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	Active
Core Processor	ZNEO
Core Size	16-Bit
Speed	20MHz
Connectivity	EBI/EMI, I²C, IrDA, LINbus, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, DMA, POR, PWM, WDT
Number of I/O	76
Program Memory Size	64KB (64K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	4K x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 3.6V
Data Converters	A/D 12x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	100-LQFP
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/zilog/z16f6411al20ag

Table 17. External Interface Timing for a Read Operation, ISA Mode

		Delay (ns)	
Parameter	**Abbreviation**	**Minimum**	**Maximum**
T_1	X_{IN} Rise to Address Valid Delay		10
T_2	X_{IN} Rise to Address Output Hold Time	3	
T_3	Data Input Valid to X_{IN} Rise Setup Time		3
T_4	X_{IN} Rise to Data Input Hold Time	3	
T_5	X_{IN} Rise to $\overline{CS}$ Assertion Delay		10
T_6	X_{IN} Rise to $\overline{CS}$ Deassertion Hold Time	3	
T_7	X_{IN} Fall to $\overline{RD}$ Assertion Delay		10
T_8	X_{IN} Fall to $\overline{RD}$ Deassertion Hold Time	3	
T_9	$\overline{WAIT}$ Input Pin Assertion to X_{IN} Rise Setup Time	1	
T_{10}	$\overline{WAIT}$ Input Pin Deassertion to X_{IN} Rise Setup Time	1	
T_{11}	X_{IN} Rise to $\overline{DMAACK}$ Assertion Delay		10
T_{12}	X_{IN} Rise to $\overline{DMAACK}$ Deassertion Hold Time	3	
T_{13}	X_{IN} Rise to $\overline{BHEN}$ or $\overline{BLEN}$ Assertion Delay		10
T_{14}	X_{IN} Rise to $\overline{BHEN}$ or $\overline{BLEN}$ Deassertion Hold Time	3	

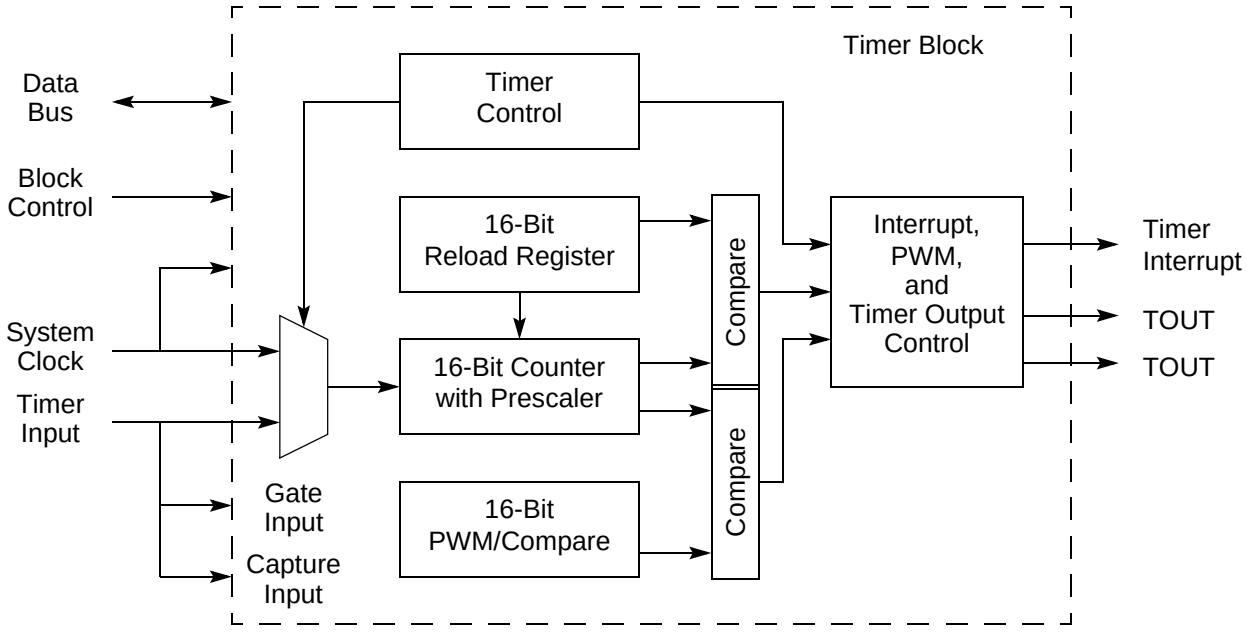

Figure 20. Timer Block Diagram

Operation

The general-purpose timer is a 16-bit up-counter. In normal operation, the timer is initialized to 0001h. When the timer is enabled, it counts up to the value contained in the Reload High and Low Byte registers, then resets to 0001h. The counter either halts or continues depending on the mode.

Minimum time-out delay (1 system clock) is set by loading the value 0001h into the Timer Reload High and Low byte registers and setting the prescale value to 1.

Maximum time-out delay ($2^{16} * 2^7$ system clocks) is set by loading the value 0000h into the Timer Reload High and Low byte registers and setting the prescale value to 128. When the timer reaches FFFFh, the timer rolls over to 0000h.

If the reload register is set to a value less than the current counter value, the counter continues counting until it reaches FFFFh and then resets to 0000h. Then the timer continues to count until it reaches the reload value and it resets to 0001h.

Note: When T0IN0, T0IN1 and T0IN2 functions are enabled on the PB0, PB1 and PB2 pins, each Timer 0 input will have the same effect as the single Timer 0 Input pin T0IN. For example, if the Timer 0 is in Capture Mode, any transitions on any of the PB0, PB1 and PB2 pins will cause a Capture.

PWM Reload High and Low Byte Registers

The PWM Reload High and Low Byte (PWMRH and PWMRL) registers, shown in Tables 66 and 67, store a 12-bit reload value, {PWMRH[3:0], PWMRL[7:0]}. The PWM reload value is held in buffer registers. The PWM reload value written to the buffer registers are not used by the PWM generator until the next PWM reload event occurs. Reads from these registers always return the values from the buffer registers.

$$\text{Edge-Aligned PWM Mode Period} = \frac{\text{Prescaler} \times \text{Reload Value}}{f_{PWMclk}}$$

$$\text{Center-Aligned PWM Mode Period} = \frac{2 \times \text{Prescaler} \times \text{Reload Value}}{f_{PWMclk}}$$

Table 66. PWM Reload High Byte Register (PWMRH)

Bits	7	6	5	4	3	2	1	0
Field	Reserved				PWMRH			
RESET	0h				Fh			
R/W	R/W				R/W			
Addr	FF_E38EH							

Bit	Description
[7:4]	**Reserved** These bits are reserved and must be programmed to 0000.
[3:0] PWMRH	**PWM Reload Register High and Low** These two bytes form the 12-bit reload value, {PWMRH[3:0], PWMRL[7:0]}. This value sets the PWM period.

Table 67. PWM Reload Low Byte Register (PWMRL)

Bits	7	6	5	4	3	2	1	0
Field	PWMRL							
RESET	FF							
R/W	R/W							
Addr	FF_E38Fh							

Bit	Description
[7:0] PWMRL	**PWM Reload Register High and Low** These two bytes form the 12-bit reload value, {PWMRH[3:0], PWMRL[7:0]}. This value sets the PWM period.

- Parity error (PE bit in Status 0 Register) is redefined as the Physical Layer Error (PLE) bit. The PLE bit indicates that receive data does not match transmit data when the LIN-UART is transmitting. This applies to both Master and Slave operating modes.
- The break detect interrupt (BRKD bit in Status 0 Register) indicates when a Break is detected by the slave (break condition for at least 11 bit times). Software uses this interrupt to start a timer checking for message frame time-out. The duration of the break is read in the RxBreakLength[3:0] field of the Mode Status Register.
- The break detect interrupt (BRKD bit in Status 0 Register) indicates when a wake-up message has been received if the LIN-UART is in a LINSLEEP state.
- In LIN Slave Mode, if the BRG counter overflows while measuring the autobaud period (Start bit to beginning of bit 7 of autobaud character) an overrun error is indicated (OE bit in the Status 0 Register). In this case, software sets the LinState field back to `10b`, where the slave ignores the current message and waits for the next Break signal. The baud reload high and low registers are not updated by hardware if this autobaud error occurs. The OE bit is also set if a data overrun error occurs.

LIN System Clock Requirements

The LIN master provides the timing reference for the LIN network and is required to have a clock source with a tolerance of ±0.5%. A slave with autobaud capability is required to have a baud clock matching the master oscillator within ±14%. The slave nodes autobaud to lock onto the master timing reference with an accuracy of ±2%. If a slave does not contain autobaud capability, it must include a baud clock which deviates from the masters by no more than ±1.5%. These accuracy requirements must include effects such as voltage and temperature drift during operation.

Before sending or receiving messages, the baud reload High/Low registers must be initialized. Unlike standard UART modes, the baud reload High/Low registers must be loaded with the baud interval rather than 1/16 of the baud interval.

In order to autobaud with the required accuracy, the LIN slave system clock must be at least 100 times the baud rate.

LIN Mode Initialization and Operation

A LIN protocol mode is selected by setting either the LIN master (LMST) or LIN slave (LSLV) and optionally (for LIN slave) the autobaud enable (ABEN) bits in the LIN Control Register. To access the LIN Control Register, the mode select (MSEL) field of the LIN-UART Mode Select/Status Register must be `010b`. The LIN-UART Control 0 Register must be initialized with TEN = 1, REN = 1, all other bits = 0.

In addition to the LMST, LSLV and ABEN bits in the LIN Control Register, a LinState[1:0] field exists that defines the current state of the LIN logic. This field is initially

Transmitter Interrupts

The transmitter generates a single interrupt when the Transmit Data Register empty bit (TDRE) is set to 1. This indicates that the transmitter is ready to accept new data for transmission. The TDRE interrupt occurs when the transmitter is initially enabled and after the transmit shift register has shifted the first bit of a character out. At this point, the Transmit Data Register is written with the next character to send. This provides 7 bit periods of latency to load the Transmit Data Register before the transmit shift register completes shifting the current character. Writing to the LIN-UART Transmit Data Register clears the TDRE bit to 0.

Receiver Interrupts

The receiver generates an interrupt when any of the following occurs:

- A data byte is received and is available in the LIN-UART Receive Data Register. This interrupt is disabled independent of the other receiver interrupt sources using the RDAIRQ bit (this feature is useful in devices, which support DMA). The received data interrupt occurs after the receive character is placed in the Receive Data Register. To avoid an overrun error, the software responds to this received data available condition before the next character is completely received.

Note: In Multiprocessor Mode (MPEN = 1), the receive data interrupts are dependent on the multiprocessor configuration and the most recent address byte.

- A break is received.
- A receive data overrun or LIN slave autobaud overrun error is detected.
- A data framing error is detected.
- A parity error is detected (physical layer error in LIN Mode).

LIN-UART Overrun Errors

When an overrun error condition occurs, the LIN-UART prevents overwriting of the valid data currently in the Receive Data Register. The break detect and overrun status bits are not displayed until the valid data is read.

When the valid data is read, the OE bit of the Status 0 Register is updated to indicate the overrun condition (and Break Detect, ifapplicable). The RDA bit is set to 1 to indicate that the Receive Data Register contains a data byte. However, because the overrun error occurred, this byte may not contain valid data and must be ignored. The BRKD bit indicates if the overrun is caused due to a break condition on the line. After reading the status

Table 96. LIN-UART Baud Rates (Continued)

5.5296 MHz System Clock				3.579545 MHz System Clock			
Desired Rate (kHz)	BRG Divisor (Decimal)	Actual Rate (kHz)	Error (%)	Desired Rate (kHz)	BRG Divisor (Decimal)	Actual Rate (kHz)	Error (%)
1250.0	N/A	N/A	N/A	1250.0	N/A	N/A	N/A
625.0	N/A	N/A	N/A	625.0	N/A	N/A	N/A
250.0	1	345.6	38.24	250.0	1	223.72	–10.51
115.2	3	115.2	0.00	115.2	2	111.9	–2.90
57.6	6	57.6	0.00	57.6	4	55.9	–2.90
38.4	9	38.4	0.00	38.4	6	37.3	–2.90
19.2	18	19.2	0.00	19.2	12	18.6	–2.90
9.60	36	9.60	0.00	9.60	23	9.73	1.32
4.80	72	4.80	0.00	4.80	47	4.76	–0.83
2.40	144	2.40	0.00	2.40	93	2.41	0.23
1.20	288	1.20	0.00	1.20	186	1.20	0.23
0.60	576	0.60	0.00	0.60	373	0.60	–0.04
0.30	1152	0.30	0.00	0.30	746	0.30	–0.04

Transfer Format with Phase Equals Zero

Figure 36 displays the timing diagram for an SPI type transfer in which PHASE = 0. For SPI transfers the clock only toggles during the character transfer. The two SCK waveforms show polarity with CLKPOL = 0 and with CLKPOL = 1. The diagram is interpreted as either a Master or Slave timing diagram as the SCK MISO and MOSI pins are directly connected between the master and the slave.

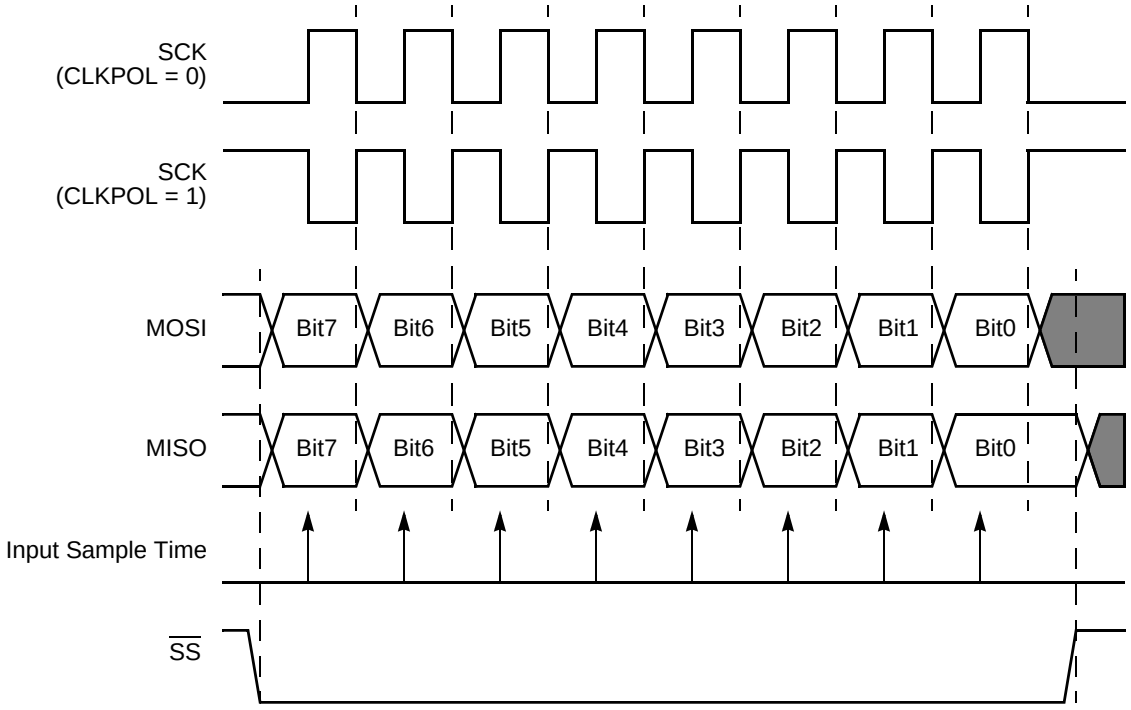

Figure 36. ESPI Timing when PHASE = 0

Transfer Format with Phase Equals One

Figure 37 displays the timing diagram for an SPI type transfer in which PHASE = 1. For SPI transfers the clock only toggles during the character transfer. Two waveforms are depicted for SCK, one for CLKPOL = 0 and another for CLKPOL = 1.

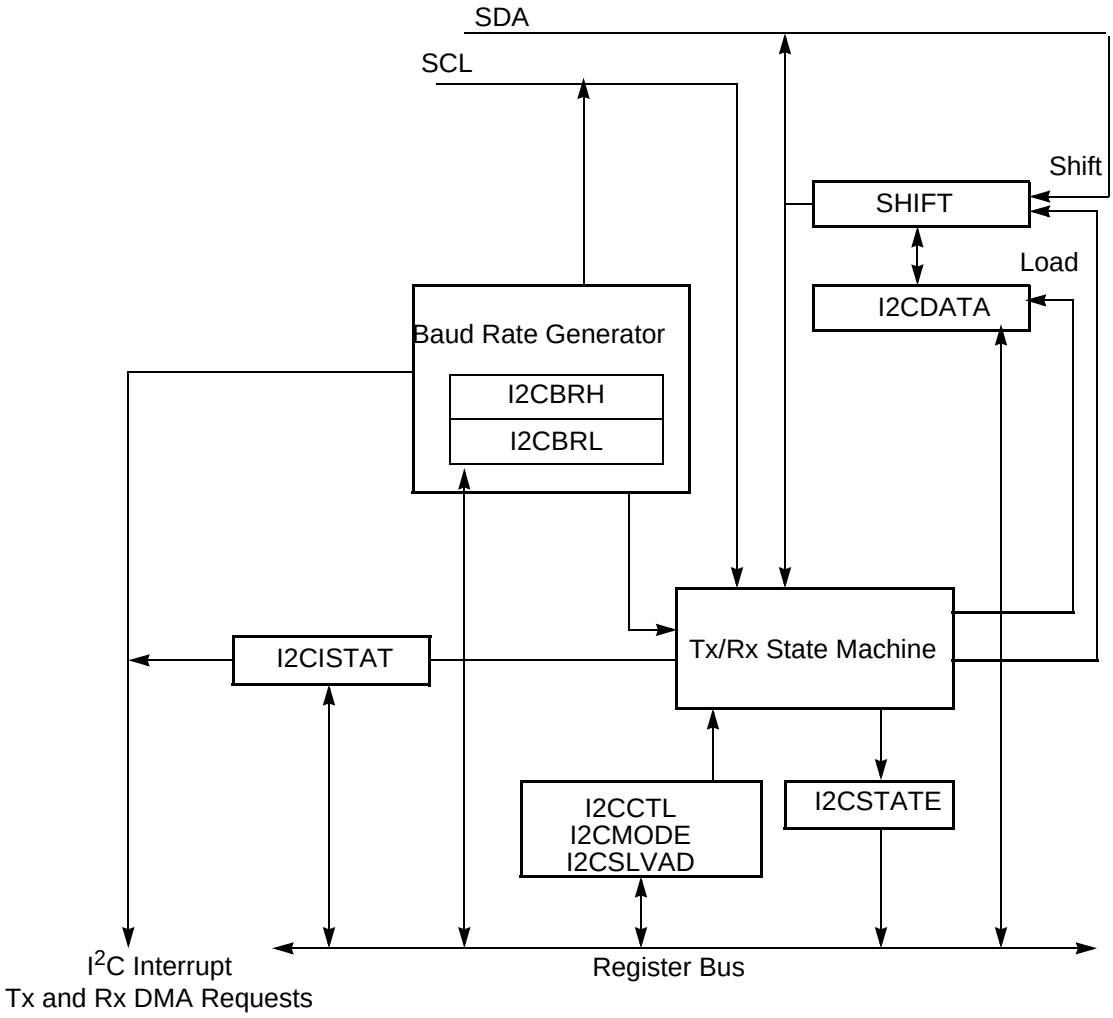

Figure 43. I2C Controller Block Diagram

6. When the first bit is shifted out, a Transmit interrupt asserts.
7. Software responds by writing the least significant eight bits of address to the I^2C Data Register.
8. The I^2C Controller completes shifting of the first address byte.
9. The I^2C Slave sends an acknowledge by pulling the SDA signal Low during the next High period of SCL.

   If the slave does not acknowledge the address byte, the I^2C Controller sets the NCKI bit in the I^2C Status Register, sets the ACKV bit and clears the ACK bit in the I^2C State Register. Software responds to the Not Acknowledge interrupt by setting the Stop bit and clearing the TXI bit. The I^2C Controller flushes the Transmit Data Register, sends the Stop condition on the bus and clears the Stop and NCKI bits. The transaction is complete (ignore the following steps).
10. The I^2C Controller loads the I^2C Shift Register with the contents of the I^2C Data Register (lower byte of 10 bit address).
11. The I^2C Controller shifts out the next eight bits of address. After the first bit shifts, the I^2C Controller generates a Transmit interrupt.
12. Software responds by setting the Start bit of the I^2C Control Register to generate a repeated Start.
13. Software responds by writing `11110b` followed by the 2-bit Slave address and a 1 (read) to the I^2C Data Register.
14. If you want to read only one byte, software responds by setting the NAK bit of the I^2C Control Register.
15. After the I^2C Controller shifts out the address bits mentioned in step 9 (second address transfer), the I^2C Slave sends an acknowledge by pulling the SDA signal Low during the next High period of SCL.

    If the slave does not acknowledge the address byte, the I^2C Controller sets the NCKI bit in the I^2C Status Register, sets the ACKV bit and clears the ACK bit in the I^2C State Register. Software responds to the Not Acknowledge interrupt by setting the Stop bit and clearing the TXI bit. The I^2C Controller flushes the Transmit Data Register, sends the Stop condition on the bus and clears the Stop and NCKI bits. The transaction is complete (ignore the following steps).
16. The I^2C Controller sends the repeated Start condition.
17. The I^2C Controller loads the I^2C Shift Register with the contents of the I^2C Data Register (third address transfer).
18. The I^2C Controller sends `11110b` followed by the two most significant bits of the Slave read address and a 1 (read).

Bit	Description (Continued)
[5] STOP	**Send Stop Condition** When set, this bit causes the I2C Controller (when configured as the Master) to send the Stop condition after the byte in the I2C Shift Register has completed transmission or after a byte has been received in a receive operation. When set, this bit is reset by the I2C Controller after a Stop condition has been sent or by deasserting the IEN bit. If this bit is 1, it cannot be cleared to 0 by writing to the register. If Stop is set while a Slave Mode transaction is underway, the Stop bit will be cleared by hardware.
[4] BIRQ	**Baud Rate Generator Interrupt Request** This bit is ignored when the I2C Controller is enabled. If this bit is set = 1 when the I2C Controller is disabled (IEN = 0) the baud rate generator is used as an additional timer causing an interrupt to occur every time the baud rate generator counts down to 1. The baud rate generator runs continuously in this Mode, generating periodic interrupts.
[3] TXI	**Enable TDRE Interrupts** This bit enables interrupts when the I2C Data Register is empty.
[2] NAK	**Send NAK** Setting this bit sends a Not Acknowledge condition after the next byte of data has been received. It is automatically deasserted after the Not Acknowledge is sent or the IEN bit is cleared. If this bit is 1, it cannot be cleared to 0 by writing to the register.
[1] FLUSH	**Flush Data** Setting this bit clears the I2C Data Register and sets the TDRE bit to 1. This bit allows flushing of the I2C Data Register when an NAK condition is received after the next data byte has been written to the I2C Data Register. Reading this bit always returns 0.
[0] FILTEN	**I2C Signal Filter Enable** Setting this bit enables low-pass digital filters on the SDA and SCL input signals. This function provides the spike suppression filter required in I2C Fast Mode. These filters reject any input pulse with periods less than a full system clock cycle. The filters introduce a 3-system clock cycle latency on the inputs.

I2C Baud Rate High and Low Byte Registers

The I2C Baud Rate High and Low Byte registers, shown in Tables 115 and 116, combine to form a 16-bit reload value, BRG[15:0], for the I2C Baud Rate Generator. The baud rate High and Low Byte Registers must be programmed for the I2C baud rate in Slave Mode as well as in Master Mode. In Slave Mode, the baud rate value programmed must match the master's baud rate within ±25% for proper operation.

The I2C baud rate is calculated using the following equation.

$$\text{I2C Baud Rate (bps)} = \frac{\text{System Clock Frequency (Hz)}}{4 \times \text{BRG[15:0]}}$$

Bit	Description (Continued)
[1] SCLOUT	**Serial Clock Output** Current value of Serial Clock being output onto the bus. The actual values of the SCL and SDA signals on the I2C bus is observed via the GPIO Input Register.
[0] BUSY	**I2C Bus Busy** 0 = No activity on the I2C Bus. 1 = A transaction is underway on the I2C bus.

Table 118. I2C State Register (I2CSTATE), Description when DIAG = 1

Bits	7	6	5	4	3	2	1	0
Field	I2CSTATE_H				I2CSTATE_L			
RESET	0	0	0	0	0	0	0	0
R/W	R	R	R	R	R	R	R	R
Addr	FF–E245h							

Bit	Description
[7:4] I2CSTATE_H	**I2C State High** This field defines the current state of the I2C Controller. It is the most significant nibble of the internal state machine. Table 119 defines the states for this field.
[3:0] I2CSTATE_L	**I2C State Low** Least significant nibble of the I2C state machine. This field defines the substates for the states defined by I2CSTATE_H. Table 120 defines the values for this field.

Table 119. I2CSTATE_H

State Encoding	State Name	State Description
0000	Idle	I2C bus is idle or I2C Controller is disabled.
0001	Slave Start	I2C Controller has received a start condition.
0010	Slave Bystander	Address did not match–ignore remainder of transaction.
0011	Slave Wait	Waiting for Stop or Restart condition after sending a Not Acknowledge instruction.
0100	Master Stop2	Master completing Stop condition (SCL = 1, SDA = 1).
0101	Master Start/Restart	Master Mode sending Start condition (SCL = 1, SDA = 0).
0110	Master Stop1	Master initiating Stop condition (SCL = 1, SDA = 0).

Table 119. I2CSTATE_H (Continued)

State Encoding	State Name	State Description
0111	Master Wait	Master received a Not Acknowledge instruction, waiting for software to assert Stop or Start control bits.
1000	Slave Transmit Data	Nine substates, one for each data bit and one for the acknowledge.
1001	Slave Receive Data	Nine substates, one for each data bit and one for the acknowledge.
1010	Slave Receive Addr1	Slave Receiving first address byte (7 and 10 bit addressing) Nine substates, one for each address bit and one for the acknowledge.
1011	Slave Receive Addr2	Slave Receiving second address byte (10 bit addressing) Nine substates, one for each address bit and one for the acknowledge.
1100	Master Transmit Data	Nine substates, one for each data bit and one for the acknowledge.
1101	Master Receive Data	Nine substates, one for each data bit and one for the acknowledge.
1110	Master Transmit Addr1	Master sending first address byte (7- and 10-Bit Addressing) Nine substates, one for each address bit and one for the acknowledge.
1111	Master Transmit Addr2	Master sending second address byte (10-Bit Addressing) Nine substates, one for each address bit and one for the acknowledge.

Table 120. I2CSTATE_L

State I2CSTATE_H	Sub-State I2CSTATE_L	Sub-State Name	State Description
0000–0100	0000	—	There are no substates for these I2CSTATE_H values.
0110–0111	0000	—	There are no substates for these I2CSTATE_H values.

Watchdog Timer Register Definitions

Watchdog Timer Reload High and Low Byte Registers

The Watchdog Timer Reload High and Low Byte (WDTH, WDTL) registers, shown in Table 124 through Table 125) form the 16-bit reload value that is loaded into the WDT when a WDT instruction executes. The 16-bit reload value is {WDTH[7:0], WDTL[7:0]}. Writing to these registers following the unlock sequence sets the appropriate reload value. Reading from these registers returns the current WDT count value.

Caution: The 16-bit WDT Reload Value must not be set to a value less than 0004h.

Table 124. Watchdog Timer Reload High Byte Register (WDTH)

Bits	7	6	5	4	3	2	1	0
Field	WDTH							
RESET	0	0	0	0	0	1	0	0
R/W	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*
Addr	FF_E042h							

Note: R/W* = Read returns the current WDT count value. Write sets the appropriate Reload Value.

Bit	Description
[7:0] WDTH	**WDT Reload High Byte** Most significant byte (MSB), Bits[15:8], of the 16-bit WDT reload value.

Table 125. Watchdog Timer Reload Low Byte Register (WDTL)

Bits	7	6	5	4	3	2	1	0
Field	WDTL							
RESET	0	0	0	0	0	0	0	0
R/W	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*
Addr	FF_E043h							

Note: R/W* = Read returns the current WDT count value. Write sets the appropriate Reload Value.

Bit	Description
[7:0] WDTL	**WDT Reload Low Byte** Least significant byte (LSB), Bits[7:0], of the 16-bit WDT reload value.

Flash Control Register Definitions

Flash Command Register

The Flash Command Register, shown in Table 139, unlocks the Flash Controller for programming and erase operations. The Write-only Flash Command Register shares its address with the Read-only Flash Status Register.

Table 139. Flash Command Register (FCMD)

Bits	7	6	5	4	3	2	1	0
Field	FCMD							
RESET	XXH							
R/W	W							
Addr	FF_E060h							

Bit	Description
[7:0] FCMD	**Flash Command** 73h = First unlock command. 8Ch = Second unlock command. 95h = Page erase command. 63h = Mass erase command.

Note: *All other commands, or any commands out of sequence, lock the Flash Controller.

Buffer Closure

A DMA buffer closure is requested in two ways. The first is when the transfer length reaches zero. The second is when the DMA receives a request end of frame from the peripheral. When either of these cases occur, the DMA begins closure of the buffer.

Loop Mode Closure

If the LOOP bit is set then the current buffer descriptor is not modified. The DMAxLAR increments or a new LAR value is fetched from the descriptor.

EOF Closure

The DMAxEN bit is reset to 0. If the EOF bit is set, the CMDSTAT field is set with the status data from the peripheral. If the channel is in LINKED LIST Mode then the DMAxCTL word is written back to the CONTROL word of the descriptor. The DMAxLAR increments or is loaded with new LAR data from the descriptor if the TXFR bit is set.

Normal Closure

The DMAxEN bit is reset to 0. If the channel is in LINKED LIST Mode then the DMAxCTL word is written back to the CONTROL word of the descriptor. The DMAxLAR increments or is loaded with new LAR data from the descriptor if the TXFR bit is set.

DMA Modes

Each DMA channel operates in two modes, direct and linked list. Both modes use the DMA Channel registers. The only difference is in how they are loaded. In DIRECT Mode, the DMA Channel registers are directly loaded by software and when the transfer is complete, the DMA stops. In LINKED LIST Mode, the DMA will load its own registers from a descriptor list which is pointed to by the DMAxLAR Register. It then loads the next descriptor in the list and continues executing.

The descriptor Control/Status field and address bytes maintain the same format as the control and address registers in the DMA.

Direct Mode

DIRECT Mode only uses the registers in the DMA for operation. The software writes these registers directly to set up and enable the DMA. DIRECT Mode is entered by directly setting the DMAxEN bit in the DMAxCTL0 Register.

Figure 58 displays the DMA registers and how they point to the buffers allocated in memory.

- When the upper eight bits of the transfer length equal zero and the lower eight bits of the transfer length is equal to the DMAxLAR[23:16] and the DMA is in DIRECT Mode
- If a buffer has been terminated by a Request EOF

For additional information about interrupts, see the Interrupt Controller chapter on page 80.

DMA Request Select Register

The DMA Request Select Register, shown in Table 148, governs the state of the DMA Channel.

Table 148. DMA Select Register (DAMxREQSEL)

Bits	7	6	5	4	3	2	1	0
Field	CHANSTATE				REQSEL			
RESET	0	0	0	0	0	0	0	0
R/W	R	R	R	R	R/W	R/W	R/W	R/W
Addr	FFE400h, FFE401h, FFE402h, FFE403h							

Bit	Description
[7:4] CHANSTATE	**Channel State** 0000 = DMA Off 0001 = DIRECT Mode, Waiting for End of Frame signal 0010 = LINKED LIST Mode, Waiting for End of Frame signal 0011 = Reserved 0100 = DIRECT Mode, First byte transfer, send command 0101 = LINKED LIST Mode, First byte transfer, send command 0110 = DIRECT Mode, Transfer of buffer in progress 0111 = LINKED LIST Mode, Transfer of buffer in progress 1000 = DIRECT Mode, Close Descriptor 1001 = LINKED LIST Mode, New List 1010 = LINKED LIST Mode, Close Descriptor 1011-1111 = Reserved

Status Register

The Status Register (DBGSTAT), shown in Table 174, contains status information about the state of the UART.

Table 174. Status Register (DBGSTAT)

Bits	7	6	5	4	3	2	1	0
Field	RDRF	RXOV	RXFE	RXBRK	TDRE	TXCOL	RXBUSY	TXBUSY
RESET	0	0	0	0	1	0	0	0
R/W	R/W1C	R/W1C	R/W1C	R/W1C	R/W1S	R/W1C	R	R
Addr	FF_E085							

Bit	Description
[7] RDRF	**Receive Data Register Full** This bit reflects the status of the Receive Data register. When data is written to the Receive Data register, or data is transferred from the shift register to the Receive Data register, this bit is set to 1. When the Receive Data register is read, this bit is cleared to 0. This bit is also cleared to 0 by writing a one to this bit. 0 = Receive Data register is empty. 1 = Receive Data register is full.
[6] RXOV	**Receive Overrun** This bit is set when a Receive Overrun occurs. A Receive Overrun occurs when there is data in the Receive Data register and another byte is written to this register. 0 = Receive Overrun has not occurred 1 = Receive Overrun has occurred.
[5] RXFE	**Receive Framing Error** This bit is set when a Receive Framing error has been detected. This bit is cleared by writing a one to this bit. 0 = No Framing Error detected. 1 = Receive Framing Error detected.
[4] RXBRK	**Receive Break Detect** This bit is set when a Break condition has been detected. This occurs when 10 or more bits received are Low. This bit is cleared by writing a one to this bit. 0 = No Break detected. 1 = Break detected.
[3] TDRE	**Transmit Data Register Empty** This bit reflects the status of the Transmit Data register. When the Transmit Data register is written, this bit is cleared to 0. When data from the Transmit Data Register is read or transferred to the transmit shift register, this bit is set to 1. This bit is written to 1 to abort the transmission of data being held in the Transmit Data Register. 0 = Transmit Data register is full. 1 = Transmit Data register is empty.

SPI Master Mode Timing

Figure 77 and Table 197 provides timing information for SPI Master Mode pins. Timing is shown with SCK rising edge used to source MOSI output data, SCK falling edge used to sample MISO input data. Timing on the SS output pin(s) is controlled by software.

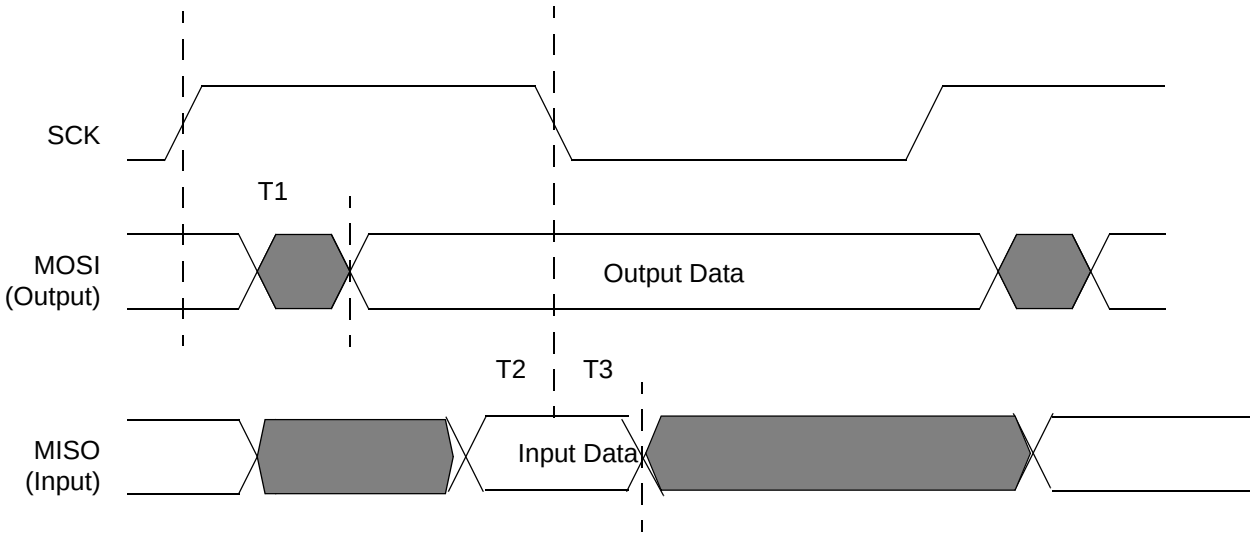

Figure 77. SPI Master Mode Timing

Table 197. SPI Master Mode Timing

		Delay (ns)	
Parameter	**Description**	**Min**	**Max**
SPI Master			
T_1	SCK Rise to MOSI output Valid Delay	–5	+5
T_2	MISO input to SCK (receive edge) Setup Time	20	
T_3	MISO input to SCK (receive edge) Hold Time	0	

SPI Slave Mode Timing

Figure 78 and Table 198 provide timing information for the SPI Slave Mode pins. Timing is shown with SCK rising edge used tosource MISO output data, SCK falling edge used to sample MOSI input data.