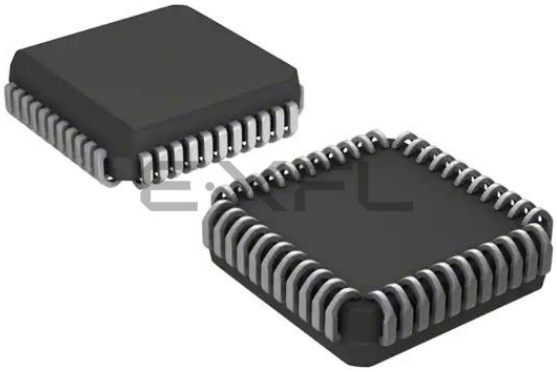

Welcome to E-XFL.COM

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	eZ8
Core Size	8-Bit
Speed	20MHz
Connectivity	I²C, IrDA, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, DMA, POR, PWM, WDT
Number of I/O	31
Program Memory Size	16KB (16K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	2K x 8
Voltage - Supply (Vcc/Vdd)	3V ~ 3.6V
Data Converters	A/D 8x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 105°C (TA)
Mounting Type	Surface Mount
Package / Case	44-LCC (J-Lead)
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/zilog/z8f1621vn020ec00tr

Braces

The curly braces, { }, indicate a single register or bus created by concatenating some combination of smaller registers, buses, or individual bits.

- Example: The 12-bit register address {0H, RP[7:4], R1[3:0]} is composed of a 4-bit hexadecimal value (0H) and two 4-bit register values taken from the Register Pointer (RP) and Working Register R1. 0H is the most-significant nibble (4-bit value) of the 12-bit register, and R1[3:0] is the least significant nibble of the 12-bit register.

Parentheses

The parentheses, ( ), indicate an indirect register address lookup.

- Example: (R1) is the memory location referenced by the address contained in the Working Register R1.

Parentheses/Bracket Combinations

The parentheses, ( ), indicate an indirect register address lookup and the square brackets, [ ], indicate a register or bus.

- Example: Assume PC[15:0] contains the value 1234h. (PC[15:0]) then refers to the contents of the memory location at address 1234h.

Use of the Words *Set*, *Reset* and *Clear*

The word *set* implies that a register bit or a condition contains a logical 1. The words re*set* or *clear* imply that a register bit or a condition contains a logical 0. When either of these terms is followed by a number, the word *logical* may not be included; however, it is implied.

Notation for Bits and Similar Registers

A field of bits within a register is designated as: Register[*n*:*n*].

- Example: ADDR[15:0] refers to bits 15 through bit 0 of the Address.

Use of the Terms *LSB*, *MSB*, *lsb*, and *msb*

In this document, the terms *LSB* and *MSB,* when appearing in upper case, mean *least significant byte* and *most significant byte*, respectively. The lowercase forms, *lsb* and *msb*, mean *least significant bit* and *most significant bit*, respectively.

Use of Initial Uppercase Letters

Initial uppercase letters designate settings and conditions in general text.

- Example 1: The receiver forces the SCL line to Low.
- Example 2: The Master can generate a Stop condition to abort the transfer.

UART0 Control 1
U0CTL1 (F43H - Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0

- Infrared Encoder/Decoder Enable
  0 = Infrared endec is disabled
  1 = Infrared endec is enabled
- Received Data Interrupt Enable
  0 = Received data and errors generate interrupt requests
  1 = Only errors generate interrupt requests. Received data does not.
- Baud Rate Registers Control
  Refer to UART chapter for operation
- Driver Enable Polarity
  0 = DE signal is active High
  1 = DE signal is active Low
- Multiprocessor Bit Transmit
  0 = Send a 0 as the multiprocessor bit
  1 = Send a 1 as the multiprocessor bit
- Multiprocessor Mode [0]
  See Multiprocessor Mode [1] below
- Multiprocessor (9-bit) Enable
  0 = Multiprocessor mode is disabled
  1 = Multiprocessor mode is enabled
- Multiprocessor Mode [1]
  with Multiprocess Mode bit 0:
  00 = Interrupt on all received bytes
  01 = Interrupt only on address bytes
  10 = Interrupt on address match and following data
  11 = Interrupt on data following an address match

UART0 Status 1
U0STAT1 (F44H - Read Only)

D7	D6	D5	D4	D3	D2	D1	D0

- Mulitprocessor Receive
  Returns value of last multiprocessor bit
- New Frame
  0 = Current byte is not start of frame
  1 = Current byte is start of new frame
- Reserved

UART0 Address Compare
U0ADDR (F45H - Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0

- UART0 Address Compare [7:0]

UART0 Baud Rate Generator High Byte
U0BRH (F46H - Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0

- UART0 Baud Rate divisor [15:8]

UART0 Baud Rate Generator Low Byte
U0BRL (F47H - Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0

- UART0 Baud Rate divisor [7:0]

UART1 Transmit Data
U1TXD (F48H - Write Only)

D7	D6	D5	D4	D3	D2	D1	D0

- UART1 transmitter data byte[7:0]

UART1 Receive Data
U1RXD (F48H - Read Only)

D7	D6	D5	D4	D3	D2	D1	D0

- UART receiver data byte [7:0]

Port C Control
PCCTL (FD9H - Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0

Port C Control[7:0]
Provides Access to Port Sub-Registers

Port C Input Data
PCIN (FDAH - Read Only)

D7	D6	D5	D4	D3	D2	D1	D0

Port C Input Data [7:0]

Port C Output Data
PCOUT (FDBH - Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0

Port C Output Data [7:0]

Port D Address
PDADDR (FDCH - Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0

Port D Address[7:0]
Selects Port Sub-Registers:
00H = No function
01H = Data direction
02H = Alternate function
03H = Output control (open-drain)
04H = High drive enable
05H = Stop Mode Recovery enable
06H-FFH = No function

Port D Control
PDCTL (FDDH - Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0

Port D Control[7:0]
Provides Access to Port Sub-Registers

Port D Input Data
PDIN (FDE H- Read Only)

D7	D6	D5	D4	D3	D2	D1	D0

Port D Input Data [7:0]

Port D Output Data
PDOUT (FDFH - Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0

Port D Output Data [7:0]

Port E Address
PEADDR (FE0H - Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0

Port E Address[7:0]
Selects Port Sub-Registers:
00H = No function
01H = Data direction
02H = Alternate function
03H = Output control (open-drain)
04H = High drive enable
05H = Stop Mode Recovery enable
06H-FFH = No function

Port E Control
PECTL (FE1H - Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0

Port E Control[7:0]
Provides Access to Port Sub-Registers

Port E Input Data
PEIN (FE2H - Read Only)

D7	D6	D5	D4	D3	D2	D1	D0

Port E Input Data [7:0]

Port E Output Data
PEOUT (FE3H - Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0

Port E Output Data [7:0]

Port F Address
PFADDR (FE4H - Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0

Port F Address[7:0]
Selects Port Sub-Registers:
00H = No function
01H = Data direction
02H = Alternate function
03H = Output control (open-drain)
04H = High drive enable
05H = Stop Mode Recovery enable
06H-FFH = No function

Caution: *The following style of coding to generate software interrupts by setting bits in the Interrupt Request registers is NOT recommended. All incoming interrupts that are received between execution of the first LDX command and the last LDX command are lost.*

Poor coding style that can result in lost interrupt requests:

```
LDX r0, IRQ0
OR r0, MASK
LDX IRQ0, r0
```

To avoid missing interrupts, the following style of coding to set bits in the Interrupt Request registers is recommended:

Good coding style that avoids lost interrupt requests:

```
ORX IRQ0, MASK
```

Interrupt Control Register Definitions

For all interrupts other than the Watchdog Timer interrupt, the interrupt control registers enable individual interrupts, set interrupt priorities, and indicate interrupt requests.

Interrupt Request 0 Register

The Interrupt Request 0 (IRQ0) register (Table 24) stores the interrupt requests for both vectored and polled interrupts. When a request is presented to the interrupt controller, the corresponding bit in the IRQ0 register becomes 1. If interrupts are globally enabled (vectored interrupts), the interrupt controller passes an interrupt request to the eZ8™ CPU. If interrupts are globally disabled (polled interrupts), the eZ8 CPU can read the Interrupt Request 0 register to determine if any interrupt requests are pending

Table 24. Interrupt Request 0 Register (IRQ0)

BITS	7	6	5	4	3	2	1	0
FIELD	T2I	T1I	T0I	U0RXI	U0TXI	I2CI	SPII	ADCI
RESET	0							
R/W	R/W							
ADDR	FC0H							

T2I—Timer 2 Interrupt Request
0 = No interrupt request is pending for Timer 2.
1 = An interrupt request from Timer 2 is awaiting service.

Table 29. IRQ0 Enable Low Bit Register (IRQ0ENL)

BITS	7	6	5	4	3	2	1	0
FIELD	T2ENL	T1ENL	T0ENL	U0RENL	U0TENL	I2CENL	SPIENL	ADCENL
RESET	0							
R/W	R/W							
ADDR	FC2H							

T2ENL—Timer 2 Interrupt Request Enable Low Bit
T1ENL—Timer 1 Interrupt Request Enable Low Bit
T0ENL—Timer 0 Interrupt Request Enable Low Bit
U0RENL—UART 0 Receive Interrupt Request Enable Low Bit
U0TENL—UART 0 Transmit Interrupt Request Enable Low Bit
I2CENL—I^2C Interrupt Request Enable Low Bit
SPIENL—SPI Interrupt Request Enable Low Bit
ADCENL—ADC Interrupt Request Enable Low Bit

IRQ1 Enable High and Low Bit Registers

The IRQ1 Enable High and Low Bit registers (see Table 31 and Table 32 on page 76) form a priority encoded enabling for interrupts in the Interrupt Request 1 register. Priority is generated by setting bits in each register. Table 30 describes the priority control for IRQ1.

Table 30. IRQ1 Enable and Priority Encoding

IRQ1ENH[*x*]	IRQ1ENL[*x*]	Priority	Description
0	0	Disabled	Disabled
0	1	Level 1	Low
1	0	Level 2	Nominal
1	1	Level 3	High

Note: where *x* indicates the register bits from 0 through 7.

2. Write to the Timer High and Low Byte registers to set the starting count value (usually 0001H), affecting only the first pass in CONTINUOUS mode. After the first timer Reload in CONTINUOUS mode, counting always begins at the reset value of 0001H.
3. Write to the Timer Reload High and Low Byte registers to set the Reload value.
4. If desired, enable the timer interrupt and set the timer interrupt priority by writing to the relevant interrupt registers.
5. If using the Timer Output function, configure the associated GPIO port pin for the Timer Output alternate function.
6. Write to the Timer Control 1 register to enable the timer and initiate counting.

In CONTINUOUS mode, the system clock always provides the timer input. The timer period is given by the following equation:

$$\text{CONTINUOUS Mode Time-Out Period (s)} = \frac{\text{Reload Value} \times \text{Prescale}}{\text{System Clock Frequency (Hz)}}$$

If an initial starting value other than 0001H is loaded into the Timer High and Low Byte registers, the ONE-SHOT mode equation must be used to determine the first time-out period.

COUNTER Mode

In COUNTER mode, the timer counts input transitions from a GPIO port pin. The timer input is taken from the GPIO Port pin Timer Input alternate function. The TPOL bit in the Timer Control 1 Register selects whether the count occurs on the rising edge or the falling edge of the Timer Input signal. In COUNTER mode, the prescaler is disabled.

Caution: *The input frequency of the Timer Input signal must not exceed one-fourth the system clock frequency.*

Upon reaching the Reload value stored in the Timer Reload High and Low Byte registers, the timer generates an interrupt, the count value in the Timer High and Low Byte registers is reset to 0001H and counting resumes. Also, if the Timer Output alternate function is enabled, the Timer Output pin changes state (from Low to High or from High to Low) at timer Reload.

Follow the steps below for configuring a timer for COUNTER mode and initiating the count:

1. Write to the Timer Control 1 register to:
   – Disable the timer
   – Configure the timer for COUNTER mode

Follow the steps below for configuring a timer for COMPARE mode and initiating the count:

1. Write to the Timer Control 1 register to:
   – Disable the timer
   – Configure the timer for COMPARE mode
   – Set the prescale value
   – Set the initial logic level (High or Low) for the Timer Output alternate function, if desired
2. Write to the Timer High and Low Byte registers to set the starting count value.
3. Write to the Timer Reload High and Low Byte registers to set the Compare value.
4. If desired, enable the timer interrupt and set the timer interrupt priority by writing to the relevant interrupt registers.
5. If using the Timer Output function, configure the associated GPIO port pin for the Timer Output alternate function.
6. Write to the Timer Control 1 register to enable the timer and initiate counting.

In COMPARE mode, the system clock always provides the timer input. The Compare time is given by the following equation:

$$\text{COMPARE Mode Time (s)} = \frac{(\text{Compare Value} - \text{Start Value}) \times \text{Prescale}}{\text{System Clock Frequency (Hz)}}$$

GATED Mode

In GATED mode, the timer counts only when the Timer Input signal is in its active state (asserted), as determined by the TPOL bit in the Timer Control 1 register. When the Timer Input signal is asserted, counting begins. A timer interrupt is generated when the Timer Input signal is deasserted or a timer reload occurs. To determine if a Timer Input signal deassertion generated the interrupt, read the associated GPIO input value and compare to the value stored in the TPOL bit.

The timer counts up to the 16-bit Reload value stored in the Timer Reload High and Low Byte registers. The timer input is the system clock. When reaching the Reload value, the timer generates an interrupt, the count value in the Timer High and Low Byte registers is reset to 0001H and counting resumes (assuming the Timer Input signal is still asserted). Also, if the Timer Output alternate function is enabled, the Timer Output pin changes state (from Low to High or from High to Low) at timer reset.

Follow the steps below for configuring a timer for GATED mode and initiating the count:

1. Write to the Timer Control 1 register to:
   – Disable the timer
   – Configure the timer for GATED mode

If interrupts are enabled, following completion of the Stop Mode Recovery the eZ8 CPU responds to the interrupt request by fetching the Watchdog Timer interrupt vector and executing code from the vector address.

WDT Reset in Normal Operation

If configured to generate a Reset when a time-out occurs, the Watchdog Timer forces the device into the Reset state. The WDT status bit in the Watchdog Timer Control register is set to 1. For more information on Reset, see Reset and Stop Mode Recovery on page 47.

WDT Reset in STOP Mode

If enabled in STOP mode and configured to generate a Reset when a time-out occurs and the device is in STOP mode, the Watchdog Timer initiates a Stop Mode Recovery. Both the WDT status bit and the STOP bit in the Watchdog Timer Control register are set to 1 following WDT time-out in STOP mode. Default operation is for the WDT and its RC oscillator to be enabled during STOP mode.

WDT RC Disable in STOP Mode

To minimize power consumption in STOP Mode, the WDT and its RC oscillator can be disabled in STOP mode. The following sequence configures the WDT to be disabled when the 64K Series devices enter STOP Mode following execution of a STOP instruction:

1. Write 55H to the Watchdog Timer Control register (WDTCTL).
2. Write AAH to the Watchdog Timer Control register (WDTCTL).
3. Write 81H to the Watchdog Timer Control register (WDTCTL) to configure the WDT and its oscillator to be disabled during STOP Mode. Alternatively, write 00H to the Watchdog Timer Control register (WDTCTL) as the third step in this sequence to reconfigure the WDT and its oscillator to be enabled during STOP mode.

This sequence only affects WDT operation in STOP mode.

Watchdog Timer Reload Unlock Sequence

Writing the unlock sequence to the Watchdog Timer (WDTCTL) Control register address unlocks the three Watchdog Timer Reload Byte registers (WDTU, WDTH, and WDTL) to allow changes to the time-out period. These write operations to the WDTCTL register address produce no effect on the bits in the WDTCTL register. The locking mechanism prevents spurious writes to the Reload registers. Follow the steps below to unlock the Watchdog Timer Reload Byte registers (WDTU, WDTH, and WDTL) for write access.

1. Write 55H to the Watchdog Timer Control register (WDTCTL).
2. Write AAH to the Watchdog Timer Control register (WDTCTL).
3. Write the Watchdog Timer Reload Upper Byte register (WDTU).
4. Write the Watchdog Timer Reload High Byte register (WDTH).

when a byte is written to the UART Transmit Data register. The Driver Enable signal asserts at least one UART bit period and no greater than two UART bit periods before the Start bit is transmitted. This timing allows a setup time to enable the transceiver. The Driver Enable signal deasserts one system clock period after the last Stop bit is transmitted. This one system clock delay allows both time for data to clear the transceiver before disabling it, as well as the ability to determine if another character follows the current character. In the event of back to back characters (new data must be written to the Transmit Data Register before the previous character is completely transmitted) the DE signal is not deasserted between characters. The DEPOL bit in the UART Control Register 1 sets the polarity of the Driver Enable signal.

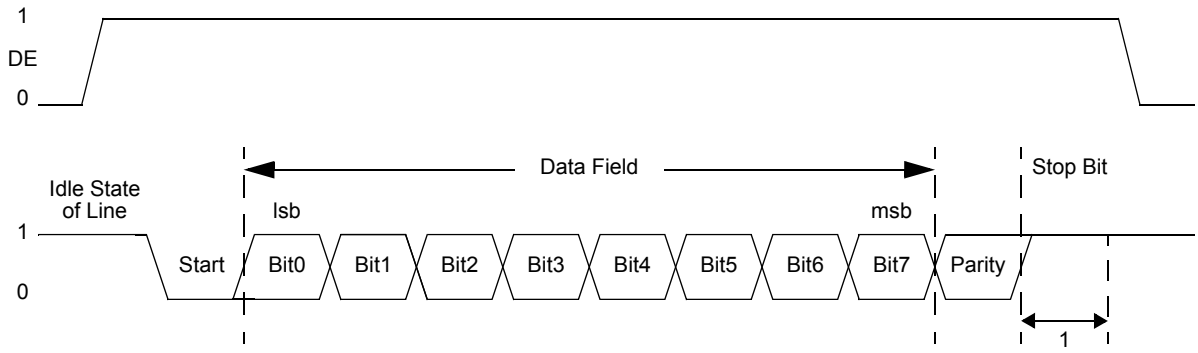

Figure 17. UART Driver Enable Signal Timing (shown with 1 Stop Bit and Parity)

The Driver Enable to Start bit setup time is calculated as follows:

$$\left(\frac{1}{\text{Baud Rate (Hz)}}\right) \leq \text{DE to Start Bit Setup Time (s)} \leq \left(\frac{2}{\text{Baud Rate (Hz)}}\right)$$

UART Interrupts

The UART features separate interrupts for the transmitter and the receiver. In addition, when the UART primary functionality is disabled, the Baud Rate Generator can also function as a basic timer with interrupt capability.

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 after the Transmit shift register has shifted the first bit of data out. At this point, the Transmit Data register may be 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 UART Transmit Data register clears the TDRE bit to 0.

PE—Parity Error
This bit indicates that a parity error has occurred. Reading the UART Receive Data register clears this bit.
0 = No parity error occurred.
1 = A parity error occurred.

OE—Overrun Error
This bit indicates that an overrun error has occurred. An overrun occurs when new data is received and the UART Receive Data register has not been read. If the RDA bit is reset to 0, then reading the UART Receive Data register clears this bit.
0 = No overrun error occurred.
1 = An overrun error occurred.

FE—Framing Error
This bit indicates that a framing error (no Stop bit following data reception) was detected. Reading the UART Receive Data register clears this bit.
0 = No framing error occurred.
1 = A framing error occurred.

BRKD—Break Detect
This bit indicates that a break occurred. If the data bits, parity/multiprocessor bit, and Stop bit(s) are all zeros then this bit is set to 1. Reading the UART Receive Data register clears this bit.
0 = No break occurred.
1 = A break occurred.

TDRE—Transmitter Data Register Empty
This bit indicates that the UART Transmit Data register is empty and ready for additional data. Writing to the UART Transmit Data register resets this bit.
0 = Do not write to the UART Transmit Data register.
1 = The UART Transmit Data register is ready to receive an additional byte to be transmitted.

TXE—Transmitter Empty
This bit indicates that the transmit shift register is empty and character transmission is finished.
0 = Data is currently transmitting.
1 = Transmission is complete.

CTS—$\overline{\text{CTS}}$ signal
When this bit is read it returns the level of the $\overline{\text{CTS}}$ signal.

UART Status 1 Register

This register contains multiprocessor control and status bits.

01 = The UART generates an interrupt request only on received address bytes.
10 = The UART generates an interrupt request when a received address byte matches the value stored in the Address Compare Register and on all successive data bytes until an address mismatch occurs.
11 = The UART generates an interrupt request on all received data bytes for which the most recent address byte matched the value in the Address Compare Register.

MPEN—MULTIPROCESSOR (9-bit) Enable
This bit is used to enable MULTIPROCESSOR (9-bit) mode.
0 = Disable MULTIPROCESSOR (9-bit) mode.
1 = Enable MULTIPROCESSOR (9-bit) mode.

MPBT—MULTIPROCESSOR Bit Transmit
This bit is applicable only when MULTIPROCESSOR (9-bit) mode is enabled.
0 = Send a 0 in the multiprocessor bit location of the data stream (9^{th} bit).
1 = Send a 1 in the multiprocessor bit location of the data stream (9^{th} bit).

DEPOL—Driver Enable Polarity
0 = DE signal is Active High.
1 = DE signal is Active Low.

BRGCTL—Baud Rate Control
This bit causes different UART behavior depending on whether the UART receiver is enabled (REN = 1 in the UART Control 0 Register).

When the UART receiver is not enabled, this bit determines whether the Baud Rate Generator issues interrupts.
0 = Reads from the Baud Rate High and Low Byte registers return the BRG Reload Value
1 = The Baud Rate Generator generates a receive interrupt when it counts down to 0. Reads from the Baud Rate High and Low Byte registers return the current BRG count value.

When the UART receiver is enabled, this bit allows reads from the Baud Rate Registers to return the BRG count value instead of the Reload Value.
0 = Reads from the Baud Rate High and Low Byte registers return the BRG Reload Value.
1 = Reads from the Baud Rate High and Low Byte registers return the current BRG count value. Unlike the Timers, there is no mechanism to latch the High Byte when the Low Byte is read.

$\overline{\text{RDAIRQ}}$—Receive Data Interrupt $\overline{\text{Enable}}$
0 = Received data and receiver errors generates an interrupt request to the Interrupt Controller.
1 = Received data does not generate an interrupt request to the Interrupt Controller. Only receiver errors generate an interrupt request.

IREN—Infrared Encoder/Decoder Enable
0 = Infrared Encoder/Decoder is disabled. UART operates normally operation.

The first seven bits transmitted in the first byte are 11110XX. The two bits XX are the two most-significant bits of the 10-bit address. The lowest bit of the first byte transferred is the read/write control bit (=0). The transmit operation is carried out in the same manner as 7-bit addressing.

Follow the steps below for a transmit operation on a 10-bit addressed slave:

1. Software asserts the IEN bit in the I^2C Control register.
2. Software asserts the TXI bit of the I^2C Control register to enable Transmit interrupts.
3. The I^2C interrupt asserts because the I^2C Data register is empty.
4. Software responds to the TDRE interrupt by writing the first slave address byte to the I^2C Data register. The least-significant bit must be 0 for the write operation.
5. Software asserts the START bit of the I^2C Control register.
6. The I^2C Controller sends the START condition to the I^2C slave.
7. The I^2C Controller loads the I^2C Shift register with the contents of the I^2C Data register.
8. After one bit of address is shifted out by the SDA signal, the Transmit interrupt is asserted.
9. Software responds by writing the second byte of address into the contents of the I^2C Data register.
10. The I^2C Controller shifts the rest of the first byte of address and write bit out the SDA signal.
11. If the I^2C slave acknowledges the first address byte by pulling the SDA signal low during the next high period of SCL, the I^2C Controller sets the ACK bit in the I^2C Status register. Continue with step 12.

    If the slave does not acknowledge the first address byte, the I^2C Controller sets the NCKI bit and clears the ACK bit in the I^2C Status register. Software responds to the Not Acknowledge interrupt by setting the STOP and FLUSH bits and clearing the TXI bit. The I^2C Controller sends the STOP condition on the bus and clears the STOP and NCKI bits. The transaction is complete (ignore the following steps).
12. The I^2C Controller loads the I^2C Shift register with the contents of the I^2C Data register.
13. The I^2C Controller shifts the second address byte out the SDA signal. After the first bit has been sent, the Transmit interrupt is asserted.
14. Software responds by writing a data byte to the I^2C Data register.
15. The I^2C Controller completes shifting the contents of the shift register on the SDA signal.

Caution: *Software must be cautious in making decisions based on this bit within a transaction because software cannot tell when the bit is updated by hardware. In the case of write transactions, the I²C pauses at the beginning of the Acknowledge cycle if the next transmit data or address byte has not been written (TDRE = 1) and STOP and START = 0. In this case the ACK bit is not updated until the transmit interrupt is serviced and the Acknowledge cycle for the previous byte completes. For examples of how the ACK bit can be used, see* Address Only Transaction with a 7-bit Address *on page 148* and Address Only Transaction with a 10-bit Address *on page 150*.

10B—10-Bit Address
This bit indicates whether a 10- or 7-bit address is being transmitted. After the START bit is set, if the five most-significant bits of the address are 11110B, this bit is set. When set, it is reset once the first byte of the address has been sent.

RD—Read
This bit indicates the direction of transfer of the data. It is active high during a read. The status of this bit is determined by the least-significant bit of the I²C Shift register after the START bit is set.

TAS—Transmit Address State
This bit is active high while the address is being shifted out of the I²C Shift register.

DSS—Data Shift State
This bit is active high while data is being shifted to or from the I²C Shift register.

NCKI—NACK Interrupt
This bit is set high when a Not Acknowledge condition is received or sent and neither the START nor the STOP bit is active. When set, this bit generates an interrupt that can only be cleared by setting the START or STOP bit, allowing you to specify whether to perform a STOP or a repeated START.

I²C Control Register

The I²C Control register (Table 72) enables the I²C operation.

Table 72. I²C Control Register (I2CCTL)

BITS	7	6	5	4	3	2	1	0
FIELD	IEN	START	STOP	BIRQ	TXI	NAK	FLUSH	FILTEN
RESET	0							
R/W	R/W	R/W1	R/W1	R/W	R/W	R/W1	W1	R/W
ADDR	F52H							

DMA*x*_IO[7:0]}. When the DMA is configured for two-byte word transfers, the DMA*x* I/O Address register must contain an even numbered address.

Table 78. DMA*x* I/O Address Register (DMA*x*IO)

BITS	7	6	5	4	3	2	1	0
FIELD	DMA_IO							
RESET	X							
R/W	R/W							
ADDR	FB1H, FB9H							

DMA_IO—DMA on-chip peripheral control register address
This byte sets the low byte of the on-chip peripheral control register address on Register File Page FH (addresses F00H to FFFH).

DMA*x* Address High Nibble Register

The DMA*x* Address High register (Table 79) specifies the upper four bits of address for the Start/Current and End Addresses of DMA*x*.

Table 79. DMA*x* Address High Nibble Register (DMA*x*H)

BITS	7	6	5	4	3	2	1	0
FIELD	DMA_END_H				DMA_START_H			
RESET	X							
R/W	R/W							
ADDR	FB2H, FBAH							

DMA_END_H—DMA*x* End Address High Nibble
These bits, used with the DMA*x* End Address Low register, form a 12-bit End Address. The full 12-bit address is given by {DMA_END_H[3:0], DMA_END[7:0]}.

DMA_START_H—DMA*x* Start/Current Address High Nibble
These bits, used with the DMA*x* Start/Current Address Low register, form a 12-bit Start/Current Address. The full 12-bit address is given by {DMA_START_H[3:0], DMA_START[7:0]}.

0101 = ADC Analog Inputs 0-5 updated.
0110 = ADC Analog Inputs 0-6 updated.
0111 = ADC Analog Inputs 0-7 updated.
1000 = ADC Analog Inputs 0-8 updated.
1001 = ADC Analog Inputs 0-9 updated.
1010 = ADC Analog Inputs 0-10 updated.
1011 = ADC Analog Inputs 0-11 updated.
1100-1111 = Reserved.

DMA Status Register

The DMA Status register (Table 85 on page 173) indicates the DMA channel that generated the interrupt and the ADC Analog Input that is currently undergoing conversion. Reads from this register reset the Interrupt Request Indicator bits (IRQA, IRQ1, and IRQ0) to 0. Therefore, software interrupt service routines that read this register must process all three interrupt sources from the DMA.

Table 85. DMA_ADC Status Register (DMAA_STAT)

BITS	7	6	5	4	3	2	1	0
FIELD	CADC[3:0]				Reserved	IRQA	IRQ1	IRQ0
RESET	0							
R/W	R							
ADDR	FBFH							

CADC[3:0]—Current ADC Analog Input
This field identifies the Analog Input that the ADC is currently converting.

Reserved
This bit is reserved and must be 0.

IRQA—DMA_ADC Interrupt Request Indicator
This bit is automatically reset to 0 each time a read from this register occurs.
0 = DMA_ADC is not the source of the interrupt from the DMA Controller.
1 = DMA_ADC completed transfer of data from the last ADC Analog Input and generated an interrupt.

IRQ1—DMA1 Interrupt Request Indicator
This bit is automatically reset to 0 each time a read from this register occurs.
0 = DMA1 is not the source of the interrupt from the DMA Controller.
1 = DMA1 completed transfer of data to/from the End Address and generated an interrupt.

IRQ0—DMA0 Interrupt Request Indicator
This bit is automatically reset to 0 each time a read from this register occurs.

On-Chip Oscillator

Overview

The products in the 64K Series feature an on-chip oscillator for use with external crystals with frequencies from 32 kHz to 20 MHz. In addition, the oscillator can support external RC networks with oscillation frequencies up to 4 MHz or ceramic resonators with oscillation frequencies up to 20 MHz. This oscillator generates the primary system clock for the internal eZ8™ CPU and the majority of the on-chip peripherals. Alternatively, the X_{IN} input pin can also accept a CMOS-level clock input signal (32 kHz–20 MHz). If an external clock generator is used, the X_{OUT} pin must be left unconnected.

When configured for use with crystal oscillators or external clock drivers, the frequency of the signal on the X_{IN} input pin determines the frequency of the system clock (that is, no internal clock divider). In RC operation, the system clock is driven by a clock divider (divide by 2) to ensure 50% duty cycle.

Operating Modes

The 64K Series products support four different oscillator modes:

- On-chip oscillator configured for use with external RC networks (<4 MHz).
- Minimum power for use with very low frequency crystals (32 kHz to 1.0 MHz).
- Medium power for use with medium frequency crystals or ceramic resonators (0.5 MHz to 10.0 MHz).
- Maximum power for use with high frequency crystals or ceramic resonators (8.0 MHz to 20.0 MHz).

The oscillator mode is selected through user-programmable Option Bits. For more information, see Option Bits on page 195.

Crystal Oscillator Operation

Figure 40 on page 212 displays a recommended configuration for connection with an external fundamental-mode, parallel-resonant crystal operating at 20 MHz. Recommended 20 MHz crystal specifications are provided in Table 104 on page 212. Resistor R1 is optional and limits total power dissipation by the crystal. The printed circuit board layout

Figure 43 displays the typical active mode current consumption while operating at 25 ºC versus the system clock frequency. All GPIO pins are configured as outputs and driven High.

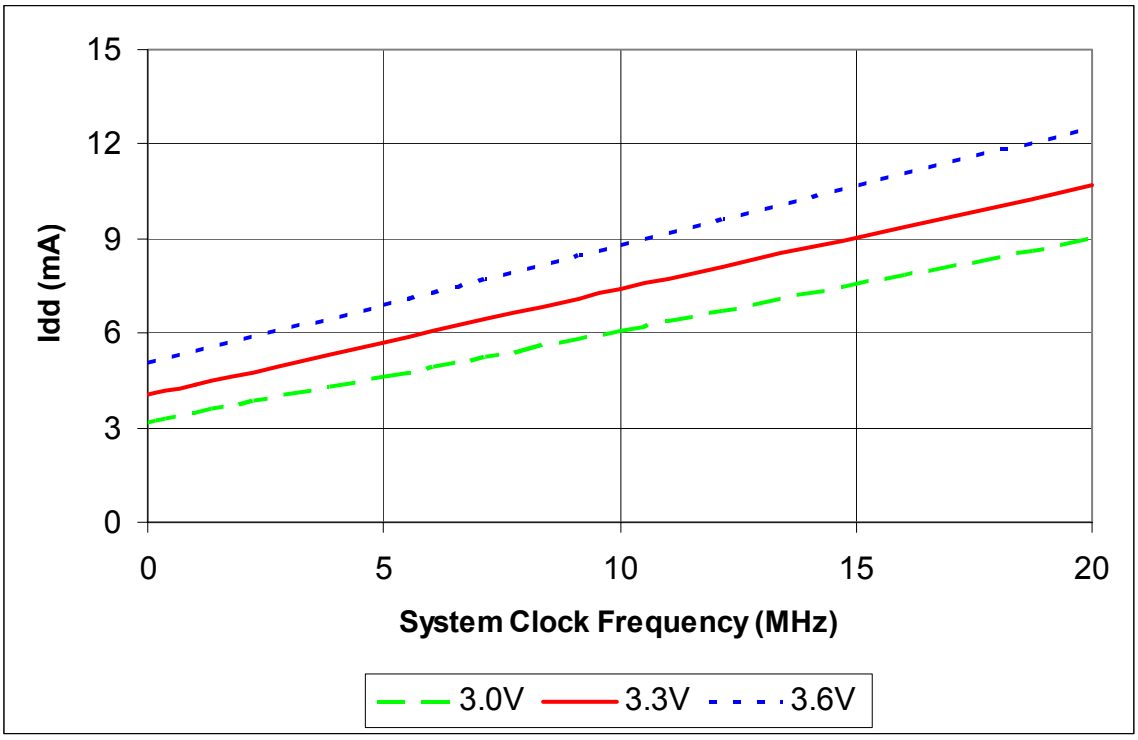

Figure 43. Typical Active Mode Idd Versus System Clock Frequency

Table 133. eZ8 CPU Instruction Summary (Continued)

Assembly Mnemonic	Symbolic Operation	Address Mode dst	Address Mode src	Opcode(s) (Hex)	Flags C	Z	S	V	D	H	Fetch Cycles	Instr. Cycles
EI	IRQCTL[7] ← 1			9F	-	-	-	-	-	-	1	2
HALT	HALT Mode			7F	-	-	-	-	-	-	1	2
INC dst	dst ← dst + 1	R		20	-	*	*	*	-	-	2	2
		IR		21							2	3
		r		0E-FE							1	2
INCW dst	dst ← dst + 1	RR		A0	-	*	*	*	-	-	2	5
		IRR		A1							2	6
IRET	FLAGS ← @SP SP ← SP + 1 PC ← @SP SP ← SP + 2 IRQCTL[7] ← 1			BF	*	*	*	*	*	*	1	5
JP dst	PC ← dst	DA		8D	-	-	-	-	-	-	3	2
		IRR		C4							2	3
JP cc, dst	if cc is true PC ← dst	DA		0D-FD	-	-	-	-	-	-	3	2
JR dst	PC ← PC + X	DA		8B	-	-	-	-	-	-	2	2
JR cc, dst	if cc is true PC ← PC + X	DA		0B-FB	-	-	-	-	-	-	2	2
LD dst, rc	dst ← src	r	IM	0C-FC	-	-	-	-	-	-	2	2
		r	X(r)	C7							3	3
		X(r)	r	D7							3	4
		r	Ir	E3							2	3
		R	R	E4							3	2
		R	IR	E5							3	4
		R	IM	E6							3	2
		IR	IM	E7							3	3
		Ir	r	F3							2	3
		IR	R	F5							3	3

Index

Symbols

Numerics

A

B

C