NUC501
User's Manual

Publication Release Date: Sep. 2011

Table of Contents

1 General Description 6

2 Feature 6

3 Pad and Pin Configuration 9

4 System Diagram 19

5 Block Diagram 20

5.1 System block diagram 20

5.2 On-Chip Bus block diagram 21

6 Functional Description 22

6.1 ARM7TDMI CPU Core 22

6.2 System Manager 23

6.2.1 Overview 23

6.2.2 System Memory Mapping 23

6.2.3 AHB Bus Arbitration 25

6.2.4 Fixed Priority Mode 25

6.2.5 Power-On Settings 26

6.2.6 System Manager Control Registers 27

6.3 Clock Controller 45

6.3.1 Function Description 45

6.3.2 Clock Control Registers 45

6.4 SPI Synchronous Serial Interface Controller (Master Mode) 58

6.4.1 Overview 58

6.4.2 Features 58

6.4.3 SPIM Timing Diagram 59

6.4.4 SPIM Programming Example without DMA 59

6.4.5 SPIM Programming Example with DMA 60

6.4.6 Direct memory mapping mode 60

6.4.7 SPIM Serial Interface Control Registers Mapping 62

6.5 Audio Processing Unit 76

6.5.1 Overview and Features 76

6.5.2 APU Functional Description 76

6.5.3 AUDIO DAC Clock 76

6.5.4 APU Run Procedures 76

6.5.5 APU Control Register Mapping 78

6.5.6 APU Control Registers 79
6.6 SRAM Controller 87
6.6.1 Overview 87
6.6.2 Features 87
6.6.3 SRAM Block Diagram 88
6.6.4 SRAM System Diagram 89
6.6.5 SRAM Function Description 90
6.6.6 SRAM Register Mapping 91
6.7 USB Device Controller 94
6.7.1 Overview 94
6.7.2 Features 94
6.7.3 Functional Descriptions 95
6.7.4 Memory Mapping 96
6.7.5 USB Control Registers Mapping 97
6.9 Advanced Interrupt Controller 113
6.9.1 Overview 113
6.9.2 Features 113
6.9.3 Interrupt Sources 114
6.9.4 AIC Functional Descriptions 116
6.9.5 AIC Registers Mapping 118
6.9.6 AIC Control Registers 120
6.10 General Purpose I/O 134
6.10.1 Overview and Features 134
6.10.2 GPIO Control Register Mapping 135
6.10.3 GPIO Control Register Description 136
6.11 I2C Synchronous Serial Interface 157
6.11.1 Overview 157
6.11.2 Feature 157
6.11.3 I^2C Protocol 158
6.11.4 I^2C Programming Examples 160
6.11.5 Software I^2C Operation 162
6.11.6 I^2C Serial Interface Control Registers Mapping 164
6.12 PWM-Timer 172
6.12.1 Introduction 172
6.12.2 Features 173
6.12.3 PWM Timer Start Procedure 173
6.12.4 PWM Architecture 174
6.12.5 Basic Timer Operation 176

6.12.6 PWM Double Buffering and Automatic Reload 176
6.12.7 Modulate Duty Ratio 177
6.12.8 Dead-Zone Generator 178
6.12.9 PWM Timer Start Procedure 179
6.12.10 PWM Timer Stop Procedure 179
6.12.11 PWM Timer Register Mapping 181
6.13 Register Description 182
6.14 Real Time Clock (RTC) 199
6.14.1 Overview 199
6.14.2 RTC Features 199
6.14.3 RTC Function Description 200
6.14.4 RTC Register Mapping 202
6.14.5 RTC Register Descriptions 203
6.15 Serial Peripheral Interface Controller (SPI Master/Slave) 216
6.15.1 SPI Function Description and Features 216
6.15.2 SPIMS Timing Diagram 217
6.15.3 SPIMS Programming Example 219
6.15.4 SPIMS Serial Interface Control Register Map 220
6.15.5 SPIMS Control Register Description 221
6.16 TIMER Controller 228
6.16.1 General Timer Controller 228
6.16.2 Watchdog Timer 228
6.16.3 Timer Control Registers Map 230
6.17 UART Interface Controller 239
6.17.1 Overview 239
6.17.2 Features: 239
6.17.3 Block Diagram 239
6.17.4 Functional Blocks Descriptions 240
6.17.5 Finite State Machine 242
6.17.6 UART Interface Control Registers Mapping 245
6.18 Analog to Digital Converter 263
6.18.1 Features 263
6.18.2 ADC Functional Description 263
6.18.3 ADC Control Register Mapping 264
6.18.4 ADC Control Register Description 266
7 Electrical Characteristics 277
7.1 Absolute Maximum Ratings 277
7.2 DC Specifications 277

7.3 AC Specifications 277
7.3.1 Audio DAC Characteristic 277
7.3.2 ADC Characteristic 278
7.3.3 Voice Recorder Characteristic 278
8 Package Specifications 279

1 General Description

The NUC501 is an ARM7TDMI-based MCU, specifically designed to offer low-cost and high performance for various applications, like interactive toys, edutainment robots, and home appliances. It integrates the 32-bit RISC CPU with 32KB high-speed SRAM, crypto engine with OTP key, boot ROM, LDO regulator, ADC, DAC, I2C, SPI, USB2.0 FS Device, & GPIO into a cost-affordable while feature-rich micro-controller.

Owing to the simplicity of the NUC501 architecture that boots SpiMemory into the high-speed SRAM for program execution, the total system BOM is reduced to its minimum. Unlike usual ARM-based MCU products, the NUC501 operates without the use of SDRAM, which is usually the source of complexity, higher power consumption, and cost.

The ARM7TDMI runs up to 81MHz on the high-speed SRAM to offer enough horsepower for many MIPS-hungry tasks, while the remaining MIPS is still able to serve the need of application program. For those applications, like cartridge games, that require large code storage and variation of game play scenarios, the patented Extensible XIP Addressing on SpiMemory gives the flexibility whenever program execution speed is not a critical concern.

To protect the code against illegal pirating, the NUC501 provides a crypto engine that works with internal OTP2 key to encrypt the data stored at external SpiMemory when the design-in is finished. Without the knowledge of the OTP key, others can't decrypt the data even by means of ICE debugging.

The NUC501 is designed with special care to minimize the power consumption while allowing for the flexibility to reach for high performance. It includes the clock gating, variable frequency control for individual IP's, and bus control to reduce signal toggle. Besides, the NUC501 can be further operated under different power-saving modes: idle, power down with RTC active, and power down mode.

With so many practical peripherals integrated around the high-performance ARM7 CPU, the NUC501 is suitable for such applications as Interactive toys, edutainment robots, and home appliances. Whenever MIPS-hungry task meets cost-effective demand, you'll find the NUC501 truly useful to satisfy the requirement.

2 Feature

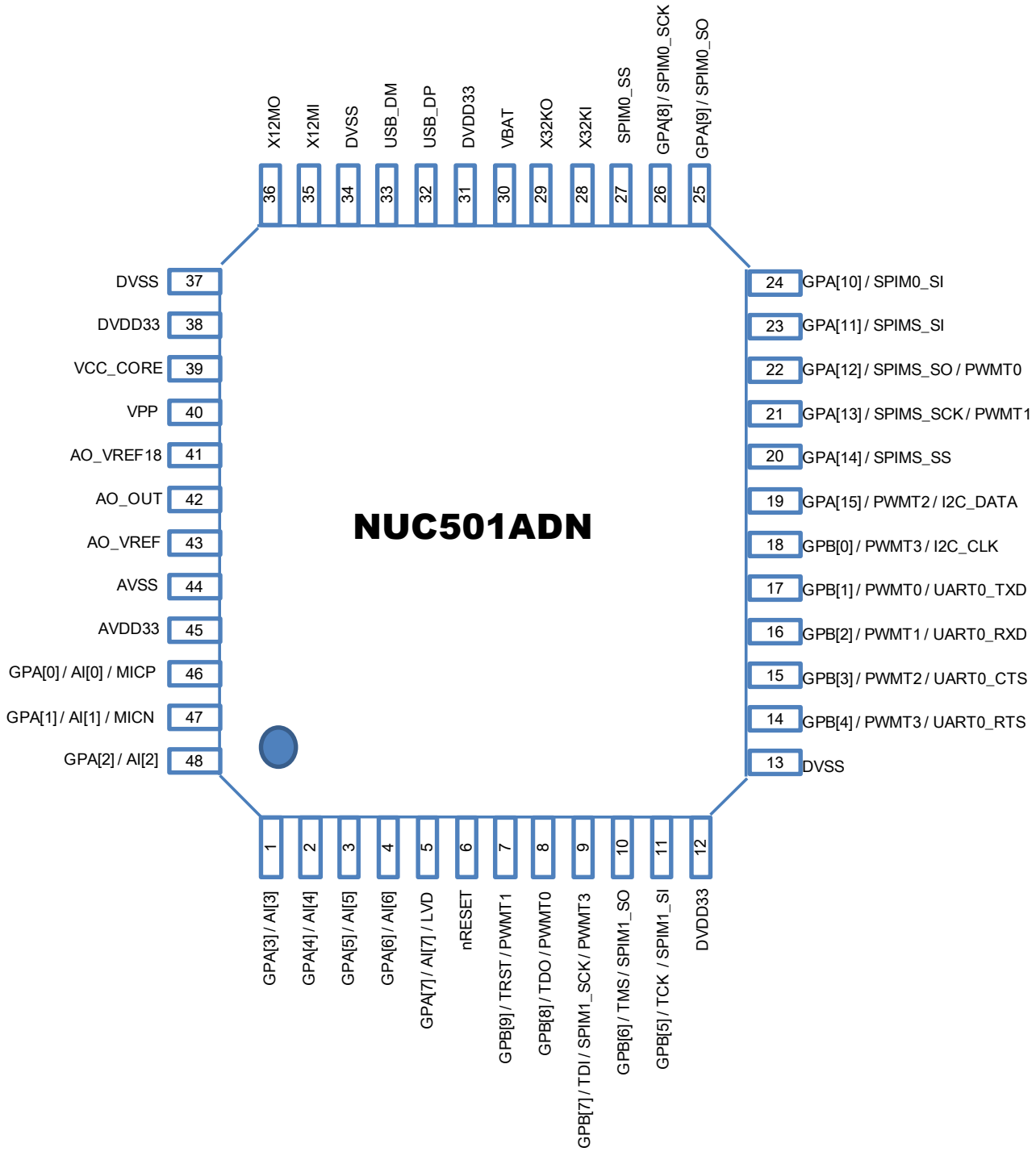
- **32-bit RISC CPU**
  - ARM7TDMI @ 81 MHz
  - 16-bit Thumb mode supported to save code size
  - Embedded *32* KB Local Memory divided into 16 segments for easier S/W programming
  - Boot from SpiMemory or USB
  - Program download into SRAM through JTAG before OTP key programmed
  - Integrate JTAG port to support real time, non-stop ICE function for system development and debugging

- **6KB internal ROM**
  - Boot loader
  - ICP for SpiFlash & security OTP key via USB
- **32KB internal SRAM**
  - Embedded 32KB SRAM for code and data
  - 16 segments with address tags
- **SpiMemory interface with code protection**
  - DMA mode for code booting from SpiMemory to internal SRAM
  - Direct CPU read access from SpiMemory
  - 128-bit OTP key for code protection against illegal pirating
  - 2-bit SPI mode supported for doubling data transfer rate
  - Shared (when not in use) with other SPI device for high-speed transfer via DMA
- **Audio Process Unit**
  - Mono 16-bit Sigma-Delta DAC output
  - Equalization function supported
- **USB 2.0 Full speed device**
  - 6 programmable endpoints for Control, Bulk In/Out, Interrupt and Isochronous transfers
  - 512-byte buffer
  - Auto suspend function
  - Remote wakeup capability
- **I^2C**
  - Compatible with Philips I^2C standard
  - Master mode
- **SPI**
  - Programmable master/slave mode
  - Speed up to 40MHz
- **4 Channel PWM**
  - Four 16-bit timers
  - Programmable duty control of output waveform (PWM)
  - Auto reload mode or one-shot pulse mode
  - Capture and compare function
- **Analog to Digital Converter**
  - 10-bit x 8-ch ADC for sensor, MIC, LVD, LVR
  - Maximum conversion rate: 400K samples per second
  - Power supply voltage: 3.3V
  - Analog input voltage range: 0 ~ 3.3 volts
  - Support wait-for-trigger mode & standby mode
- **Dedicated LVD/LVR**

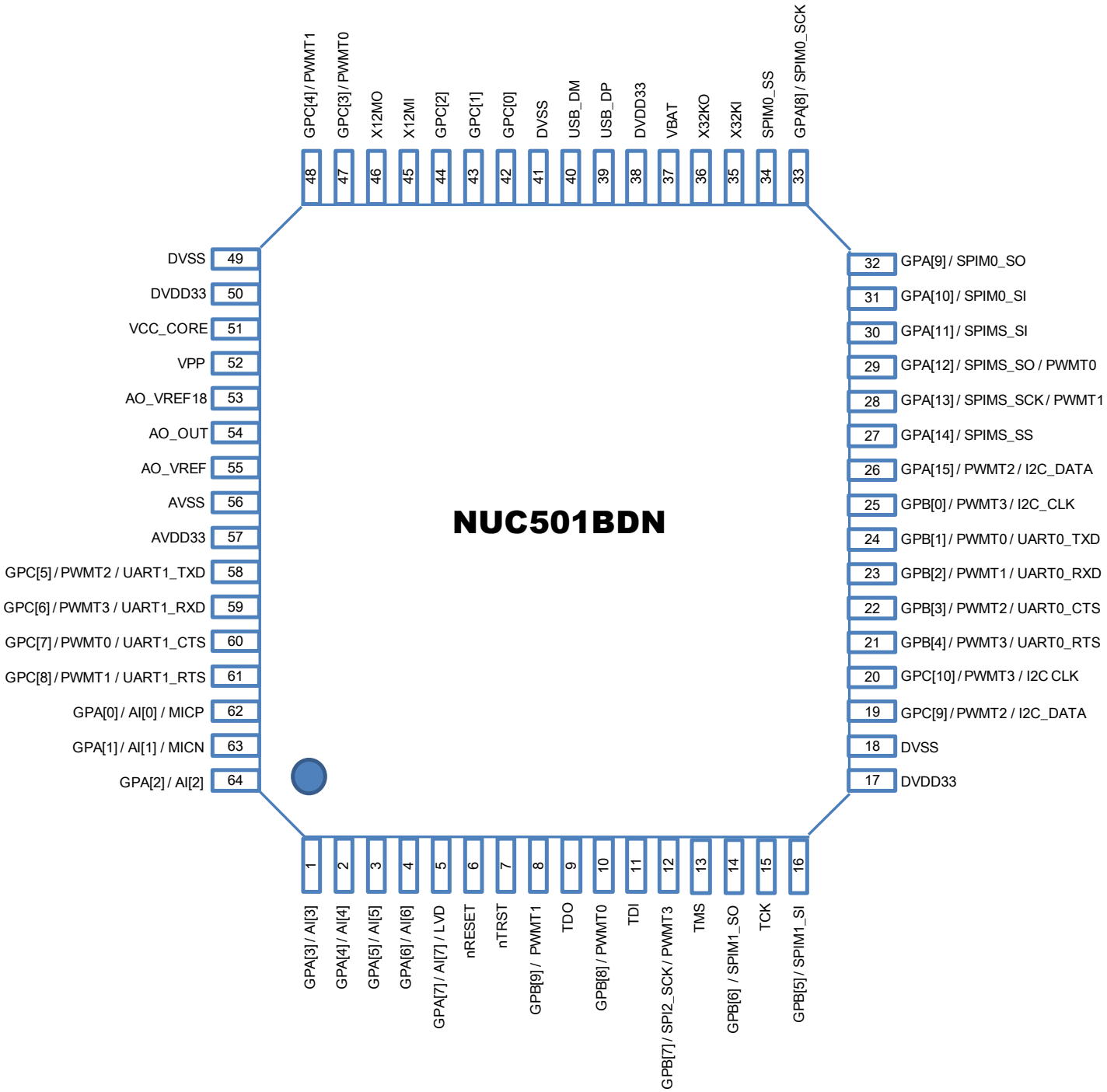
- 8-level voltage detection

- **Miscellaneous**
  - Two programmable 32-bit timers with 8-bit pre-scale
  - One 32-bit watch dog timer
  - 32.768KHz RTC function support
  - Up to 26/37 GPIO pins for LQFP-48 / LQFP-64
  - Two UART ports with flow control (TX, RX, CTS and RTS) and UART0 is for high speed
  - Power management modes: normal, idle, power down with RTC, and power down
  - 3.3V to 1.8V 200mA LDO regulator
- **Software Support**
  - GNU-based, open-source IDE: compiler, linker and debugger
- **Technology & Package**
  - 0.18um CMOS
  - 3.3-volt single supply
  - LQFP-48 (NUC501ADN)/ LQFP-64(NUC501BDN)

3 Pad and Pin Configuration



LQFP-48 Pin Out



LQFP-64 Pin Out

Pin Descriptions

In order to maximize the NUC501 application for different field, each pin of NUC501 is very flexible and can play up to four different functions. The user can program each pin to the wanted function for the different product.

The pin functions are controlled by the registers PAD_REG0, PAG_REG1 and PAD_REG2. For each multiple function pin, the default function is the GPIO. When the user programs the PAD_REG, the pins play the alternative function. If the different alternative functions are enabled simultaneous, the priority is

Alternative Function 1 > Alternative Function 2 > Alternative Function 3 > Default Function

For example:

If the GPA[12] is configured to be SPIMS_SO by PAD_REG1 and it is also configured to be PWMT0 by PAD_REG0, the actual function of GPA[12] would be SPIMS_SO because the SPIMS_SO function priority is higher than PWMT0.

Except the multiple functions, each NUC501 output driving current strength is also controllable. The driving strength control register is the GPA_DS, GPB_DS and GPC_DS. For different pin the driving can be 4mA or 8mA and 12mA or 16mA. For example, user can control the GPB[1] strength to 16mA and directed drive the high current LED to save PCB extra component to reduce the BOM cost.

Default Function Name	Alternative Function 1	Alternative Function 2	Alternative Function 3	Power on setting
GPIO				
GPA[0]	AI[0]	MIC+		
GPA[1]	AI[1]	MIC-		
GPA[2]	AI[2]			
GPA[3]	AI[3]			
GPA[4]	AI[4]			
GPA[5]	AI[5]			
GPA[6]	AI[6]			
GPA[7]	AI[7]	LVD		
	SPIM0_SS(Master)			
GPA[8]	SPIM0_SCK			Power on set (IBR)
GPA[9]	SPIM0_SO			Power on set (IBR)
GPA[10]	SPIM0_SI			

GPA[11]	SPIMS_SI			
GPA[12]	SPIMS_SO	PWMT0		Power on set (IBR)
GPA[13]	SPIMS_SCK	PWMT1		Power on set (48/64)
GPA[14]	SPIMS_SS(Slave)	USB_DET		
GPA[15]	PWMT2	USB_DET	I2C_DATA	
GPB[0]	PWMT3	USB_DET	I2C_CLK	
GPB[1]	PWMT0	USB_DET	UART0_TXD	Power on set (ICE)
GPB[2]	PWMT1	USB_DET	UART0_RXD	
GPB[3]	PWMT2	USB_DET	UART0_CTS	
GPB[4]	PWMT3	USB_DET	UART0_RTS	Power on set (SPI_S0)
GPB[5]	TCK	SPIM1_SI		
GPB[6]	TMS	SPIM1_SO	PWMT2	
GPB[7]	TDI	SPIM1_SCK	PWMT3	
GPB[8]	TDO	USB_DET	PWMT0	Power on set (SPI_S1)
GPB[9]	nTRST	USB_DET	PWMT1	
Audio DAC				
AO_OUT0				
AO_REF18				
AO_VREF				
USB2.0 Device				
USB_DP				
USB_DM				
MISC				
nRESET				
X12M				
EX12M				
X32K				
EX32K				
POWER				

VPP (6.5V)				
VBAT				
USBVDD33				
DVDD33				
DVDD33				
AVDD33				
DVSS				
DVSS				
DVSS				
AVSS				
VCC_CORE (OUTPUT)				
Pin Function for LQFP 64				
TCK				
TMS				
TDI				
TDO				
nTRST				
GPC[0]	SPIM1_SO	USB_DET		
GPC[1]	SPIM1_SI	USB_DET		
GPC[2]	SPIM1_SCK	USB_DET		
GPC[3]	PWMT0	USB_DET		
GPC[4]	PWMT1	USB_DET		
GPC[5]	PWMT2	UART1_TXD		
GPC[6]	PWMT3	UART1_RXD		
GPC[7]	PWMT0	UART1_CTS		
GPC[8]	PWMT1	UART1_RTS		
GPC[9]	PWMT2	I2C_DATA		
GPC[10]	PWMT3	I2C_CLK		

Table4.1 Pin function

Symbol	COB	LQFP64	LQFP48	TYPE	Description
GPA[0] / AI[0] / MICP	79	62	46	4/8mA I/O with Analog input	GPA[0] – General purpose input/output digital pin AI[0] – ADC analog input 0 MICP – MIC+
GPA[1] / AI[1] / MICN	80	63	47	4/8mA I/O with Analog input	GPA[1] – General purpose input/output digital pin AI[1] – ADC analog input 1 MICN – MIC-
GPA[2] / AI[2]	81	64	48	4/8mA I/O with Analog input	GPA[2] – General purpose input/output digital pin AI[2] – ADC analog input 2
GPA[3] / AI[3]	2	1	1	4/8mA I/O with Analog input	GPA[3] – General purpose input/output digital pin AI[3] – ADC analog input 3
GPA[4]/ AI[4]	3	2	2	4/8mA I/O with Analog input	GPA[4] – General purpose input/output digital pin AI[4] – ADC analog input 4
GPA[5] / AI[5]	4	3	3	4/8mA I/O with Analog input	GPA[5]– General purpose input/output digital pin AI[5] – ADC analog input 5
GPA[6] / AI[6]	5	4	4	4/8mA I/O with Analog input	GPA[6]– General purpose input/output digital pin AI[6] – ADC analog input 6
GPA[7] / AI[7] / LVD	6	5	5	4/8mA I/O with Analog input	GPA[7] – General purpose input/output digital pin AI[7] – ADC analog input 7 LVD – Low voltage detection
GPA[8] / SPIM0_SCK	42	33	26	4/8mA I/O	GPA[8] – General purpose input/output digital pin SPIM0_SCK - Serial clock output pin for SPIM0
GPA[9] / SPIM0_SO	41	32	25	4/8mA I/O	GPA[9] – General purpose input/output digital pin SPIM0_SO - Serial data input/output pin for SPIM0. Normal SPI mode, this pin is used as data out. Fast SPI read mode, this pin is the 2nd bit for data in.
GPA[10] / SPIM0_SI	40	31	24	4/8mA I/O	GPA[10] – General purpose input/output digital pin SPIM0_SI - Serial data input pin for SPIM0.
GPA[11] / SPIMS_SI	39	30	23	4/8mA I/O	GPA[11] – General purpose input/output digital pin SPIMS_SI - Serial data input pin for SPIMS.
GPA[12] / SPIMS_SO /	38	29	22	4/8mA I/O	GPA[12] – General purpose input/output digital pin

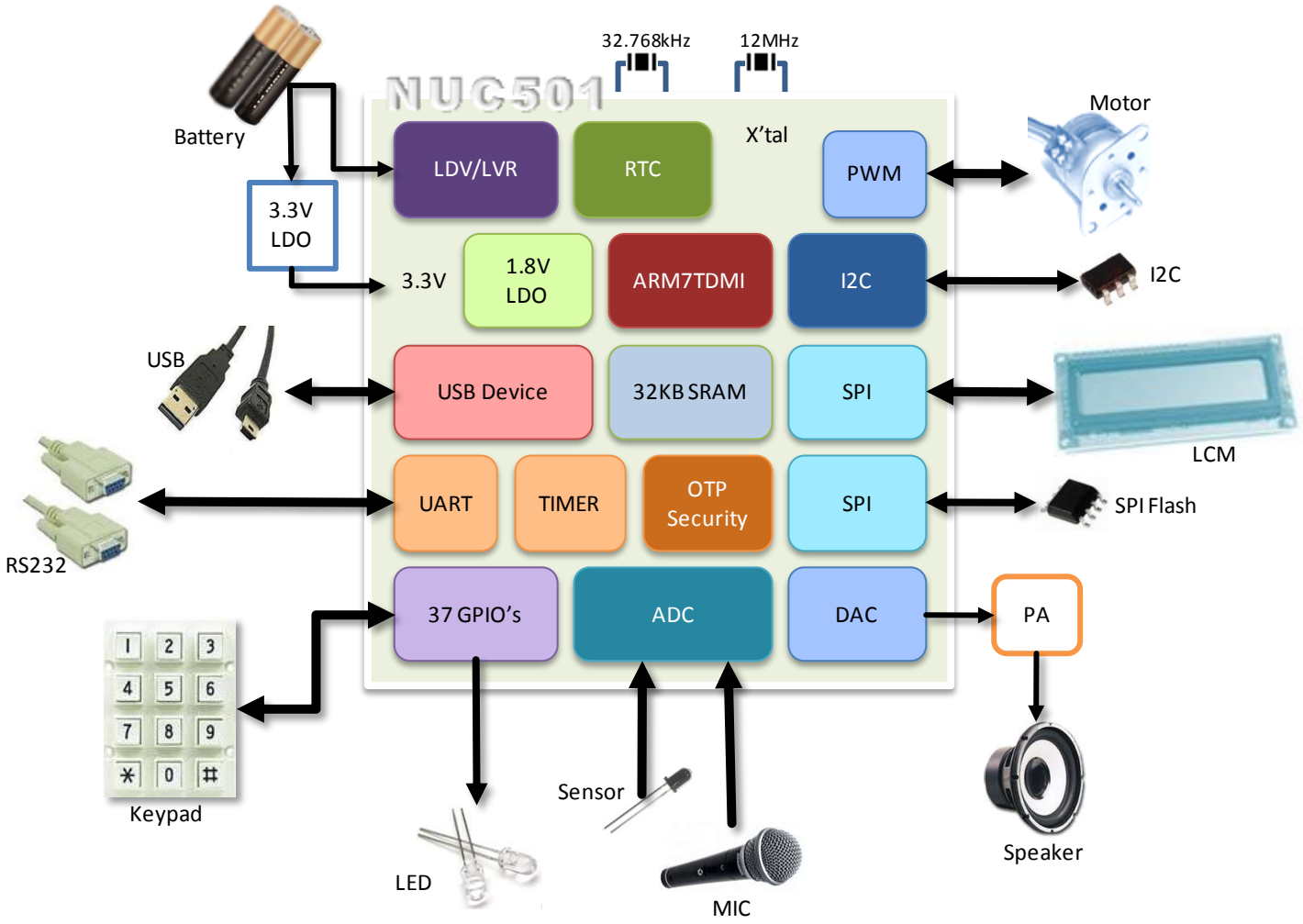
PWMT0					SPIMS_SO - Serial data output pin for SPIMS. PWMT0 – PWM output for timer 0
GPA[13] / SPIMS_SCK / PWMT1	37	28	21	4/8mA I/O	GPA[13] – General purpose input/output digital pin SPIMS_SCK - Serial clock pin for SPIMS (master/slave). PWMT1 – PWM output for timer 1
GPA[14] / SPIMS_SS / USB_DET	36	27	20	4/8mA I/O	GPA[14] – General purpose input/output digital pin SPIMS_SS - Serial chip select pin for SPIMS slave mode. USB_DET – USB detected pin
GPA[15] / PWMT2 / USB_DET / IC2_DATA	35	26	19	4/8mA I/O	GPA[15] – General purpose input/output digital pin PWMT2 – PWM output for timer 2 USB_DET – USB detected pin I2C_DATA – I2C data input/output pin, if this pin is select for I2C function
GPB[0] / PWMT3 / USB_DET / I2C_CLK	34	25	18	4/8mA I/O	GPB[0] – General purpose input/output digital pin PWMT3 – PWM output for timer 3 USB_DET– USB detected input I2C_CLK_ –I2C data output pin, if this pin is select for I2C function
GPB[1] / PWMT0 / USB_DET / UART0_TXD	31	24	17	12/16mA I/O	GPB[1] – General purpose input/output digital pin PWMT0– PWM output for timer 0 USB_DET– USB detected input UART0_TXD – Data transmitter output pin for UART0 (High speed)
GPB[2] / PWMT1 / USB_DET / UART0_RXD	30	23	16	12/16mA I/O	GPB[2] – General purpose input/output digital pin PWMT1 – PWM output for timer 1 USB_DET– USB detected input UART0_RXD – Data receiver input pin for UART0 (High speed)
GPB[3] / PWMT2 / USB_DET / UART0_CTS	29	22	15	12/16mA I/O	GPB[3] – General purpose input/output digital pin PWMT2 – PWM output for timer 2 c USB_DET– USB detected input UART0_CTS – Clear to Send input pin for UART0 (High speed)
GPB[4] / PWMT3 / USB_DET / UART0_RTS	28	21	14	12/16mA I/O	GPB[4] – General purpose input/output digital pin PWMT3– PWM output for timer 3 USB_DET– USB detected input UART0_RTS –Request to Send output pin for UART0 (High speed)
GPB[5] / TCK / SPIM1_SI	20	16	11	12/16mA I/O	GPB[5] – General purpose input/output digital pin TCK - JTAG ICE Test Clock pin (LQFP48 only)

					SPI2_SI_A –Serial data input pin for SPIM1 (master)
GPB[6] / TMS / SPIM1_SO/ PWMT2	18	14	10	12/16mA I/O	GPB[6] – General purpose input/output digital pin TMS - JTAG ICE Test Mode Select pin (LQFP48 only) SPI2_SO –Serial data output pin for SPIM1 (master) PWMT2 – PWM output for timer 2c
GPB[7] / TDI / SPIM1_SCK / PWMT3	15	12	9	12/16mA I/O	GPB[7] – General purpose input/output digital pin TDI – JTAG ICE TDO pin (LQFP48 only) SPIM1_SCK –Serial clock output pin for SPIM1 (master) PWMT3 – PWM output for timer 3
GPB[8] / TDO / USB_DET / PWMT0	13	10	8	12/16mA I/O	GPB[8] – General purpose input/output digital pin TDO – JTAG ICE TDO interface (LQFP48 only) USB_DET– USB detected input PWMT0 – PWM output for timer 0
GPB[9] / nTRST / USB_DET / PWMT1	10	8	7	4/8mA I/O	GPB[9] – General purpose input/output digital pin nTRST – JTAG ICE reset pin (LQFP48 only) USB_DET– USB detected input PWMT1 – PWM output for timer 1
GPC[0] / SPIM1_S0 / USB_DET	54	42		4/8mA I/O	GPC[0] – General purpose input/output digital pin SPIM1_SO –Serial data output pin for SPIM1 (master) USB_DET– USB detected input
GPC[1] / SPIM1_SI / USB_DET	55	43		4/8mA I/O	GPC[1] – General purpose input/output digital pin SPIM1_SI –Serial data input pin for SPIM1 (master) USB_DET– USB detected input
GPC[2] / SPIM1_SCK / USB_DET	56	44		4/8mA I/O	GPC[2] – General purpose input/output digital pin SPIM1_SCK –Serial clock output pin for SPIM1 (master) USB_DET– USB detected input
GPC[3] / PWMT0 / USB_DET	59	47		4/8mA I/O	GPC[3] – General purpose input/output digital pin PWMT0 – PWM output for timer 0 USB_DET– USB detected input
GPC[4] / PWMT1 / USB_DET	60	48		4/8mA I/O	GPC[4] – General purpose input/output digital pin PWMT1 – PWM output for timer 1 USB_DET– USB detected input
GPC[5] / PWMT2 /	75	58		4/8mA I/O	GPC[5] – General purpose input/output digital pin

UART1_TXD					PWMT2 – PWM output for timer 2 UART1_TXD – Data transmitter output pin for UART1
GPC[6] / PWMT3 / UART1_RXD	76	59		4/8mA I/O	GPC[6] – General purpose input/output digital pin PWMT3 – PWM output for timer 3 UART1_RXD – Data Receiver input pin for UART1
GPC[7] / PWMT0 / UART1_CTS	77	60		4/8mA I/O	GPC[7] – General purpose input/output digital pin PWMT0 – PWM output for timer 0 UART1_CTS – Clear to Send input pin for UART1
GPC[8] / PWMT1 / USRT1_RTS	78	61		4/8mA I/O	GPC[8] – General purpose input/output digital pin PWMT1 – PWM output for timer 1 UART1_RTS – Request to Send output pin for UART1
GPC[9] / PWMT2 / I2C_DATA	25	19		4/8mA I/O	GPC[9] – General purpose input/output digital pin PWMT2 – PWM output for timer 2 I2C_DATA – I2C data input/output pin, if this pin is select for I2C function
GPC[10] / PWMT3 / I2C_CLK	26	20		4/8mA I/O	GPC[10] – General purpose input/output digital pin PWMT3 – PWM output for timer 3 I2C_CLK_ –I2C data output pin, if this pin is select for I2C function
SPIM0_SS	44	34	27	8mA O	Chip Select pin for SPIM0, the SPIM0 is used for SPI memory
USB_DP	50	39	32	I/O	USB device signal D+
USB_DM	51	40	33	I/O	USB device signal D-
XTALI	57	45	35	I	Crystal input pin
XTALO	58	46	36	O	Crystal output pin
X32KI	45	35	28	I	RTC 32.768KHz crystal input pin
X32KO	46	36	29	O	RTC 32.768KHz crystal output pin
nRESET	8	6	6	I	External reset input – Low active, set this pin low reset the NUC501 to the chip initial state
TCK	19	15		I	JTAG ICE Test Clock pin
TMS	17	13		I	JTAG ICE Test Mode Select pin
nTRST	9	7		I	JTAG ICE Reset pin
TDO	12	9		8mA O	JTAG ICE TDO interface
TDI	14	11		I	JTAG ICE TDI interface
AO_OUT	70	54	42	AO	Audio DAC output pin
AO_VREF18	68,69	53	41	AI	1.8V power for analog circuit
AO_VREF	71	55	43	AO	Analog circuit voltage reference pin

DVDD33	7, 16, 21,22,32, 48, 49, 63, 64	17, 38, 50	12, 31, 38	I	3.3V power supply for I/O ports and LDO source for internal PLL and digital circuit
AVDD33	74	57	45		3.3V power supply for internal analog circuit
VBAT	47	37	30		1.8V Power supply for internal RTC circuit
VCC_CORE	65, 66	51	39		LDO 1.8V output pin
VPP	67	52	40		OTP 6.5V VPP pin. For OTP write, this pin supply is 6.5V for read, this pin supply is 1.8V
DVSS	1, 11, 23, 24, 27, 33, 44, 52, 53, 61, 62	18, 41, 49	13, 34, 37		Ground Pin for digital circuit
AVSS	73	56	44		Ground Pin for analog circuit

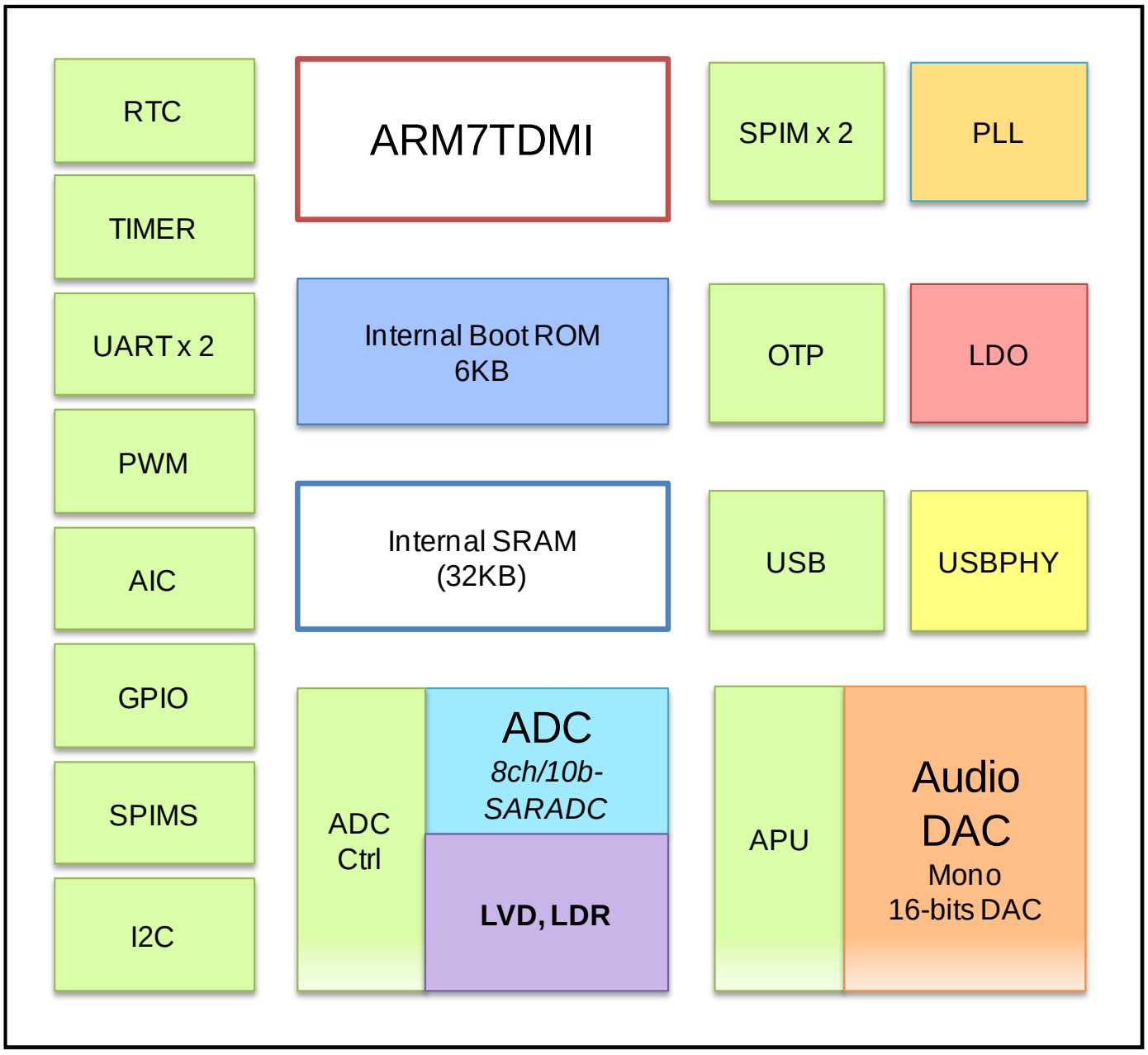
4 System Diagram



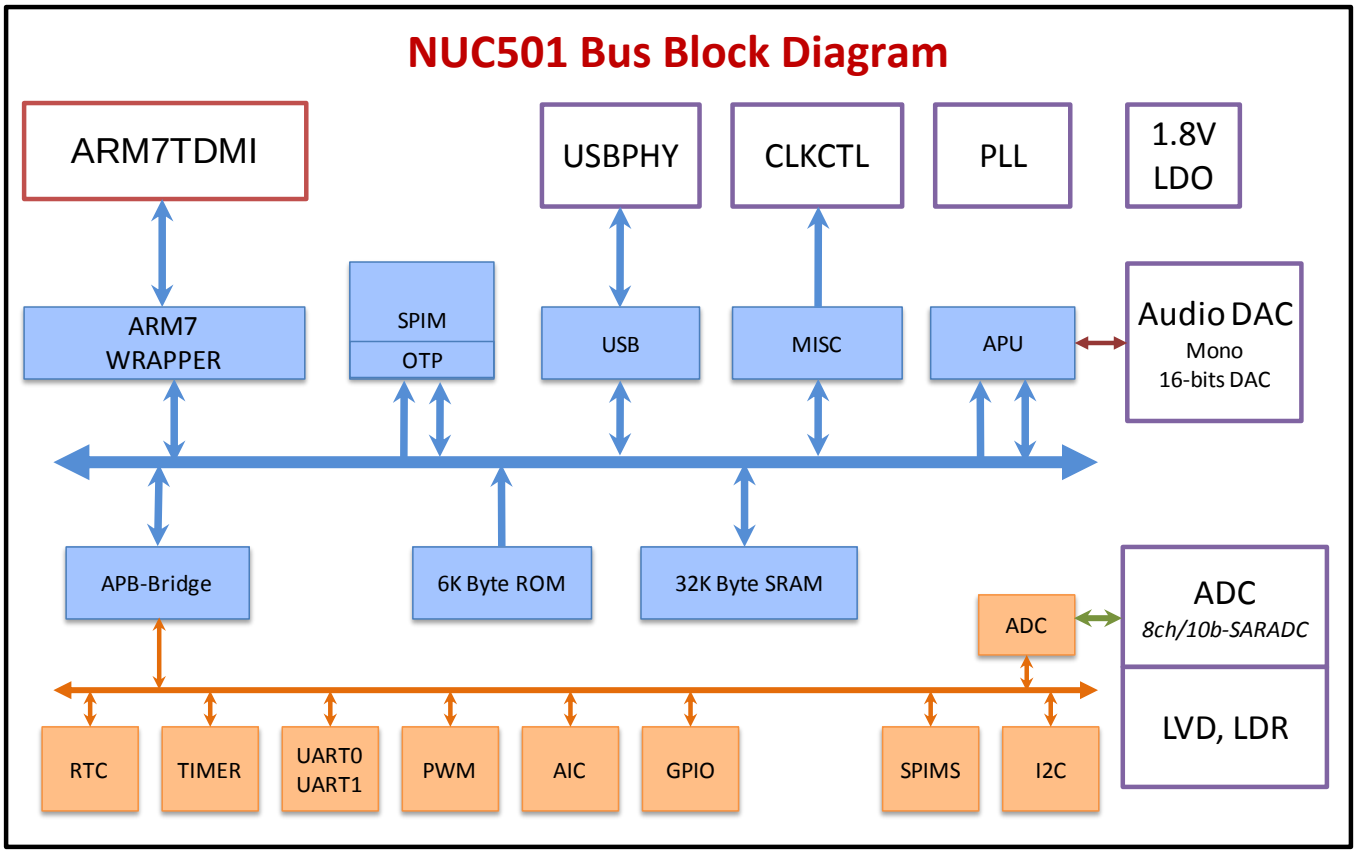
System Block Diagram

5 Block Diagram

5.1 System block diagram



5.2 On-Chip Bus block diagram



6 Functional Description

6.1 ARM7TDMI CPU Core

The ARM7TDMI CPU core, a member of the Advanced RISC Machines (ARM) family of general-purpose 32-bit microprocessors, offers high performance with very low power consumption. The architecture is based on Reduced Instruction Set Computer (RISC) principles, and the instruction set and related decode mechanism are much simpler than those of micro-programmed Complex Instruction Set Computers. Pipelining is employed so that all parts of the processing and memory systems can operate continuously. The high instruction throughput and impressive real-time interrupt response are the major benefits.

The ARM7TDMI CPU core has two instruction sets:

(1) The standard 32-bit ARM code

(2) 16-bit THUMB code

The THUMB code is 16-bit instruction set that allows it to increase the code density compare to standard ARM core while retaining most of the ARM performance advantage over a traditional 16-bit processor using 16-bit registers. THUMB instructions operate with the standard ARM register configuration, allowing excellent interoperability between ARM and THUMB states. Each 16-bit THUMB instruction has a corresponding 32-bit ARM instruction with the same effect on the processor model.

ARM7TDMI CPU core has 31 x 32-bit registers. At any one time, 16 registers are visible; the other registers are used to speed up exception processing. All the register specifies in ARM instructions can address any of the 16 registers. The CPU also supports 5 types of exception, such as two levels of interrupt, memory aborts, attempted execution of an undefined instruction and software interrupts.

6.2 System Manager

6.2.1 Overview

The following functions are included in system manager section

- System memory map
- Bus arbitration algorithm
- Power-on setting
- Product identify register
- System control registers for reset/share pin/GPIO
- Clock control registers

6.2.2 System Memory Mapping

NUC501 provides a 4G-byte address space for programmers. The memory locations assigned to each on-chip modules are shown in table 6.2-1. The detailed register and memory addressing and programming will be described in the following sections for individual on-chip modules. NUC501 only supports little-endian data format.

Address Space	Token	Modules
Memory Space		
0x0000_0000 – 0x0000_7FFF	IBR_BA	Internal Boot ROM (IBR) Memory Space (IBR_remap = 0)
0x0000_0000 – 0x1FFF_FFFF	SRAM_BA	SRAM Memory Space (IBR_remap = 1)
0x2000_0000 – 0x3FFF_FFFF	SRAM_BA	SRAM Memory Space (IBR_remap = 0)
0x4000_0000 – 0x4FFF_FFFF		SPI Flash/ROM Memory Space (SPIM0)
0x6000_0000 – 0x6000_7FFF	IBR_BA	Internal Boot ROM (IBR) Memory Space (IBR_remap = 1)
AHB Modules Space		
0xB100_0000 – 0xB100_01FF	GCR_BA	Global Control Registers
0xB100_0200 – 0xB100_02FF	CLK_BA	Clock Control Registers
0xB100_4000 – 0xB100_4FFF	SRAMCTL_BA	SRAM Control Registers
0xB100_7000 – 0xB100_7FFF	SPIM_BA	SPIM Control Register
0xB100_8000 – 0xB100_8FFF	APU_BA	Audio Process Unit (APU) Controller Registers
0xB100_9000 – 0xB100_9FFF	USB_BA	USB Device Controller Registers
APB Modules Space		
0xB800_1000 – 0xB800_1FFF	ADC_BA	Analog-Digital-Converter (ADC) Controller Registers
0xB800_2000 – 0xB800_2FFF	AIC_BA	Interrupt Controller Registers
0xB800_3000 – 0xB800_3FFF	GPIO_BA	GPIO Controller Registers

0xB800_4000 – 0xB800_4FFF	I2C_BA	I2C Interface Control Registers
0xB800_7000 – 0xB800_7FFF	PWM_BA	PWM Controller Registers
0xB800_8000 – 0xB800_8FFF	RTC_BA	Real Time Clock (RTC) Control Register
0xB800_A000 – 0xB800_AFFF	SPIMS_BA	SPI master/slave function Controller Registers
0xB800_B000 – 0xB800_BFFF	TIMER_BA	Timer Control Registers
0xB800_C000 – 0xB800_CFFF	UART_BA	UART Control Registers

Table 6.2-1 Address Space Assignments for On-Chip Modules

6.2.3 AHB Bus Arbitration

The internal bus of NUC501 chip is an AHB-compliant Bus and supports to connect with the standard AHB master or slave. NUC501's AHB arbiter provides a choice of two arbitration algorithms for simultaneous requests. These two arbitration algorithms are the d-priority mode and the round-robin-priority (rotate) mode. The selection of modes and types is determined on the PRTMOD0 control register in the Arbitration Control Register.

AHB bus arbiter also provides a mechanism for the maximum burst length for each AHB bus transfer. The maximum burst length is 16, and when the current AHB data transfer count is equal to the maximum burst length, the access of current AHB bus owner will be broken.

6.2.4 Fixed Priority Mode

Fixed priority mode is selected if **PRTMODx** = 0. The order of priorities on the AHB mastership among the on-chip master modules, listed in Table 6.2-2. If two or more master modules request to access AHB bus at the same time, the higher priority request will get the permission to access AHB bus.

Table 6.2-2 AHB Bus Priority Order in Fixed Priority Mode

Priority Sequence	AHB Bus Priority
	PRTMOD[0] = 0
1 (Lowest)	ARM7TDMI
2	SPIM0
3 (Highest)	APU

The SPI flash controller normally has the lowest priority under the fixed priority mode. NUC501 provides a mechanism to raise the priority of CPU request to the highest. If the **IPEN** bit (bit-4 of *AHB Control Register*) is set to 1, the **IPACT** bit (bit-5 of *AHB Control Register*) will be automatically set to 1 while an unmasked external NFIQ or NIRQ occurs. Under this circumstance, the ARM core will become the highest priority to access AHB bus.

The programmer can recover the original priority order by directly writing "1" to clear the **IPACT** bit. For example, this can be done that at the end of an interrupt service routine. Note that **IPACT** only can be automatically set to 1 by an external interrupt when **IPEN** = 1. It will not take effect for a programmer to directly write 1 to **IPACT** to raise ARM core's AHB priority.

6.2.4.1 Round Robin Priority Mode

Round-robin priority mode is selected if **PRTMODx** = 1. The AHB bus arbiter uses a round robin arbitration scheme for every master module to gain the bus ownership in turn. That is the requestor

having the highest priority becomes the lowest-priority requestor after it has been granted access.

6.2.4.2 Rotate rule Example:

In the default sequence of AHB Master Bus, the priority is APU > SPIM0 > ARM.

6.2.5 Power-On Settings

The power-on setting is used to configure the chip to enter the specified state when the chip is power-up or reset. Application board needs to add the proper pull-down or pull-up resistor for the relative configuration pins.

Pin Name	Descriptions	Register Bit Mapping
GPB[8] **GPB[4]**	**SPI flash speed selection (SCLK)** 00 : 72 MHz 01 : 36 MHz 10 : 18 MHz 11 : 50 KHz	**SPOCR[6:5]**
GPB[1]	**ICE Mode configuration setting** "0" : ICE mode enable and the disable the cipher function "1" : Normal mode	**SPOCR[4]**
GPA[13]	**LQFP48 ICE mode configuration setting** "0" : 48-pins package and GPB[9:5] for ICE connection "1" : 48-pins package and GPB[9:5] use the normal function	**SPOCR[3]**
GPA[12] **GPA[9]** **GPA[8]**	• 3'b000 : test mode • 3'b001 : test mode • 3'b010 : test mode • 3'b011 : test mode • 3'b100 : Boot from SRAM • 3'b101 : Boot from USB • 3'b110 : OTP program mode • 3'b111 : Boot from SpiMemory	**SPOCR[2:0]**

6.2.6 System Manager Control Registers

Register	Address	R/W	Description	Default Value
GCR_BA = 0xB100_0000				
PDID	GCR_BA+0x00	R	Product Identification Register	0x0055_0501
SPOCR	GCR _BA+0x04	R/W	System Power-On Configuration Register	0x0000_00XX
CPUCR	GCR _BA+0x08	R/W	CPU Control Register	0x0000_0000
MISCR	GCR _BA+0x0C	R/W	Miscellaneous Control Register	0x0000_0000
IPRST	GCR _BA+0x14	R/W	IP Reset Control Resister	0x0000_0000
AHB_CTRL	GCR _BA+0x20	R/W	AHB Bus Control register	0x0000_0000
PAD_REG0	GCR _BA+0x30	R/W	PAD function	0x0000_0000
PAD_REG1	GCR _BA+0x34	R/W	PAD function	0x0000_0000
PAD_REG2	GCR _BA+0x38	R/W	PAD function	0x0000_0000
GPA_DS	GCR _BA+0x74	R/W	GPIOA pads driving strength control	0x0000_0000
GPB_DS	GCR _BA+0x78	R/W	GPIOB pads driving strength control	0x0000_0000
GPC_DS	GCR _BA+0x7C	R/W	GPIOC pads driving strength control	0x0000_0000

Product Identifier Register (PDID)

This register provides specific read-only information for software to identify this chip.

Register	Address	R/W	Description	Default Value
PDID	GCR_BA+00	R	Product Identifier Register	0x0x55_0501

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

Bits	Descriptions	
[31:24]	**Reserved**	**Reserved**
[27:24]	**CVI**	**Chip Version Identifier** Chip version identifier is "4'h0" for 1st version
[23:0]	**CID**	**Chip Identifier** Chip identifier is "24'h55_0501" for NUC501.

System Power On Configuration Register (SPOCR)

This register provides specific information for software to identify this chip's power-on setting. SPOCR[6:0] are the status of the power-on setting pins. They can be modified by software programming.

Register	Address	R/W	Description	Default Value
SPOCR	GCR_BA+04	R/W	System Power-On Configuration Register	0x0000_00XX

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

Bits	Descriptions	
[31:21]	**Reserved**	**Reserved**
[20]	**IBR_remap**	**IBR_remap** 0: Boot ROM address mapping at 0x0000_0000 – 0x0000_7FFF and the SRAM address mapping at 0x2000_0000 – 0x3FFF_FFFF 1: Boot ROM address mapping at 0x6000_0000 – 0x6000_7FFF and the SRAM address mapping at 0x0000_0000 – 0x1FFF_FFFF
[19:7]	**Reserved**	**Reserved**
[6:5]	**SYS_CFG**	**SPI flash speed selection (SCLK)** 00 : 72 MHz 01 : 36 MHz 10 : 18 MHz 11 : 50 KHz
[4]	**SYS_CFG**	**ICE Mode configuration setting (Read Only)** 0: ICE mode enable and the disable the cipher function 1: Normal mode

[3]		**LQFP48 ICE mode configuration setting** 0: 48-pins package and GPB[9:5] for ICE connection 1: 48-pins package and GPB[9:5] use the normal function
[2:0]		3'b000 : test mode 3'b001 : test mode 3'b010 : test mode 3'b011 : test mode 3'b100 : Boot from SRAM 3'b101 : Boot from USB 3'b110 : OTP program mode 3'b111 : Boot from SpiMemory

CPU Control Register (CPUCR)

Register	Address	R/W	Description	Reset Value
CPUCR	GCR_BA+08	R/W	CPU control register	0x0000_0000

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							
7	6	5	4	3	2	1	0
Reserved							**CPURST**

Bits	Descriptions	
[31:1]	**Reserved**	**Reserved**
[0]	**CPURST**	CPU one shut reset. Write this bit 1 will reset the CPU. This bit will auto clear after the CPU reset 0 : Normal 1 : Reset CPU

MISC Control Register (MISCR)

Register	Address	R/W	Description	Reset Value
MISCR	GCR_BA+0C	R/W	Miscellaneous Control Register	0x0000_0000

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							
7	6	5	4	3	2	1	0
Reserved						**LVR_WARM**	**LVD_EN**

Bits	Descriptions	
[31:2]	**Reserved**	**Reserved**
[1]	**LVR_WARM**	**Low Voltage Reset Warm Up** 0 = Disable 1 = Low Voltage Reset function is warmed up to operate
[0]	**LVD_EN**	**Low Voltage Detect Enable** 0 = Disable 1 = Low Voltage Reset is selected and enabled

Note:

1. Enable LVR_WARM first, waiting 5us and then enable LVD_EN.
2. Disable LVD_EN first, and then disable LVR_WARM.
3. System will be reset, when low voltage reset (LVR) function was enabled and AVDD drops below 2.4V
4. When LVR_WARM is 1, the LVR function block will consumes about several tens uA.

IP Reset Control Register (IPRST)

This register provides specific read-only information for software to identify this chip.

Register	Address	R/W	Description	Default Value
IPRST	GCR_BA+14	R/W	IP Reset Control Resister	0x0000_0000

31	30	29	28	27	26	25	24
Reserved	**SPIMS_RST**	**Reserved**	**ADC_RST**	**GPIO_RST**	**Reserved**	**SRAM_RST**	**Reserved**
23	22	21	20	19	18	17	16
Reserved							**APU_RST**
15	14	13	12	11	10	9	8
Reserved				**UDC_RST**	**SPIM_RST**	**I2C_RST**	**PWM_RST**
7	6	5	4	3	2	1	0
Reserved		**TMR_RST**	**Reserved**			**UR1_RST**	**UR0_RST**

Bits	Descriptions	
[31]	**Reserved**	**Reserved**
[30]	**SPIMS_RST**	**SPIMS Reset** "0": Normal operation "1": IP reset
[29]	**Reserved**	**Reserved**
[28]	**ADC_RST**	**ADC Reset** "0": Normal operation "1": IP reset
[27]	**GPIO_RST**	**GPIO Reset** "0": Normal operation "1": IP reset
[26]	**Reserved**	**Reserved**
[25]	**SRAM_RST**	**SRAM Controller Reset** "0": Normal operation "1": IP reset
[24:17]	**Reserved**	**Reserved**

[16]	**APU_RST**	**APU controller Reset** "0": Normal operation "1": IP reset
[15:12]	**Reserved**	**Reserved**
[11]	**UDC_RST**	**USB Device controller Reset** "0": Normal operation "1": IP reset
[10]	**SPIM_RST**	**SPIM0 and SPI1 controller Reset** "0": Normal operation "1": IP reset
[9]	**I2C_RST**	**I2C controller Reset** "0": Normal operation "1": IP reset
[8]	**PWM_RST**	**PWM controller Reset** "0": Normal operation "1": IP reset
[7:6]	**Reserved**	**Reserved**
[5]	**TMR_RST**	**Timer and Watch Dog controller Reset** "0": Normal operation "1": IP reset
[4:2]	**Reserved**	**Reserved**
[1]	**UR1_RST**	**UART1 controller Reset** "0": Normal operation "1": IP reset
[0]	**UR0_RST**	**UART0 controller Reset** "0": Normal operation "1": IP reset

AHB Control Register (AHB_CTRL)

Register	Address	R/W	Description	Default Value
AHB_CTRL	GCR_BA+0x20	R/W	AHB Control Register	0x0000_0000

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							
7	6	5	4	3	2	1	0
Reserved		**IPACT**	**IPEN**	**Reserved**			**PRTMOD0**

Bits	Descriptions	
[31:6]	**Reserved**	**Reserved**
[5]	**IPACT**	**Interrupt active status in IPEN enabled mode** This bit is set when the IPEN is enabled and the external FIQ or IRQ is active Write "1" to clear the status 0: Inactive 1: Active
[4]	**IPEN**	**Enable raising the Priority of CPU in IRQ or FIQ period** It can be used to reduce the interrupt latency in a real-time system, set this bit, the CPU will has the highest AHB priority 0: Disable 1: Enable
[3:1]	**Reserved**	**Reserved**
[0]	**PRTMOD0**	**AHB Bus Arbitration mode control** 0: fixed priority mode 1: round-robin priority mode (rotate) The priority mode for fixed priority mode is APU > SPIM0 > ARM7Tdmi

PAD Control Register (PAD_REG0)

There are four PWM timers within the NUC501 and each PWM can free output to several GPIO pin by user setting for different application.

Register	Address	R/W	Description	Reset Value
PAD_REG0	GCR_BA+30	R/W	PAD Control Register	0x0000_0000

31	30	29	28	27	26	25	24
PWM_TMR3_I			**PWM_TMR3_O**				
23	22	21	20	19	18	17	16
PWM_TMR2_I			**PWM_TMR2_O**				
15	14	13	12	11	10	9	8
PWM_TMR1_I			**PWM_TMR1_O**				
7	6	5	4	3	2	1	0
PWM_TMR0_I			**PWM_TMR0_O**				

Bits	Descriptions	
[31:29]	**PWM_TMR3_I**	PWM Timer 3 input pin selection 000 = PWM Timer 3 input from GPIOB[0] 001 = PWM Timer 3 input from GPIOB[4] 010 = PWM Timer 3 input from GPIOC[6] 011 = PWM Timer 3 input from GPIOC[10] 100 = PWM Timer 3 input from GPIOB[7] Others : disable PWM Timer 3 input function
[28:24]	**PWM_TMR3_O**	PWM Timer 3 output pin selection 1 = output enable 0 = output disable [24] = PWM Timer 3 output to GPIOB[0] [25] = PWM Timer 3 output to GPIOB[4] [26] = PWM Timer 3 output to GPIOC[6] [27] = PWM Timer 3 output to GPIOC[10] [28] = PWM Timer 3 output to GPIOB[7]

[23:21]	**PWM_TMR2_I**	PWM Timer 2 input pin selection 000 = PWM Timer 2 input from GPIOA[15] 001 = PWM Timer 2 input from GPIOB[3] 010 = PWM Timer 2 input from GPIOC[5] 011 = PWM Timer 2 input from GPIOC[9] 100 = PWM Timer 2 input from GPIOB[6] Others : disable PWM Timer 2 input function
[20:16]	**PWM_TMR2_O**	PWM Timer 2 output pin selection 1 = output enable 0 = output disable [16] = PWM Timer 2 output to GPIOA[15] [17] = PWM Timer 2 output to GPIOB[3] [18] = PWM Timer 2 output to GPIOC[5] [19] = PWM Timer 2 output to GPIOC[9] [20] = PWM Timer 2 output to GPIOB[6]
[15:13]	**PWM_TMR1_I**	PWM Timer 1 input pin selection 000 = PWM Timer 1 input from GPIOA[13] 001 = PWM Timer 1 input from GPIOB[2] 010 = PWM Timer 1 input from GPIOC[4] 011 = PWM Timer 1 input from GPIOC[8] 100 = PWM Timer 1 input from GPIOB[9] Others : disable PWM Timer 1 input function
[12:8]	**PWM_TMR1_O**	PWM Timer 1 output pin selection 1 = output enable 0 = output disable [8] = PWM Timer 1 output to GPIOA[13] [9] = PWM Timer 1 output to GPIOB[2] [10] = PWM Timer 1 output to GPIOC[4] [11] = PWM Timer 1 output to GPIOC[8] [12] = PWM Timer 1 output to GPIOB[9]
[7:5]	**PWM_TMR0_I**	PWM Timer 0 input pin selection 000 = PWM Timer 0 input from GPIOA[12] 001 = PWM Timer 0 input from GPIOB[1] 010 = PWM Timer 0 input from GPIOC[3] 011 = PWM Timer 0 input from GPIOC[7] 100 = PWM Timer 0 input from GPIOB[8] Others : disable PWM Timer 0 input function

[4:0]	**PWM_TMR0**	PWM Timer 0 output pin selection 1 = output enable 0 = output disable [0] = PWM Timer 0 output to GPIOA[12] [1] = PWM Timer 0 output to GPIOB[1] [2] = PWM Timer 0 output to GPIOC[3] [3] = PWM Timer 0 output to GPIOC[7] [4] = PWM Timer 0 output to GPIOB[8]

PAD Control Register (PAD_REG1)

Register	Address	R/W	Description	Reset Value
PAD_REG1	GCR_BA+34	R/W	PAD Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
ADCP_EN							
15	14	13	12	11	10	9	8
Reserved			**UART1_MEN**	**UART0_MEN**	**Reserved**	**UART1_EN**	**UART0_EN**
7	6	5	4	3	2	1	0
Reserved	**SPIM0_EN**	**SPIMS_EN**	**SPIM1_EN**		**I2CP_EN**		**ICE_EN**

Bits	Descriptions	
[31:24]	**Reserved**	**Reserved**
[23:16]	**ADCP_EN**	ADC pins enable [23:16] represents GPIOA[7:0] respectively 0 = disable 1 = enable
[15:13]	**Reserved**	**Reserved**
[12]	**UART1_MEN**	UART1 Modem pin enable 0 = disable 1 = GPIOC[8:7] used as the pin of UART1 CTSn and RTSn
[11]	**UART0_MEN**	UART0 Modem pin enable 0 = disable 1 = GPIOB[4:3] used as the pin of UART0 CTSn and RTSn
[10]	**Reserved**	**Reserved**
[9]	**UART1_EN**	UART1 TxD and RxD pin enable 0 = disable 1 = GPIOC[6:5] used as the pins of UART1 TxD and RxD

[8]	**UART0_EN**	UART0 TxD and RxD pin enable 0 = disable 1 = GPIOB[2:1] used as the pins of UART0 TxD and RxD
[7]	**Reserved**	**Reserved**
[6]	**SPIM0_EN**	SPIM0 pin enable GPIOA[10:8] used as pins of the SPIM0 (SPI_ROM) 0 = disable 1 = enable
[5]	**SPIMS_EN**	SPIMS pin enable (SPIMS pins at GPIOA[14:11]) GPA[14] used as the CS_ pin of SPIMS, and was controlled by SPIMS CNTRL[16] 0 = disable 1 = enable
[4:3]	**SPIM1_EN**	SPIM1 pin enable 2'b00 = disable 2'b01 = SPIM1 pins at GPIOB[7:5] 2'b10 = SPIM1 pins at GPIOC[2:0] 2'b11 = Unacceptable
[2:1]	**I2CP_EN**	I2C pin enable 2'b00 = disable 2'b01 = I2C pins at GPIOA[15] and GPIOB[0] 2'b10 = I2C pins at GPIOC[10:9] 2'b11 = Unacceptable
[0]	**ICE_EN**	For 48-pins package, change GPIOB[9:5] to JTAG interface 0 = GPIO 1 = JTAG Note: the bit can be set by power-on-setting

PAD Control Register (PAD_REG2)

Register	Address	R/W	Description	Reset Value
PAD_REG2	GCR_BA+38	R/W	PAD Control Register	0x0000_0000

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							
7	6	5	4	3	2	1	0
Reserved				**USBDET_SEL**			

Bits	Descriptions	
[31:4]	**Reserved**	**Reserved**
[3:0]	**USBDET_SEL**	USB detection selection 0000 : disable 0001 : USB connection detect pin from GPA[14] 0010 : USB connection detect pin from GPA[15] 0011 : USB connection detect pin from GPB[0] 0100 : USB connection detect pin from GPB[1] 0101 : USB connection detect pin from GPB[2] 0110 : USB connection detect pin from GPB[3] 0111 : USB connection detect pin from GPB[4] 1000 : USB connection detect pin from GPB[8] 1001 : USB connection detect pin from GPB[9] 1010 : USB connection detect pin from GPC[0] 1011 : USB connection detect pin from GPC[1] 1100 : USB connection detect pin from GPC[2] 1101 : USB connection detect pin from GPC[3] 1110 : USB connection detect pin from GPC[4] 1111 : disable

GPIOA driving strength (GPA_DS)

Register	Address	R/W	Description	Reset Value
GPA_DS	GCR_BA+74	R/W	GPIOA driving strength	0x0000_0000

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
GPA_DS[15:8]							
7	6	5	4	3	2	1	0
GPA_DS[7:0]							

Bits	Descriptions	
[31:16]	**Reserved**	**Reserved**
[15:0]	**GPA_DS**	GPIOA driving strength 0: 4mA driving strength IO 1: 8mA driving strength IO

GPIOB driving strength (GPB_DS)

Register	Address	R/W	Description	Reset Value
GPB_DS	GCR_BA+78	R/W	GPIOB driving strength	0x0000_0000

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						**GPB_DS[9:8]**	
7	6	5	4	3	2	1	0
GPB_DS[7:0]							

Bits	Descriptions	
[31:10]	**Reserved**	**Reserved**
[9:0]	**GPB_DS**	GPIOB driving strength [0] and [9] 0: 4mA driving strength IO 1: 8mA driving strength IO others 0: 12mA driving strength IO 1: 16mA driving strength IO

GPIOC driving strength (GPC_DS)

Register	Address	R/W	Description	Reset Value
GPC_DS	GCR_BA+7C	R/W	GPIOC driving strength	0x0000_0000

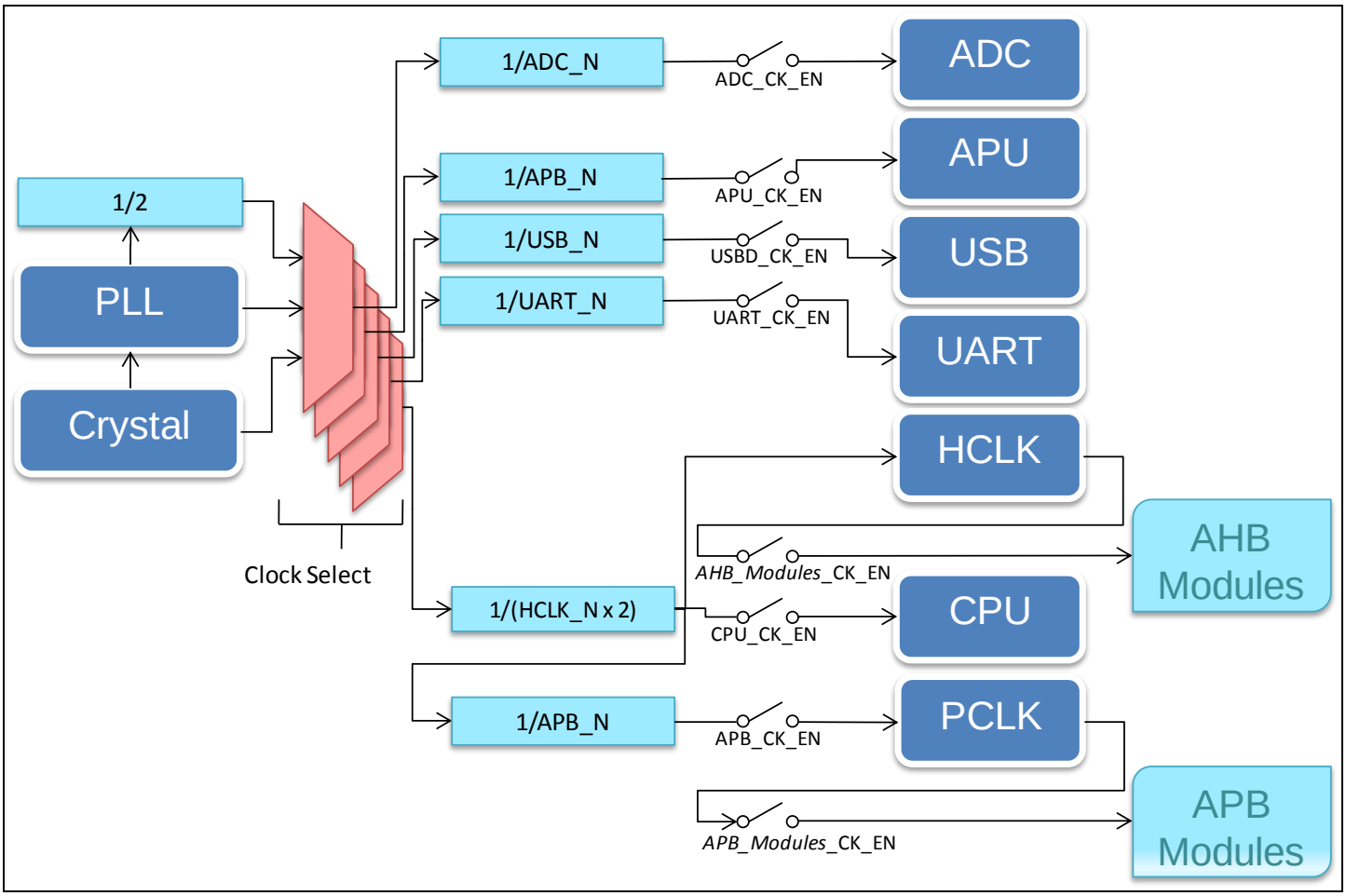
31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved					**GPC_DS[10:8]**		
7	6	5	4	3	2	1	0
GPC_DS[7:0]							

Bits	Descriptions	
[31:11]	**Reserved**	**Reserved**
[10:0]	**GPC_DS**	GPIOC driving strength 0: 4mA driving strength IO 1: 8mA driving strength IO

6.3 Clock Controller

6.3.1 Function Description

The clock controller generates the clocks for the whole chip, it include all AMBA interface modules and all peripheral clocks, the USB, UART, APU and so on. There is one PLL modules in this chip, and the PLL clock source is from the external crystal input.



The clock controller implements the power control function, include the individually clock on or off control register, clock source select and the divided number from clock source. These functions minimize the extra power consumption and the chip run on the just condition. On the power down mode the controller turn off the crystal oscillator to minimize the chip power consumption.

The clock HCLK is the source for all the AMBA modules. The HCLK is the operating clock for the SRAM and it is divided by two from one of the sources, Crystal, PLL, PLL/2 and the crystal 32 KHz, the HCLK is used for the AMBA AHB BUS clock. The ARM7 CPU uses the same frequency as the HCLK. The APB clock is divided from the HCLK too.

6.3.2 Clock Control Registers

Register	Address	R/W	Description	Reset Value
CLK_BA = 0xB100_0200				
PWRCON	CLK_BA + 00	R/W	System Power Down Control Register	0x00FF_FF03
AHBCLK	CLK_BA + 04	R/W	AHB Device Clock Enable Control Register	0x0000_0083
APBCLK	CLK_BA + 08	R/W	APB Device Clock Enable Control Register	0x0000_0007
CLKSEL	CLK_BA + 10	R/W	Clock Source Select Control Register	0x0000_0000
CLKDIV0	CLK_BA_+ 14	R/W	Clock Divider Number Register 0	0x0000_0000
CLKDIV1	CLK_BA_+ 18	R/W	Clock Divider Number Register 1	0x0000_0000
MPLLCON	CLK_BA + 20	R/W	MPLL Control Register	0x0001_4035

Power Down Control Register （PWRCON）

The chip clock source is from an external crystal. The crystal oscillator can be control on/off by the register XTAL_EN. When turn off the crystal, the chip into power down state.
Crystal wake up pre-scale counter value. After the clock counter count pre-scale × 256 crystal cycle, the clock controller output the clock to system.

Register	Address	R/W	Description	Reset Value
PWRCON	CLK_BA + 00	R/W	System Power Down Control Register	0x00FF_FF03

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Pre-Scale[15:8]							
15	14	13	12	11	10	9	8
Pre-Scale[7:0]							
7	6	5	4	3	2	1	0
Reserved				**INT_EN**	**INTSTS**	**XIN_CTL**	**XTAL_EN**

Bits	Descriptions	
[31:24]	**Reserved**	**Reserved**
[23:8]	**Pre-Scale**	Pre-Scale counter Assume the crystal is stable after the Pre-Scale * 256 crystal cycle
[7:4]	**Reserved**	**Reserved**
[3]	**INTSTS**	Power Down interrupt status Read 0 = normal 1 = Indicate crystal enable change from low to high, the chip is resume form power down state. The interrupt is active if the GPIO, USB or Host controller wakeup Write 0 = no action. 1 = Clear interrupt
[2]	**INT_EN**	Power On Interrupt Enable 0 = Disable 1 = Enable. The interrupt will occur when the Crystal enable signal (XTAL_EN) change from LOW to HIGH.

[1]	**XIN_CTL**	Crystal Pre-Divide Control for Wake-Up from Power Down Mode. The chip will delay 256 * Pre-scale cycles after the reset signal to wait the Crystal to stable. 1 = Enable the pre-scale counter 0 = Disable the pre-scale, assume the crystal is stable
[0]	**XTAL_EN**	Crystal Oscillator (Power Down) Control 1: Crystal oscillation enable (Normal operation) 0: Crystal oscillation disable (Power down)

AHB Devices Clock Enable Control Register （AHBCLK）

These register bits are used to enable/disable clock for AMBA clock, AHB engine and peripheral

Register	Address	R/W	Description	Reset Value
AHBCLK	CLK_BA + 04	R/W	AHB Devices Clock Enable Control Register	0x0000_0083

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							**APU_CK_EN**
7	6	5	4	3	2	1	0
SPIM_CK_EN	**USBD_CK_EN**	**Reserved**				**APB_CK_EN**	**CPU_CK_EN**

Bits	Descriptions	
[31:9]	**Reserved**	**Reserved**
[8]	**APU_CK_EN**	APU Clock Enable Control. 0 = Disable 1 = Enable
[7]	**SPIM_CK_EN**	SPI FLASH/ROM Controller Clock Enable Control (SPIM0 & SPIM1) 0 = Disable 1 = Enable
[6]	**USBD_CK_EN**	USB Device Clock Enable Control 0 = Disable 1 = Enable
[5:2]	**Reserved**	**Reserved**
[1]	**APB_CK_EN**	APB Clock Enable Control. 0 = Disable 1 = Enable
[0]	**CPU_CK_EN**	CPU Clock Enable Control 0 = Disable 1 = Enable

APB Devices Clock Enable Control Register （APBCLK）

These register bits are used to enable/disable clock for APB engine and peripheral.

Register	Address	R/W	Description	Reset Value
APBCLK	CLK_BA + 08	R/W	APB Devices Clock Enable Control Register	0x0000_0007

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						**ADC_CK_EN**	**SPIMS_CK_EN**
7	6	5	4	3	2	1	0
Reserved	**I2C_CK_EN**	**PWM_CK_EN**	**UART1_CK_EN**	**UART0_CK_EN**	**RTC_CK_EN**	**WD_CK_EN**	**TIMER_CK_EN**

Bits	Descriptions	
[31:10]	**Reserved**	**Reserved**
[9]	**ADC_CK_EN**	Analog-Digital-Converter Clock Enable Control. 0 = Disable 1 = Enable
[8]	**SPIMS_CK_EN**	SPI (master & slave) Clock Enable Control. 0 = Disable 1 = Enable
[7]	**Reserved**	**Reserved**
[6]	**I2C_CK_EN**	I2C Clock Enable Control. 0 = Disable 1 = Enable
[5]	**PWM_CK_EN**	PWM_0 (channel 3-0) Clock Enable Control. 0 = Disable 1 = Enable
[4]	**UART1_CK_EN**	UART1 Clock Enable Control. 0 = Disable 1 = Enable
[3]	**UART0_CK_EN**	UART0 Clock Enable Control. 0 = Disable 1 = Enable

[2]	**RTC_CK_EN**	Real-Time-Clock APB interface Clock Control. This bit is used to control the APB clock only, The RTC engine clock source is from the 32.768 KHz crystal input. 0 = Disable 1 = Enable
[1]	**WD_CK_EN**	Watch Dog Clock Enable. The Watch Dog engine clock source is from the crystal input 0 = Disable 1 = Enable
[0]	**TIMER_CK_EN**	Timer Clock Enable. The Timer clock engine source is from the crystal input. 0 = Disable 1 = Enable

Clock Source Select Control Register （CLKSEL）

Before clock switch the related clock sources (pre-select and new-select) must be turn on.

Register	Address	R/W	Description	Reset Value
CLKSEL	CLK_BA + 10	R/W	Clock Source Select Control Register	0x0000_0000

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
ADC_S		**Reserved**					
7	6	5	4	3	2	1	0
UART_S		**APU_S**		**USB_S**		**HCLK_S**	

Bits	Descriptions	
[31:16]	**Reserved**	**Reserved**
[15:14]	**ADC_S**	ADC clock source select. 00: clock source from crystal clock in. 01: clock source from divided MPLL clock 1x: clock source from divided MPLL clock / 2
[13:8]	**Reserved**	**Reserved**
[7:6]	**UART_S**	UART clock source select. 00: clock source from crystal clock in. 01: clock source from divided MPLL clock 1x: clock source from divided MPLL clock / 2
[5:4]	**APU_S**	Audio-Process-Unit clock source select. 00: clock source from crystal clock in. 01: clock source from divided MPLL clock 1x: clock source from divided MPLL clock / 2
[3:2]	**USB_S**	USB clock source select. 00: clock source from crystal clock in. 01: clock source from divided MPLL clock 1x: clock source from divided MPLL clock / 2

[1:0]	**HCLK_S**	HCLK clock source select. [1:0] 00: clock source from crystal clock in. 01: clock source from divided MPLL clock 10: clock source from divided MPLL clock / 2 11: clock source from crystal 32k input

Clock Divider Register0 (CLKDIV0)

Register	Address	R/W	Description	Reset Value
CLKDIV0	CLK_BA_+ 14	R/W	Clock Divider Number Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
USB_N				**UART_N**			
15	14	13	12	11	10	9	8
APU_N							
7	6	5	4	3	2	1	0
APB_N		**Reserved**		**HCLK_N**			

Bits	Descriptions	
[31:24]	**Reserved**	**Reserved**
[23:20]	**USB_N**	USB clock divide number from USB clock source The USB clock frequency = (USB clock source frequency) / (USB_N + 1)
[19:16]	**UART_N**	UART clock divide number from UART clock source The UART clock frequency = (UART clock source frequency) / (UART_N + 1)
[15:8]	**APU_N**	APU clock divide number from APU clock source The APU clock frequency = (APU clock source frequency) / (APU_N + 1)
[7:6]	**APB_N**	APB clock divide number from CPU The APB clock frequency = (CPU clock frequency) / (APB_N + 1)
[5:4]	**Reserved**	**Reserved**
[3:0]	**HCLK_N**	HCLK clock divide number from HCLK clock source The HCLK clock frequency = (HCLK clock source frequency / 2) / (HCLK_N + 1)

Clock Divider Register1 (CLKDIV1)

Register	Address	R/W	Description	Reset Value
CLKDIV1	CLK_BA_+ 18	R/W	Clock Divider Number Register	0x0000_0000

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

Bits	Descriptions	
[31:24]	**Reserved**	**Reserved**
[23:16]	**ADC_N**	ADC engine clock divide number from ADC clock source The ADC engine clock frequency = (ADC engine clock source frequency) / (ADC_N + 1)
[15:0]	**Reserved**	**Reserved**

MPLL Control Register （MPLLCON）

The MPLL reference clock input is directly from the external clock input, and the other PLL control inputs are connected to bits of the registers.

Register	Address	R/W	Description	Reset Value
MPLLCON	CLK_BA + 20	R/W	MPLL Control Register	0x0001_4035

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved					**OE**	**BP**	**PD**
15	14	13	12	11	10	9	8
OUT_DV		**IN_DV**					**FB_DV**
7	6	5	4	3	2	1	0
FB_DV							

Bits	Descriptions	
[31:19]	**Reserved**	**Reserved**
[18]	**OE**	PLL OE (FOUT enable) pin Control 0: PLL FOUT enable 1: PLL FOUT is fixed low
[17]	**BP**	PLL Bypass Control 0: PLL is in normal mode (default) 1: PLL clock output is same as clock input (XTALin)
[16]	**PD**	Power Down Mode 0: PLL is in normal mode (default) 1: PLL is in power-down mode
[15:14]	**OUT_DV**	PLL Output Divider Control Pins (PLL_OD[1:0])
[13:9]	**IN_DV**	PLL Input Divider Control Pins (PLL_R[4:0])
[8:0]	**FB_DV**	PLL Feedback Divider Control Pins (PLL_F[6:0])

Output Clock Frequency Setting

$$FOUT = FIN \times \frac{NF}{NR} \times \frac{1}{NO}$$

Constrain:

1. $3.2MHz < FIN < 150MHz$
2. $800KHz < \frac{FIN}{NR} < 8MHz$
3. $200MHz < FCO = FIN \times \frac{NF}{NR} < 500MHz$

   $250MHz < FCO$ is preferred

FOUT	Output Clock Frequency
FIN	Input (Reference) Clock Frequency
NR	Input Divider (2 x (IN_DV + 2))
NF	Feedback Divider (2 x (FB_DV + 2))
NO	OUT_DV = "00" : NO = 1 OUT_DV = "01" : NO = 2 OUT_DV = "10" : NO = 2 OUT_DV = "11" : NO = 4

Default Setting

The default value: 0x4035
FIN = 12 MHz
NR = 2 x (0+2) = 4
NF = 2 x (53+2) = 110
NO = 2
FOUT = 12/4 x 110 x 1/2 = 165 MHz

6.4 SPI Synchronous Serial Interface Controller (Master Mode)

6.4.1 Overview

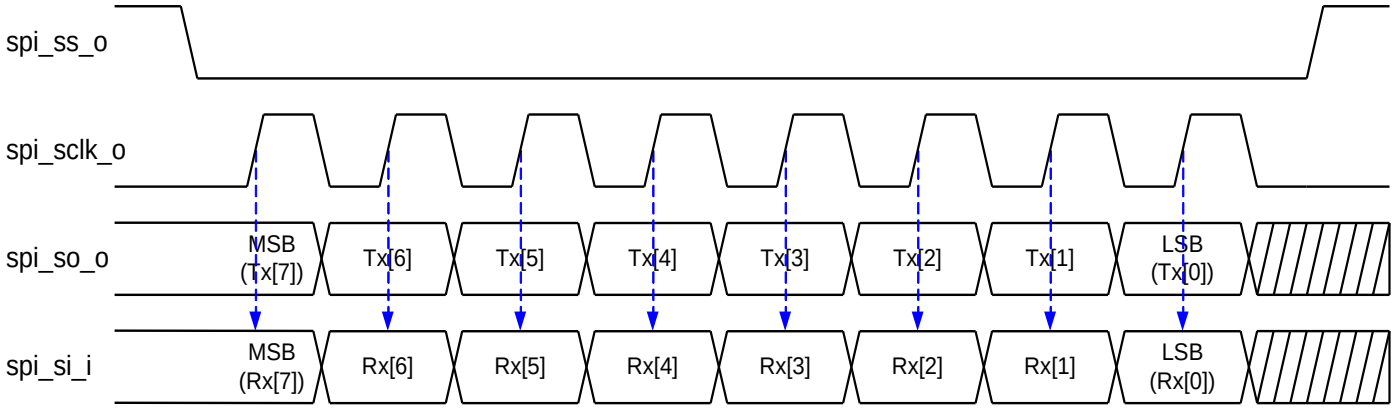
The SPI Synchronous Serial Interface controller performs a serial-to-parallel conversion on data received from the peripheral, and a parallel-to-serial conversion on data received from CPU. This controller can drive up to 2 external peripherals and is seen as the master. It can generate an interrupt signal when data transfer is finished and can be cleared by writing 1 to the interrupt flag. The active level of device/slave select signal can be chosen to low active or high active, which depends on the peripheral it's connected. Writing a divisor into DIVIDER register can program the frequency of serial clock output to peripherals. This controller contains four 32-bit transmit/receive buffers, and can provide 1 to 4 burst mode operation. The maximum bits can be transmitted/received is 32 bits in each transaction, and can transmit/receive data up to four successive transactions in one transfer.

6.4.2 Features

- AMBA AHB interface compatible
- Support SPI master mode
- Variable length of transaction bits up to 32 bits
- Provide burst mode operation, transmit/receive transaction can be executed up to four times in one transfer
- MSB or LSB first transaction
- 2 slave/device select lines. SPIM0_SS is a dedicated I/O. However SPIM1 needs a GPIO pin to be SPIM1_SS and it can't be configured as auto-select to perform DMM or DMA mode.
- Fully static synchronous design with one clock domain

6.4.3 SPIM Timing Diagram

The timing diagram of SPI transaction is shown as following:



CNTRL[LSB]=0, CNTRL[Tx_NUM]=0x0, CNTRL[Tx_BIT_LEN]=0x08,
CNTRL[Tx_NEG]=1, CNTRL[Rx_NEG]=0, SSR[SS_LVL]=0

SPI Timing

6.4.4 SPIM Programming Example without DMA

If you want to access a device with following specifications:

- Data is transferred with the MSB first
- Only one byte transmits/receives in a transfer
- Chip select signal is active low

You should do following actions basically (you should refer to the specification of device for the detailed steps):

1) Write a divisor into DIVIDER to determine the frequency of serial clock.
2) Write in SSR, set ASS = 0, SS_LVL = 0 and SSR[0] to 1 to activate the device you want to access.

When transmit (write) data to device:

3) Write the data you want to transmit into Tx0[7:0].

When receive (read) data from device:

4) Write in CNTRL, Tx_BIT_LEN = 0x08, Tx_NUM = 0x0, LSB = 0, SLEEP = 0x1 and GO_BUSY = 1 to start the transfer.
   –– Wait for interrupt or polling the GO_BUSY bit until it turns to 0
5) Read out the received data from Rx0 in received mode..
6) Go to 3) to continue data transfer or set SSR[0] to 0 to inactivate the device.

6.4.5 SPIM Programming Example with DMA

If users want to access a device with DMA function, 3 additional registers need to be configured. They are CODE_LEN, AHB_ADDR and SPI_ADDR. DMA function can be used to support loading boot code, reading data from system memory into peripherals or copy data from peripherals, reading data from peripherals into system memory. Users must define the length and destination and hardware will automatically move the desired length of code to specific target address.

6.4.5.1 Code Boot Process:

Step 1: Read the Check ID and code length in device.
Step 2:

1. Set the target memory address in AHB_ADDR (system memory address)
2. Set the boot code length which read from step 1 into CODE_LEN register
3. Set the SPI start address in SPI_ADDR (peripheral address)
4. Set SSR register to select spi slave. (no support ASS in dma mode)
5. Set the READ command (03) and 3-Byte SPI Start Address into Tx0, Tx1, Tx2, Tx3.
6. Set **SPI_CNTRL = 0x1a1345**.for control information.
7. Wait code read finish. Wait interrupt.
8. Set SSR register to un-select spi slave. (no support ASS in dma mode)

If used SPI flash supports other read mode, users can also use the following mode.

1. Fast read (0b), set read command (0b) into Tx0, & **SPI_CNTRL = 0x0b1a1b45**.
2. Fast dual read (3b), set read command (3b) into Tx0, & **SPI_CNTRL = 0x3b1a1b45**.

6.4.5.2 Move data from system memory to peripheral (Program SPI Flash):

Step 1: Erase the spi flash before program it.
Step 2:

1. Send Write Enable command to SPI flash
2. Set the source memory address in AHB_ADDR
3. Set the code length into CODE_LEN register
4. Set the spi start address in SPI_ADDR
5. Set SSR register to select spi slave. (no support ASS in dma mode)
6. Set the Page Program command (02) and 3-Byte SPI Start Address into Tx0, Tx1, Tx2, Tx3.
7. Set **SPI_CNTRL = 0x161345** for control information.
8. Wait code write finish. Wait interrupt
9. Set SSR register to un-select spi slave. (no support ASS in dma mode)
10. Check the BUSY status in SPI Flash

6.4.6 Direct memory mapping mode

Users can see SPI flash as a ROM when in direct memory mapping mode. This controller will convert the AHB cycle to SPI flash without CPU setting related SPI command. The only setting CPU needs to do is

to disable AHB master function (CNTRL[DIS_M] high), disable flash data read (CNTRL[F_DRD] low), set sleep interval to 1 (CNTRL[SLEEP] = 4'h1) and set SPI flash read command(CNTRL[SPI_MODE] 0x03, 0x0b, or 0x3b). Then users can access SPI flash as a ROM module. Direct memory mapping mode supports these following modes:

Standard Read : Set CNTRL = 0x0332_1344
Fast Read : Set CNTRL = 0x0b32_1344
Fast dual Read : Set CNTRL = 0x3b32_1344

Note1: In direct memory mapping mode, the SPI flash IP will pre-fetch 4-word flash data after a direct memory mapping access. If users want to change the control registers (CNTRL, SSR, DIVIDER, Tx, Rx) after the direct mapping access, remember to check the busy state(GO_BUSY = 0).

Note2: Sleep interval (CNTRL[SLEEP]) can't set to zero when DIVIDER is zero.

6.4.7 SPIM Serial Interface Control Registers Mapping

R: read only, **W**: write only, **R/W**: both read and write, **C**: Only value 0 can be written

Register	Address	R/W/C	Description	Reset Value
Base Address: 0xB100_7000				
CNTRL	SPI_BA + 0x00	R/W	Control and Status Register	0x0000_0004
DIVIDER	SPI_BA + 0x04	R/W	Clock Divider Register	0x0000_0000
SSR	SPI_BA + 0x08	R/W	Slave Select Register	0x0000_0000
Reserved	SPI_BA + 0x0C	N/A	Reserved	0xFFFF_FFFF
Rx0	SPI_BA + 0x10	R	Data Receive Register 0	0x0000_0000
Rx1	SPI_BA + 0x14	R	Data Receive Register 1	0x0000_0000
Rx2	SPI_BA + 0x18	R	Data Receive Register 2	0x0000_0000
Rx3	SPI_BA + 0x1C	R	Data Receive Register 3	0x0000_0000
Tx0	SPI_BA + 0x20	R/W	Data Transmit Register 0	0x0000_0000
Tx1	SPI_BA + 0x24	R/W	Data Transmit Register 1	0x0000_0000
Tx2	SPI_BA + 0x28	R/W	Data Transmit Register 2	0x0000_0000
Tx3	SPI_BA + 0x2C	R/W	Data Transmit Register 3	0x0000_0000
AHB_ADDR	SPI_BA + 0x30	R/W	AHB memory address	0x0000_0000
CODE_LEN	SPI_BA + 0x34	R/W	Boot code length	0x0000_0000
Reserved	SPI_BA + 0x38	N/A	Reserved	0xFFFF_FFFF
Reserved	SPI_BA + 0x3C	N/A	Reserved	0xFFFF_FFFF
SPIM_ADDR	SPI_BA + 0x40	N/A	SPI Flash Start Address	0x0000_0000

NOTE1: When software programs CNTRL, the GO_BUSY bit should be written last.

Control and Status Register (CNTRL)

Register	Address	R/W/C	Description	Reset Value
CNTRL	SPI_BA + 0x00	R/W	Control and Status Register	0x0000_0004

31	30	29	28	27	26	25	24
SPI_MODE							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
OEN	**COMMAND**	**DIS_M**	**BOOT_SPI**	**F_DRD**	**F_TYPE**	**IE**	**IF**
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
SLEEP				**Insert_ dummy**	**LSB**	**Tx_NUM**	
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Tx_BIT_LEN					**Tx_NEG**	**Rx_NEG**	**GO_BUSY**

Bits	Descriptions	
[31:24]	**SPIM_MODE**	**SPI read mode selection** 8'h03: standard read mode 8'h0b: fast read mode 8'h3b: fast read dual output mode
[23]	**OEN**	The direction control of spi_so_o (see block diagram) In most case, spi_so_o is output. But in flash fast dual read mode, spi_so_o is bi-direction pin. 0: spi_so_o is output 1: spi_so_o is input
[22]	**COMMAND**	**For cipher IP** 0: current transfer is data phase 1: current transfer is command phase
[21]	**DIS_M**	**Disable AHB master in boot mode**

		NOTE: When want to access SPI flash through direct memory mapping, please set this bit high**.**
[20]	**BOOT_SPIM**	**SPI ROM Boot /Page Write enable** • 0 = **Disable** ROM boot or page write operation. • 1 = **Enable** ROM boot or page write operation. **NOTE:** When want to access SPI flash through direct memory mapping, please set this bit high.
[19]	**F_DRD**	**Flash Data Read** • 0 = **Enable** write data to Flash operation when BOOT_SPI high. • 1 = **Enable** read data from Flash operation when BOOT_SPI high. **NOTE: When want to access SPI flash through direct memory mapping, please set this bit LOW.**
[18]	**F_TYPE**	**Flash Type** • 0 = **SST** 16Mbit SPI Serial Flash (ST25VF016B). • 1 = **PMC** 512Kbit Serial Flash Memory with SPI Bus Interface
[17]	**IE**	**Interrupt Enable** • 0 = **Disable** SPI Interrupt. • 1 = **Enable** SPI Interrupt.
[16]	**IF**	**Interrupt Flag** • 0 = It indicates that the transfer dose not finish yet. • 1 = It indicates that the transfer is done. The interrupt flag is set if it was enable. **NOTE:** This bit is read only, but can be cleared by writing 1 to this bit.
[15:12]	**SLEEP**	**Suspend Interval** These four bits provide the configuration of suspend interval between two successive transmit/receive in a transfer. The default value is 0x0. When CNTRL[Tx_NUM] = 00, setting this field has no effect on transfer. The desired interval is obtained according to the following equation (from the last falling edge of current sclk to the first rising edge of next sclk):

		(CNTRL[SLEEP] + 2)*period of SCLK SLEEP = 0x0 … 2 SCLK clock cycle SLEEP = 0x1 … 3 SCLK clock cycle …… SLEEP = 0xe … 16 SCLK clock cycle SLEEP = 0xf … 17 SCLK clock cycle Note: SLEEP can't set to zero when DIVIDER is zero
[11]	**Insert_Dummy**	**This bit is used to insert a dummy phase in SPI flash fast dual read/fast read mode when in DMA mode.**
[10]	**LSB**	**Send LSB First** • 0 = The **MSB** is transmitted/received first (which bit in TxX/RxX register that is depends on the Tx_BIT_LEN field in the CNTRL register). • 1 = The **LSB** is sent first on the line (bit TxX[0]), and the first bit received from the line will be put in the LSB position in the Rx register (bit RxX[0]).
[9:8]	**Tx_NUM**	**Transmit/Receive Numbers** This field specifies how many transmit/receive numbers should be executed in one transfer. 00 = Only one transmit/receive will be executed in one transfer. 01 = Two successive transmit/receive will be executed in one transfer. 10 = Three successive transmit/receive will be executed in one transfer. 11 = Four successive transmit/receive will be executed in one transfer.
[7:3]	**Tx_BIT_LEN**	**Transmit Bit Length** This field specifies how many bits are transmitted in one transmit/receive. Up to 32 bits can be transmitted. Tx_BIT_LEN = 0x01 … 1 bit

		Tx_BIT_LEN = 0x02 … 2 bits …… Tx_BIT_LEN = 0x1f … 31 bits Tx_BIT_LEN = 0x00 … 32 bits
[2]	**Tx_NEG**	**Transmit On Negative Edge (Read Only) --- 1'b1** • This module only supports transmitting on negative edge
[1]	**Rx_NEG**	**Receive On Negative Edge (Read Only) --- 1'b0** • This module only supports receiving on positive edge
[0]	**GO_BUSY**	**Go and Busy Status** • 0 = Writing 0 to this bit has no effect. • 1 = Writing 1 to this bit starts the transfer. This bit remains set during the transfer and is automatically cleared after transfer finished. **NOTE:** All registers should be set before writing 1 to the GO_BUSY bit in the CNTRL register. When a transfer is in progress, writing to any register of the SPI master core has no effect.

Divider Register (DIVIDER)

Register	Address	R/W/C	Description	Reset Value
DIVIDER	SPIM_BA + 0x04	R/W	Clock Divider Register	0x0000_0000

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

Bits	Descriptions	
[31:23]	**Reserved**	**Reserved**
[22:20]	**SCLK_IN_DLY**	**Serial CLK Input Delay register** Set this register to adjust the spi_sclki clock input delay. There are total 8 buffers in this delay path. The actual delay value depends on process. 000: one buffer delay …… 111: 8 buffer delay
[19:16]	**IDLE_CNT**	The idle interval of slave select. In direct memory mapping mode, IDLE_CNT is used to ensure the slave select idle interval in peripheral specification between two successive flash access.
[15:0]	**DIVIDER**	**Clock Divider Register** The value in this field is the frequency divider of the system clock pclk to generate the serial clock on the output spi_sclk_o. The desired frequency is obtained according to the following equation: $f_{sclk} = \frac{f_{pclk}}{(DIVIDER)*2}$

		NOTE: When set DIVIDER to zero, SPI clock will be equal to engine clock. **NOTE: when set DIVIER to zero, sleep(CNTRL[SLEEP]) can't set to zero.**

Slave Select Register (SSR)

Register	Address	R/W/C	Description	Reset Value
SSR	SPIM_BA + 0x08	R/W	Slave Select Register	0x0000_0000

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							
7	6	5	4	3	2	1	0
Reserved				**ASS**	**SS_LVL**	**Reserved**	**SSR**

Bits	Descriptions	
[31:4]	**Reserved**	**Reserved**
[3]	**ASS**	**Automatic Slave Select** • 0 = If this bit is cleared, slave select signals are asserted and de-asserted by setting and clearing related bits in SSR register. • 1 = If this bit is set, spi_ss_o signals are generated automatically. It means that device/slave select signal, which is set in SSR register is asserted by the SPI controller when transmit/receive is started by setting CNTRL[GO_BUSY], and is de-asserted after every transmit/receive is finished.
[2]	**SS_LVL**	**Slave Select Active Level** It defines the active level of device/slave select signal (spi_ss_o). • 0 = the spi_ss_o slave select signal is active **Low**. • 1 = the spi_ss_o slave select signal is active **High**.
[1]	**Reserved**	**Reserved**
[0]	**SSR**	**Slave Select Register** If SSR[ASS] bit is cleared, writing 1 to any bit location of this field sets the proper spi_ss_o line to an active state and writing 0 sets the line back to inactive state.

		If SSR[ASS] bit is set, writing 1 to any bit location of this field will select appropriate spi_ss_o line to be automatically driven to active state for the duration of the transmit/receive, and will be driven to inactive state for the rest of the time. (The active level of spi_ss_o is specified in SSR[SS_LVL]). **NOTE:** This interface can only drive one device/slave at a given time. Therefore, the slave select of the selected device must be set to its active level before starting any read or write transfer.

Data Receive Register 0/1/2/3 (RX0/0/2/3)

Register	Address	R/W/C	Description	Reset Value
Rx0	SPIM_BA + 0x10	R	Data Receive Register 0	0x0000_0000
Rx1	SPIM_BA + 0x14	R	Data Receive Register 1	0x0000_0000
Rx2	SPIM_BA + 0x18	R	Data Receive Register 2	0x0000_0000
Rx3	SPIM_BA + 0x1C	R	Data Receive Register 3	0x0000_0000

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

Bits	Descriptions	
[31:0]	**Rx**	**Data Receive Register** The Data Receive Registers hold the value of received data of the last executed transfer. Valid bits depend on the transmit bit length field in the CNTRL register. For example, if CNTRL[Tx_BIT_LEN] is set to 0x08 and CNTRL[Tx_NUM] is set to 0x0, bit Rx0[7:0] holds the received data.

Data Transmit Register 0/1/2/3 (TX0/1/2/3)

Register	Address	R/W/C	Description	Reset Value
TX0	SPIM_BA + 0x20	R/W	Data Transmit Register 0	0x0000_0000
TX1	SPIM_BA + 0x24	R/W	Data Transmit Register 1	0x0000_0000
TX2	SPIM_BA + 0x28	R/W	Data Transmit Register 2	0x0000_0000
TX3	SPIM_BA + 0x2C	R/W	Data Transmit Register 3	0x0000_0000

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

Bits	Descriptions	
[31:0]	**Tx**	**Data Transmit Register** The Data Transmit Registers hold the data to be transmitted in the next transfer. Valid bits depend on the transmit bit length field in the CNTRL register. For example, if CNTRL[Tx_BIT_LEN] is set to 0x08 and the CNTRL[Tx_NUM] is set to 0x0, the bit Tx0[7:0] will be transmitted in next transfer. If CNTRL[Tx_BIT_LEN] is set to 0x00 and CNTRL[Tx_NUM] is set to 0x3, the core will perform four 32-bit transmit/receive successive using the same setting (the order is Tx0[31:0], Tx1[31:0], Tx2[31:0], Tx3[31:0]).

AHB Address Register (AHB_ADDR)

Register	Address	R/W/C	Description	Reset Value
AHB_ADDR	SPIM_BA + 0x30	R/W	AHB address Register	0x0000_0000

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

Bits	Descriptions	
[31:0]	**AHB_ADDR**	**AHB address** **This is the system memory address when in DMA mode.**

Code Length Register (CODE_LEN)

Register	Address	R/W/C	Description	Reset Value
CODE_LEN	SPIM_BA + 0x34	R/W	Code length Register	0x0000_0000

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

Bits	Descriptions	
[31:24]	**Reserved**	**Reserved**
[23:0]	**CODE_LEN**	Code length(data length) when users want to use DMA function.

SPI Flash Start Address Register (SPIM_ADDR)

Register	Address	R/W/C	Description	Reset Value
SPIM_ADDR	SPIM_BA + 0x40	R/W	SPI Flash Start Address Register	0x0000_0000

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

Bits	Descriptions	
[31:24]	**Reserved**	**Reserved**
[23:2]	**SPIM_ADDR**	**SPI Flash start access address** Note: SPI Flash starting address must be word alignment
[1:0]	**Reserved**	**Reserved**

6.5 Audio Processing Unit

The main purpose of Audio Processing Unit (APU) is used to playback the audio data (PCM format) which CPU decoded and stored in global RAM. The APU built in a monophonic DAC with 16-bit resolution per channel which supports speakerphone output and monophonic output for headphone. The APU is composed of an AHB Master and built in FIFO and timer.

6.5.1 Overview and Features

- Built in a monophonic DAC with 16-bit resolution per channel
- AHB Master with DMA.
- Built in FIFO with length 16Bytes x 2.

6.5.2 APU Functional Description

6.5.2.1 Audio DAC

- Monophonic Digital to Analog Converter with 16-bit resolution per channel.
- Supports speakerphone output and stereophonic output for headphone.

6.5.2.2 Register Bank

- AHB Slave interface on AHB.
- A bridge that CPU control and observe the state of APU.

6.5.2.3 Buffer Interface and Timer

- AHB Master interface on AHB.
- Read Audio PCM data from global RAM
- Built in FIFO with length 16Bytes x 2.
- Built in timer to generate conversion trigger signal automatically.

6.5.3 AUDIO DAC Clock

This is the clock input for DAC from clock control module. You can set CLKSEL[5:4] and CLKDIV0[15:8] to generate a clock with needed frequency. The frequency of this clock must be equal to (input audio data sampling rate) x 128. For example, if the sampling rate is 48KHz, then the clock should be 128 x 48KHz = 6.144Mhz. The APU module internally handles a 7-bit counter that it will generate a STORBE signal to DAC whenever the counter reaches 128. This makes DAC to output the audio data with correct sampling rate.

6.5.4 APU Run Procedures

1. Setup clock source :
   - If the PLL output frequency set by MPLLCON is 160Mhz and you the sample rate of input data is 48KHz, the AUDIO_DAC clock should be set to 6.144Mhz. Therefore, CLKSEL[5:4] and CLKDIV0[15:8] should be set a proper value to divide the clock source by about 26 (166/6.144)). We my set CLKSEL[5:4]=2'b01 and set CLKDIV0[15:8]=0x19.

2. Set base address and threshold addresses
   - The APU implement the ping-pong buffer mechanism and the buffers are consecutive. You can set the start address of first buffer in BSAD register, set the end address of fist buffer in THAD1 register, and set the end address of second buffer in THAD2 register. Remember to set APUINT register to enable threshold 1/2 interrupts. If the registers are set properly, every time the APU reach the end of each one buffer, it will issue an interrupt and then you can update the buffer.
3. Reset APU before start
   - Set bit 16 of APUCON register to 1 and then set it to 0 again. This action will reset internal buffers and registers. Remember to do this step before you start to run APU.
4. Start APU
   - Set bit 0 of APUCON register to 1. This makes APU start to transfer audio data from buffer to DAC.

6.5.5 APU Control Register Mapping

R: read only, **W**: write only, **R/W**: both read and write, **C**: Only value 0 can be written

Register	Address	R/W	Description	Reset Value
APU_BA = 0xB100_8000				
APUCON	APU_BA + 0x00	R/W	APU Control Register	0x0000_0000
PARCON	APU_BA + 0x04	R/W	Parameter Control Register	0x0000_0001
PDCON	APU_BA + 0x08	R/W	Power Down Control Register	0x0001_0000
APUINT	APU_BA + 0x0C	R/W	APU Interrupt Register	0x0000_0000
RAMBSAD	APU_BA + 0x10	R/W	RAM Base Address Register	0x0000_0000
THAD1	APU_BA + 0x14	R/W	Threshold 1 Address Register	0x0000_0000
THAD2	APU_BA + 0x18	R/W	Threshold 2 Address Register	0x0000_0000
CURAD	APU_BA + 0x1C	R	Current Access RAM Address Register	0x0000_0000
Reserved	APU_BA + 0x20	R/W	Reserved	0x7777_7777
Reserved	APU_BA + 0x24	R/W	Reserved	0x000D_0077
Reserved	APU_BA + 0x2C	R/W	Reserved r	0x0000_0000

6.5.6 APU Control Registers

APU Control Register

Register	Address	R/W	Description	Reset Value
APUCON	APU_BA+0x00	R/W	APU Control Register	0x0000_0000

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

Bits	Descriptions	
[31:17]	**Reserved**	**Reserved**
[16]	**APURST**	**APU Reset** 0 = No action 1 = Reset the whole ADC except register value.
[15:1]	**Reserved**	**Reserved**
[0]	**APURUN**	**APU Run** 0 = Disable 1 = Enable

Parameter Control Register

Register	Address	R/W	Description	Reset Value
PARCON	APU_BA+0x04	R/W	Parameter Control Register	0x0000_0001

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

Bits	Descriptions	
[31:26]	**Reserved**	**Reserved**
[25]	**ZERO_EN**	**Zero cross detection enable** 0 = Disable 1 = Enable
[24]	**Reserved**	**Reserved** **Fix 0**
[23:17]	**Reserved**	**Reserved**
[16]	**SWAP**	**PCM data format** 0 = MSB is sample data 2, LSB is sample data 1 1 = MSB is sample data 1, LSB is sample data 2
[15:0]	**Reserved**	Reserved

APU Power Down Control Register

Register	Address	R/W	Description	Reset Value
PDCON	APU_BA+0x08	R/W	Power Down Control Register	0x0001_0000

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

Bits	Descriptions	
[31:17]	**Reserved**	**Reserved**
[16]	**ANA_PD**	**Audio DAC Power Down** 0 = Normal operation 1 = Power down
[15:0]	**Reserved**	**Reserved**

APU Interrupt Register

Register	Address	R/W	Description	Reset Value
APUINT	APU_BA+0x0C	R/W	APU Interrupt Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved						**T2INTEN**	**T1INTEN**
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved						**T2INTS**	**T1INTS**

Bits	Descriptions	
[31:18]	**Reserved**	**Reserved**
[17]	**T2INTEN**	**Threshold 2 Interrupt Enable** 0 = Disable 1 = Enable
[16]	**T1INTEN**	**Threshold 1 Interrupt Enable** 0 = Disable 1 = Enable
[15:2]	**Reserved**	**Reserved**
[1]	**T2INTS**	**Threshold 2 Interrupt Status** APU fetch data from Threshold 2 complete. Write 0 to clear it.
[0]	**T1INTS**	**Threshold 1 Interrupt Status** APU fetch data from Threshold 1 complete. Write 0 to clear it.

RAM Base Address Register

Register	Address	R/W	Description	Reset Value
RAMBSAD	APU_BA+0x10	R/W	RAM Base Address Register	0x0000_0000

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

Bits	Descriptions	
[31:0]	**BSAD**	**Global RAM Base Address**

Threshold 1 Address Register

Register	Address	R/W	Description	Reset Value
THAD1	APU_BA+0x14	R/W	Threshold 1 Address Register	0x0000_0000

31	30	29	28	27	26	25	24
TH1[31:24]							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
TH1[23:16]							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
TH1[15:8]							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TH1[7:0]							

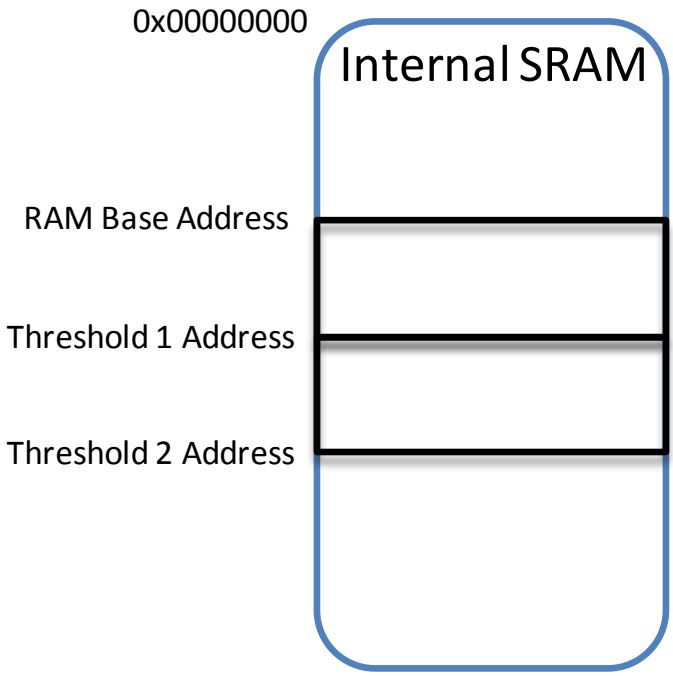
Bits	Descriptions	
[31:0]	**TH1**	**Threshold 1 Address**

Threshold 2 Address Register

Register	Address	R/W	Description	Reset Value
THAD2	APU_BA+0x18	R/W	Threshold 2 Address Register	0x0000_0000

31	30	29	28	27	26	25	24
TH2[31:24]							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
TH2[23:16]							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
TH2[15:8]							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TH2[7:0]							

Bits	Descriptions	
[31:0]	**TH2**	**Threshold 2 Address**



Current Address Register

Register	Address	R/W	Description	Reset Value
CURAD	APU_BA+0x1C	R	Current Access RAM Address Register	0x0000_0000

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

Bits	Descriptions	
[31:0]	**CURAD**	**Current APU Access RAM Address**

6.6 SRAM Controller

6.6.1 Overview

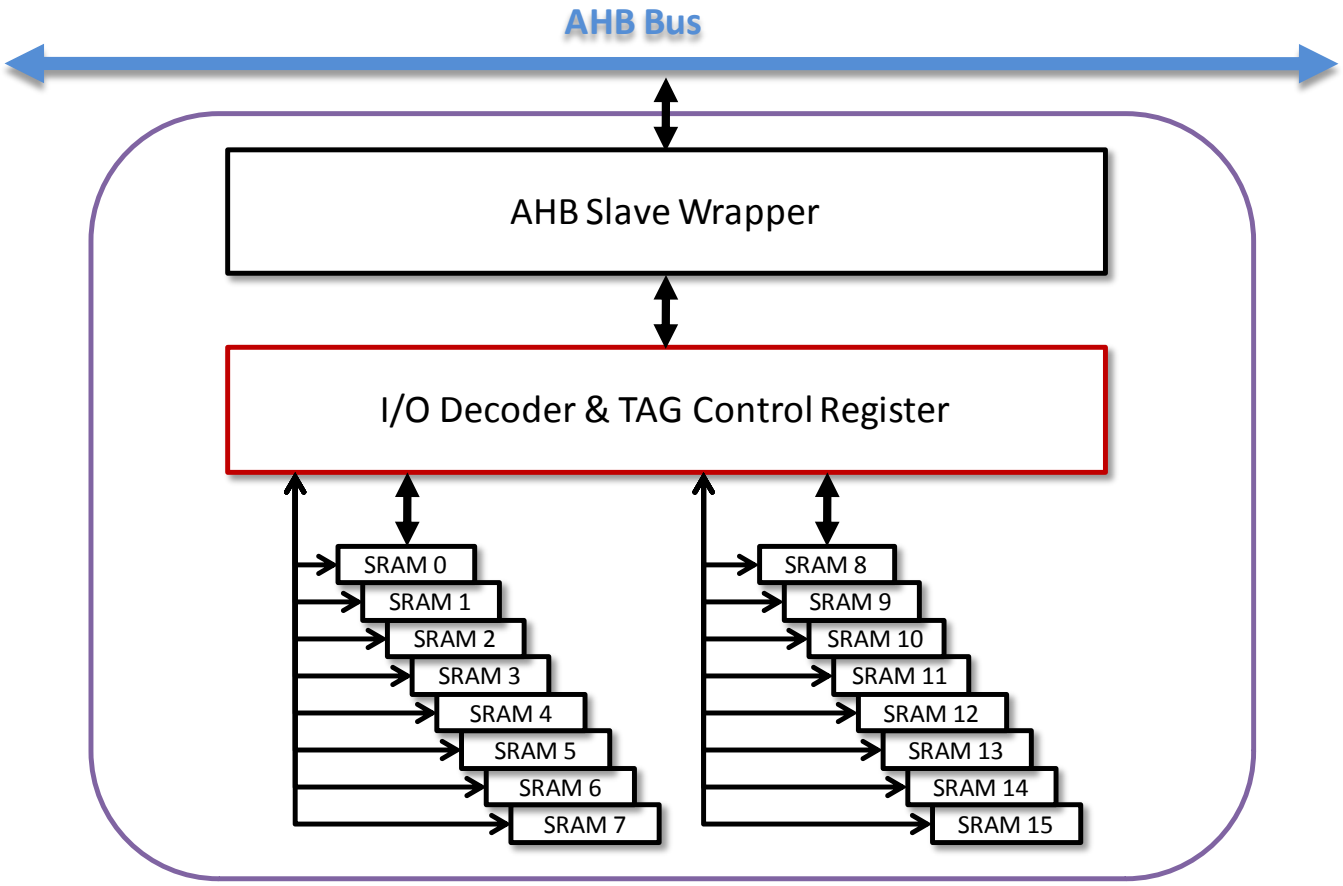
The SRAM controller is design for program code and data storage. It's an AHB slave and SRAM size is up to 32KB. This 32KB memory is separated into 16 memory block and the size of each memory block is 2KB. Each memory block could be randomly mapped to any 2KB space of 0x0000_0000 ~ 0x1FFF_FFFF of system memory by modifying the control register. Each 2KB memory block could also be disabled individually by modifying control register.
By default, these 16 2KB memory blocks are all enabled and mapped to 0x0000_0000 ~ 0x0000_7FFF sequentially.

6.6.2 Features

- Support 1 AHB slave interface
- Support 16 separated 2KB memory block and SRAM size is up to 32KB
- Support random memory address mapping in 2KB space of 0x0000_0000 ~ 0x1FFF_FFFF of system memory

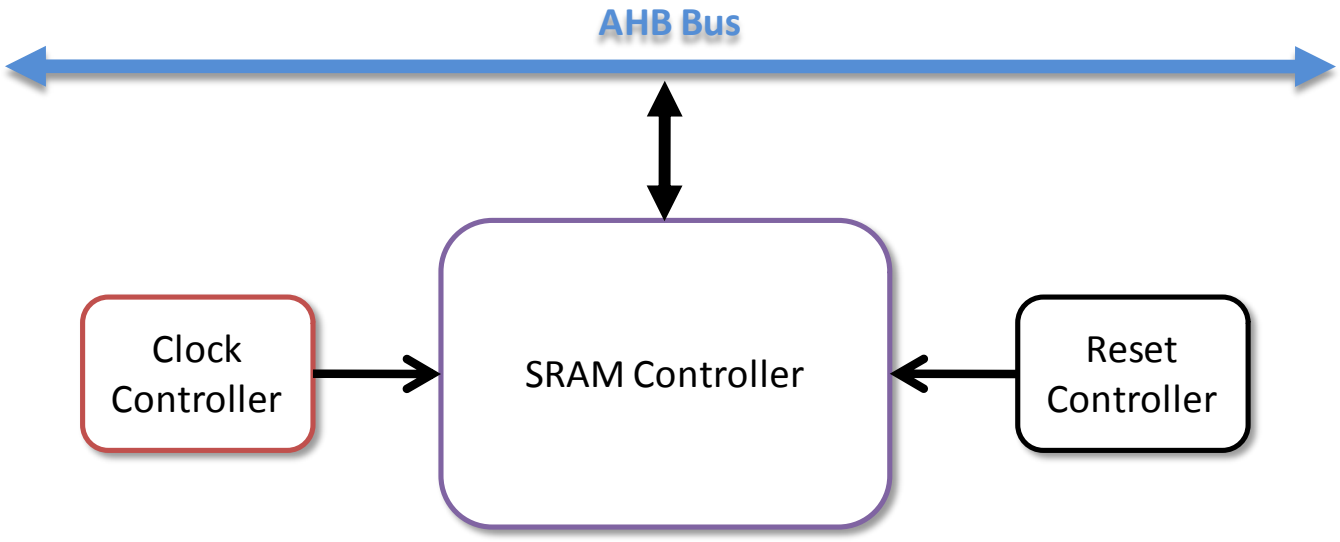
6.6.3 SRAM Block Diagram

The block diagram of SRAM Controller is depicted as following:



6.6.4 SRAM System Diagram

The following diagram briefs the related circuit with SRAM Controller:



6.6.5 SRAM Function Description

It's an AHB slave and SRAM size is up to 32KB. The 32KB memory is separated into 16 memory block and the size of each memory block is 2KB. Each memory block could be randomly mapped to any 2KB space of 0x0000_0000 ~ 0x1FFF_FFFF of system memory. For this purpose, 16 tag registers are implemented to keep the base address of each 2KB memory block. Besides, each 2KB memory block could also be disabled individually by modifying control register. 16 memory block enable bits are implemented for this purpose.

6.6.6 SRAM Register Mapping

Register	Address	R/W	Description	Reset Value
SRAMCTRL_BA = 0xB100_4000				
SCTRL0	SRAMCTRL_BA + 000	R/W	SRAM Control Register 0	0x0000_0001
SCTRL1	SRAMCTRL_BA + 004	R/W	SRAM Control Register 1	0x0000_0801
SCTRL2	SRAMCTRL_BA + 008	R/W	SRAM Control Register 2	0x0000_1001
SCTRL3	SRAMCTRL_BA + 00C	R/W	SRAM Control Register 3	0x0000_1801
SCTRL4	SRAMCTRL_BA + 010	R/W	SRAM Control Register 4	0x0000_2001
SCTRL5	SRAMCTRL_BA + 014	R/W	SRAM Control Register 5	0x0000_2801
SCTRL6	SRAMCTRL_BA + 018	R/W	SRAM Control Register 6	0x0000_3001
SCTRL7	SRAMCTRL_BA + 01C	R/W	SRAM Control Register 7	0x0000_3801
SCTRL8	SRAMCTRL_BA + 020	R/W	SRAM Control Register 8	0x0000_4001
SCTRL9	SRAMCTRL_BA + 024	R/W	SRAM Control Register 9	0x0000_4801
SCTRL10	SRAMCTRL_BA + 028	R/W	SRAM Control Register 10	0x0000_5001
SCTRL11	SRAMCTRL_BA + 02C	R/W	SRAM Control Register 11	0x0000_5801
SCTRL12	SRAMCTRL_BA + 030	R/W	SRAM Control Register 12	0x0000_6001
SCTRL13	SRAMCTRL_BA + 034	R/W	SRAM Control Register 13	0x0000_6801
SCTRL14	SRAMCTRL_BA + 038	R/W	SRAM Control Register 14	0x0000_7001
SCTRL15	SRAMCTRL_BA + 03C	R/W	SRAM Control Register 15	0x0000_7801

6.6.6.1 Register Descriptions

SRAM Control Register 0 (SCTRL0 ~ SCTRL15)

Register	Address	R/W/C	Description	Reset Value
SCTRL0	SRAMCTRL_BA+0x000	R/W	SRAM Control Register 0	0x0000_0001
SCTRL1	SRAMCTRL_BA+0x004	R/W	SRAM Control Register 1	0x0000_0801
SCTRL2	SRAMCTRL_BA+0x008	R/W	SRAM Control Register 2	0x0000_1001
SCTRL3	SRAMCTRL_BA+0x00C	R/W	SRAM Control Register 3	0x0000_1801
SCTRL4	SRAMCTRL_BA+0x010	R/W	SRAM Control Register 4	0x0000_2001
SCTRL5	SRAMCTRL_BA+0x014	R/W	SRAM Control Register 5	0x0000_2801
SCTRL6	SRAMCTRL_BA+0x018	R/W	SRAM Control Register 6	0x0000_3001
SCTRL7	SRAMCTRL_BA+0x01C	R/W	SRAM Control Register 7	0x0000_3801
SCTRL8	SRAMCTRL_BA+0x020	R/W	SRAM Control Register 8	0x0000_4001
SCTRL9	SRAMCTRL_BA+0x024	R/W	SRAM Control Register 9	0x0000_4801
SCTRL10	SRAMCTRL_BA+0x028	R/W	SRAM Control Register 10	0x0000_5001
SCTRL11	SRAMCTRL_BA+0x02C	R/W	SRAM Control Register 11	0x0000_5801
SCTRL12	SRAMCTRL_BA+0x030	R/W	SRAM Control Register 12	0x0000_6001
SCTRL13	SRAMCTRL_BA+0x034	R/W	SRAM Control Register 13	0x0000_6801
SCTRL14	SRAMCTRL_BA+0x038	R/W	SRAM Control Register 14	0x0000_7001
SCTRL15	SRAMCTRL_BA+0x03C	R/W	SRAM Control Register 15	0x0000_7801

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

Bits	Descriptions	
[31:29]	**Reserved**	**Reserved**
[28:11]	**TAG**	**TAG Address** This field keeps the base address of each 2KB memory block. Once the address bits [28:11] from system bus are the same with the content of this filed, and the **VALID** flag is enabled, the related memory block will be opened for access.

[10:1]	**Reserved**	**Reserved**
[0]	**VALID**	**TAG Valid Flag** This bit indicates if the TAG value is valid. This bit 1'b0 indicates the corresponding 2KB memory block was disabled and inaccessible. 0 = corresponding 2KB memory block is disabled 1 = corresponding 2KB memory block is enabled

6.7 USB Device Controller

6.7.1 Overview

The USB device is an interface that transmits and receives data packets between host and USB device controller. It also handles the routing data between the bus interface and various endpoints on the device controller.

On the device controller, it includes the AHB bus and USB bus which comes from the USB PHY transceiver. The AHB bus includes the slave interface only and the CPU programs the USB controller registers through it. There are 512 bytes internal SRAM as USB buffer in the device controller. For IN or OUT transfer, the USB device controller needs to write data to SRAM or read data from SRAM through the AHB slave interface or SIE. The BUFSEGx define the effective starting address for each endpoint buffer on the SRAM. The USB device controller is complaint with USB full speed device and it contains 6 configurable endpoints.

These endpoints could be configured as IN, OUT or ISO state on CFGx[6:4] and the endpoint number can be set on CFGx[3:0]. The transmit length in each endpoint is defined in MXPLD. Most handshakes between Host and Device are handled by hardware. Any USB event will cause an interrupt, and user can just check related event flags in EVF to acknowledge the events and store required data into buffer, which is then sent to host by hardware. A software-disable function is also available for this USB device, which simulates the disconnection of this device from the host.

6.7.2 Features

This Universal Serial Bus (USB) performs a serial interface with a single connector type for attaching all USB peripherals to the host system. Following is the feature list of this USB.

- Conforming to USB2.0 full speed device
- Full Speed 12Mbps.
- Provide 1 interrupt source with 4 interrupt events.
- Support Control, Bulk In/Out, Interrupt and Isochronous transfers.
- Suspend when no bus signaling for 3 ms.
- Provide 6 endpoints.
- Include 512 Bytes internal SRAM as USB buffer.
- Provide remote wakeup capability.

6.7.3 Functional Descriptions

6.7.3.1 SIE (Serial Interface Engine)

The SIE is the front-end of the device controller and handles most of the USB packet protocol. The SIE typically comprehends signaling up to the transaction level. The functions that it handles could include:

- Packet recognition, transaction sequencing
- SOP, EOP, RESET, RESUME signal detection/generation
- Clock/Data separation
- NRZI Data encoding/decoding and bit-stuffing
- CRC generation and checking (Token and Data)
- Packet ID (PID) generation and checking/ decoding
- Serial-Parallel/ Parallel-Serial conversion

6.7.3.2 UIE

The UIE is the endpoints management. All the operations include Control, Bulk, Interrupt and Isochronous transfer are implemented in it.

6.7.3.3 Digital Phase Lock Loop

The bit rate of USB data is 12MHz. The DPLL use the 48MHz which comes from the clock control to lock the input data RXDP and RXDM. The 12MHz bit rate clock is also converted from DPLL.

6.7.3.4 Floating De-bounce

A USB device may be plug-in or unplug from the USB. In order to monitor the state of a USB device when it is detached from the USB, the device controller provides hardware de-bounce for USB floating detect interrupt to avoid bounce problems on USB plug in and unplug. Floating detect interrupt appears about 10 ms later than USB plug-in and unplug. A user can acknowledge USB plug-in/unplug by reading SFR "FLODET", and should understand that the flag in "FLODET" represents the current state on the bus without de-bounce. If the user poling this flag to check USB state, he/she must add software de-bounce if necessary.

6.7.3.5 Interrupt

This USB provides 1 interrupt source with 4 interrupt events (WAKEUP, FLO, USB, BUS). WAKEUP interrupt is for stop wakeup only, FLO interrupt is for USB plug-in or unplug, USB event notifies users of

some USB requests, like IN ACK, OUT ACK etc., and BUS event notifies users of some bus events, like suspend, resume, etc. User must enable both AIC and IEF of USB to enable USB interrupts.

Wakeup interrupt is only present after stop wakeup. After the IC enters power down mode, any change on D+, D- and floating detect pin can wake up NUC501 (provided that USB wakeup function is enabled). If this change is not desired, for example, a noise on floating detect pin, no interrupt but wakeup interrupt will occur. After USB wakeup, this interrupt will occur when no other USB interrupt events are present for more than 20mS.

USB interrupt is to notify users of any USB event on the bus, and a user can read SFR "STSX" and "EPTF" to acknowledge what kind of request is to which endpoint and take necessary responses.

Same as USB interrupt, BUS interrupt notifies users of some bus events, like USB reset, suspend, timeout, and resume. A user can read SFR "ATTR" to acknowledge bus events.

6.7.3.6 Power Saving

USB turns off PHY automatically to save power while NUC501 enter power down mode. Furthermore, a user can write 0 into SFT ATTR[4] to turn off PHY under special circumstances like suspend to save power.

6.7.4 Memory Mapping

Address	Size	Type	Description
USB_BA + (000h ~ 0FFh)	256 Bytes	Register	Special function register
USB_BA + (100h ~ 1FFh)	256 Bytes	SRAM	USB buffer

6.7.5 USB Control Registers Mapping

Register	Address	R/W	Description	Reset Value
USB_BA = 0xB100_9000				
IEF	USB_BA+0x000	R/W	Interrupt Enable Flag	0x0000_0000
EVF	USB_BA+0x004	R	Interrupt Event Flag	0x0000_0000
FADDR	USB_BA+0x008	R/W	Function Address	0x0000_0000
STS	USB_BA+0x00C	R, W	System state	0x0000_00x0
ATTR	USB_BA+0x010	R/W	Bus state & attribution	0x0000_0040
FLODET	USB_BA+0x014	R	Floating detect	0x0000_0000
BUFSEG	USB_BA+0x018	R/W	Buffer Segmentation	0x0000_0000
BUFSEG0	USB_BA+0x020	R/W	Buffer Segmentation of endpoint 0	0x0000_0000
MXPLD0	USB_BA+0x024	R/W	Maximal payload of endpoint 0	0x0000_0000
CFG0	USB_BA+0x028	R/W	Configuration of endpoint 0	0x0000_0000
CFGP0	USB_BA+0x02C	R/W	stall control register and In/out ready clear flag of endpoint 0	0x0000_0000
BUFSEG1	USB_BA+0x030	R/W	Buffer Segmentation of endpoint 1	0x0000_0000
MXPLD1	USB_BA+0x034	R/W	Maximal payload of endpoint 1	0x0000_0000
CFG1	USB_BA+0x038	R/W	Configuration of endpoint 1	0x0000_0000
CFGP1	USB_BA+0x03C	R/W	stall control register and In/out ready clear flag of endpoint 1	0x0000_0000
BUFSEG2	USB_BA+0x040	R/W	Buffer Segmentation of endpoint 2	0x0000_0000
MXPLD2	USB_BA+0x044	R/W	Maximal payload of endpoint 2	0x0000_0000
CFG2	USB_BA+0x048	R/W	Configuration of endpoint 2	0x0000_0000
CFGP2	USB_BA+0x04C	R/W	stall control register and In/out ready clear flag of endpoint 2	0x0000_0000
BUFSEG3	USB_BA+0x050	R/W	Buffer Segmentation of endpoint 3	0x0000_0000
MXPLD3	USB_BA+0x054	R/W	Maximal payload of endpoint 3	0x0000_0000
CFG3	USB_BA+0x058	R/W	Configuration of endpoint 3	0x0000_0000
CFGP3	USB_BA+0x05C	R/W	stall control register and In/out ready clear flag of endpoint 3	0x0000_0000
BUFSEG4	USB_BA+0x060	R/W	Buffer Segmentation of endpoint 4	0x0000_0000
MXPLD4	USB_BA+0x064	R/W	Maximal payload of endpoint 4	0x0000_0000
CFG4	USB_BA+0x068	R/W	Configuration of endpoint 4	0x0000_0000
CFGP4	USB_BA+0x06C	R/W	stall control register and In/out ready clear flag of endpoint 4	0x0000_0000

BUFSEG5	USB_BA+0x070	R/W	Buffer Segmentation of endpoint 5	0x0000_0000
MXPLD5	USB_BA+0x074	R/W	Maximal payload of endpoint 5	0x0000_0000
CFG5	USB_BA+0x078	R/W	Configuration of endpoint 5	0x0000_0000
CFGP5	USB_BA+0x07C	R/W	In ready clear flag of endpoint 5	0x0000_0000
USBSE0	USB_BA+0x090	R/W	Set D+ and D- bus to idle state	0x0000_0000

Interrupt Enable Register (IEF)

Register	Address	R/W	Description	Reset Value
IEF	USB_BA+0x000	R/W	Interrupt Enable Flag	0x0000_0000

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
INNAKEN	Reserved						WAKEFUEN
7	6	5	4	3	2	1	0
Reserved				WAKEUPEN	FLDEN	USBEN	BUSEN

Bits	Descriptions	
[31:16]	**Reserved**	**Reserved**
[15]	**INNAKEN**	0 = Disable IN NAK INT (**Write Only**) 1 = Enable
[14:9]	**Reserved**	**Reserved**
[8]	**WAKEFUEN**	0 = Disable USB wakeup function 1 = Enable
[7:4]	**Reserved**	**Reserved**
[3]	**WAKEUPEN**	0 = Disable Wakeup Interrupt 1 = Enable
[2]	**FLDEN**	0 = Disable Floating detect Interrupt 1 = Enable
[1]	**USBEN**	0 = Disable USB event interrupt 1 = Enable
[0]	**BUSEN**	0 = Disable BUS event interrupt 1 = Enable

Interrupt Event Flag Register (EVF)

This register is USB Interrupt Event Flag register, clear by read STS, ATTR or FLODET.

Register	Address	R/W	Description	Reset Value
EVF	USB_BA+0x004	R/W	Interrupt Event Flag	0x0000_0000

31	30	29	28	27	26	25	24
Setup	Reserved						
23	22	21	20	19	18	17	16
Reserved		EPTF5	EPTF4	EPTF3	EPTF2	EPTF1	EPTF0
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved				WAKEUP	FLD	USB	BUS

Bits	Descriptions	
[31]	**Setup**	1: Setup event occurred, cleared by read register "STS" or write 1 to EVF[31].
[30:22]	**Reserved**	**Reserved**
[21]	**EPTF5**	1: USB event occurred, check STSX[17:15] to know which kind of USB event was occurred, cleared by read register "STS" or write 1 to EVF[21].
[20]	**EPTF4**	1: USB event occurred, check STSX[14:12] to know which kind of USB event was occurred, cleared by read register "STS" or write 1 to EVF[20].
[19]	**EPTF3**	1: USB event occurred, check STSX[11:9] to know which kind of USB event was occurred, cleared by read register "STS" or write 1 to EVF[19].
[18]	**EPTF2**	1: USB event occurred, check STSX[8:6] to know which kind of USB event was occurred, cleared by read register "STS" or write 1 to EVF[18].
[17]	**EPTF1**	1: USB event occurred, check STSX[5:3] to know which kind of USB event was occurred, cleared by read register "STS" or write 1 to EVF[17].
[16]	**EPTF0**	1: USB event occurred, check STSX[2:0] to know which kind of USB event was occurred, cleared by read register "STS" or write 1 to EVF[16].
[15:4]	**Reserved**	**Reserved**
[3]	**WAKEUP**	Wake up event occurred, cleared by write 1 to EVF[3]

[2]	**FLD**	Floating detect event occurred, cleared by write 1 to EVF[2].
[1]	**USB**	USB event occurred, check STS[6:4] or STS0~5[2:0] to know which kind of USB event was occurred, cleared by write 1 to EVF[1] or EPTF0~5 and Setup = 0.
[0]	**BUS**	Bus event occurred, check ATTR[3:0] to know which kind of bus event was occurred, cleared by write 1 to EVF[0].

Function Address Register (FADDR)

A seven-bit value uses as the address of a device on the USB BUS

Register	Address	R/W	Description	Reset Value
FADDR	USB_BA+0x008	R/W	Function Address	0x0000_0000

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							
7	6	5	4	3	2	1	0
Reserved	FADDR						

Bits	Descriptions	
[31:7]	**Reserved**	**Reserved**
[6:0]	**FADDR**	Function Address of this USB device.

System States Register (STS)

Register	Address	R/W	Description	Reset Value
STS	USB_BA+0x00C	R	System states	0x0000_00x0

31	30	29	28	27	26	25	24
Reserved						STS5	
23	22	21	20	19	18	17	16
STS5	STS4			STS3			STS2
15	14	13	12	11	10	9	8
STS2		STS1			STS0		
7	6	5	4	3	2	1	0
Overrun	STS			EPT			

Bits	Descriptions	
[31:26]	**Reserved**	**Reserved**
[25:23]	**STS5**	System states of endpoint 5 000: In ACK 001: In NAK 010: Out 0 ACK 110: Out 1 ACK 011: Setup ACK 111: Isochronous translation end
[22:20]	**STS4**	System states of endpoint 4 000: In ACK 001: In NAK 010: Out 0 ACK 110: Out 1 ACK 011: Setup ACK 111: Isochronous translation end

[19:17]	**STS3**	System states of endpoint 3 000: In ACK 001: In NAK 010: Out 0 ACK 110: Out 1 ACK 011: Setup ACK 111: Isochronous translation end
[16:14]	**STS2**	System states of endpoint 2 000: In ACK 001: In NAK 010: Out 0 ACK 110: Out 1 ACK 011: Setup ACK 111: Isochronous translation end
[13:11]	**STS1**	System states of endpoint 1 000: In ACK 001: In NAK 010: Out 0 ACK 110: Out 1 ACK 011: Setup ACK 111: Isochronous translation end
[10:8]	**STS0**	System states of endpoint 0 000: In ACK 001: In NAK 010: Out 0 ACK 110: Out 1 ACK 011: Setup ACK 111: Isochronous translation end
[7]	**Overrun**	Out Data more than Max Payload or Setup Data more than 8 Bytes
[6:4]	**STS**	000: In ACK 001: In NAK 010: Out 0 ACK 110: Out 1 ACK 011: Setup ACK 111: Isochronous translation end
[3:0]	**EPT**	Endpoint number

Bus States & Attribution Register (ATTR)

Register	Address	R/W	Description	Reset Value
ATTR	USB_BA+0x010	R/W	Bus states & attribution	0x0000_0040

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							
7	6	5	4	3	2	1	0
enUSB	Reserved	RWakeup	enPHY	Timeout	Resume	Suspend	usbRST

Bits	Descriptions	
[31:8]	**Reserved**	**Reserved**
[7]	**enUSB**	0 = Disable USB 1 = Enable
[6]	**Reserved**	**Reserved**
[5]	**RWakeUp**	0 = Nothing 1 = force USB bus to K state, used for remote wake-up.
[4]	**enPHY**	0 = Disable PHY 1 = Enable
[3]	**Timeout**	No response more than 18 bits time (**Read Only**)
[2]	**Resume**	Resume from suspension (**Read Only**)
[1]	**Suspend**	Bus idle more than 3mS, either cable is plugged off or host is sleeping. (**Read Only**)
[0]	**usbRST**	Bus reset when SE0 (single-ended 0) more than 2.5uS. (**Read Only**)

Floating detection Register (FLODET)

Register	Address	R/W	Description	Reset Value
FLODET	USB_BA+0x014	R	Floating detection	0x0000_0000

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							
7	6	5	4	3	2	1	0
Reserved							FLODET

Bits	Descriptions	
[31:1]	**Reserved**	**Reserved**
[0]	**FLODET**	0 = floating 1 = connected

Buffer Segmentation Register (BUFSEG)

For Setup token only.

Register	Address	R/W	Description	Reset Value
BUFSEG	USB_BA+0x018	R/W	Buffer segmentation	0x0000_0000

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							BUFSEG
7	6	5	4	3	2	1	0
BUFSEG					Reserved		

Bits	Descriptions	
[31:9]	**Reserved**	**Reserved**
[8:3]	**BUFSEG**	For Setup token only. Effective starting address USB buffer = {BUFSEG[8:3], 3'b000}
[2:0]	**Reserved**	**Reserved**

Buffer Segmentation Register (BUFSEGx) x = 0~5

Register	Address	R/W	Description	Reset Value
BUFSEG0	USB_BA+0x020	R/W	Buffer segmentation of endpoint 0	0x0000_0000
BUFSEG1	USB_BA+0x030	R/W	Buffer segmentation of endpoint 1	0x0000_0000
BUFSEG2	USB_BA+0x040	R/W	Buffer segmentation of endpoint 2	0x0000_0000
BUFSEG3	USB_BA+0x050	R/W	Buffer segmentation of endpoint 3	0x0000_0000
BUFSEG4	USB_BA+0x060	R/W	Buffer segmentation of endpoint 4	0x0000_0000
BUFSEG5	USB_BA+0x070	R/W	Buffer segmentation of endpoint 5	0x0000_0000

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							BUFSEG
7	6	5	4	3	2	1	0
BUFSEG					Reserved		

Bits	Descriptions	
[31:9]	**Reserved**	**Reserved**
[8:3]	**BUFSEGx**	Effective starting address USB buffer = {BUFSEGx[8:3], 3'b000}
[2:0]	**Reserved**	**Reserved**

Maximal Payload Register (MXPLDx) x = 0~5

Register	Address	R/W	Description	Reset Value
MXPLD0	USB_BA+0x024	R/W	Maximal Payload of endpoint 0	0x0000_0000
MXPLD1	USB_BA+0x034	R/W	Maximal Payload of endpoint 1	0x0000_0000
MXPLD2	USB_BA+0x044	R/W	Maximal Payload of endpoint 2	0x0000_0000
MXPLD3	USB_BA+0x054	R/W	Maximal Payload of endpoint 3	0x0000_0000
MXPLD4	USB_BA+0x064	R/W	Maximal Payload of endpoint 4	0x0000_0000
MXPLD5	USB_BA+0x074	R/W	Maximal Payload of endpoint 5	0x0000_0000

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							
7	6	5	4	3	2	1	0
MXPLD							

Bits	Descriptions	
[31:8]	**Reserved**	**Reserved**
[7:0]	**MXPLD**	**Read:** IN: Length of Data transmitting to host. Out: Max Length of Data receiving from host. **Write:** IN: Length of Data to transmit to host, assert INrdy (IN buffer ready). OUT: Max Length of Data receiving from host, assert OUTrdy (OUT buffer ready). **Note:** once MXPLD is filled out, the data packets will be transmitted/received immediately after IN/OUT token arrived.

Configuration Register (CFGx) x = 0~5

Register	Address	R/W	Description	Reset Value
CFG0	USB_BA+0x028	R/W	Configuration of Endpoint 0	0x0000_0000
CFG1	USB_BA+0x038	R/W	Configuration of Endpoint 1	0x0000_0000
CFG2	USB_BA+0x048	R/W	Configuration of Endpoint 2	0x0000_0000
CFG3	USB_BA+0x058	R/W	Configuration of Endpoint 3	0x0000_0000
CFG4	USB_BA+0x068	R/W	Configuration of Endpoint 4	0x0000_0000
CFG5	USB_BA+0x078	R/W	Configuration of Endpoint 5	0x0000_0000

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						stall_ctl	Reserved
7	6	5	4	3	2	1	0
DSQ	state		ISOCH	EPT			

Bits	Descriptions	
[31:10]	**Reserved**	**Reserved**
[9]	**stall_ctl**	0 = disable auto clear stall 1 = enable auto clear stall in setup stage
[8]	**Reserved**	**Reserved**
[7]	**DSQ**	Specify Data 0 or 1 after IN token toggle automatically after host ACK.
[6:5]	**state**	00 = endpoint is disabled 01 = Out endpoint 10 = IN endpoint 11 = undefined
[4]	**ISOCH**	Isochronous, no handshake.
[3:0]	**EPT**	Endpoint number.

Extra Configuration Register (CFGPx) x = 0~5

Register	Address	R/W	Description	Reset Value
CFGP0	USB_BA+0x02C	R/W	stall control register and In/out ready clear flag of endpoint 0	0x0000_0000
CFGP 1	USB_BA+0x03C	R/W	stall control register and In/out ready clear flag of endpoint 1	0x0000_0000
CFGP 2	USB_BA+0x04C	R/W	stall control register and In/out ready clear flag of endpoint 2	0x0000_0000
CFGP 3	USB_BA+0x05C	R/W	stall control register and In/out ready clear flag of endpoint 3	0x0000_0000
CFGP 4	USB_BA+0x06C	R/W	stall control register and In/out ready clear flag of endpoint 4	0x0000_0000
CFGP 5	USB_BA+0x07C	R/W	stall control register and In/out ready clear flag of endpoint 5	0x0000_0000

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							
7	6	5	4	3	2	1	0
Reserved						stall	CFGP

Bits	Descriptions	
[31:2]	**Reserved**	**Reserved**
[1]	**stall**	Force device to response STALL (R/W)
[0]	**CFGP**	IN: Write '1' to clear in ready that was set by MXPLD. OUT: Write '1' to clear out ready that was set by MXPLD

6.8

USBSE0

Register	Address	R/W	Description	Reset Value
USBSE0	USB_BA+0x090	R/W	Set D+ and D- to idle state	0x0000_0000

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							
7	6	5	4	3	2	1	0
Reserved							SE0

Bits	Descriptions	
[31:1]	**Reserved**	**Reserved**
[0]	**SE0**	0: Normal state 1: SE0 state In SE0 state, the USB D+ and D- will be drive low and cause host doesn't see the device even the USB cable is still connected. Therefore SE0 could be used to force host to re-connect the USB device.

6.10 Advanced Interrupt Controller

6.10.1 Overview

An interrupt temporarily changes the execution sequence of a program to react to a particular event such as power failure, watchdog timer timeout, and engine complete, system events, external event trigger and so on. The ARM processor provides two modes of interrupts, the **Fast Interrupt (FIQ)** mode for critical session and the Interrupt **(IRQ)** mode for general purpose. The IRQ exception mode is occurred when the NIRQ input is asserted. Similarly, the FIQ exception mode is occurred when the NFIQ input is asserted. The FIQ mode has privilege over the IRQ mode and can preempt an ongoing IRQ mode. It is possible to ignore the NFIQ and the NIRQ by setting the F-bit and I-bit in the **current program status register (CPSR)**.

The NUC501 incorporates the **advanced interrupt controller (AIC)** that is capable of dealing with the interrupt requests from different sources. Each interrupt source is uniquely assigned to an *interrupt channel*. For example, the watchdog timer interrupt is assigned to channel 2. The AIC implements a proprietary eight-level priority scheme that differentiates the available interrupt sources into eight priority levels. Interrupt sources within the priority level 0 have the highest priority and the priority level 7 has the lowest. To work this scheme properly, you must specify a certain priority level to each interrupt source during power-on initialization; otherwise, the system shall behave unexpectedly. Within each priority level, interrupt source that is positioned in a lower channel has a higher priority. Interrupt source that is active, enabled, and positioned in the lowest channel within the priority level 0 is promoted to the FIQ mode. Interrupt sources within the priority levels other than 0 can petition for the IRQ mode. The IRQ mode can be preempted by the occurrence of the FIQ mode. Interrupt nesting is performed automatically by the AIC. A higher priority interrupt source will cause the NIRQ to CPU be asserted again when CPU is servicing a lower priority interrupt if the I-bit in CPSR is enabled.

Though interrupt sources originated from the NUC501 itself are intrinsically high-level sensitive, the AIC can be configured as either low-level sensitive, high-level sensitive, negative-edge triggered, or positive-edge triggered to each interrupt source.

6.10.2 Features

- AMBA APB bus interface and Individual mask for each interrupt source
- External interrupts can be programmed as either edge-triggered or level-sensitive
- External interrupts can be programmed as either low-active or high-active
- Has flags to reflect the status of each interrupt source
- Proprietary 8-level interrupt scheme to ease the burden from the interrupt
- Daisy-chain priority mechanism is applied to interrupts set as the same priority level.
- Automatically masking out the lower priority interrupt during interrupt nesting
- Automatically clearing the interrupt flag when the external interrupt source is programmed to be edge-triggered

6.10.3 Interrupt Sources

The following table lists all interrupts from various peripheral interface modules or external devices.

Channel	Name	SCR	Source	Reset (default) level
1	WDT_INT	SCR1[15:8]	Watch Dog Timer Interrupt	Low
2	**Reserved**	**Reserved**	**Reserved**	Low
3	INT_GPIO0	SCR1[31:24]	GPIO Interrupt0	Low
4	INT_GPIO1	SCR2[7:0]	GPIO Interrupt1	Low
5	INT_GPIO2	SCR2[15:8]	GPIO Interrupt2	Low
6	INT_GPIO3	SCR2[23:16]	GPIO Interrupt3	Low
7	INT_APU	SCR2[31:24]	Audio Processing Unit Interrupt	Low
8	**Reserved**	**Reserved**	**Reserved**	Low
9	**Reserved**	**Reserved**	**Reserved**	Low
10	INT_ADC	SCR3[23:16]	AD Converter Interrupt	Low
11	INT_RTC	SCR3[31:24]	RTC Interrupt	Low
12	INT_UART0	SCR4[7:0]	UART-0 Interrupt	Low
13	INT_UART1	SCR4[15:8]	UART-1 Interrupt	Low
14	INT_TMR1	SCR4[23:16]	Timer-1 Interrupt	Low
15	INT_TMR0	SCR4[31:24]	Timer-0 Interrupt	Low
16	**Reserved**	**Reserved**	**Reserved**	Low
17	**Reserved**	**Reserved**	**Reserved**	Low
18	**Reserved**	**Reserved**	**Reserved**	Low
19	INT_USB	SCR5[31:24]	USB Device Interrupt(Notes)	Low
20	**Reserved**	**Reserved**	**Reserved**	Low
21	**Reserved**	**Reserved**	**Reserved**	Low
22	INT_PWM0	SCR6[23:16]	PWM Interrupt0	Low
23	INT_PWM1	SCR6[31:24]	PWM Interrupt1	Low

24	INT_PWM2	SCR7[7:0]	PWM Interrupt2	Low
25	INT_PWM3	SCR7[15:8]	PWM Interrupt3	Low
26	INT_I2C	SCR7[23:16]	I2C Interface Interrupt	Low
27	INT_SPIMS	SCR7[31:24]	SPI (Master/Slave) Serial Interface Interrupt	Low
28	**Reserved**	**Reserved**	**Reserved**	Low
29	INT_PWR	SCR8[15:8]	System Wake-Up Interrupt	Low
30	INT_SPIM	SCR8[23:16]	SPIM0/1 Interrupt	Low
31	**Reserved**	**Reserved**	**Reserved**	Low

31	30	29	28	27	26	25	24
---	**SPIM**	**PWR**	**---**	**SPIMS**	**I2C**	**PWM3**	**PWM2**
23	22	21	20	19	18	17	16
PWM1	**PWM0**	**---**	**---**	**USB**	**---**	**---**	**---**
15	14	13	12	11	10	9	8
TMR0	**TMR1**	**UART1**	**UART0**	**RTC**	**ADC**	**---**	**---**
7	6	5	4	3	2	1	0
APU	**GPIO3**	**GPIO2**	**GPIO1**	**GPIO0**	**---**	**WDT**	**---**

6.10.4 AIC Functional Descriptions

Hardware Interrupt Vectoring

The hardware interrupt vectoring can be used to shorten the interrupt latency. If not used, priority determination must be carried out by software. When the Interrupt Priority Encoding Register (AIC_IPER) is read, it will return an integer representing the channel that is active and having the highest priority. This integer is equivalent to multiplied by 4 (shifted left two bits to word-align it) such that it may be used directly to index into a branch table to select the appropriate interrupt service routine vector.

Priority Controller

An 8-level priority encoder controls the NIRQ and NFIQ line. Each interrupt source belongs to priority group between of 0 to 7. Group 0 has the highest priority and group 7 the lowest. Group 0 means FIQ mode group. When more than one unmasked interrupt channels are active at a time, the interrupt with the highest priority is serviced first. If all active interrupts have equal priority, the interrupt with the lowest interrupt source number is serviced first.

The current priority level is defined as the priority level of the interrupt with the highest priority at the time the register AIC_IPER is read. In the case when a higher priority unmasked interrupt occurs while an interrupt already exits, there are two possible outcomes depending on whether the AIC_IPER has been read.

- If the processor has already read the AIC_IPER and caused the NIRQ line to be de-asserted, then the NIRQ line is reasserted. When the processor has enabled nested interrupts and reads the AIC_IPER again, it reads the new, higher priority interrupt vector. At the same time, the current priority level is updated to the higher priority.
- If the AIC_IPER has not been read after the NIRQ line has been asserted, then the processor will read the new higher priority interrupt vector in the AIC_IPER register and the current priority level is updated.

When the End of Service Command Register (AIC_EOSCR) is written, the current interrupt level is updated with the last stored interrupt level from the stack (if any). Therefore, at the end of a higher priority interrupt, the AIC returns to the previous state corresponding to the preceding lower priority interrupt which had been interrupted.

Interrupt Handling

When the NIRQ line is asserted, the interrupt handler must read the AIC_IPER as soon as possible. This can de-assert the NIRQ request to the processor and clears the interrupt if it is programmed to be edge triggered. This allows the AIC to assert the NIRQ line again when a higher priority unmasked interrupt occurs.

The AIC_EOSCR (End of Service Command Register) must be written at the end of the interrupt service routine. This permits pending interrupts to be serviced.

Interrupt Masking

Each interrupt source can be enabled or disabled individually by using the command registers AIC_MECR and AIC_MDCR. The status of interrupt mask can be read in the read only register AIC_IMR. A disabled interrupt doesn't affect the servicing of other interrupts.

Interrupt Clearing and Setting

All interrupt sources can be individually set or clear by respectively writing to the registers AIC_SSCR and AIC_SCCR when they are programmed to be edge triggered. This feature of the AIC is useful in auto-testing or software debugging.

Fake Interrupt

When the AIC asserts the NIRQ line, the processor enters interrupt mode and the interrupt handler reads the AIC_IPER, it may happen that interrupt sources de-assert NIRQ lines after the processor has taken into account the NIRQ assertion and before the read of the AIC_IPER.
This behavior is called a fake interrupt.

The AIC is able to detect these fake interrupts and returns all zero when AIC_IPER is read. The same mechanism of fake interrupt occurs if the processor reads the AIC_IPER (application software or ICE) when no interrupt pending. The current priority level is not updated in this situation. Hence, the AIC_EOSCR shouldn't be written.

ICE/Debug Mode

This mode allows reading of the AIC_IPER without performing the associated automatic operations. This is necessary when working with a debug system. When an ICE or debug monitor reads the AIC user interface, the AIC_IPER can be read. This has the following consequences in normal mode:

- If there is no enabled pending interrupt, the fake vector will be returned.
- If an enabled interrupt with a higher priority than the current one is pending, it will be stacked.

In the second case, an End-of-Service command would be necessary to restore the state of the AIC. This operation is generally not performed by the debug system. Therefore, the debug system would become strongly intrusive, and could cause the application to enter an undesired state.

This can be avoided by using ICE/Debug Mode. When this mode is enabled, the AIC performs interrupt stacking only when a write access is performed on the AIC_IPER. Hence, the interrupt service routine must write to the AIC_IPER (any value) just after reading it. When AIC_IPER is written, the new status of AIC, including the value of interrupt source number register (AIC_ISNR), is updated with the value that is kept at previous reading of AIC_IPER, the debug system must not write to the AIC_IPER as this would cause undesirable effects.

The following table shows the main steps of an interrupt and the order in which they are performed according to the mode:

Action	Normal Mode	ICE/Debug Mode
Calculate active interrupt	Read AIC_IPER	Read AIC_IPER
Determine and return the vector of the active interrupt	Read AIC_IPER	Read AIC_IPER
Push on internal stack the current priority level	Read AIC_IPER	Write AIC_IPER
Acknowledge the interrupt (Note 1)	Read AIC_IPER	Write AIC_IPER
No effect (Note 2)	Read AIC_IPER	

Notes:

- NIRQ de-assertion and automatic interrupt clearing if the source is programmed as level sensitive.
- Note that software which has been written and debugged using this mode will run correctly in normal mode without modification. However, in normal mode writing to AIC_IPER has no effect and can be removed to optimize the code.

6.10.5 AIC Registers Mapping

Register	Address	R/W	Description	Reset Value
AIC_BA = 0xB800_2000				
AIC_SCR1	AIC_BA+000	R/W	Source Control Register 1	0x4747_4747
AIC_SCR2	AIC_BA+004	R/W	Source Control Register 2	0x4747_4747
AIC_SCR3	AIC_BA+008	R/W	Source Control Register 3	0x4747_4747
AIC_SCR4	AIC_BA+00C	R/W	Source Control Register 4	0x4747_4747
AIC_SCR5	AIC_BA+010	R/W	Source Control Register 5	0x4747_4747
AIC_SCR6	AIC_BA+014	R/W	Source Control Register 6	0x4747_4747
AIC_SCR7	AIC_BA+018	R/W	Source Control Register 7	0x4747_4747
AIC_SCR8	AIC_BA+01C	R/W	Source Control Register 8	0x4747_4747
AIC_IRSR	AIC_BA+100	R	Interrupt Raw Status Register	0x0000_0000
AIC_IASR	AIC_BA+104	R	Interrupt Active Status Register	0x0000_0000
AIC_ISR	AIC_BA+108	R	Interrupt Status Register	0x0000_0000
AIC_IPER	AIC_BA+10C	R	Interrupt Priority Encoding Register	0x0000_0000
AIC_ISNR	AIC_BA+110	R	Interrupt Source Number Register	0x0000_0000
AIC_IMR	AIC_BA+114	R	Interrupt Mask Register	0x0000_0000
AIC_OISR	AIC_BA+118	R	Output Interrupt Status Register	0x0000_0000

Reserved	**Reserved**		**Reserved**	Undefined
AIC_MECR	AIC_BA+120	W	Mask Enable Command Register	Undefined
AIC_MDCR	AIC_BA+124	W	Mask Disable Command Register	Undefined
AIC_SSCR	AIC_BA+128	W	Source Set Command Register	Undefined
AIC_SCCR	AIC_BA+12C	W	Source Clear Command Register	Undefined
AIC_EOSCR	AIC_BA+130	W	End of Service Command Register	Undefined
AIC_TEST	AIC_BA+134	W/R	ICE/Debug mode Register	0x0000_0000

6.10.6 AIC Control Registers

AIC Source Control Registers (AIC_SCR1 ~ AIC_SCR8)

Register	Address	R/W/C	Description	Reset Value
AIC_SCR1 ~ AIC_SCR8	AIC_BA+000 ~ AIC_BA+01C	R/W	Source Control Register 1 ~ Source Control Register 8	0x4747_4747

31	30	29	28	27	26	25	24
TYPE (Channel 3)		**Reserved**			**PRIORITY (Channel 3)**		
23	22	21	20	19	18	17	16
TYPE (Channel 2)		**Reserved**			**PRIORITY (Channel 2)**		
15	14	13	12	11	10	9	8
TYPE (Channel 1)		**Reserved**			**PRIORITY (Channel 1)**		
7	6	5	4	3	2	1	0
TYPE (channel 0)		**Reserved**			**PRIORITY (Channel 0)**		

Bits	Descriptions	
[31:30] [23:22] [15:14] [7:6]	**TYPE**	**Interrupt Type** **[7] Indicates the level (0)/edge (1) triggered.** **[6] Indicates the low (0)/high (1) level.** • 00: low-active level triggered • 01: high-active level triggered • 10: low-active edge triggered • 11: high-active edge triggered Interrupts other than INT_EXT can be configured as level triggered during normal operation unless in the test mode.
[29:27] [21:19] [13:11] [5:3]	**Reserved**	**Reserved**

[26:24] [18:16] [11:8] [2:0]	**PRIORITY**	**Priority Level (0 – 7)** • The level 0 indicates the highest priority and the level 7 indicates the lowest priority. • An interrupt is treated as a FIQ mode for the priority level 0, and is treated as an IRQ mode for other levels. • If two or more interrupts have the identical priority level, the interrupts located in the upper rows of the interrupt source table, have higher priorities.

AIC Interrupt Raw Status Register (AIC_IRSR)

Register	Address	R/W	Description	Reset Value
AIC_IRSR	AIC_BA+100	R	Interrupt Raw Status Register	0x0000_0000

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

This register records the intrinsic state within each interrupt channel.

Bits	Descriptions	
[31:0]	**IRS*x***	**Interrupt Status** Indicate the intrinsic status of the corresponding interrupt source • 0 = Interrupt channel is in the voltage level 0 • 1 = Interrupt channel is in the voltage level 1

AIC Interrupt Active Status Register (AIC_IASR)

Register	Address	R/W	Description	Reset Value
AIC_IASR	AIC_BA+104	R	Interrupt Active Status Register	0x0000_0000

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

This register indicates the status of each interrupt channel in consideration of the interrupt source type as defined in the corresponding Source Control Register, but regardless of its mask setting.

Bits	Descriptions	
[31:0]	**IAS*x***	**Interrupt Active Status** Indicate the status of the corresponding interrupt source • 0 = Corresponding interrupt channel is inactive • 1 = Corresponding interrupt channel is active

AIC Interrupt Status Register (AIC_ISR)

Register	Address	R/W	Description	Reset Value
AIC_ISR	AIC_BA+108	R	Interrupt Status Register	0x0000_0000

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

This register identifies those interrupt channels whose are both active and enabled.

Bits	Descriptions	
[31:0]	**ISR*x*:**	**Interrupt Status Register** Indicates the status of corresponding interrupt channel • 0 = Two possibilities: (a)The corresponding interrupt channel is inactive no matter whether it is enabled or disabled (b)It is active but not enabled • 1 = Corresponding interrupt channel is both active and enabled (can assert an interrupt)

AIC IRQ Priority Encoding Register (AIC_IPER)

Register	Address	R/W	Description	Reset Value
AIC_IPER	AIC_BA+10C	R	Interrupt Priority Encoding Register	0x0000_0000

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							
7	6	5	4	3	2	1	0
Reserved	**VECTOR**					**Reserved**	

When the AIC generates the interrupt, **VECTOR** represents the interrupt channel number that is active, enabled, and has the highest priority. If the representing interrupt channel possesses a priority level 0, then the interrupt asserted is FIQ mode; otherwise, it is IRQ mode. The value of **VECTOR** is copied to the register AIC_ISNR thereafter by the AIC. This register is restored a value 0 after it was read by the interrupt handler. This register can help indexing into a branch table to quickly jump to the corresponding interrupt service routine. The reserved bits are set to zero.

Bits	Descriptions	
[31:7]	**Reserved**	**Reserved**
[6:2]	**VECTOR**	**Interrupt Vector** • 0 = no interrupt occurs • 1 ~ 31 = representing the interrupt channel that is active, enabled, and having the highest priority
[1:0]	**Reserved**	**Reserved**

AIC Interrupt Source Number Register (AIC_ISNR)

Register	Address	R/W	Description	Reset Value
AIC_ISNR	AIC_BA+110	R	Interrupt Source Number Register	0x0000_0000

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							
7	6	5	4	3	2	1	0
Reserved			**IRQID**				

The purpose of this register is to record the interrupt channel number that is active, enabled, and has the highest priority. The reserved bits are set to zero.

Bits	Descriptions	
[31:5]	**Reserved**	**Reserved**
[4:0]	**IRQID**	**IRQ Identification** Stands for the interrupt channel number

AIC Interrupt Mask Register (AIC_IMR)

Register	Address	R/W	Description	Reset Value
AIC_IMR	AIC_BA+114	R	Interrupt Mask Register	0x0000_0000

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

Bits	Descriptions	
[31:0]	**IM*x***	**Interrupt Mask** This bit determines whether the corresponding interrupt channel is enabled or disabled. Every interrupt channel can be active no matter whether it is enabled or disabled. If an interrupt channel is enabled, it does not definitely mean it is active. Every interrupt channel can be authorized by the AIC only when it is both active and enabled. • 0 = Corresponding interrupt channel is disabled • 1 = Corresponding interrupt channel is enabled

AIC Output Interrupt Status Register (AIC_OISR)

Register	Address	R/W	Description	Reset Value
AIC_OISR	AIC_BA+118	R	Output Interrupt Status Register	0x0000_0000

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							
7	6	5	4	3	2	1	0
Reserved						**IRQ**	**FIQ**

The AIC classifies the interrupt into FIQ mode and IRQ mode. This register indicates whether the asserted interrupt is NFIQ or NIRQ. If both NIRQ and NFIQ are equal to 0, it means there is no interrupt occurred.

Bits	Descriptions	
[31:2]	**Reserved**	**Reserved**
[1]	**IRQ**	**Interrupt Request** • 0 = NIRQ line is inactive. • 1 = NIRQ line is active.
[0]	**FIQ**	**Fast Interrupt Request** • 0 = NFIQ line is inactive. • 1 = NFIQ line is active

AIC Mask Enable Command Register (AIC_MECR)

Register	Address	R/W	Description	Reset Value
AIC_MECR	AIC_BA+120	W	Mask Enable Command Register	Undefined

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

Bits	Descriptions	
[31:0]	**MEC*x***	**Mask Enable Command** • 0 = No effect • 1 = Enables the corresponding interrupt channel

AIC Mask Disable Command Register (AIC_MDCR)

Register	Address	R/W	Description	Reset Value
AIC_MDCR	AIC_BA+124	W	Mask Disable Command Register	Undefined

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

Bits	Descriptions	
[31:0]	**MDC*x***	**Mask Disable Command** • 0 = No effect • 1 = Disables the corresponding interrupt channel

AIC Source Set Command Register (AIC_SSCR)

Register	Address	R/W	Description	Reset Value
AIC_SSCR	AIC_BA+128	W	Source Set Command Register	Undefined

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

When the NUC501 is under debugging or verification, software can activate any interrupt channel by setting the corresponding bit in this register. This feature is useful in hardware verification or software debugging.

Bits	Descriptions	
[31:0]	**SSC*x***	**Source Set Command** • 0 = No effect. • 1 = Activates the corresponding interrupt channel

AIC Source Clear Command Register (AIC_SCCR)

Register	Address	R/W	Description	Reset Value
AIC_SCCR	AIC_BA+12C	W	Source Clear Command Register	Undefined

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

When the NUC501 is under debugging or verification, software can deactivate any interrupt channel by setting the corresponding bit in this register. This feature is useful in hardware verification or software debugging.

Bits	Descriptions	
[31:0]	**SCC*x***	**Source Clear Command** • 0 = No effect. • 1 = Deactivates the corresponding interrupt channels

AIC End of Service Command Register (AIC_EOSCR)

Register	Address	R/W	Description	Reset Value
AIC_EOSCR	AIC_BA+130	W	End of Service Command Register	Undefined

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
---	---	---	---	---	---	---	---

This register is used by the interrupt service routine to indicate that it is completely served. Thus, the interrupt handler can write any value to this register to indicate the end of its interrupt service.

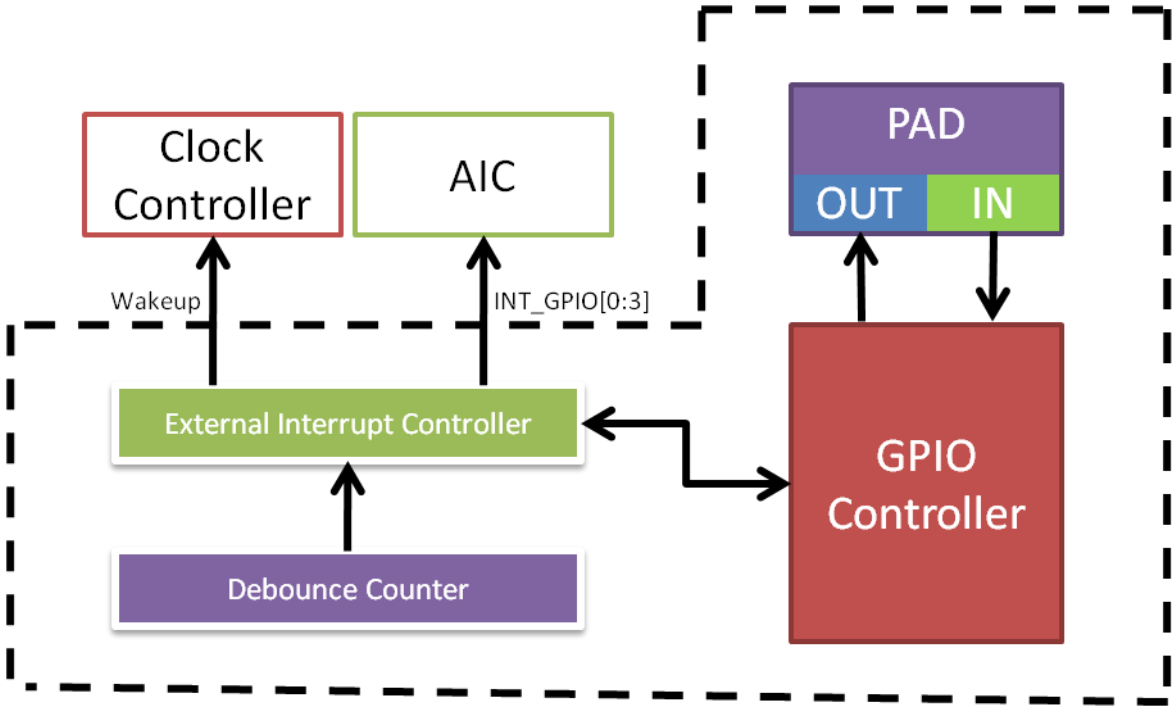
Bits	Descriptions	
[31:0]	---	---

6.11 General Purpose I/O

6.11.1 Overview and Features

26 pins for 48-pins package and 37 pins for 64-pins package and COB of General Purpose I/O are shared with special feature functions.

Supported Features of these I/O are: input or output facilities, pull-up resistors.
All these general purpose I/O functions are achieved by software programming setting. And the following figures illustrate the control mechanism to achieve the GPIO functions.



6.11.2 GPIO Control Register Mapping

R: read only, **W:** write only, **R/W:** both read and write

Register	Address	R/W	Description	Reset Value
GP_BA = 0xB800_3000				
GPIOA_OMD	GP_BA+0x00	R/W	GPIO Port A Bit Output Mode Enable	0x0000_0000
GPIOA_PUEN	GP_BA+0x04	R/W	GPIO Port A Bit Pull-up Resistor Enable	0x0000_0000
GPIOA_DOUT	GP_BA+0x08	R/W	GPIO Port A Data Output Value	0x0000_0000
GPIOA_PIN	GP_BA+0x0C	R	GPIO Port A Pin Value	0xXXXX_XXXX
GPIOB_OMD	GP_BA+0x10	R/W	GPIO Port B Bit Output Mode Enable	0x0000_0000
GPIOB_PUEN	GP_BA+0x14	R/W	GPIO Port B Bit Pull-up Resistor Enable	0x0000_0000
GPIOB_DOUT	GP_BA+0x18	R/W	GPIO Port B Data Output Value	0x0000_0000
GPIOB_PIN	GP_BA+0x1C	R	GPIO Port B Pin Value	0xXXXX_XXXX
GPIOC_OMD	GP_BA+0x20	R/W	GPIO Port C Bit Output Mode Enable	0x0000_0000
GPIOC_PUEN	GP_BA+0x24	R/W	GPIO Port C Bit Pull-up Resistor Enable	0x0000_0000
GPIOC_DOUT	GP_BA+0x28	R/W	GPIO Port C Data Output Value	0x0000_0000
GPIOC_PIN	GP_BA+0x2C	R	GPIO Port C Pin Value	0xXXXX_XXXX
DBNCECON	GP_BA+0x70	R/W	External Interrupt De-bounce Control	0x0000_0000
IRQSRCGPA	GP_BA+0x80	R/W	GPIO Port A IRQ Source Grouping	0x0000_0000
IRQSRCGPB	GP_BA+0x84	R/W	GPIO Port B IRQ Source Grouping	0x5555_5555
IRQSRCGPC	GP_BA+0x88	R/W	GPIO Port C IRQ Source Grouping	0xAAAA_AAAA
IRQENGPA	GP_BA+0x90	R/W	GPIO Port A Interrupt Enable	0x0000_0000
IRQENGPB	GP_BA+0x94	R/W	GPIO Port B Interrupt Enable	0x0000_0000
IRQENGPC	GP_BA+0x98	R/W	GPIO Port C Interrupt Enable	0x0000_0000
IRQLHSEL	GP_BA+0xA0	R/W	Interrupt Latch Trigger Selection Register	0x0000_0000
IRQLHGPA	GP_BA+0xA4	R	GPIO Port A Interrupt Latch Value	0x0000_0000
IRQLHGPB	GP_BA+0xA8	R	GPIO Port B Interrupt Latch Value	0x0000_0000
IRQLHGPC	GP_BA+0xAC	R	GPIO Port C Interrupt Latch Value	0x0000_0000
IRQTGSRC0	GP_BA+0xB4	R/C	IRQ0~3 Interrupt Trigger Source Indicator from GPIO Port A and GPIO Port B	0x0000_0000
IRQTGSRC1	GP_BA+0xB8	R/C	IRQ0~3 Interrupt Trigger Source Indicator from GPIO Port C	0x0000_0000

6.11.3 GPIO Control Register Description

GPIO Port [A] Bit Output Mode Enable (GPIOA_OMD)

Register	Address	R/W	Description	Reset Value
GPIOA_OMD	GP_BA+0x00	R/W	GPIO Port A Bit Output Mode Enable	0x0000_0000

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
OMD15	**OMD14**	**OMD13**	**OMD12**	**OMD11**	**OMD10**	**OMD9**	**OMD8**
7	6	5	4	3	2	1	0
OMD7	**OMD6**	**OMD5**	**OMD4**	**OMD3**	**OMD2**	**OMD1**	**OMD0**

GPIO Port [B] Bit Output Mode Enable (GPIOB_OMD)

Register	Address	R/W	Description	Reset Value
GPIOB_OMD	GP_BA+0x10	R/W	GPIO Port B Bit Output Mode Enable	0x0000_0000

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						**OMD9**	**OMD8**
7	6	5	4	3	2	1	0
OMD7	**OMD6**	**OMD5**	**OMD4**	**OMD3**	**OMD2**	**OMD1**	**OMD0**

GPIO Port [C] Bit Output Mode Enable (GPIOC_OMD)

Register	Address	R/W	Description	Reset Value

GPIOC_OMD	GP_BA+0x20	R/W	GPIO Port C Bit Output Mode Enable	0x0000_0000

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					**OMD10**	**OMD9**	**OMD8**
7	6	5	4	3	2	1	0
OMD7	**OMD6**	**OMD5**	**OMD4**	**OMD3**	**OMD2**	**OMD1**	**OMD0**

Bits	Descriptions	
[n]	**OMDn**	**Bit Output Mode Enable** 1 = GPIO port [A/B/C] bit [n] output mode is enabled, the bit value contained in the corresponding bit [n] of GPIO[A/B/C]_DOUT is driven on the pin. 0 = GPIO port [A/B/C] bit [n] output mode is disabled, the corresponding pin is in INPUT mode.

GPIO Port [A] Bit Pull-up Resistor Enable (GPIOA_PUEN)

Register	Address	R/W	Description	Reset Value
GPIOA_PUEN	GP_BA+0x04	R/W	GPIO Port A Bit Pull-up Resistor Enable	0x0000_0000

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
PUEN15	**PUEN14**	**PUEN13**	**PUEN12**	**PUEN11**	**PUEN10**	**PUEN9**	**PUEN8**
7	6	5	4	3	2	1	0
PUEN7	**PUEN6**	**PUEN5**	**PUEN4**	**PUEN3**	**PUEN2**	**PUEN1**	**PUEN0**

GPIO Port [B] Bit Pull-up Resistor Enable (GPIOB_PUEN)

Register	Address	R/W	Description	Reset Value
GPIOB_PUEN	GP_BA+0x14	R/W	GPIO Port B Bit Pull-up Resistor Enable	0x0000_0000

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						**PUEN9**	**PUEN8**
7	6	5	4	3	2	1	0
PUEN7	**PUEN6**	**PUEN5**	**PUEN4**	**PUEN3**	**PUEN2**	**PUEN1**	**PUEN0**

GPIO Port [C] Bit Pull-up Resistor Enable (GPIOC_PUEN)

Register	Address	R/W	Description	Reset Value
GPIOC_PUEN	GP_BA+0x24	R/W	GPIO Port C Bit Pull-up Resistor Enable	0x0000_0000

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					**PUEN10**	**PUEN9**	**PUEN8**
7	6	5	4	3	2	1	0
PUEN7	**PUEN6**	**PUEN5**	**PUEN4**	**PUEN3**	**PUEN2**	**PUEN1**	**PUEN0**

Bits	Descriptions	
[n]	**PUENn**	**PUEN[n]: Bit Pull-up Resistor Enable** 1 = GPIO port [A/B/C] bit [n] pull-up resistor is enabled. 0 = GPIO port [A/B/C] bit [n] pull-up resistor is disabled.

GPIO Port [A] Data Output Value (GPIOA_DOUT)

Register	Address	R/W	Description	Reset Value
GPIOA_DOUT	GP_BA+0x08	R/W	GPIO Port A Data Output Value	0x0000_0000

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
DOUT15	**DOUT14**	**DOUT13**	**DOUT12**	**DOUT11**	**DOUT10**	**DOUT9**	**DOUT8**
7	6	5	4	3	2	1	0
DOUT7	**DOUT6**	**DOUT5**	**DOUT4**	**DOUT3**	**DOUT2**	**DOUT1**	**DOUT0**

GPIO Port [B] Data Output Value (GPIOB_DOUT)

Register	Address	R/W	Description	Reset Value
GPIOB_DOUT	GP_BA+0x18	R/W	GPIO Port B Data Output Value	0x0000_0000

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						**DOUT9**	**DOUT8**
7	6	5	4	3	2	1	0
DOUT7	**DOUT6**	**DOUT5**	**DOUT4**	**DOUT3**	**DOUT2**	**DOUT1**	**DOUT0**

GPIO Port [C] Data Output Value (GPIOC_DOUT)

Register	Address	R/W	Description	Reset Value
GPIOC_DOUT	GP_BA+0x28	R/W	GPIO Port C Data Output Value	0x0000_0000

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					**DOUT10**	**DOUT9**	**DOUT8**
7	6	5	4	3	2	1	0
DOUT7	**DOUT6**	**DOUT5**	**DOUT4**	**DOUT3**	**DOUT2**	**DOUT1**	**DOUT0**

Bits	Descriptions	
[n]	**DOUTn**	**Bit Output Value** 1 = GPIO port [A/B/C] bit [n] will drive High if the corresponding output mode enabling bit is set. 0 = GPIO port [A/B/C] bit [n] will drive Low if the corresponding output mode enabling bit is set.

GPIO Port [A] Pin Value (GPIOA _PIN)

Register	Address	R/W	Description	Reset Value
GPIOA_PIN	GP_BA+0x0C	R	GPIO Port A Pin Value	0x0000_XXXX

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
PIN[15:8]							
7	6	5	4	3	2	1	0
PIN[7:0]							

GPIO Port [B] Pin Value (GPIOB_PIN)

Register	Address	R/W	Description	Reset Value
GPIOB_PIN	GP_BA+0x1C	R	GPIO Port B Pin Value	0x0000_0XXX

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						**PIN[9:8]**	
7	6	5	4	3	2	1	0
PIN[7:0]							

GPIO Port [C] Pin Value (GPIOC_PIN)

Register	Address	R/W	Description	Reset Value
GPIOC_PIN	GP_BA+0x2C	R	GPIO Port C Pin Value	0x0000_0XXX

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					**PIN[10:8]**		
7	6	5	4	3	2	1	0
PIN[7:0]							

Bits	Descriptions	
[n]	**PIN**	**Port [A/B/C] Pin Values**

Interrupt Debounce Control (DBNCECON)

Register	Address	R/W	Description	Reset Value
DBNCECON	GP_BA+0x70	R/W	External Interrupt De-bounce Control	0x0000_0000

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							
7	6	5	4	3	2	1	0
DBCLKSEL				**DBEN**			

Bits	Descriptions	
[31:8]	**Reserved**	**Reserved**
[7:4]	**DBCLKSEL**	**Debounce sampling cycle selection** DBCLKSEL / Description 0 Sample interrupt input once per 1 APB clocks 1 Sample interrupt input once per 2 APB clocks 2 Sample interrupt input once per 4 APB clocks 3 Sample interrupt input once per 8 APB clocks 4 Sample interrupt input once per 16 APB clocks 5 Sample interrupt input once per 32 APB clocks 6 Sample interrupt input once per 64 APB clocks 7 Sample interrupt input once per 128 APB clocks 8 Sample interrupt input once per 256 APB clocks 9 Sample interrupt input once per 2*256 APB clocks 10 Sample interrupt input once per 4*256 APB clocks 11 Sample interrupt input once per 8*256 APB clocks 12 Sample interrupt input once per 16*256 APB clocks 13 Sample interrupt input once per 32*256 APB clocks 14 Sample interrupt input once per 64*256 APB clocks 15 Sample interrupt input once per 128*256 APB clocks
[3:0]	**DBEN**	**DBEN[x]: debounce sampling enable for each IRQx, x = 0 ~ 3** 1 = Interrupt input IRQx is filtered with de-bounce sampling 0 = Interrupt input IRQx is input directly without de-bounce sampling

IRQ Source Grouping (IRQSRCGPA)

Register	Address	R/W	Description	Reset Value
IRQSRCGPA	GP_BA+0x80	R/W	GPIO Port A IRQ Source Grouping	0x0000_0000

31	30	29	28	27	26	25	24
GPA15SEL		**GPA14SEL**		**GPA13SEL**		**GPA12SEL**	
23	22	21	20	19	18	17	16
GPA11SEL		**GPA10SEL**		**GPA9SEL**		**GPA8SEL**	
15	14	13	12	11	10	9	8
GPA7SEL		**GPA6SEL**		**GPA5SEL**		**GPA4SEL**	
7	6	5	4	3	2	1	0
GPA3SEL		**GPA2SEL**		**GPA1SEL**		**GPA0SEL**	

Bits	Descriptions	
[2x+1:2x]	**GPAxSEL**	Selection for GPAx as one of input Pins to IRQ0, IRQ1, IRQ2, or IRQ3 interrupt source

Where x=0~15.

GPAxSEL = 0, GPAx pin is grouped as one of interrupt sources to IRQ0.
1, GPAx pin is grouped as one of interrupt sources to IRQ1.
2, GPAx pin is grouped as one of interrupt sources to IRQ2.
3, GPAx pin is grouped as one of interrupt sources to IRQ3.

IRQ Source Grouping (IRQSRCGPB)

Register	Address	R/W	Description	Reset Value
IRQSRCGPB	GP_BA+0x84	R/W	GPIO Port B IRQ Source Grouping	0x0005_5555

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved				**GPB9SEL**		**GPB8SEL**	
15	14	13	12	11	10	9	8
GPB7SEL		**GPB6SEL**		**GPB5SEL**		**GPB4SEL**	
7	6	5	4	3	2	1	0
GPB3SEL		**GPB2SEL**		**GPB1SEL**		**GPB0SEL**	

Bits	Descriptions		Default
[2x+1:2x]	**GPBxSEL**	Selection for GPBx as one of input Pins to IRQ0, IRQ1, IRQ2, or IRQ3 interrupt source	0x1

Where x=0~15.

GPAxSEL = 0, GPBx pin is grouped as one of interrupt sources to IRQ0.
1, GPBx pin is grouped as one of interrupt sources to IRQ1.
2, GPBx pin is grouped as one of interrupt sources to IRQ2.
3, GPBx pin is grouped as one of interrupt sources to IRQ3.

IRQ Source Grouping (IRQSRCGPC)

Register	Address	R/W	Description	Reset Value
IRQSRCGPC	GP_BA+0x88	R/W	GPIO Port C IRQ Source Grouping	0x002A_AAAA

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved		**GPC10SEL**		**GPC9SEL**		**GPC8SEL**	
15	14	13	12	11	10	9	8
GPC7SEL		**GPC6SEL**		**GPC5SEL**		**GPC4SEL**	
7	6	5	4	3	2	1	0
GPC3SEL		**GPC2SEL**		**GPC1SEL**		**GPC0SEL**	

Bits	Descriptions	
[2x+1:2x]	**GPCxSEL**	Selection for GPCx as one of input Pins to IRQ0, IRQ1, IRQ2, or IRQ3 interrupt source

Where x=0~15.
GPExSEL = 0, GPCx pin is grouped as one of interrupt sources to IRQ0.
1, GPCx pin is grouped as one of interrupt sources to IRQ1.
2, GPCx pin is grouped as one of interrupt sources to IRQ2.
3, GPCx pin is grouped as one of interrupt sources to IRQ3.

GPIO A Interrupt Enable (IRQENGPA)

Register	Address	R/W	Description	Reset Value
IRQENGPA	GP_BA+0x90	R/W	GPIO Port A Interrupt Enable	0x0000_0000

31	30	29	28	27	26	25	24
PA15ENR	**PA14ENR**	**PA13ENR**	**PA12ENR**	**PA11ENR**	**PA10ENR**	**PA9ENR**	**PA8ENR**
23	22	21	20	19	18	17	16
PA7ENR	**PA6ENR**	**PA5ENR**	**PA4ENR**	**PA3ENR**	**PA2ENR**	**PA1ENR**	**PA0ENR**
15	14	13	12	11	10	9	8
PA15ENF	**PA14ENF**	**PA13ENF**	**PA12ENF**	**PA11ENF**	**PA10ENF**	**PA9ENF**	**PA8ENF**
7	6	5	4	3	2	1	0
PA7ENF	**PA6ENF**	**PA5ENF**	**PA4ENF**	**PA3ENF**	**PA2ENF**	**PA1ENF**	**PA0ENF**

Bits	Descriptions	
[x]	**PAxENF**	**Enable GPAx input falling edge to trigger one of interrupt sources IRQ0~IRQ3. IRQSRCGPA register determines which IRQn (n=0~3) is the destination.**
[x+16]	**PAxENR**	Enable GPAx input rising edge to trigger one of interrupt sources IRQ0~IRQ3. IRQSRCGPA register determines which IRQn (n=0~3) is the destination.

Where x=0~15.
PAxENF and PAxENR can be set "1" at the same time.

NOTE1: In normal operation mode, for each pin, PAxENF and PAxENR can be set both to detect both rising and falling edge.
NOTE2: When use a pin as powerdown wake up source, the setting of edges must be explained as level trigger. For example, if set one pin for rising, user must keep this pin low while start to enter power down; a high level will make power-down entrance be ignored. After entering power down, a high level at this pin will make chip leave power-down.
NOTE3: When use a pin as power-down wake up source, if both edges are set, the high level will be set as wake up level.

GPIO B Interrupt Enable (IRQENGPB)

Register	Address	R/W	Description	Reset Value
IRQENGPB	GP_BA+0x94	R/W	GPIO Port B Interrupt Enable	0x0000_0000

31	30	29	28	27	26	25	24
Reserved						**PB9ENR**	**PB8ENR**
23	22	21	20	19	18	17	16
PB7ENR	**PB6ENR**	**PB5ENR**	**PB4ENR**	**PB3ENR**	**PB2ENR**	**PB1ENR**	**PB0ENR**
15	14	13	12	11	10	9	8
Reserved						**PB9ENF**	**PB8ENF**
7	6	5	4	3	2	1	0
PB7ENF	**PB6ENF**	**PB5ENF**	**PB4ENF**	**PB3ENF**	**PB2ENF**	**PB1ENF**	**PB0ENF**

Bits	Descriptions	
[x]	**PBxENF**	**Enable GPBx input falling edge to trigger one of interrupt sources IRQ0~IRQ3. IRQSRCGPB register determines which IRQn (n=0~3) is the destination.**
[x+16]	**PBxENR**	Enable GPBx input rising edge to trigger one of interrupt sources IRQ0~IRQ3. IRQSRCGPB register determines which IRQn (n=0~3) is the destination.

Where x=0~15.
PBxENF and PBxENR can be set "1" at the same time.

NOTE1: In normal operation mode, for each pin, PBxENF and PBxENR can be set both to detect both rising and falling edge.
NOTE2: When use a pin as powerdown wake up source, the setting of edges must be explained as level trigger. For example, if set one pin for rising, user must keep this pin low while start to enter power down; a high level will make power-down entrance be ignored. After entering power down, a high level at this pin will make chip leave power-down.
NOTE3: When use a pin as power-down wake up source, if both edges are set, the high level will be set as wake up level.

GPIO C Interrupt Enable (IRQENGPC)

Register	Address	R/W	Description	Reset Value
IRQENGPC	GP_BA+0x98	R/W	GPIO Port C Interrupt Enable	0x0000_0000

31	30	29	28	27	26	25	24
Reserved					**PC10ENR**	**PC9ENR**	**PC8ENR**
23	22	21	20	19	18	17	16
PC7ENR	**PC6ENR**	**PC5ENR**	**PC4ENR**	**PC3ENR**	**PC2ENR**	**PC1ENR**	**PC0ENR**
15	14	13	12	11	10	9	8
Reserved					**PC10ENF**	**PC9ENF**	**PC8ENF**
7	6	5	4	3	2	1	0
PC7ENF	**PC6ENF**	**PC5ENF**	**PC4ENF**	**PC3ENF**	**PC2ENF**	**PC1ENF**	**PC0ENF**

Bits	Descriptions	
[x]	**PCxENF**	**Enable GPCx input falling edge to trigger one of interrupt sources IRQ0~IRQ3. IRQSRCGPC register determines which IRQn (n=0~3) is the destination.**
[x+16]	**PCxENR**	Enable GPCx input rising edge to trigger one of interrupt sources IRQ0~IRQ3. IRQSRCGPC register determines which IRQn (n=0~3) is the destination.

Where x=0~15.
PCxENF and PCxENR can be set "1" at the same time.

NOTE1: In normal operation mode, for each pin, PCxENF and PCxENR can be set both to detect both rising and falling edge.
NOTE2: When use a pin as powerdown wake up source, the setting of edges must be explained as level trigger. For example, if set one pin for rising, user must keep this pin low while start to enter power down; a high level will make power-down entrance be ignored. After entering power down, a high level at this pin will make chip leave power-down.
NOTE3: When use a pin as power-down wake up source, if both edges are set, the high level will be set as wake up level.

Interrupt Latch Trigger Selection (IRQLHSEL)

Register	Address	R/W	Description	Reset Value
IRQLHSEL	GP_BA+0xA0	R/W	Interrupt Latch Trigger Selection Register	0x0000_0000

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							**IRQ_SRCC**
7	6	5	4	3	2	1	0
IRQ3Wake	**IRQ2Wake**	**IRQ1Wake**	**IRQ0Wake**	**IRQ3LHE**	**IRQ2LHE**	**IRQ1LHE**	**IRQ0LHE**

Bits	Descriptions	
[31:9]	Reserved	**Reserved**
[8]	**IRQ_SRCC**	Interrupt Request Source Control 0 = While the gpio interrupt occur, the gpio interrupt controller generate one clock pulse to the AIC **1 = While the gpio interrupt occur, the interrupt from gpio to AIC will keep till the CPU clear the interrupt trigger source. (**IRQTGSRC0, IRQTGSRC1)
[7:4]	**IRQxWake**	GPIO interrupt wake up system enable While IRQxWake is "1", enable the GPIO IRQx wake up the chip from power down mode.
[3:0]	**IRQxLHE**	While IRQxLTH is "1", it enables active IRQx interrupt to latch the input values of GPAx/GPBx/GPCx to IRQLHGPA/IRQLHGPB/IRQLHGPC register simultaneously.

Where x=0~3.

GPIO A Interrupt Latch (IRQLHGPA)

Register	Address	R/W	Description	Reset Value
IRQLHGPA	GP_BA+0xA4	R	GPIO Port A Interrupt Latch Value	0x0000_0000

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
PA15LHV	**PA14LHV**	**PA13LHV**	**PA12LHV**	**PA11LHV**	**PA10LHV**	**PA9LHV**	**PA8LHV**
7	6	5	4	3	2	1	0
PA7LHV	**PA6LHV**	**PA5LHV**	**PA4LHV**	**PA3LHV**	**PA2LHV**	**PA1LHV**	**PA0LHV**

Bits	Descriptions	
[x]	**PAxLHV**	Latched value of GPAx while the IRQ (IRQ0~IRQ3) selected by IRQLHSEL is active.

Where x=0~15.

GPIO B Interrupt Latch (IRQLHGPB)

Register	Address	R/W	Description	Reset Value
IRQLHGPB	GP_BA+0xA8	R	GPIO Port B Interrupt Latch Value	0x0000_0000

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						PB9LHV	PB8LHV
7	6	5	4	3	2	1	0
PB7LHV	PB6LHV	PB5LHV	PB4LHV	PB3LHV	PB2LHV	PB1LHV	PB0LHV

Bits	Descriptions	
[x]	**PBxLHV**	Latched value of GPBx while the IRQ (IRQ0~IRQ3) selected by IRQLHSEL is active.

Where x=0~15.

GPIO C Interrupt Latch (IRQLHGPC)

Register	Address	R/W	Description	Reset Value
IRQLHGPC	GP_BA+0xAC	R	GPIO Port C Interrupt Latch Value	0x0000_0000

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					**PC10LHV**	**PC9LHV**	**PC8LHV**
7	6	5	4	3	2	1	0
PC7LHV	**PC6LHV**	**PC5LHV**	**PC4LHV**	**PC3LHV**	**PC2LHV**	**PC1LHV**	**PC0LHV**

Bits	Descriptions	
[x]	**PCxLHV**	Latched value of GPCx while the IRQ (IRQ0~IRQ3) selected by IRQLHSEL is active.

Where x=0~15.

NOTE: When a latched pin value is '0', there will be 2 meanings: either the pin's input is recognized as LOW or the pin is setup as output mode, so the input value is masked as '0'.

IRQ Interrupt Trigger Source 0 (IRQTGSRC0)

Register	Address	R/W	Description	Reset Value
IRQTGSRC0	GP_BA+0xB4	R/C	IRQ0~3 Interrupt Trigger Source Indicator from GPIO Port A and GPIO Port B	0x0000_0000

31	30	29	28	27	26	25	24
Reserved						**PB9TG**	**PB8TG**
23	22	21	20	19	18	17	16
PB7TG	**PB6TG**	**PB5TG**	**PB4TG**	**PB3TG**	**PB2TG**	**PB1TG**	**PB0TG**
15	14	13	12	11	10	9	8
PA15TG	**PA14TG**	**PA13TG**	**PA12TG**	**PA11TG**	**PA10TG**	**PA9TG**	**PA8TG**
7	6	5	4	3	2	1	0
PA7TG	**PA6TG**	**PA5TG**	**PA4TG**	**PA3TG**	**PA2TG**	**PA1TG**	**PA0TG**

Bits	Descriptions	
[x]	**PAxTG**	When this bit is read as "1", it indicates GPAx is the trigger source to generate interrupt to the IRQ (IRQ0~IRQ3) selected by IRQLHSEL[4]. Write 1 to the bit[x] will clear the correspond interrupt source
[x+16]	**PBxTG**	When this bit is read as "1", it indicates GPBx is the trigger source to generate interrupt to the IRQ (IRQ0~IRQ3) selected by IRQLHSEL[4]. Write 1 to the bit[x] will clear the correspond interrupt source

Where x=0~15.

NOTE: The trigger source will be latched when the corresponding rising or falling trigger enable is setup and the pin state toggle is recognized (through de-bounce or without de-bounce), no matter whether the source is an input or output pin.

IRQ Interrupt Trigger Source 1 (IRQTGSRC1)

Register	Address	R/W	Description	Reset Value
IRQTGSRC1	GP_BA+0xB8	R/C	IRQ0~3 Interrupt Trigger Source Indicator from GPIO Port C	0x0000_0000

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					**PC10TG**	**PC9TG**	**PC8TG**
7	6	5	4	3	2	1	0
PC7TG	**PC6TG**	**PC5TG**	**PC4TG**	**PC3TG**	**PC2TG**	**PC1TG**	**PC0TG**

Bits	Descriptions	
[x]	**PCxTG**	When this bit is read as "1", it indicates GPCx is the trigger source to generate interrupt to the IRQ (IRQ0~IRQ3) selected by IRQLHSEL[4]. Write 1 to the bit[x] will clear the correspond interrupt source

Where x=0~15.

NOTE: The trigger source will be latched when the corresponding rising or falling trigger enable is setup and the pin state toggle is recognized (through de-bounce or without de-bounce), no matter whether the source is an input or output pin.

Other NOTE for related setup

NOTE1: For the AIC's normal functionality to be triggered by external IO interrupt, the external IRQ source settings (for IRQ0/1/2/3) can only be set as positive edge, see AIC_SCRxx/bit7~6: SRCTYPE.
NOTE2: For power-down wake up setting, in order to keep normal wake up functionality, the wake up source polarity should be set as positive level, see IRQWAKECON/bit7~4: IRQWAKEUPPOL.

6.12 I2C Synchronous Serial Interface

6.12.1 Overview

I^2C is a two-wire, bi-directional serial bus that provides a simple and efficient method of data exchange between devices. The I^2C standard is a multi-master bus with integrated addressing and data-transfer protocols. It includes collision and arbitration loses detection that prevents data corruption if two or more masters attempt to control the bus simultaneously.

Serial, 8-bit oriented bi-directional data transfers can be made up to 100k bit/s in Standard-mode, up to 400k bit/s in the Fast-mode.

Data is transferred between a Master and a Slave synchronously to SCL on the SDA line on a byte-by-byte basis. Each data byte is 8 bits long. There is one SCL clock pulse for each data bit with the MSB being transmitted first. An acknowledge bit follows each transferred byte. Each bit is sampled during the high period of SCL; therefore, the SDA line may be changed only during the low period of SCL and must be held stable during the high period of SCL. A transition on the SDA line while SCL is high is interpreted as a command (START or STOP).

6.12.2 Feature

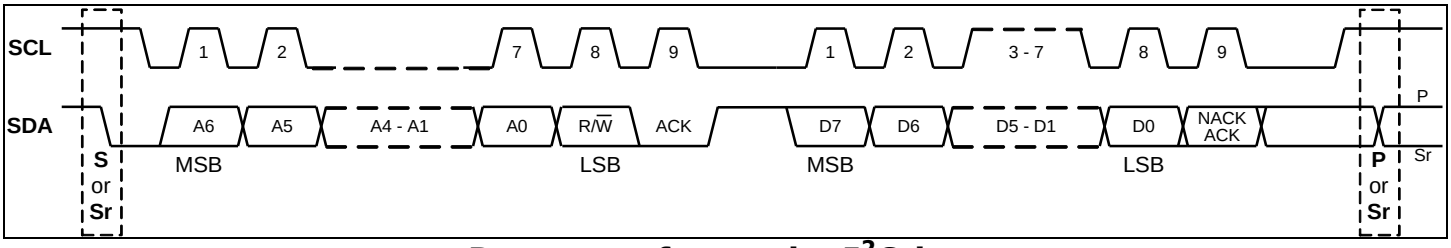
The I^2C Master Core includes the following features:

- AMBA APB interface compatible
- Compatible with Philips I^2C standard, support master mode
- Multi Master Operation
- Clock stretching and wait state generation
- Provide multi-byte transmit operation, up to 4 bytes can be transmitted in a single transfer
- Software programmable acknowledge bit
- Arbitration lost interrupt, with automatic transfer cancellation
- Start/Stop/Repeated Start/Acknowledge generation
- Start/Stop/Repeated Start detection
- Bus busy detection
- Supports 7 bit addressing mode
- Fully static synchronous design with one clock domain
- Software mode I^2C

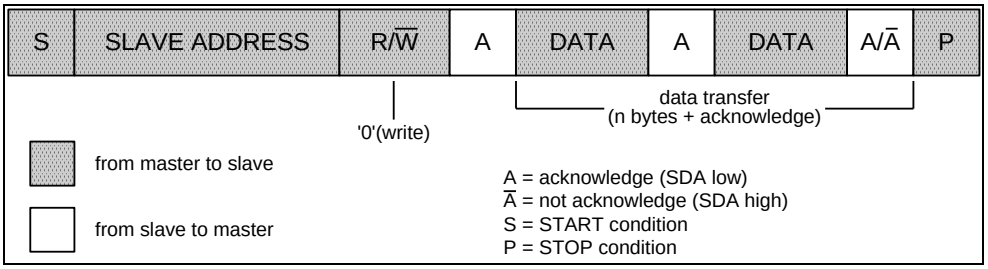
6.12.3 I²C Protocol

Normally, a standard communication consists of four parts:

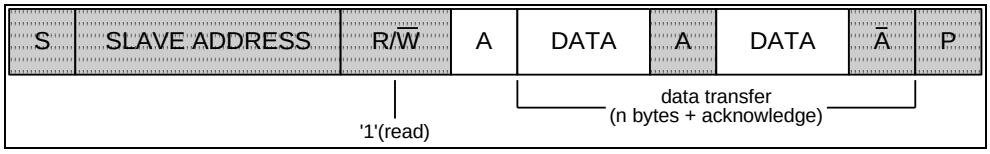
1) STA^{R}T or Repeated START signal generation
2) Slave address transfer
3) Data transfer
4) STOP signal generation



Data transfer on the I²C-bus



A master-transmitter addressing a slave receiver with a 7-bit address
The transfer direction is not changed



A master reads a slave immediately after the first byte (address)

START or Repeated START signal

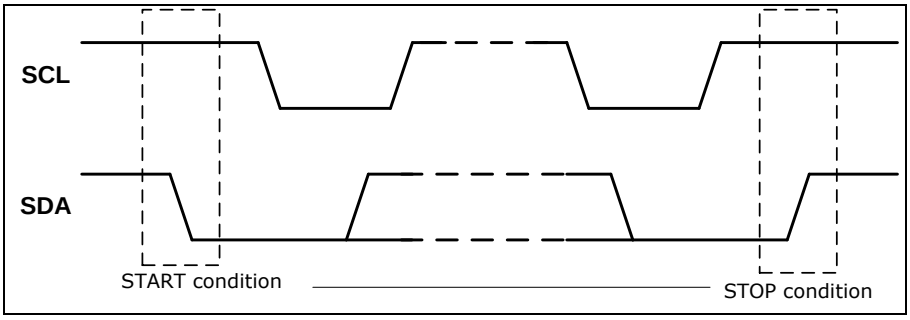
When the bus is free/idle, meaning no master device is engaging the bus (both SCL and SDA lines are high), a master can initiate a transfer by sending a START signal. A START signal, usually referred to as the **S-bit**, is defined as a **HIGH to LOW** transition on the SDA line while SCL is **HIGH**. The START signal denotes the beginning of a new data transfer.

A Repeated START (Sr) is a START signal without first generating a STOP signal. The master uses this method to communicate with another slave or the same slave in a different transfer direction (e.g. from writing to a device to reading from a device) without releasing the bus.

The I^2C core generates a START signal when the START bit in the Command Register (CMDR) is set and the READ or WRITE bits are also set. Depending on the current status of the SCL line, a START or Repeated START is generated.

STOP signal

The master can terminate the communication by generating a STOP signal. A STOP signal, usually referred to as the **P-bit**, is defined as a **LOW to HIGH** transition on the SDA line while SCL is **HIGH**.

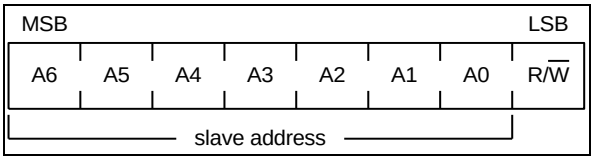


START and STOP conditions

Slave Address Transfer

The first byte of data transferred by the master immediately after the START signal is the slave address. This is a 7-bits calling address followed by a RW bit. The RW bit signals the slave the data transfer direction. No two slaves in the system can have the same address. Only the slave with an address that matches the one transmitted by the master will respond by returning an acknowledge bit by pulling the SDA low at the 9th SCL clock cycle.

The core treats a Slave Address Transfer as any other write action. Store the slave device's address in the Transmit Register (TxR) and set the WRITE bit. The core will then transfer the slave address on the bus.



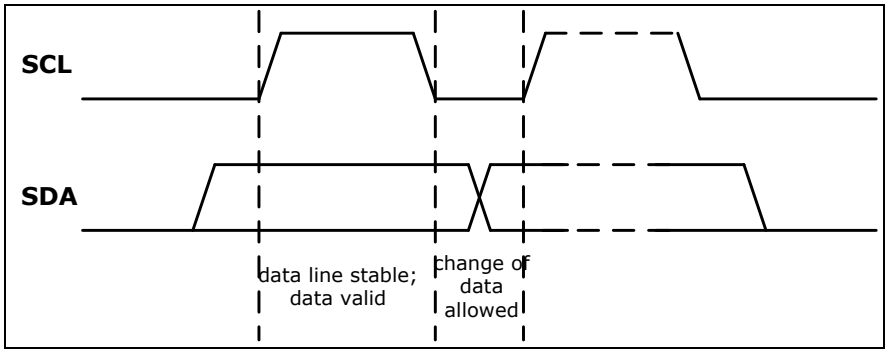
The first byte after the START procedure

Data Transfer

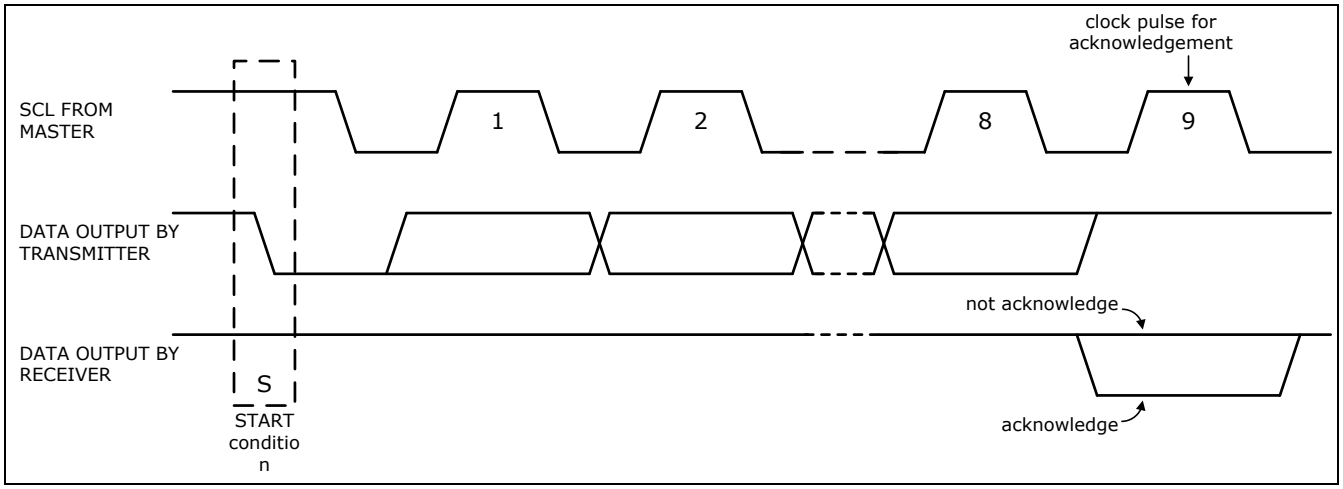
Once successful slave addressing has been achieved, the data transfer can proceed on a byte-by-byte basis in the direction specified by the RW bit sent by the master. Each transferred byte is followed by an acknowledge bit on the 9th SCL clock cycle. If the slave signals a **Not Acknowledge (NACK)**, the master can generate a STOP signal to abort the data transfer or generate a Repeated START signal and start a new transfer cycle.

If the master, as the receiving device, does **Not Acknowledge (NACK)** the slave, the slave releases the SDA line for the master to generate a STOP or Repeated START signal.

To write data to a slave, store the data to be transmitted in the Transmit Register (TxR) and set the WRITE bit. To read data from a slave, set the READ bit. During a transfer the core set the I2C_TIP flag, indicating that a **Transfer is in Progress**. When the transfer is done the I2C_TIP flag is cleared, the IF the flag set if enabled, then an interrupt generated. The Receive Register (RxR) contains valid data after the IF flag has been set. The software may issue a new write or read command when the I2C_TIP flag is cleared.



Bit transfer on the I²C-bus



Acknowledge on the I²C-bus

6.12.4 I²C Programming Examples

Example 1

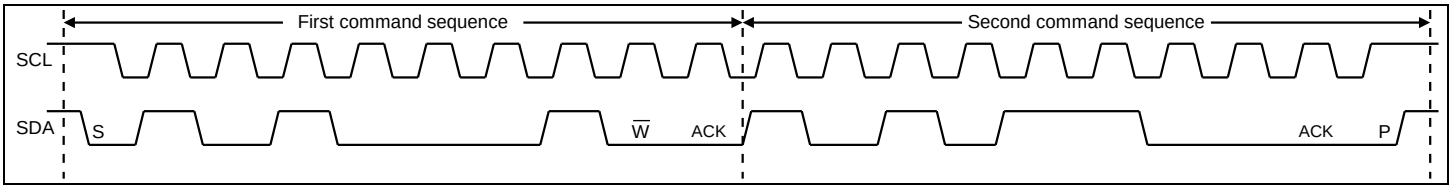
Write 1 byte of data to a slave (using multi-byte transmit mode).

Slave address = 0x51 (7b'1010001)
Data to write = 0xAC

I²C Sequence:

1) generate start command
2) write slave address + write bit
3) receive acknowledge from slave
4) write data
5) receive acknowledge from slave
6) generate stop command

Commands:
1) Write a value into DIVIDER to determine the frequency of serial clock.
2) Set Tx_NUM = 0x1 and set I2C_EN = 1 to enable I^2C core.
3) Write 0xA2 (address + write bit) to Transmit Register (TxR[15:8]) and 0xAC to TxR[7:0].
4) Set START bit and WRITE bit.
   –– Wait for interrupt or I2C_TIP flag to negate ––
5) Read I2C_RxACK bit from CSR Register, it should be '0'.
6) Set Tx_NUM = 0x0.
7) Set STOP bit.
   –– Wait for interrupt or I2C_TIP flag to negate ––



NOTE: Please note that the time for the Interrupt Service Routine is not shown here. It is assumed that the ISR is much faster then the I^2C cycle time, and therefore not visible.

Example 2

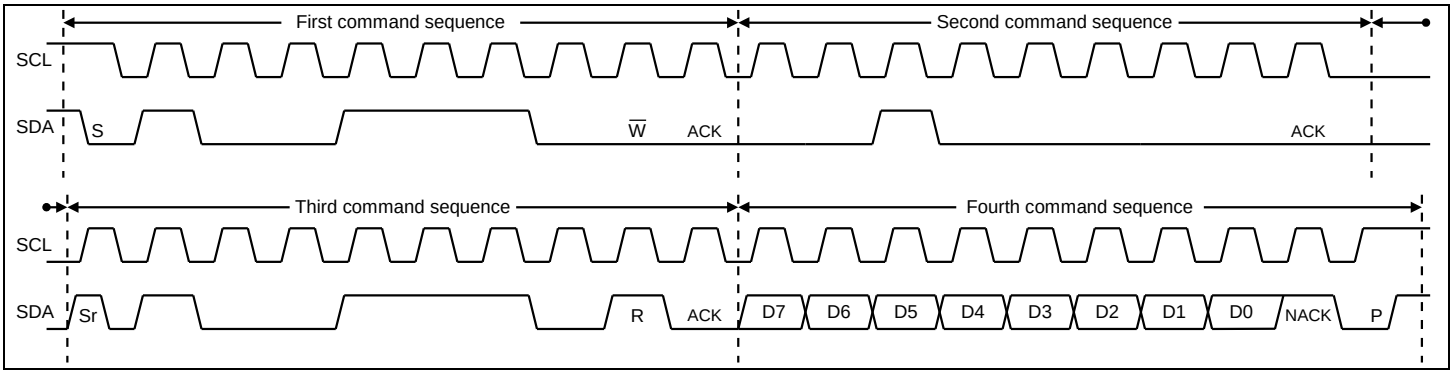
Read a byte of data from an I2C memory device (using single byte transfer mode).

Slave address = 0x4E (7'b1001110)
Memory location to read from = 0x20

I2C sequence:
1) generate start signal
2) write slave address + write bit, then receive acknowledge from slave
3) write memory location, then receive acknowledge from slave
4) generate repeated start signal
5) write slave address + read bit, then receive acknowledge from slave
6) read byte from slave
7) write not acknowledge (NACK) to slave, indicating end of transfer
8) generate stop signal

Commands:

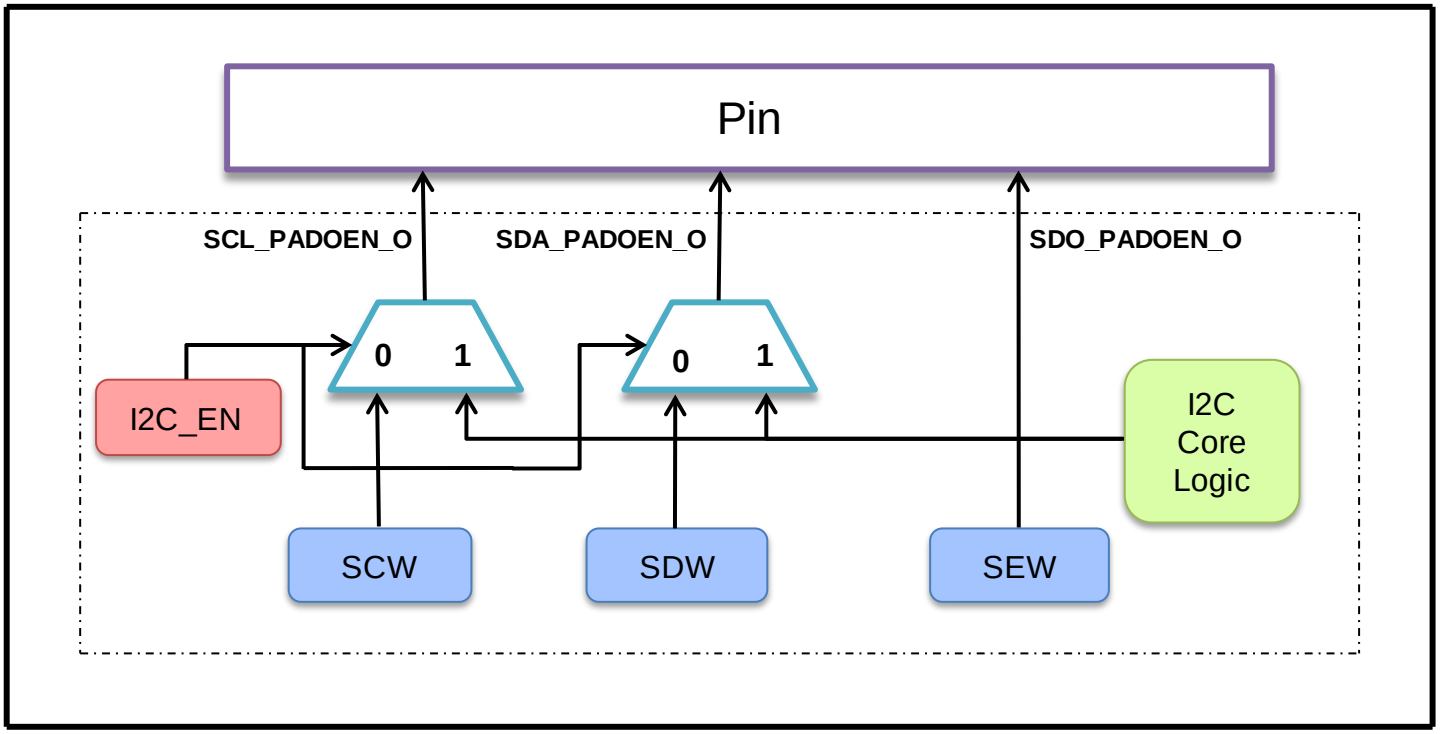
1) Write a value into DIVIDER to determine the frequency of serial clock.
2) Set Tx_NUM = 0x0 and set I2C_EN = 1 to enable I^2C core.
3) Write 0x9C (address + write bit) to TxR[7:0], set START bit and WRITE bit.
   –– Wait for interrupt or I2C_TIP flag to negate ––
4) Read I2C_RxACK bit from CSR Register, it should be '0'.
5) Write 0x20 to TxR[7:0], set WRITE bit.
   –– Wait for interrupt or I2C_TIP flag to negate ––
6) Read I2C_RxACK bit from CSR Register, it should be '0'.
7) Write 0x9D (address + read bit) to TxR[7:0], set START bit, set WRITE bit.
   –– Wait for interrupt or I2C_TIP flag to negate ––
8) Read I2C_RxACK bit from CSR Register, it should be '0'.
9) Set READ bit, set ACK to '1' (NACK), set STOP bit.
10) Read out received data from RxR, it will put on RxR[7:0].



NOTE: Please note that the time for the Interrupt Service Routine is not shown here. It is assumed that the ISR is much faster then the I^2C cycle time, and therefore not visible.

6.12.5 Software I^2C Operation

The software I^2C function contains 3 registers for software to control the output enable of pad actually. The implementation of software I^2C is shown bellow.



The other two registers – SCW and , SDW just represent the status of input port – scl pin, sda pin. Software can read/write this register at any time, but the output enable – scl pin and sda pin are controlled by software only when I2C_EN = 0.

6.12.6 I²C Serial Interface Control Registers Mapping

R: read only, **W**: write only, **R/W**: both read and write

Register	Offset	R/W/C	Description	Reset Value
I2C_BA = 0xB800_4000				
CSR	I2C_BA+0x00	R/W	Control and Status Register	0x0000_0000
DIVIDER	I2C_BA+0x04	R/W	Clock Pre-scale Register	0x0000_0000
CMDR	I2C_BA+0x08	R/W	Command Register	0x0000_0000
SWR	I2C_BA+0x0C	R/W	Software Mode Control Register	0x0000_003F
RxR	I2C_BA+0x10	R	Data Receive Register	0x0000_0000
TxR	I2C_BA+0x14	R/W	Data Transmit Register	0x0000_0000

NOTE: The reset value of SWR is 0x3F only when SCR, SDR and SER are connected to pull high resistor.

Control and Status Register (CSR)

Register	Offset	R/W/C	Description	Reset Value
CSR	0x00	R/W	Control and Status Register	0x0000_0000

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				**I2C_RxACK**	**I2C_BUSY**	**I2C_AL**	**I2C_TIP**
7	6	5	4	3	2	1	0
Reserved		**Tx_NUM**		**Reserved**	**IF**	**IE**	**I2C_EN**

Bits	Descriptions	
[31:12]	**Reserved**	**Reserved**
[11]	**I2C_RxACK**	**Received Acknowledge From Slave (Read only)** This flag represents acknowledge from the addressed slave. 0 = Acknowledge received (ACK). 1 = Not acknowledge received (NACK).
[10]	**I2C_BUSY**	**I^2C Bus Busy (Read only)** 0 = After STOP signal detected. 1 = After START signal detected.
[9]	**I2C_AL**	**Arbitration Lost (Read only)** This bit is set when the I^2C core lost arbitration. Arbitration is lost when: A STOP signal is detected, but no requested. The master drives SDA high, but SDA is low.
[8]	**I2C_TIP**	**Transfer In Progress (Read only)** 0 = Transfer complete. 1 = Transferring data. **NOTE:** When a transfer is in progress, you will not allow writing to any register of the I^2C master core except SWR.
[7:6]	**Reserved**	**Reserved**
[5:4]	**Tx_NUM**	**Transmit Byte Counts** These two bits represent how many bytes are remained to transmit. When a byte has been transmitted, the Tx_NUM will decrease 1 until all bytes are transmitted (Tx_NUM = 0x0) or NACK received from slave. Then the interrupt signal will assert if IE was set. 0x0 = Only one byte is left for transmission. 0x1 = Two bytes are left to for transmission. 0x2 = Three bytes are left for transmission. 0x3 = Four bytes are left for transmission. **NOTE:** When NACK received, Tx_NUM will not decrease.
[3]	**Reserved**	**Reserved**
[2]	**IF**	**Interrupt Flag**

		The Interrupt Flag is set when: Transfer has been completed. Transfer has not been completed, but slave responded NACK (in multi-byte transmit mode). Arbitration is lost. **NOTE:** This bit is read only, but can be cleared by writing 1 to this bit.
[1]	**IE**	**Interrupt Enable** 0 = **Disable** I^2C Interrupt. 1 = **Enable** I^2C Interrupt.
[0]	**I2C_EN**	**I^2C Core Enable** 0 = **Disable** I^2C core, serial bus outputs are controlled by SDW/SCW. 1 = **Enable** I^2C core, serial bus outputs are controlled by I^2C core.

Pre-scale Register (DIVIDER)

Register	Offset	R/W/C	Description	Reset Value
DIVIDER	0x04	R/W	Clock Pre-scale Register	0x0000_0000

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
DIVIDER[15:8]							
7	6	5	4	3	2	1	0
DIVIDER[7:0]							

Bits	Descriptions	
[31:16]	**Reserved**	**Reserved**
[15:0]	**DIVIDER**	**Clock Pre-scale Register** It is used to pre-scale the SCL clock line. Due to the structure of the I^2C interface, the core uses a 5*SCL clock internally. The pre-scale register must be programmed to this 5*SCL frequency (minus 1). Change the value of the pre-scale register only when the "I2C_EN" bit is cleared. Example: PCLK = 32MHz, desired SCL = 100KHz $prescale = \frac{32MHz}{5*100KHz} - 1 = 63(dec) = 3F(hex)$

Command Register (CMDR)

Register	Offset	R/W/C	Description	Reset Value
CMDR	0x08	R/W	Command Register	0x0000_000x

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							
7	6	5	4	3	2	1	0
Reserved	**START**	**STOP**		**READ**	**WRITE**		**ACK**

NOTE: Software can write this register only when I2C_EN = 1.

Bits	Descriptions	
[31:5]	**Reserved**	**Reserved**
[4]	**START**	**Generate Start Condition** Generate (repeated) start condition on I^2C bus when this bit set 1.
[3]	**STOP**	**Generate Stop Condition** Generate stop condition on I^2C bus when this bit set 1.
[2]	**READ**	**Read Data From Slave** Retrieve data from slave when this bit set 1.
[1]	**WRITE**	**Write Data To Slave** Transmit data to slave when this bit set 1.
[0]	**ACK**	**Send Acknowledge To Slave** When I^2C behaves as a receiver, sent ACK (ACK = '0') or NACK (ACK = '1') to slave.

NOTE: The START, STOP, READ and WRITE bits are cleared automatically while transfer finished. READ and WRITE cannot be set concurrently.

Software Mode Register (SWR)

Register	Offset	R/W/C	Description	Reset Value
SWR	0x0C	R/W	Software Mode Control Register	0x0000_003F

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							
7	6	5	4	3	2	1	0
Reserved		**SER**	**SDR**	**SCR**	**SEW**	**SDW**	**SCW**

NOTE: This register is used as software mode of I^2C. Software can read/write this register no matter I2C_EN is 0 or 1. But SCL and SDA are controlled by software only when I2C_EN = 0.

Bits	Descriptions	
[31:6]	**Reserved**	**Reserved**
[5]	**SER**	**Serial Interface SDO Status (Read only)** 0 = SDO is **Low**. 1 = SDO is **High**.
[4]	**SDR**	**Serial Interface SDA Status (Read only)** 0 = SDA is **Low**. 1 = SDA is **High**.
[3]	**SCR**	**Serial Interface SCK Status (Read only)** 0 = SCL is **Low**. 1 = SCL is **High**.
[2]	**SEW**	**Serial Interface SDO Output Control** 0 = SDO pin is driven **Low**. 1 = SDO pin is **tri-state**.
[1]	**SDW**	**Serial Interface SDA Output Control** 0 = SDA pin is driven **Low**. 1 = SDA pin is **tri-state**.
[0]	**SCW**	**Serial Interface SCK Output Control** 0 = SCL pin is driven **Low**. 1 = SCL pin is **tri-state**.

Data Receive Register (RxR)

Register	Offset	R/W/C	Description	Reset Value
RxR	0x10	R	Data Receive Register	0x0000_0000

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							
7	6	5	4	3	2	1	0
Rx[7:0]							

Bits	Descriptions	
[31:8]	**Reserved**	**Reserved**
[7:0]	**Rx**	**Data Receive Register** The last byte received via I^2C bus will put on this register. The I^2C core only used 8-bit receive buffer.

Data Transmit Register (TxR)

Register	Offset	R/W/C	Description	Reset Value
TxR	0x14	R/W	Data Transmit Register	0x0000_0000

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

Bits	Descriptions	
[31:0]	**Tx**	**Data Transmit Register** The I^2C core used 32-bit transmit buffer and provide multi-byte transmit function. Set CSR[Tx_NUM] to a value that you want to transmit. I^2C core will always issue a transfer from the highest byte first. For example, if CSR[Tx_NUM] = 0x3, Tx[31:24] will be transmitted first, then Tx[23:16], and so on. In case of a data transfer, all bits will be treated as data. In case of a slave address transfer, the first 7 bits will be treated as 7-bit address and the LSB represent the R/W bit. In this case, LSB = 1, reading from slave LSB = 0, writing to slave

6.13 PWM-Timer

6.13.1 Introduction

There are 4 PWM-Timers enclosed. The 4 PWM-Timers has 2 Pre-scale, 2 clock divider, 4 clock selectors, 4 16-bit counters, 4 16-bit comparators, 2 Dead-Zone generators. They are all driven by APB clock. Each can be used as a timer and issues interrupt independently.

Each two PWM-Timers share the same pre-scale (0-1 share prescale0 and 2-3 share prescale1). Clock divider provides each timer with 5 clock sources (1, 1/2, 1/4, 1/8, 1/16). Each timer receives its own clock signal from clock divider which receives clock from 8-bit pre-scale. The 16-bit counter in each timer receive clock signal from clock selector and can be used to handle one PWM period. The 16-bit comparator compares number in counter with threshold number in register loaded previously to generate PWM duty cycle.

The clock signal from clock divider is called PWM clock. Dead-Zone generator utilize PWM clock as clock source. Once Dead-Zone generator is enabled, output of two PWM-Timers are blocked. Two output pin are all used as Dead-Zone generator output signal to control off-chip power device. Dead-Zone generator 0 is used to control outputs of timer 0&1, and Dead-Zone generator 1 is used to control outputs of timer 2&3.

To prevent PWM driving output pin with unsteady waveform, 16-bit counter and 16-bit comparator are implemented with double buffering feature. User can feel free to write data to counter buffer register and comparator buffer register without generating glitch.

When 16-bit down counter reaches zero, the interrupt request is generated to inform CPU that time is up. When counter reaches zero, if counter is set as toggle mode, it is reloaded automatically and start to generate next cycle. User can set counter as one-shot mode instead of toggle mode. If counter is set as one-shot mode, counter will stop and generate one interrupt request when it reaches zero.

The value of comparator is used for pulse width modulation. The counter control logic changes the output level when down-counter value matches the value of compare register.

Each PWM-Timer includes a capture channel. The Capture 0 and PWM 0 share a timer that included in PWM 0; and the Capture 1 and PWM 1 share another timer, and etc. Therefore user must setup the PWM-Timer before turn on Capture feature. Please reference the section of PWM-Timer for more detail description of setup PWM-Timer. After enabling capture feature, the capture always latched PWM-counter to CRLR when input channel has a rising transition and latched PWM-counter to CFLR when input channel has a falling transition. Capture channel 0 interrupt is programmable by setting CCR0[1] (Rising latch Interrupt enable) and CCR0[2] (Falling latch Interrupt enable) to decide the condition of interrupt occur. Capture channel 1 has the same feature by setting CCR0[17] and CCR0[18]. And capture channel 2 & 3 has the same feature by setting CCR1[1],CCR1[2] and CCR1[17], CCR1[18] respectively. Whenever Capture issues Interrupt 0/1/2/3, the PWM counter 0/1/2/3 will be reload at this moment.

There are only four interrupts from PWM to advanced interrupt controller (AIC). PWM 0 and Capture 0

share the same interrupt; PWM1 and Capture 1 share the same interrupt and so on. Therefore, PWM function and Capture function in the same channel cannot be used at the same time.

6.13.2 Features

- Two 8-bit pre-scales and Two clock dividers
- Four clock selectors
- Four 16-bit counters and four 16-bit comparators
- Two Dead-Zone generator
- Capture function

6.13.3 PWM Timer Start Procedure

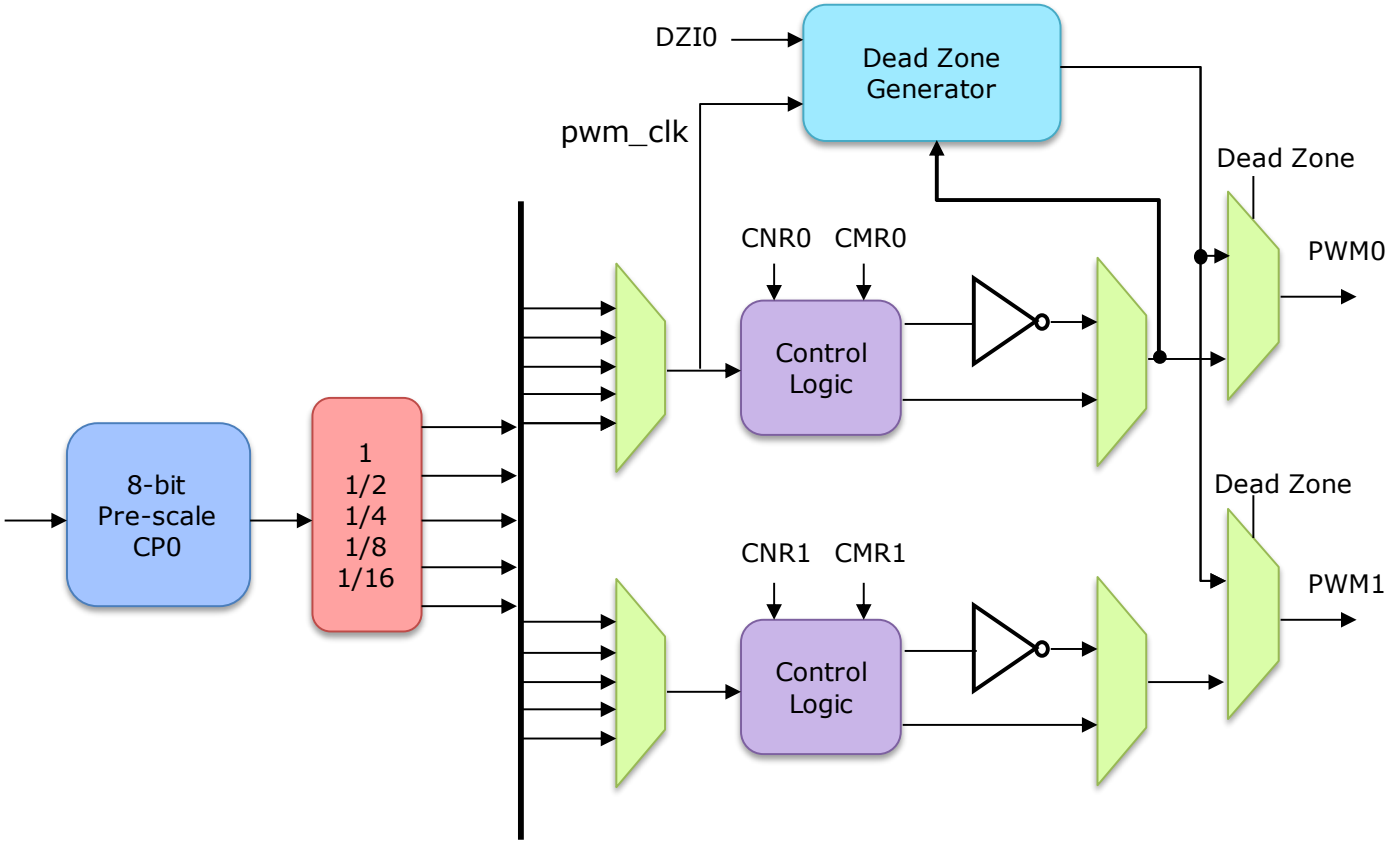
1. Setup clock selector (CSR)
2. Setup prescale & dead zone interval (PPR)
3. Setup inverter on/off, dead zone generator on/off, toggle mode /one-shot mode, and PWM timer off. (PCR)
4. Setup comparator register (CMR)
5. Setup counter register (CNR)
6. Setup interrupt enable register (PIER)
7. Setup pwm output enable (POE)
8. Enable PWM timer (PCR)

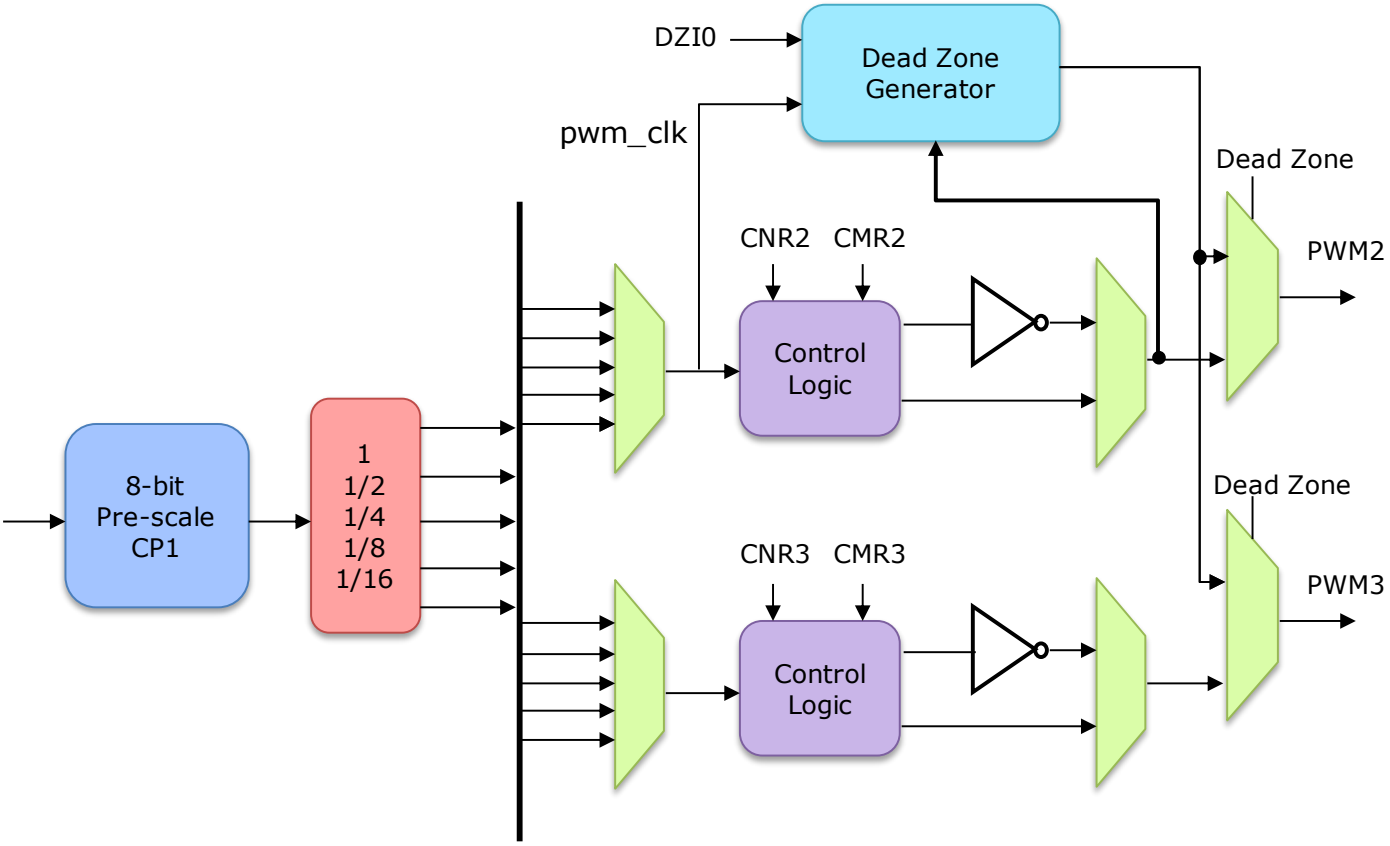
6.13.4 PWM Architecture

PWM_OE[0] enable → timer PWM0 output → GPA[12] or GPB[1] or GPB[8] or GPC[3] or GPC[7]
PWM_OE[1] enable → timer PWM1 output → GPA[13] or GPB[2] or GPB[9] or GPC[4] or GPC[8]
PWM_OE[2] enable → timer PWM2 output → GPA[15] or GPB[3] or GPB[6] or GPC[5] or GPC[9]
PWM_OE[3] enable → timer PWM3 output → GPB[0] or GPB[4] or GPB[7] or GPC[6] or GPC[10]

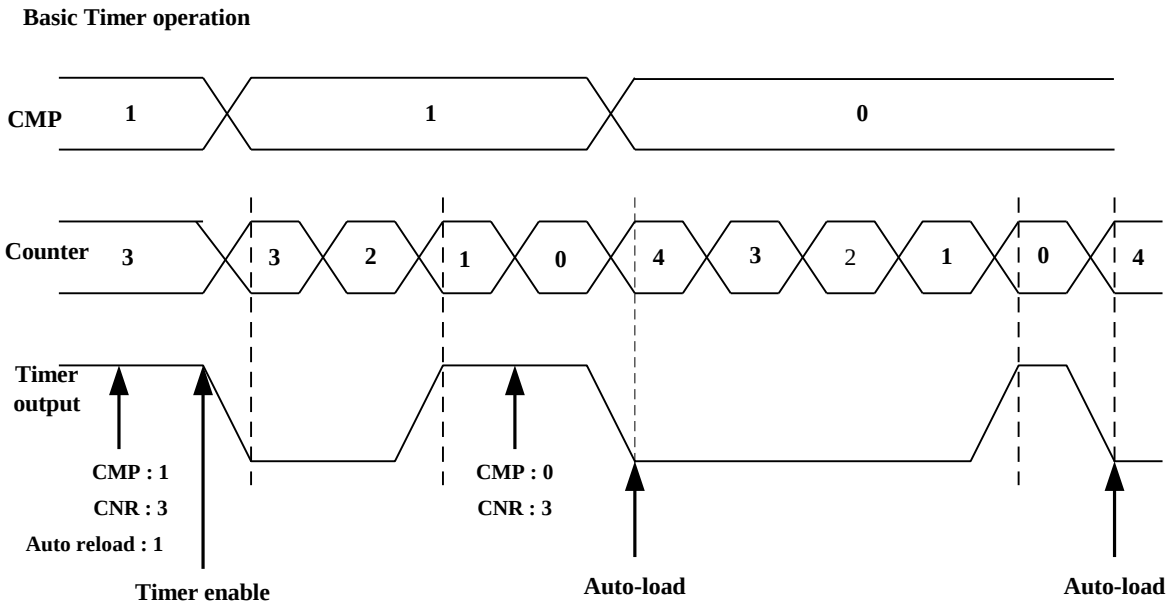
(timer output GPIO pin select by PAD Control register(PAD_REG0))

The following figure describes the architecture of PWM in one group. (Timer 0&1 are in one group and timer 2&3 are in another group)




DZI0
Dead Zone Generator
pwm_clk
Dead Zone
CNR2
CMR2
PWM2
Control Logic
8-bit Pre-scale CP1
1
1/2
1/4
1/8
1/16
Dead Zone
CNR3
CMR3
PWM3
Control Logic

6.13.5 Basic Timer Operation



Basic Timer Operation Timing

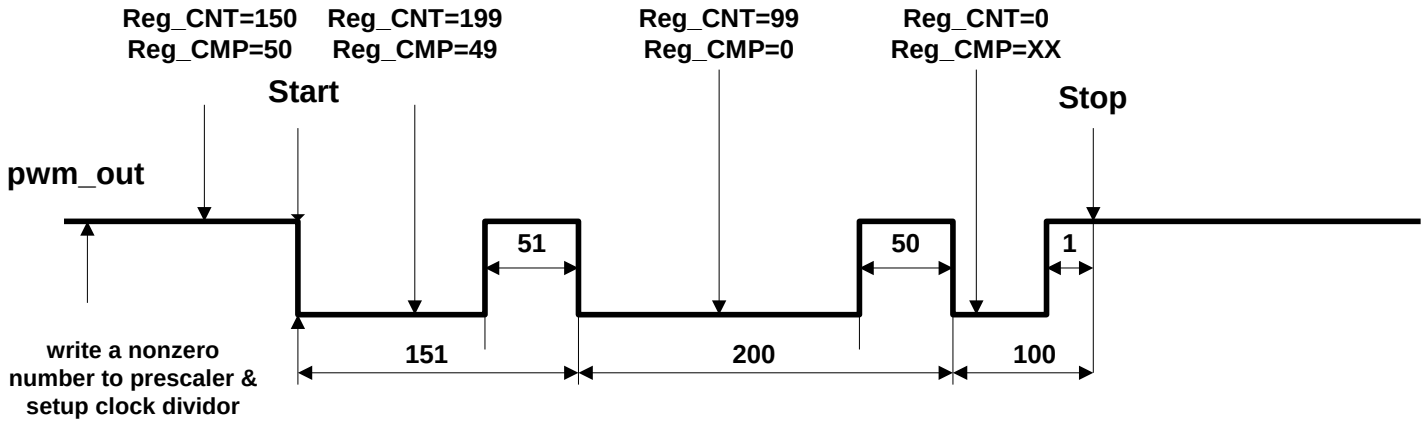
6.13.6 PWM Double Buffering and Automatic Reload

PWM-Timers have a double buffering function, enabling the reload value changed for next timer operation without stopping current timer operation. Although new timer value is set, current timer operation still operate successfully.

The counter value can be written into CNR0~3 and current counter value can be read from PDR0~3.

The auto-reload operation will copy from CNR0~3 to down-counter when down-counter reaches zero. If CNR0~3 are set as zero, counter will be halt when counter count to zero. If auto-reload bit is set as zero, counter will be stopped immediately.

PWM double buffering

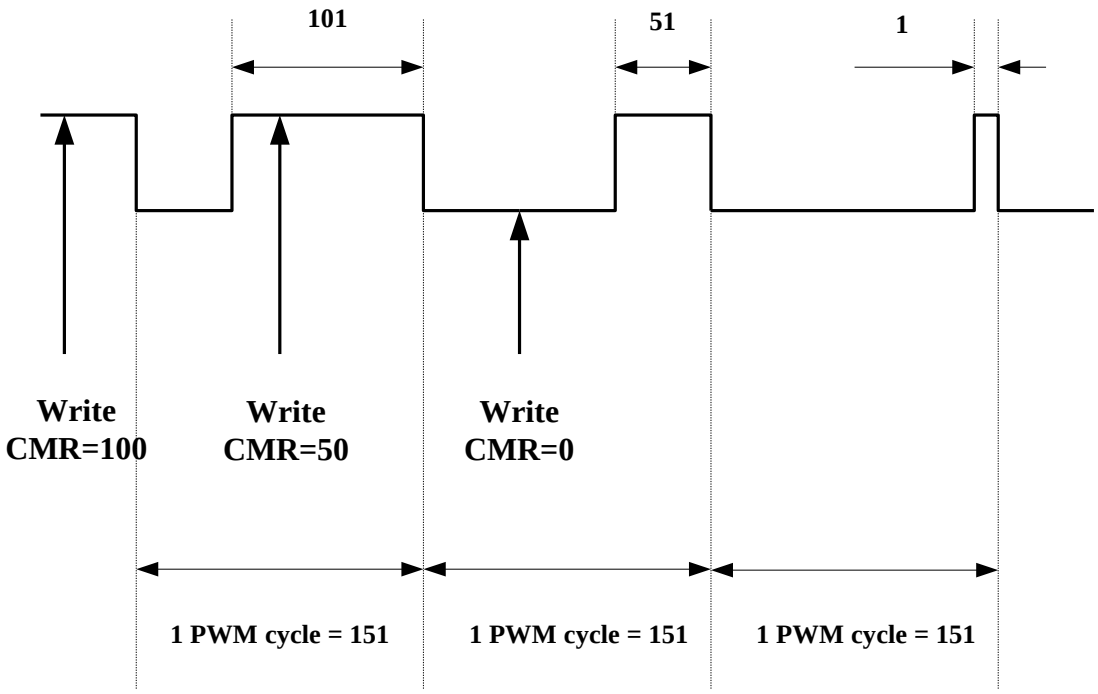


PWM Double Buffering Illustration

6.13.7 Modulate Duty Ratio

The double buffering function allows CMR written at any point in current cycle. The loaded value will take effect from next cycle.

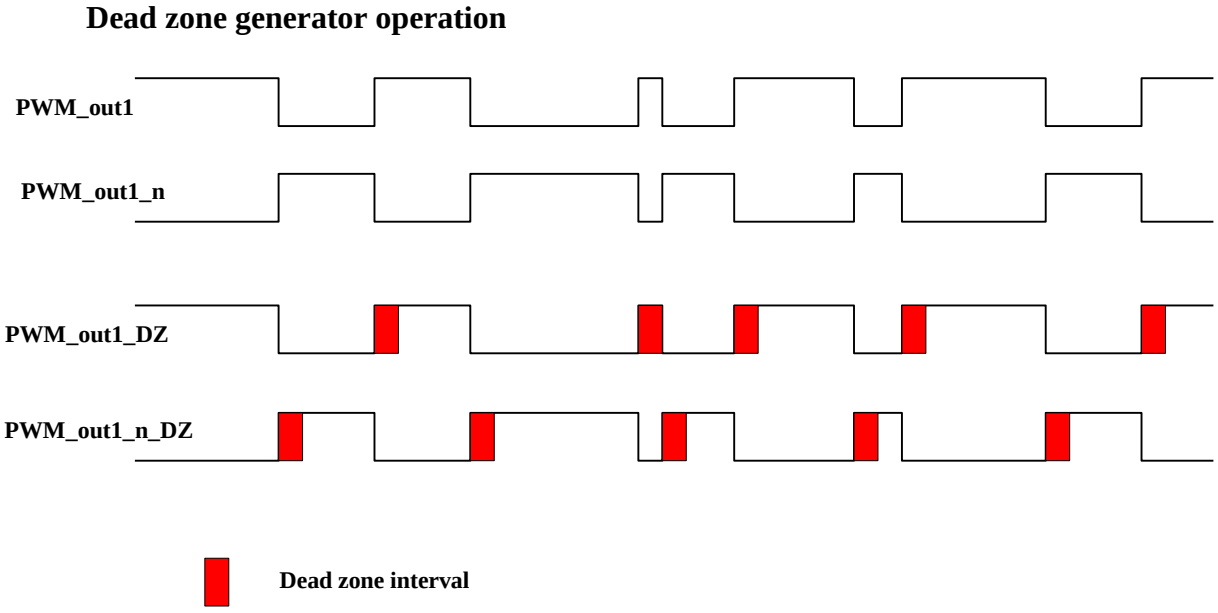
Modulate PWM controller ouput duty ratio(CNR = 150)



PWM Controller Output Duty Ratio

6.13.8 Dead-Zone Generator

PWM is implemented with Dead Zone generator. They are built for power device protection. This function enables generation of a programmable time gap at the rising of PWM output waveform. User can program PPR [31:24] and PPR [23:16] to determine the two Dead Zone interval respectively.



Dead Zone Generation Operation

6.13.9 PWM Timer Start Procedure

1. Setup clock selector (CSR)
2. Setup prescale & dead zone interval (PPR)
3. Setup inverter on/off, dead zone generator on/off, toggle mode /one-shot mode, and PWM timer off. (PCR)
4. Setup the comparator register (CMR)
5. Setup the counter register (CNR)
6. Setup the interrupt enable register (PIER)
7. Setup PWM output enables (POE)
8. Enable PWM timer (PCR)

6.13.10 PWM Timer Stop Procedure

Method 1:

Set 16-bit down counter (CNR) as 0, and monitor PDR. When PDR reaches to 0, disable PWM timer (PCR). ***(Recommended)***

Method 2:

Set 16-bit down counter (CNR) as 0. When interrupt request happen, disable PWM timer (PCR). ***(Recommended)***

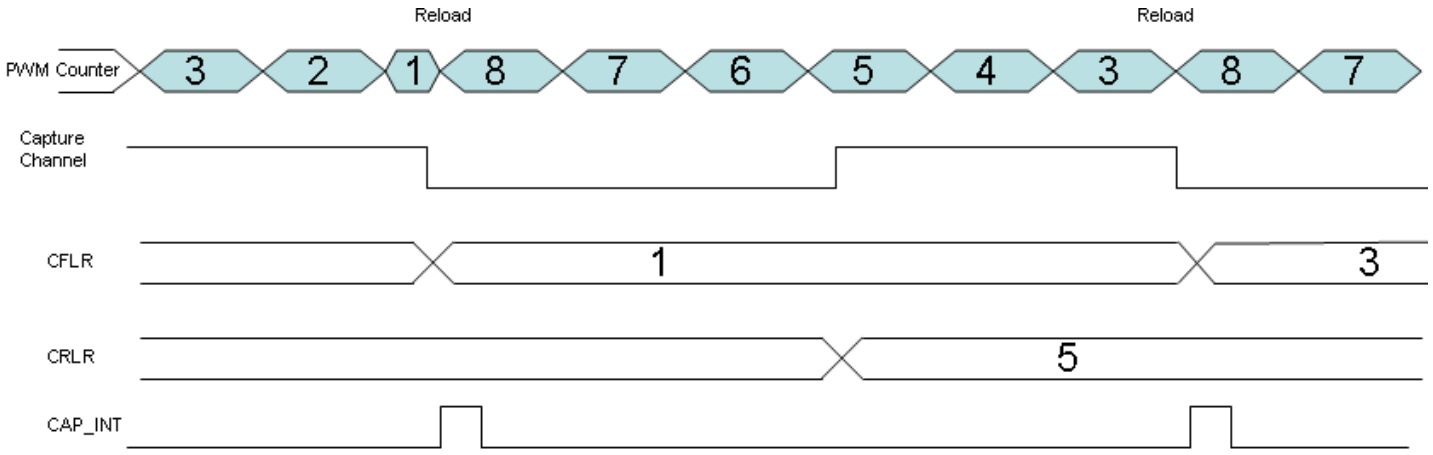
Method 3:

Disable PWM timer directly (PCR). ***(Not recommended)***

6.13.10.1 Capture Start Procedure

1. Setup clock selector (CSR)
2. Setup pre-scale & dead zone interval (PPR)
3. Setup inverter on/off, dead zone generator on/off, toggle mode /one-shot mode, and PWM timer off. (PCR)
4. Setup the comparator register (CMR)
5. Setup the counter register (CNR)
6. Setup the capture register (CCR)
7. Setup PWM output enables (POE)
8. Enable PWM timer (PCR)

6.13.10.2 Capture Basic Timer Operation



At this case, the CNR is 8:

1. When set falling interrupt enable, the PWM counter will be reload at time of interrupt occur.
2. The channel low pulse width is (CNT – CRLR).
3. The channel high pulse width is (CRLR - CFLR).
4. The channel cycle time is (CNR – CFLR).

6.13.11 PWM Timer Register Mapping

R: read only, **W**: write only, **R/W**: both read and write, **C**: Only value 0 can be written

Register	Address	R/W	Description	Reset Value
PWM_BA = 0xB800_7000				
PPR	PWM_BA+0x000	R/W	PWM Pre-scale Register	0x0000_0000
CSR	PWM_BA+0x004	R/W	PWM Clock Select Register	0x0000_0000
PCR	PWM_BA+0x008	R/W	PWM Control Register	0x0000_0000
CNR0	PWM_BA+0x00C	R/W	PWM Counter Register 0	0x0000_0000
CMR0	PWM_BA+0x010	R/W	PWM Comparator Register 0	0x0000_0000
PDR0	PWM_BA+0x014	R	PWM Data Register 0	0x0000_0000
CNR1	PWM_BA+0x018	R/W	PWM Counter Register 1	0x0000_0000
CMR1	PWM_BA+0x01C	R/W	PWM Comparator Register 1	0x0000_0000
PDR1	PWM_BA+0x020	R	PWM Data Register 1	0x0000_0000
CNR2	PWM_BA+0x024	R/W	PWM Counter Register 2	0x0000_0000
CMR2	PWM_BA+0x028	R/W	PWM Comparator Register 2	0x0000_0000
PDR2	PWM_BA+0x02C	R	PWM Data Register 2	0x0000_0000
CNR3	PWM_BA+0x030	R/W	PWM Counter Register 3	0x0000_0000
CMR3	PWM_BA+0x034	R/W	PWM Comparator Register 3	0x0000_0000
PDR3	PWM_BA+0x038	R	PWM Data Register 3	0x0000_0000
PIER	PWM_BA+0x040	R/W	PWM Interrupt Enable Register	0x0000_0000
PIIR	PWM_BA+0x044	R/C	PWM Interrupt Indication Register	0x0000_0000
CCR0	PWM_BA+0x050	R/W	Capture Control Register 0	0x0000_0000
CCR1	PWM_BA+0x054	R/W	Capture Control Register 1	0x0000_0000
CRLR0	PWM_BA+0x058	R/W	Capture Rising Latch Register (Channel 0)	0x0000_0000
CFLR0	PWM_BA+0x05C	R/W	Capture Falling Latch Register (Channel 0)	0x0000_0000
CRLR1	PWM_BA+0x060	R/W	Capture Rising Latch Register (Channel 1)	0x0000_0000
CFLR1	PWM_BA+0x064	R/W	Capture Falling Latch Register (Channel 1)	0x0000_0000
CRLR2	PWM_BA+0x068	R/W	Capture Rising Latch Register (Channel 2)	0x0000_0000
CFLR2	PWM_BA+0x06C	R/W	Capture Falling Latch Register (Channel 2)	0x0000_0000
CRLR3	PWM_BA+0x070	R/W	Capture Rising Latch Register (Channel 3)	0x0000_0000
CFLR3	PWM_BA+0x074	R/W	Capture Falling Latch Register (Channel 3)	0x0000_0000
CAPENR	PWM_BA+0x078	R/W	Capture Input Enable Register	0x0000_0000
POE	PWM_BA+0x07C	R/W	PWM Output Enable	0x0000_0000

6.14 Register Description

PWM Pre-Scale Register (PPR)

Register	Offset	R/W	Description	Reset Value
PPR	PWM_BA+0x000	R/W	PWM Pre-scale Register	0x0000_0000

31	30	29	28	27	26	25	24
DZI1							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
DZI0							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
CP1							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CP0							

Bits	Descriptions	
[31:24]	**DZI11**	**Dead zone interval register 1** These 8-bit determine dead zone length. The 1 unit time of dead zone length is received from clock selector 1.
[23:16]	**DZI0**	**Dead zone interval register 0** These 8-bit determine dead zone length. The 1 unit time of dead zone length is received from clock selector 0.
[15:8]	**CP1**	**Clock pre-scale 1 for PWM Timer 2 & 3** Clock input is divided by (CP1 + 1) before it is fed to the counter. 2 & 3 If CP1=0, then the pre-scale 1 output clock will be stopped.
[7:0]	**CP0**	**Clock pre-scale 0 for PWM Timer 0 & 1** Clock input is divided by (CP0 + 1) before it is fed to the counter. 0 & 1 If CP0=0, then the pre-scale 0 output clock will be stopped.

PWM Clock Selector Register (CSR)

Register	Offset	R/W	Description	Reset Value
CSR	PWM_BA+0x004	R/W	PWM Clock Selector Register (CSR)	0x0000_0000

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	**CSR3**			**Reserved**	**CSR2**		
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved	**CSR1**			**Reserved**	**CSR0**		

Bits	Descriptions	
[31:15]	**Reserved**	**Reserved**
[14:12]	**CSR3**	**Timer 3 Clock Source Selection** Select clock input for timer 3. CSR3 [14:12] / Input clock divided by: 100 – 1 011 – 16 010 – 8 001 – 4 000 – 2
[11]	**Reserved**	**Reserved**
[10:8]	**CSR2**	**Timer 2 Clock Source Selection** Select clock input for timer 0. (Table is the same as CSR3)
[7]	**Reserved**	**Reserved**
[6:4]	**CSR1**	**Timer 1 Clock Source Selection** Select clock input for timer 0. (Table is the same as CSR3)
[3]	**Reserved**	**Reserved**
[2:0]	**CSR0**	**Timer 0 Clock Source Selection** Select clock input for timer 0. (Table is the same as CSR3)

PWM Control Register (PCR)

Register	Offset	R/W	Description	Reset Value
PCR	PWM_BA+0x008	R/W	PWM Control Register (PCR)	0x0000_0000

31	30	29	28	27	26	25	24
Reserved				**CH3MOD**	**CH3INV**	**Reserved**	**CH3EN**
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved				**CH2MOD**	**CH2INV**	**Reserved**	**CH2EN**
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved				**CH1MOD**	**CH1INV**	**Reserved**	**CH1EN**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**DZEN1**	**DZEN0**	**CH0MOD**	**CH0INV**	**Reserved**	**CH0EN**

Bits	Descriptions	
[31:28]	**Reserved**	**Reserved**
[27]	**CH3MOD**	**Timer 3 Toggle/One-Shot Mode** 1: Toggle Mode 0: One-Shot Mode NOTE: If there is a rising transition at this bit, it will cause CNR3 and CMR3 be clear.
[26]	**CH3INV**	**Timer 3 Inverter ON/OFF** 1: Inverter ON 0: Inverter OFF
[25]	**Reserved**	**Reserved**
[24]	**CH3EN**	**Timer 3 Enable/Disable** 1: Enable 0: Disable
[23:20]	**Reserved**	**Reserved**
[19]	**CH2MOD**	**Timer 2 Toggle/One-Shot Mode** 1: Toggle Mode 0: One-Shot Mode NOTE: If there is a rising transition at this bit, it will cause CNR2 and CMR2 be clear.
[18]	**CH2INV**	**Timer 2 Inverter ON/OFF** 1: Inverter ON 0: Inverter OFF
[17]	**Reserved**	**Reserved**
[16]	**CH2EN**	**Timer2 Enable/Disable**

		1: Enable 0: Disable
[15:10]	**Reserved**	**Reserved**
[11]	**CH1MOD**	**Timer 1 Toggle/One-Shot Mode** 1: Toggle Mode 0: One-Shot Mode NOTE: If there is a rising transition at this bit, it will cause CNR1 and CMR1 be clear.
[10]	**CH1INV**	**Timer 1 Inverter ON/OFF** 1: Inverter ON 0: Inverter OFF
[9]	**Reserved**	**Reserved**
[8]	**CH1EN**	**Timer 1 Enable/Disable** 1: Enable 0: Disable
[7:6]	**Reserved**	**Reserved**
[5]	**DZEN1**	**Dead-Zone 1 Generator Enable/Disable** 1: Enable 0: Disable
[4]	**DZEN0**	**Dead-Zone 0 Generator Enable/Disable** 1: Enable 0: Disable
[3]	**CH0MOD**	**Timer 0 Toggle/One-Shot Mode** 1: Toggle Mode 0: One-Shot Mode NOTE: If there is a rising transition at this bit, it will cause CNR0 and CMR0 be clear.
[2]	**CH0INV**	**Timer 0 Inverter ON/OFF** 1: Inverter ON 0: Inverter OFF
[1]	**Reserved**	**Reserved**
[0]	**CH0EN**	**Timer 0 Enable/Disable** 1: Enable 0: Disable

PWM Counter Register 3-0 (CNR3-0)

Register	Offset	R/W	Description	Reset Value
CNR0	PWM_BA+0x00C	R/W	PWM Counter Register 0	0x0000_0000
CNR1	PWM_BA+0x018	R/W	PWM Counter Register 1	0x0000_0000
CNR2	PWM_BA+0x024	R/W	PWM Counter Register 2	0x0000_0000
CNR3	PWM_BA+0x030	R/W	PWM Counter Register 3	0x0000_0000

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**
CNR [15:8]							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CNR [7:0]							

Bits	Descriptions	
[31:16]	**Reserved**	**Reserved**
[15:0]	**CNR**	**PWM Counter/Timer Loaded Value** Inserted data range : 65535~0 (Unit : 1 PWM clock cycle) **Note 1:** One PWM cycle width = CNR + 1. If CNR equal zero, PWM counter/timer will be stopped. **Note 2:** Programmer can feel free to write a data to CNR at any time, and it will take effect in next cycle.

PWM Comparator Register 3-0 (CMR3-0)

Register	Offset	R/W	Description	Reset Value
CMR0	PWM_BA+0x010	R/W	PWM Comparator Register 0	0x0000_0000
CMR1	PWM_BA+0x01C	R/W	PWM Comparator Register 1	0x0000_0000
CMR2	PWM_BA+0x028	R/W	PWM Comparator Register 2	0x0000_0000
CMR3	PWM_BA+0x034	R/W	PWM Comparator Register 3	0x0000_0000

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**
CMR [15:8]							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CMR [7:0]							

Bits	Descriptions	
[31:16]	**Reserved**	**Reserved**
[15:0]	**CMR**	**PWM Comparator Register** Inserted data range : 65535~0 (Unit : 1 PWM clock cycle) CMR are used to determine PWM output duty ratio. Assumption : PWM output initial : high CMR >= CNR : PWM output is always high CMR < CNR : PWM output high => (CMR + 1) unit CMR = 0 : PWM output high => 1 unit **Note 1:** PWM duty = CMR + 1. If CMR equal zero, PWM duty = 1 **Note 2:** Programmer can feel free to write a data to CMR at any time, and it will take effect in next cycle.

PWM Data Register 3-0 (PDR 3-0)

Register	Offset	R/W	Description	Reset Value
PDR0	PWM_BA+0x014	R	PWM Data Register 0	0x0000_0000
PDR1	PWM_BA+0x020	R	PWM Data Register 1	0x0000_0000
PDR2	PWM_BA+0x02C	R	PWM Data Register 1	0x0000_0000
PDR3	PWM_BA+0x038	R	PWM Data Register 1	0x0000_0000

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**
PDR [15:8]							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PDR [7:0]							

Bits	Descriptions	
[31:16]	**Reserved**	**Reserved**
[15:0]	**PDR**	**PWM Data Register** User can monitor PDR to know current value in 16-bit down counter.

PWM Interrupt Enable Register (PIER)

Register	Offset	R/W	Description	Reset Value
PIER	PWM_BA+0x040	R/W	PWM Interrupt Enable Register	0x0000_0000

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							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved				**PIER3**	**PIER2**	**PIER1**	**PIER0**

Bits	Descriptions	
[31:4]	**Reserved**	**Reserved**
[3]	**PIER3**	**PWM Timer 3 Interrupt Enable** 1: Enable 0: Disable
[2]	**PIER2**	**PWM Timer 2 Interrupt Enable** 1: Enable 0: Disable
[1]	**PIER1**	**PWM Timer 1 Interrupt Enable** 1: Enable 0: Disable
[0]	**PIER0**	**PWM Timer 0 Interrupt Enable** 1: Enable 0: Disable

PWM Interrupt Indication Register (PIIR)

Register	Offset	R/W	Description	Reset Value
PIIR	PWM_BA+0x044	R/W	PWM Interrupt Indication Register	0x0000_0000

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							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved				**PIIR3**	**PIIR2**	**PIIR1**	**PIIR0**

Bits	Descriptions	
[31:4]	**Reserved**	**Reserved**
[3]	**PIIR3**	**PWM Timer 3 Interrupt Flag** 1: Interrupt Flag ON 0: Interrupt Flag OFF
[2]	**PIIR2**	**PWM Timer 2 Interrupt Flag** 1: Interrupt Flag ON 0: Interrupt Flag OFF
[1]	**PIIR1**	**PWM Timer 1 Interrupt Flag** 1: Interrupt Flag ON 0: Interrupt Flag OFF
[0]	**PIIR0**	**PWM Timer 0 Interrupt Flag** 1: Interrupt Flag ON 0: Interrupt Flag OFF

Note: User can clear each interrupt flag by writing a zero to corresponding bit in PIIR.

Capture Control Register (CCR0)

Register	Offset	R/W	Description	Reset Value
CCR0	PWM_BA+0x050	R/W	Capture Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
CFLRD1	**CRLRD1**	**Reserved**	**CIIR1**	**CAPCH1EN**	**FL&IE1**	**RL&IE1**	**INV1**
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CFLRD0	**CRLRD0**	**Reserved**	**CIIR0**	**CAPCH0EN**	**FL&IE0**	**RL&IE0**	**INV0**

Bits	Descriptions	
[31:24]	**Reserved**	**Reserved**
[23]	**CFLRD1**	**CFLR1 dirty bit** When input channel 1 has a rising transition, CFLR1 was updated and this bit was "1".
[22]	**CRLRD1**	**CRLR1 dirty bit** When input channel 1 has a falling transition, CRLR1 was updated and this bit was "1".
[21]	**Reserved**	**Reserved**
[19]	**CAPCH1EN**	**Capture Channel 1 transition Enable/Disable** 1: Enable 0: Disable When Enable, Capture latched the PMW-counter and saved to CRLR (Rising latch) and CFLR (Falling latch). When Disable, Capture does not update CRLR and CFLR, and disable Channel 1 Interrupt.
[18]	**FL&IE1**	**Channel1 Falling Interrupt Enable ON/OFF** 1: Enable 0: Disable When Enable, if Capture detects Channel 1 has falling transition, Capture issues an Interrupt.
[17]	**RL&IE1**	**Channel 1 Rising Interrupt Enable ON/OFF** 1: Enable 0: Disable When Enable, if Capture detects Channel 1 has rising transition, Capture

		issues an Interrupt.
[16]	**INV1**	**Channel 1 Inverter ON/OFF** 1: Inverter ON 0: Inverter OFF
[15:7]	**Reserved**	**Reserved**
[6]	**CRLRD0**	**CRLR0 dirty bit** When input channel 0 has a falling transition, CRLR0 was updated and this bit was "1".
[5:4]	**Reserved**	**Reserved**
[3]	**CAPCH0EN**	**Capture Channel 0 transition Enable/Disable** 1: Enable 0: Disable When Enable, Capture latched the PWM-counter value and saved to CRLR (Rising latch) and CFLR (Falling latch). When Disable, Capture does not update CRLR and CFLR, and disable Channel 0 Interrupt.
[2]	**FL&IE0**	**Channel 0 Falling Interrupt Enable ON/OFF** 1: Enable 0: Disable When Enable, if Capture detects Channel 0 has falling transition, Capture issues an Interrupt.
[1]	**RL&IE0**	**Channel 0 Rising Interrupt Enable ON/OFF** 1: Enable 0: Disable When Enable, if Capture detects Channel 0 has rising transition, Capture issues an Interrupt.
[0]	**INV0**	**Channel 0 Inverter ON/OFF** 1: Inverter ON 0: Inverter OFF

Capture Control Register (CCR1)

Register	Offset	R/W	Description	Reset Value
CCR1	PWM_BA+0x054	R/W	Capture Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved	**CRLRD3**	**Reserved**	**CIIR3**	**CAPCH3EN**	**FL&IE3**	**RL&IE3**	**INV3**
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
CFLRD2	**CRLRD2**	**Reserved**	**CIIR2**	**CAPCH2EN**	**FL&IE2**	**RL&IE2**	**INV2**

Bits	Descriptions	
[31:23]	**Reserved**	**Reserved**
[22]	**CRLRD3**	**CRLR3 dirty bit** When input channel 1 has a falling transition, CRLR3 was updated and this bit was "1".
[21]	**Reserved**	**Reserved**
[20]	**CIIR3**	**Capture Interrupt Indication 3 Enable/Disable** 1: Interrupt Flag ON 0: Interrupt Flag OFF Note: If this bit is "1", PWM-counter 3 will not reload when next capture interrupt occur.
[19]	**CAPCH3EN**	**Capture Channel 3 transition Enable/Disable** 1: Enable 0: Disable When Enable, Capture latched the PMW-counter and saved to CRLR (Rising latch) and CFLR (Falling latch). When Disable, Capture does not update CRLR and CFLR, and disable Channel 3 Interrupt.
[18]	**FL&IE3**	**Channel 3 Falling Interrupt Enable ON/OFF** 1: Enable 0: Disable When Enable, if Capture detects Channel 3 has falling transition, Capture issues an Interrupt.
[17]	**RL&IE3**	**Channel 3 Rising Interrupt Enable ON/OFF**

		1: Enable 0: Disable When Enable, if Capture detects Channel 3 has rising transition, Capture issues an Interrupt.
[16]	**INV3**	**Channel 3 Inverter ON/OFF** 1: Inverter ON 0: Inverter OFF
[15:8]	**Reserved**	**Reserved**
[7]	**CFLRD2**	**CFLR2 dirty bit** When input channel 2 has a rising transition, CFLR2 was updated and this bit was "1".
[6]	**CRLRD2**	**CRLR2 dirty bit** When input channel 2 has a falling transition, CRLR2 was updated and this bit was "1".
[5]	**Reserved**	**Reserved**
[4]	**CIIR2**	**Capture Interrupt Indication 2 Enable/Disable** 1: Interrupt Flag ON 0: Interrupt Flag OFF Note: If this bit is "1", PWM-counter 2 will not reload when next capture interrupt occur.
[3]	**CAPCH2EN**	**Capture Channel 2 transition Enable/Disable** 1: Enable 0: Disable When Enable, Capture latched the PMW-counter value and saved to CRLR (Rising latch) and CFLR (Falling latch). When Disable, Capture does not update CRLR and CFLR, and disable Channel 2 Interrupt.
[2]	**FL&IE2**	**Channel 2 Falling Interrupt Enable ON/OFF** 1: Enable 0: Disable When Enable, if Capture detects Channel 2 has falling transition, Capture issues an Interrupt.
[1]	**RL&IE2**	**Channel 2 Rising Interrupt Enable ON/OFF** 1: Enable 0: Disable When Enable, if Capture detects Channel 2 has rising transition, Capture issues an Interrupt.
[0]	**INV20**	**Channel 2 Inverter ON/OFF** 1: Inverter ON 0: Inverter OFF

Capture Rising Latch Register3-0 (CRLR3-0)

Register	Offset	R/W	Description	Reset Value
CRLR0	PWM_BA+0x058	R/W	Capture Rising Latch Register (channel 0)	0x0000_0000
CRLR1	PWM_BA+0x060	R/W	Capture Rising Latch Register (channel 1)	0x0000_0000
CRLR2	PWM_BA+0x068	R/W	Capture Rising Latch Register (channel 2)	0x0000_0000
CRLR3	PWM_BA+0x070	R/W	Capture Rising Latch Register (channel 3)	0x0000_0000

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**
CRLR0 [15:8]							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CRLR0 [7:0]							

Bits	Descriptions	
[31:16]	**Reserved**	**Reserved**
[15:0]	**CRLR0**	Capture Rising Latch Register0 Latch the PWM counter when Channel 0 has rising transition.

Capture Falling Latch Register3-0 (CFLR3-0)

Register	Offset	R/W	Description	Reset Value
CFLR0	PWM_BA+0x05C	R/W	Capture Falling Latch Register (channel 0)	0x0000_0000
CFLR1	PWM_BA+0x064	R/W	Capture Falling Latch Register (channel 1)	0x0000_0000
CFLR2	PWM_BA+0x06C	R/W	Capture Falling Latch Register (channel 2)	0x0000_0000
CFLR3	PWM_BA+0x074	R/W	Capture Falling Latch Register (channel 3)	0x0000_0000

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**
CFLR0[15:8]							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CFLR0[7:0]							

Bits	Descriptions	
[31:16]	**Reserved**	**Reserved**
[15:0]	**CFLR0**	**Capture Falling Latch Register0** Latch the PWM counter when Channel 0 has Falling transition.

Capture Input Enable Register (CAPENR)

Register	Offset	R/W	Description	Reset Value
CAPENR	PWM_BA+0x078	R/W	Capture Input Enable Register	0x0000_0000

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							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved				**CAPENR[3:0]**			

Bits	Descriptions	
[31:4]	**Reserved**	**Reserved**
[3:0]	**CAPENR**	**Capture Input Enable Register** There are eight capture inputs from pad. Bit0~Bit3 are used to control each inputs ON or OFF. (At most 4 inputs can be used at the same time) 0 : OFF / 1 : ON CAPENR[3:0] 3210 xxx1 ➔ Capture channel 0 is from GPA_DIN[12] or GPB_DIN[1] or GPB_DIN[8] or GPC_DIN[3] or GPC_DIN[7] xx1x ➔ Capture channel 1 is from GPA_DIN[13] or GPB_DIN[2] or GPB_DIN[9] or GPC_DIN[4] or GPC_DIN[8] x1xx ➔ Capture channel 2 is from GPA_DIN[15] or GPB_DIN[3] or GPB_DIN[6] or GPC_DIN[5] or GPC_DIN[9] 1xxx ➔ Capture channel 3 is from GPB_DIN[0] or GPB_DIN[4] or GPB_DIN[7] or GPC_DIN[6] or GPC_DIN[10]

PWM Output Enable Register (PWM)

Register	Offset	R/W	Description	Reset Value
POE	PWM_BA+0x07C	R/W	PWM Output Enable Register	0x0000_0000

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							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved				**PWM3**	**PWM2**	**PWM1**	**PWM0**

Bits	Descriptions	
[31:4]	**Reserved**	**Reserved**
[3]	**PWM3**	**PWM timer 3 Output Enable Setup.** 1 : Enable 0 : Disable
[2]	**PWM2**	**PWM timer 2 Output Enable Setup.** 1 : Enable 0 : Disable
[1]	**PWM1**	**PWM timer 1 Output Enable Setup.** 1 : Enable 0 : Disable
[0]	**PWM0**	**PWM timer 0 Output Enable Setup.** 1 : Enable 0 : Disable

6.15 Real Time Clock (RTC)

6.15.1 Overview

Real Time Clock (RTC) block can be operated by independent power supply while the system power is off. The RTC uses a 32.768 KHz external crystal. A built in RTC is designed to generate the periodic interrupt signal. The period can be 0.25/ 0.5/ 1/ 2/ 4/ 8 second. There is RTC overflow counter and it can be adjusted by software.

6.15.2 RTC Features

- There is a time counter (second, minute, hour) and calendar counter (day, month, year) for user to check the time.
- Alarm register (second, minute, hour, day, month, year)
- 12-hour or 24-hour mode is selectable
- Recognize leap year automatically
- The day of week counter
- Frequency compensate register (FCR)
- Beside FCR, all clock and alarm data expressed in BCD code
- Support time tick interrupt
- Support wake up function.

6.15.3 RTC Function Description

RTC Initiation

When RTC block is power on, programmer has to write a number (0xa5eb1357) to INIR to reset all logic. INIR act as hardware reset circuit. Once INIR has been set as 0xa5eb1357, there is no action for RTC if any value be programmed into INIR register.

RTC Read/Write Enable

Register AER bit 15~0 is served as RTC read/write password. It is used to avoid signal interference from system during system power off. AER bit 15~0 has to be set as 0xa965 after system power on. Once it is set, it will take effect 512 RTC clocks later (about 15ms). Programmer can read AER bit 16 to find out whether RTC register can be accessed.

Frequency Compensation

The RTC FCR allows software control digital compensation of a 32.768 KHz crystal oscillator. User can utilize a frequency counter to measure RTC clock in one of GPIO pin during manufacture, and store the value in Flash memory for retrieval when the product is first power on.

Time and Calendar counter

TLR and CLR are used to load the time and calendar. TAR and CAR are used for alarm. They are all represented by BCD.

12/24 hour Time Scale Selection

The 12/24 hour time scale selection depends on TSSR bit 0.

Day of the week counter

Count from Sunday to Saturday.

Time tick interrupt

RTC block use a counter to calibrate the time tick count value. When the value in counter reaches zero, RTC will issue an interrupt.

RTC register property

When system power is off but RTC power is on, data stored in RTC registers will not lost except RIER and RIIR. Because of clock difference between RTC clock and system clock, when user write new data to any one of the registers, the register will not be updated until 2 RTC clocks later (60us). Hence programmer should consider about access sequence between TSSR, TAR and TLR.

In addition, user must be aware that RTC block does not check whether loaded data is out of bounds or not. RTC does not check rationality between DWR and CLR either.

Note:

1. TAR, CAR, TLR and CLR registers are all BCD counter.

2. Programmer has to make sure that the loaded values are reasonable,
For example, Load CLR as 201a (year), 13 (month), 00 (day), or CLR does not match with DWR, etc.
3. Reset state :

Register	Reset State
AER	0(RTC read/write disable)
CLR	05, 1, 1 (2005-1-1)
TLR	00 hr: 00 min: 00 sec
CAR	00/00/00
TAR	00:00:00
TSSR	1 (24 hr mode)
DWR	6 (Saturday)
RIER	0
RIIR	0
LIR	0
TTR	0

4. FCR Calibration :

Example 1, Frequency counter measurement : 32773.65Hz (> 32768 Hz)
Integer part : 32773 => 0x8005
FCR_int = 0x05 – 0x01 + 0x08 = 0x0c
Fraction part : 0.65 x 60 = 39 => 0x27
FCR_fra = 0x27

Example 2, Frequency counter measurement : 32765.27Hz (≦ 32768 Hz)
Integer part : 32765 => 0x7ffd
FCR_int = 0x0d – 0x01 – 0x08 = 0x04
Fraction part : 0.27 x 60 = 16.2=> 0x10
FCR_fra = 0x10

5. In TLR and TAR, only 2 BCD digits are used to express "year". We assume 2 BCD digits of xY denote 20xY, but not 19xY or 21xY.

6.15.4 RTC Register Mapping

Register	Address	R/W	Description	Reset Value
RTC_BA = 0xB800_8000				
INIR	RTC_BA+0x000	R/W	RTC Initiation Register	0x0000_0000
AER	RTC_BA+0x004	R/W	RTC Access Enable Register	0x0000_0000
FCR	RTC_BA+0x008	R/W	RTC Frequency Compensation Register	0x0000_0700
TLR	RTC_BA+0x00C	R/W	Time Loading Register	0x0000_0000
CLR	RTC_BA+0x010	R/W	Calendar Loading Register	0x0005_0101
TSSR	RTC_BA+0x014	R/W	Time Scale Selection Register	0x0000_0001
DWR	RTC_BA+0x018	R/W	Day of the Week Register	0x0000_0006
TAR	RTC_BA+0x01C	R/W	Time Alarm Register	0x0000_0000
CAR	RTC_BA+0x020	R/W	Calendar Alarm Register	0x0000_0000
LIR	RTC_BA+0x024	R	Leap year Indicator Register	0x0000_0000
RIER	RTC_BA+0x028	R/W	RTC Interrupt Enable Register	0x0000_0000
RIIR	RTC_BA+0x02C	R/C	RTC Interrupt Indicator Register	0x0000_0000
TTR	RTC_BA+0x030	R/W	RTC Time Tick Register	0x0000_0000

6.15.5 RTC Register Descriptions

RTC Initiation Register (INIR)

Register	Address	R/W/C	Description	Reset Value
INIR	RTC_BA+0x000	R/W	RTC Initiation Register	0x0000_0000

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

Bits	Descriptions	
[0]	**Active**	**RTC Active Status** (Read only), 0: RTC is at reset state 1: RTC is at normal active state.
[31:0]	**INIR**	**RTC Initiation** When RTC block is power on, RTC is at reset state; programmer has to write a number (0x a5eb1357) to INIR to release all of logic and counters. INIR act as hardware reset circuit.

RTC Access Enable Register (AER)

Register	Address	R/W/C	Description	Reset Value
AER	RTC_BA+0x004	R/W	RTC Access Enable Register	0x0000_0000

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

Bits	Descriptions	
[31:17]	**Reserved**	**Reserved**
[16]	**ENF**	**RTC Register Access Enable Flag** (Read only) 1: RTC register read/write enable 0: RTC register read/write disable This bit will be set after AER[15:0] register is load a 0xA965, and be clear automatically in a long time or AER[15:0] is not 0xA965.
[15:0]	**AER**	**RTC Register Access Enable Password** (Write only) 0xA965: access enable Others: access disable

RTC Frequency Compensation Register (FCR)

Register	Address	R/W/C	Description	Reset Value
FCR	RTC_BA+0x008	R/W	Frequency Compensation Register	0x0000_0700

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				**INTEGER**			
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**FRACTION**					

Bits	Descriptions	
[31:12]	**Reserved**	**Reserved**
[11:8]	**INTEGER**	**Integer Part** Integer part of detected value / FCR[11:8] / Integer part of detected value / FCR[11:8] 32776 / 1111 / 32768 / 0111 32775 / 1110 / 32767 / 0110 32774 / 1101 / 32766 / 0101 32773 / 1100 / 32765 / 0100 32772 / 1011 / 32764 / 0011 32771 / 1010 / 32763 / 0010 32770 / 1001 / 32762 / 0001 32769 / 1000 / 32761 / 0000
[7:6]	**Reserved**	**Reserved**
[5:0]	**FRACTION**	**Fraction Part** Formula = (fraction part of detected value) x 60 Note: Digit in FCR must be expressed as hexadecimal number.

RTC Time Loading Register (TLR)

Register	Address	R/W/C	Description	Reset Value
TLR	RTC_BA+0x00C	R/W	Time Loading Register	0x0000_0000

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

Bits	Descriptions	
[31:22]	**Reserved**	**Reserved**
[21:20]	**10HR**	10 Hour Time Digit
[19:16]	**1HR**	1 Hour Time Digit
[15]	**Reserved**	**Reserved**
[14:12]	**10MIN**	10 Min Time Digit
[11:8]	**1MIN**	1 Min Time Digit
[7]	**Reserved**	**Reserved**
[6:4]	**10SEC**	10 Sec Time Digit
[3:0]	**1SEC**	1 Sec Time Digit

Notes: TLR is a BCD digit counter and RTC will not check loaded data.

RTC Calendar Loading Register (CLR)

Register	Address	R/W/C	Description	Reset Value
CLR	RTC_BA+0x010	R/W	Calendar Loading Register	0x0005_0101

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

Bits	Descriptions	
[31:24]	**Reserved**	**Reserved**
[23:20]	**10YEAR**	10-Year Calendar Digit
[19:16]	**1YEAR**	1-Year Calendar Digit
[15:13]	**Reserved**	**Reserved**
[12]	**10MON**	10-Month Calendar Digit
[11:8]	**1MON**	1-Month Calendar Digit
[7:6]	**Reserved**	**Reserved**
[5:4]	**10DAY**	10-Day Calendar Digit
[3:0]	**1DAY**	1-Day Calendar Digit

Notes: CLR is a BCD digit counter and RTC will not check loaded data.

RTC Time Scale Selection Register (TSSR)

Register	Address	R/W/C	Description	Reset Value
TSSR	RTC_BA+0x014	R/W	Time Scale Selection Register	0x0000_0001

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							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							**24hr/12hr**

Bits	Descriptions	
[31:1]	**Reserved**	**Reserved**
[0]	**24hr/12hr**	**24-Hour / 12-Hour Mode Selection** It indicate that TLR and TAR are in 24-hour mode or 12-hour mode 1: select 24-hour time scale 0: select 12-hour time scale with AM and PM indication

24-hour time scale	12-hour time scale	24-hour time scale	12-hour time scale
00	12(AM12)	12	32(PM12)
01	01(AM01)	13	21(PM01)
02	02(AM02)	14	22(PM02)
03	03(AM03)	15	23(PM03)
04	04(AM04)	16	24(PM04)
05	05(AM05)	17	25(PM05)
06	06(AM06)	18	26(PM06)
07	07(AM07)	19	27(PM07)
08	08(AM08)	20	28(PM08)
09	09(AM09)	21	29(PM09)
10	10(AM10)	22	30(PM10)
11	11(AM11)	23	31(PM11)

RTC Day of the Week Register (DWR)

Register	Address	R/W/C	Description	Reset Value
DWR	RTC_BA+0x018	R/W	Day of the Week Register	0x0000_0006

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							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved					**DWR**		

Bits	Descriptions	
[31:3]	**Reserved**	**Reserved**
[2:0]	**DWR**	**Day of the Week Register** 0 Sunday 1 Monday 2 Tuesday 3 Wednesday 4 Thursday 5 Friday 6 Saturday

RTC Time Alarm Register (TAR)

Register	Address	R/W/C	Description	Reset Value
TAR	RTC_BA+0x01C	R/W	Time Alarm Register	0x0000_0000

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

Bits	Descriptions	
[31:22]	**Reserved**	**Reserved**
[21:20]	**10HR**	10 Hour Time Digit
[19:16]	**1HR**	1 Hour Time Digit
[15]	**Reserved**	**Reserved**
[14:12]	**10MIN**	10 Min Time Digit
[11:8]	**1MIN**	1 Min Time Digit
[7]	**Reserved**	**Reserved**
[6:4]	**10SEC**	10 Sec Time Digit
[3:0]	**1SEC**	1 Sec Time Digit

Notes: TAR is a BCD digit counter and RTC will not check loaded data.

RTC Calendar Alarm Register (CAR)

Register	Address	R/W/C	Description	Reset Value
CAR	RTC_BA+0x020	R/W	Calendar Alarm Register	0x0000_0000

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

Bits	Descriptions	
[31:24]	**Reserved**	**Reserved**
[23:20]	**10YEAR**	10-Year Calendar Digit
[19:16]	**1YEAR**	1-Year Calendar Digit
[15:13]	**Reserved**	**Reserved**
[12]	**10MON**	10-Month Calendar Digit
[11:8]	**1MON**	1-Month Calendar Digit
[7:6]	**Reserved**	**Reserved**
[5:4]	**10DAY**	10-Day Calendar Digit
[3:0]	**1DAY**	1-Day Calendar Digit

Notes: CAR is a BCD digit counter and RTC will not check loaded data.

RTC Leap year Indication Register (LIR)

Register	Address	R/W/C	Description	Reset Value
LIR	RTC_BA+0x024	R	RTC Leap year Indication Register	0x0000_0000

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							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							**LIR**

Bits	Descriptions	
[31:1]	**Reserved**	**Reserved**
[0]	**LIR**	**Leap Year Indication REGISTER** (Real only). 1 : It indicate that this year is leap year 0 : It indicate that this year is not a leap year

RTC Interrupt Enable Register (RIER)

Register	Address	R/W/C	Description	Reset Value
RIER	RTC_BA+0x028	R/W	RTC Interrupt Enable Register	0x0000_0000

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							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved						**TIER**	**AIER**

Bits	Descriptions	
[31:2]	**Reserved**	**Reserved**
[1]	**TIER**	**Time Tick Interrupt Enable** 1 => RTC Time Tick Interrupt and counter enable 0 => RTC Time Tick Interrupt and counter disable
[0]	**AIER**	**Alarm Interrupt Enable** 1 => RTC Alarm Interrupt enable 0 => RTC Alarm Interrupt disable

RTC Interrupt Indication Register (RIIR)

Register	Address	R/W/C	Description	Reset Value
RIIR	RTC_BA+0x02C	R/C	RTC Interrupt Indication Register	0x0000_0000

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							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved						**TI**	**AI**

Bits	Descriptions	
[31:2]	**Reserved**	**Reserved**
[1]	**TI**	**RTC Time Tick Interrupt Indication** 1: It indicates that time tick interrupt has been activated. 0: It indicates that time tick interrupt never occurred. Software can also clear this bit after RTC interrupt has occur.
[0]	**AI**	**RTC Alarm Interrupt Indication** 1: It indicates that time counter and calendar counter have counted to a specified time recorded in TAR and CAR. RTC alarm interrupt has been activated. 0: It indicates that alarm interrupt never occurred. Software can also clear this bit after RTC interrupt has occurred.

RTC Time Tick Register (TTR)

Register	Address	R/W/C	Description	Reset Value
TTR	RTC_BA+0x030	R/C	RTC Time Tick Register	0x0000_0000

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							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved					**TTR[2:0]**		

Bits	Descriptions	
[31:3]	**Reserved**	**Reserved**
[2:0]	**TTR**	**Time Tick Register** The RTC time tick is used for interrupt request. TTR[2:0] / Time tick (second) 0 / 1 1 / 1/2 2 / 1/4 3 / 1/8 4 / 1/16 5 / 1/32 6 / 1/64 7 / 1/128

6.16 Serial Peripheral Interface Controller (SPI Master/Slave)

6.16.1 SPI Function Description and Features

The SPI controller performs a serial-to-parallel conversion on data characters received from the peripheral, and a parallel-to-serial conversion on data characters received from CPU. This controller can drive up to 2 external peripherals, but is time-shared and can not operate simultaneously. It also can be driven as the slave device when the CNTRL[18], SLAVE bit, be set.

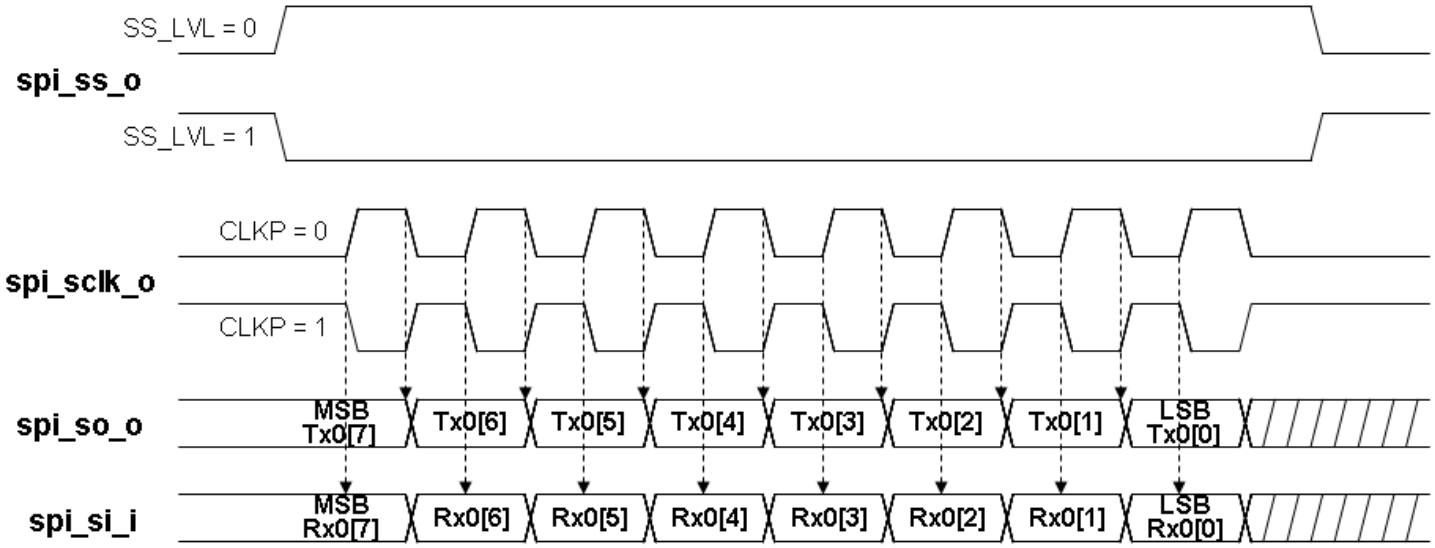
It can generate an interrupt signal when data transfer is finished and can be cleared by writing 1 to the interrupt flag. The active level of slave select signal can be chosen to low active or high active on SSR[SS_LVL] bit, which depends on the peripheral it's connected. Writing a divisor into DIVIDER register can program the frequency of serial clock output. This controller contains four 32-bit transmit/receive buffers, and can provide burst mode operation. It supports variable length transfer and the maximum transmitted/received length can be up to 128 bits.

The SPI Master/Slave Core includes the following features:

- AMBA APB interface compatible
- Support SPI master/slave mode
- Full duplex synchronous serial data transfer
- Variable length of transfer word up to 32 bits
- Provide burst mode operation, transmit/receive can be executed up to four times in one transfer
- MSB or LSB first data transfer
- Rx and Tx on both rising or falling edge of serial clock independently
- 2 slave/device select lines when it is as the master mode, and 1 slave/device select line when it is as the slave mode
- Fully static synchronous design with one clock domain
- Only Support the external master device that the frequency of its serial clock output is less 1/4 than the SPI Core clock input (PCLK) and its slave select output is edge-active trigger.

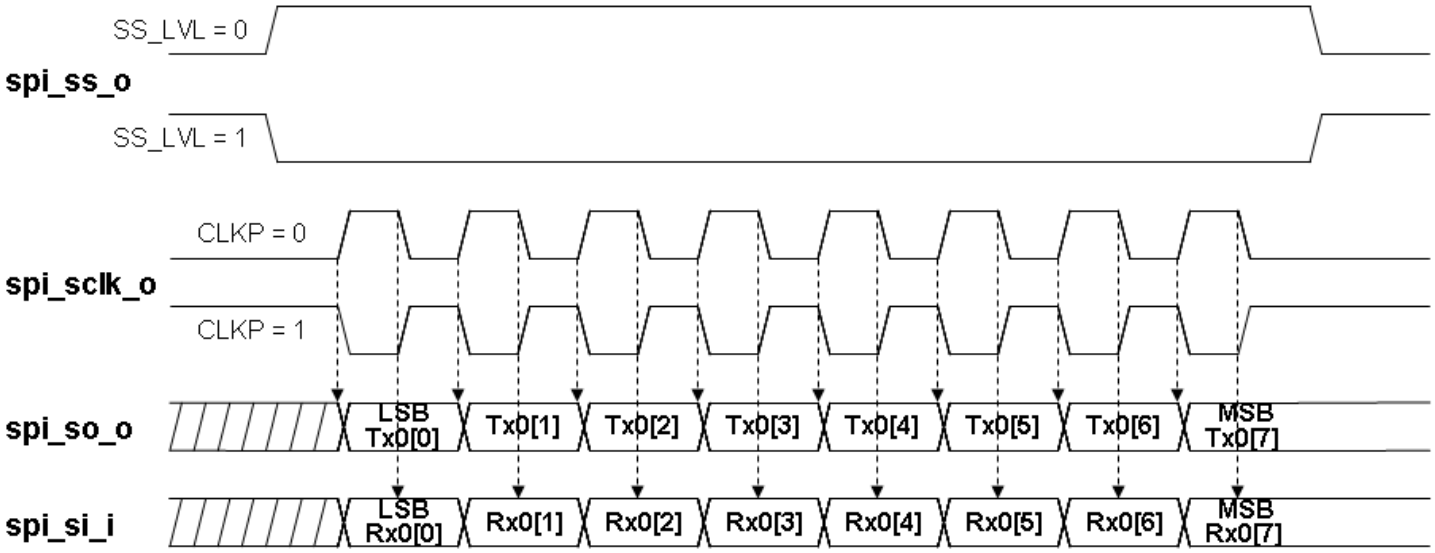
6.16.2 SPIMS Timing Diagram

The timing diagrams of SPI (Master/Slave)are shown as following.



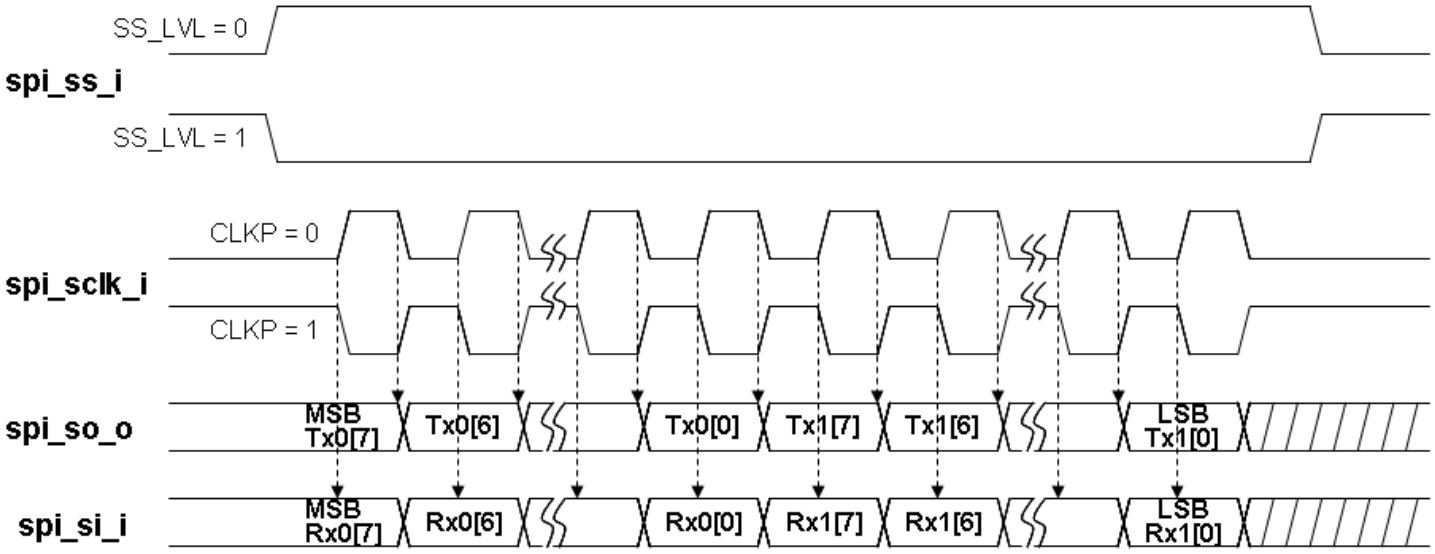
Master Mode : CNTRL[SLAVE]=0, CNTRL[LSB]=0, CNTRL[Tx_NUM]=0x0, CNTRL[Tx_BIT_LEN]=0x08,
1. CNTRL[CLKP]=0, CNTRL[Tx_NEG]=1, CNTRL[Rx_NEG]=0 or
2. CNTRL[CLKP]=1, CNTRL[Tx_NEG]=0, CNTRL[Rx_NEG]=1

SPI Timing (Master)



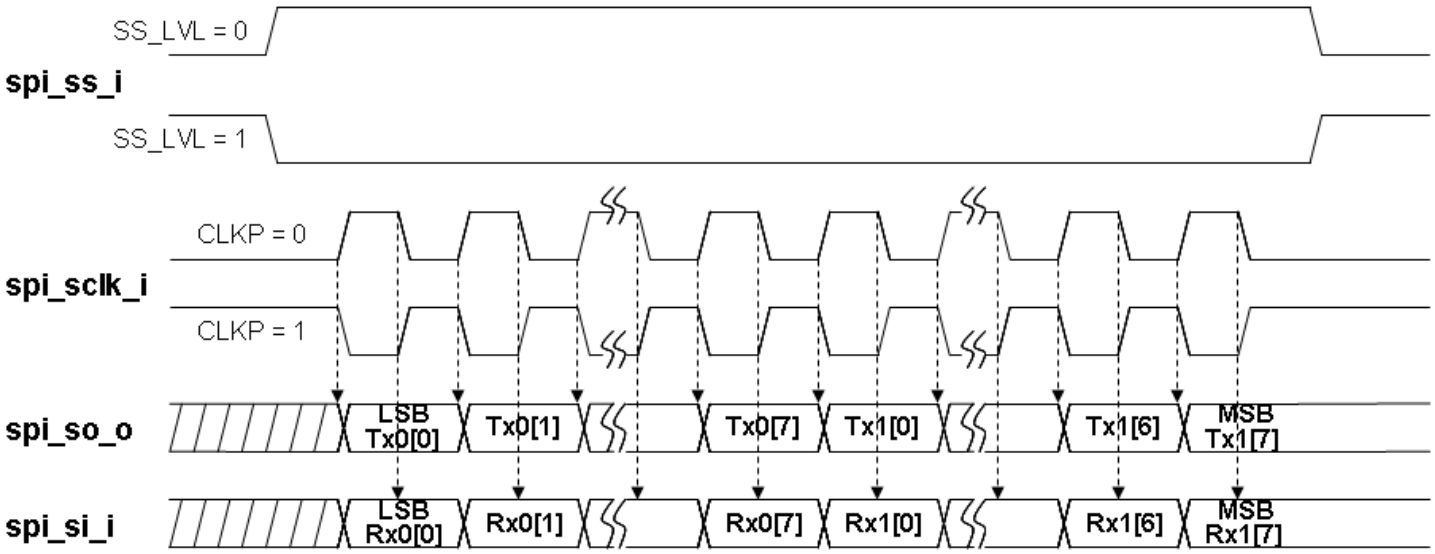
Master Mode : CNTRL[SLAVE]=0, CNTRL[LSB]=1, CNTRL[Tx_NUM]=0x0, CNTRL[Tx_BIT_LEN]=0x08,
1. CNTRL[CLKP]=0, CNTRL[Tx_NEG]=0, CNTRL[Rx_NEG]=1 or
2. CNTRL[CLKP]=1, CNTRL[Tx_NEG]=1, CNTRL[Rx_NEG]=0

Alternate Phase SCLK Clock Timing (Master)



Slave Mode : CNTRL[SLAVE]=1, CNTRL[LSB]=0, CNTRL[Tx_NUM]=0x01, CNTRL[Tx_BIT_LEN]=0x08,
1. CNTRL[CLKP]=0, CNTRL[Tx_NEG]=1, CNTRL[Rx_NEG]=0 or
2. CNTRL[CLKP]=1, CNTRL[Tx_NEG]=0, CNTRL[Rx_NEG]=1

SPI Timing (Slave)



Slave Mode : CNTRL[SLAVE]=1, CNTRL[LSB]=1, CNTRL[Tx_NUM]=0x01, CNTRL[Tx_BIT_LEN]=0x08,
1. CNTRL[CLKP]=0, CNTRL[Tx_NEG]=0, CNTRL[Rx_NEG]=1 or
2. CNTRL[CLKP]=1, CNTRL[Tx_NEG]=1, CNTRL[Rx_NEG]=0

Alternate Phase SCLK Clock Timing (Slave)

6.16.3 SPIMS Programming Example

When using this SPI controller as a master to access a slave device (as slave device) with following specifications:

- Data bit latches on positive edge of serial clock
- Data bit drives on negative edge of serial clock
- Data is transferred with the MSB first
- SCLK idle low.
- Only one byte transmits/receives in a transfer
- Chip select signal is active low

Basically, the following actions should be done (also, the specification of the connected slave device should be referred to when consider the following steps in detail):

1) Write a divisor into DIVIDER to determine the frequency of serial clock.
2) Write in SSR, set ASS = 0, SS_LVL = 0 and SSR[0] or SSR[1] to 1 to activate the device to be accessed.

When transmit (write) data to device:

3) Write the data to be transmitted into Tx0[7:0].

When receive (read) data from device:

4) Write 0xFFFFFFFF into Tx0.
5) Write in CNTRL, set SLAVE = 0, CLKP = 0, Rx_NEG = 0, Tx_NEG = 1, Tx_BIT_LEN = 0x08, Tx_NUM = 0x0, LSB = 0, SLEEP = 0x0 and GO_BUSY = 1 to start the transfer.

-- Wait for interrupt (if IE = 1) or polling the GO_BUSY bit until it turns to 0 --

6) Read out the received data from Rx0.
7) Go to 3) to continue another data transfer or set SSR[0] or SSR[1] to 0 to inactivate the device.

When using this SPI controller as a slave device and connected to a master device, suppose the external master device accesses the on chip SPI interface with the following specifications:

- Data bit latches on positive edge of serial clock
- Data bit drives on negative edge of serial clock
- Data is transferred with the LSB first
- SCLK idle high.
- Only one byte transmits/receives in a transfer
- Chip select signal is active high trigger.

Basically, the following actions should be done (also, the specification of the connected master device should be referred to when consider the following steps in detail):

1) Write in SSR, set SS_LVL = 1.

When transmit (write) data to device:

2) Write the data to be transmitted into Tx0[7:0].

When receive (read) data from device:

3) Write 0xFFFFFFFF into Tx0.

4) Write in CNTRL, set SLAVE = 1, CLKP = 1, Rx_NEG = 0, Tx_NEG = 1, Tx_BIT_LEN = 0x08, Tx_NUM = 0x0, LSB = 1, and GO_BUSY = 1 to start the transfer and waiting for the slave select input and serial clock input signals from the external master device.

-- Wait for interrupt (if IE = 1) or polling the GO_BUSY bit until it turns to 0 --

5) Read out the received data from Rx0.

6) go to 2) to continue another data transfer.

6.16.4 SPIMS Serial Interface Control Register Map

R: read only, **W**: write only, **R/W**: both read and write

Register	Offset	R/W	Description	Reset Value
SPI_BA = 0xB800_A000				
CNTRL	SPIMS_BA + 0x00	R/W	Control and Status Register	0x0000_0004
DIVIDER	SPIMS_BA + 0x04	R/W	Clock Divider Register	0x0000_0000
SSR	SPIMS_BA + 0x08	R/W	Slave Select Register	0x0000_0000
Rx0	SPIMS_BA + 0x10	R	Data Receive Register 0	0x0000_0000
Rx1	SPIMS_BA + 0x14	R	Data Receive Register 1	0x0000_0000
Rx2	SPIMS_BA + 0x18	R	Data Receive Register 2	0x0000_0000
Rx3	SPIMS_BA + 0x1C	R	Data Receive Register 3	0x0000_0000
Tx0	SPIMS_BA + 0x10	W	Data Transmit Register 0	0x0000_0000
Tx1	SPIMS_BA + 0x14	W	Data Transmit Register 1	0x0000_0000
Tx2	SPIMS_BA + 0x18	W	Data Transmit Register 2	0x0000_0000
Tx3	SPIMS_BA + 0x1C	W	Data Transmit Register 3	0x0000_0000

NOTE 1: When software programs CNTRL, the GO_BUSY bit should be written last.

6.16.5 SPIMS Control Register Description

Control and Status Register (CNTRL)

Register	Offset	R/W	Description	Reset Value
CNTRL	SPIMS_BA + 0x00	R/W	Control and Status Register	0x0000_0004

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved					**SLAVE**	**IE**	**IF**
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
SLEEP				**CLKP**	**LSB**	**Tx_NUM**	
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Tx_BIT_LEN					**Tx_NEG**	**Rx_NEG**	**GO_BUSY**

Bits	Descriptions	
[31:19]	**Reserved**	**Reserved**
[18]	**SLAVE**	**SPI Operation Mode** 0 = Master mode. 1 = Slave mode.
[17]	**IE**	**Interrupt Enable** 0 = Disable SPI Interrupt. 1 = Enable SPI Interrupt.
[16]	**IF**	**Interrupt Flag** 0 = It indicates that the transfer dose not finish yet. 1 = It indicates that the transfer is done. The interrupt flag is set if it was enable. **NOTE**: This bit is read only, but can be cleared by writing 1 to this bit.
[15:12]	**SLEEP**	**Suspend Interval (master only)** These four bits provide the configuration of suspend interval between two successive transmit/receive in a transfer. The default value is 0x0. When CNTRL[Tx_NUM] = 00, setting this field has no effect on transfer. The desired interval is obtained according to the following equation (from the last falling edge of current sclk to the first rising edge of next sclk): (CNTRL[SLEEP] + 2)*period of SCLK SLEEP = 0x0 … 2 SCLK clock cycle SLEEP = 0x1 … 3 SCLK clock cycle …… SLEEP = 0xe … 16 SCLK clock cycle

		SLEEP = 0xf … 17 SCLK clock cycle
[11]	**CLKP**	**Clock Polarity** 0 = The serial clock output, spi_sclk_o, idle low. 1 = The serial clock output , spi_sclk_o, idle high.
[10]	**LSB**	**Send LSB First** 0 = The MSB is transmitted/received first (which bit in TxX/RxX register that is depends on the Tx_BIT_LEN field in the CNTRL register). 1 = The LSB is sent first on the line (bit TxX[0]), and the first bit received from the line will be put in the LSB position in the Rx register (bit RxX[0]).
[9:8]	**Tx_NUM**	**Transmit/Receive Numbers** This field specifies how many transmit/receive numbers should be executed in one transfer. 00 = Only one transmit/receive will be executed in one transfer. 01 = Two successive transmit/receive will be executed in one transfer. 10 = Three successive transmit/receive will be executed in one transfer. 11 = Four successive transmit/receive will be executed in one transfer.
[7:3]	**Tx_BIT_LEN**	**Transmit Bit Length** This field specifies how many bits are transmitted in one transmit/receive. Up to 32 bits can be transmitted. Tx_BIT_LEN = 0x01 … 1 bit Tx_BIT_LEN = 0x02 … 2 bits …… Tx_BIT_LEN = 0x1f … 31 bits Tx_BIT_LEN = 0x00 … 32 bits
[2]	**Tx_NEG**	**Data Transmit On Negative Edge** 0 = The spi_so_o signal is changed on the rising edge of spi_sclk_o in master mode or spi_sclk_i in slave mode. 1 = The spi_so_o signal is changed on the falling edge of spi_sclk_o in master mode or spi_sclk_i in slave mode..
[1]	**Rx_NEG**	**Data Receive On Negative Edge** 0 = The spi_si_i signal is latched on the rising edge of spi_sclk_o in master mode or spi_sclk_i in slave mode.. 1 = The spi_si_i signal is latched on the falling edge of spi_sclk_o in master mode or spi_sclk_i in slave mode..
[0]	**GO_BUSY**	**Go and Busy Status** 0 = Writing 0 to this bit has no effect. 1 = Writing 1 to this bit to start the transfer. This bit remains set during the transfer and is automatically cleared after transfer finished. **NOTE**: All registers should be set before writing 1 to the GO_BUSY bit in the CNTRL register. When a transfer is in progress, writing to any register of the SPI master/slave core has no effect.

Divider Register (DIVIDER)

Register	Offset	R/W	Description	Reset Value
DIVIDER	SPIMS_BA + 0x04	R/W	Clock Divider Register	0x0000_0000

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**
DIVIDER[15:8]							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DIVIDER[7:0]							

Bits	Descriptions	
[31:16]	**Reserved**	**Reserved**
[15:0]	**DIVIDER**	**Clock Divider Register (master only)** The value in this field is the frequency divider of the system clock PCLK to generate the serial clock on the output spi_sclk_o. The desired frequency is obtained according to the following equation: $f_{sclk} = \frac{f_{pclk}}{(DIVIDER+1)*2}$ **NOTE:** Suggest DIVIDER should be at least 1.

Slave Select Register (SSR)

Register	Offset	R/W	Description	Reset Value
SSR	SPIMS_BA + 0x08	R/W	Slave Select Register	0x0000_0000

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							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved				**ASS**	**SS_LVL**	**Reserved**	**SSR**

Bits	Descriptions	
[31:4]	**Reserved**	**Reserved**
[3]	**ASS**	**Automatic Slave Select (master only)** 0 = If this bit is cleared, slave select signals are asserted and de-asserted by setting and clearing related bits in SSR register. 1 = If this bit is set, spi_ss_o signals are generated automatically. It means that device/slave select signal, which is set in SSR register is asserted by the SPI controller when transmit/receive is started by setting CNTRL[GO_BUSY], and is de-asserted after every transmit/receive is finished.
[2]	**SS_LVL**	**Slave Select Active Level** It defines the active level of device/slave select signal (spi_ss_o). 0 = The spi_ss_o slave select signal is active Low. 1 = The spi_ss_o slave select signal is active High.
[1]	**Reserved**	Reserved
[0]	**SSR**	**Slave Select Register (master only)** If SSR[ASS] bit is cleared, writing 1 to any bit location of this field sets the proper spi_ss_o line to an active state and writing 0 sets the line back to inactive state. If SSR[ASS] bit is set, writing 1 to any bit location of this field will select appropriate spi_ss_o line to be automatically driven to active state for the duration of the transmit/receive, and will be driven to inactive state for the rest of the time. (The active level of spi_ss_o is specified in SSR[SS_LVL]). **NOTE:** This interface can only drive one device/slave at a given time. Therefore, the slave select of the selected device must be set to its active level before starting any read or write transfer. NOTE: **spi_ss_o is also defined as device/slave select input spi_ss_i**

		signal in slave mode. And that the slave select input spi_ss_i must be driven by edge active trigger which level depend on the SS_LVL setting, otherwise the SPI slave core will go into dead path until the edge active trigger again or reset the SPI core by software.

Data Receive Register (RX)

Register	Offset	R/W	Description	Reset Value
Rx0	SPIMS_BA + 0x10	R	Data Receive Register 0	0x0000_0000
Rx1	SPIMS_BA + 0x14	R	Data Receive Register 1	0x0000_0000
Rx2	SPIMS_BA + 0x18	R	Data Receive Register 2	0x0000_0000
Rx3	SPIMS_BA + 0x1C	R	Data Receive Register 3	0x0000_0000

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

Bits	Descriptions	
[31:0]	**Rx**	**Data Receive Register** The Data Receive Registers hold the value of received data of the last executed transfer. Valid bits depend on the transmit bit length field in the CNTRL register. For example, if CNTRL[Tx_BIT_LEN] is set to 0x08 and CNTRL[Tx_NUM] is set to 0x0, bit Rx0[7:0] holds the received data. **NOTE:** The Data Receive Registers are **read only** registers. A Write to these registers will actually modify the Data Transmit Registers because those registers share the same FFs.

Data Transmit Register (TX)

Register	Offset	R/W	Description	Reset Value
Tx0	SPIMS_BA + 0x10	W	Data Transmit Register 0	0x0000_0000
Tx1	SPIMS_BA + 0x14	W	Data Transmit Register 1	0x0000_0000
Tx2	SPIMS_BA + 0x18	W	Data Transmit Register 2	0x0000_0000
Tx3	SPIMS_BA + 0x1C	W	Data Transmit Register 3	0x0000_0000

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

Bits	Descriptions	
[31:0]	**Tx**	**Data Transmit Register** The Data Transmit Registers hold the data to be transmitted in the next transfer. Valid bits depend on the transmit bit length field in the CNTRL register. For example, if CNTRL[Tx_BIT_LEN] is set to 0x08 and the CNTRL[Tx_NUM] is set to 0x0, the bit Tx0[7:0] will be transmitted in next transfer. If CNTRL[Tx_BIT_LEN] is set to 0x00 and CNTRL[Tx_NUM] is set to 0x3, the core will perform four 32-bit transmit/receive successive using the same setting (the order is Tx0[31:0], Tx1[31:0], Tx2[31:0], Tx3[31:0]). **NOTE:** The RxX and TxX registers share the same flip-flops, which mean that what is received from the input data line in one transfer will be transmitted on the output data line in the next transfer if no write access to the TxX register is executed between the transfers.

6.17 TIMER Controller

6.17.1 General Timer Controller

The timer module includes two channels, TIMER0~TIMER1, which allow you to easily implement a counting scheme for use. The clock source of timer is always the external crystal input clock, i.e. the TCLK speed is dependent on the external crystal clock speed. The timer can perform functions like frequency measurement, event counting, interval measurement, clock generation, delay timing, and so on. The timer possesses features such as adjustable resolution, programmable counting period, and detailed information. The timer can generate an interrupt signal upon timeout, or provide the current value of count during operation.

The general TIMER Controller includes the following features

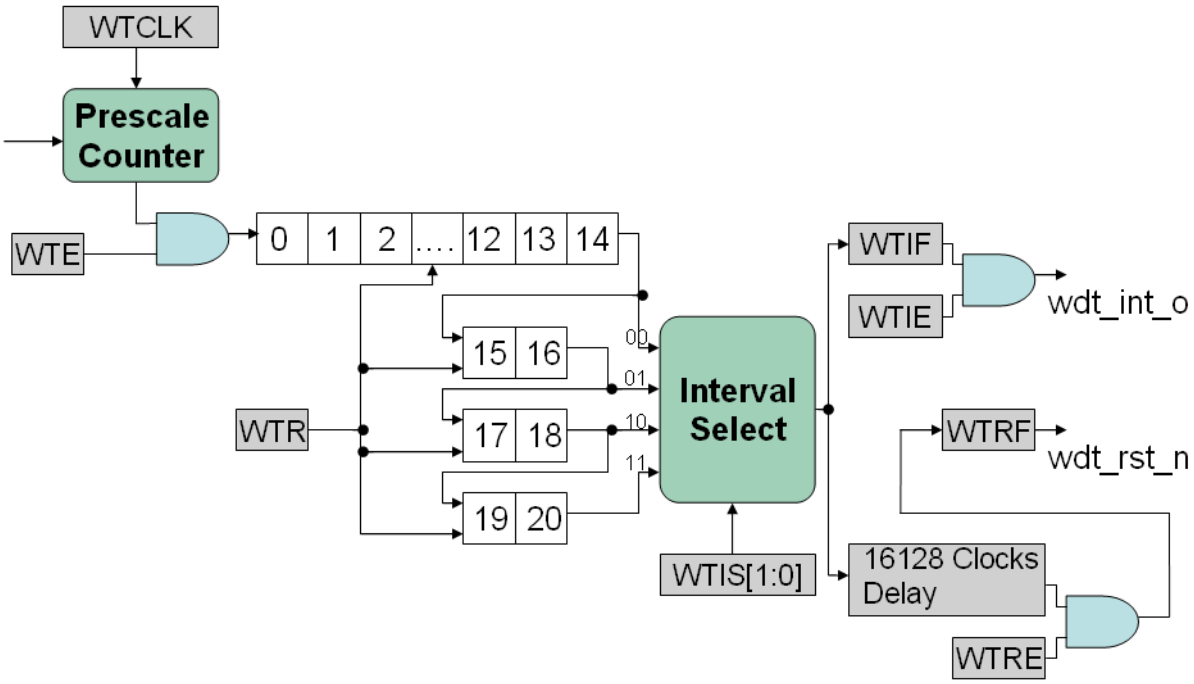
- AMBA APB interface compatible
- Each channel with a 8-bit pre-scale counter/32-bit counter and an interrupt request signal.
- Independent clock source for each channel(TCLK0,TCLK1)
- Maximum uninterrupted time = (1 / 25 MHz) * (2^8) * (2^32), if TCLK = 25 MHz

6.17.2 Watchdog Timer

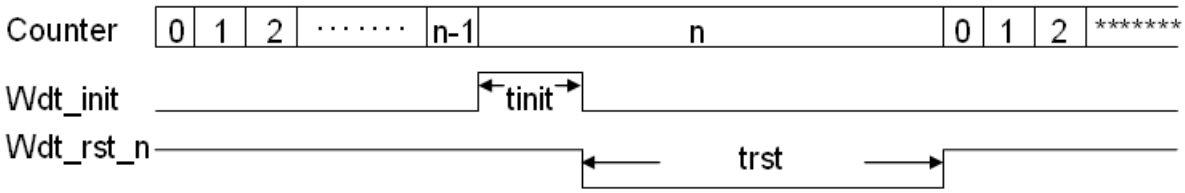
The purpose of Watchdog Timer is to perform a system restart after the software running into a problem. This prevents system from hanging for an indefinite period of time. It is a free running timer with programmable timeout intervals. When the specified time interval expires, a system reset can be generated. If the Watchdog Timer reset function is enabled and the Watchdog Timer is not being reset before timing out, then the Watchdog Timer reset is activated after **1024** WDT clock cycles (Interrupt timeout). Setting **WTE** in the register **WTCR** enables the Watchdog Timer.

The **WTR** should be set before making use of Watchdog Timer. This ensures that the Watchdog Timer restarts from a know state. Watchdog Timer will start counting and timeout after a specified period of time. The timeout interval is selected by two bits, **WTIS[1:0]**. The **WTR** is self-clearing, i.e., after setting it; the hardware will automatically reset it.

When timeout occurs, Watchdog Timer interrupt flag is set. Watchdog Timer waits for an additional **1024 WDT clock cycles** before issuing a reset signal, if the **WTRE** is set. The **WTRF** will be set and the reset signal will last for **16128 WDT clock cycles** long. When used as a simple timer, the interrupt and reset functions are disabled. Watchdog Timer will set the **WTIF** each time a timeout occurs. The **WTIF** can be polled to check the status, and software can restart the timer by setting the **WTR**. The Watchdog Timer can be put in the test mode by setting **WTTME** in the register WTCR.



Watchdog Timer Block Diagram



Parameter	Min	Max	Unit
n	2^14	2^20	wdt clock cycle
t_init		1024	wdt clock cycle
t_rst		16128	wdt clock cycle

Watchdog Timer Timing Diagram

6.17.3 Timer Control Registers Map

R: read only, **W**: write only, **R/W**: both read and write

Register	Address	R/W/C	Description	Reset Value
TMR_BA = 0xB800_B000				
TCSR0	TMR_BA+00	R/W	Timer Control and Status Register 0	0x0000_0005
TCSR1	TMR_BA+04	R/W	Timer Control and Status Register 1	0x0000_0005
TICR0	TMR_BA+08	R/W	Timer Initial Control Register 0	0x0000_0000
TICR1	TMR_BA+0C	R/W	Timer Initial Control Register 1	0x0000_0000
TDR0	TMR_BA+10	R	Timer Data Register 0	0x0000_0000
TDR1	TMR_BA+14	R	Timer Data Register 1	0x0000_0000
TISR	TMR_BA+18	R/W	Timer Interrupt Status Register	0x0000_0000
WTCR	TMR_BA+1C	R/W	Watchdog Timer Control Register	0x0000_0400

Timer Control Register 0~1 (TCSR0~TCSR1)

Register	Address	R/W	Description	Reset Value
TCSR0	TMR_BA+000	R/W	Timer Control and Status Register 0	0x0000_0005
TCSR1	TMR_BA+004	R/W	Timer Control and Status Register 1	0x0000_0005

31	30	29	28	27	26	25	24
nDBGACK_EN	**CEN**	**IE**	**MODE[1:0]**		**CRST**	**CACT**	**Reserved**
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
PRESCALE[7:0]							

Bits	Descriptions	
[31]	**nDBGACK_EN**	**ICE debug mode acknowledge enable** • 0 = When DBGACK is high, the TIMER counter will be held • 1 = No matter DBGACK is high or not, the TIMER counter will not be held
[30]	**CEN**	**Counter Enable** • 0 = Stops/Suspends counting • 1 = Starts counting
[29]	**IE**	**Interrupt Enable** • 0 = **Disable** TIMER Interrupt. • 1 = **Enable** TIMER Interrupt. If timer interrupt is enabled, the timer asserts its interrupt signal when the associated counter is equal to TICR.
[28:27]	**MODE**	**Timer Operating Mode** **MODE** / **Timer Operating Mode** 00 / The timer is operating in the one-shot mode. The associated interrupt signal is generated once (if IE is enabled) and CEN

		<table><tr><td></td><td>is automatically cleared then.</td></tr><tr><td>01</td><td>The timer is operating in the periodic mode. The associated interrupt signal is generated periodically (if IE is enabled).</td></tr><tr><td>10</td><td>The timer is operating in the toggle mode. The interrupt signal is generated periodically (if IE is enabled). And the associated signal (tout) is changing back and forth with 50% duty cycle.</td></tr><tr><td>11</td><td>The timer is operating in the uninterrupted mode. The associated interrupt signal is generated when TDR = TICR (if IE is enabled) .</td></tr></table>
[26]	**CRST**	**Counter Reset** Set this bit will reset the TIMER counter, and also **force CEN to 0**. • 0 = No effect. • 1 = Reset Timer's pre-scale counter, internal 32-bit counter and CEN.
[25]	**CACT**	**Timer is in Active** This bit indicates the counter status of timer. • 0 = Timer is **not** active. • 1 = Timer is **in** active.
[24:8]	**Reserved**	**Reserved**
[7:0]	**PRESCALE**	**Pre-scale** Clock input is divided by Prescale+1 before it is fed to the counter. If Pre-scale=0, then there is no scaling.

Timer Initial Count Register 0~1 (TICR0~TICR1)

Register	Address	R/W	Description	Reset Value
TICR0	TMR_BA+008	R/W	Timer Initial Control Register 0	0x0000_0000
TICR1	TMR_BA+00C	R/W	Timer Initial Control Register 1	0x0000_0000

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

Bits	Descriptions	
[31:0]	**TIC**	**Timer Initial Count** This is a 32-bit value representing the initial count. Timer will reload this value whenever the counter is decremented to zero. **NOTE1:** Never write 0x0 in TIC, or the core will run into unknown state. **NOTE2:** No matter CEN is 0 or 1, whenever software write a new value into this register, TIMER will restart counting using this new value and abort previous count.

Timer Data Register 0~1 (TDR0~TDR1)

Register	Address	R/W	Description	Reset Value
TDR0	TMR_BA+10	R	Timer Data Register 0	0x0000_0000
TDR1	TMR_BA+14	R	Timer Data Register 1	0x0000_0000

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

Bits	Descriptions	
[31:0]	**TDR**	**Timer Data Register** The current count is registered in this 32-bit value. **NOTE:** Software can read a correct current value on this register only when **CEN = 0**, or the value represents here could not be a correct one.

Timer Interrupt Status Register (TISR)

Register	Address	R/W	Description	Reset Value
TISR	TMR_BA+18	R/W	Timer Interrupt Status Register	0x0000_0000

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							
7	6	5	4	3	2	1	0
Reserved						**TIF1**	**TIF0**

Bits	Descriptions	
[31:2]	**Reserved**	**Reserved**
[1]	**TIF1**	**Timer Interrupt Flag 1** This bit indicates the interrupt status of Timer channel 1. • 0 = It indicates that the Timer 1 dose not countdown to zero yet. • 1 = It indicates that the counter of Timer 1 has decremented to zero. The interrupt flag is set if it was enable. **NOTE**: This bit is read only, but can be cleared by writing 1 to this bit.
[0]	**TIF0**	**Timer Interrupt Flag 0** This bit indicates the interrupt status of Timer channel 0. • 0 = It indicates that the Timer 0 dose not countdown to zero yet. • 1 = It indicates that the counter of Timer 0 has decremented to zero. The interrupt flag is set if it was enable. **NOTE**: This bit is read only, but can be cleared by writing 1 to this bit.

Watchdog Timer Control Register (WTCR)

Register	Address	R/W	Description	Reset Value
WTCR	TMR_BA+01C	R/W	Watchdog Timer Control Register	0x0000_0400

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					**WTCLK**	**nDBGACK_EN**	**WTTME**
7	6	5	4	3	2	1	0
WTE	**WTIE**	**WTIS**		**WTIF**	**WTRF**	**WTRE**	**WTR**

Bits	Descriptions	
[31:11]	**Reserved**	**Reserved**
[10]	**WTCLK**	**Watchdog Timer Clock** This bit is used for deciding whether the Watchdog timer clock input is divided by 256 or not. Clock source of Watchdog timer is Crystal input. • 0 = Using original clock input • 1 = The clock input will be divided by 256 **NOTE:** When WTTME = 1, set this bit has no effect on WDT clock (using original clock input).
[9]	**nDBGACK_EN**	**ICE debug mode acknowledge enable** • 0 = When DBGACK is high, the Watchdog timer counter will be held • 1 = No matter DBGACK is high or not, the Watchdog timer counter will not be held
[8]	**WTTME**	**Watchdog Timer Test Mode Enable** For reasons of efficiency, the 20-bit counter within the Watchdog timer is considered as two independent 10-bit counters in the test mode. They are operated concurrently and separately during the test. This approach can save a lot of time spent in the test. When the 10-bit counter overflows, a Watchdog timer interrupt is generated.

<table>
<tr><td></td><td></td><td>

0 = Put the Watchdog timer in normal operating mode
1 = Put the Watchdog timer in test mode

</td></tr>
<tr><td>[7]</td><td>WTE</td><td>Watchdog Timer Enable

0 = Disable the Watchdog timer (This action will reset the internal counter)
1 = Enable the Watchdog timer

</td></tr>
<tr><td>[6]</td><td>WTIE</td><td>Watchdog Timer Interrupt Enable

0 = Disable the Watchdog timer interrupt
1 = Enable the Watchdog timer interrupt

</td></tr>
<tr><td>[5:4]</td><td>WTIS</td><td>Watchdog Timer Interval Select
These two bits select the interval for the Watchdog timer. No matter which interval is chosen, the reset timeout is always occurred 16128 WDT clock cycles later than the interrupt timeout.
<table>
<tr><th>WTIS</th><th>Timeout</th><th>Interrupt Timeout</th><th>Real Time Interval (CLK=15MHz/256)</th></tr>
<tr><td>00</td><td>2^{14} clocks</td><td>2^{14} + 1024 clocks</td><td>0.28 sec.</td></tr>
<tr><td>01</td><td>2^{16} clocks</td><td>2^{16} + 1024 clocks</td><td>1.12 sec.</td></tr>
<tr><td>10</td><td>2^{18} clocks</td><td>2^{18} + 1024 clocks</td><td>4.47 sec.</td></tr>
</table>
</td></tr>
<tr><td>[3]</td><td>WTIF</td><td>Watchdog Timer Interrupt Flag
If the Watchdog timer interrupt is enabled, then the hardware will set this bit to indicate that the Watchdog timer interrupt has occurred. If the Watchdog timer interrupt is not enabled, then this bit indicates that a timeout period has elapsed.

0 = Watchdog timer interrupt does not occur
1 = Watchdog timer interrupt occurs

NOTE: This bit is read only, but can be cleared by writing 1 to this bit.</td></tr>
<tr><td>[2]</td><td>WTRF</td><td>Watchdog Timer Reset Flag
When the Watchdog timer initiates a reset, the hardware will set this bit. This flag can be read by software to determine the source of reset. Software is responsible to clear it up manually. If WTRE is disabled, then the Watchdog timer has no effect on this bit.

0 = Watchdog timer reset does not occur
1 = Watchdog timer reset occurs

NOTE: This bit is read only, but can be cleared by writing 1 to this bit.</td></tr>
<tr><td>[1]</td><td>WTRE</td><td>Watchdog Timer Reset Enable
Setting this bit will enable the Watchdog timer reset function.</td></tr>
</table>

		• 0 = Disable Watchdog timer reset function • 1 = Enable Watchdog timer reset function
[0]	**WTR**	**Watchdog Timer Reset** This bit brings the Watchdog timer into a known state. It helps reset the Watchdog timer before a timeout situation occurring. Failing to set **WTR** before timeout will initiates an interrupt if **WTIE** is set. If the **WTRE** bit is set, Watchdog timer reset will be occurred 16128 WDT clock cycles after timeout. This bit is self-clearing. • 0 = Writing 0 to this bit has no effect • 1 = Reset the contents of the Watchdog timer **NOTE**: This bit will auto clear after few clock cycle

6.18 UART Interface Controller

6.18.1 Overview

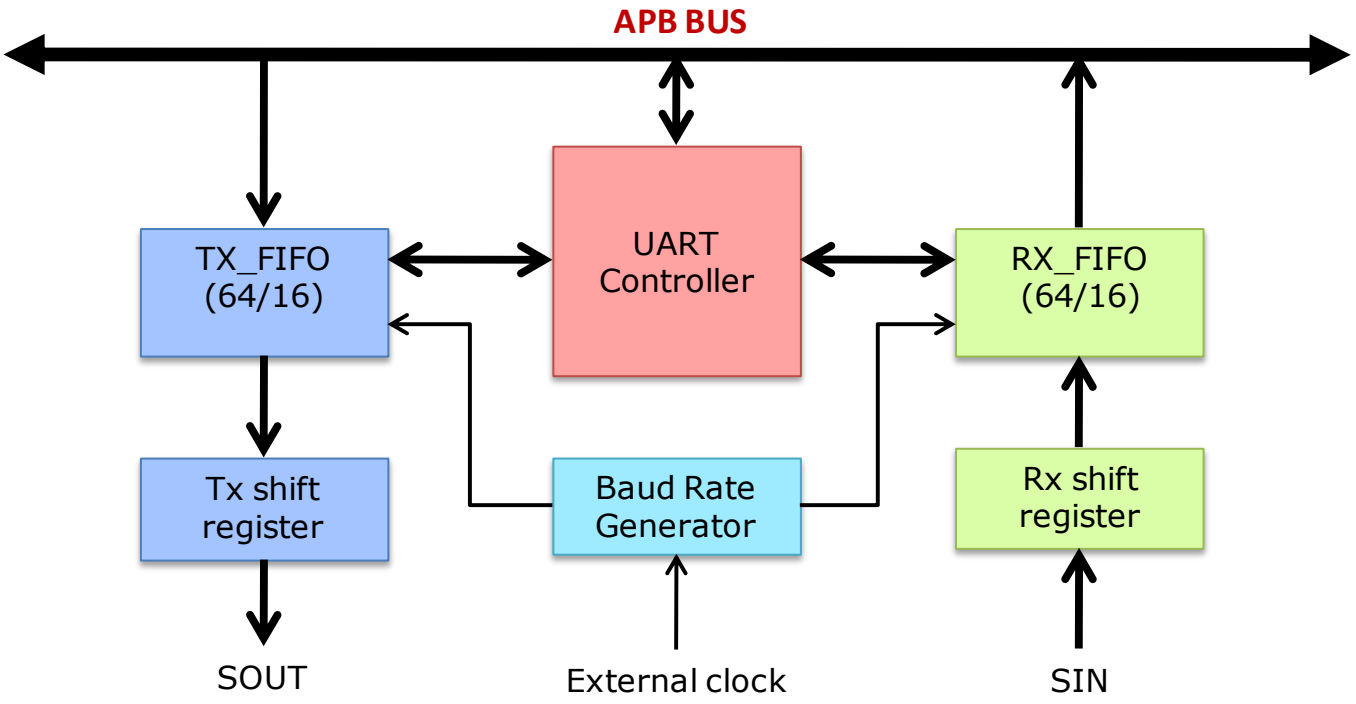
The UART interface controller module includes two channels, UART0~UARTR1. One of them is equipped with flow control function High Speed UART and the other is a Normal Speed UART. The Universal Asynchronous Receiver/Transmitter **(UART)** performs a serial-to-parallel conversion on data characters received from the peripheral, and a parallel-to-serial conversion on data characters received from the CPU. There are six types of interrupts, they are, transmitter FIFO empty interrupt**(Int_THRE)**, receiver threshold level reaching interrupt **(Int_RDA),** line status interrupt (overrun error or parity error or framing error or break interrupt) **(Int_RLS)** , time out interrupt **(Int_Tout)**, MODEM status interrupt **(Int_Modem)** and Wake up status interrupt **(Int_WakeUp).**

The two UART Interface Controller that one have a **64-byte** transmitter FIFO (TX_FIFO) and a **64-byte** (plus 3-bit of error data per byte) receiver FIFO (RX_FIFO) has been built in to reduce the number of interrupts presented to the CPU and the other have a **16-byte** transmitter FIFO (TX_FIFO) and a **16-byte** (plus 3-bit of error data per byte) receiver FIFO (RX_FIFO) has been built in to reduce the number of interrupts presented to the CPU. The CPU can completely read the status of the UART at any time during the operation. The reported status information includes the type and condition of the transfer operations being performed by the UART, as well as any error conditions (parity error, overrun error, framing error, or break interrupt) found. The UART includes a programmable baud rate generator that is capable of dividing crystal clock input by divisors to produce the clock that transmitter and receiver needed. The baud rate equation is **Baud Out = crystal clock / 16 * [Divisor + 2].**

6.18.2 Features:

- 64 byte/16 byte entry FIFOs for received and transmitted data payloads
- Flow control functions (CTS, RTS) are supported.
- Programmable baud-rate generator that allows the internal clock to be divided by 2 to (2^16 + 1) to generate an internal 16X clock.
- Fully programmable serial-interface characteristics:
  - 5-, 6-, 7-, or 8-bit character
  - Even, odd, or no-parity bit generation and detection
  - 1-, 1&1/2, or 2-stop bit generation
  - Baud rate generation
  - False start bit detection.
- Loop back mode for internal diagnostic testing

6.18.3 Block Diagram



6.18.4 Functional Blocks Descriptions

TX_FIFO

The transmitter is buffered with a 64/16 byte FIFO to reduce the number of interrupts presented to the CPU.

RX_FIFO

The receiver is buffered with a 64/16 byte FIFO (plus three error bits per byte) to reduce the number of interrupts presented to the CPU.

TX shift Register

Shifting the transmitting data out serially

RX shift Register

Shifting the receiving data in serially

Modem Control Register

This register controls the interface to the MODEM or data set (or a peripheral device emulating a MODEM).

Modem Status Register

This register provides the current status of the control lines from the MODEM and cause the MODEM status interrupt (CTS# or DSR# or RI# or DCD#)

Note: Only CTS#/RTS# can be used in this version.

Baud Rate Generator

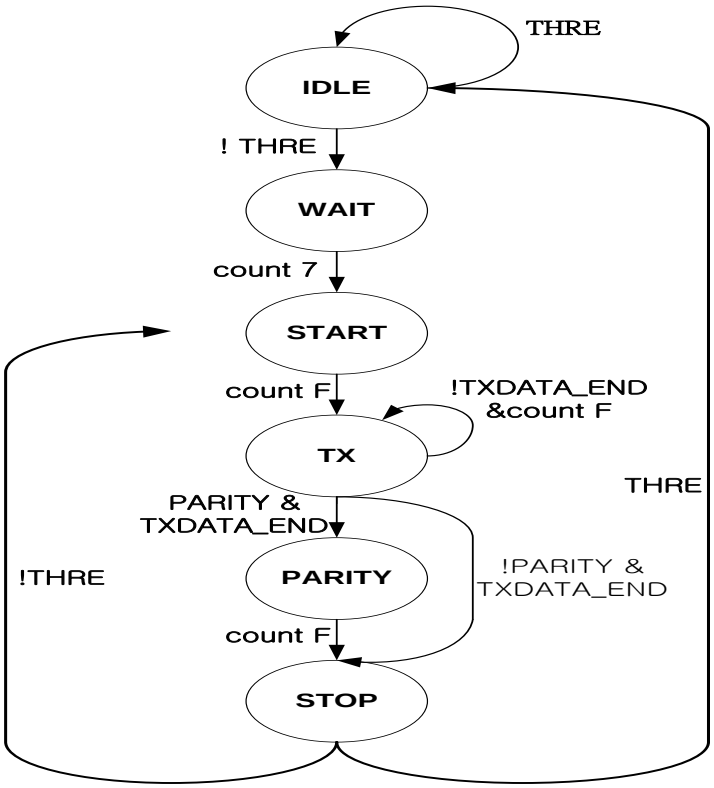
Dividing the external clock by the divisor to get the desired internal clock

Control and Status Register

This is a register set, including the FIFO control registers (FCR), FIFO status registers (FSR), and line control register (LCR) for transmitter and receiver. The line status register (LSR) provides information to the CPU concerning the data transfer. The time out control register (TOR) identifies the condition of time out interrupt. This register set also includes the interrupt enable register (IER) and interrupt identification register (IIR) to enable or disable the responding interrupt and to identify the occurrence of the responding interrupt. There are four types of interrupts: line status interrupt (overrun error or parity error or framing error or break interrupt), transmitter holding register empty interrupt, receiver threshold level reaching, and time out interrupt.

6.18.5 Finite State Machine

6.18.5.1 Transmitter



State Definition

IDLE

The transmitter has no data to transmit.

WAIT

The transmitter's FIFO is not empty.

START

The transmitter transmits the start bit.

TX

The transmitter transmits the data.

PARITY

The transmitter transmits the parity bit.

STOP

The transmitter transmits the stop bit.

Signal Description

THRE

Te transmitter holding register is empty.

Count7

The counter of clock equals to 7.

CountF

The counter of clock equals to 15.

TXDATA_END

The data part transfer is finished.

PARITY

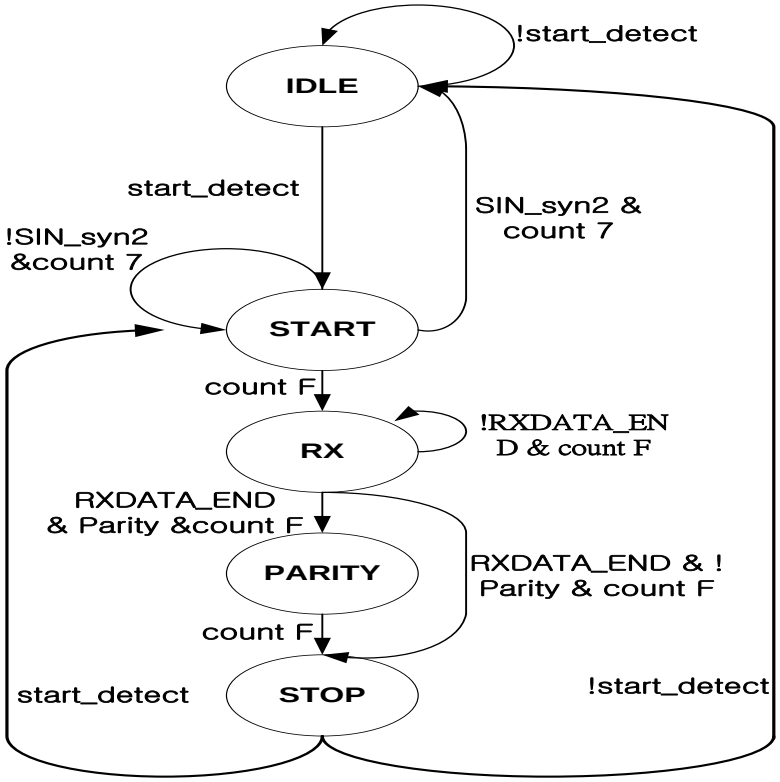
The transfer includes the parity bit.

NOTE:

The format of the transfer is as following:

One transfer = START + DATA + Parity bit (if dedicated) + Stop bit

6.18.5.2 Receiver



State Definition

IDLE

The receiver has no data to receive.

START

The receiver receives the start bit.

RX

The receiver receives the desired data.

PARITY

The receiver receives the parity bit.

STOP

The receiver receives the parity bit.

Signal Description

Start_detect

To detect the start of the transfer

SIN_syn2

The synchronized input data

Count7

The counter of clock equals to 7.

CountF

The counter of clock equals to F.

RXDATA_END

The data received finished

PARITY

Receiving the parity bit if needed

6.18.6 UART Interface Control Registers Mapping

R: read only, **W**: write only, **R/W**: both read and write, **C**: Only value 0 can be written

First set of the UART Interface register Map

Channel0: UART_Base0 (High Speed) = B800_C000

Channel1: UART_Base1 (Normal Speed) = B800_C100

Register	Address	R/W	Description	Reset Value
UART_BA = 0xB800_C000 / 0xB800_C100				
UA_RBR	UART_BA + 0x00	R	Receive Buffer Register (DLAB = 0)	Undefined
UA_THR	UART_BA + 0x00	W	Transmit Holding Register (DLAB = 0)	Undefined
UA_IER	UART_BA + 0x04	R/W	Interrupt Enable Register (DLAB = 0)	0x0000_0000
UA_DLL	UART_BA + 0x00	R/W	Divisor Latch Register (LS) (DLAB = 1)	0x0000_0000
UA_DLM	UART_BA + 0x04	R/W	Divisor Latch Register (MS) (DLAB = 1)	0x0000_0000
UA_IIR	UART_BA + 0x08	R	Interrupt Identification Register	0x8181_8181
UA_FCR	UART_BA + 0x08	W	FIFO Control Register	Undefined
UA_LCR	UART_BA + 0x0C	R/W	Line Control Register	0x0000_0000
UA_MCR	UART_BA + 0x10	R/W	Modem Control Register	0x0000_0000
UA_LSR	UART_BA + 0x14	R	Line Status Register	0x6060_6060
UA_MSR	UART_BA + 0x18	R/W	Modem Status Register	0x0000_0000
UA_TOR	UART_BA + 0x1C	R/W	Time Out Register	0x0000_0000

Receive Buffer Register (UA_RBR)

Register	Address	R/W	Description	Reset Value
UA_RBR	UA_BA + 0x00	R	Receive Buffer Register (DLAB = 0)	Undefined

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							
7	6	5	4	3	2	1	0
8-bit Received Data							

Bits	Descriptions	
[31:8]	**Reserved**	**Reserved**
[7:0]	**8-bit Received Data**	By reading this register, the UART will return an 8-bit data received from SIN pin (LSB first).

Transmit Holding Register (UA_THR)

Register	Address	R/W	Description	Reset Value
UA_THR	UA_BA + 0x00	W	Transmit Holding Register (DLAB = 0)	Undefined

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							
7	6	5	4	3	2	1	0
8-bit Transmitted Data							

Bits	Descriptions	
[31:8]	**Reserved**	**Reserved**
[7:0]	**8-bit Transmitted Data**	By writing to this register, the UART will send out an 8-bit data through the SOUT pin (LSB first).

Interrupt Enable Register (UA_IER)

Register	Address	R/W	Description	Reset Value
UA_IER	UA_BA + 0x04	R/W	Interrupt Enable Register (DLAB = 0)	0x0000.0000

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							
7	6	5	4	3	2	1	0
Wake_o_IE	**WakeIE**	**nDBGACK_EN**	**RTOIE**	**MSIE**	**RLSIE**	**THREIE**	**RDAIE**

Bits	Descriptions	
[31:8]	**Reserved**	**Reserved**
[7]	**Wake_o_IE**	**Wake up interrupt enable for Irpt_WakeUp** ● 0 = Mask off Irpt_Wakeup ● 1 = Enable Irpt_Wakeup
[6]	**WakeIE**	**Wake up interrupt enable for INTR[wakeup]** ● 0 = Mask off INTR_Wakeup ● 1 = Enable INTR_Wakeup
[5]	**nDBGACK_EN**	**ICE debug mode acknowledge enable** ● 0 = When DBGACK is high, the UART receiver time-out clock will be held ● 1 = No matter what DBGACK is high or not, the UART receiver timer-out clock will not be held
[4]	**RTOIE**	**RX Time out Interrupt Enable** ● 0 = Mask off INTR_tout ● 1 = Enable INTR_tout
[3]	**MSIE**	**MODEM Status Interrupt (INTR_MOS) Enable** ● 0 = Mask off INTR_MOS ● 1 = Enable INTR_MOS
[2]	**RLSIE**	**Receive Line Status Interrupt (INTR_RLS) Enable**

		• 0 = Mask off INTR_RLS • 1 = Enable INTR_RLS
[1]	**THREIE**	**Transmit Holding Register Empty Interrupt (INTR_THRE) Enable** • 0 = Mask off INTR_THRE • 1 = Enable INTR_THRE
[0]	**RDAIE**	**R**eceive **D**ata **A**vailable **I**nterrupt **(INTR_RDA) E**nable. • 0 = Mask off INTR_RDA • 1 = Enable INTR_RDA

Divider Latch (Low Byte) Register (UA_DLL)

Register	Address	R/W	Description	Reset Value
UA_DLL	UA_BA + 0x00	R/W	Divisor Latch Register (LS) (DLAB = 1)	0x0000_0000

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							
7	6	5	4	3	2	1	0
Baud Rate Divider (Low Byte)							

Bits	Descriptions	
[31:8]	**Reserved**	**Reserved**
[7:0]	**Baud Rate Divisor (Low Byte)**	The low byte of the baud rate divider

Divisor Latch (High Byte) Register (UA_DLM)

Register	Address	R/W	Description	Reset Value
UA_DLM	UA_BA + 0x04	R/W	Divisor Latch Register (MS) (DLAB = 1)	0x0000_0000

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							
7	6	5	4	3	2	1	0
Baud Rate Divider (High Byte)							

Bits	Descriptions	
[31:8]	**Reserved**	**Reserved**
[7:0]	**Baud Rate Divisor (High Byte)**	The high byte of the baud rate divider

This 16-bit divider {UA_DLM, UA_DLL} is used to determine the baud rate as follows

Baud Rate = Crystal Clock / {16 * [Divisor + 2]}

Interrupt Identification Register (UA_IIR)

Register	Address	R/W	Description	Reset Value
UA_IIR	UA_BA + 0x08	R	Interrupt Identification Register	0x8181_8181

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							
7	6	5	4	3	2	1	0
FMES	**RFTLS**		**Reserved**	**IID_RX**	**IID**		

Bits	Descriptions	
[31:8]	**Reserved**	**Reserved**
[7]	**FMES**	**FIFO Mode Enable Status** This bit indicates whether the FIFO mode is enabled or not. Since the FIFO mode is always enabled, this bit always shows the logical 1 when CPU is reading this register.
[6:5]	**RFTLS**	**RX FIFO Threshold Level Status** These bits show the current setting of receiver FIFO threshold level (RTHO). The meaning of RTHO is defined in the following UA_FCR description.
[4]	**Reserved**	**Reserved**
[3]	**IID_RX**	**Interrupt Identification of RX time out** This bit indicates the current interrupt request from RX time out.(The Rx buffer have data (not reach the Rx trigger level) but the time out count is equal to TOR
[2:0]	**IID**	**Interrupt Identification** The IID together with NIP indicates the current interrupt request from UART.

Interrupt Control Functions

UA_IIR [3:0]	Priority	Interrupt Type	Interrupt Source	Interrupt Reset control
---1	--	None	None	--
0110	Highest	Receiver Line Status (INTR_RLS)	Overrun error, parity error, framing error, or break interrupt	Reading the UA_LSR
0100	Second	Received Data Available (INTR_RDA)	Receiver FIFO threshold level is reached	Receiver FIFO drops below the threshold level
1100	Second	Receiver FIFO Time-out (INTR_TOUT)	Receiver FIFO is non-empty and no activities are occurred in the receiver FIFO during the UA_TOR defined time duration	Reading the UA_RBR
0010	Third	Transmitter Holing Register Empty (INTR_THRE)	Transmitter holding register empty	Reading the UA_IIR (if source of interrupt is INTR_THRE) or writing into the UA_THR
0000	Fourth	MODEM Status (INTR_MOS)	The CTS, DSR or DCD bits are changing state or the RI bit is changing from high to low.	Reading the MSR

Note1: The definition of bit-7, bit-6, bit-5 and bit-4 is different from the 16550.

Note2: Only CTS/CTS can be used in this version

FIFO Control Register (UA_FCR)

Register	Address	R/W	Description	Reset Value
UA_FCR	UA_BA + 0x08	W	FIFO Control Register	Undefined

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							
7	6	5	4	3	2	1	0
RFITL				**Reserved**	**TFR**	**RFR**	**FME**

Bits	Descriptions	
[31:8]	**Reserved**	**Reserved**
[7:4]	**RFITL**	**RX FIFO Interrupt (INTR_RDA) Trigger Level** **RFITL** / **INTR_RDA Trigger Level (Bytes)** 0000: 01 0001: 04 0010: 08 0011: 14 0100: 30/14 (High Speed/Normal Speed) 0101: 46/14 (High Speed/Normal Speed) 0110: 62/14 (High Speed/Normal Speed) others: 62/14 (High Speed/Normal Speed)
[3]	**Reserved**	**Reserved**
[2]	**TFR**	**TX FIFO Reset** Setting this bit will generate an OSC cycle reset pulse to reset TX FIFO. The TX FIFO becomes empty (TX pointer is reset to 0) after such reset. This bit is returned to 0 automatically after the reset pulse is generated.
[1]	**RFR**	**RX FIFO Reset** Setting this bit will generate an OSC cycle reset pulse to reset RX FIFO. The RX FIFO becomes empty (RX pointer is reset to 0) after such reset. This bit is returned

		to 0 automatically after the reset pulse is generated.
[0]	**FME**	**FIFO Mode Enable** Because UART is always operating in the FIFO mode, writing this bit has no effect while reading always gets logical one. This bit must be 1 when other UA_FCR bits are written to; otherwise, they will not be programmed.

Line Control Register (UA_LCR)

Register	Address	R/W	Description	Reset Value
UA_LCR	UA_BA + 0x0C	R/W	Line Control Register	0x0000_0000

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							
7	6	5	4	3	2	1	0
DLAB	**BCB**	**SPE**	**EPE**	**PBE**	**NSB**	**WLS**	

Bits	Descriptions	
[31:8]	**Reserved**	**Reserved**
[7]	**DLAB**	**Divider Latch Access Bit** • 0 = It is used to access UA_RBR, UA_THR or UA_IER. • 1 = It is used to access Divisor Latch Registers {UA_DLL, UA_DLM}.
[6]	**BCB**	**Break Control Bit** When this bit is set to logic 1, the serial data output (SOUT) is forced to the Spacing State (logic 0). This bit acts only on SOUT and has no effect on the transmitter logic.
[5]	**SPE**	**Stick Parity Enable** • 0 = Disable stick parity • 1 = Parity bit is transmitted and checked as a logic 1 if bit 4 is 0 (odd parity), or as a logic 0 if bit 4 is 1 (even parity). This bit has effect only when bit 3 (parity bit enable) is set.
[4]	**EPE**	**Even Parity Enable** • 0 = Odd number of logic 1's are transmitted or checked in the data word and parity bits. • 1 = Even number of logic 1's are transmitted or checked in the data word and parity bits. This bit has effect only when bit 3 (parity bit enable) is set.

<table>
<tr><td>[3]</td><td>PBE</td><td>Parity Bit Enable
0 = Parity bit is not generated (transmit data) or checked (receive data) during transfer.
1 = Parity bit is generated or checked between the "last data word bit" and "stop bit" of the serial data.</td></tr>
<tr><td>[2]</td><td>NSB</td><td>Number of “STOP bit”
0= One “ STOP bit” is generated in the transmitted data
1= One and a half “ STOP bit” is generated in the transmitted data when 5-bit word length is selected;
Two “STOP bit” is generated when 6-, 7- and 8-bit word length is selected.</td></tr>
<tr><td>[1:0]</td><td>WLS</td><td>Word Length Select<table><tr><th>WLS[1:0]</th><th>Character length</th></tr><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></table></td></tr>
</table>

MODEM Control Register (UA_MCR)

Register	Address	R/W	Description	Reset Value
UA_MCR	UA_BA + 0x10	R/W	MODEM Control Register	0x0000.0000

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							
7	6	5	4	3	2	1	0
Reserved			**LBME**	**Reserved**		**RTS#**	**Reserved**

Bits	Descriptions	
[31:5]	**Reserved**	**Reserved**
[4]	**LBME**	**Loop-back Mode Enable** ● 0 = Disable ● 1 = When the loop-back mode is enable, the following signals are connected internally: **SOUT connected to SIN and SOUT pin fixed at logic 1** **RTS# connected to CTS# and RTS# pin fixed at logic 1**
[3:2]	**Reserved**	**Reserved**
[1]	**RTS#**	**Complement version of RTS# (Request-To-Send) signal**
[0]	**Reserved**	**Reserved**

Note: Only RTS/RTS can be used in this version.

Line Status Control Register (UA_LSR)

Register	Address	R/W	Description	Reset Value
UA_LSR	UA_BA + 0x14	R	Line Status Register	0x6060_6060

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							
7	6	5	4	3	2	1	0
ERR_RX	**TE**	**THRE**	**BII**	**FEI**	**PEI**	**OEI**	**RFDR**

Bits	Descriptions	
[31:8]	**Reserved**	**Reserved**
[7]	**ERR_RX**	**RX FIFO Error** • 0 = RX FIFO works normally • 1 = There is at least one parity error (PE), framing error (FE), or break indication (BI) in the FIFO. ERR_RX is cleared when CPU reads the UA_LSR and if there are no subsequent errors in the RX FIFO.
[6]	**TE**	**Transmitter Empty** • 0 = Either Transmitter Holding Register (**UA_THR** - TX FIFO) or Transmitter Shift Register (**TSR**) are not empty. • 1 = Both UA_THR and TSR are empty.
[5]	**THRE**	**Transmitter Holding Register Empty** • 0 = UA_THR is not empty. • 1 = UA_THR is empty. THRE is set when the last data word of TX FIFO is transferred to Transmitter Shift Register (TSR). The CPU resets this bit when the UA_THR (or TX FIFO) is loaded. This bit also causes the UART to issue an interrupt (INTR_THRE) to the CPU when UA_IER [1]=1.
[4]	**BII**	**Break Interrupt Indicator**

		This bit is set to a logic 1 whenever the received data input is held in the "spacing state" (logic 0) for longer than a full word transmission time (that is, the total time of "start bit" + data bits + parity + stop bits) and is reset whenever the CPU reads the contents of the UA_LSR**.**
[3]	**FEI**	**Framing Error Indicator** This bit is set to logic 1 whenever the received character does not have a valid "stop bit" (that is, the stop bit following the last data bit or parity bit is detected as a logic 0), and is reset whenever the CPU reads the contents of the UA_LSR**.**
[2]	**PEI**	**Parity Error Indicator** This bit is set to logic 1 whenever the received character does not have a valid "parity bit", and is reset whenever the CPU reads the contents of the UA_LSR**.**
[1]	**OEI**	**Overrun Error Indicator** An overrun error will occur only after the RX FIFO is full and the next character has been completely received in the shift register. The character in the shift register is overwritten, but it is not transferred to the RX FIFO. OE is indicated to the CPU as soon as it happens and is reset whenever the CPU reads the contents of the UA_LSR.
[0]	**RFDR**	**RX FIFO Data Ready** • 0 = RX FIFO is empty • 1 = RX FIFO contains at least 1 received data word.

UA_LSR [4:2] (BII, FEI, PEI) are revealed to the CPU when its associated character is at the top of the RX FIFO. These three error indicators are reset whenever the CPU reads the contents of the UA_LSR.

UA_LSR [4:1] (BII, FEI, PEI, OEI) are the error conditions that produce a "receiver line status interrupt" (INTR_RLS) when UA_IER [2] =1. Reading UA_LSR clears INTR_RLS. Writing UA_LSR is a null operation (not suggested).

Modem Status Register (UA_MSR)

Register	Address	R/W	Description	Reset Value
UA_MSR	UA_BA + 0x18	R/W	Modem Status Register	0x0000_0000

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							
7	6	5	4	3	2	1	0
Reserved			**CTS#**	**Reserved**			**DCTS**

Bits	Descriptions	
[4]	**CTS#**	**Complement version of Clear to Send (CTS#) input**
[0]	**DCTS**	**CTS# State Change** This bit is set whenever CTS# input has change state; it will be reset if the CPU reads the MSR.

Note: Only CTS/RTS can be used in this version

Whenever any of MSR[3:0] is set to logic 1, a Modem Status Interrupt is generated if IE[3] = 1. Writing MSR is a null operation (not suggested).

Time out Register (UA_TOR)

Register	Address	R/W	Description	Reset Value
UA_TOR	UA_BA + 0x1C	R/W	Time Out Register	0x0000_0000

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							
7	6	5	4	3	2	1	0
Reserved	**TOIC**						

Bits	Descriptions	
[31:7]	**Reserved**	**Reserved**
[6:0]	**TOIC**	**Time Out Interrupt Comparator** The time out counter resets and starts counting (the counting clock = baud rate) whenever the RX FIFO receives a new data word. Once the content of time out counter (TOUT_CNT) is equal to that of time out interrupt comparator (TOIC), a receiver time out interrupt (INTR_TOUT) is generated if UA_TOR [7] = UA_IER [0] = 1. A new incoming data word or RX FIFO empty clears INTR_TOUT. **Note:** The time out cycles must be larger than the number of cycles to receive one byte. For example, if it needs 10 cycles to receive one byte, the TOIC should be 11 ~ 127.

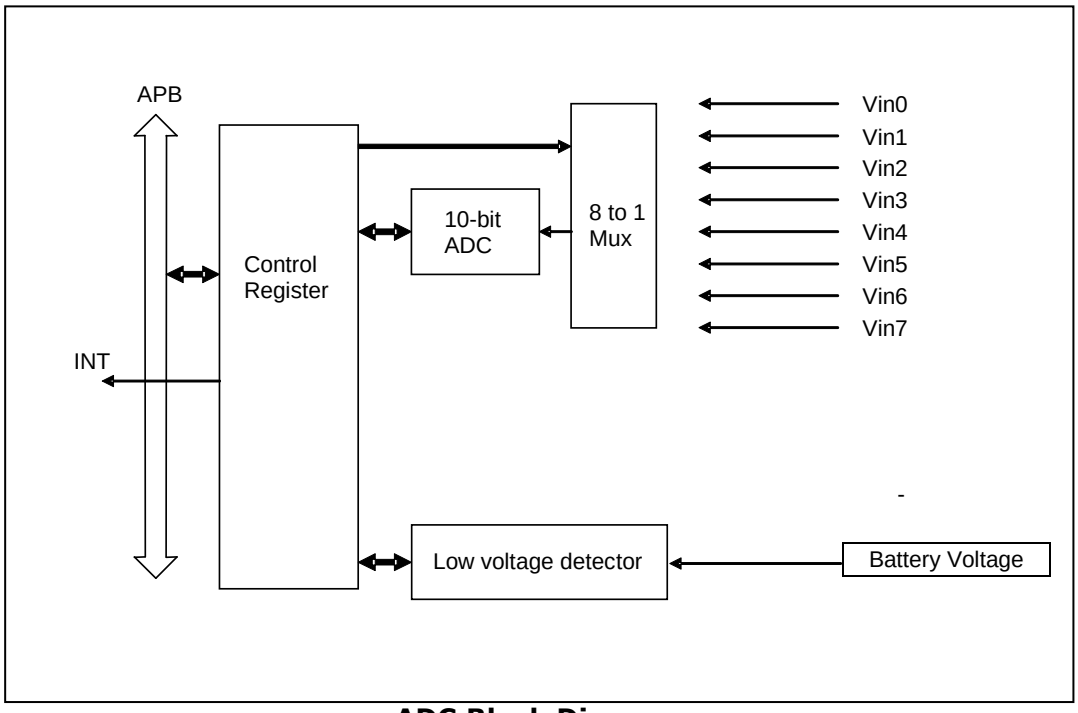
6.19 Analog to Digital Converter

The 10-bit analog to digital converter (ADC) in this chip is a successive approximation type ADC with 8-channel inputs, 2 inputs of them are dedicated for audio recorder. It needs 50 cycles to convert one sample, the maximum input clock to ADC is 25MHz, so the maximum conversion rate is 400K/sec, and the operating voltage range is 3.3V +/- 10%. The power down mode is supported in the ADC.

Beside the 10-bit ADC, an 8 levels voltage detector is included in this chip. The detector result is independent with power supply, and it could give the system a warning signal when battery voltage is lower than an absolute reference voltage.

6.19.1 Features

- Maximum conversion rate: 400K sample per second
- Power supply voltage: 3.3V
- Analog input voltage range: 0 – 3.3 volts
- Standby mode supports
- 8-level voltage detector



ADC Block Diagram

6.19.2 ADC Functional Description

6.19.2.1 Normal Conversion Mode

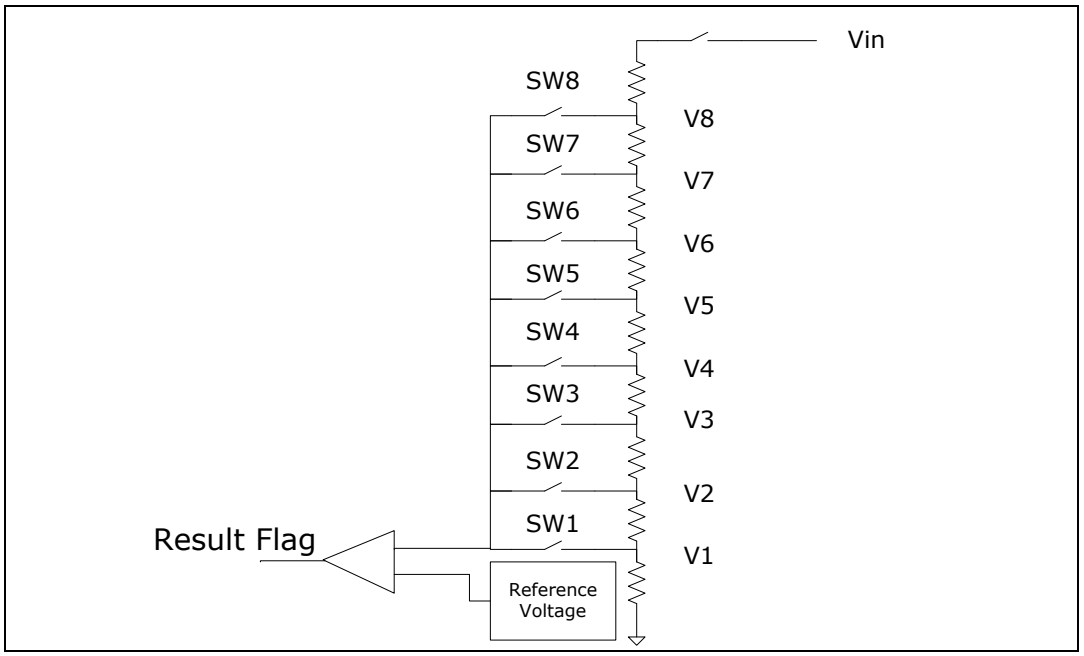
The normal conversion mode is for general purpose ADC, user could use the control register to control the 8 to 1 MUX to select analog input channel, start to conversion, wait interrupt or polling flag to confirm conversion finished, then to read the digital data. The conversion time is 50 ADC input clocks for each sample.

6.19.2.2 Standby mode

A standby mode is provided for ADC. When the ADC enable bit is cleared to 0, and the ADC clock is disable, the ADC enter standby mode. In the standby mode, the consumed current from AVDD will be less than 5uA. Note that the last conversion data is not cleared.

6.19.2.3 Voltage detector

The architecture of the voltage detector is shown as in the following figure.



By control the switch sw1, sw2, sw3, sw4, sw5, sw6, sw7 and sw8, to select the voltage V1, V2, V3, V4, V5, V6, V7 or V8 to be compared to reference voltage which will not be influenced by supply voltage or temperature.

6.19.2.4 Recording path

The audio recording path converts the audio analog to digital data by means of the ADC hardware. When the recording path is in usage, other data-conversion ADC function can't operate.

6.19.3 ADC Control Register Mapping

R: read only, **W**: write only, **R/W**: both read and write, **C**: Only value 0 can be written

Register	Address	R/W	Description	Reset Value
ADC_BA = 0xB800_1000				
ADC_CON	ADC_BA+0x000	R/W	ADC control register	0x0000_0000

Reserved	ADC_BA+0x004			
Reserved	ADC_BA+0x008			
ADC_XDATA	ADC_BA+0x00C	R	ADC XDATA register	0x0000_0000
Reserved	ADC_BA+0x010			
LV_CON	ADC_BA+0x014	R/W	Low Voltage Detector Control register	0x0000_0000
LV_STS	ADC_BA+0x018	R/W	Low Voltage Detector Status register	0x0000_0000
AUDIO_CON	ADC_BA+0x01C	R/W	Audio control register	0x0000_0000
AUDIO_BUF0	ADC_BA+0x020	R/W	Audio data buffer register	0x0000_0000
AUDIO_BUF1	ADC_BA+0x024	R/W	Audio data buffer register	0x0000_0000
AUDIO_BUF2	ADC_BA+0x028	R/W	Audio data buffer register	0x0000_0000
AUDIO_BUF3	ADC_BA+0x02C	R/W	Audio data buffer register	0x0000_0000

6.19.4 ADC Control Register Description

ADC Control Register (ADC_CON)

Register	Address	R/W	Description	Reset Value
ADC_CON	ADC_BA+0x000	R/W	ADC control register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved	**LVD_INT_EN**	**ADC_INT_EN**	**Reserved**	**LVD_INT**	**ADC_INT**	**ADC_EN**	**ADC_RST**
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
ADC_MODE		**ADC_CONV**	**ADC_READ_CONV**	**ADC_MUX**			**ADC_DIV**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ADC_DIV							**ADC_FINISH**

Bits	Descriptions	
[31:23]	**Reserved**	**Reserved**
[22]	**LVD_INT_EN**	**Low voltage detector interrupt enable bit** If LVD_INT_EN=0, The LVD interrupt is disable If LVD_INT_EN=1, The LVD interrupt is enable
[21]	**ADC_INT_EN**	**ADC interrupt enable bit** If ADC_INT_EN=0, The ADC interrupt is disable If ADC_INT_EN=1, The ADC interrupt is enable
[20]	**Reserved**	**Reserved**
[19]	**LVD_INT**	**Low voltage detector (LVD) interrupt status bit** If LV_INT=0, The LVD interrupt status is cleared If LV_INT=1, The LVD is in interrupt state and write 0 to clear it
[18]	**ADC_INT**	**ADC interrupt status bit** If ADC_INT=0, The ADC interrupt status is cleared If ADC_INT=1, The ADC is in interrupt state and write 0 to clear it
[17]	**ADC_EN**	**ADC block enable bit** If ADC_EN=0, The ADC block is disable If ADC_EN=1, The ADC block is enable
[16]	**ADC_RST**	**ADC reset control bit** If ADC_RST=1, the ADC block is at reset mode If ADC_RST=0, the ADC block is at normal mode
[15:14]	**ADC_MODE**	**The conversion mode control bits** If ADC_MODE=00, normal conversion mode is selected If ADC_MODE**=others, reserved**

[13]	**ADC_CONV**	**ADC conversion control bit** If ADC_CONV=1, inform ADC to converse, when conversion finished, this bit will be auto clear If ADC_CONV=0, the ADC no action, and this only could be cleared by hardware This bit can be wrote 1 ONLY.
[12]	**ADC_READ_CONV**	This bit control if next conversion start after ADC_XDATA register is read in normal conversion mode. If ADC_READ_CONV=1, start next conversion after the ADC_XDATA is read, and ignore the ADC_CONV bit. If ADC_READ_CONV=0, after the ADC_XDATA is read, the ADC no action
[11:9]	**ADC_MUX**	**These bits select ADC input from the 8 analog inputs in normal conversion mode.** ADC_MUX=000, not available in normal data conversion. ADC_MUX=001, not available in normal data conversion. ADC_MUX=010, select AIN2 ADC_MUX=011, select AIN3 ADC_MUX=100, select AIN4 ADC_MUX=101, select AIN5 ADC_MUX=110, select AIN6 ADC_MUX=111, select AIN7 The ADC_MUX bits are read/write. **AIN0 and AIN1 channel are differential inputs for audio recorder only.** **When in Audio Recording operation (AUDIO_CON[1] / AUDIO_EN = 1), only AIN0 and AIN1 are dedicated inputs, and ADC_MUX value is not effective in this operation.**
[8:1]	Reserved	
[0]	**ADC_FINISH**	ADC_FINISH (**Read Only**) 0 = Converting 1 = ADC conversion finish

ADC X data buffer (ADC_XDATA)

Register	Address	R/W	Description	Reset Value
ADC_XDATA	ADC_BA+0x00C	R	ADC X data buffer	0x0000_0000

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						**ADC_XDATA**	
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ADC_XDATA							

Bits	Descriptions	
[31:10]	**Reserved**	**Reserved**
[9:0]	**ADC_XDATA**	**ADC Data Buffer** When normal conversion mode, the conversion data is always put at this register.

Low Voltage Detector Control Register (LV_CON)

Register	Address	R/W	Description	Reset Value
LV_CON	ADC_BA+0x014	R/W	Low voltage detector control register	0x0000_0000

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							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved				**LV_EN**	**SW_CON**		

Bits	Descriptions	
[31:4]	**Reserved**	**Reserved**
[3]	**LV_EN**	**Low voltage detector enable control pin** If LV_EN = 0, low voltage detector is disable If LV_EN = 1, low voltage detector is enable
[2:0]	**SW_CON**	**The low voltage detector voltage level switch control bits** If SW_CON = 000, SW1 is close, others are open. If SW_CON = 001, SW2 is close, others are open. If SW_CON = 010, SW3 is close, others are open. If SW_CON = 011, SW4 is close, others are open. If SW_CON = 100, SW5 is close, others are open. If SW_CON = 101, SW6 is close, others are open. If SW_CON = 110, SW7 is close, others are open. If SW_CON = 111, SW8 is close, others are open. The relationship of SW_CON setting vs. detected voltage level

SW_CON	Voltage Level (V)
000	2.0
001	2.1
010	2.2
011	2.3
100	2.4
101	2.5
110	2.6
111	2.7

Low Voltage Detector Status Register (LV_STS)

Register	Address	R/W	Description	Reset Value
LV_STS	ADC_BA+0x018	R/W	The status register of low voltage detector	0x0000_0000

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							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							**LV_status**

Bits	Descriptions	
[31:1]	**Reserved**	**Reserved**
[0]	**LV_status**	**Low voltage detector status pin (Read Only)** • If LV_status = 0, the compared voltage is higher than reference voltage • If LV_status = 1, the compared voltage is lower than reference voltage

Audio control register (AUDIO_CON)

Register	Address	R/W	Description	Reset Value
AUDIO_CON	ADC_BA+0x01C	R/W	Audio control, status and data register	0x0000_0000

31	30	29	28	27	26	25	24
AUD_INT_MODE		**OP_OFFSET**				**Reserved**	
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved						**AUD_INT**	**VOL_EN**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
AUDIO_VOL					**AUDIO_HPEN**	**AUDIO_EN**	**AUDIO_RESET**

Bits	Descriptions	
[31:30]	**AUD_INT_MODE**	**Audio interrupt mode selection** 2'b00: If AUD_INT=1, the recording for one sample is finished. 2'b01: If AUD_INT=1, the recording for two samples are finished. 2'b10: If AUD_INT=1, the recording for four samples are finished. 2'b11: If AUD_INT=1, the recording for eight samples are finished.
[29:26]	**OP_OFFSET**	**Reserved** **For engineering used. Keep the setting value to '0'.**
[25:10]	**Reserved**	**Reserved**
[9]	**AUD_INT**	**Audio interrupt flag bits** If AUD_INT=0, the recording for 1/2/4/8 samples are not finished. If AUD_INT=1, the recording for 1/2/4/8 samples are finished. Write 0 to clear it Note: This flag can be set by above hardware event if AUDIO_EN = 1. And when it is set an interrupt signal is asserted to the interrupt controller through ADC interrupt source.
[8]	**VOL_EN**	**Volume control enable bit** If VOL_EN = 0, the hardware open the volume control path and open the recording path. If VOL_EN = 1, the hardware enable the volume control path and enable the recording path.
[7:3]	**AUDIO_VOL**	**Volume control bits** AUDIO_VOL[4:0] = [0, 1, 2, … , 31] indicate the volume from [0dB, 1dB, …, 31dB] respectively
[2]	**AUDIO_HPEN**	**Record path high pass enable bit** If AUDIO_HPEN = 0, the digital high pass filter will be bypassed. If AUDIO_HPEN = 0, the digital high pass filter will be enable.
[1]	**AUDIO_EN**	**Record operation enable bit**

		If AUDIO_EN = 0, the hardware digital decimation filter will be disabled. If AUDIO_EN = 1, the hardware digital decimation filter will be enabled.
[0]	**AUDIO_RESET**	**Digital filter reset bit** If AUDIO_RESET = 0, the digital filter is not reset. If AUDIO_RESET = 1, the digital filter is on the reset state.

Audio control register (AUDIO_BUF0)

Register	Address	R/W	Description	Reset Value
AUDIO_BUF0	ADC_BA+0x020	R/W	Audio data register	0x0000_0000

31	30	29	28	27	26	25	24
AUDIO_DATA1							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
AUDIO_DATA1							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
AUDIO_DATA0							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
AUDIO_DATA0							

Bits	Descriptions	
[31:16]	**AUDIO_DATA1**	**Converted audio data1 at buffer0 (Read Only)** 16-bit digital audio data in 2's compliment format.
[15:0]	**AUDIO_DATA0**	**Converted audio data0 at buffer0 (Read Only)** 16-bit digital audio data in 2's compliment format.

Audio control register (AUDIO_BUF1)

Register	Address	R/W	Description	Reset Value
AUDIO_BUF1	ADC_BA+0x024	R/W	Audio data register	0x0000_0000

31	30	29	28	27	26	25	24
AUDIO_DATA3							
23	22	21	20	19	18	17	16
AUDIO_DATA3							
15	14	13	12	11	10	9	8
AUDIO_DATA2							
7	6	5	4	3	2	1	0
AUDIO_DATA2							

Bits	Descriptions	
[31:16]	**AUDIO_DATA3**	**Converted audio data3 at buffer1 (Read Only)** 16-bit digital audio data in 2's compliment format.
[15:0]	**AUDIO_DATA2**	**Converted audio data2 at buffer1 (Read Only)** 16-bit digital audio data in 2's compliment format.

Audio control register (AUDIO_BUF2)

Register	Address	R/W	Description	Reset Value
AUDIO_BUF2	ADC_BA+0x028	R/W	Audio data register	0x0000_0000

31	30	29	28	27	26	25	24
AUDIO_DATA5							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
AUDIO_DATA5							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
AUDIO_DATA4							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
AUDIO_DATA4							

Bits	Descriptions	
[31:16]	**AUDIO_DATA5**	**Converted audio data5 at buffer2 (Read Only)** 16-bit digital audio data in 2's compliment format.
[15:0]	**AUDIO_DATA4**	**Converted audio data4 at buffer2 (Read Only)** 16-bit digital audio data in 2's compliment format.

Audio control register (AUDIO_BUF3)

Register	Address	R/W	Description	Reset Value
AUDIO_BUF3	ADC_BA+0x02C	R/W	Audio data register	0x0000_0000

31	30	29	28	27	26	25	24
AUDIO_DATA7							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
AUDIO_DATA7							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
AUDIO_DATA6							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
AUDIO_DATA6							

Bits	Descriptions	
[31:16]	**AUDIO_DATA7**	**Converted audio data7 at buffer3 (Read Only)** 16-bit digital audio data in 2's compliment format.
[15:0]	**AUDIO_DATA6**	**Converted audio data6 at buffer3 (Read Only)** 16-bit digital audio data in 2's compliment format.

7 Electrical Characteristics

7.1 Absolute Maximum Ratings

Ambient temperature	0 °C~ 105 °C
Storage temperature	-40 °C ~ 125 °C
Voltage on any pin	-0.3V ~ 3.6V
Power supply voltage (Core logic)	-0.5V ~ 2.5V
Power supply voltage (IO Buffer)	-0.5V ~ 4.6V
Injection current (latch-up testing)	100mA
Crystal Frequency	2MHz ~ 20MHz

7.2 DC Specifications

Parameter		Condition	Min	Typ	Max	Unit
V_{DD33}	IO Post-Driver Voltage		3.00	3.30	3.60	V
V_{BAT}	RTC Voltage		1.2	1.5	1.8	V
VCC_CORE	Core Logic, Pre-Driver Voltage		1.62	1.80	1.98	V
V_{IL}	Input Low Voltage		-0.3		0.8	V
V_{IH}	Input High Voltage		2.0		3.6	V
V_T	Threshold Point		1.30	1.36	1.42	V
V_T^+	Schmitt trig. Low to High threshold point		1.51	1.56	1.60	V
V_T^-	Schmitt trig. High to Low threshold point		1.15	1.21	1.25	V
I_{CC}	Supply Current	FCPU = 81MHz			27	mA
I_I	Input Leakage Current	VI = 3.3V or 0V	-1		1	uA
I_{OZ}	Tri-State Output Leakage Current	VO = 3.3V or 0V	-1		1	uA
V_{OL}	Output Low Voltage	I_{OL} = 4/8/12 mA			0.4	V
V_{OH}	Output High Voltage	I_{OL} = 4/8/12 mA	2.4			V
I_{OL}	Low Level Output Current, 4mA	V_{OL} = 0.4V	4			mA
	Low Level Output Current, 8mA	V_{OL} = 0.4V	8			mA
	Low Level Output Current, 12mA	V_{OL} = 0.4V	12			mA
	Low Level Output Current, 16mA	V_{OL} = 0.4V	16			
I_{OH}	High Level Output Current, 4mA	V_{OH} = 2.4V	-4			mA
	High Level Output Current, 8mA	V_{OH} = 2.4V	-8			mA
	High Level Output Current, 12mA	V_{OH} = 2.4V	-12			mA
	High Level Output Current, 16mA	V_{OH} = 2.4V	-16			
R_{PU}	Pull-up Resistor		38	54	84	KΩ

7.3 AC Specifications

7.3.1 Audio DAC Characteristic

Parameter	Min	Typ	Max	Unit
Operating Voltage	3.0	3.3	3.6	V
Maximum Output Voltage Amplitude(R_L = 50 KΩ)		1.8		V_{p-p}
THD + N(R_L = 50 KΩ, f = 1KHz)		0.025		%
SNR		85		dB

7.3.2 ADC Characteristic

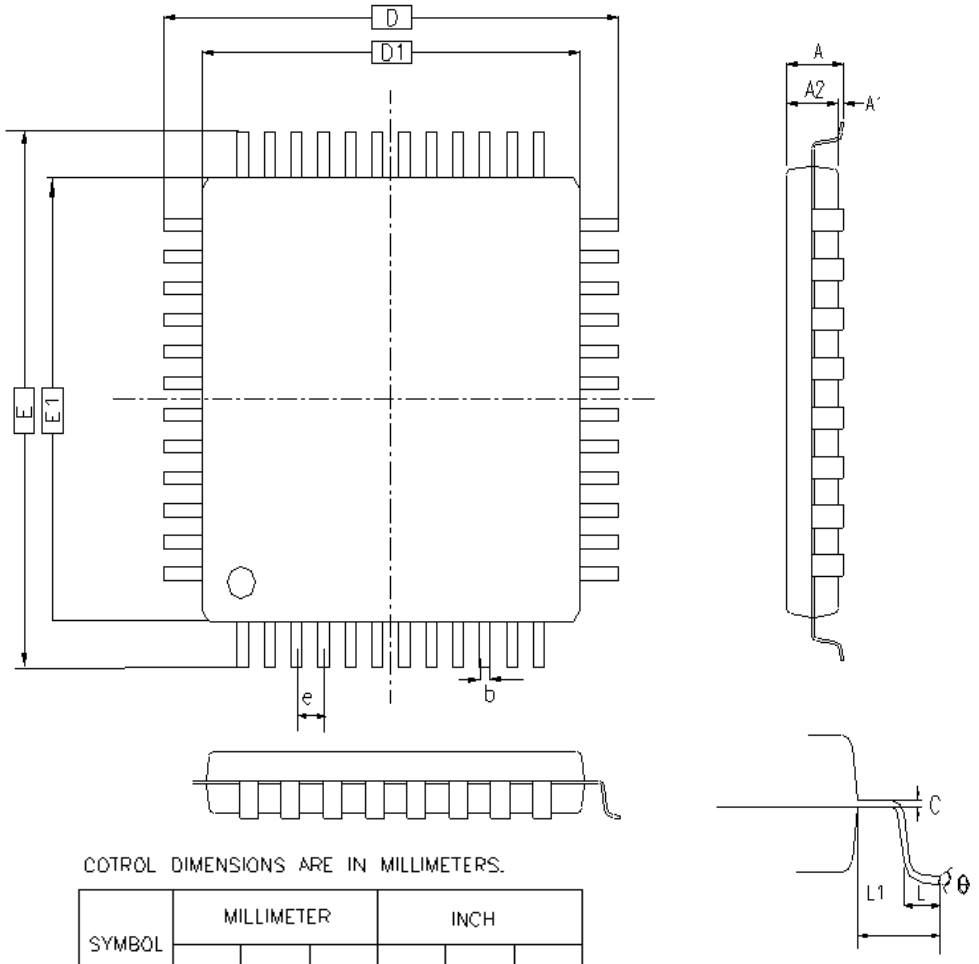
Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Operating Voltage	V_{DD}		3.0	3.3	3.6	V
Operating Current	I_{DD}			330		uA
Programmable gain			0		62	dB
Sampling rate	SR			16		KSPS
Input Resistance	M_{ICP}	Gain set to 0 dB		30		KΩ
		Gain set to 62 dB		48		Ω
	M_{ICM}			60		KΩ
THD + N				50		dB
SNR				60		dB

7.3.3 Voice Recorder Characteristic

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Operating Voltage	V_{DD}		3.0	3.3	3.6	V
Operating Current	I_{DD}			330		uA
Reference Voltage	V_{REF}		2		VDD	V
Reference Current	I_{REF}	V_{REF} = 3.3V		250		uA
		V_{REF} = 2V		150		uA
Resolution					10	bit
Conversion time			2.5		10	us
Sample rate					400	KHz
Integral non-linear error	I_{NL}				±2	LSB
Differential non-linearity	D_{NL}				±1	LSB

8 Package Specifications

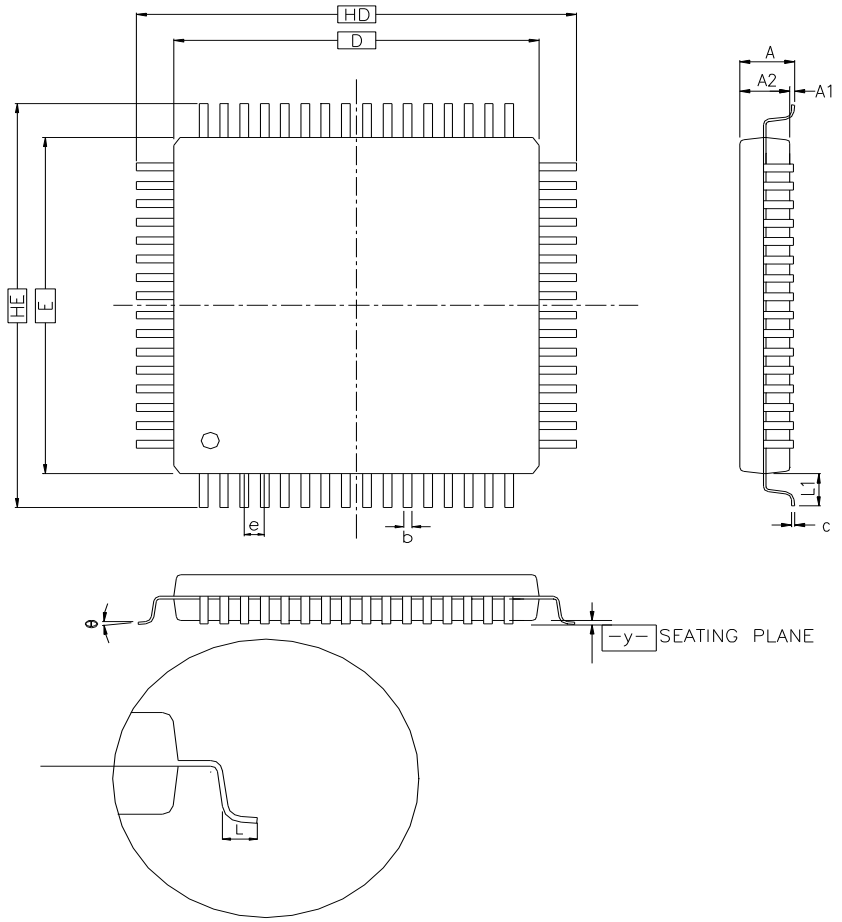
LQFP-48 (7x7x1.4mm footprint 2.0mm)



COTROL DIMENSIONS ARE IN MILLIMETERS.

SYMBOL	MILLIMETER			INCH		
	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.
A	—	—	1.60	—	—	0.063
A1	0.05	0.10	0.15	0.002	0.004	0.006
A2	1.35	1.40	1.45	0.053	0.055	0.057
D1	6.90	7.00	7.10	0.272	0.276	0.260
E1	6.90	7.00	7.10	0.272	0.276	0.260
e	0.35	0.50	0.65	0.014	0.020	0.260
D	8.9	9.00	9.10	0.350	0.354	0.358
E	8.9	9.00	9.10	0.350	0.354	0.358
L	0.45	0.60	0.75	0.018	0.024	0.030
L1	—	1.00	—	—	0.039	—
C	0.09	—	0.20	0.0035	—	0.0079
θ	0°	—	7°	0°	—	7°
b	0.17	0.22	0.27	0.007	0.0087	0.011

LQFP-64 (10x10x1.4mm)



Symbol	Dimension in mm		
	Min	Nom	Max
A	—	—	1.60
A_1	0.05	—	0.15
A_2	1.35	1.40	1.45
b	0.17	0.20	0.27
c	0.09	—	0.20
D	—	10.00	—
E	—	10.00	—
e	—	0.50	—
H_D	—	12.00	—
H_E	—	12.00	—
L	0.45	0.60	0.75
L_1	—	1.00	—
y	—	0.10	—
θ	0°	3.5°	7°

Important Notice

Nuvoton products are not designed, intended, authorized or warranted for use as components in systems or equipment intended for surgical implantation, atomic energy control instruments, airplane or spaceship instruments, transportation instruments, traffic signal instruments, combustion control instruments, or for other applications intended to support or sustain life. Furthermore, Nuvoton products are not intended for applications wherein failure of Nuvoton products could result or lead to a situation wherein personal injury, death or severe property or environmental damage could occur.

Nuvoton customers using or selling these products for use in such applications do so at their own risk and agree to fully indemnify Nuvoton for any damages resulting from such improper use or sales.

Please note that all data and specifications are subject to change without notice.
All the trademarks of products and companies mentioned in this datasheet belong to their respective owners.