

W7500 Datasheet Manual

Version 1.0.6



<http://www.wiznet.co.kr>

Table of Contents

| | | |
|----------|--|-----------|
| 1 | Documentation conventions | 10 |
| 1.1 | Glossary | 10 |
| 1.2 | Register Bit Conventions | 12 |
| 2 | System and memory overview | 13 |
| 2.1 | System architecture | 13 |
| 2.2 | Memory organization | 14 |
| 2.2.1 | Introduction | 14 |
| 2.2.2 | Memory map..... | 15 |
| 3 | System configuration controller (SYSCFG)..... | 16 |
| 3.1 | Introduction | 16 |
| 4 | Interrupt and events | 16 |
| 4.1 | Introduction | 16 |
| 4.2 | Interrupt assignments | 16 |
| 4.3 | Event | 17 |
| 5 | Power supply | 17 |
| 5.1 | Introduction | 17 |
| 5.2 | Voltage regulator..... | 17 |
| 5.3 | Power supply supervisor | 18 |
| 5.4 | Low-power modes..... | 18 |
| 5.4.1 | Sleep mode | 18 |
| 5.4.2 | Peripheral clock gating | 19 |
| 6 | System tick timer | 19 |
| 6.1 | Introduction | 19 |
| 6.2 | Features | 19 |
| 6.3 | Functional description | 19 |
| 7 | TCIPCore Offload Engine (TOE) | 20 |
| 7.1 | Introduction | 20 |
| 7.2 | Features | 20 |
| 7.3 | Functional description | 21 |
| 7.4 | TOE Memory map..... | 21 |
| 7.4.1 | Common register map | 23 |
| 7.4.2 | Socket register map..... | 23 |
| 7.4.3 | Memory | 24 |
| 8 | Bootng Sequence | 26 |
| 9 | Embedded Flash memory..... | 27 |
| 9.1 | Flash main features..... | 27 |
| 9.2 | Flash memory functional description..... | 27 |

| | | |
|-----------|--|-----------|
| 9.2.1 | Flash memory organization | 27 |
| 9.2.2 | Read operations..... | 29 |
| 9.2.3 | Flash erase operations..... | 30 |
| 9.2.4 | Flash program operation | 32 |
| 9.3 | Memory protection..... | 33 |
| 9.3.1 | Read protection..... | 33 |
| 9.3.2 | Write protection | 33 |
| 10 | Clock Reset generator (CRG)..... | 34 |
| 10.1 | Introduction | 34 |
| 10.2 | Features | 34 |
| 10.2.1 | Reset | 34 |
| 10.2.2 | Clock..... | 34 |
| 10.3 | Functional description | 35 |
| 10.3.1 | External Oscillator Clock..... | 35 |
| 10.3.2 | RC oscillator clock..... | 36 |
| 10.3.3 | PLL | 36 |
| 10.3.4 | Generated clock | 36 |
| 11 | Random number generator (RNG) | 37 |
| 11.1 | Introduction | 37 |
| 11.2 | Features | 37 |
| 11.3 | Functional description | 37 |
| 11.3.1 | Operation RNG | 38 |
| 12 | Alternate Function Controller (AFC) | 39 |
| 12.1 | Introduction | 39 |
| 12.2 | Features | 39 |
| 12.3 | Functional description | 39 |
| 13 | External Interrupt (EXTI) | 41 |
| 13.1 | Introduction | 41 |
| 13.2 | Features | 41 |
| 13.3 | Functional description | 41 |
| 14 | Pad Controller (PADCON) | 43 |
| 14.1 | Introduction | 43 |
| 14.2 | Features | 43 |
| 14.3 | Functional description | 43 |
| 15 | General-purpose I/Os(GPIO)..... | 44 |
| 15.1 | Introduction | 44 |
| 15.2 | Features | 44 |
| 15.3 | Functional description | 45 |
| 15.3.1 | Masked access..... | 46 |

| | |
|---|-----------|
| 16 Direct memory access controller (DMA) | 48 |
| 16.1 Introduction | 48 |
| 16.2 Features | 48 |
| 16.3 Functional description | 48 |
| 16.3.1 DMA request mapping | 49 |
| 16.3.2 DMA arbitration | 49 |
| 16.3.3 DMA cycle types | 49 |
| 17 Analog-to-digital converter (ADC) | 53 |
| 17.1 Introduction | 53 |
| 17.2 Features | 53 |
| 17.3 Functional description | 54 |
| 17.3.1 Operation ADC with non-interrupt | 54 |
| 17.3.2 Operation ADC with interrupt | 56 |
| 18 Pulse-Width Modulation (PWM) | 57 |
| 18.1 Introduction | 57 |
| 18.2 Features | 57 |
| 18.3 Functional description | 58 |
| 18.3.1 Timer/Counter control | 58 |
| 18.3.2 Timer/Counter | 58 |
| 18.3.3 PWM mode | 62 |
| 18.3.4 Interrupt | 63 |
| 18.3.5 Dead zone generation | 63 |
| 18.3.6 Capture event | 65 |
| 18.3.7 How to set the PWM | 66 |
| 19 Dual timers | 67 |
| 19.1 Introduction | 67 |
| 19.2 Features | 67 |
| 19.3 Functional description | 68 |
| 19.3.1 Clock and clock enable | 68 |
| 19.3.2 Timer size | 68 |
| 19.3.3 Prescaler | 68 |
| 19.3.4 Repetition mode | 68 |
| 19.3.5 Interrupt | 69 |
| 19.3.6 Operation | 69 |
| 19.3.7 How to set the dual timers | 70 |
| 20 Watchdog timer | 71 |
| 20.1 Introduction | 71 |
| 20.2 Features | 71 |
| 20.3 Functional description | 71 |

| | | |
|-----------|---|-----------|
| 20.3.1 | Clock..... | 71 |
| 20.3.2 | Interrupt and reset request..... | 71 |
| 21 | Inter-integrated circuit interface (I2C)..... | 72 |
| 21.1 | Introduction..... | 72 |
| 21.2 | Features | 72 |
| 22 | Universal Asynchronous Receive Transmit(UART0,1) | 73 |
| 22.1 | Introduction..... | 73 |
| 22.2 | Features | 73 |
| 22.3 | Functional description | 73 |
| 22.3.1 | Baud rate calculation..... | 75 |
| 22.3.2 | Data transmission..... | 76 |
| 22.3.3 | Data receive..... | 76 |
| 22.3.4 | Hardware flow control..... | 77 |
| 23 | Universal Asynchronous Receive Transmit(UART2) | 79 |
| 23.1 | Introduction..... | 79 |
| 23.2 | Feature..... | 79 |
| 23.3 | Functional description | 79 |
| 23.3.1 | Baud rate calculation..... | 79 |
| 24 | Synchronous Serial Port (SSP) | 80 |
| 24.1 | Introduction..... | 80 |
| 24.2 | Features | 80 |
| 24.3 | Functional description | 81 |
| 24.3.1 | Clock prescaler | 81 |
| 24.3.2 | Transmit FIFO | 82 |
| 24.3.3 | Receive FIFO..... | 82 |
| 24.3.4 | Interrupt generation logic | 82 |
| 24.3.5 | DMA interface | 83 |
| 24.3.6 | Interface reset | 84 |
| 24.3.7 | Configuring the SSP | 84 |
| 24.3.8 | Enable PrimeCell SSP operation..... | 85 |
| 24.3.9 | Clock ratios | 85 |
| 24.3.10 | Programming the SSPCR0 Control Register..... | 86 |
| 24.3.11 | Programming the SSPCR1 Control Register..... | 86 |
| 24.3.12 | Frame format | 87 |
| 24.3.13 | Texas Instruments synchronous serial frame format | 88 |
| 24.3.14 | Motorola SPI frame format..... | 89 |
| 24.3.15 | National Semiconductor Microwire frame format | 95 |
| 24.3.16 | Master and Slave configurations | 97 |
| 24.3.17 | SSP Flow chart | 98 |

| | |
|--|------------|
| 25 Electrical Characteristics | 100 |
| 25.1 Absolute maximum ratings | 100 |
| 25.1.1 Voltage Characteristics | 100 |
| 25.1.2 Current Characteristics | 100 |
| 25.1.3 Thermal Characteristics | 100 |
| 25.2 Operating conditions | 101 |
| 25.2.1 General Operating Conditions | 101 |
| 25.2.2 Supply Current Characteristics | 101 |
| 25.3 I/O PAD Characteristics | 102 |
| 25.3.1 DC Specification | 102 |
| 25.4 Flash memory | 102 |
| 25.5 Electrical Sensitivity Characteristics | 102 |
| 25.5.1 Electrostatic discharge (ESD) | 102 |
| 25.5.2 Static latch-up | 102 |
| 25.6 ADC Characteristics | 103 |
| 25.6.1 ADC Electrical characteristics | 103 |
| 25.6.2 ADC Transform function description | 104 |
| 25.7 I2C interface Characteristics | 104 |
| 25.8 SSP Interface Characteristics | 105 |
| 26 Package Characteristics | 106 |
| 26.1 Package dimension information | 106 |
| 26.2 Package footprint information | 107 |
| Document History Information | 108 |

List of table

| | |
|---|-----|
| Table 1 W7500 interrupt assignments | 16 |
| Table 2 W7500 sleep mode summary | 19 |
| Table 3. Offset Address for Common Register | 23 |
| Table 4. Offset Address in Socket n Register Block (n = 0,...,7, where n is Socket number) | 24 |
| Table 5 operation of mode selection | 26 |
| Table 6 description of Flash memory | 27 |
| Table 7 External oscillator clock sources | 35 |
| Table 8 functional description table | 39 |
| Table 9 Summary of the DMA requests for each channel | 49 |
| Table 10 DMA trigger points for the transmit and receive FIFOs. | 84 |
| Table 11 Voltage characteristics | 100 |
| Table 12 Current characteristics | 100 |
| Table 13 Thermal Characteristics | 100 |
| Table 14 General operating conditions | 101 |
| Table 15 Normal operation supply current | 101 |
| Table 16 Sleep mode supply current | 101 |
| Table 17 Deep sleep mode supply current | 101 |
| Table 18 DC specification of PAD | 102 |
| Table 19 Flash memory Reliability Characteristics | 102 |
| Table 20 Electrostatic discharge (ESD) | 102 |
| Table 21 Static latch-up | 102 |
| Table 22 ADC electrical characteristics | 103 |
| Table 23 I2C characteristics | 104 |
| Table 24 SSP characteristics | 105 |
| Table 25 Package mechanical data | 106 |

List of figures

| | |
|--|----|
| Figure 1 W7500 System Architecture | 13 |
| Figure 2 W7500 memory map..... | 15 |
| Figure 3 POR reset waveform..... | 18 |
| Figure 4 TOE block diagram | 21 |
| Figure 5. Register & Memory Organization | 22 |
| Figure 6. operation of boot code | 26 |
| Figure 7. Flash reading sequence | 30 |
| Figure 8. Flash erase operations | 31 |
| Figure 9. main Flash memory programming sequence | 32 |
| Figure 10 CRG block diagram | 35 |
| Figure 11. Random Number Generator block diagram | 37 |
| Figure 12. Flow chart of RNG operation | 38 |
| Figure 13. External Interrupt diagram | 42 |
| Figure 14. function schematic of digital I/O pad | 43 |
| Figure 15. function schematic of digital/analog mux IO pad | 43 |
| Figure 16. GPIO block diagram | 45 |
| Figure 17. GPIO Flow chart | 46 |
| Figure 18. MASK LOWBYTE access | 47 |
| Figure 19 MASK HIGHBYTE access..... | 47 |
| Figure 20. DMA Block diagram | 48 |
| Figure 21. DMA ping pong cycle | 52 |
| Figure 22. ADC block diagram | 54 |
| Figure 23. The ADC operation flowchart with non-interrupt..... | 55 |
| Figure 24. The ADC operation flowchart with interrupt | 56 |
| Figure 25. PWM block diagram | 57 |
| Figure 26. Periodic mode..... | 58 |
| Figure 27. one-shot mode | 59 |
| Figure 28. Up-count mode | 59 |
| Figure 29. Down-count mode | 59 |
| Figure 30 Counter mode with rising edge | 60 |
| Figure 31 Counter mode with falling edge | 60 |
| Figure 32 Counter mode with rising and falling edge | 60 |
| Figure 33 Timer/Counter timing diagram with match interrupt | 61 |
| Figure 34 Timer/Counter timing diagram with overflow interrupt | 61 |
| Figure 35 The PWM output up to match register..... | 62 |
| Figure 36 The PWM output up to limit register..... | 63 |
| Figure 37 PWM waveform with dead zone time | 64 |

| | |
|--|-----|
| Figure 38 PWM waveform with dead zone counter | 64 |
| Figure 39 Capture event with no interrupt clear | 65 |
| Figure 40 Capture event with interrupt clear | 65 |
| Figure 41. The PWM setting flow | 66 |
| Figure 42 Block diagram of Dualtimer | 67 |
| Figure 43 The Dual timer setting flow | 70 |
| Figure 44. Watchdog timer operation flow diagram..... | 72 |
| Figure 45 UART0,1 Block diagram..... | 74 |
| Figure 46 UART character frame | 74 |
| Figure 47 UART0,1 divider flow chart | 75 |
| Figure 48 UART0,1 Initial setting flow chart | 75 |
| Figure 49 Transmit and Receive data flow chart | 76 |
| Figure 50 Hardware flow control description..... | 77 |
| Figure 51 CTS Functional Timing..... | 77 |
| Figure 52 Algorithm for setting CTS/RTS flowchart..... | 78 |
| Figure 53. UART2 divider flow chart | 79 |
| Figure 54. UART2 Initial setting flow chart | 80 |
| Figure 55. SSP block diagram | 81 |
| Figure 56. DMA transfer waveforms | 84 |
| Figure 57. Texas Instruments synchronous serial frame format, single transfer | 88 |
| Figure 58. Texas Instruments synchronous serial frame format, continuous transfers | 89 |
| Figure 59. Motorola SPI frame format, single transfer, with SPO=0 and SPH=0 | 90 |
| Figure 60. Motorola SPI frame format, continuous transfers, with SPO=0 and SPH=0 | 90 |
| Figure 61. Motorola SPI frame format, single and continuous transfers, with SPO=0 and SPH=1 | 91 |
| Figure 62. Motorola SPI frame format, single transfer, with SPO=1 and SPH=0 | 92 |
| Figure 63. Motorola SPI frame format, continuous transfers, with SPO=1 and SPH=0 | 93 |
| Figure 64. Motorola SPI frame format, single and continuous transfers, with SPO=1 and SPH=1 | 94 |
| Figure 65. National Semiconductor Microwire frame format, single transfer | 95 |
| Figure 66. National Semiconductor Microwire frame format, continuous transfers .. | 97 |
| Figure 67. PrimeCell SSP master coupled to an SPI slave..... | 97 |
| Figure 68. SPI master coupled to a PrimeCell SSP slave | 98 |
| Figure 69. how to setting TI or Microwire mode flow chart..... | 98 |
| Figure 70. how to setting SPI mode flow chart..... | 99 |
| Figure 71. ADC transform function | 104 |
| Figure 72. I2C bus AC waveform | 105 |
| Figure 73 Package Dimension Information | 106 |
| Figure 74 Footprint Information | 107 |

1 Documentation conventions

1.1 Glossary

| | |
|--------|---|
| ARP | Address Resolution Protocol |
| AHB | Advanced High-performance Bus |
| AMBA | Advanced Microcontroller Bus Architecture |
| APB | Advanced Peripheral Bus |
| AFC | Alternate Function Controller |
| ADC | Analog-to-Digital Converter |
| BOD | BrownOut Detection |
| CPU | Central Processing Unit |
| CRG | Clock Reset generator |
| DMA | Direct Memory Access |
| EOP | End Of Packet |
| EXTINT | External Interrupt |
| GPIO | General Purpose Input/Output |
| IrDA | Infrared Data Association |
| I/O | Input/Output |
| ICMP | Internet Control Message Protocol |
| IGMP | Internet Group Management Protocol |
| IPv4 | Internet Protocol version 4 |
| IRQ | interrupt request |
| NMI | NonMaskable Interrupt |
| PADCON | Pad Controller |
| PLL | Phase-Locked Loop |
| PHY | Physical Layer |
| PPPoE | Point-to-Point Protocol over Ethernet |
| POR | Power Of Reset |
| PWM | Pulse Width Modulator |
| RAM | Random Access Memory |
| RNG | Random number generator |
| SR | Status Register |
| SSP | Synchronous Serial Port |

| | |
|--------|---|
| SYSCFG | System configuration controller |
| TOE | TCPIPCore Offload Engine |
| TTL | Transistor-Transistor Logic |
| TCP | Transmission Control Protocol |
| UART | Universal Asynchronous Receiver/Transmitter |
| USB | Universal Serial Bus |
| UDP | User Datagram Protocol |
| WOL | Wake On Lan |
| WDT | Watchdog Timer |

1.2 Register Bit Conventions

Each register is shown with a key indicating the accessibility of the each individual bit, and the initial condition:

| Key | Bit Accessibility |
|-----|-------------------|
| rw | Read/Write |
| r | Read Only |
| r0 | Read as 0 |
| r1 | Read as 1 |
| W | Write Only |

2 System and memory overview

2.1 System architecture

Main system consists of :

- Two masters :
 - Cortex-M0 core
 - uDMAC (PL230, 6channel)
- Ten slaves :
 - Internal BOOT ROM
 - Internal SRAM
 - Internal Flash memory
 - Two AHB2APB bridge which connects all APB peripherals
 - Four AHB dedicated to 16bit GPIOs
 - TCPIP Hardware core

System architecture and AHB-Lite bus architecture shown in Figure 1.

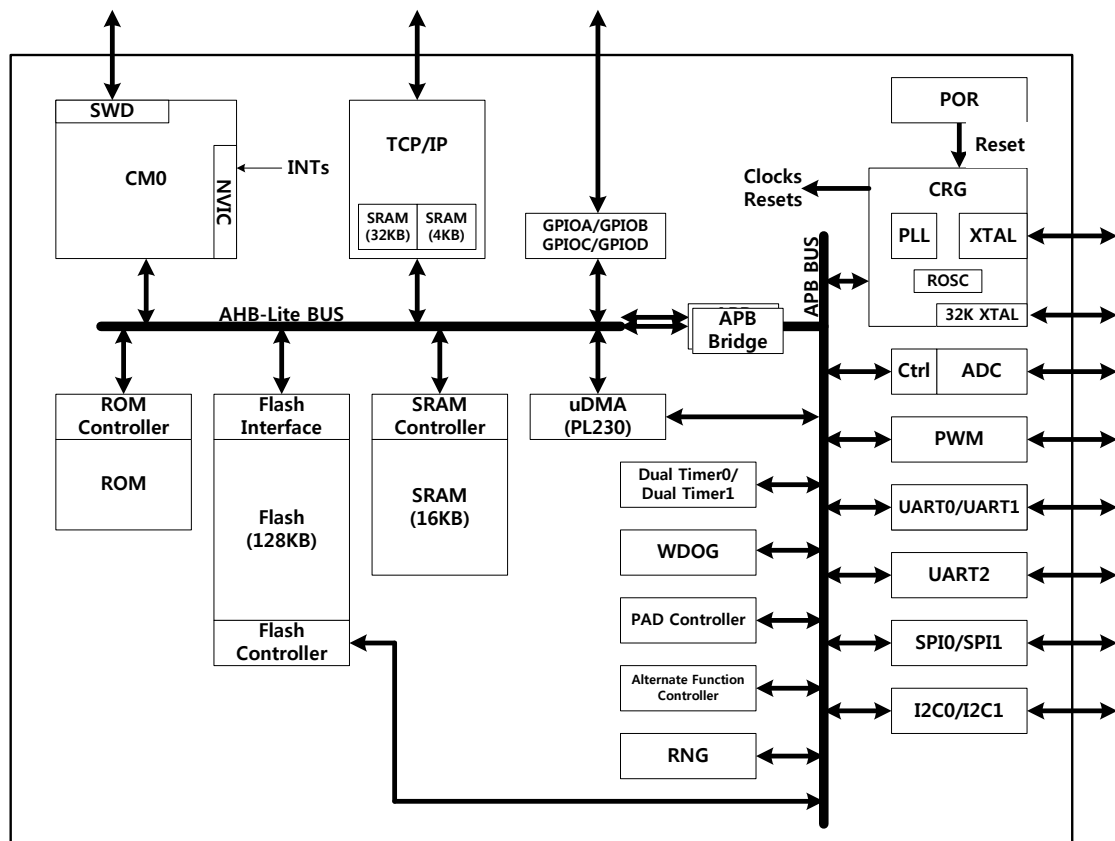


Figure 1 W7500 System Architecture

AHB-Lite BUS

- This bus connects the two masters (Cortex-M0 and uDMAC) and ten AHB slaves.

Two APB BUSs

- These buses connect Seventeen APB peripherals (Watchdog, two dual timers, pwm, two UARTs, simple UART, two I2Cs, two SSPs, random number generator, real time clock, 12bits analog digital converter, clock controller, IO configuration, PAD MUX controller)

2.2 Memory organization

2.2.1 Introduction

Program memory, data memory, registers and I/O ports are organized within the same linear 4-Gbyte address space.

The bytes are coded in memory in Little Endian format. The lowest numbered byte in a word is considered the word's least significant byte and the highest numbered byte the most significant.

2.2.2 Memory map

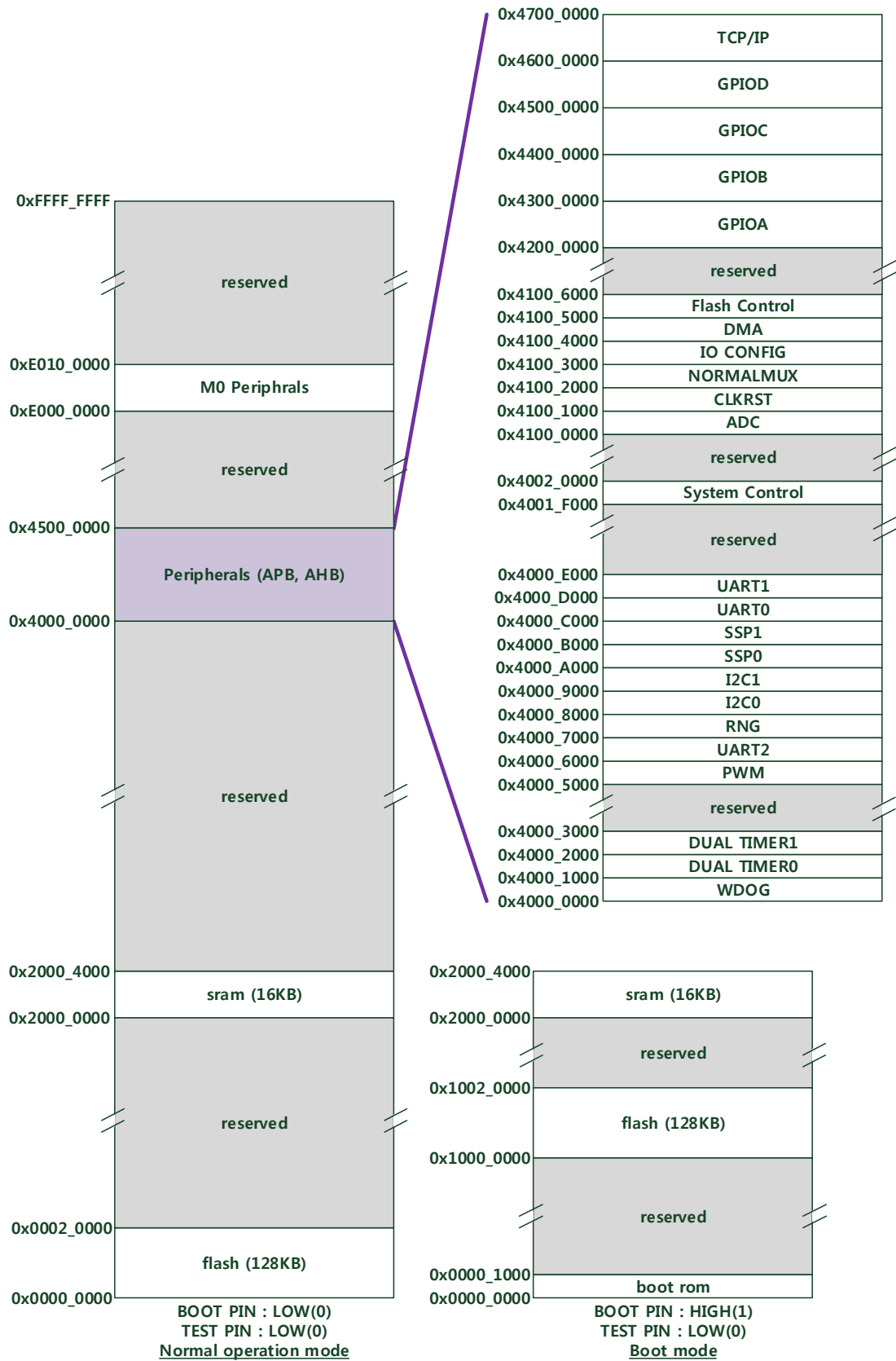


Figure 2 W7500 memory map

3 System configuration controller (SYSCFG)

3.1 Introduction

Main purposes of the system configuration controller are the following

- Control of the memory remap feature
- The ability to enable an automatic reset if the system locks up
- Information about the cause the last reset

4 Interrupt and events

4.1 Introduction

W7500 contains interrupt service and event service as below

- 26ea interrupt request (IRQ) lines.
- One NonMaskable Interrupt (NMI).
- One event signal

4.2 Interrupt assignments

Table 1 describes the W7500 interrupt assignments.

Table 1 W7500 interrupt assignments

| IRQ/NMI | Device | Description | Address |
|---------|------------|-------------------------------------|-------------|
| NMI | Watchdog | Watchdog interrupt | 0x0000_0008 |
| IRQ[0] | SSP0 | SSP0 global interrupt | 0x0000_0040 |
| IRQ[1] | SSP1 | SSP1 global interrupt | 0x0000_0044 |
| IRQ[2] | UART0 | UART0 global interrupt | 0x0000_0048 |
| IRQ[3] | UART1 | UART1 global interrupt | 0x0000_004C |
| IRQ[4] | UART2 | UART2 global interrupt | 0x0000_0050 |
| IRQ[5] | I2C0 | I2C0 global interrupt | 0x0000_0054 |
| IRQ[6] | I2C1 | I2C1 global interrupt | 0x0000_0058 |
| IRQ[7] | GPIO0 | GPIOA global interrupt | 0x0000_005C |
| IRQ[8] | GPIO1 | GPIOB global interrupt | 0x0000_0060 |
| IRQ[9] | GPIO2 | GPIOC global interrupt | 0x0000_0064 |
| IRQ[10] | GPIO3 | GPIOD global interrupt | 0x0000_0068 |
| IRQ[11] | DMA | DMA channel 1 ~ channel 5 interrupt | 0x0000_006C |
| IRQ[12] | Dualtimer0 | Dualtimer0 global interrupt | 0x0000_0070 |
| IRQ[13] | Dualtimer1 | Dualtimer1 global interrupt | 0x0000_0074 |
| IRQ[14] | PWM0 | PWM0 global interrupt | 0x0000_0078 |

| | | | |
|---------|----------|-------------------------------|-------------|
| IRQ[15] | PWM1 | PWM1 global interrupt | 0x0000_007C |
| IRQ[16] | PWM2 | PWM2 global interrupt | 0x0000_0080 |
| IRQ[17] | PWM3 | PWM3 global interrupt | 0x0000_0084 |
| IRQ[18] | PWM4 | PWM4 global interrupt | 0x0000_0088 |
| IRQ[19] | PWM5 | PWM5 global interrupt | 0x0000_008C |
| IRQ[20] | PWM6 | PWM6 global interrupt | 0x0000_0090 |
| IRQ[21] | PWM7 | PWM7 global interrupt | 0x0000_0094 |
| IRQ[22] | reserved | | 0x0000_0098 |
| IRQ[23] | ADC | ADC acquisition end interrupt | 0x0000_009C |
| IRQ[24] | TCPIP | TCPIP global interrupt | 0x0000_00A0 |
| IRQ[25] | EXT_INT | External pin interrupt | 0x0000_00A4 |
| IRQ[26] | reserved | | 0x0000_00A8 |
| IRQ[27] | reserved | | 0x0000_00AC |
| IRQ[28] | reserved | | 0x0000_00B0 |
| IRQ[29] | reserved | | 0x0000_00B4 |
| IRQ[30] | reserved | | 0x0000_00B8 |
| IRQ[31] | reserved | | 0x0000_00BC |

4.3 Event

W7500 is able to handle internal events in order to wake up the core(WFE). The wakeup event can be generated by

- When after DMA process finished

5 Power supply

5.1 Introduction

W7500 embeds a voltage regulator in order to supply the internal 1.5V digital power domain.

- Require a 2.7V ~ 5.5V operating supply voltage (VDD)
- ADC ref voltage is same as VDD

5.2 Voltage regulator

The voltage regulator is always enabled after Reset and works in only one mode.

- In Run mode, the regulator supplies full power to the 1.5V domain.
- There is no power down or sleep mode.

5.3 Power supply supervisor

W7500 has an integrated reset (POR) circuit which is always active and ensure proper operation above a threshold of 0.6V

- The POR monitors only the VDD supply voltage. During the startup phase VDD must arrive first and be greater than or equal to 0.6V

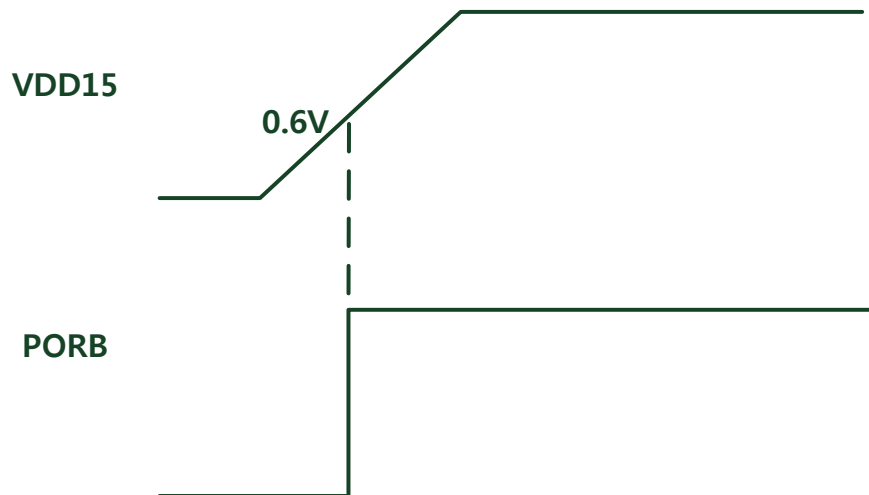


Figure 3 POR reset waveform

5.4 Low-power modes

W7500 is in RUN mode after a system or power reset. There are two low power modes to save power when the CPU does not need to be kept running. These modes are useful for instances like when the CPU is waiting for an external interrupt. Please note that there is no power-off mode for W7500.

The device features two low-power modes:

- Sleep mode
- Deep Sleep mode

Additionally, the power consumption can be reducing by following methods:

- User can slow down the system clocks
- User can gate the clocks to the peripherals when they are unused.

5.4.1 Sleep mode

W7500 has two kinds of sleep modes. One is Sleep mode and the other is Deep sleep mode.

Two of them are almost the same except the clock gated peripherals kinds. Table 2 shows the Sleep mode summary.

Table 2 W7500 sleep mode summary

| Mode | Entry | Wakeup | Effect on clocks |
|-----------------|-----------------------------|---------------|--|
| Sleep mode | DEEPSLEEP = 0 Enable WFI | Any interrupt | CPU clock OFF APB Bus Clock ON AHB Bus clock ON Memory clocks ON |
| | DEEPSLEEP = 0 Enable WFE | Wakeup event | |
| Deep Sleep mode | DEEPSLEEP = 1 Enable WFI | Any interrupt | CPU clock OFF APB Bus Clock OFF AHB Bus clock OFF Memory clocks OFF |
| | DEEPSLEEP = 1 Enable WFE | Wakeup event | |

5.4.2 Peripheral clock gating

In Run mode, individual clocks can be stopped at any time to reduce power.

Peripheral clock gating is controlled by the CRG block.

Below is the list of clocks which can be gating in CRG block.

- ADC clock (ADCCLK)
- SSP0, SSP1 clock (SSPCLK)
- UART0, UART1 clock (UARTCLK)
- Two Timer clocks (TIMCLK0, TIMCLK1)
- 8ea PWM clocks (PWMCLK0 ~ PWMCLK7)
- WDOG clock (WDOGCLK)
- Random number generator clock (RNGCLK)

6 System tick timer

6.1 Introduction

System tick timer(SysTick) is part of the ARM Cortex-M0 core

6.2 Features

Simple 24bit timer.

Clocked internally by the system clock or the system clock/2.

6.3 Functional description

The SysTick timer is an integral part of Cortex-M0. The SysTick timer is intended to generated a fixed 10 millisecond interrupt for use by an operating system or other system management software.

Since the SysTick timer is a part of the Cortex-M0, it facilitates porting of software by

providing a standard timer that is available on Cortex-M0 based devices. The SysTick timer can be used for :

- An RTOS tick timer which fires at a programmable rate (for example 100 Hz) and invokes a SysTick routine.
- A high-speed alarm timer using the core clock.
- A simple counter. Software can use this to measure time to completion and time used.
- An internal clock source control based on missing/meeting durations. The COUNTFLAG bit-field in the control and status register can be used to determine if an action completed within a set duration, as part of a dynamic clock management control loop.

7 TCIPCore Offload Engine (TOE)

7.1 Introduction

The TCP/IPCore Offload Engine (TOE) is a Hardwired TCP/IP embedded Ethernet controller that provides easier Internet connection to embedded systems. TOE enables users to have Internet connectivity in their applications by using the TCP/IP stack.

WIZnet's Hardwired TCP/IP is the market-proven technology that supports TCP, UDP, IPv4, ICMP, ARP, IGMP, and PPPoE protocols. TOE embeds the 32Kbyte internal memory buffer for the Ethernet packet processing. Using TOE allows users to implement the Ethernet application by adding the simple socket program. It's faster and easier than using any other Embedded Ethernet solutions. 8 independent hardware sockets can be used simultaneously.

TOE also provides WOL (Wake on LAN) to reduce power consumption of the system.

7.2 Features

- Supports Hardwired TCP/IP Protocols : TCP, UDP, ICMP, IPv4, ARP, IGMP, PPPoE
- Supports 8 independent sockets simultaneously
- Supports Power down mode
- Supports Wake on LAN over UDP
- Internal 32Kbytes Memory for TX/RX Buffers
- Not supports IP Fragmentation

7.3 Functional description

Figure 4 shows the TOE block diagram.

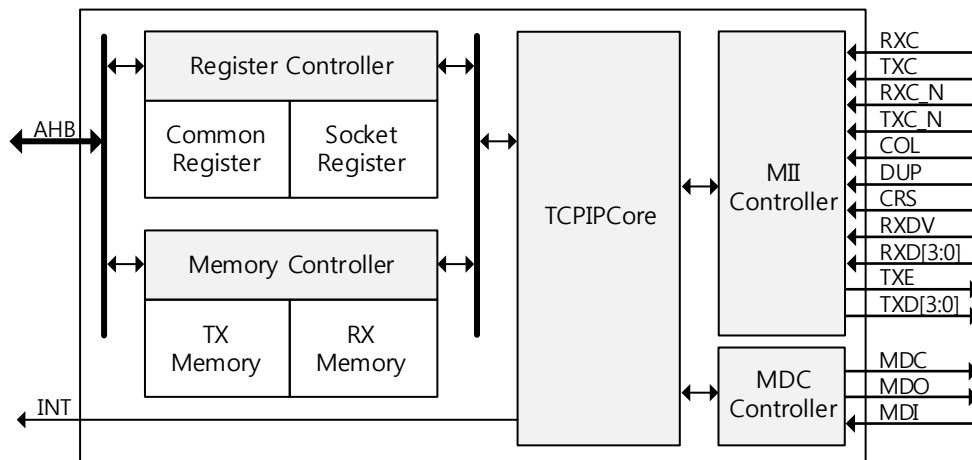


Figure 4 TOE block diagram

7.4 TOE Memory map

TOE has one Common Register Block, eight Socket Register Blocks, and TX/RX Buffer Blocks allocated to each Socket. Figure 5 shows the selected block by the base address and the available offset address range of Socket TX/RX Buffer Blocks. Each Socket's TX Buffer Block physically exists in one 16KB TX memory and is initially allocated with 2KB. Also, Each Socket's RX Buffer Block physically exists in one 16KB RX Memory and is initially allocated with 2KB. Regardless of the allocated size of each Socket TX/RX Buffer, it can be accessible within the 16 bits offset address range (From 0x0000 to 0xFFFF).

Refer to 'Chapter 7.4.3' for more information about 16KB TX/RX Memory organization and access method.

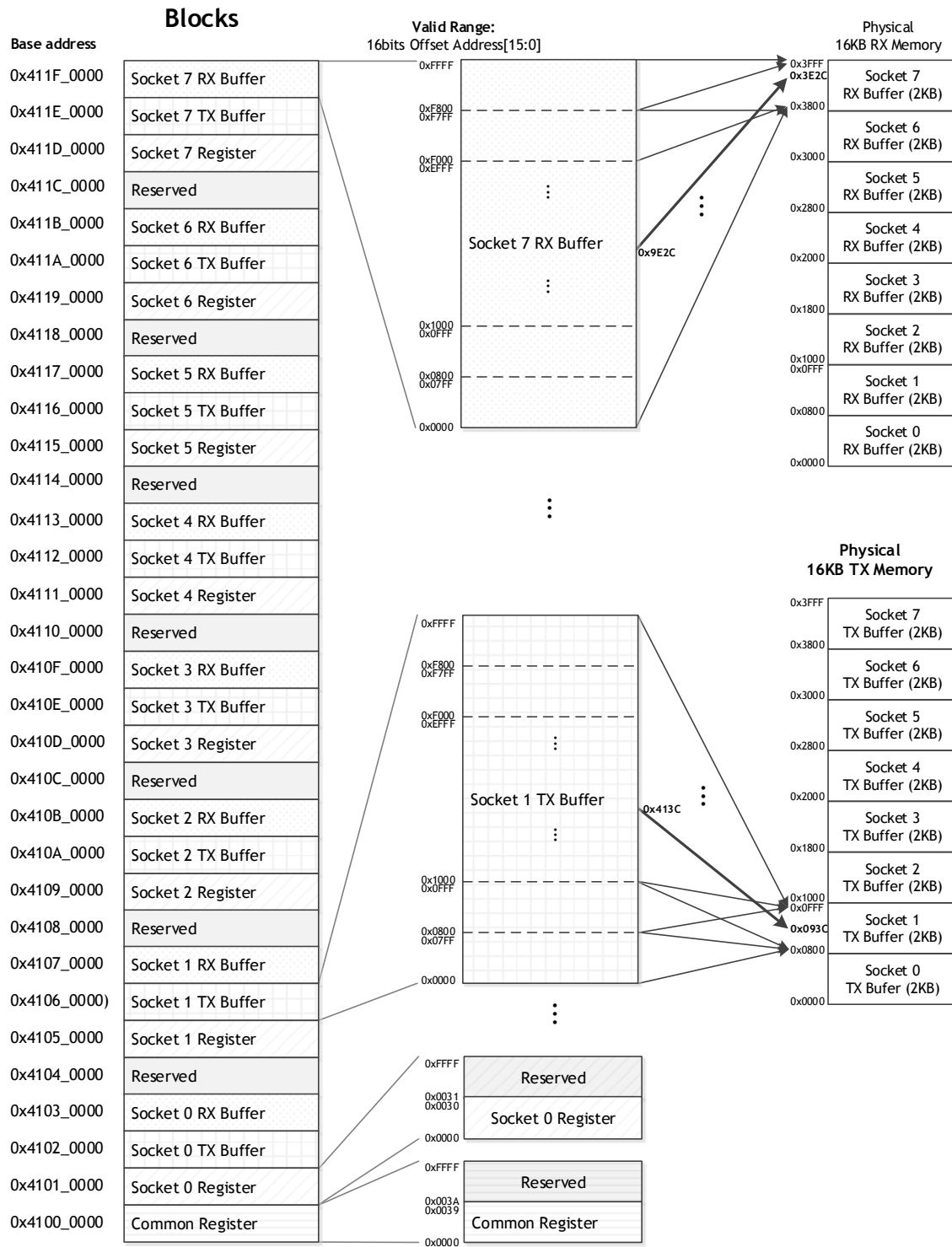


Figure 5. Register & Memory Organization

7.4.1 Common register map

Common Register Block configures the general information of TOE such as IP and MAC address.

<Table 3> defines the offset address of registers in this block.

Table 3. Offset Address for Common Register

| Address | Register |
|---------|-------------------------------------|
| 0x0000 | TOE Version (VERSIONR) |
| 0x2000 | TICKCLOCK (TCLKR) |
| 0x2100 | Interrupt (IR) |
| 0x2104 | Interrupt Mask (IMR) |
| 0x2108 | Interrupt Clear (IRCR) |
| 0x2110 | Socket Interrupt (SIR) |
| 0x2114 | Socket Mask (SIMR) |
| 0x2300 | Mode (MR) |
| 0x2400 | PPP Timer (PTIMER) |
| 0x2404 | PPP Magic (PMAGIC) |
| 0x2408 | PPP Destination MAC Address (PHAR1) |
| 0x240C | PPP Destination MAC address (PHAR0) |
| 0x2410 | PPP Session Identification (PSIDR) |
| 0x2414 | PPP Maximum Segment Size (PMSS) |
| 0x6000 | Source Hardware Address (SHAR1) |
| 0x6004 | Source Hardware Address (SHAR0) |
| 0x6008 | Gateway Address (GA) |
| 0x600C | Subnet Mask (SUB) |
| 0x6010 | Source IP Address (SIP) |
| 0x6020 | Network Configuration Lock (NCONFL) |
| 0x6040 | Retry Time (RTR) |
| 0x6044 | Retry Counter (RCR) |
| 0x6050 | Unreachable IP Address (UIP) |
| 0x6054 | Unreachable Port Address (UPORT) |

7.4.2 Socket register map

TOE supports 8 Sockets for communication channel. Each Socket is controlled by Socket n Register ($n = 0, \dots, 7$, where n is socket number). <Table 2> defines the 16bits Offset Address of registers in Socket n Register Block.

Table 4. Offset Address in Socket n Register Block (n = 0,...,7, where n is Socket number)

| Offset | Register |
|--------|---|
| 0x0000 | Socket Mode (Sn_MR) |
| 0x0010 | Socket Command (Sn_CR) |
| 0x0020 | Socket Interrupt (Sn_IR) |
| 0x0024 | Socket Interrupt Mask (Sn_IMR) |
| 0x0028 | Socket Interrupt Clear (Sn_ICR) |
| 0x0030 | Socket Status (Sn_SR) |
| 0x0100 | Socket Protocol Number (Sn_PNR) |
| 0x0104 | Socket IP Type of Service (Sn_TOS) |
| 0x0108 | Socket TTL (Sn_TTLR) |
| 0x010C | Socket Fragment Offset (Sn_FRAG) |
| 0x0110 | Socket Maximum Segment (Sn_MSSR) |
| 0x0114 | Socket Port Number (Sn_PORT) |
| 0x0118 | Socket Destination Hardware address0 (Sn_DHAR0) |
| 0x011C | Socket Destination Hardware address1 (Sn_DHAR1) |
| 0x0120 | Socket Destination Port Number (Sn_DPORTR) |
| 0x0124 | Socket Destination IP Address (Sn_DIPR) |
| 0x0180 | Socket Keep Alive Timer (Sn_KATMR) |
| 0x0184 | Socket Retry Time (Sn_RTR) |
| 0x0188 | Socket Retry Counter (Sn_RCR) |
| 0x0200 | Socket TX Memory Size (Sn_TXBUF_SIZE) |
| 0x0204 | Socket TX Free Size (Sn_TX_FSR) |
| 0x0208 | Socket TX Read Pointer (Sn_TX_RD) |
| 0x020C | Socket TX Write Pointer (Sn_TX_WR) |
| 0x0220 | Socket RX Memory Size (Sn_RXBUF_SIZE) |
| 0x0224 | Socket RX Received Size (Sn_RX_RSR) |
| 0x0228 | Socket RX Read Pointer (Sn_RX_RD) |
| 0x022C | Socket RX Write Pointer (Sn_RX_WR) |

7.4.3 Memory

TOE has one 16KB TX memory for Socket n TX Buffer Blocks and one 16KB RX memory for Socket n RX buffer Blocks.

16KB TX memory is initially allocated in 2KB size for each Socket TX Buffer Block (2KB X 8 = 16KB). The initial allocated 2KB size of Socket n TX Buffer can be re-allocated by using ‘Socket n TX Buffer Size Register (Sn_TXBUF_SIZE)’.

Once all Sn_TXBUF_SIZE registers have been configured, Socket TX Buffer is allocated with the configured size of 16KB TX Memory and is assigned sequentially from Socket 0 to Socket 7. Its physical memory address is automatically determined in 16KB TX memory. Therefore, the total sum of Sn_TXBUF_SIZE should not exceed 16 in case of error in data transmission.

The 16KB RX memory allocation method is the same as the 16KB TX memory allocation method. 16KB RX memory is initially allocated into 2KB size for each Socket RX Buffer Block (2KB X 8 = 16KB). The initial allocated 2KB size of Socket n RX Buffer can be re-allocated by using 'Socket n RX Buffer Size Register (Sn_RXBUF_SIZE)'.

When all Sn_RXBUF_SIZE registers have been configured, the Socket RX Buffer is allocated with the configured size in 16KB RX Memory and is assigned sequentially from Socket 0 to Socket 7. The physical memory address of the Socket RX Buffer is automatically determined in 16KB RX memory. Therefore, the total sum of Sn_RXBUF_SIZE should not exceed 16 or data reception error will occur.

For 16KB TX/RX memory allocation, refer to Sn_TXBUF_SIZE & Sn_RXBUF_SIZE in 'Chapter 7.4.2'. The Socket n TX Buffer Block allocated in 16KB TX memory is buffer for saving data to be transmitted by host. The 16bits Offset Address of Socket n TX Buffer Block has 64KB address space ranged from 0x0000 to 0xFFFF, and is configured with reference to 'Socket n TX Write Pointer Register (Sn_TX_WR)' & 'Socket n TX Read Pointer Register(Sn_TX_RD)'. However, the 16bits Offset Address automatically converts into the physical address to be accessible in 16KB TX memory such as Figure 5. Refer to 'Chapter 7.4.2' for Sn_TX_WR & Sn_TX_RD.

The Socket n RX Buffer Block allocated in 16KB RX memory is buffer for saving the received data through the Ethernet. The 16bits Offset Address of Socket n RX Buffer Block has 64KB address space ranged from 0x0000 to 0xFFFF, and is configured with reference to 'Socket n RX RD Pointer Register (Sn_RX_RD)' & 'Socket n RX Write Pointer Register (Sn_RX_WR)'. However, the 16bits Offset Address automatically converts into the physical address to be accessible in 16KB RX memory such as Figure 5. Refer to 'Chapter 7.4.2' for Sn_RX_RD & Sn_RX_WR.

8 Booting Sequence

W7500 has three different boot modes that can be selected through the BOOT pin and TEST pin as shown in Table 5.

Table 5 operation of mode selection

| Mode selection | | Mode | Aliasing |
|----------------|------|------|---|
| TEST | BOOT | | |
| 0 | 0 | APP | User code execute in Main Flash memory. |
| 0 | 1 | ISP | In this mode,W7500 can support ISP function in order to control flash using serial interface. |

When W7500 is reset by hardware, it will be operated as below in embedded boot code.

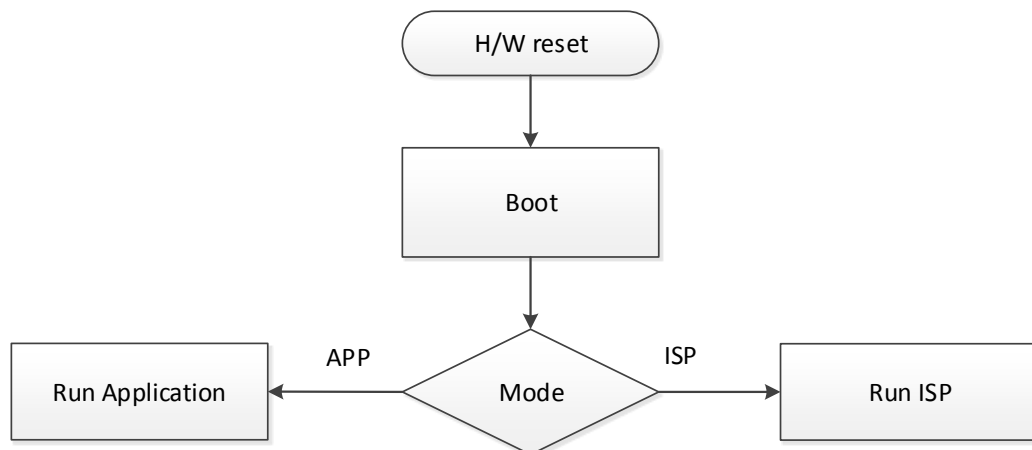


Figure 6. operation of boot code

9 Embedded Flash memory

9.1 Flash main features

- Up to 128Kbytes of Flash memory
- Memory organization:
 - Main Flash memory block:
Up to 128Kbytes
 - Information block:
Up to 512bytes
Information block is read only
 - Data block:
Up to 512bytes
- Flash memory interface features:
 - Read interface with prefetch buffer(1 x 32-bit words)
 - Flash Program / Erase operation
 - Read / Write protection

9.2 Flash memory functional description

9.2.1 Flash memory organization

The Flash memory is organized of 32-bit wide memory cells that can be used for storing both code and data constants.

The memory organization is based on a main Flash memory block containing 512 sectors of 256byte or 32 blocks of 4Kbyte. The block and sector provides read/write protection.

Table 6 description of Flash memory

| Flash area | Flash memory address | Size (bytes) | Name | Description |
|-------------------|---------------------------|--------------|-----------|-------------|
| Main Flash memory | 0x0000 0000 ~ 0x0000 00FF | 256 | Sector 0 | Block 0 |
| | 0x0000 0100 ~ 0x0000 01FF | 256 | Sector 1 | |
| | 0x0000 0200 ~ 0x0000 02FF | 256 | Sector 2 | |
| | 0x0000 0300 ~ 0x0000 03FF | 256 | Sector 3 | |
| | ⋮ | ⋮ | ⋮ | ⋮ |
| | 0x0000 7000 ~ 0x0000 70FF | 256 | Sector112 | Block 7 |
| | 0x0000 7100 ~ 0x0000 71FF | 256 | Sector113 | |
| | 0x0000 7200 ~ 0x0000 72FF | 256 | Sector114 | |

| | | | | |
|---------------------------------|---------------------------|----------|-----------|-----------|
| | 0x0000 7300 ~ 0x0000 73FF | 256 | Sector115 | |
| | ⋮ | ⋮ | ⋮ | ⋮ |
| | 0x0001 FC00 ~ 0x0001 FCFF | 256 | Sector509 | Block 32 |
| | 0x0001 FD00 ~ 0x0001 FDFF | 256 | Sector510 | |
| | 0x0001 FE00 ~ 0x0001 FEFF | 256 | Sector511 | |
| | 0x0001 FF00 ~ 0x0001 FFFF | 256 | Sector512 | |
| Information block | 0x0003 FC00 ~ 0x0003 FCFF | 256 | | Lock info |
| | 0x0003 FD00 ~ 0x0003 FDFF | Reserved | | |
| Data block | 0x0003 FE00 ~ 0x0003 FEFF | 256 | | Data0 |
| | 0x0003 FF00 ~ 0x0003 FFFF | 256 | | Data1 |
| Flash memory Interface register | 0x4100 5000 ~ 0x4100 5003 | 4 | | FACCR |
| | 0x4100 5004 ~ 0x4100 5007 | 4 | | FADDR |
| | 0x4100 5008 ~ 0x4100 500B | 4 | | FDATAR |
| | 0x4100 500C ~ 0x4100 500F | 4 | | FCTRLR |
| | 0x4100 5010 ~ 0x4100 5013 | 4 | | FSTATR |
| | 0x4100 5014 ~ 0x4100 5017 | 4 | | FLOCKR0 |
| | 0x4100 5018 ~ 0x4100 501B | 4 | | FLOCKR1 |
| | 0x4100 5030 ~ 0x4100 5033 | 4 | | FKEYR0 |
| | 0x4100 5034 ~ 0x4100 5037 | 4 | | FKEYR1 |
| | 0x4100 5038 ~ 0x4100 503B | 4 | | BSADDR0 |
| | 0x4100 503C ~ 0x4100 503F | 4 | | BSADDR1 |

The W7500 embedded Flash memory can be programmed using in-circuit programming or in-application programming.

The **in-circuit programming (ICP)** method is used to update the entire contents of the Flash memory using the SWD protocol or the boot loader to load the user application into the microcontroller. ICP offers quick and efficient design iterations and eliminates unnecessary package handling or socketing of devices.

In contrast to the ICP method, **in-application programming (IAP)** can use any communication interface supported by the microcontroller (I/Os, UART, I2C, SPI, etc.) to download programming data into memory. IAP allows the user to re-program the Flash memory while the application is running. Nevertheless, part of the application has to have been previously programmed in the Flash memory using ICP.

The program and erase operations can be performed over the whole product voltage range.

They are managed through the following seven Flash registers:

- Flash access control register (FACCR)
- Flash address register (FADDR)
- Flash data register (FDATAR)

- Flash control register (FCTRLR)
- Flash status register (FSTATR)
- Flash lock register (FLOCKR0/R1)
- Flash key register (FKEYR0/R1)

Unlocking the Flash access Control register (FACCR)

After reset, the Flash memory is protected against unwanted write or erase operations. The FACCR register is not accessible in write mode. An unlocking sequence should be written to the FKEYR0/R1 register to open the access to the FACCR register. This sequence consists of two write operations:

- Write KEY 0 (FKEYR0) = 0x52537175
- Write KEY 1 (FKEYR1) = 0xA91875FC

Any wrong sequence locks up the FACCR register.

The FACCR register can be locked again by finishing flash control operation.

9.2.2 Read operations

The embedded Flash module can be addressed directly as a common memory space. The instruction fetch and the data access are both done through the same AHB bus. Read accesses can be performed with the following options managed through the Flash control register. (FCTRLR)

The Flash reading sequence using FCTRLR register is as below:

1. Check that no main Flash memory operation is ongoing by checking the RDY bit in the FSTATR register.
2. Set KEY in FKEYR0/R1 for setting FACCR register.
3. Set FEN and CTRL bits in the FACCR register.
4. Write main Flash memory address or Data block address to FADDR register.
5. Set RDI or RD bit in FCTRLR to 1. If use RDI bit, don't need to set FADDR again due to increase automatically by SZ bit in FACCR register.
6. Read data from FDATAR register.
7. Wait until the RDY bit is 1 in the FSTATR register.(it is set when the programming operation has succeeded)
8. Set KEY in FKEYR0/R1 for clearing FACCR register.
9. Clear FEN and CTRL bits in the FACCR register

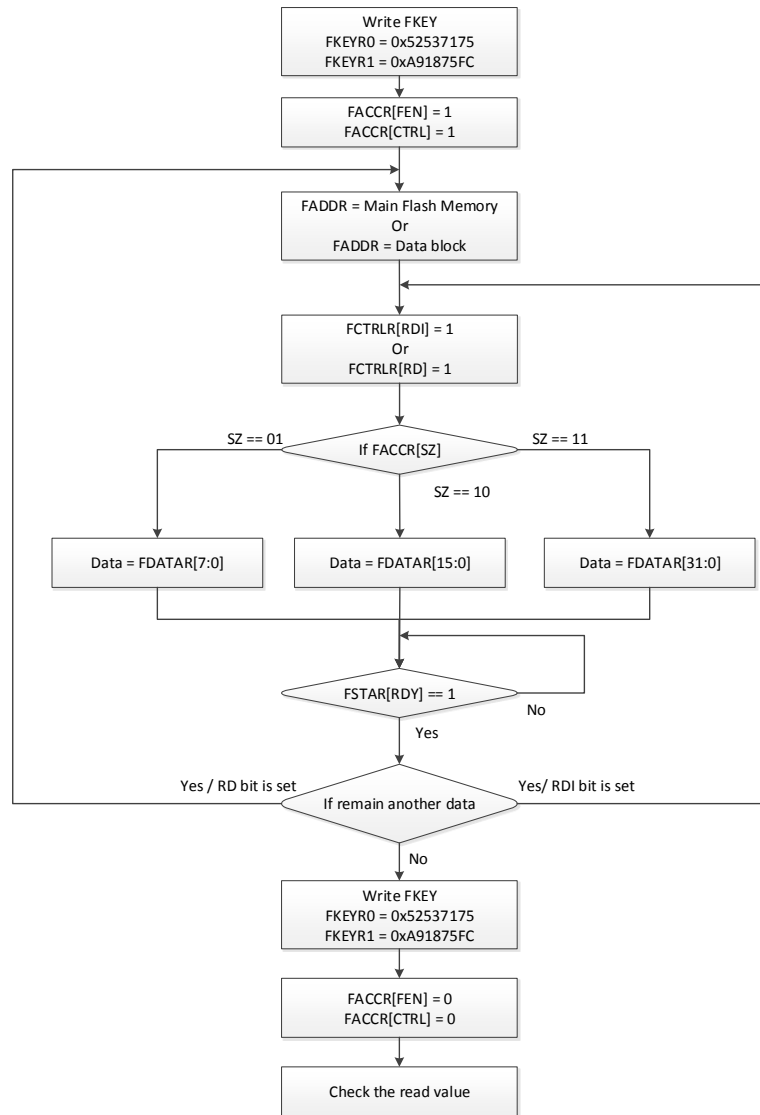


Figure 7. Flash reading sequence

9.2.3 Flash erase operations

Sector Erase

Follow the procedure below to erase a sector:

1. Check that no Flash memory operation is ongoing by checking the RDY bit in the FSTATR register.
2. Set KEY in FKEYR0/R1 for setting FACCR register.
3. Set FEN and CTRL bits in the FACCR register.
4. Write main Flash memory address to FADDR register to erase.
5. Set SER bit in FCTRLR to 1.
6. Wait until the RDY bit is 1 in the FSTATR register.
7. Set KEY in FKEYR0/R1 for clearing FACCR register.
8. Clear FEN and CTRL bits in the FACCR register

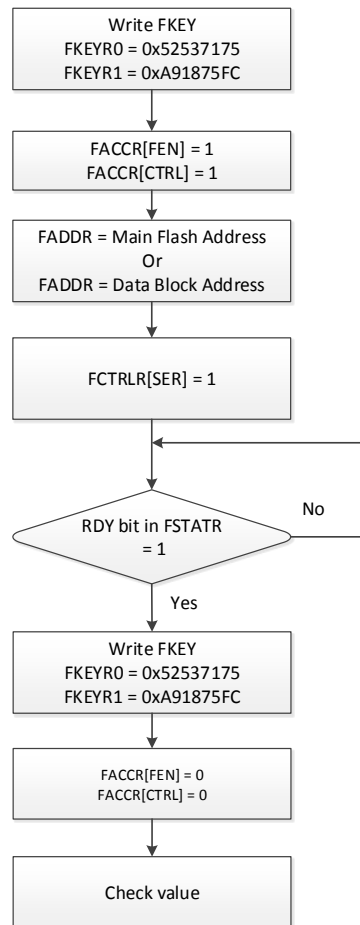


Figure 8. Flash erase operations

Block Erase

To erase a block, set BER bit in FACTRLR to 1. All other procedures are the same as the sector erase sequence.

Chip Erase (All main Flash memory erase)

To erase chip (Main Flash memory), Set CER bit in FACTRLR to 1. All other procedures are the same as the sector erase sequence.

Mass Erase (All main Flash memory erase + Data block erase)

To erase mass (Main Flash memory + Data block), Set MER bit in FACTRLR to 1. All other procedures are the same as the sector erase sequence.

9.2.4 Flash program operation

The main Flash memory can be programmed word, half word, or 1 byte at a time by SZ bit of FACCR. The program operation is started when the CPU writes a data into a main Flash memory address with the WRI or WR bit of FCTRLR register set.

The main Flash memory programming sequence in standard mode is as below:

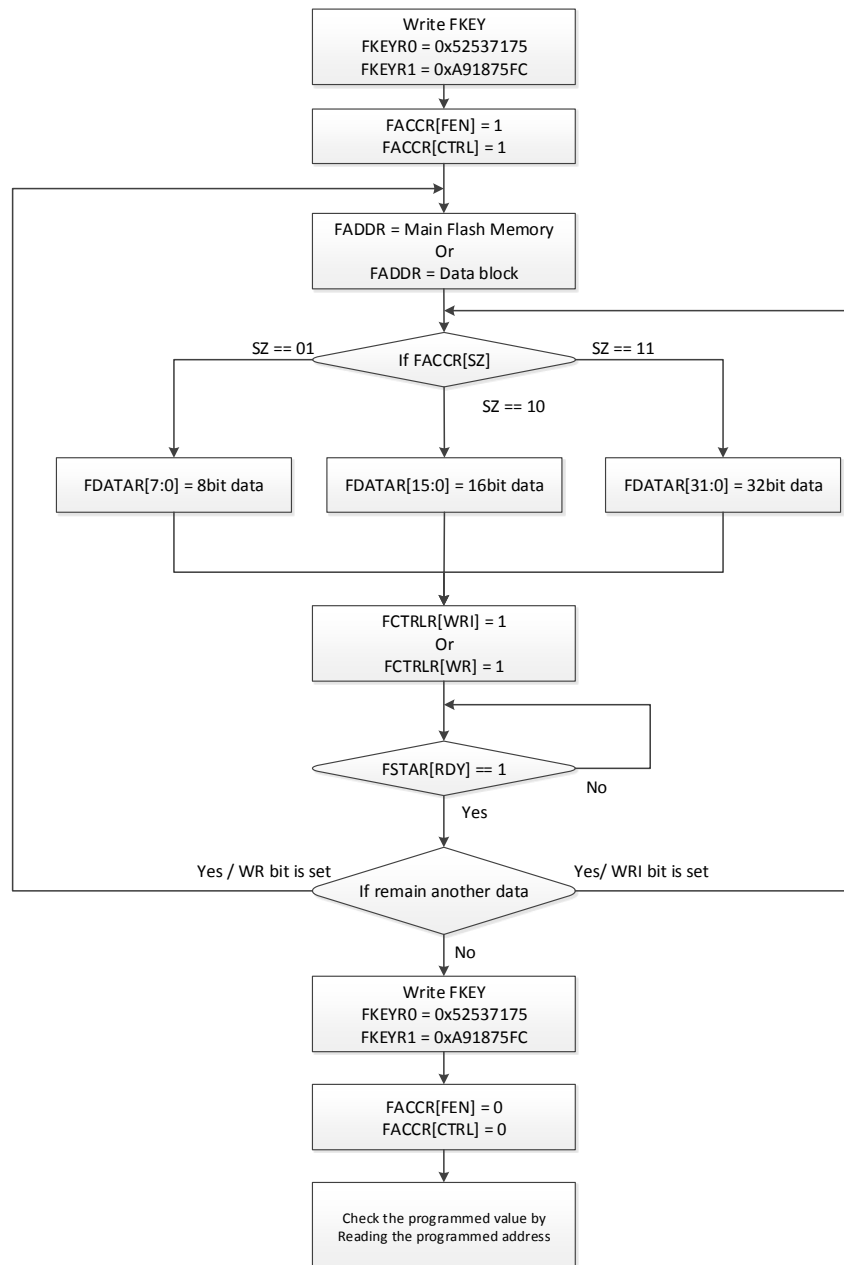


Figure 9. main Flash memory programming sequence

1. Check that no main Flash memory operation is ongoing by checking the RDY bit in the FSTATR register.
2. Set KEY in FKEYR0/R1 for setting FACCR register.

3. Set FEN and CTRL bits in the FACCR register.
4. Write main Flash memory address or Data block address to FADDR register.
5. Write data to FDATAR register.
6. Set WRI or WR bit in FACTRLR to 1. If use WRI bit, don't need to set FADDR again due to increase automatically by SZ bit in FACCR register.
7. Wait until the RDY bit is 1 in the FSTATR register.(it is set when the programming operation has succeeded)
8. Set KEY in FKEYR0/R1 for clearing FACCR register.
9. Clear FEN and CTRL bits in the FACCR register

9.3 Memory protection

The user area of the Flash memory can be protected against read by untrusted code. The blocks of the Flash memory can also be protected against unwanted write due to loss of program counter contexts. The write-protection granularity is one block (4Kbyte).

9.3.1 Read protection

The read protection is activated by DRL bit and CRL bit in FLOCKR0 register.

- DRL0 : read protection to Data0 area in Data block.
- DRL1 : read protection to Data1 area.in Data block
- CRL : read protection to main Flash memory

9.3.2 Write protection

The write protection is implemented with a granularity of one block. It is activated by configuring the FLOCKR1 register or DWL bit, CABWL bit in FLOCKR0 register.

- FLOCKR1 : write protection to main Flash memory with a granularity of one block.
- DWL0 : write protection to Data0 area in Data block.
- DWL1 : write protection to Data1 area in Data block.
- CABWL : write protection to main Flash memory all block.

10 Clock Reset generator (CRG)

10.1 Introduction

CRG is clock reset generator block for W7500 System. It provides every clock/reset for all other block include CPU and peripherals. CRG includes PLL and POR.

10.2 Features

10.2.1 Reset

- Three types of reset - external reset, Power reset, system reset
- External reset is generated by low level on the RSTn pin (external reset)
- Power reset is generated by Power-on reset (POR)
- Power on reset is generated by POR
- System reset is generated when one of the following events occurs
 - Watchdog event
 - After remapping
 - Software reset (SYSRESETREQ bit in Cortex-M0. Refer to the Cortex-M0 technical reference manual for more detail)
- Power reset sets all registers to their reset values.
- System reset sets all registers to their reset values except the CRG block registers and remap register to protect remap value

10.2.2 Clock

Two clock sources can be used to drive the system clock.

- External oscillator clock (8MHz ~ 24MHz) (OCLK)
- Internal 8MHz RC oscillator clock (RCLK)

One additional clock source

- 32.768KHz low speed external crystal which derives the real time clock.

There is a PLL

One PLL is integrated

- Input clock range is from 8MHz to 24MHz
- Frequency can be generated by M/N/OD registers. (refer register description)
- Bypass option enabled

There are many generated clocks for independent operating with system clock

- System clock (FCLK)
- ADC clock (ADCCLK)
- SSP0, SSP1 clock (SSPCLK)
- UART0, UART1 clock (UARTCLK)

- Two Timer clocks (TIMCLK0, TIMCLK1)
- 8ea PWM clocks (PWMCLK0 - PWMCLK7)
- WDOG clock (WDOGCLK)
- Random number generator clock (RNGCLK)

RNGCLK have only one source (pll output) and no prescaler

Some of the generated clocks turn off automatically when CPU enters sleep mode.

- ADCCLK, RNGCLK

Generate two Hardware TCPIP Clocks (MII_RXC, MII_TXC) are from external PADs.

Hardware TCPIP Clocks can be gated by register control.

All clocks generated from CRG can be monitored.

10.3 Functional description

Figure 10 shows the CRG block diagram.

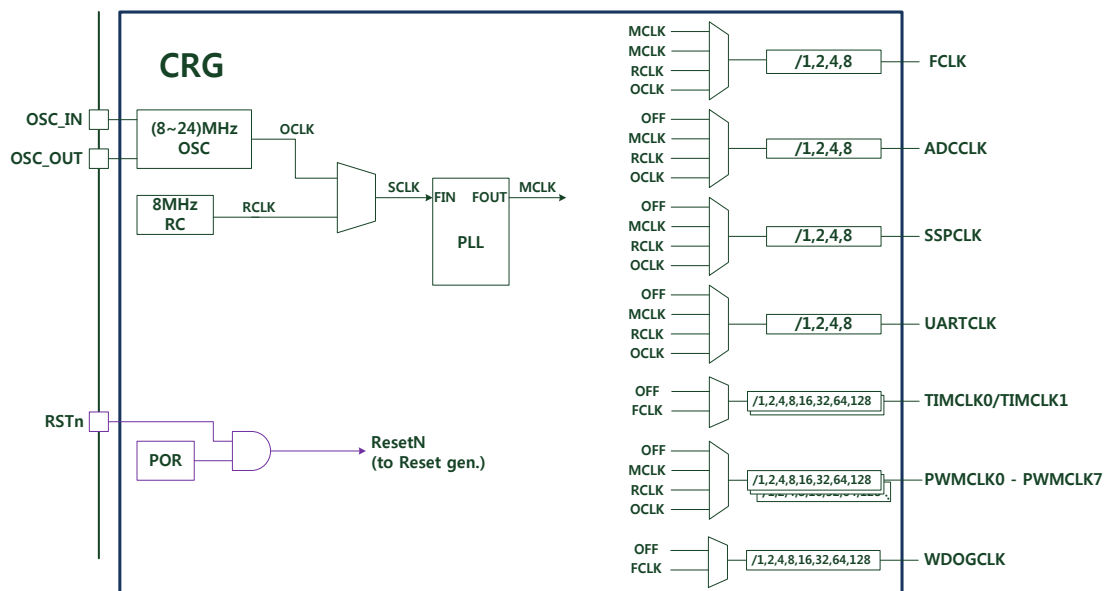


Figure 10 CRG block diagram

10.3.1 External Oscillator Clock

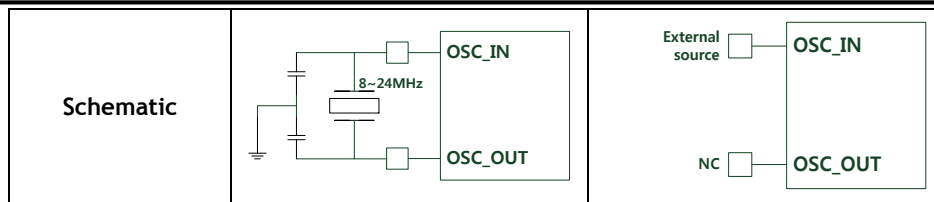
External oscillator clock (OCLK) can be generated from two possible clock source

- External crystal/ceramic resonator (8 to 24MHz external oscillator)
- User external clock

Table 7 shows the two clock sources of external oscillator clock

Table 7 External oscillator clock sources

| | | |
|--|----------------|--------------------------------|
| | External clock | Crystal/ Ceramic resonators |
|--|----------------|--------------------------------|



10.3.2 RC oscillator clock

RC oscillator clock (RCLK) signal is generated from an internal 8MHz RC oscillator.

RC oscillator has the advantage of providing a clock source at low cost (no external components). However the RC oscillator is less accurate than the external crystal or ceramic resonator.

- Accuracy : 1% at $T_A = 25^\circ\text{C}$ (User don't need to calibration)

10.3.3 PLL

The internal PLL can be used to multiply the External Oscillator Clock (OCLK) or RC Oscillator Clock (RCLK). PLL input can be selected by register.

PLL output clock can be generated by following the equations below.

- $F_{OUT} = F_{IN} \times M / N \times 1 / OD$
- Where:
- $M = M[5] \times 2^5 + M[4] \times 2^4 + M[3] \times 2^3 + M[2] \times 2^2 + M[1] \times 2 + M[0] \times 1$
- $N = N[5] \times 2^5 + N[4] \times 2^4 + N[3] \times 2^3 + N[2] \times 2^2 + N[1] \times 2 + N[0] \times 1$
- $OD = 2^{(2 \times OD[1])} \times 2^{(1 \times OD[0])}$

10.3.4 Generated clock

Each generated clock source can be selected among 3 clock source as independent by each clock source select register.

- PLL output clock (MCLK)
- Internal 8MHz RC oscillator clock (RCLK)
- External oscillator clock (8MHz ~ 24MHz) (OCLK)

Each generated clock has own prescaler which can be selected individually by each prescale value register.

- FCLK, ADCCLK, SSPCLK, UARTCLK : 1/1, 1/2, 1/4, 1/8
- TIMCLK0, TIMCLK1, PWMCLK0 - PWMCLK7, WDOGCLK : 1/1, 1/2, 1/4, 1/8, 1/16, 1/32, 1/64, 1/128

11 Random number generator (RNG)

11.1 Introduction

RNG is a 32bit random number generator. RNG generates power on random number when power on reset. RNG can run/stop by software. RNG seed value and polynomial of RNG can be modified by software.

11.2 Features

- 32bit pseudo random number generator
- Formula of pseudo random number generator (polynomial) can be modified.
- Seed value of random generator can be modified.
- Support power on reset random value
- Random value can be obtained by control start/stop by software.

11.3 Functional description

Figure 11 shows the RNG block diagram.

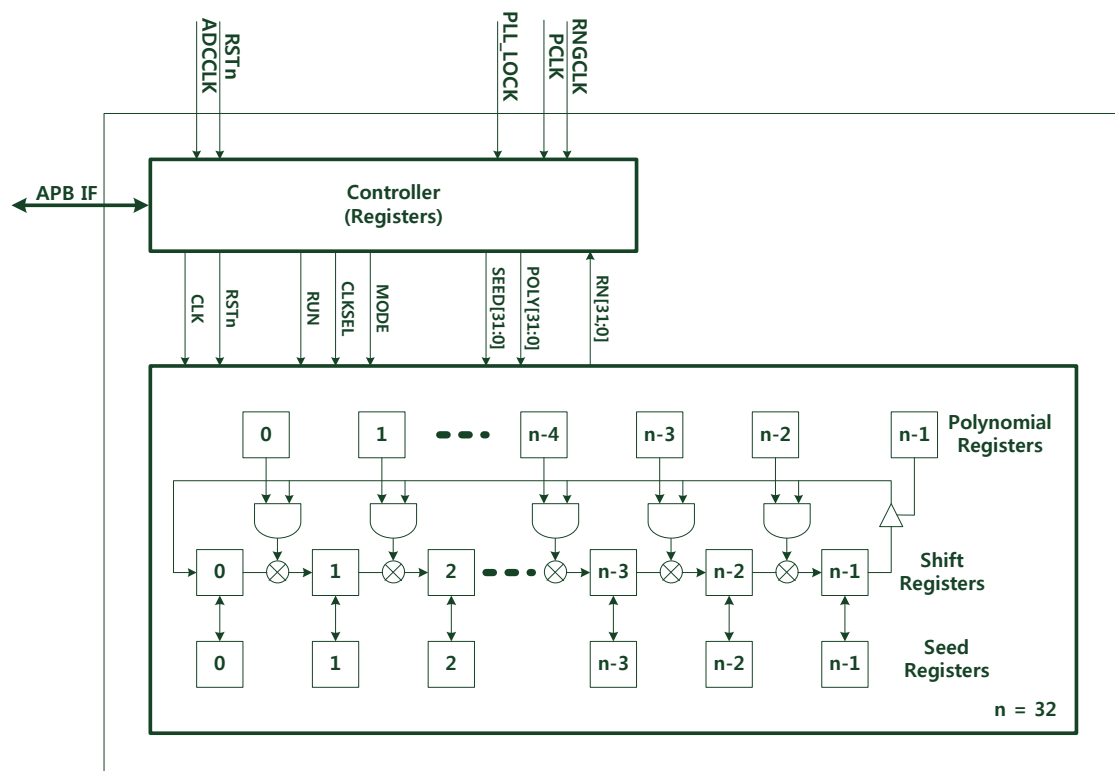


Figure 11. Random Number Generator block diagram

11.3.1 Operation RNG

Figure 12 show the flowchart of RNG operation.

A random number is automatically generated after powering on reset,

Follow the procedure below to manually generate a random number.

1. Change MODE to start/stop by register.
2. Change clock source / seed value / polynomial value if need.
3. Run and Stop the RNG.
4. Read Random value.

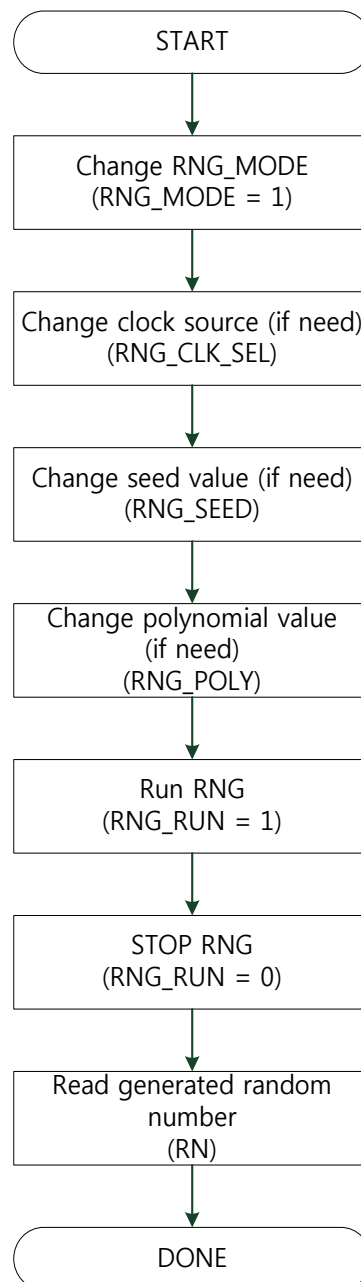


Figure 12. Flow chart of RNG operation

12 Alternate Function Controller (AFC)

12.1 Introduction

Each functional PADS have several functions.

Users can select a function in Alternate Function Controller block.

12.2 Features

Each functional pad has 2 ~ 4 functions.

Pads can be selected by each registers individually.

Each pad can be used as external interrupt source.

12.3 Functional description

Table 8 shows the function table of each functional pad.

Table 8 functional description table

| PAD | PIN | function selection register value | | | |
|-------|-----|-----------------------------------|--------------|--------------|--------------|
| | | 00 (reset value) | 01 | 10 | 11 |
| | | Normal Function | 2nd Function | 3rd Function | 4th Function |
| PA_00 | 29 | GPIOA_0 | GPIOA_0 | PWM6/CAP6 | |
| PA_01 | 30 | GPIOA_1 | GPIOA_1 | PWM7/CAP7 | |
| PA_02 | 31 | GPIOA_2 | GPIOA_2 | CLKOUT | |
| PA_03 | 49 | SWCLK | GPIOA_3 | | |
| PA_04 | 50 | SWDIO | GPIOA_4 | | |
| PA_05 | 33 | SSEL0 | GPIOA_5 | SCL1 | PWM2/CAP2 |
| PA_06 | 34 | SCLK0 | GPIOA_6 | SDA1 | PWM3/CAP3 |
| PA_07 | 35 | MISO0 | GPIOA_7 | U_CTS1 | PWM4/CAP4 |
| PA_08 | 36 | MOSI0 | GPIOA_8 | U_RTS1 | PWM5/CAP5 |
| PA_09 | 37 | SCL0 | GPIOA_9 | U_TXD1 | PWM6/CAP6 |
| PA_10 | 38 | SDA0 | GPIOA_10 | U_RXD1 | PWM7/CAP7 |
| PA_11 | 40 | U_CTS0 | GPIOA_11 | SSEL1 | |
| PA_12 | 41 | U_RTS0 | GPIOA_12 | SCLK1 | |
| PA_13 | 42 | U_TXD0 | GPIOA_13 | MISO1 | |
| PA_14 | 43 | U_RXD0 | GPIOA_14 | MOSI1 | |
| PA_15 | 44 | GPIOA_15 | GPIOA_15 | | |
| PB_00 | 45 | SSEL1 | GPIOB_0 | U_CTS0 | |
| PB_01 | 46 | SCLK1 | GPIOB_1 | U_RTS0 | |
| PB_02 | 47 | MISO1 | GPIOB_2 | U_TXD0 | |

| | | | | | |
|-------|----|-----------|----------|-----------|------|
| PB_03 | 48 | MOSI1 | GPIOB_3 | U_RXD0 | |
| PB_04 | 24 | TXEN | GPIOB_4 | | |
| PB_05 | 25 | COL | GPIOB_5 | | |
| PB_06 | 16 | RXD3 | GPIOB_6 | | |
| PB_07 | 17 | RXCLK | GPIOB_7 | | |
| PB_08 | 18 | DUP | GPIOB_8 | | |
| PB_09 | 19 | TXCLK | GPIOB_9 | | |
| PB_10 | 20 | TXD0 | GPIOB_10 | | |
| PB_11 | 21 | TXD1 | GPIOB_11 | | |
| PB_12 | 22 | TXD2 | GPIOB_12 | | |
| PB_13 | 23 | TXD3 | GPIOB_13 | | |
| PB_14 | 26 | | GPIOB_14 | | |
| PB_15 | 27 | | GPIOB_15 | | |
| PC_00 | 53 | U_CTS1 | GPIOC_0 | PWM0/CAP0 | |
| PC_01 | 54 | U_RTS1 | GPIOC_1 | PWM1/CAP1 | |
| PC_02 | 55 | U_TXD1 | GPIOC_2 | PWM2/CAP2 | |
| PC_03 | 56 | U_RXD1 | GPIOC_3 | PWM3/CAP3 | |
| PC_04 | 57 | SCL1 | GPIOC_4 | PWM4/CAP4 | |
| PC_05 | 58 | SDA1 | GPIOC_5 | PWM5/CAP5 | |
| PC_06 | 51 | GPIOC_6 | GPIOC_6 | U_TXD2 | |
| PC_07 | 52 | GPIOC_7 | GPIOC_7 | U_RXD2 | |
| PC_08 | 1 | PWM0/CAP0 | GPIOC_8 | SCL0 | AIN7 |
| PC_09 | 2 | PWM1/CAP1 | GPIOC_9 | SDA0 | AIN6 |
| PC_10 | 3 | U_TXD2 | GPIOC_10 | PWM2/CAP2 | AIN5 |
| PC_11 | 4 | U_RXD2 | GPIOC_11 | PWM3/CAP3 | AIN4 |
| PC_12 | 5 | AIN3 | GPIOC_12 | SSEL0 | AIN3 |
| PC_13 | 6 | AIN2 | GPIOC_13 | SCLK0 | AIN2 |
| PC_14 | 7 | AIN1 | GPIOC_14 | MISO0 | AIN1 |
| PC_15 | 8 | AIN0 | GPIOC_15 | MOSI0 | AIN0 |
| PD_00 | 11 | CRS | GPIOD_0 | | |
| PD_01 | 12 | RXDV | GPIOD_1 | | |
| PD_02 | 13 | RXD0 | GPIOD_2 | | |
| PD_03 | 14 | RXD1 | GPIOD_3 | | |
| PD_04 | 15 | RXD2 | GPIOD_4 | | |

13 External Interrupt (EXTI)

13.1 Introduction

Each functional pads are connected to the external interrupt(EXTINT) source.

13.2 Features

- All functional pads can be used as an external interrupt source regardless of any set of pad function.
- External Interrupt controller has the following functions and can be controlled by registers.
 - Interrupt mask (enable or disable, default : disable)
 - Interrupt polarity (rising or falling, default : rising)

13.3 Functional description

All pads are connected to the control register individually. (External interrupt mask register and External Interrupt polarity register)

External interrupt working as following expression:

- Each pad interrupt = Interrupt mask & (Interrupt polarity ^ Pad input)
- EXTINT = any Each pad interrupt

Figure 13 shows the External Interrupt diagram.

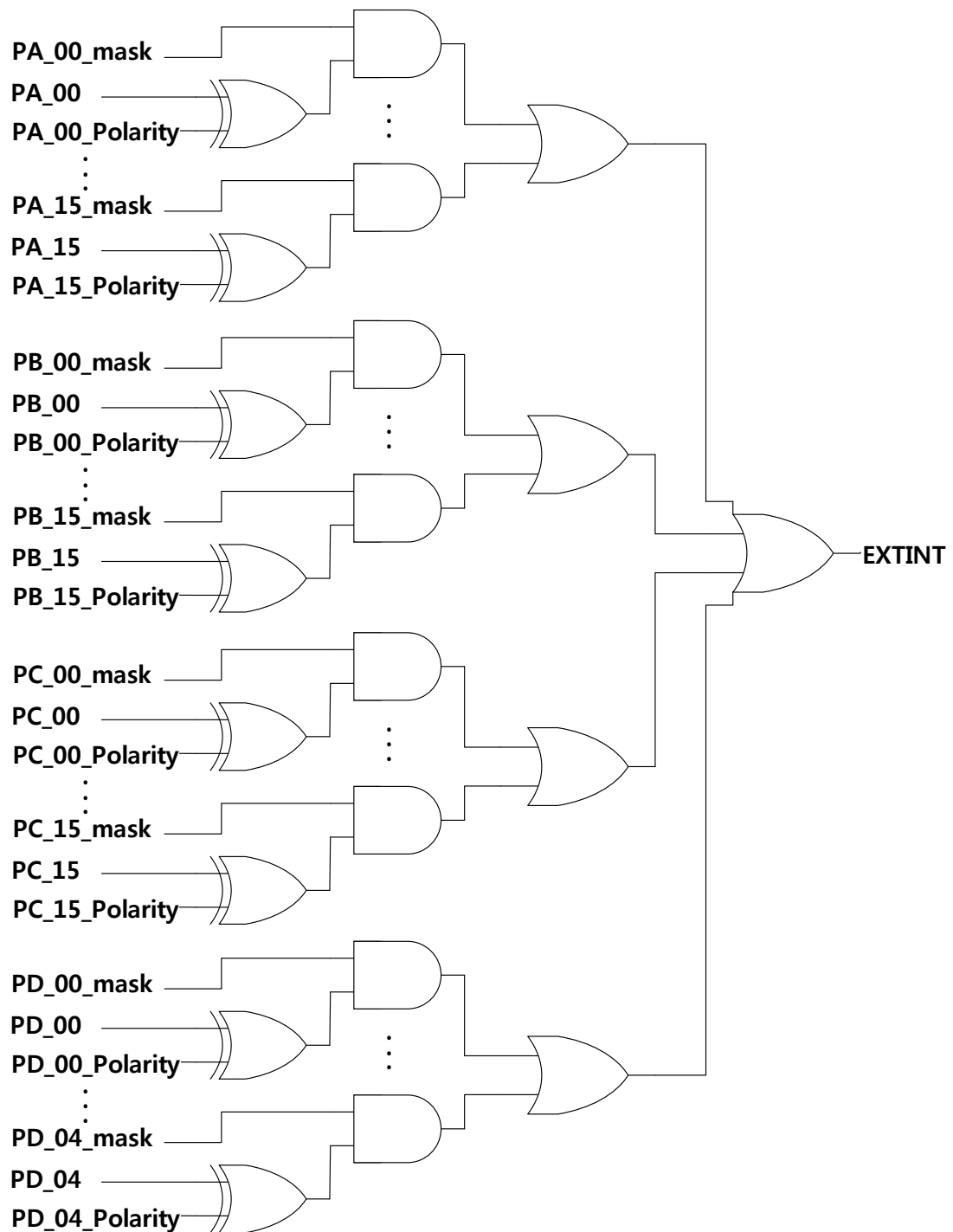


Figure 13. External Interrupt diagram

14 Pad Controller (PADCON)

14.1 Introduction

Pads of W7500 are controllable. User can control pad's characteristic.

14.2 Features

- W7500 has digital I/O pads and digital/analog mux I/O pads
- Controllable characteristics of pads are pull-up, pull-down, driving strength, input enable, and CMOS/Schmitt trigger input buffer
- Each pad can be controled individually by register.

14.3 Functional description

Figure 14 shows the function schematic of digital I/O pad of W7500.

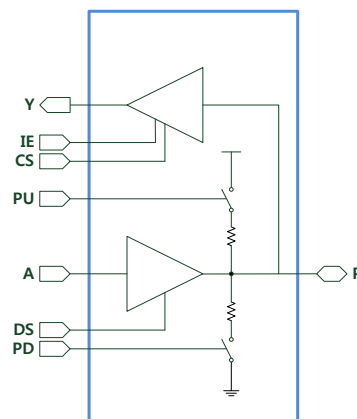


Figure 14. function schematic of digital I/O pad

Figure 15 shows the function schematic of digital/analog mux IO pad of W7500

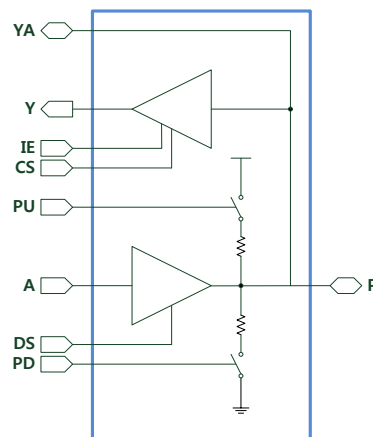


Figure 15. function schematic of digital/analog mux IO pad

Initials of Pad diagram is same as below.

P - PAD

YA - Analog Input (connect to ADC input)

Y - Digital Input

IE - Input buffer enable

| Condition | | A | Y | P |
|----------------------------------|-------------|--------|---------|-----|
| Input buffer enable (IE = 1) | Output mode | OUT | OUT | OUT |
| | Input mode | No use | IN | IN |
| Input buffer disable (IE = 0) | Output mode | OUT | Low (0) | OUT |
| | Input mode | No use | IN | IN |

CS - CMOS/Schmitt trigger input buffer select

PU - Pull-up enable

A - Digital Output

DS - Driving strength select

| Condition | | Rise/Fall Time (nSec) | | Propagation Delay (nSec) | |
|------------------|---------------------|-----------------------|-----|--------------------------|-----|
| Driving Strength | Capacitance loading | Min | Max | Min | Max |
| High (DS = 1) | 25pF | 4 | 18 | 7 | 27 |
| | 100pF | 11 | 53 | 11 | 44 |
| Low (DS = 0) | 25pF | 1 | 8 | 4 | 16 |
| | 100pF | 4 | 23 | 7 | 24 |

PD - Pull-down enable

User can set pad condition with IE, CS, PU/PD, DS by register.

And pads are can be controlled individually.

15 General-purpose I/Os(GPIO)

15.1 Introduction

The GPIO(General-Purpose I/O Port) is composed of four physical GPIO blocks, each corresponding to an individual GPIO port(PORT A, PORT B, PORT C, PORT D). The GPIO supports up to 53 programmable input/output pins, depending on the peripherals being used.

15.2 Features

- The GPIO peripheral consists the following features.
 - GPIO_DATAOUT can SET/CLEAR by the SET register and CLEAR register. (1 for set and 0 for clear)

- Mask registers allow treating sets of port bits as a group leaving other bits unchanged.
- Up to 53 GPIOs depending on configuration
- Programmable control for GPIO interrupts
 - Interrupt generation masking
 - Edge-triggered on rising, falling, or both

15.3 Functional description

Figure 16 shows the GPIO block diagram.

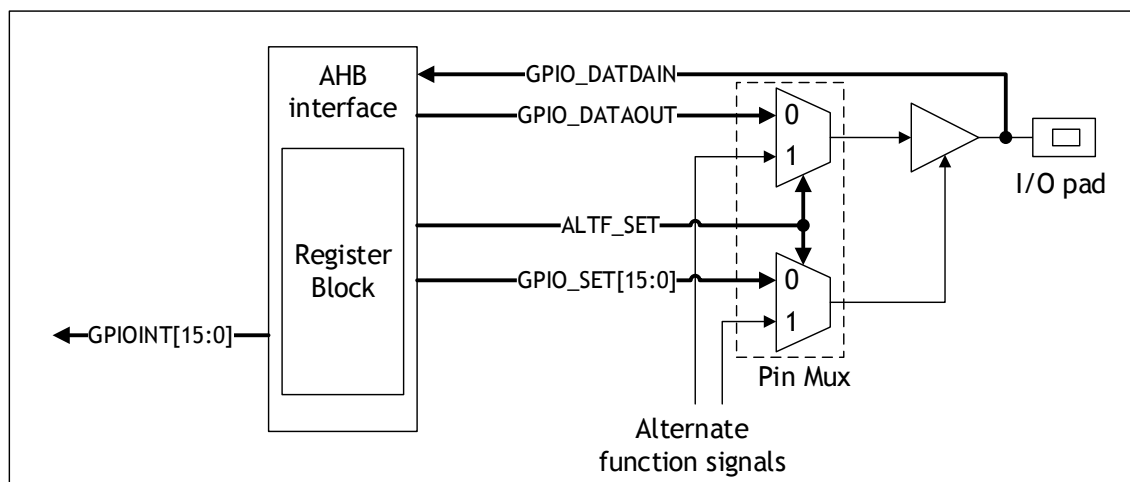


Figure 16. GPIO block diagram

Figure 17 shows the operation sequences available for the GPIO.

The pad alternate function is using the pad alternate function select register.

Refer to '12. Alternate Function Controller (AFC)' for more details about each register.

The pad control supports pull-down, pull-up, input buffer, and summit trigger input buffer.

Refer to '14. Pad Controller (PADCON)' for more details about each register.

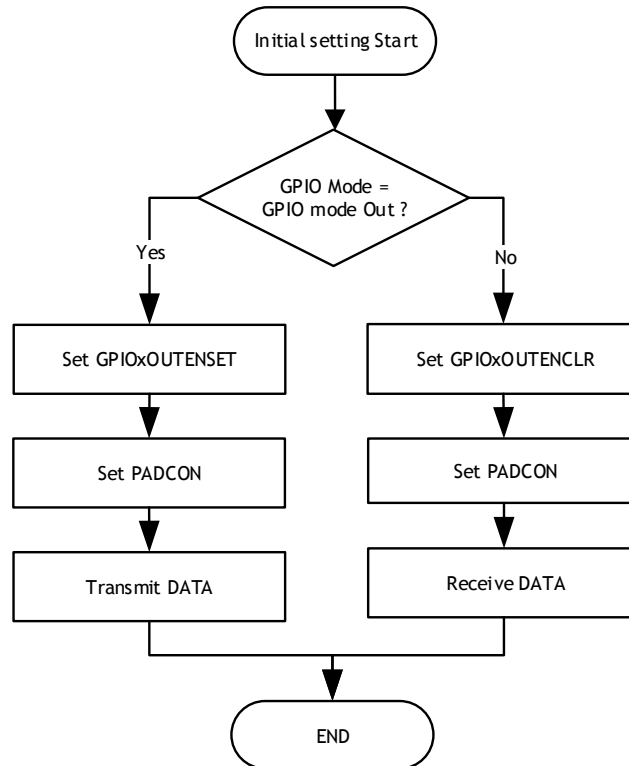


Figure 17. GPIO Flow chart

15.3.1 Masked access

The masked access feature permits individual bits or multiple bits to be read from or written to in a single transfer. This avoids software-based read-modify-write operations that are not thread safe. With the masked access operations, the 16-bit I/O is divided into two halves, lower byte and upper byte. The bit mask address spaces are defined as two arrays each containing 256 words.

For example, to set bits[1:0] to 1 and clear bits[7:6] in a single operation, users can carry out the write to the lower byte mask access address space. The required bit mask is 0xC3, and users can write the operation as MASKLOWBYTE[0xC3] = 0x03. Refer Figure 18 below.

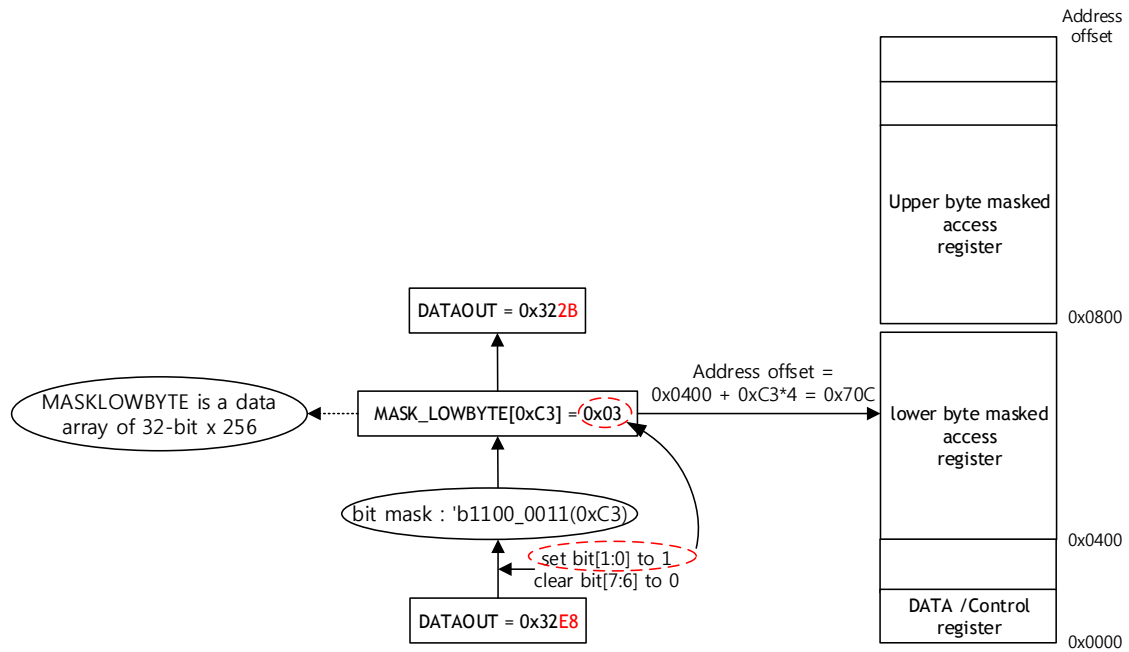


Figure 18. MASK LOWBYTE access

To update some of the bits in the upper eight bits of the GPIO port, users can use the MASKHIGHBYTE array as Figure 19 below.

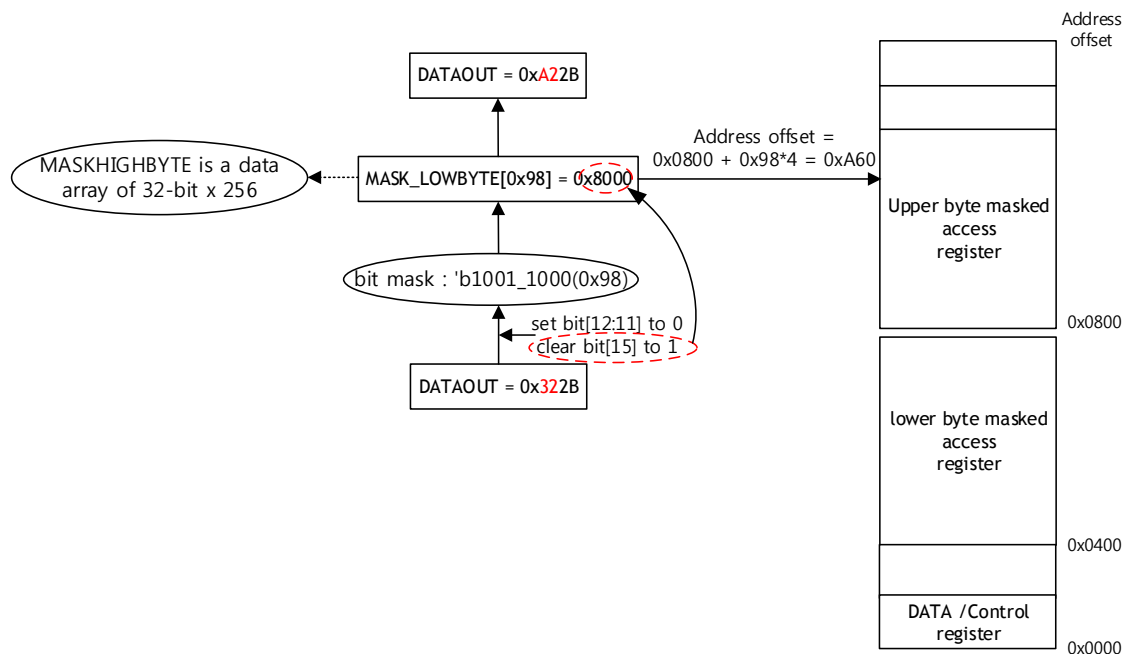


Figure 19 MASK HIGHBYTE access

16 Direct memory access controller (DMA)

16.1 Introduction

Direct memory access (DMA) is used in order to provide high-speed data transfer between peripherals and memory as well as memory to memory. Data can be quickly moved by DMA without any CPU actions. This keeps CPU resources free for other operations.

The DMA controller has up to 6 channels in total, each dedicated to managing memory access requests from one or more peripherals. It has an arbiter for handling the priority between DMA requests. For more details, refer to “PrimeCell® μ DMA Controller (PL230)” from the Technical Reference Manual

16.2 Features

- 6 channels
- Each channel is connected to dedicated hardware DMA requests and software trigger is also supported on each channel.
- Priorities between requests from the DMA channels are software programmable (2 levels consisting of high, default)
- Memory-to-memory transfer (software request only)
- TCP/IP-to-memory transfer (software request only)
- SPI/UART-to-memory transfer (hardware request and software request)
- Access to Flash, SRAM, APB and AHB peripherals as source and destination

16.3 Functional description

Figure 20 shows the DMA block diagram.

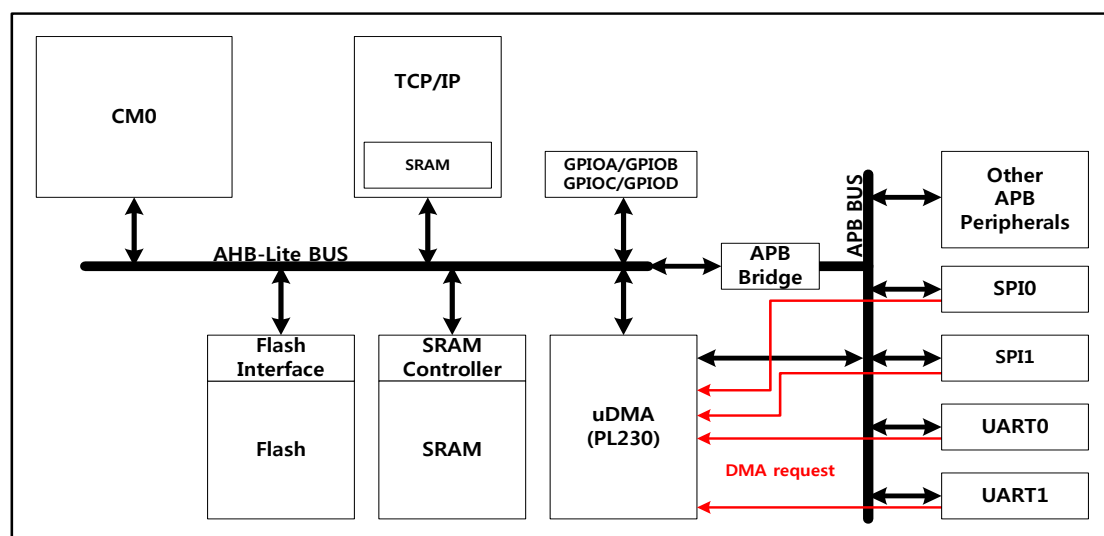


Figure 20. DMA Block diagram

16.3.1 DMA request mapping

The hardware requests from the peripherals (UART0, UART1, SSP0, SSP1) are simply connected to the DMA. Refer to Table 9 which lists the DMA requests for each channel.

Table 9 Summary of the DMA requests for each channel

| | Channel 1 | Channel 2 | Channel 3 | Channel 4 | Channel 5 | Channel 6 |
|---------------------------------|--------------------|--------------------|----------------------|----------------------|-----------|-----------|
| Hardware Request | SSP0_TX SSP0_RX | SSP1_TX SSP1_RX | UART0_TX UART0_RX | UART1_TX UART1_RX | NONE | NONE |
| Software Request ⁽¹⁾ | Support | Support | Support | Support | Support | Support |

1. Software request is the only way to use DMA for memory-to-memory or TCP/IP-to-memory.

16.3.2 DMA arbitration

The controller can be configured to perform arbitration during a DMA cycle before and after a programmable number of transfers. This reduces the latency for servicing a higher priority channel.

The controller uses four bits in the channel control data structure that configures how many AHB bus transfers occur before the controller re-arbitrates. These bits are known as the R_power bits because the value R is raised to the power of two and this determines the arbitration rate. For example, if $R = 4$, then the arbitration rate is 2^4 , which means the controller arbitrates every 16 DMA transfers.

Remark: Do not assign a low-priority channel with a large R_power value because this prevents the controller from servicing high-priority requests until it re-arbitrates.

When $N > 2^R$ and is not an integer multiple of 2^R , then the controller always performs sequences of 2^R transfers until $N < 2^R$ remain to be transferred. The controller performs the remaining N transfers at the end of the DMA cycle.

16.3.3 DMA cycle types

The cycle_ctrl bits in the channel control data structure control how the DMA controller performs a cycle.

The controller uses four cycle types described in this manual:

- Invalid
- Basic

- Auto-request
- Ping-pong

See ARM micro DMA (PL230) documentation for additional cycle types.

For all cycle types, the controller arbitrates after 2^R DMA transfers. If a low-priority channel is set to a large 2^R value then it prevents all other channels from performing a DMA transfer until the low-priority DMA transfer completes. Therefore, the user must be cautious when setting the R_power bit in the channel_cfg data structure so that the latency for high-priority channels is not significantly increased.

16.3.3.1 Invalid cycle

After the controller completes a DMA cycle, it sets the cycle type to invalid to prevent it from repeating the same DMA cycle.

16.3.3.2 Basic cycle

In this mode, the controller can be configured to use either the primary or the alternate channel control data structure. After the channel is enabled and the controller receives a request for this channel, the flow for basic cycle is as below:

1. The controller performs 2^R transfers.
If the number of transfers remaining is zero the flow continues at step 3.
2. The controller arbitrates:
 - If a higher-priority channel is requesting service, then the controller services that channel.
 - If the peripheral or software signals a request to the controller, then it continues at step 1.
3. The controller sets dma_done[c] signal for this channel HIGH for one system clock cycle.
This indicates to the host processor that the DMA cycle is complete.

16.3.3.3 Auto-request cycle

When the controller operates in this mode, it is only necessary to receive a single request to enable the controller to complete the entire DMA cycle. This enables a large data transfer to occur, without significantly increasing the latency for servicing higher priority requests or requiring multiple requests from the processor or peripheral.

The auto-request cycle is typically used for memory-to-memory requests. In this case, software generates the starting request for the 2^R transfers after setting up the DMA control data structure.

In this mode, the controller can be configured to use either the primary or the alternate channel control data structure. After the channel is enabled and the controller receives a request for this channel, the flow for the auto-request cycle is as below:

1. The controller performs 2^R transfers. If the number of transfers remaining is zero the flow continues at step 3.
2. The controller arbitrates if there are any transfers remaining after 2^R transfers. If the current channel *c* has the highest priority, the cycle continues at step 1.
3. The controller sets `dma_done[c]` signal for this channel HIGH for one system clock cycle. This indicates to the host processor that the DMA cycle is complete.

16.3.3.4 Ping-pong cycle

In this mode, the controller performs a DMA cycle using one of the data structures and then performs a DMA cycle using the other data structure. The controller continues to switch between primary and alternate structures until it either reads a data structure that is invalid, or until the user reprograms the `cycle_type` to basic, or until the host processor disables the channel.

In ping-pong mode, the user can program or reprogram one of the two channel data structures (primary or alternate) while using the other channel data structure for the active transfer. When a transfer is done, the next transfer can be started immediately using the prepared channel data structure - provided that a higher priority channel does not require servicing. If the user does not reprogram the channel control data structure not in use for a transfer, the cycle type remains invalid (which is the value at the end of the last transfer using that structure) and the ping-pong cycle completes.

The ping-pong cycle can be used for transfers to or from peripherals or for memory-to-memory transfers.

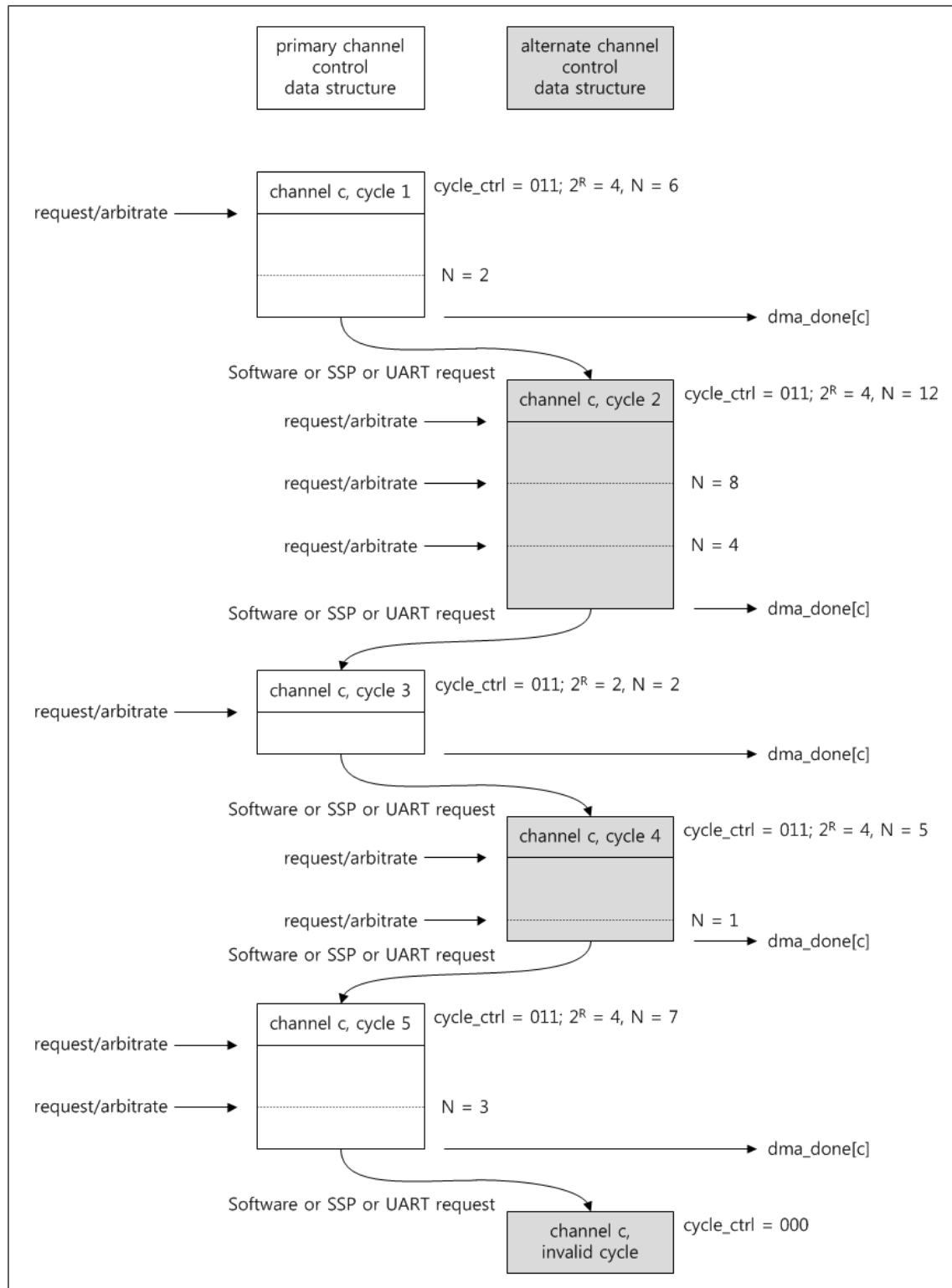


Figure 21. DMA ping pong cycle

17 Analog-to-digital converter (ADC)

17.1 Introduction

ADC is a 12bit analog-to-digital converter. It has up to 9 multiplexed channels allowing it to measure signals from 8 externals and 1 internal source.

ADC of various channels can be performed in single mode. The result of the ADC is stored in 12 bit register.

17.2 Features

- 12bit configuration resolution
- Conversion time : Max 10MHz (Sampling time can be programmable)
 - 8 channel for external analog inputs
 - CH0 : PC_15
 - CH1 : PC_14
 - CH2 : PC_13
 - CH3 : PC_12
 - CH4 : PC_11
 - CH5 : PC_10
 - CH6 : PC_09
 - CH7 : PC_08
 - 1 channel for internal LDO(1.5v) voltage.
 - CH15 : Internal voltage
- Start of conversion can be initiated by software.
- Convert selected inputs once per trigger.
- Interrupt generation at the end of conversion.

17.3 Functional description

Figure 22 shows the ADC block diagram.

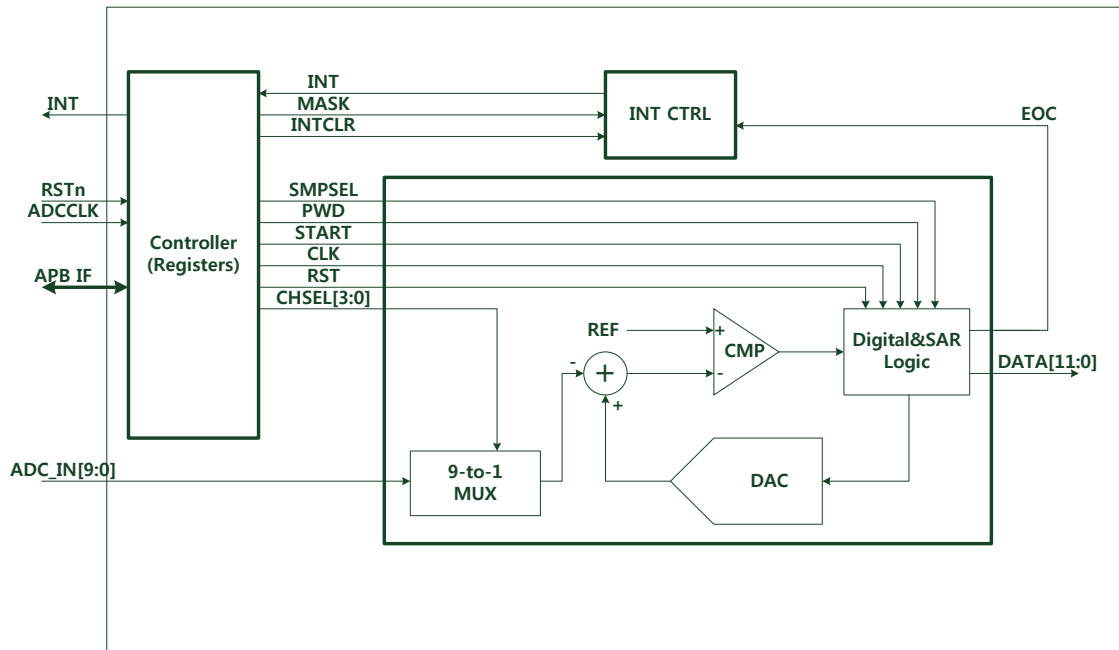


Figure 22. ADC block diagram

17.3.1 Operation ADC with non-interrupt

Figure 23 shows the flowchart of ADC operation with non-interrupt.

ADC can be used as below:

1. ADC needs to be initialized before operation.
To initialize the ADC, clear the PWD bit first.
2. Select the ADC channel from 0 to 7 and 15 (initial core voltage).
3. Run start ADC conversion by set ADC_SRT bit.
4. Check INT bit to know finish of conversion.
5. If INT bit is high (1), read ADC conversion data.
6. Finally, ADC operation is finished by setting the PWD bit.

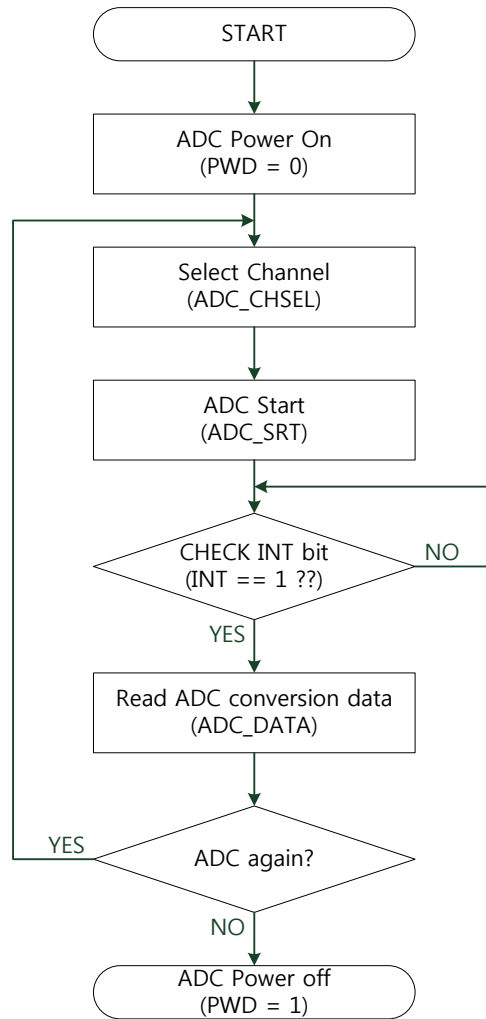


Figure 23. The ADC operation flowchart with non-interrupt

17.3.2 Operation ADC with interrupt

Figure 24 shows the flowchart of ADC operation with interrupt.

Operation is almost the same as the non-interrupt mode except checking INT register to know when enabling interrupt mask bit and conversion is completed.

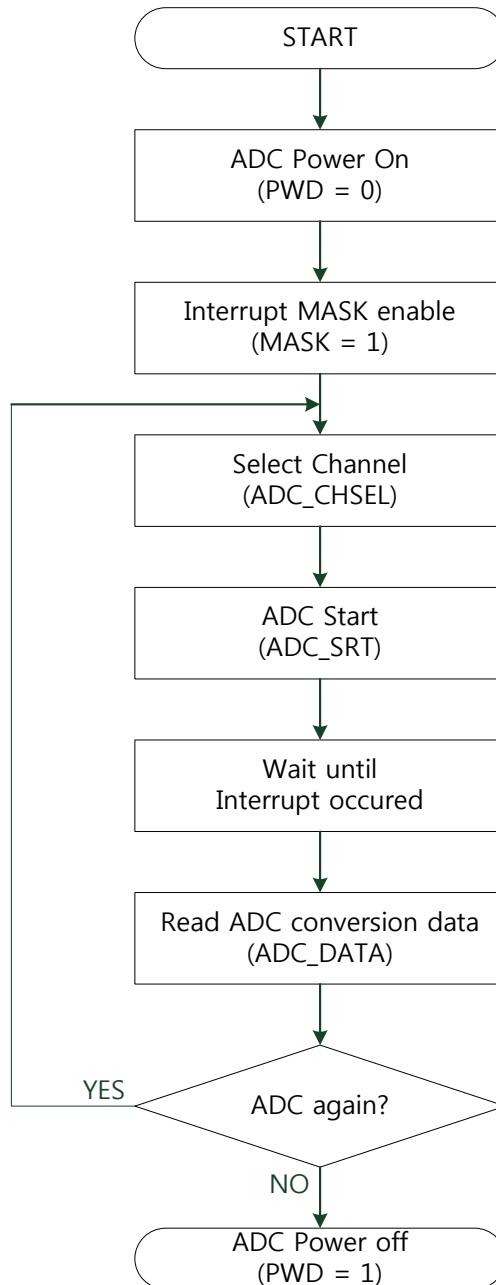


Figure 24. The ADC operation flowchart with interrupt

18 Pulse-Width Modulation (PWM)

18.1 Introduction

The PWM consists a 8-channel 32-bit Timer/Counter driven by a programmable prescaler. The function of the PWM is based on the basic Timer. Each timer and counter runs independently.

The PWM can be used to control the width of the pulse, formally the pulse duration, to generate output waveform or to count the counter triggered by external input.

18.2 Features

- Counter or Timer operation can use the peripheral clock, external clock source, or one of the capture inputs as the clock source.
- Eight independent 32-bit Timer/Counter driven by a programmable 6 bits prescaler runs as the PWM or standard timer if the PWM mode is not enabled.
- Eight PWM output waveforms.
- Each of Timer/Counter can have different or same clock source.
- Counter or timer operation.
- Eight capture registers that can take the timer value when an external input signal. A capture event can generate an interrupt signal optionally.
- 32-bit match register and limit register.

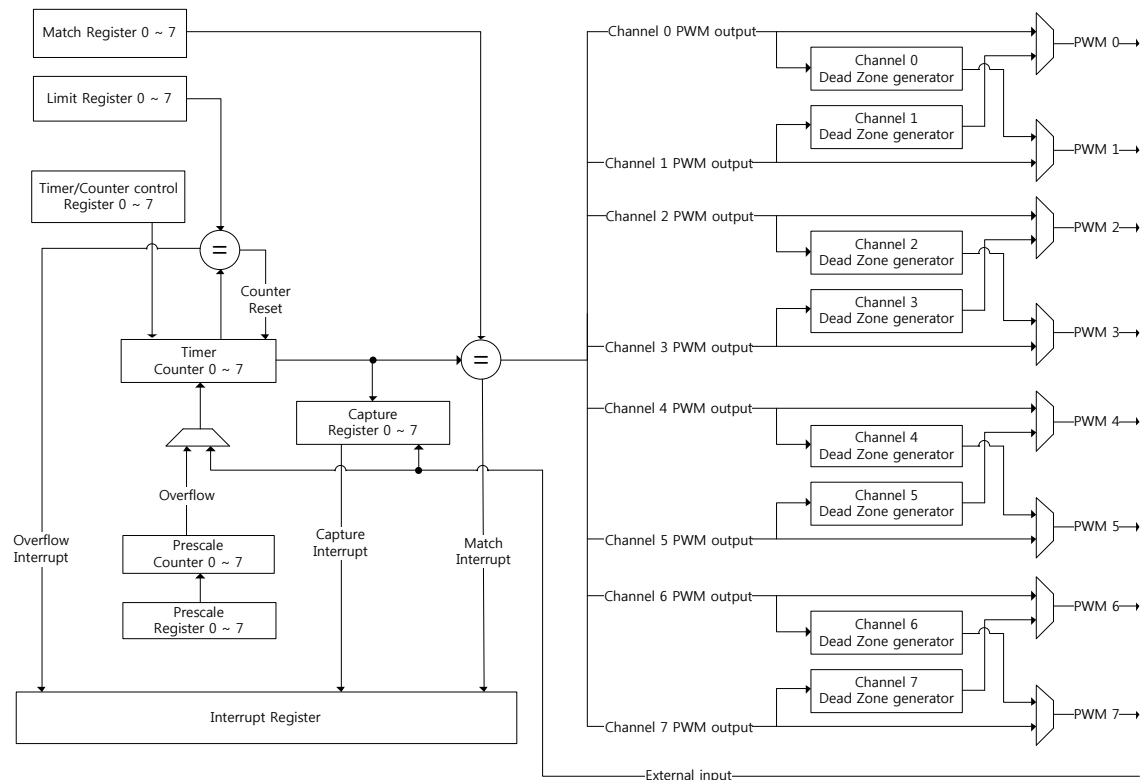


Figure 25. PWM block diagram

18.3 Functional description

18.3.1 Timer/Counter control

The PWM has Start/Stop register. It controls start or stop of the Timer/Counter. If you set this register, the Timer/Counter starts to run. If you reset this register, the Timer/Counter stops immediately. Also there is a pause register. The pause register is used to stop temporarily after one period. Although you set this register while the Timer/Counter is running, the Timer/Counter will stop when the period ends.

The registers of PWM can be updated when it stops or pauses. Users cannot update the registers while PWM is running.

18.3.2 Timer/Counter

The PWM has 8 Timer/Counter clocks, which can be divided by a prescaler. Each Timer/Counter runs independently. The Timer/Counter is designed to count cycles of the clocks or external input signal and generate interrupts when specified timer values are occurred based on match register and limit register. The Timer/Counter can count up or down.

The PWM has match registers and limit registers. The match registers control the duty cycle of PWM output waveform. The limit registers control the period of the PWM output waveform. The Timer/Counter becomes 0 when it reaches value of the limit register. If PDMR(Periodic Mode Register) is set, the Timer/Counter counts repeatedly and if PDMR is reset, the Timer/Counter stops counting.

Match register should be smaller than limit register(LR). If not, match interrupt is not occurred and PWM output waveform is always 1.

Repetition mode

The Timer/Counter has two repetition mode: periodic and one-shot mode. In periodic mode, the Timer/Counter recycles and then restarts when the Timer/Counter reaches the value of limit register. Figure 26 shows periodic mode timing diagram.

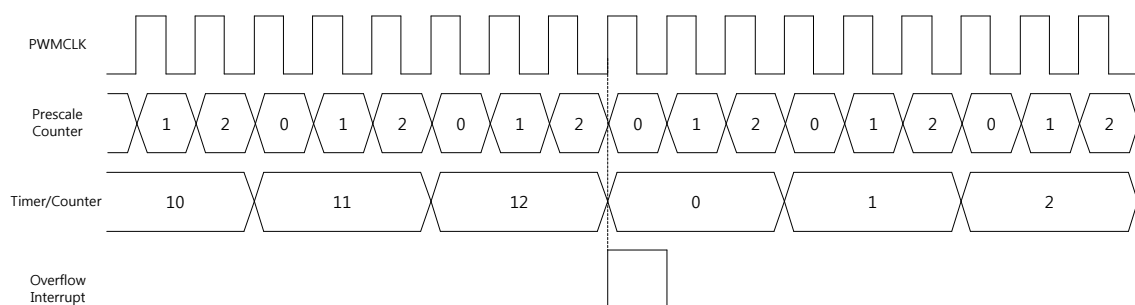


Figure 26. Periodic mode

In one-shot mode, the Timer/Counter resets to the initial value and then stops when the Timer/Counter reaches the value of limit register. Figure 27 shows one-shot mode timing diagram.

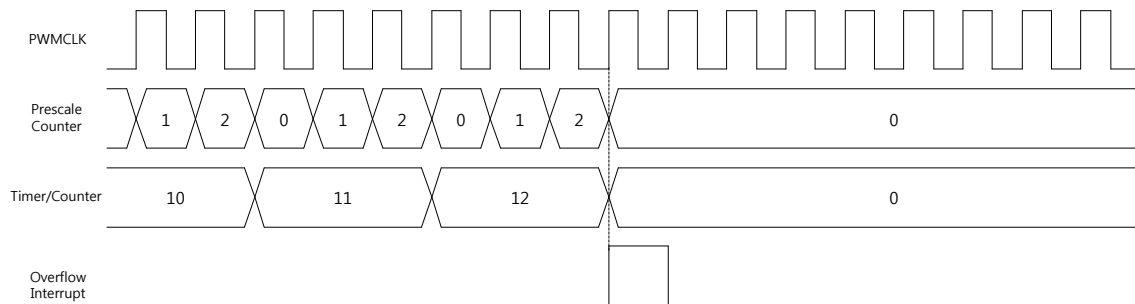


Figure 27. one-shot mode

Counting mode

The Timer/Counter has two counting mode: Up-count and Down-count mode. In up-count mode, the Timer/Counter counts up from 0 to the limit register value and then recycles. If repetition mode is periodic, the Timer/Counter restarts and if repetition mode is one-shot mode, the Timer/Counter stops. Figure 28 shows up-count mode timing diagram.

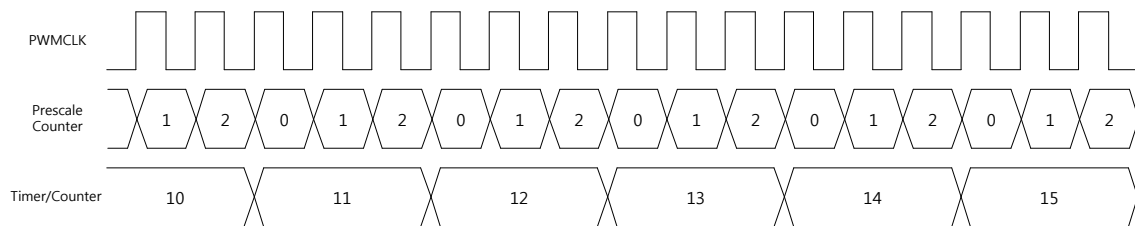


Figure 28. Up-count mode

In Down-count mode, the Timer/Counter counts from 0xFFFF_FFFF, then recycles. If repetition mode is periodic, the Timer/Counter restarts and if repetition mode is one-shot mode, the Timer/Counter stops. Figure 29 shows down-count mode timing diagram.

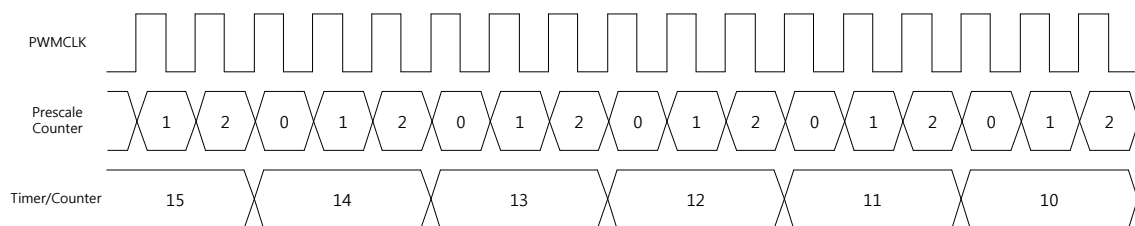


Figure 29. Down-count mode

Timer and Counter mode

The Timer/Counter can run in timer mode or counter mode. In timer mode, the Timer/Counter is counted by PWMCLK after Prescale counter is overflowed. If prescale is set by 0, the Timer/Counter counts every PWMCLK period. In counter mode, the Timer/Counter is counted by external input signal. There are three counting method: rising edge, falling edge, and both

edge. The counter mode has up-count or down-count mode and also has periodic or one-shot mode. The external input pin and PWM output pin are the same, so PWM output is disabled in counter mode.

Figure 30 is counter mode example with rising edge mode,

Figure 31 is with falling edge mode and

Figure 32 is with both rising and falling edge mode.

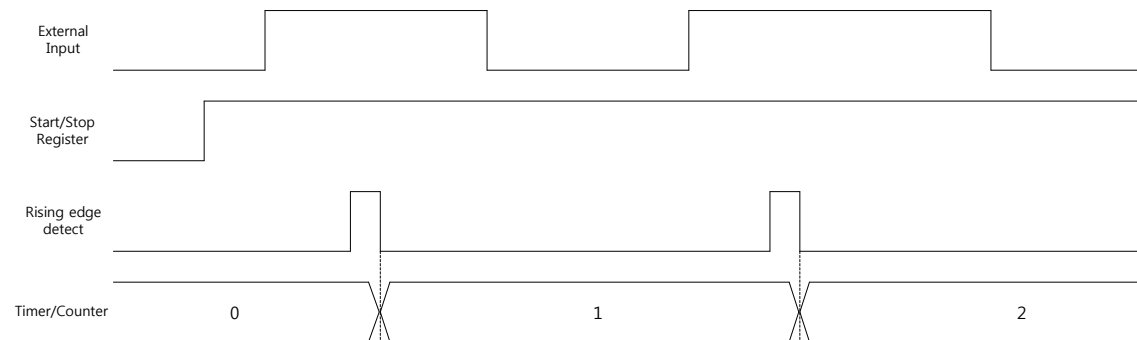


Figure 30 Counter mode with rising edge

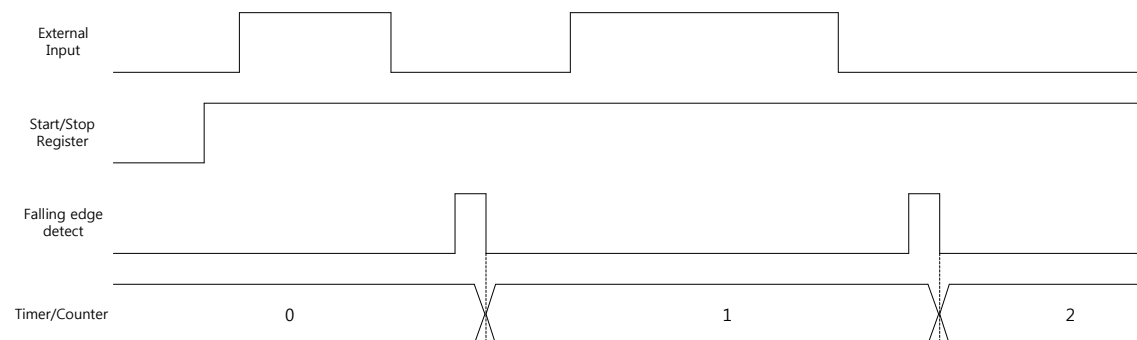


Figure 31 Counter mode with falling edge

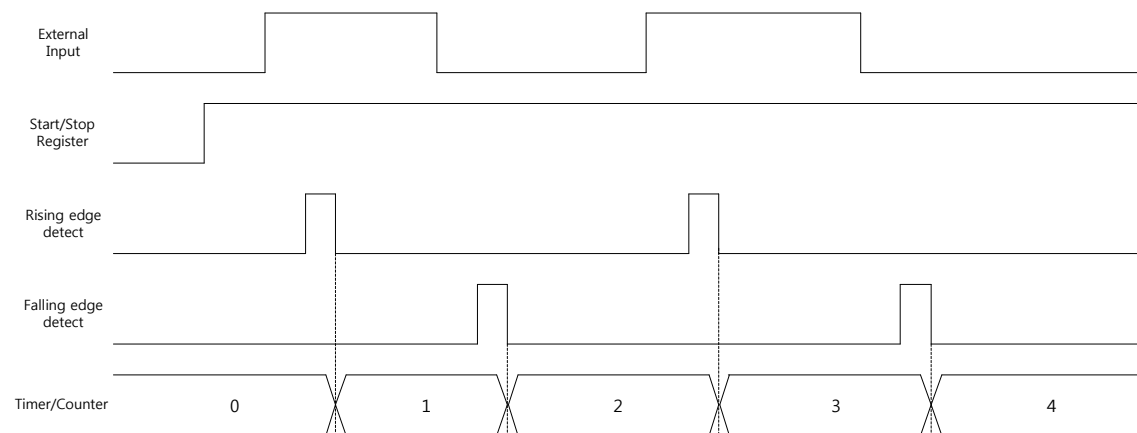


Figure 32 Counter mode with rising and falling edge

Prescaler description

The PWM has 6-bit prescale counter(PC) and the prescaler can divide the Timer/Counter clock frequency. Users can control it by Prescale Register(PR).

Figure 33 and Figure 34 shows some examples of the Timer/Counter timing with prescale register is 2, match register is 2, limit register is 12, timer mode, periodic mode, up-count mode, and no interrupt clear.

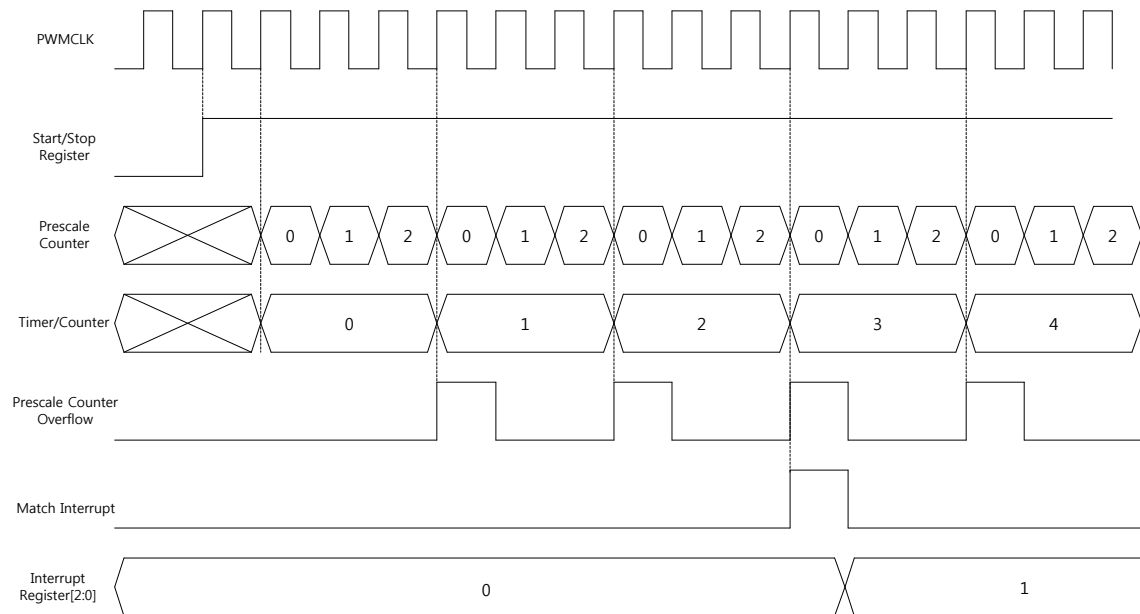


Figure 33 Timer/Counter timing diagram with match interrupt

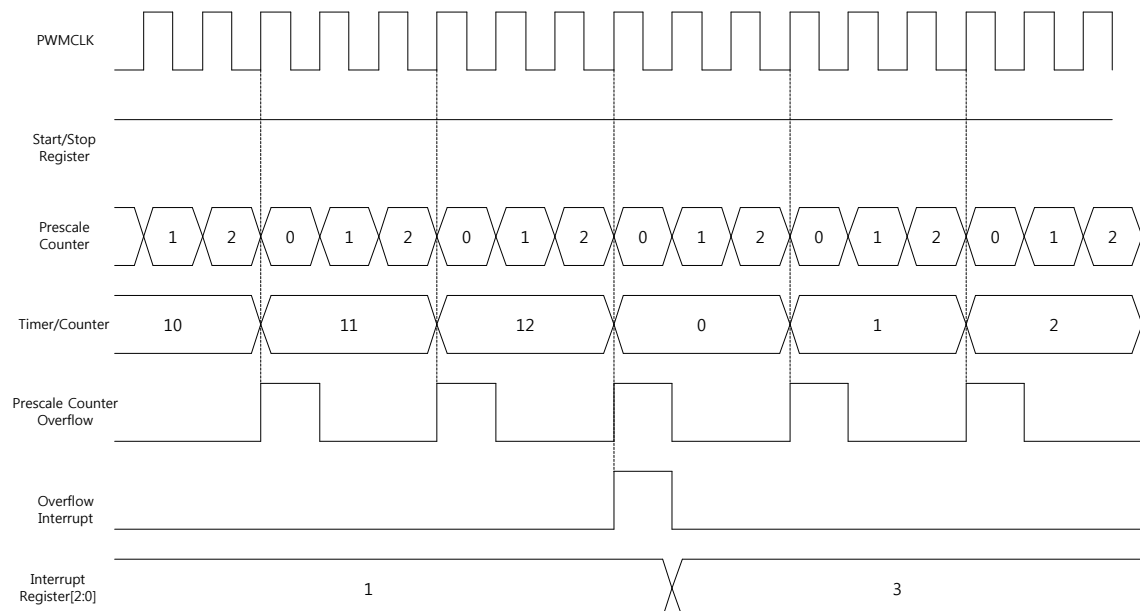


Figure 34 Timer/Counter timing diagram with overflow interrupt

18.3.3 PWM mode

Pulse Width Modulation mode generates a waveform with a period determined by the value of limit register and a duty cycle determined by the value of the match register.

The PWM output becomes always 1 when the Timer/Counter starts to count. Then the PWM output becomes 0 when the Timer/Counter reaches the value of match register. If the Timer/Counter is in periodic mode, the PWM output becomes 1 again when the Timer/Counter reaches the value of limit register. In one-shot mode, the PWM output does not change to 1 but stays 0 and the Timer/Counter stops.

The PWM mode can be selected independently on each channel(0~7) by PWM output enable and external input enable register. The external input pin and PWM output pin are the same, so external input is disabled in PWM mode.

Figure 35 is an example of the PWM output waveform when the Timer/Counter is reached to the value of match register.

Figure 36 is example of the PWM output waveform when to the Timer/Counter is reached to the value of limit register.

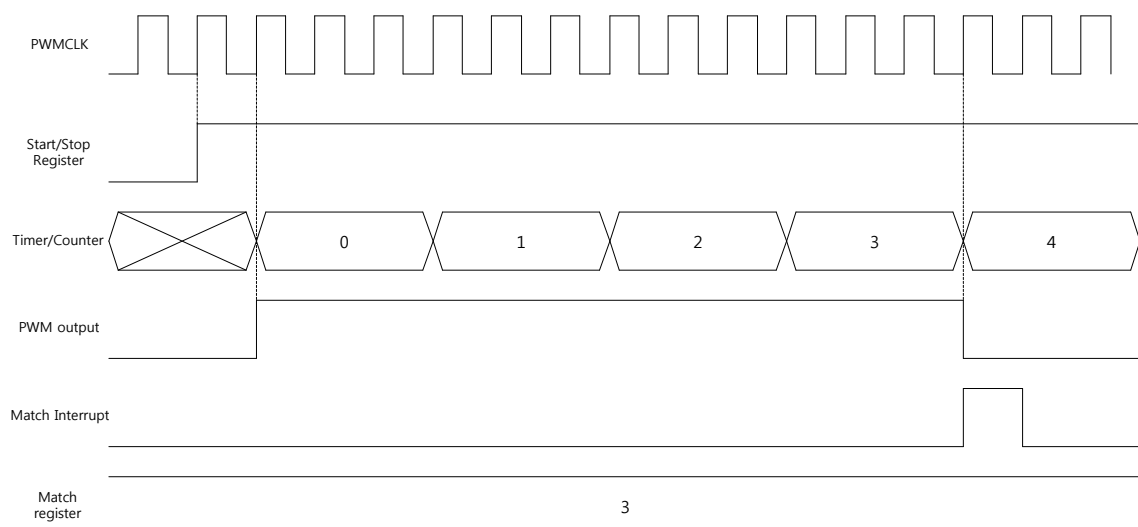


Figure 35 The PWM output up to match register

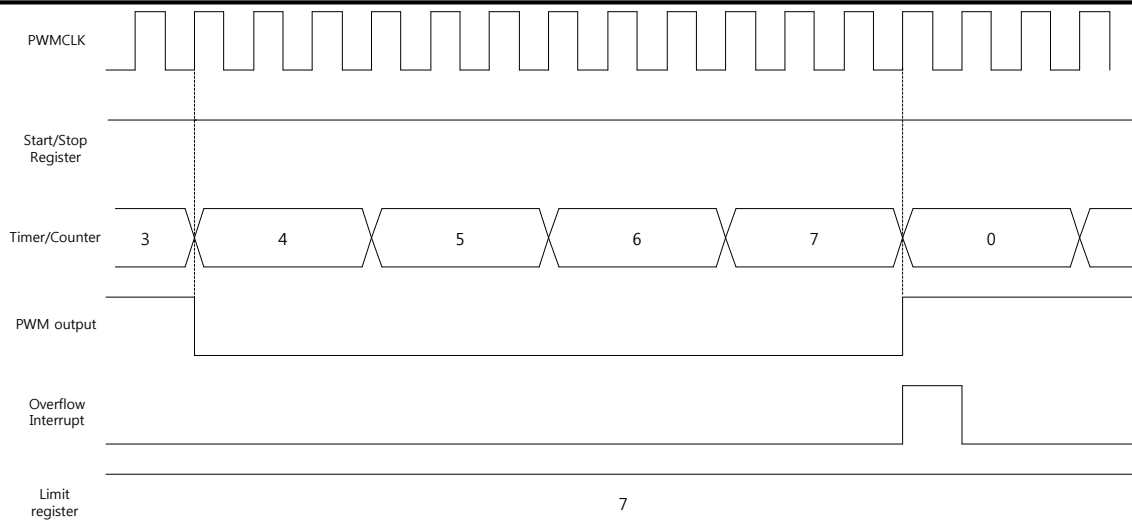


Figure 36 The PWM output up to limit register

If match register is set as 0, the PWM output will be 1 while the Timer/Counter is 0.

If the match register is bigger than the limit register, the PWM output is always 1.

18.3.4 Interrupt

The PWM has 8-bit interrupt enable register(IER) and each bit of IER corresponds to each interrupt of channel. Each PWM channel has Channel-n Interrupt Enable register(CHn_IER). The CHn_IER includes three types of interrupt: match, overflow, and capture. The match interrupt occurs when the Timer/Counter is reached to value of match register. The overflow interrupt occurs when the Timer/Counter is reached to value of limit register. The capture interrupt occurs when external input is entered for capture.

If interrupt occurs, corresponded bit of Channel-x interrupt register(CHn_IR) bit is set and PWM channel-n interrupt signal is generated. All CHn_IR is cleared by channel-n interrupt clear register(CHn_ICR) and then PWM channel-n interrupt signal is cleared.

18.3.5 Dead zone generation

Each PWM channel can output two complementary signals with dead zone time and it can be enabled by Channel-n Dead Zone Enable Register(CHn_DZER). Only 4 channels can be enabled because there are 8 PWM output pins. Channel 0 and 1 are a pair, channel 2 and 3 are a pair, channel 4 and 5 are a pair, and channel 6 and 7 are a pair. If users want to use channel-0 dead zone generation, channel-1 should be disabled. If channel 0 and 1 dead zone generation are enabled both, all outputs are 0. In that case, users should choose 1 channel.

Dead zone time are generated by the value of Channel-n Dead Zone Counter Register(DZCR). The dead zone counter counts up to value of DZCR. During the dead zone time, both

complementary signals are both 0. Users have to adjust the signal depending on the devices that are connected to the outputs and their characteristics. If DZCR is bigger than the limit register, main output signal is toggled 0 to 1 and then 1 to 0 while 1 PWMCLK and inverted output signal is always 0.

Figure 37 shows two complementary PWM outputs with dead zone time. During dead zone time, both outputs are 0. Figure 38 shows a more detailed timing with dead zone counter. The dead zone counter and the Timer/Counter starts to count together and PWM output is 0 until dead zone counter is reached to value of dead zone counter register. The PWM output becomes 1 and 0 when the Timer/Counter is reached to value of match register. The inverted PWM output is also 0 until dead zone counter is reached to value of dead zone counter register. Then inverted PWM output becomes 1 after dead zone counter is reached to the value of dead zone counter register.

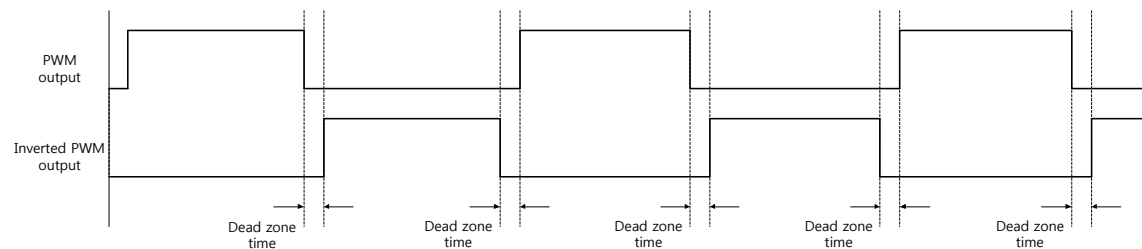


Figure 37 PWM waveform with dead zone time

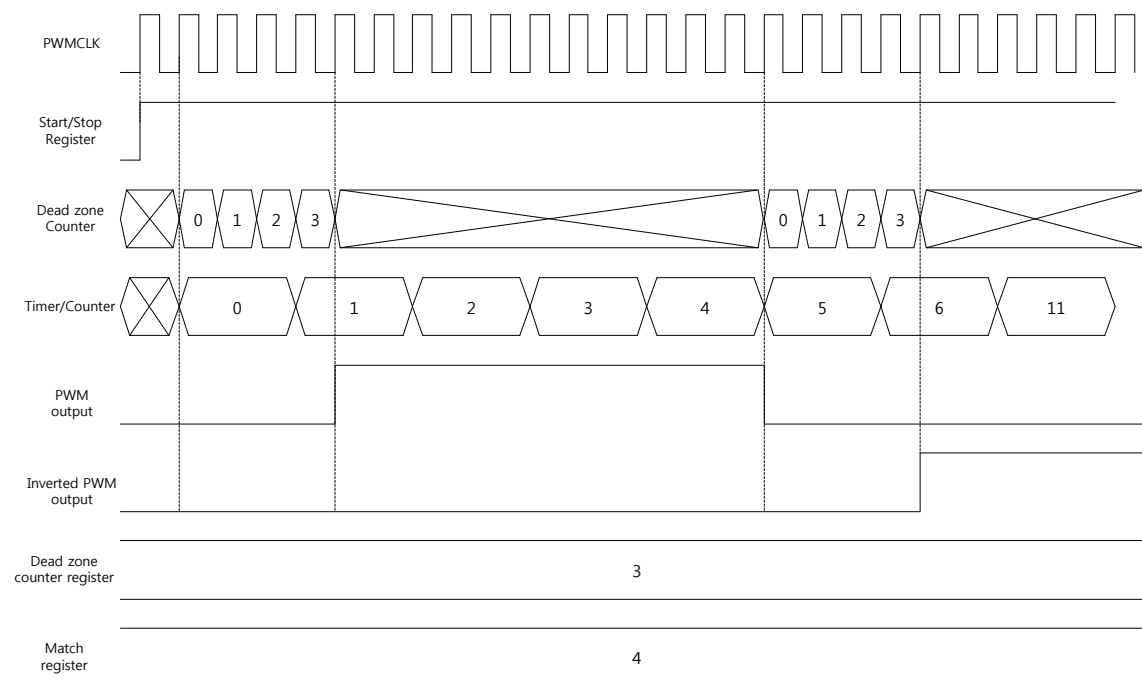


Figure 38 PWM waveform with dead zone counter

18.3.6 Capture event

Each PWM channel can capture its Timer/Counter value when an external input signal changes. Any channel could use any method of rising or falling edges. If capture interrupt is enabled, capture interrupt occurs when the external input signal is toggled. The Timer/Counter value is saved in Channel-n Capture Register(CHn_CR) and the capture register is not overwritten until capture interrupt is cleared. Figure 39 shows the capture event timing diagram. There is no interrupt clear, so second capture does not save during second rising edge detection.

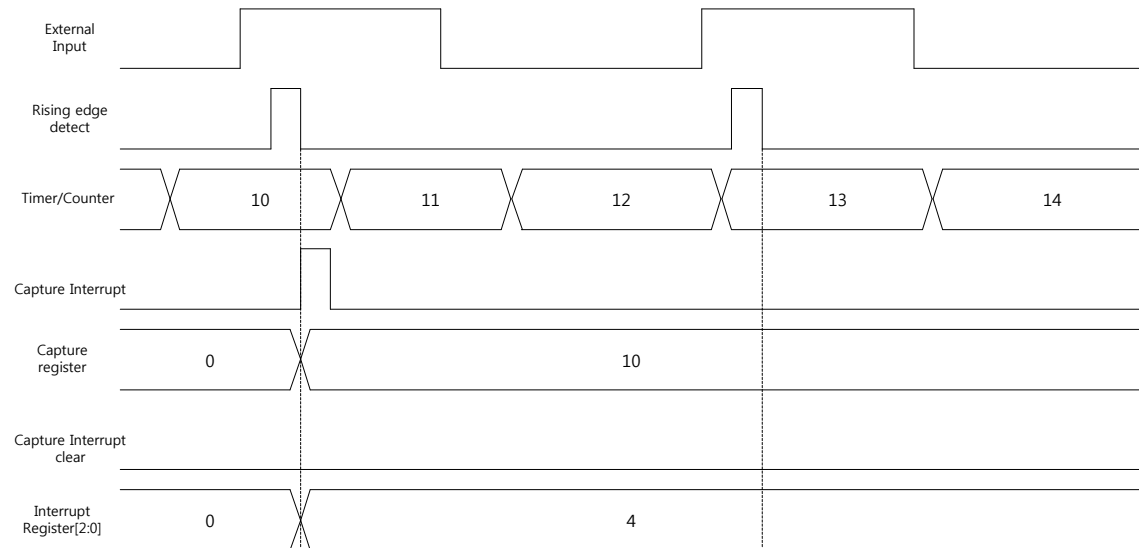


Figure 39 Capture event with no interrupt clear

Figure 40 shows, the capture event timing diagram with interrupt clear. The second capture is saved at the second rising edge detection because there is interrupt clear.

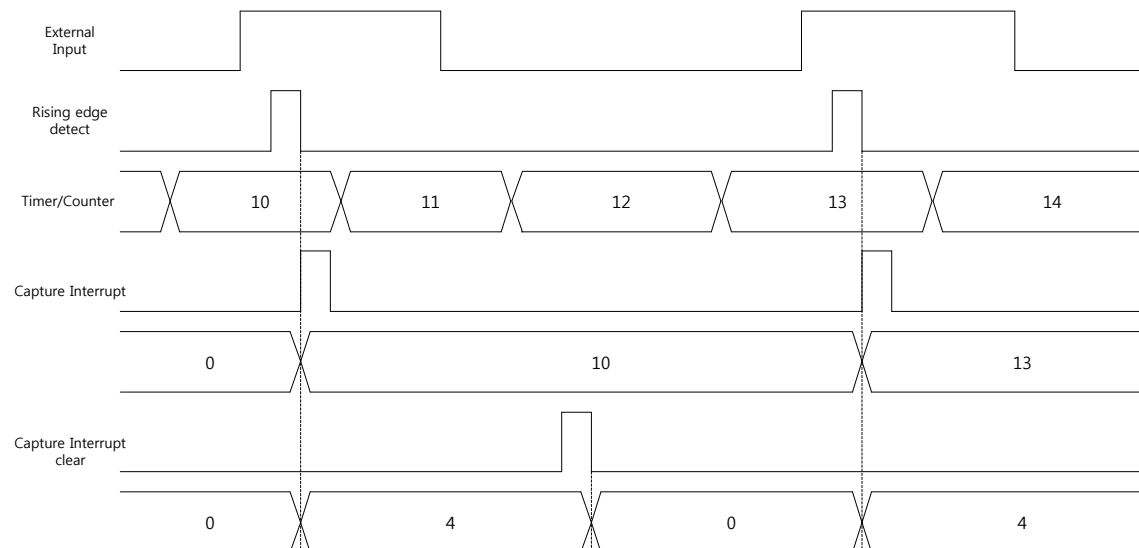


Figure 40 Capture event with interrupt clear

18.3.7 How to set the PWM

Figure 41 shows the PWM setting flow step by step.

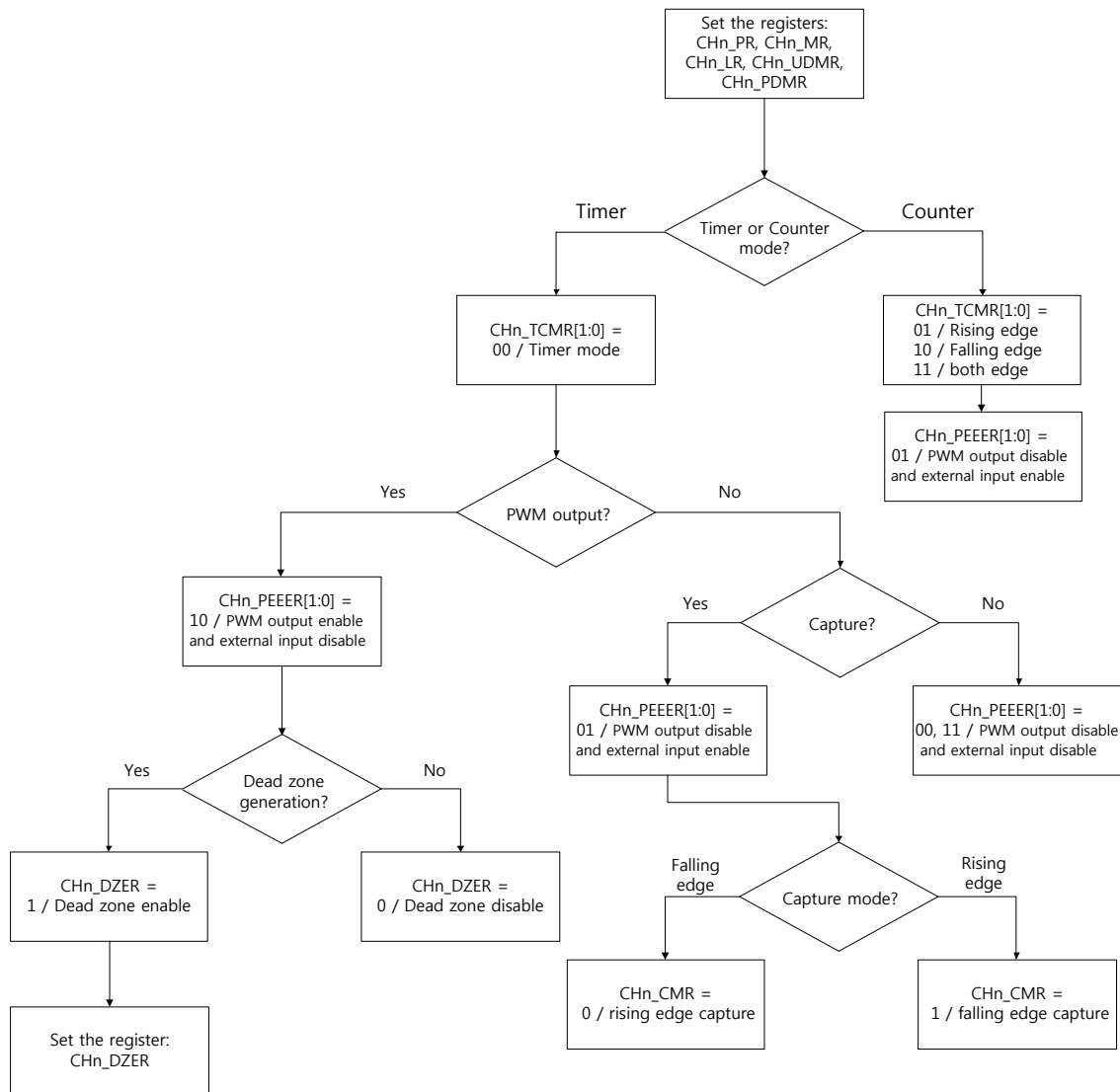


Figure 41. The PWM setting flow

19 Dual timers

19.1 Introduction

The dual timer consists two programmable 32-bit or 16-bit Free-running counters(FRCs) that can generate interrupts when they reach 0. There are two dual timers and 4 FRCs. One dual timer has one interrupt handler, resulting in two interrupts of timers. Also one dual timer has one clock but two clock enable signals. Users can select one repetition modes one-shot or wrapping mode, and wrapping mode consists free-running and periodic mode. Two FRCs are one set so two FRCs has one clock, reset, and interrupt but each FRC has an individual clock enable.

19.2 Features

- One dual timer has two Free-Running Counters(FRCs).
- One dual timer has one interrupt handler and one clock.
- One dual timer has two clock enable signals.
- There are 2 dual timers.
- A 32-bit or a 16-bit down counter.
- One of the following repetition modes: one-shot and wrapping mode.
- One of the following wrapping modes: Free-running and periodic mode.
- There is a prescaler that can divide down the clock rate by 1, 16, or 256.

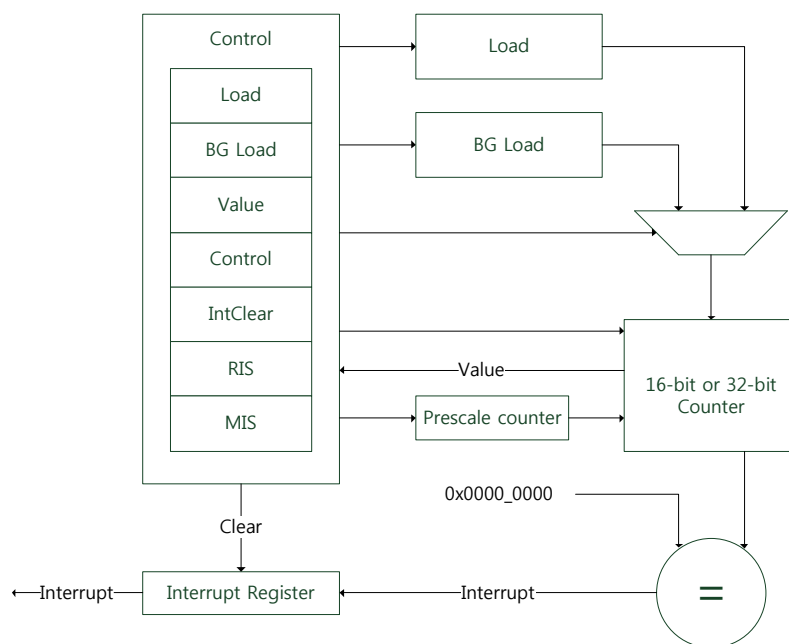


Figure 42 Block diagram of Dualtimer

19.3 Functional description

19.3.1 Clock and clock enable

The dual timers contain PCLK and TIMERCLK clock inputs. PCLK is the main APB system clock and is used by the register interface. TIMERCLK is the input to the prescale units and the decrementing counters. PCLK and TIMERCLK are synchronous.

The dual timers consist two programmable 32-bit Free-Running Counters(FRC) which operate independently. The two timers operate from one TIMERCLK but Each FRC is controlled independently by individual clock enable.

19.3.2 Timer size

Users can select FRC as 16-bit or 32-bit using the control register.

19.3.3 Prescaler

The timer has a prescaler that can divide down the enabled clock rate by 1, 16 or 256.

19.3.4 Repetition mode

There are two repetition mode: one-shot and wrapping mode. Wrapping mode has two modes: free-running and periodic mode.

One-shot mode

The counter generates an interrupt once. When the counter reaches 0, it halts until users reprogram it. Users can do this as below:

- Clear the one-shot count bit in the control register, in which case the count proceeds according to the selection of wrapping mode(free-running or periodic mode).
- Write a new value to the Load Value register.

Wrapping mode

Free-running mode

The counter wraps after reaching its zero value, and continues to count down from the maximum value. This is the default mode.

Periodic mode

The counter generates an interrupt at a constant interval, reloading the original value after wrapping past zero.

19.3.5 Interrupt

An interrupt is generated when the counter reaches 0 and is only cleared when the interrupt clear register is accessed.

The register holds the value until the interrupt is cleared.

Users can mask interrupts by writing 0 to the Interrupt Enable bit in the control register. Users can read the following from status registers:

- Raw interrupt status before masking.
- Final interrupt status after masking.

The interrupts from the individual timers after masking are logically ORed into a combined interrupt.

19.3.6 Operation

The operation of each timer is identical. The timer is loaded by writing to the load register and counts down to 0 if enabled. When a counter is already running, writing to the load register causes the counter to immediately restart at the new value. Writing to the background load value has no effect on the current count. In periodic mode, the counter continues to decrease to 0 and restart from the new load value.

An interrupt is generated when 0 is reached. Users can clear the interrupt by writing to the clear register. If users select one-shot mode, the counter halts when it reaches 0 until users deselect one-shot mode or write a new load value.

Otherwise, after reaching a zero count, if the timer is operating in free-running mode, it continues to decrease from its maximum value. If users select periodic mode, the timer reloads the count value from the load register and continues to decrease. In this mode, the counter effectively generates a periodic interrupt.

19.3.7 How to set the dual timers

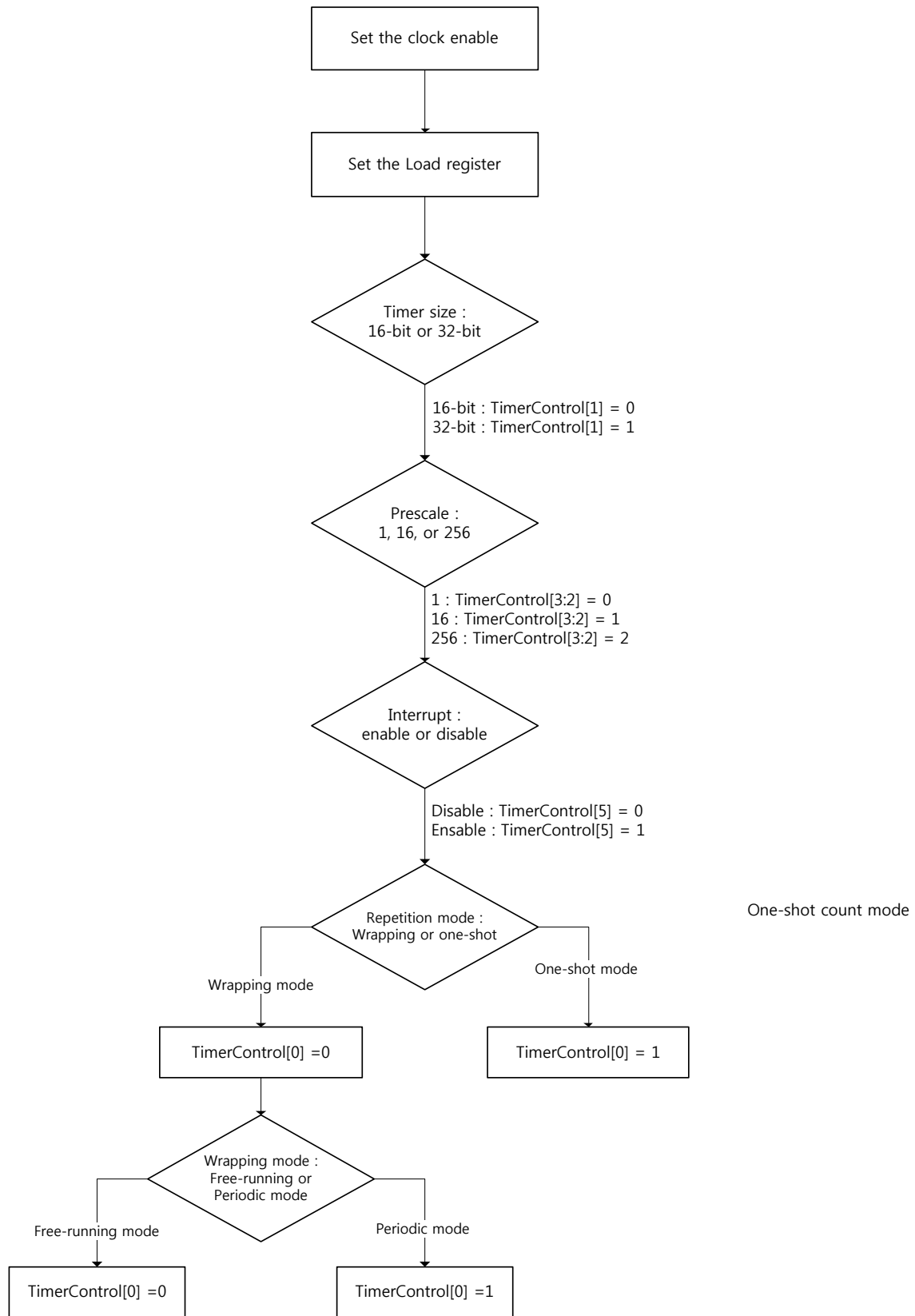


Figure 43 The Dual timer setting flow

20 Watchdog timer

20.1 Introduction

The watchdog is based on a 32-bit down-counter that is initialized from the Reload Register, WDTLoad. The watchdog generates a regular interrupt depending on a programmed value. The counter decreases by one on each positive clock edge of watchdog clock.

The watchdog monitors the interrupt and asserts a reset request signal when the counter reaches 0 and the counter is stopped. On the next enabled watchdog clock edge, the counter is reloaded from the WDTLoad Register and the countdown sequence continues. The watchdog reasserts the reset signal if the interrupt is not cleared by the time the counter next reaches 0.

The watchdog applies a reset to a system in the event of a software failure to provide a way to recover from software crashes. Users can enable or disable the watchdog unit as required.

20.2 Features

- 32-bit down counter.
- Internally resets chip if not periodically reloaded.
- The watchdog timer has lock register for to prevent rogue software from disabling the watchdog timer functionality.
- The watchdog timer clock(WDTCLK) and system clock(PCLK) are synchronous.

20.3 Functional description

20.3.1 Clock

The watchdog timer contains PCLK and WDTCLK clock inputs.

PCLK is the main APB system clock and is used by the register interface.

20.3.2 Interrupt and reset request

An interrupt is generated when the counter reaches 0 and is only cleared when the interrupt clear register is accessed.

The register holds the value until the interrupt is cleared.

Reset request is asserted when the counter reaches 0 repeatedly and is not reprogrammed.

Users can mask interrupts by writing 0 to the Interrupt Enable bit in the control register.

Users can read the following from status registers:

- Raw interrupt status, before masking.
- Final interrupt status, after masking.

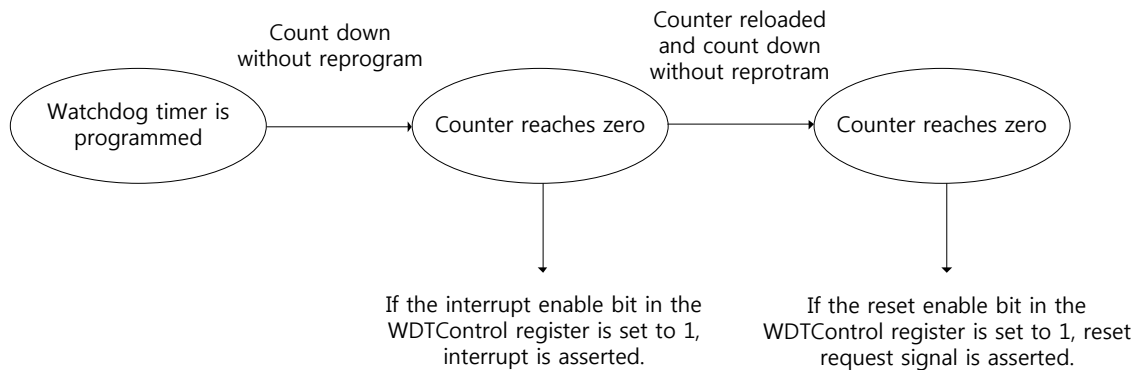


Figure 44. Watchdog timer operation flow diagram

21 Inter-integrated circuit interface (I2C)

21.1 Introduction

The I^2C (inter-integrated circuit) bus interface handles communications between the microcontroller and the serial I^2C bus. It supports standard speed mode(100Kbps).

In the case of W7500, it discards the received repeating data in continuative data transmission. It causes data loss.

To avoid it, W7500 uses GPIO instead of I^2C . In this case, SCL has limited speed, 100KHz.

For the details, the Errata Sheet is supported.

<http://wizwiki.net/wiki/doku.php?id=products:w7500:documents>

21.2 Features

- Use APB interface
- Supports Slave and Master Mode
- Standard mode (100 KHz)
- Supports 7 bit Slave address mode
- Start/Stop/Repeated Start detection
- Start/Stop/Repeated Start/Acknowledge generation
- Control the Read/Write operation
- General Call enable or disable
- Slave busy detection
- Repeated START

22 Universal Asynchronous Receive Transmit(UART0,1)

22.1 Introduction

The UART0,1 supports synchronous one-way communication, half-duplex single wire communication, and multiprocessor communications(CTS/RTS).

22.2 Features

- Serial-to-parallel conversion on data received from a peripheral device
- Parallel-to-serial conversion on data transmitted to the peripheral device
- Data size of 5,6,7 and 8 bits
- One or two stop bits
- Even, odd, stick, or no-parity bit generation and detection
- Support of hardware flow control
- Programmable FIFO disabling for 1-byte depth.
- Programmable use of UART or IrDA SIR input/output
- False start bit detection

22.3 Functional description

UART0,1 bidirectional communication requires a minimum of two pins: RX, TX

The frame are comprised of:

- An Idle Line prior to transmission or reception
- A start bit
- A data word (8 or 9 bits) least significant bit first
- 1, 1.5, 2 Stop bits indicating that the frame is complete
- The USART interface uses a baud rate generator
- A status register (UART1_RISR)
- data registers (UART1DR)
- A baud rate register (UART1_IBRD,UART1_FBRD)

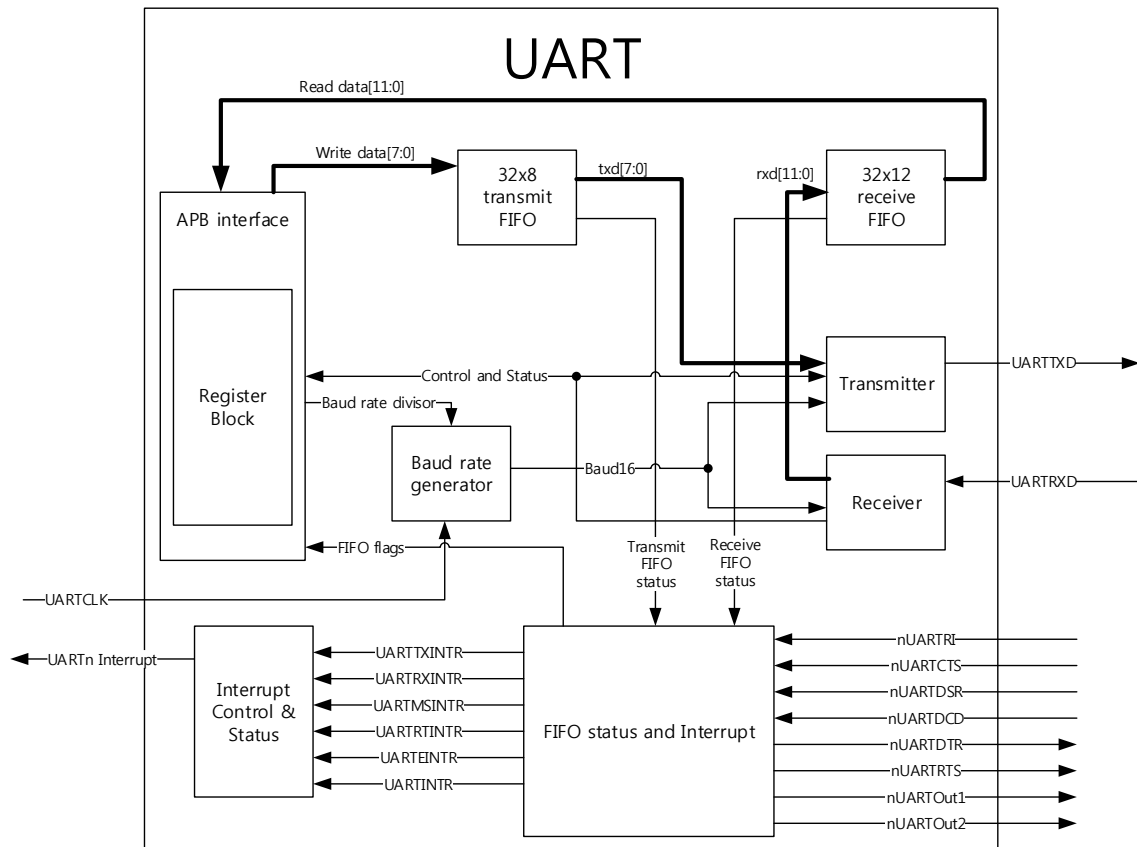


Figure 45 UART0,1 Block diagram

Figure 46 shows the UART0,1 character frame

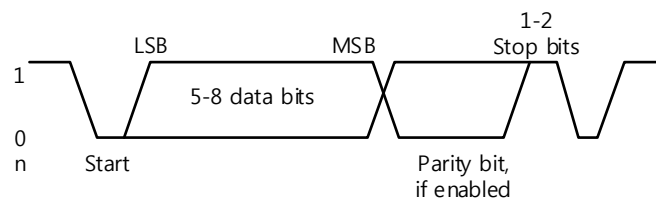


Figure 46 UART character frame

22.3.1 Baud rate calculation

UART0,1 can operate with or without using the Fractional Divider. The baud rate divisor is a 22-bit number consisting the UARTxIBRD(16-bit integer) and the UARTxFBRD(6-bit fractional). This is used by the baud rate generator to determine the bit period.

$$\text{Baud Rate Divisor} = \frac{\text{UARTCLK}}{(16 * \text{baud rate})} = \text{BRD}_I + \text{BRD}_F$$

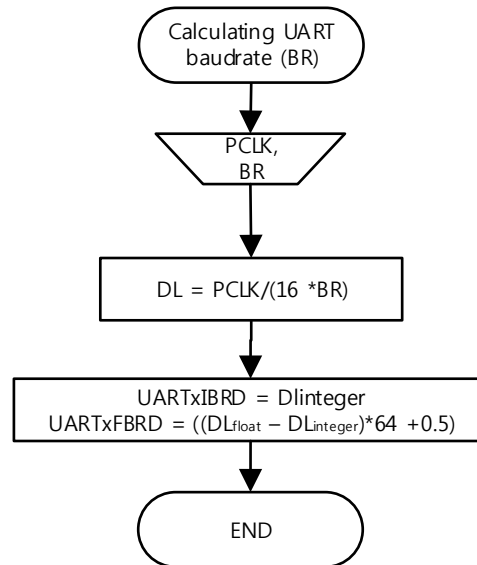


Figure 47 UART0,1 divider flow chart

Figure 48 shows how to set the UART0,1 Initial value.

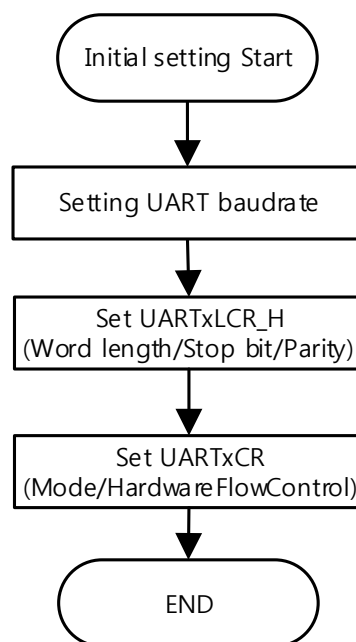


Figure 48 UART0,1 Initial setting flow chart

22.3.2 Data transmission

Data transmitted is stored in a 32-byte FIFOs. Transmit data is written into the transmit FIFO for transmission. If UART0,1 is enabled, it causes a data frame to start transmitting with parameters indicated in the UARTxLCR_H.

Data continues to transmit until there is no data left in the transmit FIFO. The BUSY bit of UARTxFR is '1' as soon as data is written to the transmit FIFO, which means the FIFO is not empty, and remains as '1' while data is being transmitted.

22.3.3 Data receive

Received data is stored in the 32-byte FIFOs. When a start bit has been received, it begins running and data is sampled on the eighth cycle of that counter in UART mode. A valid stop bit is confirmed if UARTRXD is '1'. When a full word is received, the data is stored in the receive FIFO. Error bit is stored in bit[10:8] of UARTxCR and overrun is stored in bit[11] of UARTxCR.

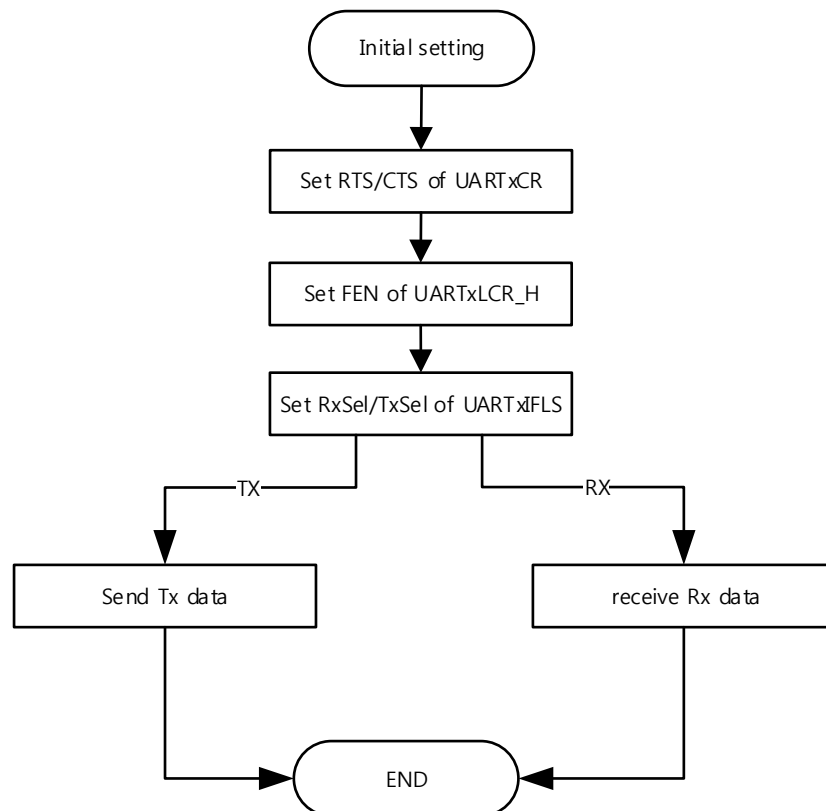


Figure 49 Transmit and Receive data flow chart

22.3.4 Hardware flow control

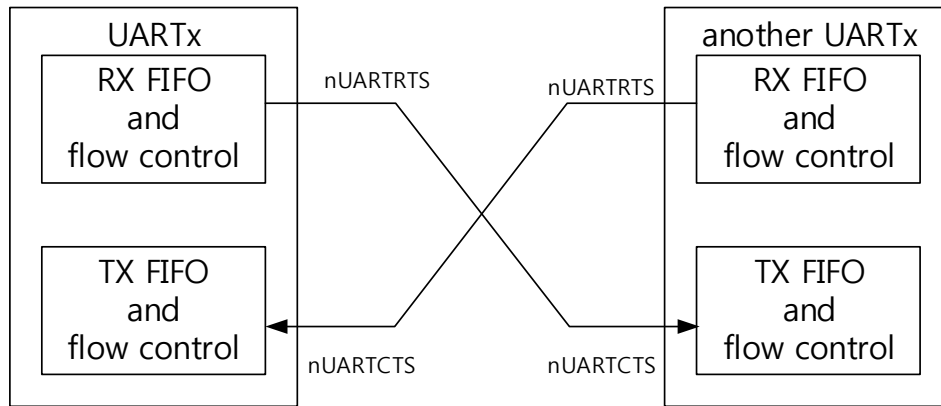


Figure 50 Hardware flow control description

The RTS flow control is enabled by setting the RTSen of UARTxCR. If RTS is enabled, the data flow is controlled as follows.

When the receiver FIFO level reaches the programmed trigger level, nUARTRTS(pin) is asserted(to a low value). nUARTRTS is reasserted(to a low level) once the receiver FIFO has reached the previous trigger level. The reasserted of nUARTRTS signals to the sending UART to continue transmitting data.

The CTS flow control is enabled, the transmitter can only transmit data when nUARTCTS is asserted. When nUARTCTR is re-asserted(to a low) the transmitter sends the next byte. To stop the transmitter from sending the following byte, nUARTCTS must be released before the middle of the last stop bit that is currently being sent.

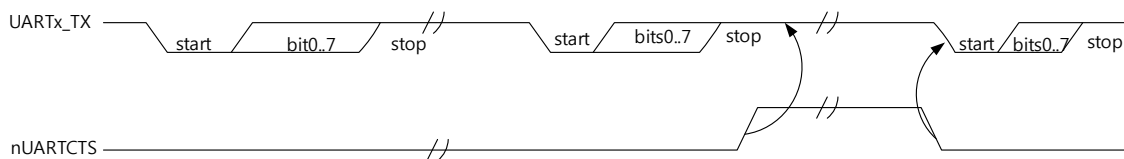


Figure 51 CTS Functional Timing

Figure 52 shows how software should use the RTS/CTR.

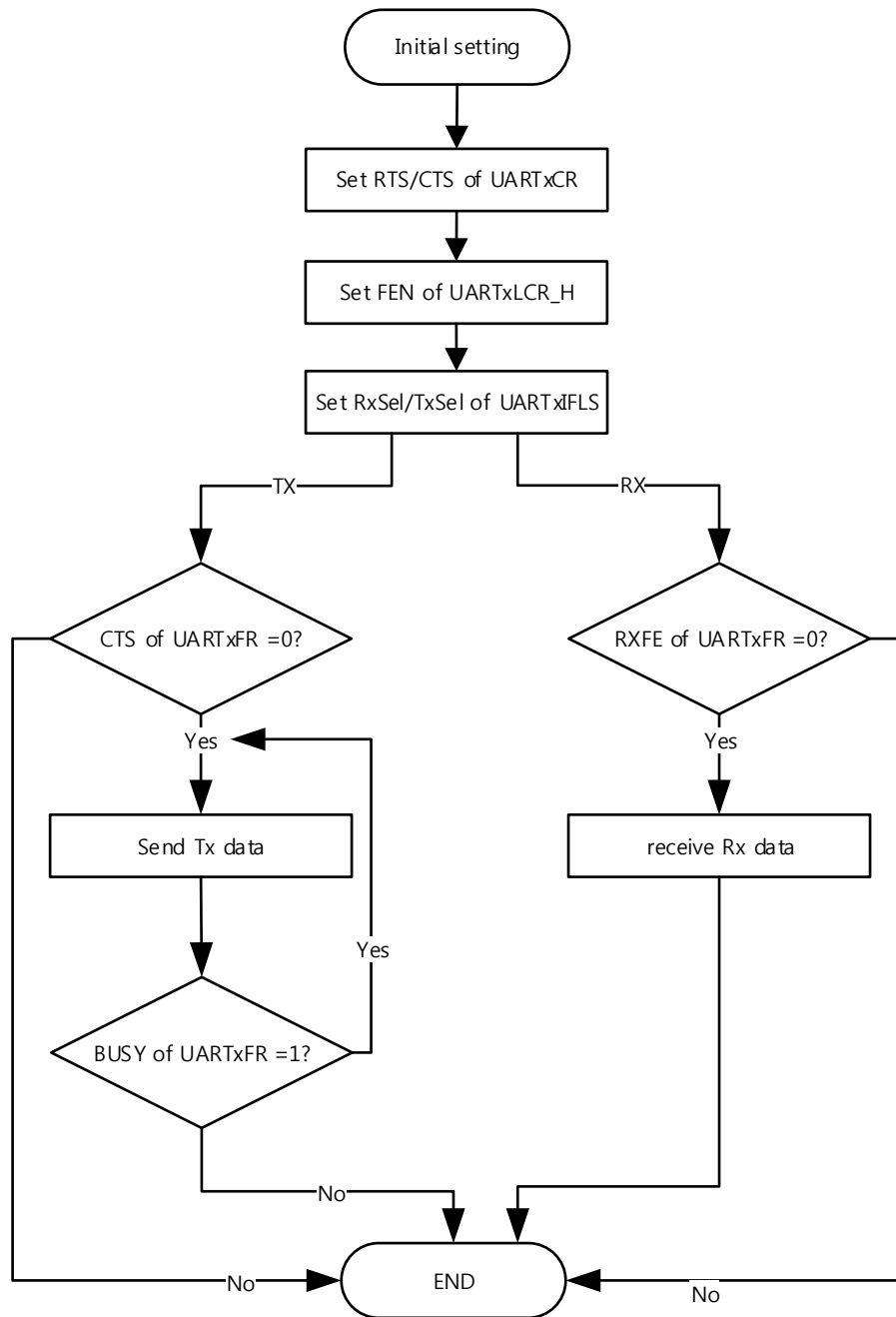


Figure 52 Algorithm for setting CTS/RTS flowchart

23 Universal Asynchronous Receive Transmit(UART2)

23.1 Introduction

The UART2 supports synchronous one-way communication, half-duplex single wire communication, and without CTS/RTS. UART2 is called the simple UART.

23.2 Feature

- Serial-to-parallel conversion on data received from a peripheral device
- Parallel-to-serial conversion on data transmitted to the peripheral device

23.3 Functional description

UART2 bidirectional communication requires a minimum of two pins: RX, TX

The frame are comprised of:

- An Idle Line prior to transmission or reception
- The UART interface uses a baud rate generator
- A status register (UART2_SR)
- data registers (UART2DR)
- A baud rate register (UART2_B, UART1_FBRD)

23.3.1 Baud rate calculation

UART2 can operate with or without using the Fractional Divider. The baud rate divisor is a 20-bit number consisting the UART2BDR.

This is used by the baud rate generator to determine the bit period.

- Baud Rate Divisor = $\frac{\text{UARTCLK}}{(\text{baud rate})}$

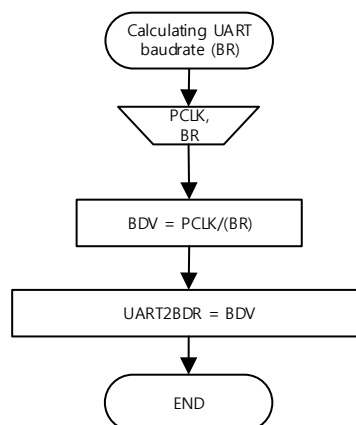


Figure 53. UART2 divider flow chart

Figure 54 show how to set the UART2 Initial value.

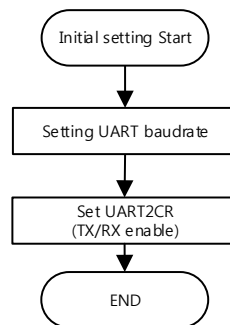


Figure 54. UART2 Initial setting flow chart

24 Synchronous Serial Port (SSP)

24.1 Introduction

The SSP block is an IP provided by ARM (PL022 “PrimeCell® Synchronous Serial Port”). Additional details about its functional blocks may be found in “ARM PrimeCell® Synchronous Serial Port (PL022) Technical Reference Manual”.

24.2 Features

- The SSP is a master or slave interface that enables synchronous serial communication with slave or master peripherals having one of the following:
 - A MOTOROLA SPI-compatible interface
 - A TEXAS INSTRUMENTS synchronous serial interface
 - A National Semiconductor MICROWIRE® interface.
- The SPI interface operates as a master or slave interface. It supports bit rates up to 20 MHz in both master and slave configurations. The SPI has the following features:
 - Parallel-to-serial conversion on data written to an internal 16-bit wide, 8-location deep transmit FIFO
 - Serial-to-parallel conversion on received data, buffering it in a 16-bit wide, 8-location deep receive FIFO
 - Programmable data frame size from 4 to 16 bits

- Programmable clock bit rate and prescaler. The input clock may be divided by a factor of 2 to 254 in steps of two to provide the serial output clock
- Programmable clock phase and polarity.

24.3 Functional description

Figure 55 shows the SSP block diagram.

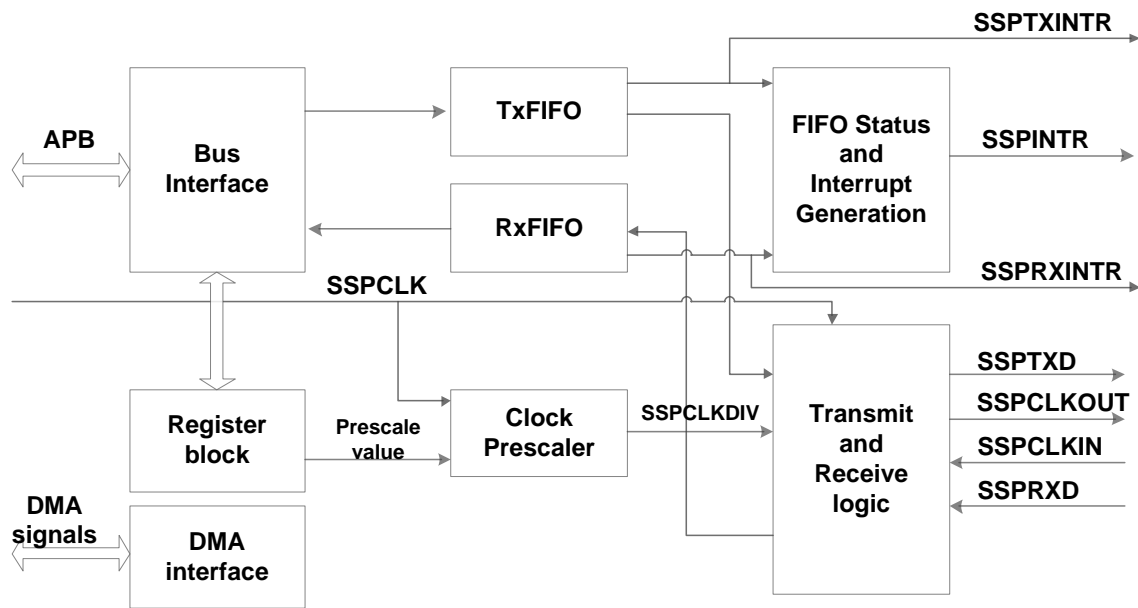


Figure 55. SSP block diagram

24.3.1 Clock prescaler

When configured as a master, an internal prescaler is used to provide the serial output clock. The prescaler may be programmed through the SSPCSR register to divide the SSPCLK by a factor of 2 to 254 in two steps. As the least significant bit of the SSPCSR register is not used, division by an odd number is impossible and this ensures a symmetrical (equal mark space ratio) clock is generated.

The output of this prescaler is further divided by a factor 1 to 256 through the programming of the SSPCR0 control register, to give a final master output clock.

24.3.2 Transmit FIFO

The common transmit FIFO is a 16-bit wide, 8-locations deep, First-In, First-Out (FIFO) memory buffer. CPU data written across the AMBA APB interface are stored in the buffer until it is read out by the transmit logic.

When configured as a master or a slave, parallel data is written into the transmit FIFO prior to serial conversion and is transmitted to the attached slave or master through the SSPTXD pin.

24.3.3 Receive FIFO

The common receive FIFO is a 16-bit wide, 8-locations deep, first-in, first-out memory buffer. Received data from the serial interface are stored in the buffer until it is read out by the CPU across the AMBA APB interface.

When configured as a master or slave, serial data received through the SSPRXD pin is registered prior to parallel loading into the attached slave or master receive FIFO.

24.3.4 Interrupt generation logic

The PrimeCell SSP generates four individual maskable, active-HIGH interrupts. A combined interrupt output is also generated as an OR function of the individual interrupt requests.

Users can use the single combined interrupt with a system interrupt controller that provides another level of masking on a per-peripheral basis. This enables use of modular device drivers that always know where to find the interrupt source control register bits.

Users can also use the individual interrupt requests with a system interrupt controller that provides masking for the outputs of each peripheral. In this way, a global interrupt controller service routine can read the entire set of sources from one wide register in the system interrupt controller. This is attractive when the time to read from the peripheral registers is significant compared to the CPU clock speed in a real-time system.

The peripheral supports both methods above.

The transmit and receive dynamic data-flow interrupts, SSPTXINTR and SSPRXINTR, are separated from the status interrupts so that data can be read or written in response to the FIFO trigger levels.

24.3.5 DMA interface

The PrimeCell SSP provides an interface to connect to the DMA controller. The PrimeCell SSP DMA control register, SSPDMACR controls the DMA operation of the PrimeCell SSP.

Receive - The DMA interface includes the following signals for receive:

- SSPRXDMASREQ
 - Single-character DMA transfer request asserted by the SSP. This signal is asserted when the receive FIFO contains at least one character.
- SSPRXDMABREQ
 - Burst DMA transfer request, asserted by the SSP. This signal is asserted when the receive FIFO contains four or more characters.
- SSPRXDMACLR
 - DMA request clear asserted by the DMA controller to clear the receive request signals. If DMA burst transfer is requested, the clear signal is asserted during the transfer of the last data in the burst.

Transmit - The DMA interface includes the following signals for transmit:

- SSPTXDMASREQ
 - Single-character DMA transfer request asserted by the SSP. This signal is asserted when there is at least one empty location in the transmit FIFO.
- SSPTXDMABREQ
 - Burst DMA transfer request asserted by the SSP. This signal is asserted when the transmit FIFO contains four characters or fewer.
- SSPTXDMACLR
 - DMA request clear asserted by the DMA controller to clear the transmit request signals. If a DMA burst transfer is requested, the clear signal is asserted during the transfer of the last data in the burst.

The burst transfer and single transfer request signals are not mutually exclusive. They can both be asserted at the same time. For example, when there is more data than the watermark level of four in the receive FIFO, the burst transfer request and the single transfer request are asserted.

When the amount of data left in the receive FIFO is less than the watermark level, the single request only is asserted. This is useful for situations when the number of characters left to be received in the stream is less than a burst.

For example, if 19 characters must be received, the DMA controller then transfers four bursts of four characters and three single transfers to complete the stream.

The PrimeCell SSP does not assert the burst request for the remaining three characters.

Each request signal remains asserted until the relevant DMA clear signal is asserted. After the request clear signal is de-asserted, a request signal can become active again depending on the conditions that previous sections describe. All request signals are de-asserted if the PrimeCell SSP is disabled or the DMA enable signal is cleared.

Table 10 shows the trigger points for DMABREQ of both the transmit and receive FIFOs.

Table 10 DMA trigger points for the transmit and receive FIFOs.

| Burst length | | |
|-----------------|-------------------------------------|-------------------------------------|
| Watermark level | Transmit, number of empty locations | Receive, number of filled locations |
| 1/2 | 4 | 4 |

Figure 56 shows the timing diagram for both a single transfer request and a burst transfer request with the appropriate DMA clear signal. The signals are all synchronous to PCLK.

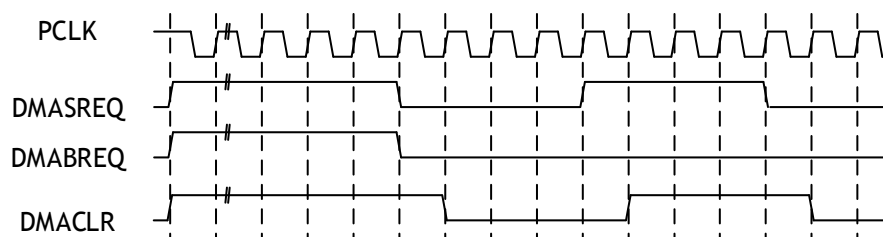


Figure 56. DMA transfer waveforms

24.3.6 Interface reset

The PrimeCell SSP is reset by the global reset signal PRESETn and a block-specific reset signal nSSPRST. An external reset controller must use PRESETn to assert nSSPRST asynchronously and negate it synchronously to SSPCLK. PRESETn must be asserted LOW for a period long enough to reset the slowest block in the on-chip system, and then taken HIGH again. The PrimeCell SSP requires PRESETn to be asserted LOW for at least one period of PCLK.

24.3.7 Configuring the SSP

The Following reset, the PrimeCell SSP logic is disabled and must be configured when in this state.

It is necessary to program control registers SSPCR0 and SSPCR1 to configure the peripheral as a master or slave operating under one of the following protocols:

- Motorola SPI

- Texas Instruments SSI
- National Semiconductor.

The bit rate derived from the external SSPCLK requires the programming of the clock prescale register SSPCPSR.

24.3.8 Enable PrimeCell SSP operation

You can either prime the transmit FIFO, by writing up to eight 16-bit values when the PrimeCell SSP is disabled, or permit the transmit FIFO service request to interrupt the CPU. Once enabled, transmission or reception of data begins on the transmit, SSPTXD, and receive, SSPRXD, pins.

24.3.9 Clock ratios

There is a constraint on the ratio of the frequencies of PCLK to SSPCLK. The frequency of SSPCLK must be less or equal to that of PCLK. This ensures that control signals from the SSPCLK domain to the PCLK domain are guaranteed to get synchronized before one frame duration:

$$F_{SSPCLK} \leq F_{PCLK}.$$

In the slave mode of operation, the SSPCLKIN signal from the external master is double-synchronized and then delayed to detect an edge. It takes three SSPCLKs to detect an edge on SSPCLKIN. SSPTXD has less setup time to the falling edge of SSPCLKIN on which the master is sampling the line.

The setup and hold times on SSPRXD, with reference to SSPCLKIN, must be more conservative to ensure that it is at the right value when the actual sampling occurs within the SSPMS. To ensure correct device operation, SSPCLK must be at least 12 times faster than the maximum expected frequency of SSPCLKIN.

The frequency selected for SSPCLK must accommodate the desired range of bit clock rates. The ratio of minimum SSPCLK frequency to SSPCLKOUT maximum frequency in the case of the slave mode is 12, and for the master mode, it is two.

To generate a maximum bit rate of 1.8432Mbps in the master mode, the frequency of SSPCLK must be at least 3.6864MHz. With an SSPCLK frequency of 3.6864MHz, the SSPCPSR register must be programmed with a value of 2, and the SCR[7:0] field in the SSPCR0 register must be programmed with a value of 0.

To work with a maximum bit rate of 1.8432Mbps in the slave mode, the frequency of SSPCLK must be at least 22.12MHz. With an SSPCLK frequency of 22.12MHz, the SSPCPSR register can be programmed with a value of 12 and the SCR[7:0] field in the SSPCR0 register can be

programmed with a value of 0. Similarly, the ratio of SSPCLK maximum frequency to SSPCLKOUT minimum frequency is 254×256 .

The minimum frequency of SSPCLK is calculated by the following equations, both of which must be satisfied:

$$F_{SSPCLK}(\min) \Rightarrow 2 \times F_{SSPCLKOUT}(\max), \text{ for master mode}$$

$$F_{SSPCLK}(\min) \Rightarrow 12 \times F_{SSPCLKIN}(\max), \text{ for slave mode.}$$

The maximum frequency of SSPCLK is calculated by the following equations, both of which must be satisfied:

$$F_{SSPCLK}(\max) \leq 254 \times 256 \times F_{SSPCLKOUT}(\min), \text{ for master mode}$$

$$F_{SSPCLK}(\max) \leq 254 \times 256 \times F_{SSPCLKIN}(\min), \text{ for slave mode.}$$

24.3.10 Programming the SSPCR0 Control Register

The SSPCR0 register is used to:

- program the serial clock rate
- select one of the three protocols
- select the data word size, where applicable.

The Serial Clock Rate (SCR) value in conjunction with the SSPCPSR clock prescale divisor value, CPSPVSR, is used to derive the PrimeCell SSP transmit and receive bit rate from the external SSPCLK.

The frame format is programmed through the FRF bits and the data word size through the DSS bits.

Bit phase and polarity applicable to Motorola SPI format only are programmed through the SPH and SPO bits.

24.3.11 Programming the SSPCR1 Control Register

The SSPCR1 register is used to:

- select master or slave mode
- enable a loop back test feature
- enable the PrimeCell SSP peripheral.

To configure the PrimeCell SSP as a master, clear the SSPCR1 register master or slave selection bit, MS, to 0. This is the default value on reset.

Setting the SSPCR1 register MS bit to 1 configures the PrimeCell SSP as a slave. When configured as a slave, enabling or disabling of the PrimeCell SSP SSPTXD signal is provided through the SSPCR1 slave mode SSPTXD output disable bit, SOD. You can use this in some multi-slave environments where masters might parallel broadcast.

Set the Synchronous Serial Port Enable (SSE) bit to 1 to enable the operation of the PrimeCell SSP.

Bit rate generation

The serial bit rate is derived by dividing down the input clock SSPCLK. The clock is first divided by an even prescale value CPSDVSR in the range of 2-254, and is programmed in SSPCPSR. The clock is divided again by a value in the range of 1-256, that is $1 + SCR$, where SCR is the value programmed in SSPCR0.

The following equation defines the frequency of the output signal bit clock, SSPCLKOUT:

$$F_{SSPCLKOUT} = \frac{F_{SSPCLK}}{CPSDVSR \times (1 + SCR)}$$

For example, if SSPCLK is 20MHz, and CPSDVSR = 2, then SSPCLKOUT has a frequency range of 39.06kHz-10MHz.

24.3.12 Frame format

Each data frame is between 4-16 bits long depending on the size of data programmed and is transmitted starting with the MSB. Users can select the following basic frame types:

- Texas Instruments synchronous serial
- Motorola SPI
- National Semiconductor Microwire.

For all formats, the serial clock SSPCLKOUT is held inactive while the PrimeCell SSP is idle and transitions at the programmed frequency only during active transmission or reception of data. The idle state of SSPCLKOUT is utilized to provide a receive timeout indication that occurs when the receive FIFO still contains data after a timeout period.

For Motorola SPI and National Semiconductor Microwire frame formats, the serial frame SSPFSSOUT pin is active-LOW and is asserted and pulled-down during the entire transmission of the frame.

For Texas Instruments synchronous serial frame format, the SSPFSSOUT pin is pulsed for one serial clock period starting at its rising edge prior to the transmission of each frame. For this frame format, both the PrimeCell SSP and the off-chip slave device drive their output data on the rising edge of SSPCLKOUT and latch data from the other device on the falling edge.

Unlike the full-duplex transmission of the other two frame formats, the National Semiconductor Microwire format uses a special master-slave messaging technique which operates at half-duplex. In this mode, an 8-bit control message is transmitted to the off-chip slave when a frame begins. During this transmit, the SSS receives no incoming data. After the message has been sent, the off-chip slave decodes it and responds with the requested data after waiting one serial clock after the last bit of the 8-bit control message has been sent. The returned data can be 4-16 bits in length making the total frame length in the range of 13-25 bits.

24.3.13 Texas Instruments synchronous serial frame format

Figure 57 shows the Texas Instruments synchronous serial frame format for a single transmitted frame.

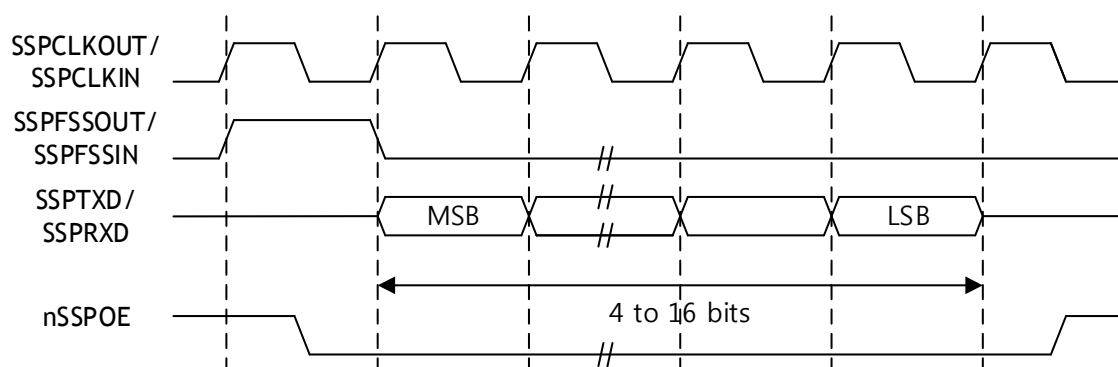


Figure 57. Texas Instruments synchronous serial frame format, single transfer

In this mode, SSPCLKOUT and SSPFSSOUT are forced LOW and the transmit data line SSPTXD is tristated whenever the PrimeCell SSP is idle. When the bottom entry of the transmit FIFO contains data, SSPFSSOUT is pulsed HIGH for one SSPCLKOUT period. The value to be transmitted is also transferred from the transmit FIFO to the serial shift register of the transmit logic. On the next rising edge of SSPCLKOUT, the MSB of the 4-bit to 16-bit data frame is shifted out on the SSPTXD pin. In a similar way, the MSB of the received data is shifted onto the SSPRXD pin by the off-chip serial slave device.

Both the PrimeCell SSP and the off-chip serial slave device then clock each data bit into their serial shifter on the falling edge of each SSPCLKOUT. The received data is transferred from the serial shifter to the receive FIFO on the first rising edge of PCLK after the LSB has been latched.

Figure 58 shows the Texas Instruments synchronous serial frame format when back-to-back frames are transmitted.

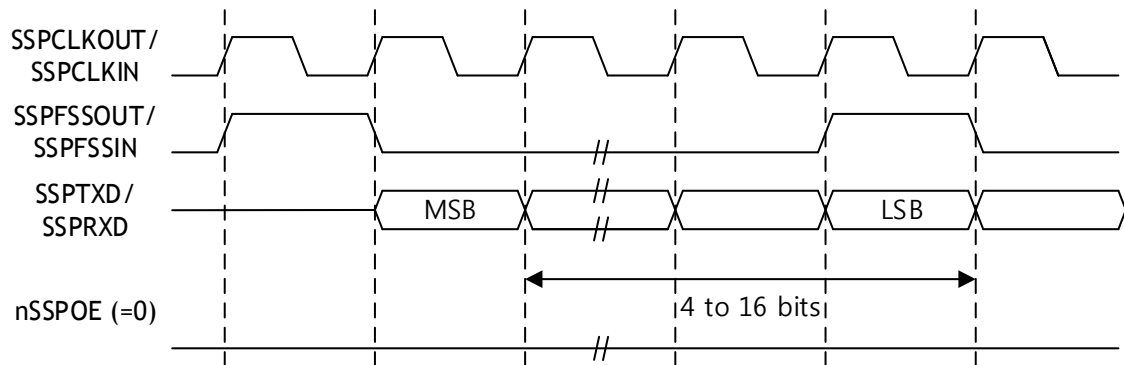


Figure 58. Texas Instruments synchronous serial frame format, continuous transfers

24.3.14 Motorola SPI frame format

The Motorola SPI interface is a four-wire interface where the SSPFSSOUT signal behaves as a slave select. The main feature of the Motorola SPI format is that you can program the inactive state and phase of the SSPCLKOUT signal using the SPO and SPH bits of the SSPSCR0 control register.

SPO, clock polarity

When the SPO clock polarity control bit is LOW, it produces a steady state LOW value on the SSPCLKOUT pin. If the SPO clock polarity control bit is HIGH, a steady state HIGH value is placed on the SSPCLKOUT pin when data is not being transferred.

SPH, clock phase

The SPH control bit selects the clock edge that captures data and enables it to change state. It has the most impact on the first bit transmitted by either permitting or not permitting a clock transition before the first data capture edge.

When the SPH phase control bit is LOW, data is captured on the first clock edge transition.

When the SPH clock phase control bit is HIGH, data is captured on the second clock edge transition.

Figure 59 and Figure 60 show single and continuous transmission signal sequences for Motorola SPI format with SPO=0, SPH=0.

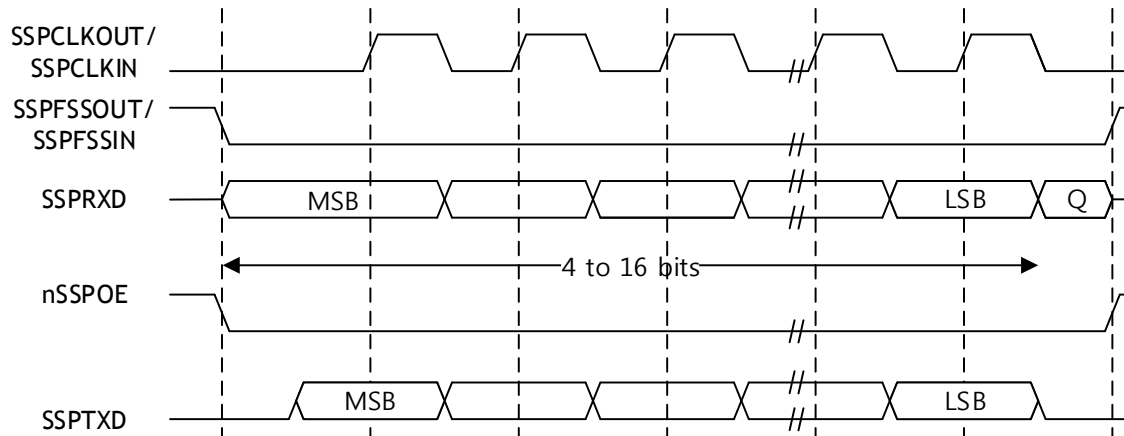


Figure 59. Motorola SPI frame format, single transfer, with SPO=0 and SPH=0

Figure 60 shows a continuous transmission signal sequence for Motorola SPI frame format with SPO=0, SPH=0.

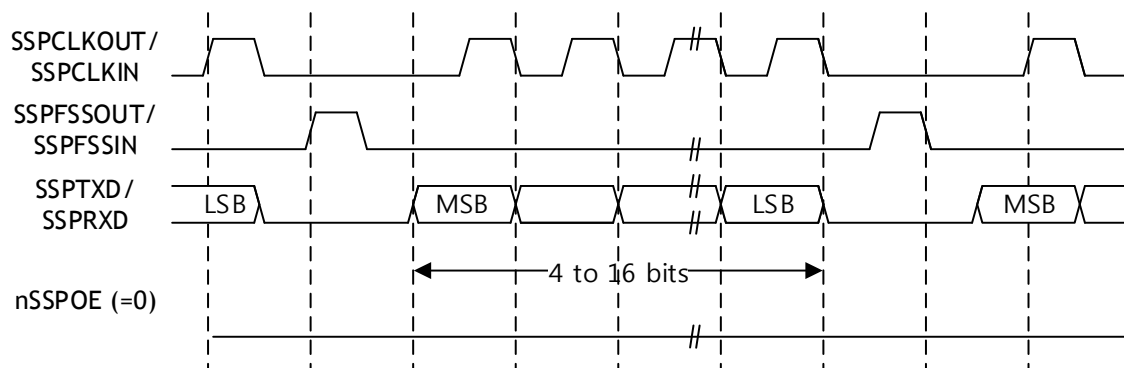


Figure 60. Motorola SPI frame format, continuous transfers, with SPO=0 and SPH=0

In this configuration, during idle periods:

- the SSPCLKOUT signal is forced LOW
- the SSPFSSOUT signal is forced HIGH
- the transmit data line SSPTXD is arbitrarily forced LOW
- the nSSPOE pad enable signal is forced HIGH, making the transmit pad high impedance
- when the PrimeCell SSP is configured as a master, the nSSPCTLOE line is driven LOW, enabling the SSPCLKOUT pad, active-LOW enable
- when the PrimeCell SSP is configured as a slave, the nSSPCTLOE line is driven HIGH, disabling the SSPCLKOUT pad, active-LOW enable.

If the PrimeCell SSP is enabled and there is valid data within the transmit FIFO, the start of transmission is signified by the SSPFSSOUT master signal being driven LOW. This causes the slave data to be enabled onto the SSPTXD input line of the master. The nSSPOE line is driven LOW, enabling the master SSPTXD output pad.

One half SSPCLKOUT period later, valid master data is transferred to the SSPTXD pin. Now that both the master and slave data have been set, the SSPCLKOUT master clock pin goes HIGH after one additional half SSPCLKOUT period.

The data is now captured on the rising and propagated on the falling edges of the SSPCLKOUT signal.

In the case of a single word transmission after all bits of the data word have been transferred, the SSPFSSOUT line is returned to its idle HIGH state one SSPCLKOUT period after the last bit has been captured.

However, in the case of continuous back-to-back transmissions, the SSPFSSOUT signal must be pulsed HIGH between each data word transfer. This is because the slave select pin freezes the data in its serial peripheral register and does not permit it to be altered if the SPH bit is logic zero. Therefore, the master device must raise the SSPFSSIN pin of the slave device between each data transfer to enable the serial peripheral data write. On completion of the continuous transfer, the SSPFSSOUT pin is returned to its idle state one SSPCLKOUT period after the last bit has been captured.

Figure 61 shows the transfer signal sequence for Motorola SPI format with SPO=0, SPH=1, and it covers both single and continuous transfers.

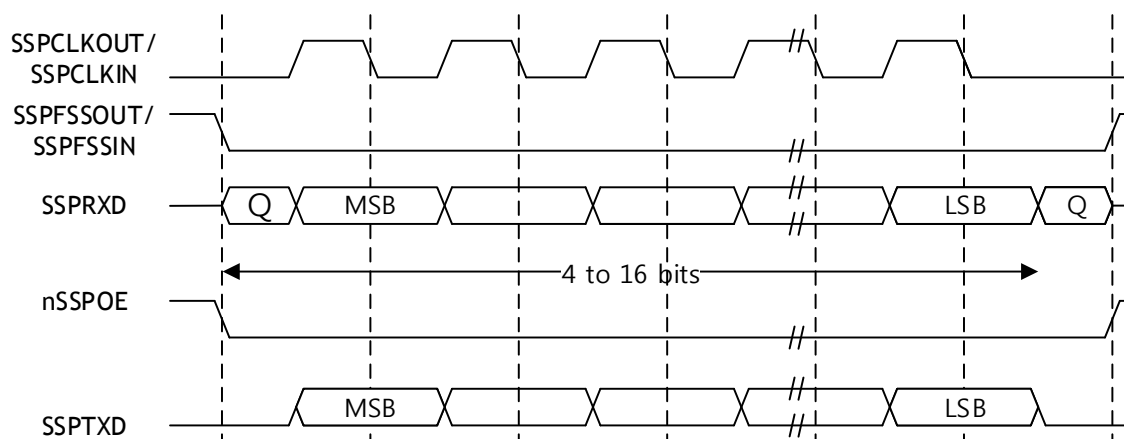


Figure 61. Motorola SPI frame format, single and continuous transfers, with SPO=0 and SPH=1

In this configuration, during idle periods:

- the SSPCLKOUT signal is forced LOW
- The SSPFSSOUT signal is forced HIGH
- the transmit data line SSPTXD is arbitrarily forced LOW

- the nSSPOE pad enable signal is forced HIGH, making the transmit pad high impedance
- when the PrimeCell SSP is configured as a master, the nSSPCTLOE line is driven LOW, enabling the SSPCLKOUT pad, active-LOW enable
- when the PrimeCell SSP is configured as a slave, the nSSPCTLOE line is driven HIGH, disabling the SSPCLKOUT pad, active-LOW enable.

If the PrimeCell SSP is enabled and there is valid data within the transmit FIFO, the start of transmission is signified by the SSPFSSOUT master signal being driven LOW. The nSSPOE line is driven LOW, enabling the master SSPTXD output pad. After an additional one half SSPCLKOUT period, both master and slave valid data is enabled onto their respective transmission lines. At the same time, the SSPCLKOUT is enabled with a rising edge transition.

Data is then captured on the falling edges and propagated on the rising edges of the SSPCLKOUT signal.

In the case of a single word transfer after all bits have been transferred, the SSPFSSOUT line is returned to its idle HIGH state one SSPCLKOUT period after the last bit has been captured.

For continuous back-to-back transfers, the SSPFSSOUT pin is held LOW between successive data words and termination is the same as that of the single word transfer.

Figure 62 shows a single transmission signal sequence for Motorola SPI format with SPO=1, SPH=0.

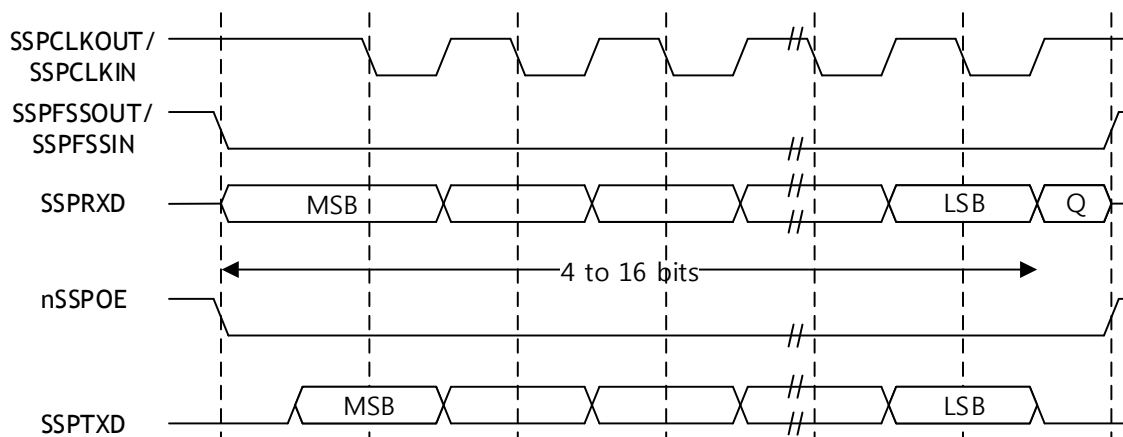


Figure 62. Motorola SPI frame format, single transfer, with SPO=1 and SPH=0

Figure 63 shows a continuous transmission signal sequence for Motorola SPI format with SPO=1, SPH=0. In Figure 9, Q is an undefined signal.

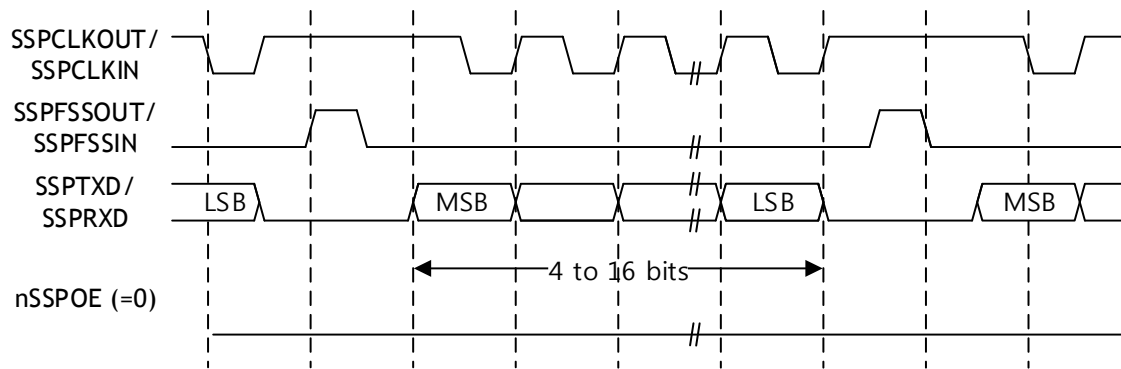


Figure 63. Motorola SPI frame format, continuous transfers, with SPO=1 and SPH=0

In this configuration, during idle periods:

- the SSPCLKOUT signal is forced HIGH
- the SSPFSSOUT signal is forced HIGH
- the transmit data line SSPTXD is arbitrarily forced LOW
- the nSSPOE pad enable signal is forced HIGH, making the transmit pad high impedance
- when the PrimeCell SSP is configured as a master, the nSSPCTLOE line is driven LOW, enabling the SSPCLKOUT pad, active-LOW enable
- when the PrimeCell SSP is configured as a slave, the nSSPCTLOE line is driven HIGH, disabling the SSPCLKOUT pad, active-LOW enable.

If the PrimeCell SSP is enabled and there is valid data within the transmit FIFO, the start of transmission is signified by the SSPFSSOUT master signal being driven LOW, and this causes slave data to be immediately transferred onto the SSPRXD line of the master. The nSSPOE line is driven LOW, enabling the master SSPTXD output pad.

One half period later, valid master data is transferred to the SSPTXD line. Now that both the master and slave data have been set, the SSPCLKOUT master clock pin becomes LOW after one additional half SSPCLKOUT period. This means that data is captured on the falling edges and be propagated on the rising edges of the SSPCLKOUT signal.

In the case of a single word transmission after all bits of the data word are transferred, the SSPFSSOUT line is returned to its idle HIGH state one SSPCLKOUT period after the last bit has been captured.

However, in the case of continuous back-to-back transmissions, the SSPFSSOUT signal must be pulsed HIGH between each data word transfer. This is because the slave select pin freezes the data in its serial peripheral register and does not permit it to be altered if the SPH bit is logic zero. Therefore, the master device must raise the SSPFSSIN pin of the slave device between

each data transfer to enable the serial peripheral data write. On completion of the continuous transfer, the SSPFSSOUT pin is returned to its idle state one SSPCLKOUT period after the last bit has been captured.

Figure 64 shows the transfer signal sequence for Motorola SPI format with SPO=1, SPH=1, and it covers both single and continuous transfers. In Figure 10, Q is an undefined signal.

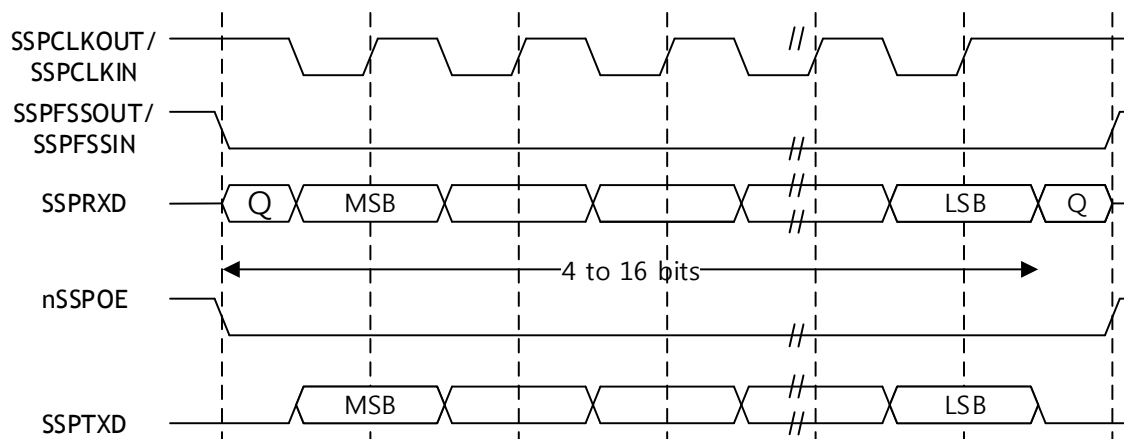


Figure 64. Motorola SPI frame format, single and continuous transfers, with SPO=1 and SPH=1

In this configuration, during idle periods:

- the SSPCLKOUT signal is forced HIGH
- the SSPFSSOUT signal is forced HIGH
- the transmit data line SSPTXD is arbitrarily forced LOW
- the nSSPOE pad enable signal is forced HIGH, making the transmit pad high impedance
- when the PrimeCell SSP is configured as a master, the nSSPCTLOE line is driven LOW, enabling the SSPCLKOUT pad, active-LOW enable
- when the PrimeCell SSP is configured as a slave, the nSSPCTLOE line is driven HIGH, disabling the SSPCLKOUT pad, active-LOW enable.

If the PrimeCell SSP is enabled and there is valid data within the transmit FIFO, the start of transmission is signified by the SSPFSSOUT master signal being driven LOW. The nSSPOE line is driven LOW, enabling the master SSPTXD output pad. After an additional one half SSPCLKOUT period, both master and slave data are enabled onto their respective transmission lines. At the same time, the SSPCLKOUT is enabled with a falling edge transition. Data is then captured on the rising edges and propagated on the falling edges of the SSPCLKOUT signal.

After all bits have been transferred in the case of a single word transmission, the SSPFSSOUT line is returned to its idle HIGH state one SSPCLKOUT period after the last bit has been captured.

For continuous back-to-back transmissions, the SSPFSSOUT pin remains in its active-LOW state, until the final bit of the last word has been captured, and then returns to its idle state as the previous section describes.

For continuous back-to-back transfers, the SSPFSSOUT pin is held LOW between successive data words and termination is the same as that of the single word transfer.

24.3.15 National Semiconductor Microwire frame format

Figure 65 shows the National Semiconductor Microwire frame format for a single frame.

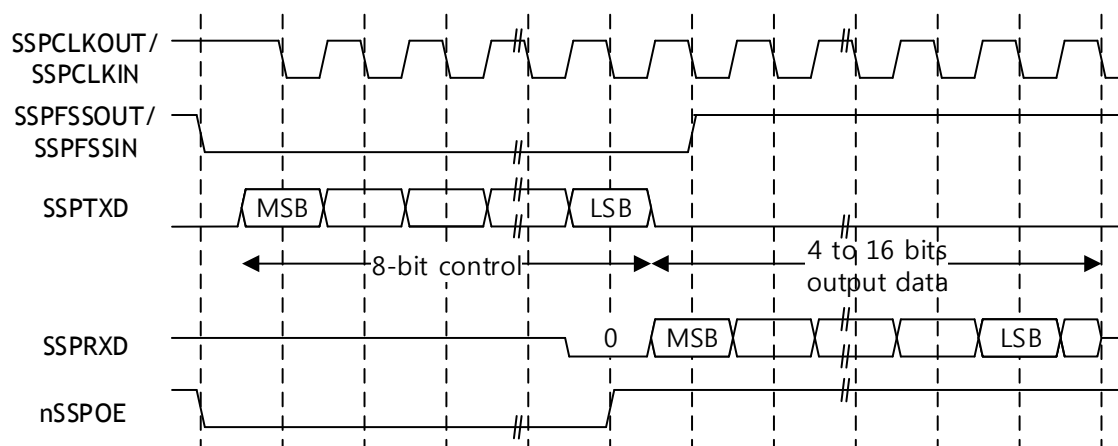


Figure 65. National Semiconductor Microwire frame format, single transfer

Microwire format is very similar to the SPI format except that transmission is half-duplex instead of full-duplex using a master-slave message passing technique. Each serial transmission begins with an 8-bit control word that is transmitted from the PrimeCell SSP to the off-chip slave device. During this transmission, the PrimeCell SSP receives no incoming data. After the message has been sent, the off-chip slave decodes it and responds with the required data after waiting one serial clock after the last bit of the 8-bit control message has been sent. The returned data is 4 to 16 bits in length, making the total frame length in the range of 13-25 bits.

In this configuration, during idle periods:

- SSPCLKOUT is forced LOW

- SSPFSSOUT is forced HIGH
- the transmit data line, SSPTXD, is arbitrarily forced LOW
- the nSSPOE pad enable signal is forced HIGH, making the transmit pad high impedance.

A transmission is triggered by writing a control byte to the transmit FIFO. The falling edge of SSPFSSOUT causes the value contained in the bottom entry of the transmit FIFO to be transferred to the serial shift register of the transmit logic and the MSB of the 8-bit control frame to be shifted out onto the SSPTXD pin. SSPFSSOUT remains LOW for the duration of the frame transmission. The SSPRXD pin remains tristated during this transmission.

The off-chip serial slave device latches each control bit into its serial shifter on the rising edge of each SSPCLKOUT. After the last bit is latched by the slave device, the control byte is decoded during a one clock wait-state and the slave responds by transmitting data back to the PrimeCell SSP. Each bit is driven onto the SSPRXD line on the falling edge of SSPCLKOUT. The PrimeCell SSP in turn latches each bit on the rising edge of SSPCLKOUT. At the end of the frame for single transfers, the SSPFSSOUT signal is pulled HIGH one clock period after the last bit has been latched in the receive serial shifter, which causes the data to be transferred to the receive FIFO.

The off-chip slave device can tristate the receive line either on the falling edge of SSPCLKOUT after the LSB has been latched by the receive shifter or when the SSPFSSOUT pin goes HIGH.

For continuous transfers, data transmission begins and ends in the same manner as a single transfer. However, the SSPFSSOUT line is continuously asserted, held LOW, and transmission of data occurs back-to-back. The control byte of the next frame follows directly after the LSB of the received data from the current frame. Each of the received values is transferred from the receive shifter on the falling edge SSPCLKOUT, after the LSB of the frame has been latched into the PrimeCell SSP.

Figure 66 shows the National Semiconductor Microwire frame format when back-to-back frames are transmitted.

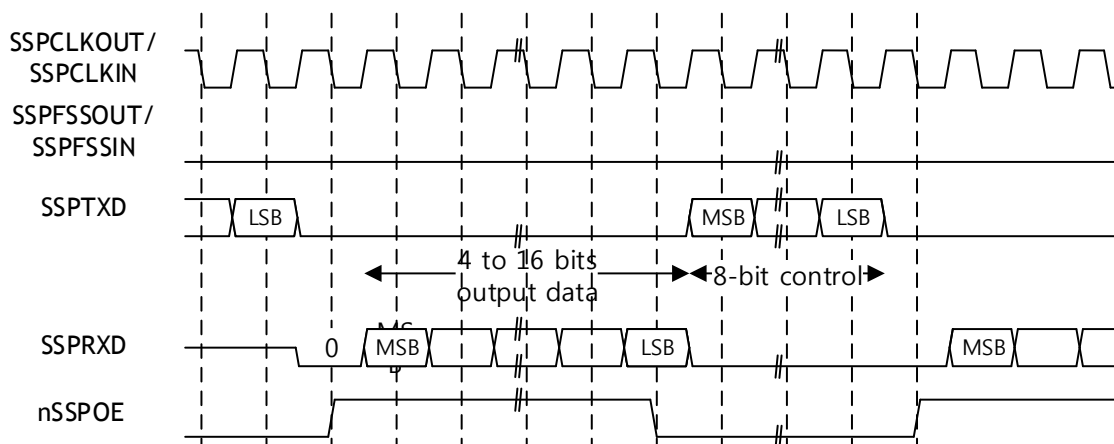


Figure 66. National Semiconductor Microwire frame format, continuous transfers

24.3.16 Master and Slave configurations

Figure 67 shows how a PrimeCell SSP (PL022) configured as master, interfaces to a Motorola SPI slave. The SPI Slave Select (SS) signal is permanently tied LOW and configures it as a slave. Similar to the above operation, the master can broadcast to the slave through the master PrimeCell SSP SSPTXD line. In response, the slave drives its SPI MISO port onto the SSPRXD line of the master.

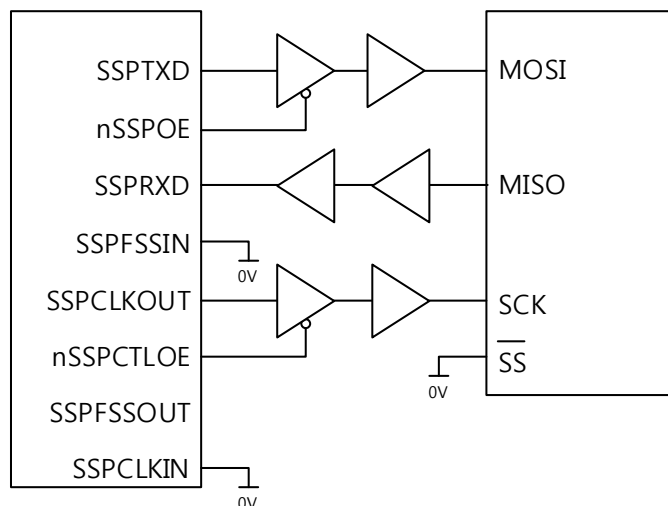


Figure 67. PrimeCell SSP master coupled to an SPI slave

Figure 68 shows a Motorola SPI configured as a master and interfaced to an instance of a PrimeCell SSP (PL022) configured as a slave. In this case, the slave Select Signal (SS) is permanently tied HIGH to configure it as a master. The master can broadcast to the slave through the master SPI MOSI line. In response, the slave drives its nSSPOE signal LOW.

This enables its SSPTXD data onto the MISO line of the master.

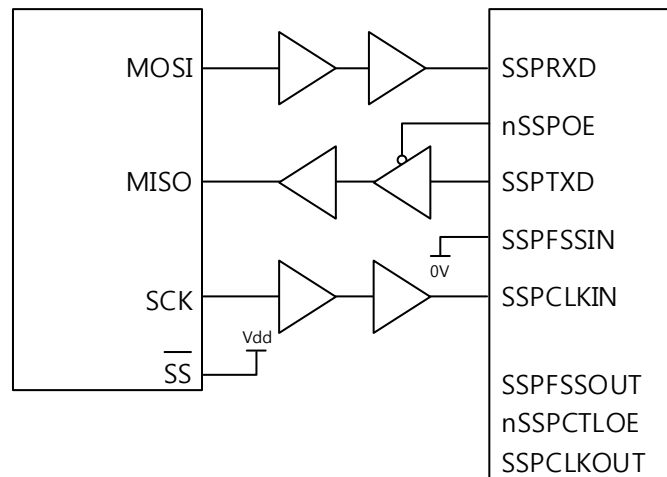


Figure 68. SPI master coupled to a PrimeCell SSP slave

24.3.17 SSP Flow chart

Figure 69 shows how to setting TI or Microwire mode.

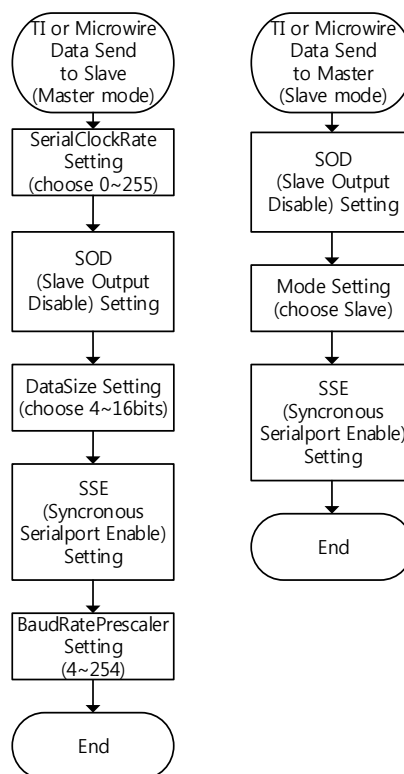


Figure 69. how to setting TI or Microwire mode flow chart

Figure 70 shows how to set SPI mode.

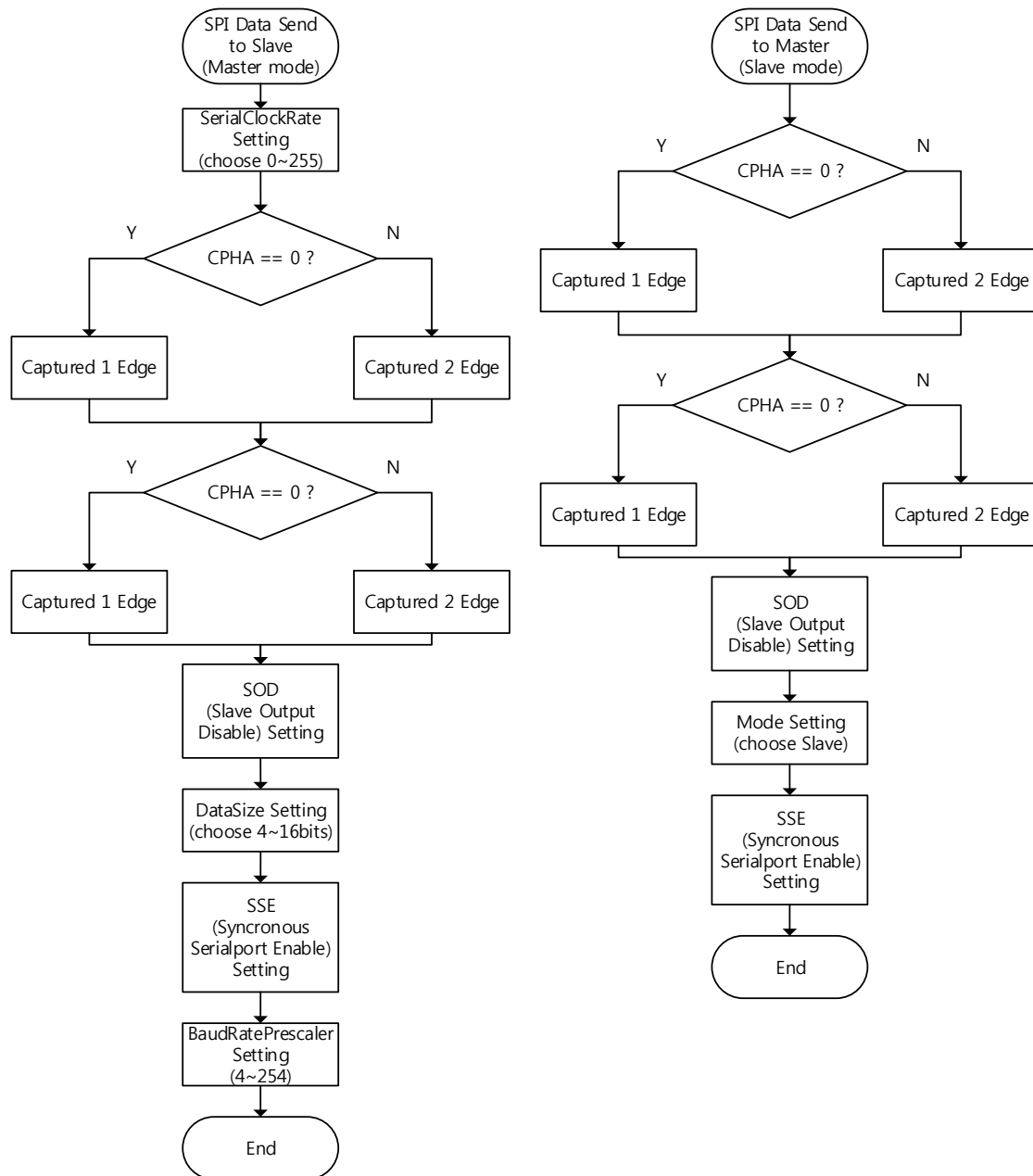


Figure 70. how to setting SPI mode flow chart

25 Electrical Characteristics

25.1 Absolute maximum ratings

These are stress ratings only and functional operation of the device at these conditions is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.

25.1.1 Voltage Characteristics

Table 11 shows the voltage characteristics of W7500.

Table 11 Voltage characteristics

| Symbol | Ratings | Min | Max | Unit |
|-----------------|--|----------------|---------|------|
| V_{DD-VSS} | External main supply voltage (VDD) | -0.3 | 5.8 | V |
| V_{IN} | Input voltage on IO pins | $V_{SS} - 0.3$ | 5.8 | V |
| S_{VDDH} | I/O Power on slope | 5V/Sec | 1V/uSec | - |
| ΔV_{DD} | Variations between difference VDD power pins | | 50 | mV |
| ΔV_{SS} | Variations between different ground pins | | 50 | mV |

25.1.2 Current Characteristics

Table 12 shows the current characteristics of W7500.

Table 12 Current characteristics

| Symbol | Ratings | Max | Unit |
|----------------|--|----------|------|
| I_{VDD_SUM} | Total current into sum of all VDD power lines (source) | 100 | mA |
| I_{VDD} | Maximum current into each ADD power pin (source) | 90 | mA |
| I_{IO_PAD} | Total output current sunk by sum of all IOs and control pins | 75 | mA |
| I_{INJ_PAD} | Single pin input injected current | ± 10 | mA |
| I_{INJ_SUM} | Sum of all input injected current | ± 50 | mA |

25.1.3 Thermal Characteristics

Table 13 shows the thermal characteristics of W7500.

Table 13 Thermal Characteristics

| Symbol | Ratings | Min | Max | Unit |
|---------------|---|-----|------|------|
| $T_{Storage}$ | Storage temperature range | -55 | +150 | °C |
| T_{Junc} | Maximum junction temperature under bias | -40 | +150 | °C |

25.2 Operating conditions

25.2.1 General Operating Conditions

Table 14 shows the general operating conditions of W7500.

Table 14 General operating conditions

| Symbol | Parameter | Conditions | Min | Max | Unit |
|-------------------|------------------------------|------------|----------------------|-----|------|
| f _{FCLK} | Internal CPU clock frequency | | 0 | 48 | MHz |
| V _{DD} | Standard operating voltage | | 2.7 | 3.6 | V |
| V _{IO} | Input voltage on PIN | | V _{SS} -0.3 | 3.6 | V |
| T _A | Ambient temperature | | -40 | 85 | °C |
| T _J | Junction Temperature range | | -30 | 105 | °C |

25.2.2 Supply Current Characteristics

25.2.2.1 Normal operation

Table 15 shows the Normal operation supply current.

Table 15 Normal operation supply current

| Symbol | Parameter | Conditions1 | Condition2 | Typ | Unit |
|---------------------|----------------|--|----------------------|-------|------|
| I _{DD_NOR} | Supply current | Active mode; code While(1) {} Executed from flash memory | System clock = 10MHz | 6.14 | mA |
| | | | System clock = 20MHz | 8.82 | mA |
| | | | System clock = 40MHz | 14.09 | mA |

25.2.2.2 Sleep mode

Table 16 shows the Normal operation supply current.

Table 16 Sleep mode supply current

| Symbol | Parameter | Conditions1 | Condition2 | Typ | Unit |
|---------------------|----------------|--|----------------------|------|------|
| I _{DD_SLP} | Supply current | After enter sleep mode All peripheral clocks ON (same as system clock) | System clock = 10MHz | 3.51 | mA |
| | | | System clock = 20MHz | 5.65 | mA |
| | | | System clock = 40MHz | 9.61 | mA |

25.2.2.3 Deep sleep mode

Table 17 shows the deep sleep mode operation supply current.

Table 17 Deep sleep mode supply current

| Symbol | Parameter | Conditions1 | Condition2 | Typ | Unit |
|----------------------|----------------|--|------------|------|------|
| I _{DD_DSLP} | Supply current | After enter deep sleep mode All peripheral clocks OFF | - | 2.49 | mA |

25.3 I/O PAD Characteristics

25.3.1 DC Specification

Table 18 shows the DC specification of W7500 I/O PAD.

Table 18 DC specification of PAD

| Symbol | Parameter | Conditions | Min | Max | Unit |
|--------------------------------------|----------------------------|---|-------|-------|------|
| VIH | I/O Input high voltage | | 2.145 | | V |
| VIL | I/O Input low voltage | | | 1.155 | V |
| V _{HYS} | Schmitt trigger hysteresis | | 0.33 | | V |
| IIH | I/O Input high current | | | 1 | uA |
| IIL | I/O Input low current | | -1 | | uA |
| VOH | I/O Output high voltage | High driving strength Iload = 6mA Low driving strength Iload = 3mA | 2.5 | | V |
| VOL | I/O Output high voltage | High driving strength Iload = 6mA Low driving strength Iload = 3mA | | 0.5 | V |
| R _{pup} R _{pdn} | Pull-up/Pull-down resistor | | 20 | 100 | KOhm |

25.4 Flash memory

Table 19 shows the flash memory reliability characteristics of W7500

Table 19 Flash memory Reliability Characteristics

| Symbol | Parameter | Min | Unit |
|------------------|------------------|--------|--------|
| N _{END} | Sector Endurance | 10,000 | Cycles |
| T _{DR} | Data Retention | 10 | Years |

25.5 Electrical Sensitivity Characteristics

25.5.1 Electrostatic discharge (ESD)

Table 20 shows the ESD information of W7500

Table 20 Electrostatic discharge (ESD)

| Symbol | Parameter | Test Method | Min | Max | Unit |
|-----------------------|---|--------------|-------|-----|------|
| V _{ESD(HBM)} | Electrostatic discharge (Human body model) | AEC-Q100-002 | ±2000 | - | V |
| V _{ESD(CDM)} | Electrostatic discharge (Charge device model) | AEC-Q100-011 | ±500 | - | V |

25.5.2 Static latch-up

Table 21 shows the Static latch-up information of W7500

Table 21 Static latch-up

| Symbol | Parameter | Test Method | Min | Max | Unit |
|------------------|---|--------------|------|-----|------|
| I _{LAT} | Latch up current at 125°C ambient temperature | AEC-Q100-004 | ±100 | - | V |

25.6 ADC Characteristics

25.6.1 ADC Electrical characteristics

Table 22 shows the ADC electrical characteristics of W7500

Table 22 ADC electrical characteristics

| Symbol | Parameter | Conditions | Min | Typ | Max | Unit |
|--------------|------------------------------------|------------|--------------------|-----------------|-------|--------------------|
| IN[15:0] | Analog input channel | | V _{SS} | - | VREFP | V |
| VREFP | Reference voltage of REFP | | | V _{DD} | | V |
| RES | Resolution | | | 12 | | Bits |
| Offset error | | | -3.0 | ±1.5 | 3.0 | LSB |
| INL | Integral non-linearity error | | -2.0 | ±1.0 | 2.0 | LSB |
| DNL | Differential non-linearity error | | -1.0 | ±0.8 | 1.5 | LSB |
| Fclk | Clock frequency | | | | 16 | MHz |
| SPS | Sampling rate | | 30 | 500 | 1000 | K |
| TS | Sampling time | | 4/F _{clk} | | | |
| TC | Conversion time | | | 12 | | 1/F _{clk} |
| SNDR | Signal-noise plus distortion ratio | At 10KHz | | 64 | | dB |
| THD | Total harmonic distortion | At 10KHz | | -65 | | dB |
| SFDR | Spurious-free dynamic range | At 10KHz | | 64 | | dB |

25.6.2 ADC Transform function description

Figure 71. ADC transform function shows the ADC transform function of W7500.

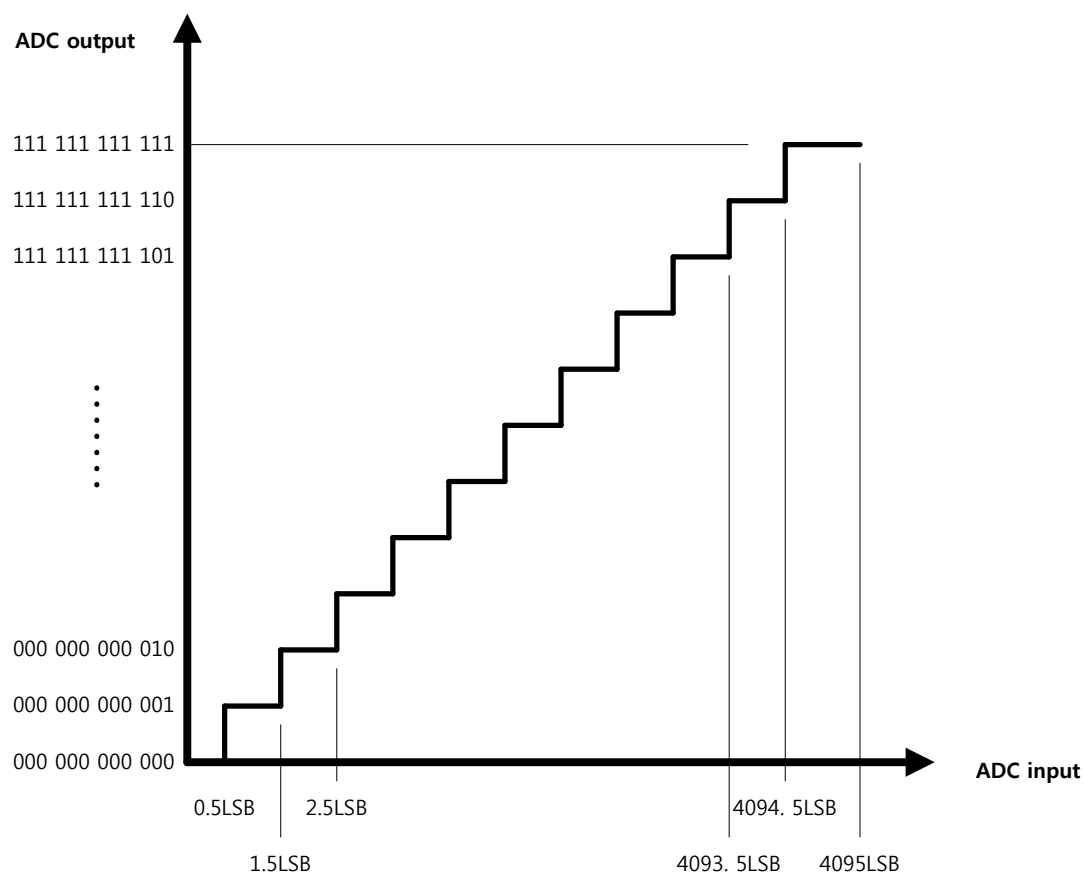


Figure 71. ADC transform function

25.7 I2C interface Characteristics

Table 23 shows the I2C characteristics of W7500. And also refer to Figure 72. This is AC wave form of I2C.

Table 23 I2C characteristics

| Symbol | Parameter | Standard | | Fast | | Unit |
|---------------|-----------------------------|----------|------|------|-----|------|
| | | Min | Max | Min | Max | |
| f_{SCL} | CLK clock frequency | 0 | 100 | 0 | 400 | KHz |
| t_{LOW} | Low period of the SCL clock | 4.5 | | 1.0 | | us |
| t_{HIGH} | High period of SCL clock | 3.8 | | 0.5 | | us |
| t_r | Rise time of SCL and SDA | | 1000 | | 300 | ns |
| t_f | Fall time SCL and SDA | | 300 | | 300 | ns |
| t_{HD_DAT} | Data hold time | 0 | | 0 | | us |
| t_{VD_DAT} | Data valid time | | 3.5 | | 1.0 | us |

| | | | | | | |
|---------------|--|-----|-----|-----|-----|----|
| t_{SU_DAT} | Data setup time | 200 | | 90 | | ns |
| t_{VD_ACK} | Data valid acknowledge time | | 3.5 | | 1.0 | us |
| t_{HD_STA} | Hold time START condition | 3.8 | | 0.5 | | us |
| t_{SU_STA} | Set-up time for a repeat START condition | 4.5 | | 0.5 | | us |
| t_{SU_STO} | Set-up time for STOP condition | 3.8 | | 0.5 | | us |

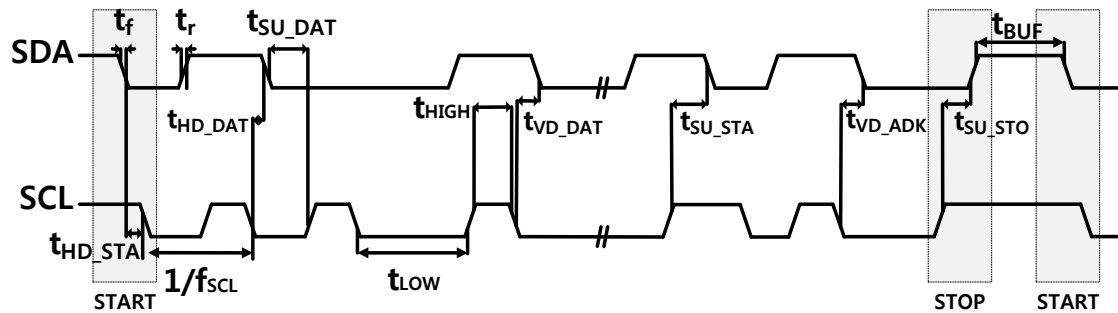


Figure 72. I2C bus AC waveform

25.8 SSP Interface Characteristics

Table 24 shows the SSP characteristics of W7500.

Table 24 SSP characteristics

| Symbol | Parameter | Conditions | Min | Max | Unit |
|--------------|----------------------------------|----------------------------|-----|-----|------|
| f_{SCL} | SSP clock frequency | Master mode | | 20 | MHz |
| | | Slave mode | | 20 | MHz |
| t_{r_SCK} | SSP clock rising and fall time | Capacitive load : C = 25pF | | 8 | ns |
| t_{SU_M} | Data input setup time | Master mode | 5 | | ns |
| t_{SU_S} | | Slave mode | 6 | | ns |
| t_{H_M} | Data input hold time | Master mode | 5 | | ns |
| t_{H_S} | | Slave mode | 6 | | ns |
| t_{V_M} | Data output valid time | Master mode | | 20 | ns |
| t_{V_S} | | Slave mode | | 5 | ns |
| t_{H_M} | Data output hold time | Master mode | 13 | | |
| t_{H_S} | | Slave mode | 3 | | |
| DuCy | SPI slave input clock duty cycle | Slave mode | 45 | 55 | % |

26 Package Characteristics

26.1 Package dimension information

Figure 73 shows the package dimension information.

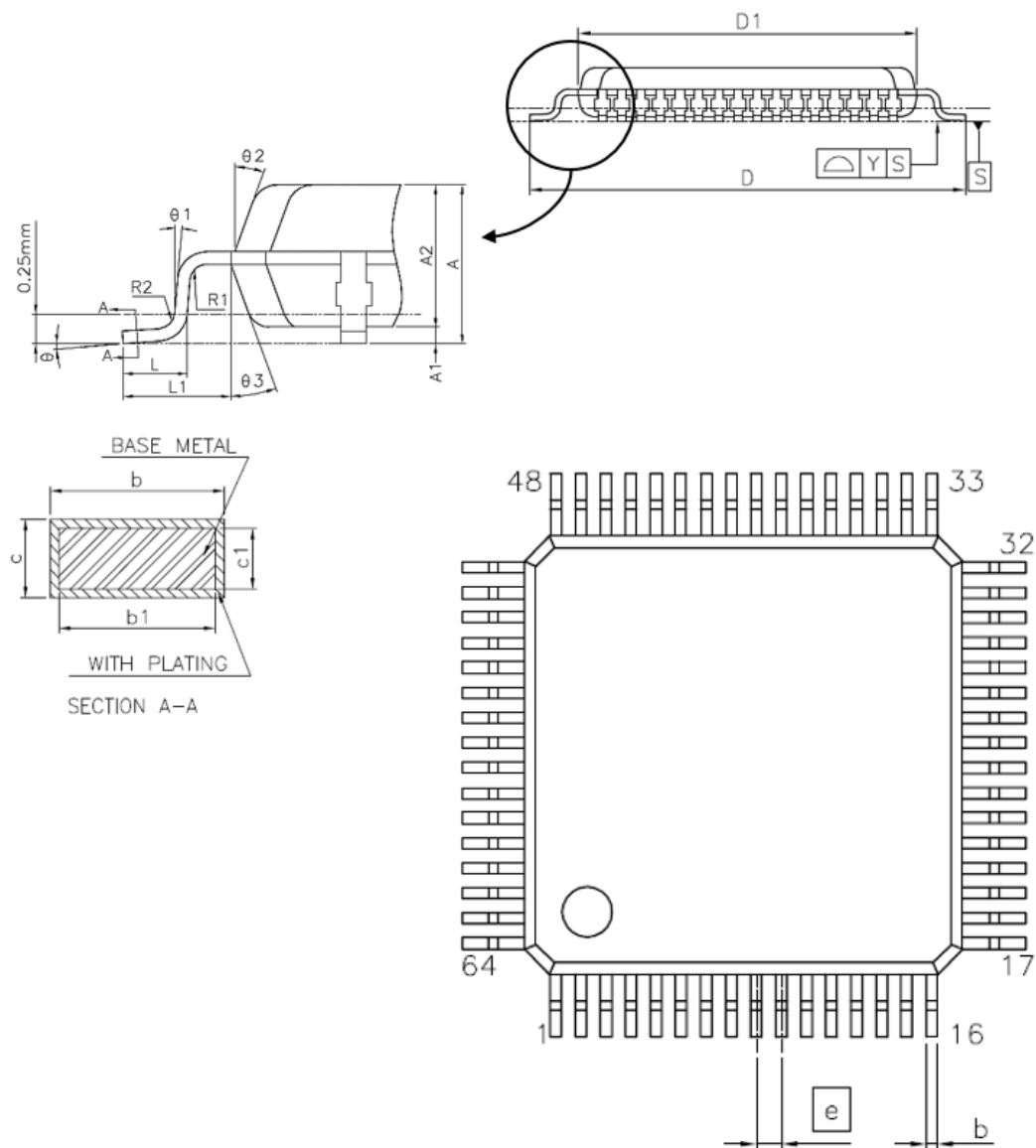


Figure 73 Package Dimension Information

Table 25 Package mechanical data

| Symbol | DIMENSION (MM) | | | DIMENSION (MIL) | | |
|--------|----------------|------|------|-----------------|------|------|
| | MIN | NOM | MAX | MIN | NOM | MAX |
| A | | | 1.20 | | | 47.2 |
| A1 | 0.05 | 0.10 | 0.15 | 2.0 | 3.9 | 5.9 |
| A2 | 0.95 | 1.00 | 1.05 | 37.4 | 39.4 | 41.3 |

| | | | | | | |
|----|------|------|------|-------|-------|-------|
| b | 0.13 | 0.18 | 0.23 | 5.1 | 7.1 | 9.1 |
| b1 | 0.13 | 0.16 | 0.19 | 5.1 | 6.3 | 7.5 |
| c | 0.09 | | 0.20 | 3.5 | | 7.9 |
| c1 | 0.09 | | 0.16 | 3.5 | | 6.3 |
| D | 8.90 | 9.00 | 9.10 | 350.4 | 354.3 | 358.3 |
| D1 | 6.90 | 7.00 | 7.10 | 271.7 | 275.6 | 279.5 |
| e | 0.35 | 0.40 | 0.45 | 13.8 | 15.7 | 17.7 |
| L | 0.45 | 0.60 | 0.75 | 17.7 | 23.6 | 29.5 |
| L1 | 0.90 | 1.00 | 1.10 | 35.4 | 39.4 | 43.3 |
| R1 | 0.08 | | | 3.1 | | |
| R2 | 0.08 | | 0.20 | 3.1 | | 7.9 |
| Y | | | 0.08 | | | 3.1 |
| θ | 0° | 3.5° | 7° | 0° | 3.5° | 7° |
| θ1 | 0° | | | 0° | | |
| θ2 | 11° | 12° | 13° | 11° | 12° | 13° |
| θ3 | 11° | 12° | 13° | 11° | 12° | 13° |

26.2 Package footprint information

Figure 74 shows the Footprint information.

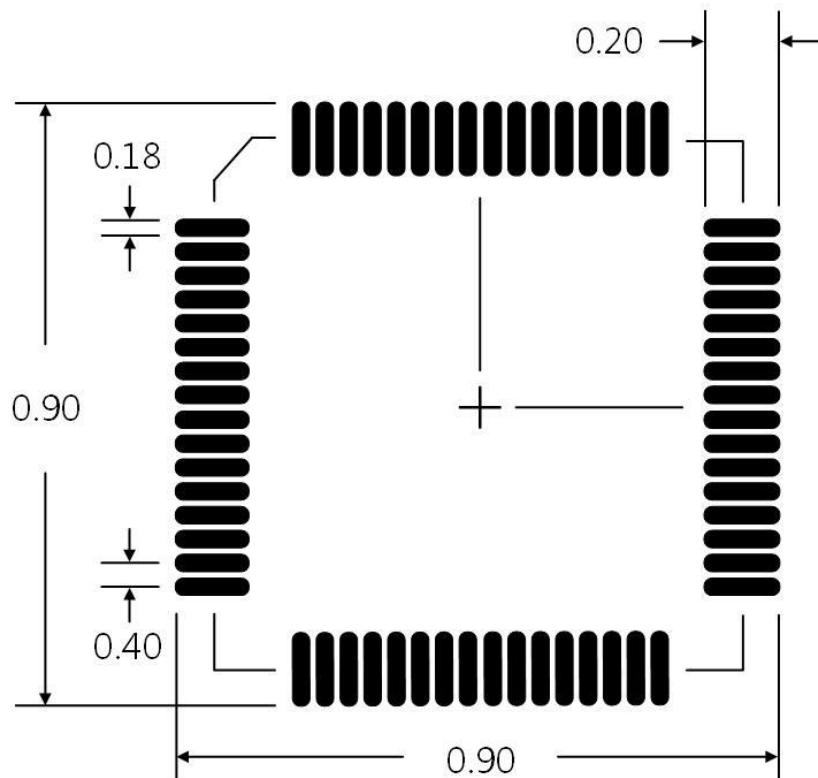


Figure 74 Footprint Information

Document History Information

| Version | Date | Descriptions |
|------------|------------|---|
| Ver. 1.0.0 | 01MAY2015 | Initial Release |
| Ver. 1.0.1 | 12JUNE2015 | Corrected typing error: I2C block |
| Ver. 1.0.2 | 17AUG2015 | <ol style="list-style-type: none">1. Deleted 1.1 List of abbreviations.2. Corrected Register acronym in Table 3. Offset Address for Common Register : PSID -> PSIDR.3. Deleted PP mode with Figure 6 & Table 5 in 8. Booting Sequence4. Changed words : polling INT bit -> checking INT register in 17.3.2 Operation ADC with interrupt.5. Deleted unneeded content in 9.2.2 Read operations.6. Corrected nUARTRTS asserted condition : HIGH -> LOW , Corrected the sentence, 'When nUARTCTR is re-asserted(to a low value) the transmitter sends the next byte.' : de-asserted -> re-asserted in 22.3.4 Hardware flow control.7. Deleted RTCCLK (P36, P37) |
| Ver. 1.0.3 | 14DEC2015 | Corrected Package diagram error |
| Ver. 1.0.4 | 08JAN2016 | <ol style="list-style-type: none">1. Bit rate in SPI interface changed 'up to 2MHz and higher' to 'up to 20MHz'.2. Bit rate generation example changed in 23.3.11 Programing the SSPCR1 Control Register |
| Ver. 1.0.5 | 08MAR2016 | <ol style="list-style-type: none">1. In 2.2.2 Memory map, GPIO 0,1,2,3 changed to GPIO A,B,C,D |
| Ver. 1.0.6 | 03JUN2016 | <ol style="list-style-type: none">1. Added UART2 description2. Deleted the greater part of I^2C with data receiving problem, set up the Errata Sheet Link (http://wizwiki.net/wiki/doku.php?id=products:w7500:documents) |

Copyright Notice

Copyright 2015 WIZnet Co., Ltd. All Rights Reserved.

Technical Support: <http://wizwiki.net/forum>

Sales & Distribution: sales@wiznet.co.kr

For more information, visit our website at <http://www.wiznet.co.kr>