDATE-DR state.

INTEST (1100)

The BS (boundary-scan) register is placed in test mode by the INTEST instruction. The INTEST instruction connects the BS register between **TDI** and **TDO**. When the instruction register is loaded with the INTEST instruction, all the boundary-scan cells are placed in their test mode of operation. In the CAPTURE-DR state, the complement of the data supplied to the core logic from input boundary-scan cells is captured, while the true value of the data that is output from the core logic to output boundary-scan cells is captured. Note that CAPTURE-DR captures the complemented value of the input cells for testability reasons. In the SHIFT-DR state, the previously captured test data is shifted out of the BS register via the **TDO** pin, whilst new test data is shifted in via the **TDI** pin to the BS register parallel input latch. In the UPDATE-DR state, the new test data is transferred into the BS register parallel output latch. Note that this data is applied immediately to the system logic and system pins. The first INTEST vector should be clocked into the boundary-scan register, using the SAMPLE/PRELOAD instruction, prior to selecting INTEST to ensure that known data is applied to the system logic. Single-step operation is possible using the INTEST instruction.

IDCODE (1110)

The IDCODE instruction connects the device identification register (or ID register) between **TDI** and **TDO**. The ID register is a 32-bit register that allows the manufacturer, part number and version of a component to be determined through the TAP. The IDCODE returned will be that for the ARM720T core. When the instruction register is loaded with the IDCODE instruction, all the boundary-scan cells are placed in their normal (system) mode of operation. In the CAPTURE-DR state, the device identification code (specified at the end of this section) is captured by the ID register.

In the SHIFT-DR state, the previously captured device identification code is shifted out of the ID register via the **TDO** pin, whilst data is shifted in via the **TDI** pin into the ID register. In the UPDATE-DR state, the ID register is unaffected.

BYPASS (1111)

The BYPASS instruction connects a 1 bit shift register (the BYPASS register) between **TDI** and **TDO**. When the BYPASS instruction is loaded into the instruction register, all the boundary-scan cells are placed in their normal (system) mode of operation. This instruction has no effect on the system pins. In the CAPTURE-DR state, a logic 0 is captured by the bypass register. In the SHIFT-DR state, test data is shifted into the bypass

register via **TDI** and out via **TDO** after a delay of one **TCK** cycle. Note that the first bit shifted out will be a zero. The bypass register is not affected in the UPDATE-DR state.

11.3.5 Test Data Registers

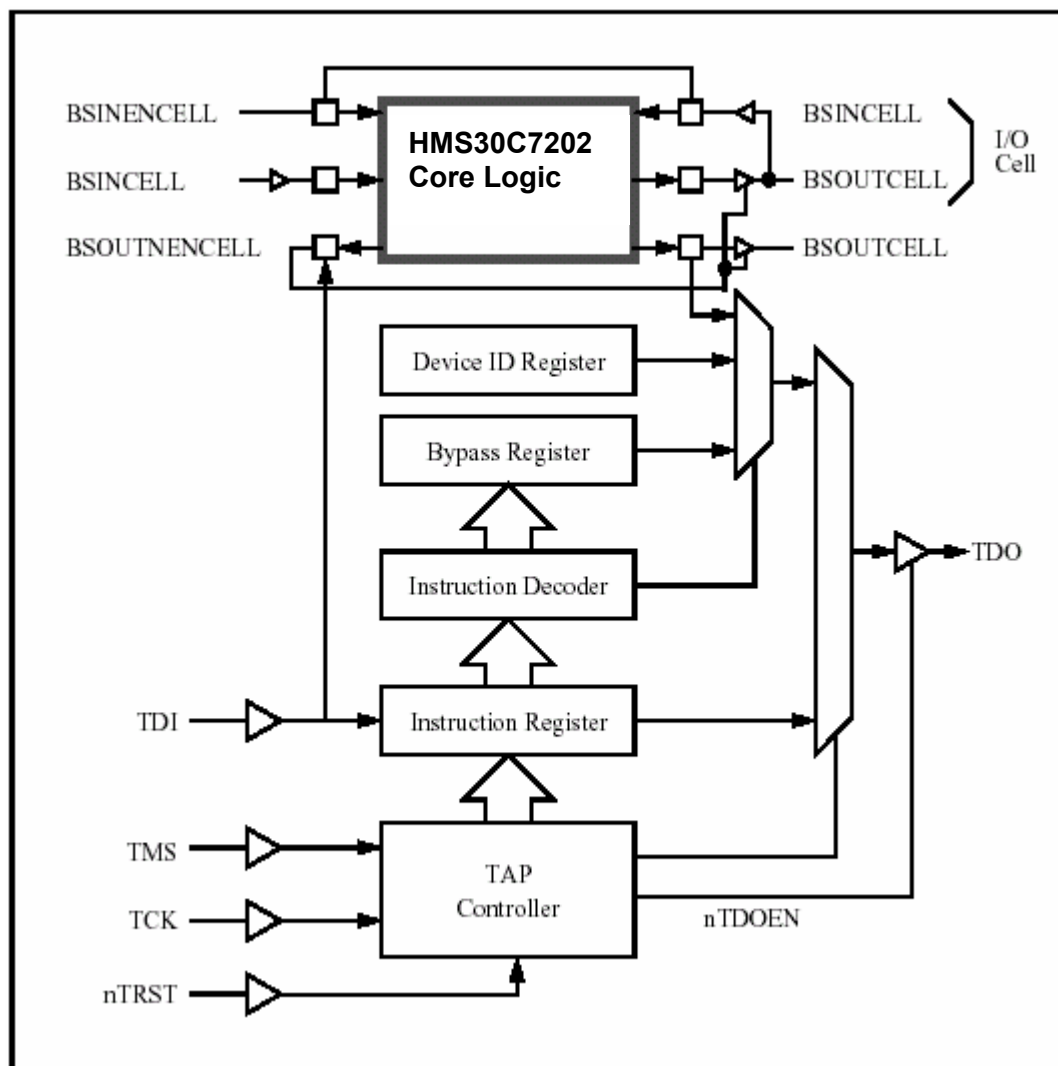


Figure 11-2: Boundary Scan Block Diagram

Bypass Register

Purpose: This is a single bit register which can be selected as the path between **TDI** and **TDO** to allow the device to be bypassed during boundary-scan testing.

Length: 1 bit

Operating Mode: When the **BYPASS** instruction is the current instruction in the instruction register, serial data is transferred from **TDI** to **TDO** in the **SHIFT-DR** state with a delay of one **TCK** cycle.

There is no parallel output from the bypass register.

A logic 0 is loaded from the parallel input of the bypass register in the **CAPTURE-DR** state.

Boundary Scan (BS) Register

Purpose: The **BS** register consists of a serially connected set of cells around the periphery of the device, at the interface between the core logic and the system input/output pads. This register can be used to isolate the

core logic from the pins and then apply tests to the core logic, or conversely to isolate the pins from the core logic and then drive or monitor the system pins. Operating modes: The BS register is selected as the register to be connected between **TDI** and **TDO** only during the SAMPLE/PRELOAD, EXTEST and INTEST instructions. Values in the BS register are used, but are not changed, during the CLAMP and CLAMPZ instructions. In the normal (system) mode of operation, straight-through connections between the core logic and pins are maintained and normal system operation is unaffected. In TEST mode (i.e. when either EXTEST or INTEST is the currently selected instruction), values can be applied to the core logic or output pins independently of the actual values on the input pins and core logic outputs respectively. On the HMS30C7202 all of the boundary scan cells include an update register and thus all of the pins can be controlled in the above manner.

Additional boundary-scan cells are interposed in the scan chain in order to control the enabling of tristateable buses. The values stored in the BS register after power-up are not defined. Similarly, the values previously clocked into the BS register are not guaranteed to be maintained across a Boundary Scan reset (from forcing **nTRST** LOW or entering the Test Logic Reset state).

Single-step Operation

HMS30C7202 is a static design and there is no minimum clock speed. It can therefore be single-stepped while the INTEST instruction is selected and the PLLs are bypassed.

This can be achieved by serializing a parallel stimulus and clocking the resulting serial vectors into the boundary-scan register. When the boundary-scan register is updated, new test stimuli are applied to the core logic inputs; the effect of these stimuli can then be observed on the core logic outputs by capturing them in the boundary-scan register.

11.3.6 Boundary Scan Interface Signals

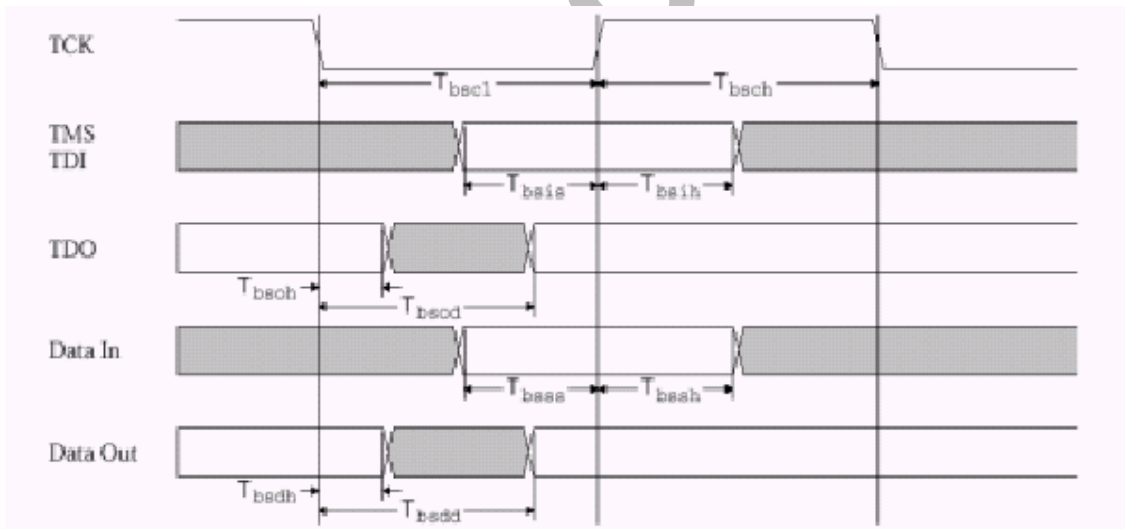


Figure 11-3: Boundary Scan General Timing

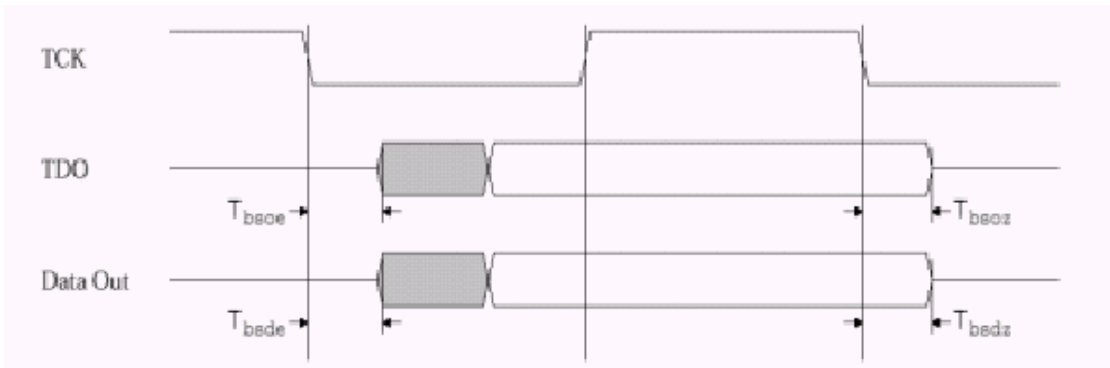


Figure 11-4: Boundary Scan Tristate Timing

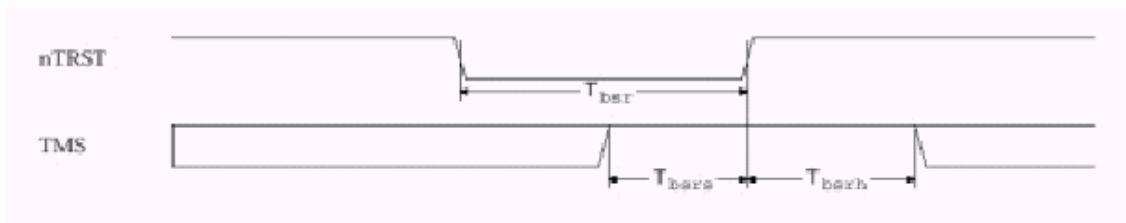


Figure 11-5: Boundary Scan Reset Timing

| Symbol | Parameter | Min | Max |
|--------|--|-----|-----|
| Tbscl | TCK low period | 50 | - |
| Tbsch | TCK high period | 50 | - |
| Tbsis | TMS, TDI setup to TCKr | 0 | - |
| Tbsih | TMS, TDI hold from TCKr | 2 | - |
| Tbsoh | TDO output hold from TCKf | 3 | - |
| Tbsod | TDO output delay from TCKf | - | 20 |
| Tbsss | Test mode Data in setup to TCKr | 2 | - |
| Tbssh | Test mode Data in hold from TCKf | 5 | - |
| Tbsdh | Test mode Data out hold from TCKf | 3 | - |
| Tbsdd | Test mode Data out delay from TCKf | - | 20 |
| Tbsoe | TDO output enable delay from TCKf | 2 | 15 |
| Tbsoz | Test mode Data enable delay from TCKf | 2 | 15 |
| Tbsde | TDO output disable delay from TCKf | 2 | 15 |
| Tbsdz | Test mode Data disable delay from TCKf | 2 | 15 |
| Tbsr | NTRST minimum pulse width | 25 | - |
| Tbsrs | TMS setup to nTRSTr | 20 | - |
| Tbsrh | TMS hold from nTRSTr | 20 | - |

The AC parameters are based on simulation results using 0.0pf circuit signal loads. Delays should be calculated using manufacturers output derating values for the actual circuit capacitance loading.

The correspondence between boundary-scan cells and system pins, system direction controls and system output enables is shown below. The cells are listed in the order in which they are connected in the boundary-scan register, starting with the cell closest to TDI. All outputs are three-state outputs. All boundary-scan register cells at input pins can apply tests to the on-chip system logic.

EXTEST/CLAMP guard values specified in the table below should be clocked into the boundary-scan register (using the SAMPLE/PRELOAD instruction) before the EXTEST, CLAMP or CLAMPZ instructions are selected to ensure that known data is applied to the system logic during the test. The INTEST guard values shown in the table below should be clocked into the boundary-scan register (using the SAMPLE/PRELOAD instruction) before the INTEST instruction is selected to ensure that all outputs are disabled. An asterisk in the guard

value column indicates that any value can be submitted (as test requires), but ones and zeros should always be placed as shown.

| Num | PAD Cell Name | PIN | TYPE | Output Enable BS Cell | Guard Value | |
|-----|---------------|-----------|-------|-----------------------|-------------|---|
| 1 | uLD4 | LD[4] | OUT | - | 0 | * |
| 2 | uLD3 | LD[3] | IN | - | * | * |
| 3 | uLD3 | LD[3] | OUT | - | * | * |
| 4 | uLD3 | - | OUTEN | LDPADOutEn[3] | 1 | * |
| 5 | uLD2 | LD[2] | IN | - | * | * |
| 6 | uLD2 | LD[2] | OUT | - | * | * |
| 7 | uLD2 | - | OUTEN | LDPADOutEn[2] | 1 | * |
| 8 | uLD1 | LD[1] | IN | - | * | * |
| 9 | uLD1 | LD[1] | OUT | - | * | * |
| 10 | uLD1 | - | OUTEN | LDPADOutEn[1] | 1 | * |
| 11 | uLD0 | LD[0] | IN | - | * | * |
| 12 | uLD0 | LD[0] | OUT | - | * | * |
| 13 | uLD0 | - | OUTEN | LDPADOutEn[0] | 1 | * |
| 14 | uKSCAN00 | KSCAN0[0] | IN | - | * | * |
| 15 | uKSCAN00 | KSCAN0[0] | OUT | - | * | * |
| 16 | uKSCAN00 | - | OUTEN | MuxPORTAOutEn[0] | 1 | * |
| 17 | uKSCAN01 | KSCAN0[1] | IN | - | * | * |
| 18 | uKSCAN01 | KSCAN0[1] | OUT | - | * | * |
| 19 | uKSCAN01 | - | OUTEN | MuxPORTAOutEn[1] | 1 | * |
| 20 | uKSCAN02 | KSCAN0[2] | IN | - | * | * |
| 21 | uKSCAN02 | KSCAN0[2] | OUT | - | * | * |
| 22 | uKSCAN02 | - | OUTEN | MuxPORTAOutEn[2] | 1 | * |
| 23 | uKSCAN03 | KSCAN0[3] | IN | - | * | * |
| 24 | uKSCAN03 | KSCAN0[3] | OUT | - | * | * |
| 25 | uKSCAN03 | - | OUTEN | MuxPORTAOutEn[3] | 1 | * |
| 26 | uKSCAN04 | KSCAN0[4] | IN | - | * | * |
| 27 | uKSCAN04 | KSCAN0[4] | OUT | - | * | * |
| 28 | uKSCAN04 | - | OUTEN | MuxPORTAOutEn[4] | 1 | * |
| 29 | uKSCAN05 | KSCAN0[5] | IN | - | * | * |
| 30 | uKSCAN05 | KSCAN0[5] | OUT | - | * | * |
| 31 | uKSCAN05 | - | OUTEN | MuxPORTAOutEn[5] | 1 | * |
| 32 | uKSCAN06 | KSCAN0[6] | IN | - | * | * |
| 33 | uKSCAN06 | KSCAN0[6] | OUT | - | * | * |
| 34 | uKSCAN06 | - | OUTEN | MuxPORTAOutEn[6] | 1 | * |
| 35 | uKSCAN07 | KSCAN0[7] | IN | - | * | * |
| 36 | uKSCAN07 | KSCAN0[7] | OUT | - | * | * |
| 37 | uKSCAN07 | - | OUTEN | MuxPORTAOutEn[7] | 1 | * |
| 38 | uKSCANI0 | KSCANI[0] | IN | - | * | * |
| 39 | uKSCANI0 | KSCANI[0] | OUT | - | * | * |
| 40 | uKSCANI0 | - | OUTEN | MuxPORTAOutEn[8] | 1 | * |
| 41 | uKSCANI1 | KSCANI[1] | IN | - | * | * |
| 42 | uKSCANI1 | KSCANI[1] | OUT | - | * | * |
| 43 | uKSCANI1 | - | OUTEN | MuxPORTAOutEn[9] | 1 | * |
| 44 | uKSCANI2 | KSCANI[2] | IN | - | * | * |
| 45 | uKSCANI2 | KSCANI[2] | OUT | - | * | * |
| 46 | uKSCANI2 | - | OUTEN | MuxPORTAOutEn[10] | 1 | * |
| 47 | uKSCANI3 | KSCANI[3] | IN | - | * | * |
| 48 | uKSCANI3 | KSCANI[3] | OUT | - | * | * |
| 49 | uKSCANI3 | - | OUTEN | MuxPORTAOutEn[11] | 1 | * |
| 50 | uKSCANI4 | KSCANI[4] | IN | - | * | * |
| 51 | uKSCANI4 | KSCANI[4] | OUT | - | * | * |
| 52 | uKSCANI4 | - | OUTEN | MuxPORTAOutEn[12] | 1 | * |
| 53 | uKSCANI5 | KSCANI[5] | IN | - | * | * |
| 54 | uKSCANI5 | KSCANI[5] | OUT | - | * | * |

| | | | | | | |
|-----|-------------|------------|-------|-------------------|---|---|
| 55 | uKSCANI5 | - | OUTEN | MuxPORTAOutEn[13] | 1 | * |
| 56 | uKSCANI6 | KSCANI[6] | IN | - | * | * |
| 57 | uKSCANI6 | KSCANI[6] | OUT | - | * | * |
| 58 | uKSCANI6 | - | OUTEN | MuxPORTAOutEn[14] | 1 | * |
| 59 | uKSCANI7 | KSCANI[7] | IN | - | * | * |
| 60 | uKSCANI7 | KSCANI[7] | OUT | - | * | * |
| 61 | uKSCANI7 | - | OUTEN | MuxPORTAOutEn[15] | 1 | * |
| 62 | uATSXP | ATSXP | IN | - | * | * |
| 63 | uATSXP | ATSXP | OUT | - | * | * |
| 64 | uATSXP | - | OUTEN | ATSXPEn | 1 | * |
| 65 | uATSXN | ATSXN | OUT | - | 0 | * |
| 66 | uATSXN | - | OUTEN | ATSXNEn | 1 | * |
| 67 | uATSYP | ATSYP | IN | - | * | * |
| 68 | uATSYP | ATSYP | OUT | - | * | * |
| 69 | uATSYP | - | OUTEN | ATSYPEn | 1 | * |
| 70 | uATSYN | ATSYN | IN | - | * | * |
| 71 | uATSYN | ATSYN | OUT | - | * | * |
| 72 | uATSYN | - | OUTEN | ATSYNEn | 1 | * |
| 73 | unPMWAKEUP | nPMWAKEUP | IN | - | * | 0 |
| 74 | unPOR | nPOR | IN | - | * | 0 |
| 75 | unRESET | nRESET | IN | - | * | * |
| 76 | unRESET | nRESET | OUT | - | * | * |
| 77 | unRESET | - | OUTEN | nRESETEn | 1 | * |
| 78 | uPMADAPOK | PMADAPOK | IN | - | * | 0 |
| 79 | uPMBATOK | PMBATOK | IN | - | * | 0 |
| 80 | unPLLENABLE | nPLLENABLE | IN | - | * | 0 |
| 81 | unURING | nURING | IN | - | * | * |
| 82 | unURING | nURING | OUT | - | * | * |
| 83 | unURING | - | OUTEN | MuxnPORTBOutEn[0] | 1 | * |
| 84 | unUDTR | nUDTR | IN | - | * | * |
| 85 | unUDTR | nUDTR | OUT | - | * | * |
| 86 | unUDTR | - | OUTEN | MuxnPORTBOutEn[1] | 1 | * |
| 87 | unUCTS | nUCTS | IN | - | * | * |
| 88 | unUCTS | nUCTS | OUT | - | * | * |
| 89 | unUCTS | - | OUTEN | MuxnPORTBOutEn[2] | 1 | * |
| 90 | unURTS | nURTS | IN | - | * | * |
| 91 | unURTS | nURTS | OUT | - | * | * |
| 92 | unURTS | - | OUTEN | MuxnPORTBOutEn[3] | 1 | * |
| 93 | unUDSR | nUDSR | IN | - | * | * |
| 94 | unUDSR | nUDSR | OUT | - | * | * |
| 95 | unUDSR | - | OUTEN | MuxnPORTBOutEn[4] | 1 | * |
| 96 | unUDCD | nUDCD | IN | - | * | * |
| 97 | unUDCD | nUDCD | OUT | - | * | * |
| 98 | unUDCD | - | OUTEN | MuxnPORTBOutEn[5] | 1 | * |
| 99 | uUSIN0 | USIN0 | IN | - | * | 0 |
| 100 | uUSOUT0 | USOUT0 | OUT | - | 0 | * |
| 101 | uUSIN1 | USIN1 | iN | - | * | 0 |
| 102 | uUSOUT1 | USOUT1 | OUT | - | 0 | * |
| 103 | uCANTx0 | CANTx[0] | IN | - | * | * |
| 104 | uCANTx0 | CANTx[0] | OUT | - | * | * |
| 105 | uCANTx0 | - | OUTEN | MuxnPORTCOutEn[1] | 1 | * |
| 106 | uCANRx0 | CANRx[0] | IN | - | * | * |
| 107 | uCANRx0 | CANRx[0] | OUT | - | * | * |
| 108 | uCANRx0 | - | OUTEN | MuxnPORTCOutEn[2] | 1 | * |
| 109 | uPORTB6 | PORTB[6] | IN | - | * | * |
| 110 | uPORTB6 | PORTB[6] | OUT | - | * | * |
| 111 | uPORTB6 | - | OUTEN | MuxnPORTBOutEn[6] | 1 | * |
| 112 | uPORTB7 | PORTB[7] | IN | - | * | * |
| 113 | uPORTB7 | PORTB[7] | OUT | - | * | * |

| | | | | | | |
|-----|-----------|-----------|-------|--------------------|---|---|
| 114 | uPORTB7 | - | OUTEN | MuxnPORTBOutEn[7] | 1 | * |
| 115 | uPORTB8 | PORTB[8] | IN | - | * | * |
| 116 | uPORTB8 | PORTB[8] | OUT | - | * | * |
| 117 | uPORTB8 | - | OUTEN | MuxnPORTBOutEn[8] | 1 | * |
| 118 | uPORTB9 | PORTB[9] | IN | - | * | * |
| 119 | uPORTB9 | PORTB[9] | OUT | - | * | * |
| 120 | uPORTB9 | - | OUTEN | MuxnPORTBOutEn[9] | 1 | * |
| 121 | uPORTB10 | PORTB[10] | IN | - | * | * |
| 122 | uPORTB10 | PORTB[10] | OUT | - | * | * |
| 123 | uPORTB10 | - | OUTEN | MuxnPORTBOutEn[10] | 1 | * |
| 124 | uPORTB11 | PORTB[11] | IN | - | * | * |
| 125 | uPORTB11 | PORTB[11] | OUT | - | * | * |
| 126 | uPORTB11 | - | OUTEN | MuxnPORTBOutEn[11] | 1 | * |
| 127 | uTimerOut | TimerOut | IN | - | * | * |
| 128 | uTimerOut | TimerOut | OUT | - | * | * |
| 129 | uTimerOut | - | OUTEN | MuxnPORTCcutEn[0] | 1 | * |
| 130 | uPSDAT | PSDAT | IN | - | * | * |
| 131 | uPSDAT | PSDAT | OUT | - | * | * |
| 132 | uPSDAT | - | OUTEN | MuxnPORTCcutEn[3] | 1 | * |
| 133 | uPSCLK | PSCLK | IN | - | * | * |
| 134 | uPSCLK | PSCLK | OUT | - | * | * |
| 135 | uPSCLK | - | OUTEN | MuxnPORTCcutEn[4] | 1 | * |
| 136 | uPWM0 | PWM[0] | IN | - | * | * |
| 137 | uPWM0 | PWM[0] | OUT | - | * | * |
| 138 | uPWM0 | - | OUTEN | MuxnPORTCcutEn[5] | 1 | * |
| 139 | uPWM1 | PWM[1] | IN | - | * | * |
| 140 | uPWM1 | PWM[1] | OUT | - | * | * |
| 141 | PWM1 | - | OUTEN | MuxnPORTCcutEn[6] | 1 | * |
| 142 | uCANTx1 | CANTx[1] | IN | - | * | * |
| 143 | uCANTx1 | CANTx[1] | OUT | - | * | * |
| 144 | uCANTx1 | - | OUTEN | - | 1 | * |
| 145 | uCANRx1 | CANRx[1] | IN | MuxnPORTEcutEn[23] | * | * |
| 146 | uCANRx1 | CANRx[1] | OUT | - | * | * |
| 147 | uCANRx1 | - | OUTEN | MuxnPORTEcutEn[22] | 1 | * |
| 148 | uMMCCMD | MMCCMD | IN | - | * | * |
| 149 | uMMCCMD | MMCCMD | OUT | - | * | * |
| 150 | uMMCCMD | - | OUTEN | MuxnPORTEcutEn[18] | 1 | * |
| 151 | uMMCDAT | MMCDAT | IN | - | * | * |
| 152 | uMMCDAT | MMCDAT | OUT | - | * | * |
| 153 | uMMCDAT | - | OUTEN | MuxnPORTEcutEn[19] | 1 | * |
| 154 | unMMCCD | nMMCCD | IN | - | * | * |
| 155 | unMMCCD | nMMCCD | OUT | - | * | * |
| 156 | unMMCCD | - | OUTEN | MuxnPORTEcutEn[20] | 1 | * |
| 157 | uMMCLK | MMCLK | IN | - | * | * |
| 158 | uMMCLK | MMCLK | OUT | - | * | * |
| 159 | uMMCLK | - | OUTEN | MuxnPORTEcutEn[21] | 1 | * |
| 160 | unDMAREQ | nDMAREQ | IN | - | * | * |
| 161 | unDMAREQ | nDMAREQ | OUT | - | * | * |
| 162 | unDMAREQ | - | OUTEN | MuxnPORTCOutEn[7] | 1 | * |
| 163 | unDMAACK | nDMAACK | IN | - | * | * |
| 164 | unDMAACK | nDMAACK | OUT | - | * | * |
| 165 | unDMAACK | - | OUTEN | MuxnPORTCOutEn[8] | 1 | * |
| 166 | unRCS3 | nRCS[3] | IN | - | * | * |
| 167 | unRCS3 | nRCS[3] | OUT | - | * | * |
| 168 | unRCS3 | - | OUTEN | MuxnPORTCOutEn[10] | 1 | * |
| 169 | unRCS2 | nRCS[2] | IN | - | * | * |
| 170 | unRCS2 | nRCS[2] | OUT | - | * | * |
| 171 | unRCS2 | - | OUTEN | MuxnPORTCOutEn[9] | 1 | * |
| 172 | unRCS1 | nRCS[1] | OUT | - | 0 | * |

| | | | | | | |
|-----|-----------|------------|-------|--------------------|---|---|
| 173 | unRCS0 | nRCS[0] | OUT | - | 0 | * |
| 174 | uBOOTBIT1 | BOOTBIT[1] | IN | - | * | 0 |
| 175 | uBOOTBIT0 | BOOTBIT[0] | IN | - | * | 0 |
| 176 | unROE | nROE | OUT | - | 0 | * |
| 177 | uEXPRDY | EXPRDY | IN | - | * | 0 |
| 178 | unRWE3 | nRWE[3] | IN | - | * | * |
| 179 | unRWE3 | nRWE[3] | OUT | - | * | * |
| 180 | unRWE3 | - | OUTEN | MuxnPORTEOutEn[17] | 1 | * |
| 181 | unRWE2 | nRWE[2] | IN | - | * | * |
| 182 | unRWE2 | nRWE[2] | OUT | - | * | * |
| 183 | unRWE2 | - | OUTEN | MuxnPORTEOutEn[16] | 1 | * |
| 184 | unRWE1 | nRWE[1] | OUT | - | 0 | * |
| 185 | unRWE0 | nRWE[0] | OUT | - | 0 | * |
| 186 | uRD31 | RD[31] | IN | - | * | * |
| 187 | uRD31 | RD[31] | OUT | - | * | * |
| 188 | uRD31 | - | OUTEN | MuxnPORTEOutEn[15] | 1 | * |
| 189 | uRD30 | RD[30] | IN | - | * | * |
| 190 | uRD30 | RD[30] | OUT | - | * | * |
| 191 | uRD30 | - | OUTEN | MuxnPORTEOutEn[14] | 1 | * |
| 192 | uRD29 | RD[29] | IN | - | * | * |
| 193 | uRD29 | RD[29] | OUT | - | * | * |
| 194 | uRD29 | - | OUTEN | MuxnPORTEOutEn[13] | 1 | * |
| 195 | uRD28 | RD[28] | IN | - | * | * |
| 196 | uRD28 | RD[28] | OUT | - | * | * |
| 197 | uRD28 | - | OUTEN | MuxnPORTEOutEn[12] | 1 | * |
| 198 | uRD27 | RD[27] | IN | - | * | * |
| 199 | uRD27 | RD[27] | OUT | - | * | * |
| 200 | uRD27 | - | OUTEN | MuxnPORTEOutEn[11] | 1 | * |
| 201 | uRD26 | RD[26] | IN | - | * | * |
| 202 | uRD26 | RD[26] | OUT | - | * | * |
| 203 | uRD26 | - | OUTEN | MuxnPORTEOutEn[10] | 1 | * |
| 204 | uRD25 | RD[25] | IN | - | * | * |
| 205 | uRD25 | RD[25] | OUT | - | * | * |
| 206 | uRD25 | - | OUTEN | MuxnPORTEOutEn[9] | 1 | * |
| 207 | uRD24 | RD[24] | IN | - | * | * |
| 208 | uRD24 | RD[24] | OUT | - | * | * |
| 209 | uRD24 | - | OUTEN | MuxnPORTEOutEn[8] | 1 | * |
| 210 | uRD23 | RD[23] | IN | - | * | * |
| 211 | uRD23 | RD[23] | OUT | - | * | * |
| 212 | uRD23 | - | OUTEN | MuxnPORTEOutEn[7] | 1 | * |
| 213 | uRD22 | RD[22] | IN | - | * | * |
| 214 | uRD22 | RD[22] | OUT | - | * | * |
| 215 | uRD22 | - | OUTEN | MuxnPORTEOutEn[6] | 1 | * |
| 216 | uRD21 | RD[21] | IN | - | * | * |
| 217 | uRD21 | RD[21] | OUT | - | * | * |
| 218 | uRD21 | - | OUTEN | MuxnPORTEOutEn[5] | 1 | * |
| 219 | uRD20 | RD[20] | IN | - | * | * |
| 220 | uRD20 | RD[20] | OUT | - | * | * |
| 221 | uRD20 | - | OUTEN | MuxnPORTEOutEn[4] | 1 | * |
| 222 | uRD19 | RD[19] | IN | - | * | * |
| 223 | uRD19 | RD[19] | OUT | - | * | * |
| 224 | uRD19 | - | OUTEN | MuxnPORTEOutEn[3] | 1 | * |
| 225 | uRD18 | RD[18] | IN | - | * | * |
| 226 | uRD18 | RD[18] | OUT | - | * | * |
| 227 | uRD18 | - | OUTEN | MuxnPORTEOutEn[2] | 1 | * |
| 228 | uRD17 | RD[17] | IN | - | * | * |
| 239 | uRD17 | RD[17] | OUT | - | * | * |
| 230 | uRD17 | - | OUTEN | MuxnPORTEOutEn[1] | 1 | * |
| 231 | uRD16 | RD[16] | IN | - | * | * |

| | | | | | | |
|-----|-------|--------|-------|-------------------|---|---|
| 232 | uRD16 | RD[16] | OUT | - | * | * |
| 233 | uRD16 | - | OUTEN | MuxnPORTEOutEn[0] | 1 | * |
| 234 | uRD15 | RD[15] | IN | - | * | * |
| 235 | uRD15 | RD[15] | OUT | - | * | * |
| 236 | uRD15 | - | OUTEN | nRDEn[1] | 1 | * |
| 237 | uRD14 | RD[14] | IN | - | * | * |
| 238 | uRD14 | RD[14] | OUT | jnRDEn[1] | * | * |
| 239 | uRD13 | RD[13] | IN | - | * | * |
| 240 | uRD13 | RD[13] | OUT | jnRDEn[1] | * | * |
| 241 | uRD12 | RD[12] | IN | - | * | * |
| 242 | uRD12 | RD[12] | OUT | jnRDEn[1] | * | * |
| 243 | uRD11 | RD[11] | IN | - | * | * |
| 244 | uRD11 | RD[11] | OUT | jnRDEn[1] | * | * |
| 245 | uRD10 | RD[10] | IN | - | * | * |
| 246 | uRD10 | RD[10] | OUT | jnRDEn[1] | * | * |
| 247 | uRD9 | RD[9] | IN | - | * | * |
| 248 | uRD9 | RD[9] | OUT | jnRDEn[1] | * | * |
| 249 | uRD8 | RD[8] | IN | - | * | * |
| 250 | uRD8 | RD[8] | OUT | jnRDEn[1] | * | * |
| 251 | uRD7 | RD[7] | IN | - | * | * |
| 252 | uRD7 | RD[7] | OUT | - | * | * |
| 253 | uRD7 | - | OUTEN | nRDEn[0] | 1 | * |
| 254 | uRD6 | RD[6] | IN | - | * | * |
| 255 | uRD6 | RD[6] | OUT | jnRDEn[0] | * | * |
| 256 | uRD5 | RD[5] | IN | - | * | * |
| 257 | uRD5 | RD[5] | OUT | jnRDEn[0] | * | * |
| 258 | uRD4 | RD[4] | IN | - | * | * |
| 259 | uRD4 | RD[4] | OUT | jnRDEn[0] | * | * |
| 260 | uRD3 | RD[3] | IN | - | * | * |
| 261 | uRD3 | RD[3] | OUT | jnRDEn[0] | * | * |
| 262 | uRD2 | RD[2] | IN | - | * | * |
| 263 | uRD2 | RD[2] | OUT | jnRDEn[0] | * | * |
| 264 | uRD1 | RD[1] | IN | - | * | * |
| 265 | uRD1 | RD[1] | OUT | jnRDEn[0] | * | * |
| 266 | uRD0 | RD[0] | IN | - | * | * |
| 267 | uRD0 | RD[0] | OUT | jnRDEn[0] | * | * |
| 268 | uRA0 | RA[0] | OUT | - | 0 | * |
| 269 | uRA1 | RA[1] | OUT | - | 0 | * |
| 270 | uRA2 | RA[2] | OUT | - | 0 | * |
| 271 | uRA3 | RA[3] | OUT | - | 0 | * |
| 272 | uRA4 | RA[4] | OUT | - | 0 | * |
| 273 | uRA5 | RA[5] | OUT | - | 0 | * |
| 274 | uRA6 | RA[6] | OUT | - | 0 | * |
| 275 | uRA7 | RA[7] | OUT | - | 0 | * |
| 276 | uRA8 | RA[8] | OUT | - | 0 | * |
| 277 | uRA9 | RA[9] | OUT | - | 0 | * |
| 278 | uRA10 | RA[10] | OUT | - | 0 | * |
| 279 | uRA11 | RA[11] | OUT | - | 0 | * |
| 280 | uRA12 | RA[12] | OUT | - | 0 | * |
| 281 | uRA13 | RA[13] | OUT | - | 0 | * |
| 282 | uRA14 | RA[14] | OUT | - | 0 | * |
| 283 | uRA15 | RA[15] | OUT | - | 0 | * |
| 284 | uRA16 | RA[16] | OUT | - | 0 | * |
| 285 | uRA17 | RA[17] | OUT | - | 0 | * |
| 286 | uRA18 | RA[18] | OUT | - | 0 | * |
| 287 | uRA19 | RA[19] | OUT | - | 0 | * |
| 288 | uRA20 | RA[20] | OUT | - | 0 | * |
| 289 | uRA21 | RA[21] | OUT | - | 0 | * |
| 290 | uRA22 | RA[22] | OUT | - | 0 | * |

| | | | | | | |
|-----|--------|---------|-------|--------------------|---|---|
| 291 | uRA23 | RA[23] | OUT | - | 0 | * |
| 292 | uRA24 | RA[24] | IN | - | * | * |
| 293 | uRA24 | RA[24] | OUT | - | * | * |
| 294 | uRA24 | - | OUTEN | MuxnPORTEOutEn[24] | 1 | * |
| 295 | uSA3 | SA[3] | OUT | - | 0 | * |
| 296 | uSA4 | SA[4] | OUT | - | 0 | * |
| 297 | uSA2 | SA[2] | OUT | - | 0 | * |
| 298 | uSA5 | SA[5] | OUT | - | 0 | * |
| 299 | uSA1 | SA[1] | OUT | - | 0 | * |
| 300 | uSA6 | SA[6] | OUT | - | 0 | * |
| 301 | uSA0 | SA[0] | OUT | - | 0 | * |
| 302 | uSA7 | SA[7] | OUT | - | 0 | * |
| 303 | uSA8 | SA[8] | OUT | - | 0 | * |
| 304 | uSA9 | SA[9] | OUT | - | 0 | * |
| 305 | uSA10 | SA[10] | OUT | - | 0 | * |
| 306 | uSA11 | SA[11] | OUT | - | 0 | * |
| 307 | uSA12 | SA[12] | OUT | - | 0 | * |
| 308 | uSA13 | SA[13] | OUT | - | 0 | * |
| 309 | uSA14 | SA[14] | OUT | - | 0 | * |
| 310 | unSCS1 | nSCS[1] | OUT | - | 0 | * |
| 311 | unSCS0 | nSCS[0] | OUT | - | 0 | * |
| 312 | unSRAS | nSRAS | OUT | - | 0 | * |
| 313 | unRCAS | nSCAS | OUT | - | 0 | * |
| 314 | unSWE | nSWE | OUT | - | 0 | * |
| 315 | uSCKE1 | SCKE[1] | OUT | - | 0 | * |
| 316 | uSCKE0 | SCKE[0] | OUT | - | 0 | * |
| 317 | uSCLK | SCLK | IN | - | * | * |
| 318 | uSCLK | SCLK | OUT | - | * | * |
| 319 | uSCLK | - | OUTEN | 1'b0 | 1 | * |
| 320 | uSDQMU | SDQMU | OUT | - | 0 | * |
| 321 | uSDQML | SDQML | OUT | - | 0 | * |
| 322 | uSD8 | SD[8] | IN | - | * | * |
| 323 | uSD8 | SD[8] | OUT | jnSDEn | * | * |
| 324 | uSD7 | SD[7] | IN | - | * | * |
| 325 | uSD7 | SD[7] | OUT | jnSDEn | * | * |
| 326 | uSD9 | SD[9] | IN | - | * | * |
| 327 | uSD9 | SD[9] | OUT | jnSDEn | * | * |
| 328 | uSD6 | SD[6] | IN | - | * | * |
| 329 | uSD6 | SD[6] | OUT | jnSDEn | * | * |
| 330 | uSD10 | SD[10] | IN | - | * | * |
| 331 | uSD10 | SD[10] | OUT | jnSDEn | * | * |
| 332 | uSD5 | SD[5] | IN | - | * | * |
| 333 | uSD5 | SD[5] | OUT | jnSDEn | * | * |
| 334 | uSD11 | SD[11] | IN | - | * | * |
| 335 | uSD11 | SD[11] | OUT | jnSDEn | * | * |
| 336 | uSD4 | SD[4] | IN | - | * | * |
| 337 | uSD4 | SD[4] | OUT | jnSDEn | * | * |
| 338 | uSD12 | SD[12] | IN | - | * | * |
| 339 | uSD12 | SD[12] | OUT | jnSDEn | * | * |
| 340 | uSD3 | SD[3] | IN | - | * | * |
| 341 | uSD3 | SD[3] | OUT | jnSDEn | * | * |
| 342 | uSD13 | SD[13] | IN | - | * | * |
| 343 | uSD13 | SD[13] | OUT | jnSDEn | * | * |
| 344 | uSD2 | SD[2] | IN | - | * | * |
| 345 | uSD2 | SD[2] | OUT | jnSDEn | * | * |
| 346 | uSD14 | SD[14] | IN | - | * | * |
| 347 | uSD14 | SD[14] | OUT | jnSDEn | * | * |
| 348 | uSD1 | SD[1] | IN | - | * | * |
| 349 | uSD1 | SD[1] | OUT | jnSDEn | * | * |

| | | | | | | |
|-----|--------|--------|-------|-------------------|---|---|
| 350 | uSD15 | SD[15] | IN | - | * | * |
| 351 | uSD15 | SD[15] | OUT | - | * | * |
| 352 | usD15 | - | OUTEN | nSDEn | 1 | * |
| 353 | uSD0 | SD[0] | IN | - | * | * |
| 354 | uSD0 | SD[0] | OUT | jnSDEn | * | * |
| 355 | uLLP | LLP | OUT | - | 0 | * |
| 356 | uLAC | LAC | OUT | - | 0 | * |
| 367 | uLBLEN | LBLEN | IN | - | * | * |
| 358 | uLBLEN | LBLEN | OUT | - | * | * |
| 359 | uLBLEN | - | OUTEN | MuxnPORTDOutEn[8] | 1 | * |
| 360 | uLCP | LCP | OUT | - | 0 | * |
| 361 | uLFP | LFP | OUT | - | 0 | * |
| 362 | uLCDEN | LCDEN | OUT | - | 0 | * |
| 363 | uLD15 | LD[15] | IN | - | * | * |
| 364 | uLD15 | LD[15] | OUT | - | * | * |
| 365 | uLD15 | - | OUTEN | MuxnPORTDOutEn[7] | 1 | * |
| 366 | uLD14 | LD[14] | IN | - | * | * |
| 367 | uLD14 | LD[14] | OUT | - | * | * |
| 368 | uLD14 | - | OUTEN | MuxnPORTDOutEn[6] | 1 | * |
| 369 | uLD13 | LD[13] | IN | - | * | * |
| 370 | uLD13 | LD[13] | OUT | - | * | * |
| 371 | uLD13 | - | OUTEN | MuxnPORTDOutEn[5] | 1 | * |
| 372 | uLD12 | LD[12] | IN | - | * | * |
| 373 | uLD12 | LD[12] | OUT | - | * | * |
| 374 | uLD12 | - | OUTEN | MuxnPORTDOutEn[4] | 1 | * |
| 375 | uLD11 | LD[11] | IN | - | * | * |
| 376 | uLD11 | LD[11] | OUT | - | * | * |
| 377 | uLD11 | - | OUTEN | MuxnPORTDOutEn[3] | 1 | * |
| 378 | uLD10 | LD[10] | IN | - | * | * |
| 379 | uLD10 | LD[10] | OUT | - | * | * |
| 380 | uLD10 | - | OUTEN | MuxnPORTDOutEn[2] | 1 | * |
| 381 | uLD9 | LD[9] | IN | - | * | * |
| 382 | uLD9 | LD[9] | OUT | - | * | * |
| 383 | uLD9 | - | OUTEN | MuxnPORTDOutEn[1] | 1 | * |
| 384 | uLD8 | LD[8] | IN | - | * | * |
| 385 | uLD8 | LD[8] | OUT | - | * | * |
| 386 | uLD8 | - | OUTEN | MuxnPORTDOutEn[0] | 1 | * |
| 387 | uLD7 | LD[7] | OUT | - | 0 | * |
| 388 | uLD6 | LD[6] | OUT | - | 0 | * |
| 389 | uLD5 | LD[5] | OUT | - | 0 | * |

11.4 Production Test Features

In order to generate test vectors suitable for use on a production tester by the chip manufacturer, some special test modes have been introduced. These modes come into operation whenever the pin nTEST is forced LOW.

Full details of these modes are available from ARM in a special Test Document on request.

12 ELECTRICAL CHARACTERISTICS

12.1 Absolute Maximum Ratings

| Symbol | Parameter | Min | Max | Units |
|------------------|----------------------|------|-----|-------|
| V _{DD} | Power Supply Voltage | -0.5 | 4.6 | V |
| V _{IN} | DC Input Voltage | -0.3 | 6 | V |
| I _{IN} | DC Input Current | -50 | 50 | mA |
| T _{STG} | Storage Temperature | -65 | 150 | °C |

Note : Permanent damage can be occur if maximum ratings are exceeded.
 Device modules may not operate normally while being exposed to electrical extremes.
 Although sections of the device contain circuitry to protect against damages from high static voltages or electrical fields, take normal pre-cautions to avoid exposure to voltages higher than maximum rated voltages.

Recommended Operating Range

| Symbol | Parameter | Min | Max | Units |
|------------------|--|------------|------------|--------|
| VDD (3.3V) | DC Power Supply Voltage (3.3V) → use for I/O | 3.0 2.3 | 3.6 2.7 | V V |
| VDD (2.5V) | DC Power Supply Voltage (2.5V) → use for a Core | -40 | 85 | °C |
| T _{OPR} | Operating Temperature (Industrial Temperature) | | | |

12.2 DC characteristics

All characteristics are specified at $V_{DD} = 3.0$ to $3.6V$ and $V_{SS} = 0V$ over the junction temperature range of 0 to $100\text{ }^{\circ}C$.

Power Dissipation

| Symbol | Parameter | Min | Max | Units |
|-----------|-----------------------|-----|-----|---------|
| P_D | [Run Mode] | | | |
| | With LCD @70.04MHz | | 190 | mW |
| P_{DWN} | Without LCD @70.04MHz | | 140 | mW |
| | [Deep Sleep Mode] | | | |
| | RTC Enable | 120 | 160 | μW |
| | RTC Disable | 30 | 70 | μW |

CMOS/TTL Compatible Pin

| Symbol | Parameter | Min | Max | Conditions |
|----------|--------------------------------|--------------|--------------|-----------------------------------|
| V_{IL} | Low-level Input Voltage | | $0.3XV_{DD}$ | Guaranteed Input Low Voltage |
| V_{IH} | High-level Input Voltage | $0.7XV_{DD}$ | | Guaranteed Input High Voltage |
| V_{OL} | Low-level Output Voltage | | 0.4 V | $I_{OL} = 1\text{ mA}$ (*Group A) |
| | | | 0.4 V | $I_{OL} = 2\text{ mA}$ (Group B) |
| | | | 0.4 V | $I_{OL} = 4\text{ mA}$ (Group C) |
| V_{OH} | High-level Output Voltage | 2.4 V | | $I_{OH} = -1\text{ mA}$ (Group A) |
| | | 2.4 V | | $I_{OH} = -2\text{ mA}$ (Group B) |
| | | 2.4 V | | $I_{OH} = -4\text{ mA}$ (Group C) |
| I_{IL} | Input Low Current | -10 μA | 10 μA | $V_{IN} = V_{SS}$ |
| I_{IH} | Input High Current | -10 μA | 10 μA | $V_{IN} = V_{DD}$ |
| I_{OZ} | 3-state Output Leakage Current | -10 μA | 10 μA | $V_{PAD} = V_{SS}$ OR V_{DD} |

* : It means the drive strength (Group A = 1, Group B = 2, Group C = 4)
Refer to GPIO part (page 122)

I/O Circuit Pull-up Pin

The following current values are used for I/Os with internal pull-up devices.

| Symbol | Parameter | Min($V_{IN} = V_{SS}$) | Max($V_{IN} = V_{DD}$) |
|----------|-----------|--------------------------|--------------------------|
| I_{PU} | Pull-up | -100 μA | - 4 μA |

Note : The following pins are used with internal pull-up devices.
TDI, TCK, TMS, PMADAOK, PMBATOK, nTEST, nPMWAKEUP

I/O Circuit Pull-down Pin

The following current values are used for I/Os with internal pull-down devices.

| Symbol | Parameter | Min($V_{IN} = V_{SS}$) | Max($V_{IN} = V_{DD}$) |
|----------|-----------|--------------------------|--------------------------|
| I_{PD} | Pull-down | 4 μA | 100 μA |

Note : The following pins are used with internal pull-down devices.
nTRST, TESTSCAN, nPllenable, SCAN_EN

12.3 A/D Converter Electrical Characteristics

| Symbol | Parameter | Test Condition | Min | Typ | Max | Unit |
|-----------------------|----------------------------------|---|----------|------|-----------|------|
| I _{dd} | Normal | aclk=8MHz * Input=AV _{ref} V fin=2KHz ramp | | 6.0 | | mA |
| | Power Down | aclk=8MHz | | 60 | | uA |
| An** | Analog Input Voltage | | AVSS+0.2 | | AVref-0.2 | V |
| Accuracy | Resolution | | | | 10 | Bits |
| INL | Integral Non-linearity | aclk=8MHz Input=0 - AV _{ref} V fin=2KHz ramp | | ±2.0 | | LSB |
| DNL | Differential Non-linearity | aclk=8MHz Input=0 - AV _{ref} V fin=2KHz ramp | | ±1.0 | | LSB |
| SNR | Signal-to-Noise Ratio | F _{sample} = 500Ksps fin = 2KHz | 51 | 54 | | dB |
| SNDR | Signal-to-Noise Distortion Ratio | | 49 | 52 | | dB |
| aclk | | | 2 | 4 | 8 | MHz |
| t _c | Conversion Time | t _c = [aclk/16] ⁻¹ | 2 | 4 | 8 | us |
| AV _{ref} *** | Analog Reference Voltage | | | | AVDD | V |
| T _{cal} | Power-up Time | Calibration Time | | 22 | | ms |
| THD | Total Harmonic Distortion | | 51 | 54 | | dB |
| AVDD | Analog Power | | 3.0 | 3.3 | 3.6 | V |
| DVDD | Digital Power | | 3.0 | 3.3 | 3.6 | V |
| fin | Analog Input Frequency | | | 5 | | KHz |

(For Test, Analog Input Freq. = 2KHz, aclk=8MHz, AVDD=DVDD=AV_{ref}=3.3V, Temperature=25°C)

aclk : To determine electrical characteristic of ADC, used 8MHz clock as aclk.
but for 7202 ADC, used 3.6864MHz for aclk.

an* : Analog input is sample and hold with 500Ω resistor and 300 fF capacitor in series and connected with gate of CMOS transistor.

So, in normal, input resistance of an analog input pin has a couple of Mega Ohms.

AVref** : The equivalent impedance of AVREF is about 5kΩ of resistance to GND.

12.4 D/A Converter Electrical Characteristics

| Symbol | Parameter | Test Condition | Min | Typ | Max | Unit |
|----------------|----------------------------------|-------------------------------------|-------|------|-------|------|
| I_{dd} | Normal | $f_{CLK}=50KHz$ | 3.6 | 4.1 | 4.6 | mA |
| | Power Down | TBD | | | | uA |
| Accuracy | Resolution | | | 8 | | Bits |
| INL | Integral Non-linearity | DC | -0.6 | | +0.6 | LSB |
| DNL | Differential Non-linearity | DC | -0.2 | | +0.2 | LSB |
| SNR | Signal-to-Noise Ratio | $f_{con}=50KHz$ Temperature=25°C | 47.5 | 47.7 | 47.8 | dB |
| SNDR | Signal-to-Noise Distortion Ratio | | 47.1 | 47.4 | 47.7 | dB |
| THD | Total Harmonic Distortion | | 57.5 | 61.8 | 65.9 | dB |
| f_{con} | Conversion Speed | | | 50 | | KHz |
| tr/tf | rise/fall time | with $\pm 10\%$ error | | 0.4 | | us |
| $V_{out(p-p)}$ | Output Voltage Range | | 1.025 | | 2.675 | V |
| t_d | Output Delay Time | | | 1.4 | | Us |

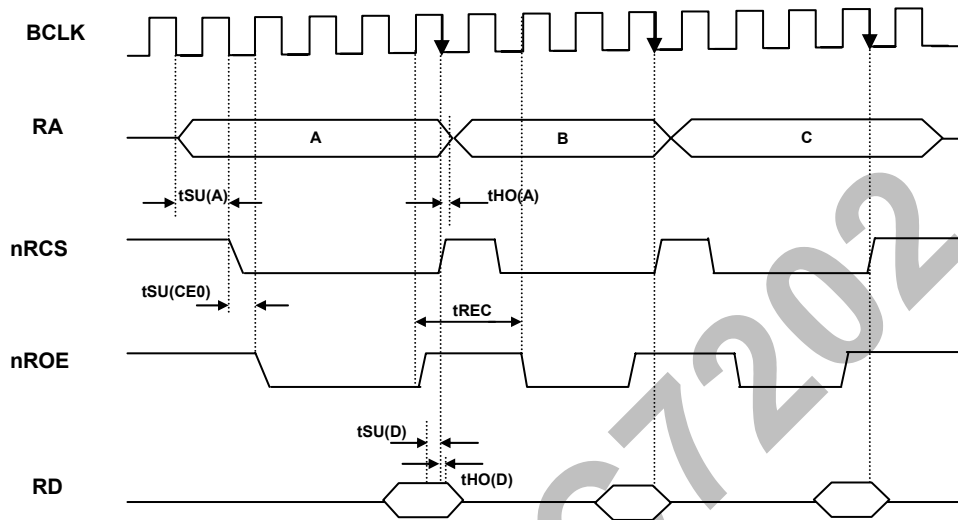
The current drive capability is about 500uA on output of DAC.

Typical load is about 10k Ω of resistance and 10pF of capacitance on output of DAC.

12.5 AC Characteristics

12.5.1 Static Memory Interface

12.5.1.1 READ Access Timing (Single Mode)



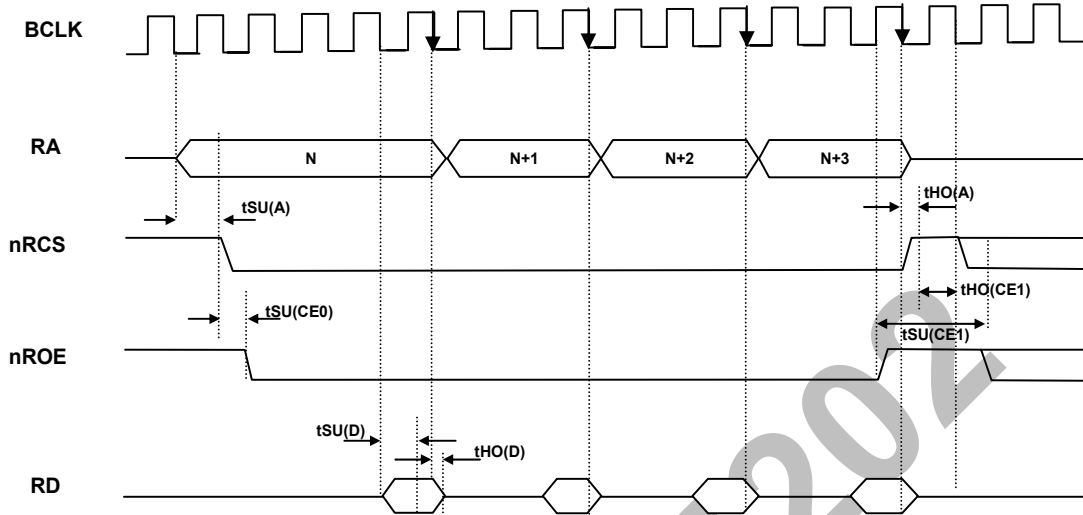
| Symbol | Parameter | Min | Max | Unit |
|----------|--|-----|-----|------|
| tSU(A) | Address to nRCS falling-edge setup time | 25 | | ns |
| tHO(A) | nROE rising-edge to Address hold time | 0 | | |
| tSU(CE0) | nRCS falling-edge to nROE falling-edge setup time | 13 | | |
| tHO(CE0) | nROE rising-edge to nRCS rising-edge setup time | -13 | | |
| tHO(CE1) | nROE or nRWE rising-edge to nRCS falling-edge hold time | 15 | | |
| tSU(CE1) | nRCS rising-edge to nROE or nRWE falling-edge setup time | 25 | | |
| tREC | nROE negate to start of next cycle | 50 | | |
| tSU(D) | Data setup time before latch | 5 | | |
| tHO(D) | Data hold time after latch | 0 | | |

Timing values for read access in single mode data transfer

Memory Configuration Register Setting = 0x060

| | | | | | | | | | | | | | | | |
|--|--|--|--|----|----|---|---|---|---|---|---|---|---|---|---|
| | | | | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | | | | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |

12.5.1.2 READ Access Timing (Burst Mode)



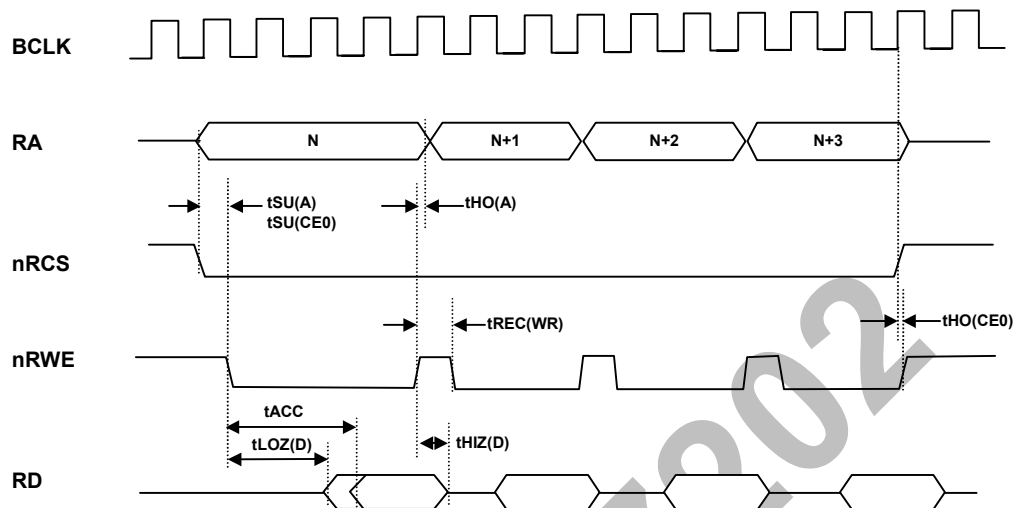
| Symbol | Parameter | Min | Max | Unit |
|----------|--|-----|-----|------|
| tSU(A) | Address to nRCS falling-edge setup time | 13 | | ns |
| tHO(A) | nROE rising-edge to Address hold time | -15 | | |
| tSU(CE0) | nRCS falling-edge to nROE falling-edge setup time | 13 | | |
| tHO(CE0) | nROE rising-edge to nRCS rising-edge setup time | -13 | | |
| tHO(CE1) | nROE or nRWE rising-edge to nRCS falling-edge hold time | 25 | | |
| tSU(CE1) | nROE or nRWE rising-edge to nRCS falling-edge setup time | 50 | | |
| tSU(D) | Data setup time before latch | 5 | | |
| tHO(D) | Data hold time after latch | 0 | | |

Timing values for read access in burst mode data transfer

Memory Configuration Register Setting = 0xE00

| | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--|----|----|---|---|---|---|---|---|---|---|---|---|
| | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

12.5.1.3 WRITE Access Timing



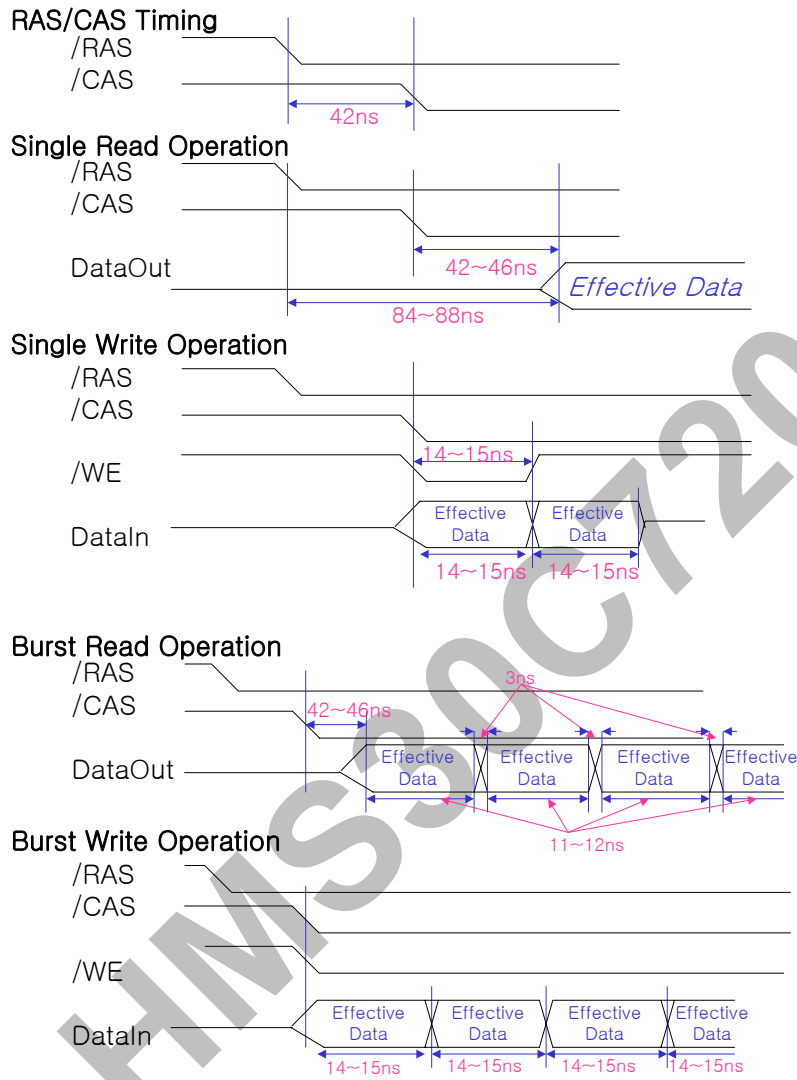
| Symbol | Parameter | Min | Max | Unit |
|----------|--|-----|-----|------|
| tSU(A) | Address to nRWE falling-edge setup time | 15 | | ns |
| tHO(A) | nRWE rising-edge to Address hold time | 0 | | |
| tSU(CE0) | nRCS falling-edge to nRWE falling-edge setup time | 15 | | |
| tHO(CE0) | nRWE rising-edge to nRCS rising-edge setup time | 27 | | |
| tHO(CE1) | nROE or nRWE rising-edge to nRCS falling-edge hold time | 39 | | |
| tSU(CE1) | nRCS rising-edge to nROE or nRWE falling-edge setup time | 25 | | |
| tREC(WR) | nRWE negate to start of next cycle | 26 | | |
| tHIZ(D) | nRWE rising edge to D Hi-Z delay | 25 | | |
| tACC | write access time | 4.5 | | |
| tLOZ(D) | nRWE falling-edge to D driven | 0 | | |

Timing values for write access

Memory Configuration Register Setting = 0x068

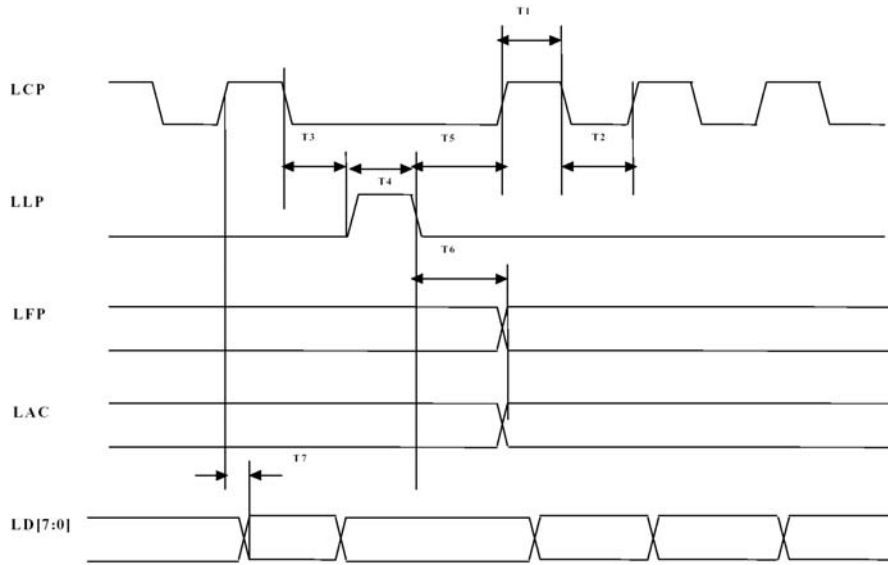
| | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--|----|----|---|---|---|---|---|---|---|---|---|---|
| | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 |

12.5.2 SDRAM Interface

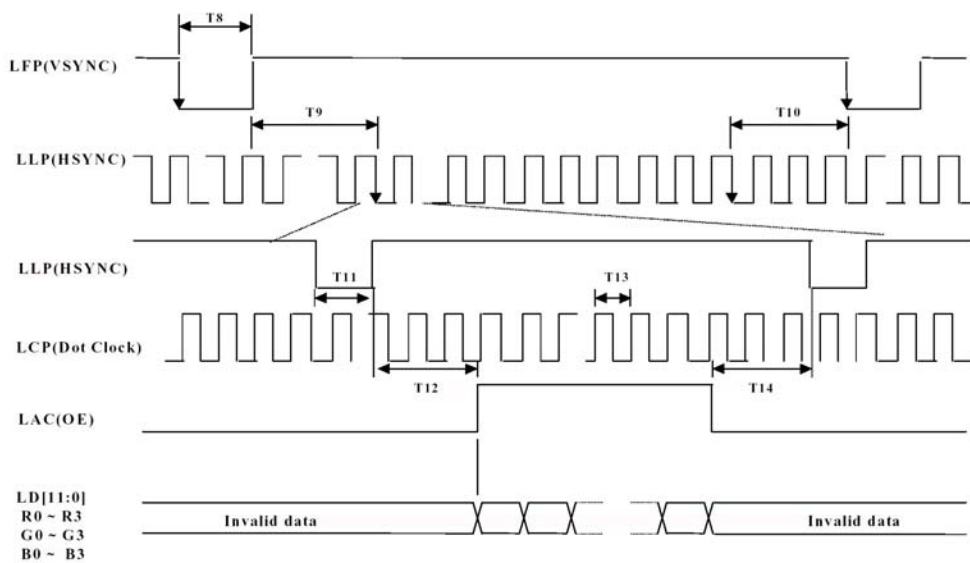


Condition : 70MHz CPU clock speed

12.5.3 LCD Interface



LCD Controller Timing(STN Mode)



LCD Controller Timing(Active-TFT Mode)

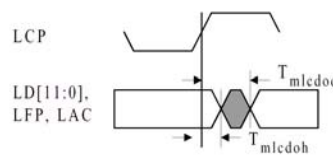
| Symbol | Parameter | Min | Typ | Max | Unit |
|--------|--------------------------------|-----|-----|-----|-------------|
| T1 | LCP High Time | 1 | - | 16 | tCLK(Notes) |
| T2 | LCP Low Time | 1 | - | 17 | tCLK |
| T3 | LLP Front-Porch | 1 | - | 256 | tCLK |
| T4 | LLP Pulse Width | 1 | - | 256 | tCLK |
| T5 | LLP Back-Porch | 1 | - | 256 | tCLK |
| T6 | Falling LLP to LFP(LAC) Toggle | 1 | - | 256 | tCLK |

| | | | | | |
|-----|-----------------------------------|-----|---|-----|-----------------|
| T7 | Rising LCP to Display Data Change | TBD | | TBD | ns |
| T8 | VSYNC Width | 1 | | 64 | tHperiod(Notes) |
| T9 | VSYNC Back-Porch | 1 | | 256 | tHperiod |
| T10 | VSYNC Front-Porch | 1 | | 256 | tHperiod |
| T11 | HSYNC Width | 1 | | 256 | tCLK |
| T12 | HSYNC Back-Porch | 1 | - | 256 | tCLK |
| T13 | HSYNC Front-Porch | 1 | - | 256 | tCLK |
| T14 | Dot Clock Period | 1 | - | - | tCLK |

LCD Interface Signal Timing Parameters

Note : tCLK is BCLK or VCLK(LCD Controller Internal Clock Source : 31.5 or 40 MHz).

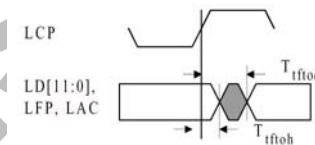
tHperiod Max = 1408 tCLK


STN Mode Signal Delay

| Symbol | Parameter | Min | Max |
|---------|-----------------------------------|-----|-----|
| Tmlcdod | Output Delay Time from LCP rising | - | 5 |
| Tmlcdoh | Output Hold Time from LCP Rising | - | -5 |

STN Mode Signal Delay Parameters

Timing values are derived from simulations using 0pF signal loading. Actual circuit output delays should be calculated by adding manufacturers signal load de-rating delay values.

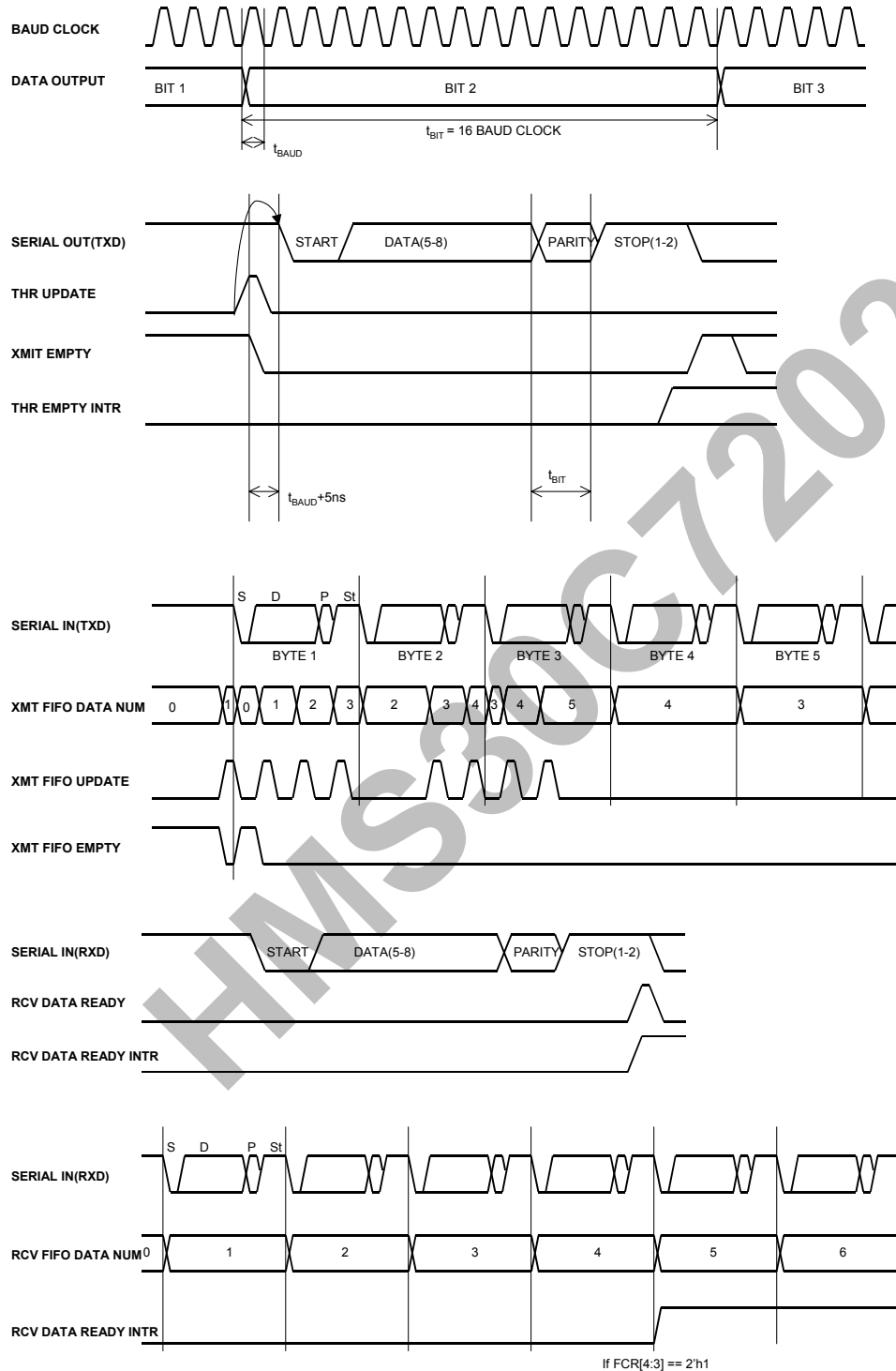

TFT Mode Signal Delay

| Symbol | Parameter | Min | Max |
|--------|-----------------------------------|-----|-----|
| Ttftod | Output Delay Time from LCP rising | - | 3 |
| Ttftoh | Output Hold Time from LCP Rising | - | -3 |

TFT Mode Signal Delay Parameters

Timing values are derived from simulations using 0pF signal loading. Actual circuit output delays should be calculated by adding manufacturers signal load de-rating delay values.

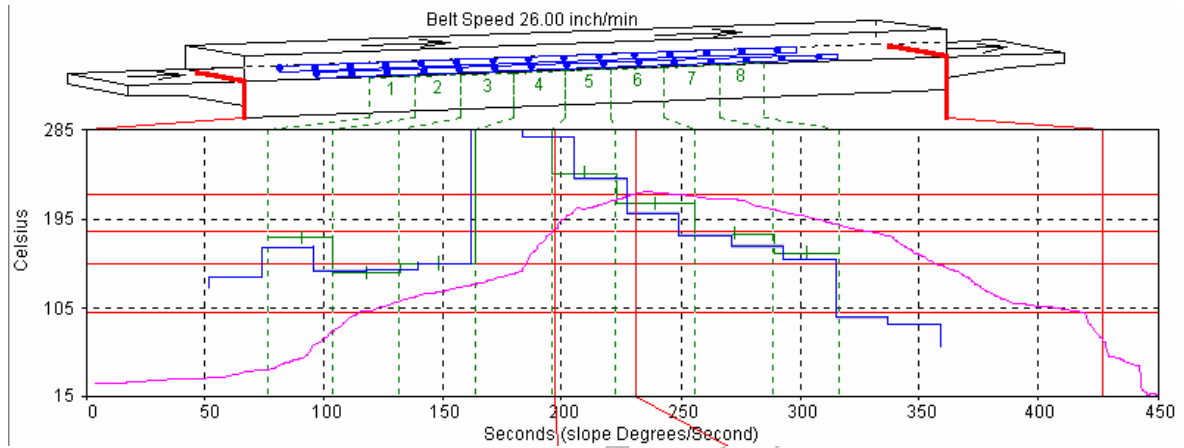
12.5.4 UART(Universal Asynchronous Receiver Transmitter)



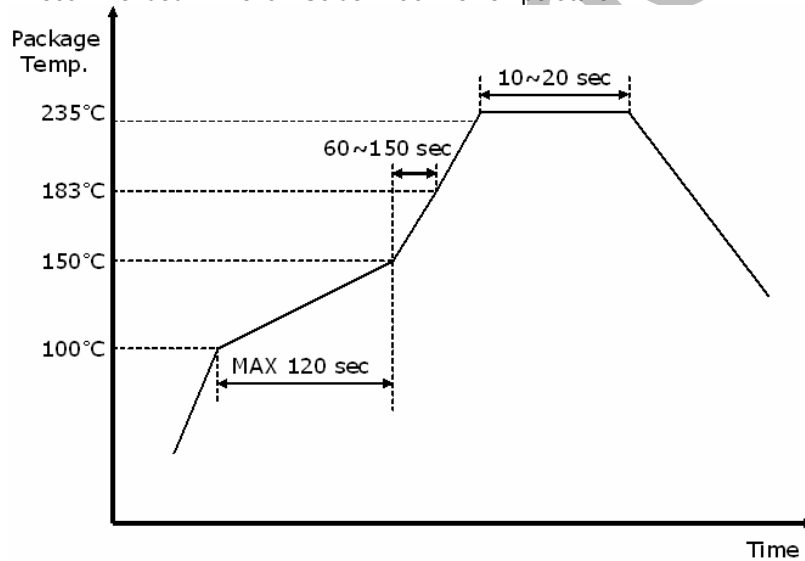
12.6 Package

12.6.1 Recommended Soldering Conditions

12.6.1.1 MQFP(Metric Quad Flat Pack) Type



- Recommended IP-Reflow Solder Machine Temperature

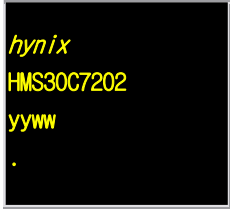
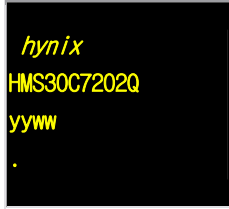


12.6.1.2 FBGA(Chip Array Ball Grid Array) Type

The soldering condition of FBGA type package is the same as that of MQFP type package.

- Recommended IP-Reflow Solder Machine Temperature

12.6.2 Pictures of Package Marking

| Package Type | 256FBGA | 256MQFP |
|-----------------|---|--|
| Package Marking |  |  |

HMS30C7202

13 APPENDIX

13.1 Deep-sleep, Wake-up Issues of HMS30C7202 PMU

13.1.1 Wake-up

HMS30C7202 has four external wake-up sources, and at least one of two power condition pins (PMADAPOK, PMBATOK) should be high. MRING (nURING), nPMWAKEUP, RTC event can not be masked. PMU only has interrupt mask bits for interrupt controller. It means even though HMS30C7202 wake-up from deep-sleep, there might be no interrupt for interrupt controller. But every time, HMS30C7202 would wake up when any one of wake-up sources asserted.

- **Wake-up sources**

MRING : It's connected nURING pin ("n" of nURING pin means "low active")

This signal can not be masked in PMU.

HOTSYNC : HotSync condition or user defined condition (ex. Plugging power adaptor)

This signal is connected with GPIOB[10] interrupt.

nRESET : nRESET signal wake up from deep-sleep.

nPMWAKEUP : active low external signal. This signal can not be masked.

RTC Event: from RTC. This signal isn't able to mask in PMU.

All wake-up sources are filtered by debounce circuit (except RTC) with 250Hz clock from RTC clock source, so if RTC clock stopped, wake-up sequence would not work.

- **Needed condition for wake-up**

One of PMADAPOK and PMBATOK should be high, it means there's no power problem.

If user wants to make wake-up regardless power source condition, set "WAKEUP" bit of PMU Mode register (PMUMODE) bit [3].

13.1.2 Deep-sleep

- To go deep-sleep state, all wake-up conditions are cleared. If any wake-up pin stays in wake-up condition, 7202 would not go into "deep-sleep mode".

- Once Deep-sleep mode is set (in Slow mode) and no wake-up signal condition, State machine wait, until *Bus Idle* state. And after state machine jump into *Bus Idle*, in the very next "bus access" operation, PMU get bus mastership from CPU and state machine keep going into deep-sleep mode through short sleep state. Sometimes S/W need to wait until *Bus Idle*(ex. DMA cases) and to prevent un-wanted next instruction execution after deep-sleep instruction set PMU Mode Register(PMUMODE), usually dummy loop is used for this purpose.

- In some cases (in some S/W), to keep going into deep-sleep, dummy bus (ex. just single read of a peripheral register) access is helpful after dummy loop. We think it is related with changing bus mastership. (or may need longer dummy loop) But we can't sure it.