**Welcome to E-XFL.COM**

## What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

## Applications of "Embedded - Microcontrollers"

| Details | |
|---|---|
| Product Status | Obsolete |
| Core Processor | ARM® Cortex®-M3 |
| Core Size | 32-Bit Single-Core |
| Speed | 80MHz |
| Connectivity | I²C, SPI, UART/USART |
| Peripherals | Brown-out Detect/Reset, DMA, LVD, POR, PWM, WDT |
| Number of I/O | 86 |
| Program Memory Size | 384KB (384K x 8) |
| Program Memory Type | FLASH |
| EEPROM Size | - |
| RAM Size | 16K x 8 |
| Voltage - Supply (Vcc/Vdd) | 3V ~ 5.5V |
| Data Converters | A/D 16x12b |
| Oscillator Type | Internal |
| Operating Temperature | -40°C ~ 85°C (TA) |
| Mounting Type | Surface Mount |
| Package / Case | 100-LQFP |
| Supplier Device Package | 100-LQFP (14x14) |
| Purchase URL | https://www.e-xfl.com/product-detail/zilog/z32f38412als |

# Revision History

Each instance in this document's revision history reflects a change from its previous edition. For more details, refer to the corresponding page(s) or appropriate links furnished in the table below.

| Date | Revision Level | Description | Page |
|---|---|---|---|
| Jun 2017 | 03 | Updated part numbers to include the Cortex M identifier. | All |
| Mar 2016 | 02 | Update to reflect new part, revision B (0x0002);<br>Added timing information for most of the peripherals; corrected typos. | All |
| Dec 2015 | 01 | Original issue. | |

# MCCR4 Miscellaneous Clock Control Register 4

The Miscellaneous Clock Control Register 4 is the alterntative ADC and NMI Debounce Clock Control register.

0x4000_00A8

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | ADCCSEL | | | ADCCDIV | | | | | | | | | | | | | | NMICSEL | | | NMIDDIV | | | | | | | |
| 0 | 0 | 0 | 0 | 0 | 000 | | | 0x00 | | | | | | | | 0 | 0 | 0 | 0 | 0 | 000 | | | 0x01 | | | | | | | |
| | | | | | RW | | | RW | | | | | | | | | | | | | RW | | | RW | | | | | | | |

| | | |
|---|---|---|
| 26<br>24 | ADCCSEL | ADC clock source select bit<br>0xx RING OSC 1MHz<br>100 MCLK (bus clock)<br>101 INT OSC 20MHz<br>110 External Main OSC<br>111 PLL Clock |
| 23<br>16 | ADCCDIV | ADC Clock N divider |
| 10<br>8 | NMIDCSEL | Debounce Clock for NMI source select bit<br>0xx RING OSC 1MHz<br>100 MCLK (bus clock)<br>101 INT OSC 20MHz<br>110 External Main OSC<br>111 PLL Clock |
| 7<br>0 | NMIDDIV | NMI Debounce Clock N divider |

# 8. Internal SRAM

## Overview

The Z32F3841 MCU implements zero-wait on the chip's SRAM. The size of the SRAM is 16 KB. The SRAM base address is 0x2000_0000.

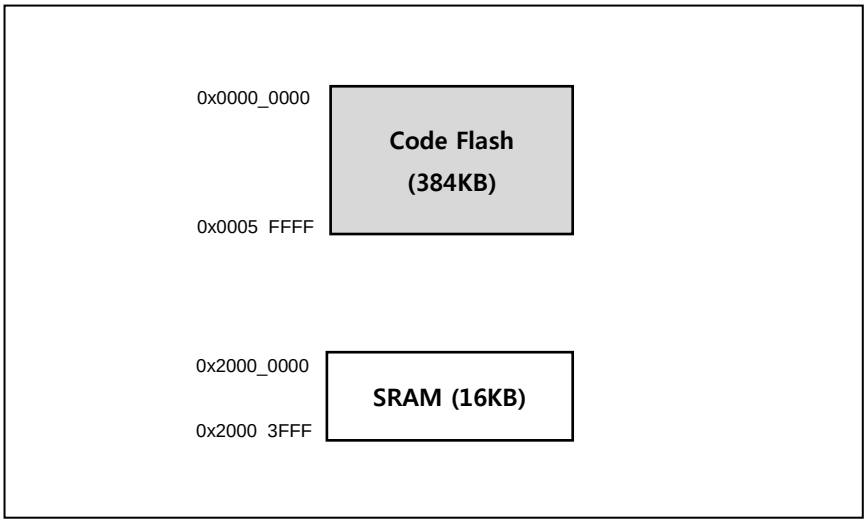



**Figure 8.1. SRAM Block Diagram**

# Registers

The base address of the watchdog timer is `0x4000_0200` and the register map is listed in Table 10.1. Initial watchdog time-out period is set to 2000-miliseconds.

**Table 10.1. Watchdog Timer Register Map**

| Name | Offset | R/W | Description | Reset |
|---|---|---|---|---|
| **WDT.LR** | 0x0000 | W | WDT Load register | 0x00000000 |
| **WDT.CNT** | 0x0004 | R | WDT Current counter register | 0x0000FFFF |
| **WDT.CON** | 0x0008 | R/W | WDT Control register | 0x0000805C |

## WDT.LR Watchdog Timer Load Register

The Watchdog Timer Load register is used to update the WDTCNT register. To update the WDTCNT register, the WEN bit of WDTCON should be set to 1 and written into the WDTLR register with a target value of WDTCNT.

**WDT.LR=0x4000_0200**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **WDTLR** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| **0x0000_0000** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| **RW** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

| | | |
|---|---|---|
| 31<br>0 | WDTLR | Watchdog timer load value register<br>Keeping WEN bit as '1', write WDTLR register will update WDTCNT value with written value |

## WDT.CNT Watchdog Timer Current Counter Register

The Watchdog Timer Current Counter register represents the current count value of the 32-bit down counter .When the counter value reaches 0, the interrupt or reset is awoken.

**WDT.CNT=0x4000_0204**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **WDTCNT** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| **0x0000_FFFF** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| **RW** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

| | | |
|---|---|---|
| 31<br>0 | WDTCNT | Watchdog timer current counter register<br>32-bit down counter will run from the written value. |

# Functional Description

The MCCR3 register must be configured to enable the clock source and divider for the Watch Dog Timer (WDT) to run. To prevent the WDT from resetting or interrupting, load a new value into the WDTLR register before the WDTCNT reaches 0.

The watchdog timer count is enabled by setting WDTEN (WDT.CON[4]) to 1. When the watchdog timer is enabled, the down counter starts counting from the load value. If WDTRE (WDT.CON[6]) is set to 1, WDT reset is asserted when the WDT counter value reaches 0 (underflow event) from the WDTLR value. Before the WDT counter goes down to 0, the software can write a certain value to the WDTLR register to reload the WDT counter.

## Timing Diagram

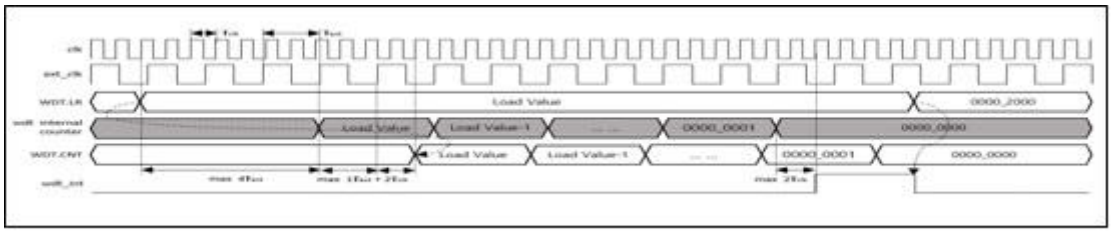



**Figure 10.2. Timing Diagram in Interrupt Mode Operation when WDT Clock is External Clock**

In WDT interrupt mode, after the WDT underflow occurs, a certain count value is reloaded to prevent the next WDT interrupt in a short time period. This reloading action can only be activated when the watchdog timer counter is set to Interrupt mode (set WDTIE of WDT.CON). It takes up to 5 cycles from the load value to the CNT value. The WDT interrupt signal and CNT value data might be delayed by a maximum of 2 system bus clocks in synchronous logic.

## Prescale Table

The WDT includes a 32-bit down counter with programmable prescaler to define different time-out intervals.

The clock sources of the watchdog timer can include the peripheral clock (PCLK) or one of 3 external clock sources. An external clock source can be enabled by CKSEL (WDT.CON[3]) set to ‘1’ and external clock source chosen in MCCR3 register of the System Control Unit block.

To make the WDT counter base clock, users can control the 3-bit prescaler WPRS [2:0] in the WDT.CON register and the maximum prescaled value is “clock source frequency/256”. The prescaled WDT counter clock frequency values are listed in Table 10.2.

**Selectable clock source (40 kHz ~ 16 MHz) and the time out interval when 1 count**

**Time out period = {(Load Value) * (1/pre-scaled WDT counter clock frequency) + max $5T_{ext}$} + max $4T_{clk}$**

*Time out period (time out period from load Value to interrupt set ‘1’)

## PWM Timer Output

Figure 11.5 shows the timing diagram in PWM output mode. The Tn.GRB value decides the PWM pulse period. An additional comparison point is provided with the Tn.GRA register value which defines the pulse width of PWM output.

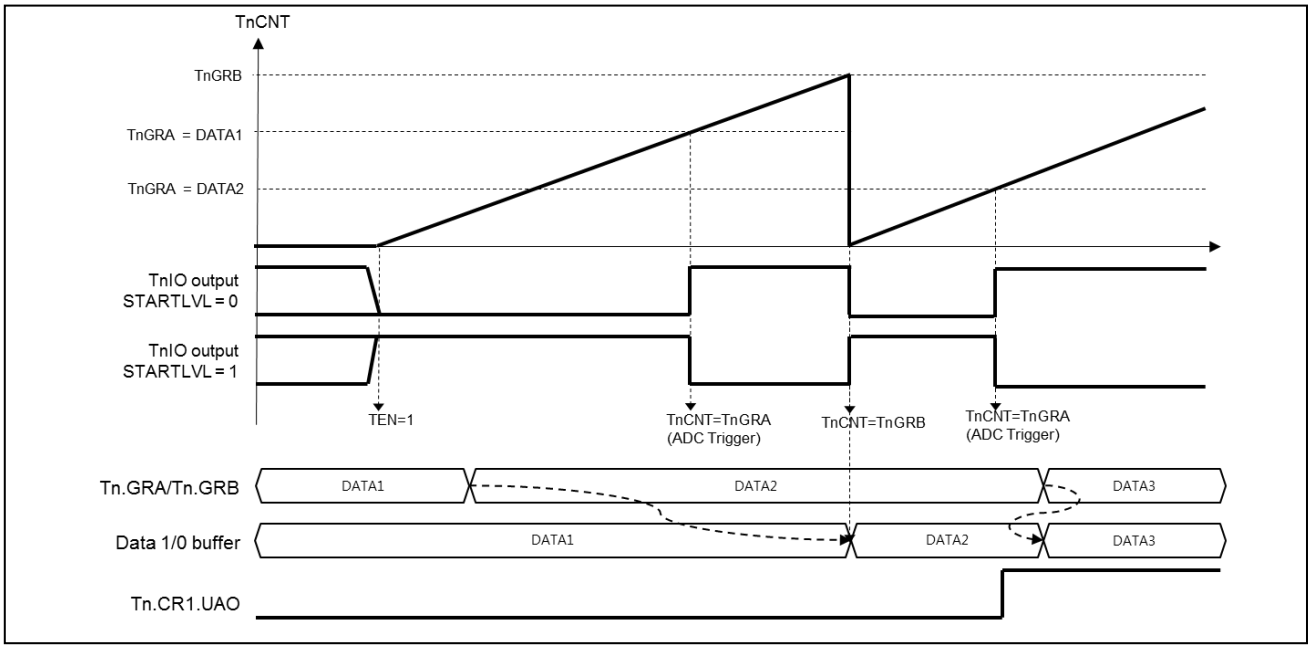



**Figure 11.5. PWM Output Operation**

The period of PWM pulse can be calculated as shown in the following equation:

**The period = TMCLK Period * Tn.GRB value**
**Match A interrupt time = TMCLK Period * Tn.GRA value**

If Tn.GRB = 0, the timer cannot be started even if TnCR2.TEN is “1” because the period is “0”.

The value in Tn.GRA and Tn.GRB is loaded into the internal compare data buffer 0 and 1 when the loading condition occurs. In this periodic mode with TnCR1.UAO =0, the Tn.CR2.TCLR write operation and the GRB match event will load the compare data buffers.

When TnCR1.UAO is 1, the internal compare data buffer is updated when the Tn.GRA or Tn.GRB data is updated.

The TnIO output signal generates a PWM pulse. The Tn.GRB value defines the output pulse period and the Tn.GRA value defines the pulse width of one shot pulse.The active level of the PWM pulse can be controlled by the Tn.CR1.STARTLVL bit value.

ADC Trigger generation is available at Match A interrupt time.

## FRT.MR FRT Mode Register

FRT is a 32-bit up counter. It can be used in power down mode. The SUB OSC clock is directly connected to FRT. The clock is uncontrollable in the SCU block. The FRT Mode Register is an 8-bit register.

**FRTMR=0x4000_0600**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| | | | CLKSEL | | MCD | OVIE | MIE |
| 0 | 0 | 0 | 00 | | 0 | 0 | 0 |
| | | | RW | | RW | RW | RW |

| | | |
|---|---|---|
| 3 | CLKSEL | FRT counter clock source control |
| 2 | | 0 Internal Oscillator clock divided by 16 |
| | | 1 External Oscillator clock divided by 16 |
| | | 2 Sub Oscillator clock |
| | | 3 Reserved |
| 1 | MCD | Counter Match Clear Disable bit |
| | | 0 Counter Match Clear function is enabled. Whenever the counter matches FRT.PER, the counter will be set zero and waiting for MF to be cleared. |
| | | 1 Counter Match Clear function is disabled. The counter will keep countering without set zero |
| 1 | OVIE | Overflow Interrupt Enable bit |
| | | 0 Not effect |
| | | 1 Interrupt enabled |
| 0 | MIE | Match Interrupt Enable bit |
| | | 0 Not effect |
| | | 1 Interrupt enabled |

## FRT.CR FRT Control Register

The FRT Control register is an 8-bit register.

**FRTCR=0x4000_3E04**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| | | | | CNTREQ | FCLR | FHOLD | FEN |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | | | RW | WO | R/W | R/W |

| | | |
|---|---|---|
| 3 | CNTREQ | FRT Counter read request bit |
| | | 0 No |
| | | 1 Request to read FCNT (cleared when CNTACK(FSR[1]) is high) |
| 2 | FCLR | FRT Counter register clear bit |
| | | 0 No |
| | | 1 Clear the counter |
| 1 | FHOLD | FRT Counter register hold bit |
| | | 0 No |
| | | 1 Hold the counter |
| 0 | FEN | FRT enable bit |
| | | 0 FRT Disabled |
| | | 1 FRT Enabled |

## FRT.PER FRT Period Match Register

The FRT Period Match register is a 32-bit register.

**FRTPER=0x4000_3E08**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DATA | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x0000_0000 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| R/W | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

| | | |
|---|---|---|
| 32<br>0 | DATA | FRT match data |

## FRT.CNT FRT Counter Register

The FRT Counter Register is a 32-bit register.

**FRTCNT=0x4000_3E0C**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CNT | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0x0000_0000 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| RO | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

| | | |
|---|---|---|
| 32<br>0 | CNT | FRT Counter |

## FRT.SRFRT Status Register

**FRT Status Register is an 8-bit register.FRTSR=0x4000_0610**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| | | | | | RACK | OVIF | MIF |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | | | | WC1 | WC1 | WC1 |

| | | |
|---|---|---|
| 2 | RACK | Read Counter Acknowledge bit<br>0 Not ready to read CNT value<br>1 Ready to read CNT value |
| 1 | OVIF | Overflow interrupt flag bit<br>0 Overflow interrupt did not occur<br>1 Overflow interrupt occurred |
| 0 | MIF | Match interrupt flag bit<br>0 Match interrupt did not occur.<br>1 Match Interrupt occurred<br>In Counter Match Clear mode, this bit should be cleared for restarting the counter. |

## Un.RBR Receive Buffer Register

The UART Receive Buffer Register is an 8-bit read-only register.

**U0.RBR=0x4000_8000, U1.RBR=0x4000_8100**
**U2.RBR=0x4000_8200, U3.RBR=0x4000_8300**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| RBR[7:0] | | | | | | | |
| - | | | | | | | |
| RO | | | | | | | |

| | | |
|---|---|---|
| 7<br>0 | RBR | Receive Buffer Register |

## Un.THR Transmit Data Hold Register

The UART Transmit Data Hold Register is an 8-bit write-only register.

**U0.THR=0x4000_8000, U1.THR=0x4000_8100**
**U2.THR=0x4000_8200, U3.THR=0x4000_8300**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| THR | | | | | | | |
| - | | | | | | | |
| WO | | | | | | | |

| | | |
|---|---|---|
| 7<br>0 | THR | Transmit Data Hold Register |

# Un.LCR UART Line Control Register

The UART Line Control Register is an 8-bit register.

**U0.LCR=0x4000_800C, U1.LCR=0x4000_810C**
**U2.LCR=0x4000_820C, U3.LCR=0x4000_830C**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| | BREAK | STICKP | PARITY | PEN | STOPBIT | DLEN | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| | | | |
|---|---|---|---|
| 6 | BREAK | When this bit is set, TxD pin will be driven at low state in order to notice the alert to the receiver. | |
| | | 0 | Normal transfer mode |
| | | 1 | Break transmit mode |
| 5 | STICKP | Force parity and it will be effective when PEN bit is set. | |
| | | 0 | Parity stuck is disabled |
| | | 1 | Parity stuck is enabled and parity always the bit of PARITY. |
| 4 | PARITY | Parity mode selection bit and stuck parity select bit | |
| | | 0 | Odd parity mode |
| | | 1 | Even parity mode |
| 3 | PEN | Parity bit transfer enable | |
| | | 0 | The parity bit disabled |
| | | 1 | The parity bit enabled |
| 2 | STOPBIT | The number of stop bit followed by data bits. | |
| | | 0 | 1 stop bit |
| | | 1 | 1.5 / 2 stop bit<br>In case of 5 bit data case, 1.5 stop bit is added. In case of 6,7 or 8 bit data, 2 stop bit is added |
| 1<br>0 | DLEN | The data length in one transfer word. | |
| | | 00 | 5 bit data |
| | | 01 | 6 bit data |
| | | 10 | 7 bit data |
| | | 11 | 8 bit data |

The parity bit is generated according to bits 3,4,5 of the UnLCR register. Table 13.4 shows the variation of parity bit generation.

**Table 13.4. Parity Bit Generation**

| STICKP | PARITY | PEN | Parity |
|---|---|---|---|
| X | X | 0 | No Parity |
| 0 | 0 | 1 | Odd Parity |
| 0 | 1 | 1 | Even Parity |
| 1 | 0 | 1 | Force parity as “1” |
| 1 | 1 | 1 | Force parity as “0” |

## Un.IDTR Inter-frame Delay Time Register

The UART Inter-frame Time Register is an 8-bit register. A dummy delay can be inserted between two continuous transmits.

**U0.IDTR=0x4000_8030, U1.IDTR=0x4000_8130**
**U2.IDTR=0x4000_8230, U3.IDTR=0x4000_8330**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| - | | | | | WAITVAL | | |
| 0 | 0 | 0 | 0 | 0 | 000 | | |
| | | | | | RW | | |

| | | |
|---|---|---|
| 2 0 | WAITVAL | Wait time is decided by this value<br>$\text{Wait Time} = \frac{WAITVAL}{BAUDRATE}$ |

# Functional Description

The UART module is compatible with the 16450 UART. Additionally, dedicated DMA channels and fractional baud rate compensation logic are provided. The UART does not have an internal FIFO block. Therefore, data transfers are established interactively or with DMA support.

Two DMA channels are provided for each UART module – one channel for TX transfer and the other channel for RX transfer. Each channel has a 32-bit memory address register and a 16-bit transfer counter register. Prior to the DMA operation, the DMA memory address register and transfer count register should be configured. For the RX operation, the memory address is the destination memory address and for the TX operation, the memory address is the source memory address.

The transfer counter register stores the number of data transfers. When a single transfer is done, the counter is decremented by 1. When the counter reaches zero, the DMA done flag is delivered to the UART control block. If the interrupt is enabled, this flag generates the interrupt.

## Receiver Sampling Timing

The UART operates per the following timing:

If the falling edge is on the receive line, the UART judges it as the start bit. From the start timing, the UART oversamples 16 times of 1-bit and detects the bit value at the 7th sample of 16 samples.

# Pin Description

**Table 15.1. I²C Interface External Pins**

| PIN NAME | TYPE | DESCRIPTION |
|---|---|---|
| **SCL0** | I/O | I²C channel 0 Serial clock bus line (open-drain) |
| **SDA0** | I/O | I²C channel 0 Serial data bus line (open-drain) |
| **SCL1** | I/O | I²C channel 1 Serial clock bus line (open-drain) |
| **SDA1** | I/O | I²C channel 1 Serial data bus line (open-drain) |

# Registers

The base address of I²C0 is `0x4000_A000` and the base address of I²C1 is `0x4000_A100`. The register map is described in Tables 15.2 and 15.3.

**Table 15.2. I²C Interface Base Address**

| Channel | Base address |
|---|---|
| **I²C0** | 0x4000_A000 |
| **I²C1** | 0x4000_A100 |

**Table 15.3. I²C Register Map**

| Name | Offset | R/W | Description | Reset |
|---|---|---|---|---|
| **IC0.DR** | 0xA000 | R/W | I²C0 Data Register | 0xFF |
| **IC0.SR** | 0xA008 | R, R/W | I²C0 Status Register | 0x00 |
| **IC0.SAR** | 0xA00C | R/W | I²C0 Slave Address Register | 0x00 |
| **IC0.CR** | 0xA014 | R/W | I²C0 Control Register | 0x00 |
| **IC0.SCLL** | 0xA018 | R/W | I²C0 SCL LOW duration Register | 0xFFFF |
| **IC0.SCLH** | 0xA01C | R/W | I²C0 SCL HIGH duration Register | 0xFFFF |
| **IC0.SDH** | 0xA020 | R/W | I²C0 SDA Hold Register | 0x7F |
| **IC1.DR** | 0xA100 | R/W | I²C1 Data Register | 0xFF |
| **IC1.SR** | 0xA108 | R, R/W | I²C1 Status Register | 0x00 |
| **IC1.SAR** | 0xA10C | R/W | I²C1 Slave Address Register | 0x00 |
| **IC1.CR** | 0xA114 | R/W | I²C1 Control Register | 0x00 |
| **IC1.SCLL** | 0xA118 | R/W | I²C1 SCL LOW duration Register | 0xFFFF |
| **IC1.SCLH** | 0xA11C | R/W | I²C1 SCL HIGH duration Register | 0xFFFF |
| **IC1.SDH** | 0xA120 | R/W | I²C1 SDA Hold Register | 0x007F |

# ICn.SCLH I²C SCL HIGH Duration Register

ICnSCLH is a 16-bit read/write register. The SCL HIGH time is set by writing this register in master mode.

IC0.SDLH=0x4000_A01C, IC1.SDLH=0x4000_A11C

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

| | | |
|---|---|---|
| 15 | SCLH | SCL HIGH duration value.<br>SCLH = ( PCLK * SCLH[15:0] ) + 3 PCLKs |
| 0 | | Default value is 0xFFFF. |

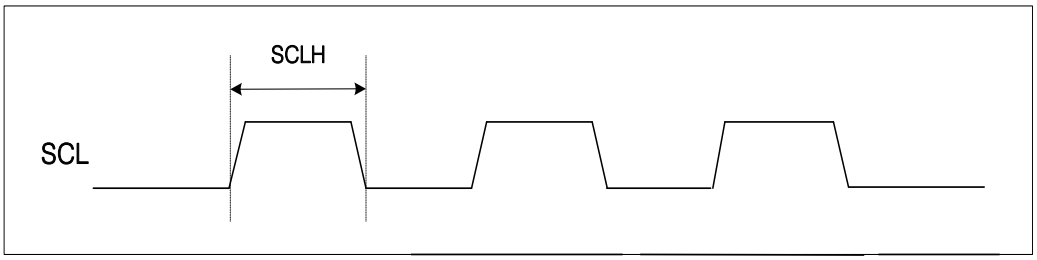



**Figure 15.3.SCL HIGH Timing**

A message which starts with such an address can be terminated by generation of a STOP conditions, even during the transmission of a byte. In this case, no acknowledge is generated.

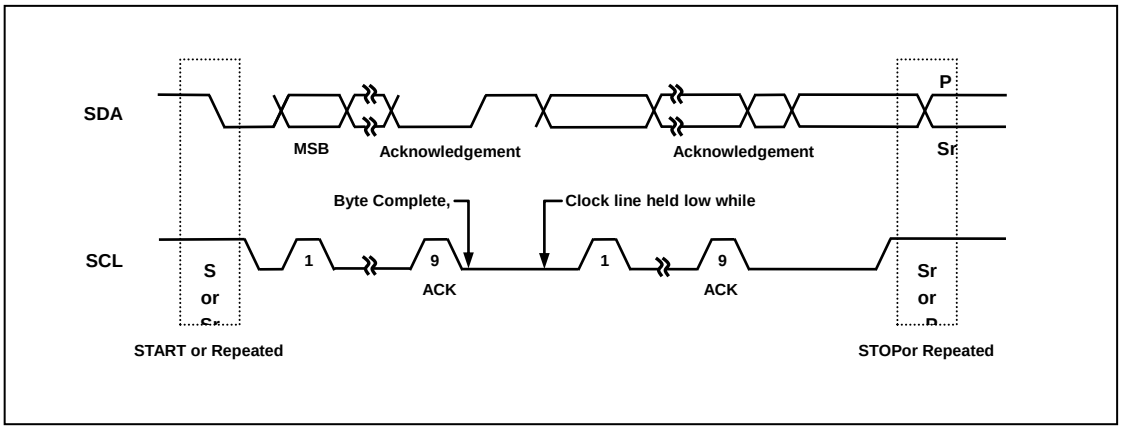



**Figure 15.7. I²C Bus Data Transfer**

## Slave Transmitter

The slave transmitter shows the flow of the transmitter in Slave mode, as shown in Figure 15.13.

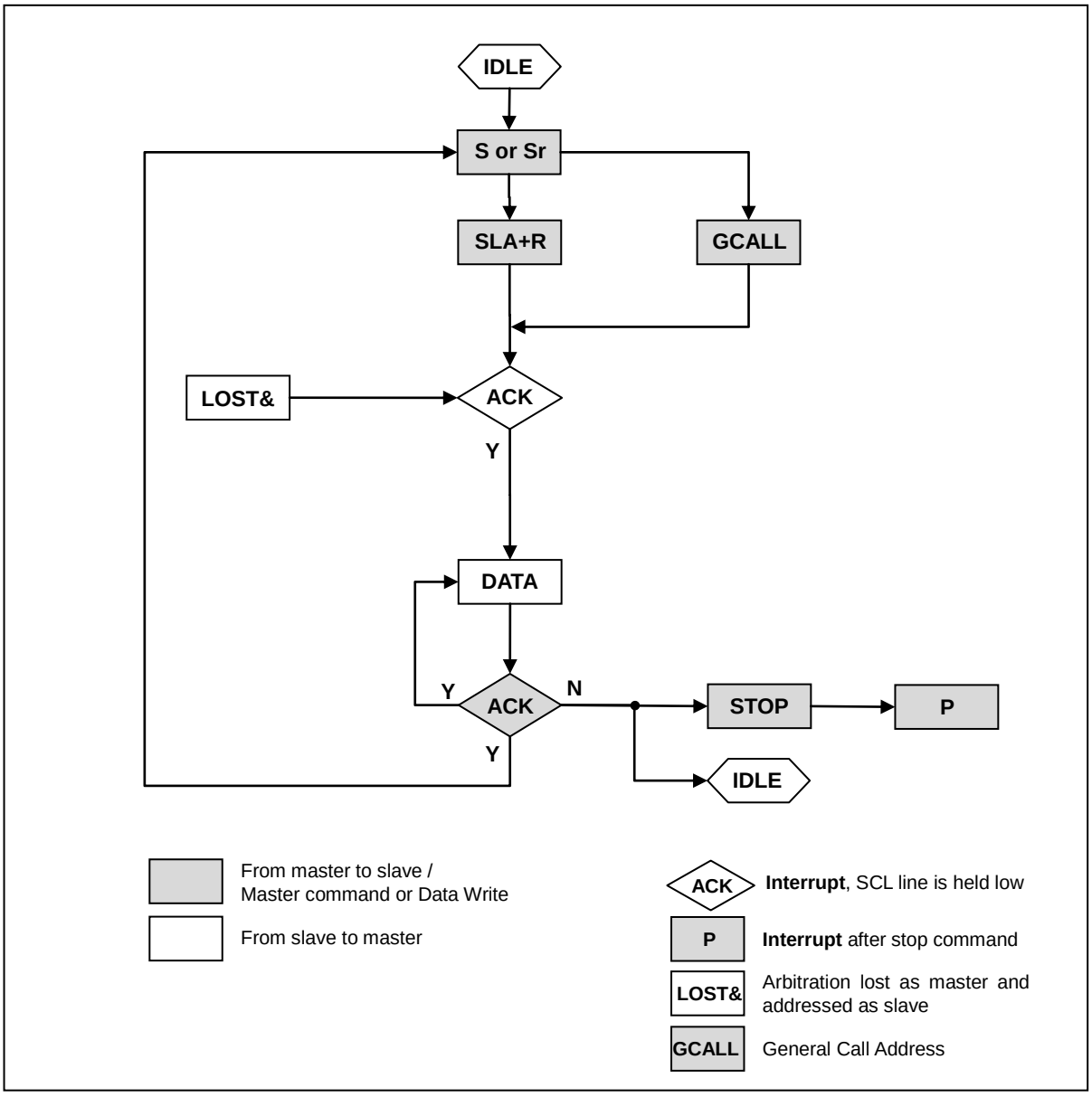



**Figure 15.13. Transmitter Flowchart in Slave Mode**

## Slave Receiver

The slave receiver shows the flow of the receiver in Slave mode, as shown in Figure 15.14.

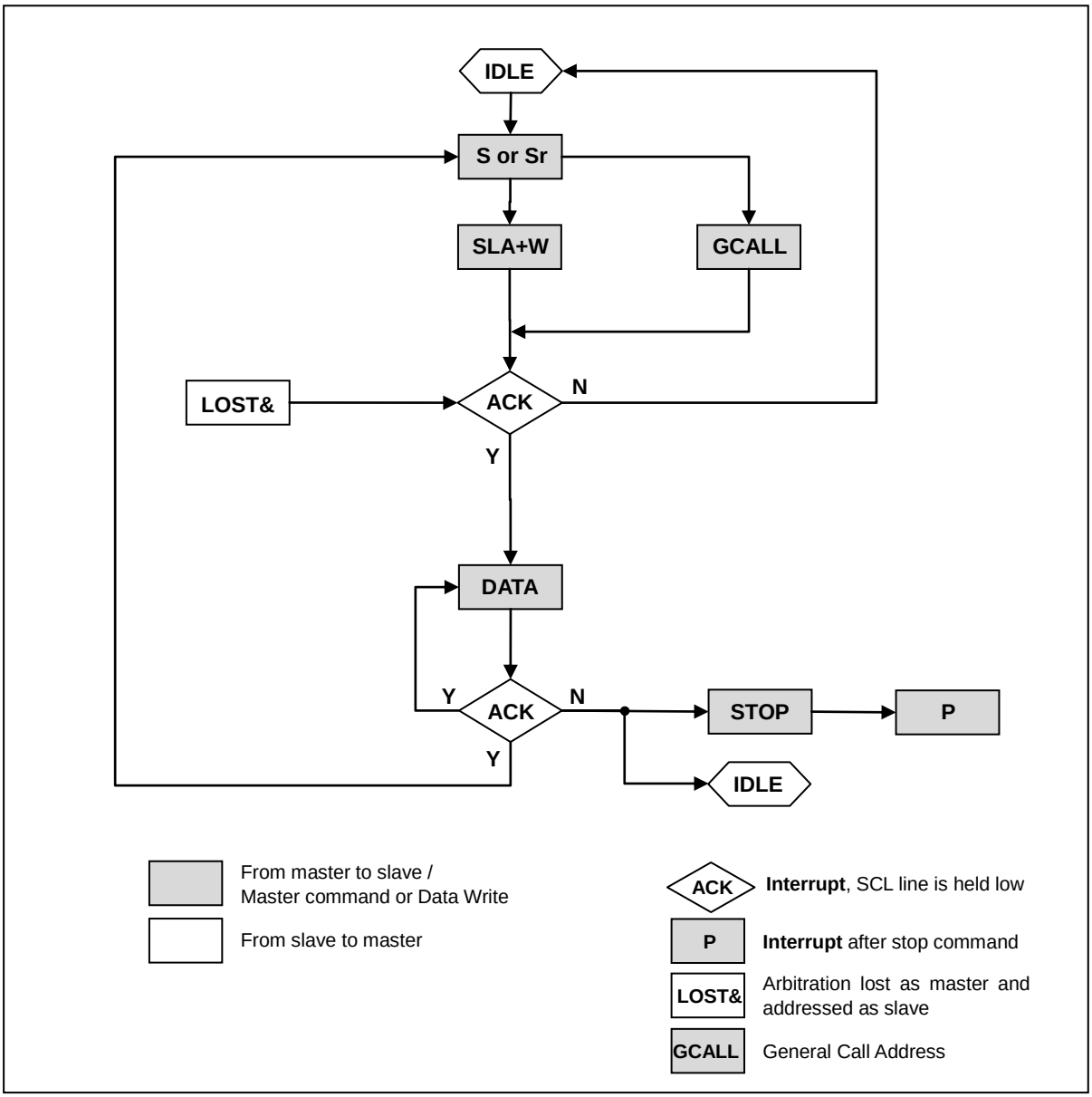



**Figure 15.14. Receiver Flowchart in Slave Mode**

**Table 16.1. MPWM Register Map**

| PWM Output | Level | NORMAL mode | | MOTOR mode |
|---|---|---|---|---|
| | | UP mode | UPDOWN mode | |
| WH | Default level | LOW | HIGH | LOW |
| | Active level | HIGH | LOW | HIGH |
| WL | Default level | LOW | LOW | HIGH |
| | Active level | HIGH | HIGH | LOW |
| VH | Default level | LOW | HIGH | LOW |
| | Active level | HIGH | LOW | HIGH |
| VL | Default level | LOW | LOW | HIGH |
| | Active level | HIGH | HIGH | LOW |
| UH | Default level | LOW | HIGH | LOW |
| | Active level | HIGH | LOW | HIGH |
| UL | Default level | LOW | LOW | HIGH |
| | Active level | HIGH | HIGH | LOW |

Figure 16.2 shows the polarity control block. This is an example of WH signal polarity control.

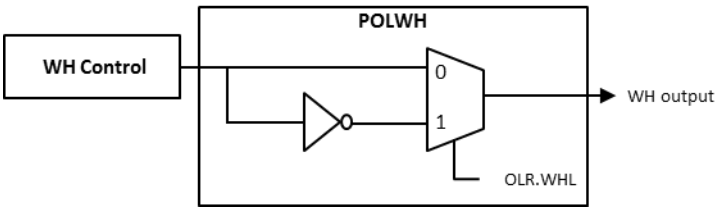



**Figure 16.2 Polarity Control Block**

## MPn.FOR MPWM Force Output Level Register

The PWM force output register is an 8-bit register. The PWM output level can be forced by an abnormal event from an external or user-intended condition. When the forced condition occurs, each PWM output level which is programmed in the FOLR register will be forced.

**MP0.FOR=0x4000_4008, MP1.FOR=0x4000_5008,**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| | | WHFL | VHFL | UHFL | WLFL | VLFL | ULFL |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | RW | RW | RW | RW | RW | RW |

| | | |
|---|---|---|
| 5 | **WHFL** | Select WH Output Force Level |
| | | 0 Output Force Level is 'L' |
| | | 1 Output Force Level is 'H' |
| 4 | **VHFL** | Select VH Output Force Level |
| | | 0 Output Force Level is 'L' |
| | | 1 Output Force Level is 'H' |
| 3 | **UHFL** | Select UH Output Force Level |
| | | 0 Output Force Level is 'L' |

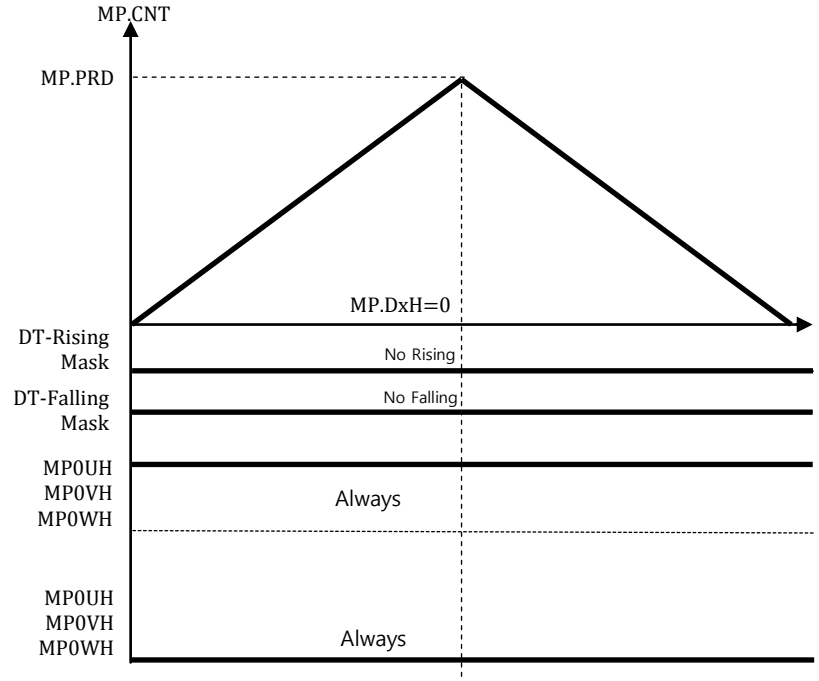



**Figure 16.16. H-side Always On ($T_{DUTY}$=Period: Dead-time Disabled)**

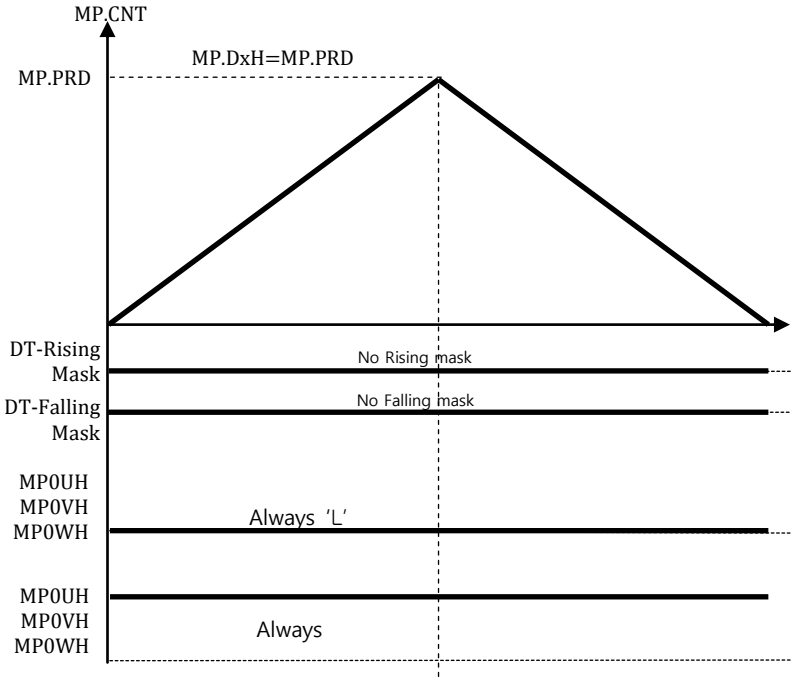



**Figure 16.17. L-side Always On ($T_{DUTY}$='0': Dead-time Disabled)**

## Symmetrical Mode vs Asymmetrical Mode

In symmetrical mode, the wave form is symmetrical on both sides of the mid-point of the period. The duty comparison is performed twice in both up and down count periods.

## ADn.SR ADC*n* Status Register

The ADC Status Register is a 32-bit register.

**AD0.SR=0x4000_B024, AD1.SR=0x4000_B124**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| EOC | ABUSY | DOVRUN | DMAIRQ | TRGIRQ | EOSIRQ | - | EOCIRQ |
| 0 | 0 | 0 | 0 | 0 | 0 | - | 0 |
| RO | RO | RO | RO | RC | RC | - | RC |

| | | |
|---|---|---|
| 7 | EOC | ADC End-of-Conversion flag<br>(Start-of-Conversion made by ADC_CLK clears this bit , not ASTART) |
| 6 | ABUSY | ADC conversion busy flag |
| 5 | DOVRUN | DMA overrun flag (not interrupt)<br>(DMA ACK didn't come until end of next conversion) |
| 4 | DMAIRQ | DMA done received (DMA transfer is completed) |
| 3 | TRGIRQ | ADC Trigger interrupt flag(Write "1" to clear flag)<br>(0: no int / 1: int occurred) |
| 2 | EOSIRQ | This flag will be set upon final end of a sequence (Write "1" to clear flag) |
| | | 0 None. |
| | | 1 End-of-Sequence(burst) Interrupt occurred |
| 0 | EOCIRQ | This flag will be set upon each conversion in a sequence is occurred(Write "1" to clear flag) |
| | | 0 None. |
| | | 1 End-of-Conversion Interrupt occurred |

## ADn.IER Interrupt Enable Register

**AD0.IER=0x4000_B028, AD1.IER=0x4000_B128**

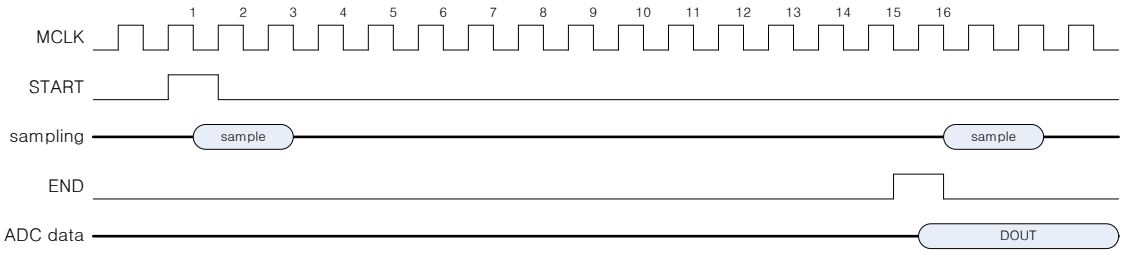
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| | | | DMAIRQE | TRGIRQE | EOSIRQE | - | EOCIRQE |
| | | | 0 | 0 | 0 | - | 0 |
| | | | RW | RW | RW | - | RW |

| | | |
|---|---|---|
| 4 | DMAIRQE | DMA done interrupt enable<br>0: interrupt disable<br>1: interrupt enable |
| 3 | TRGIRQE | ADC trigger conversion interrupt enable |
| 2 | EOSIRQE | ADC sequence conversion interrupt enable |
| 0 | EOCIRQE | ADC single conversion interrupt enable |

**General ADC Setup Procedure**

1. Allow the modification of the I/O pins to use the ADC inputs needed by writing the unlock sequence as described in PORT CONTROL UNIT (PCU), no pullups enabled.
2. Enable the ADC peripherals needed in PER2 register.
3. Enable the ADC peripheral clock in the PCER2 register.
4. Select the alternating function for the ADC inputs (Port n MUX registers).
5. Configure the ADC Pins to Analog
6. Configure the ADC mode in the ADCnMODE register and enabled the channel ADCn.
7. Configure the ADCnCR1 register and write an appropriate clock divider value.
8. Configure any special features such as triggers, sequencing, etc.
9. Start ADC conversion

## ADC Single Mode Timing Diagram




## ADC Sequencing Mode Timing Diagram