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/z8f1621vn020ec

Z8 Encore! XP® 64K Series Flash Microcontrollers
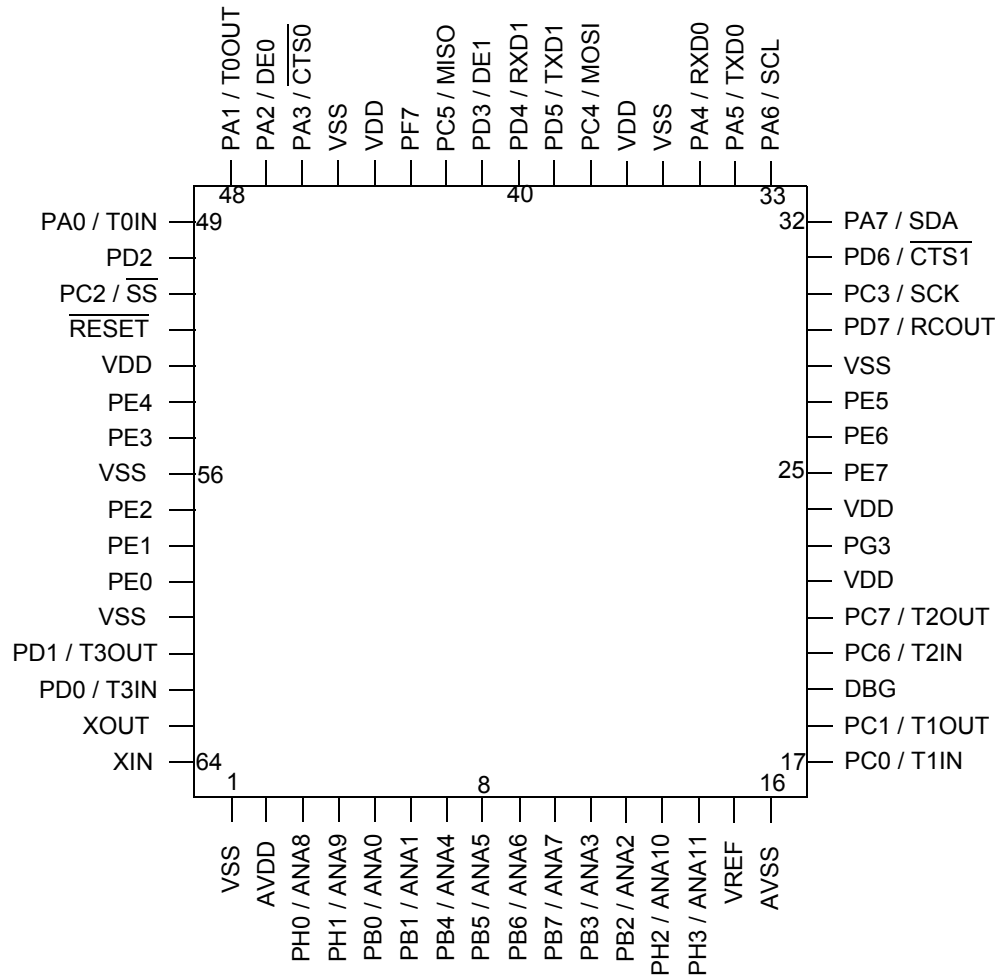**Product Specification**

zilog 11

Figure 5. Z8 Encore! XP 64K Series Flash Microcontrollers in 64-Pin Low-Profile Quad Flat Package (LQFP)

Z8 Encore! XP® 64K Series Flash Microcontrollers
Product Specification

13
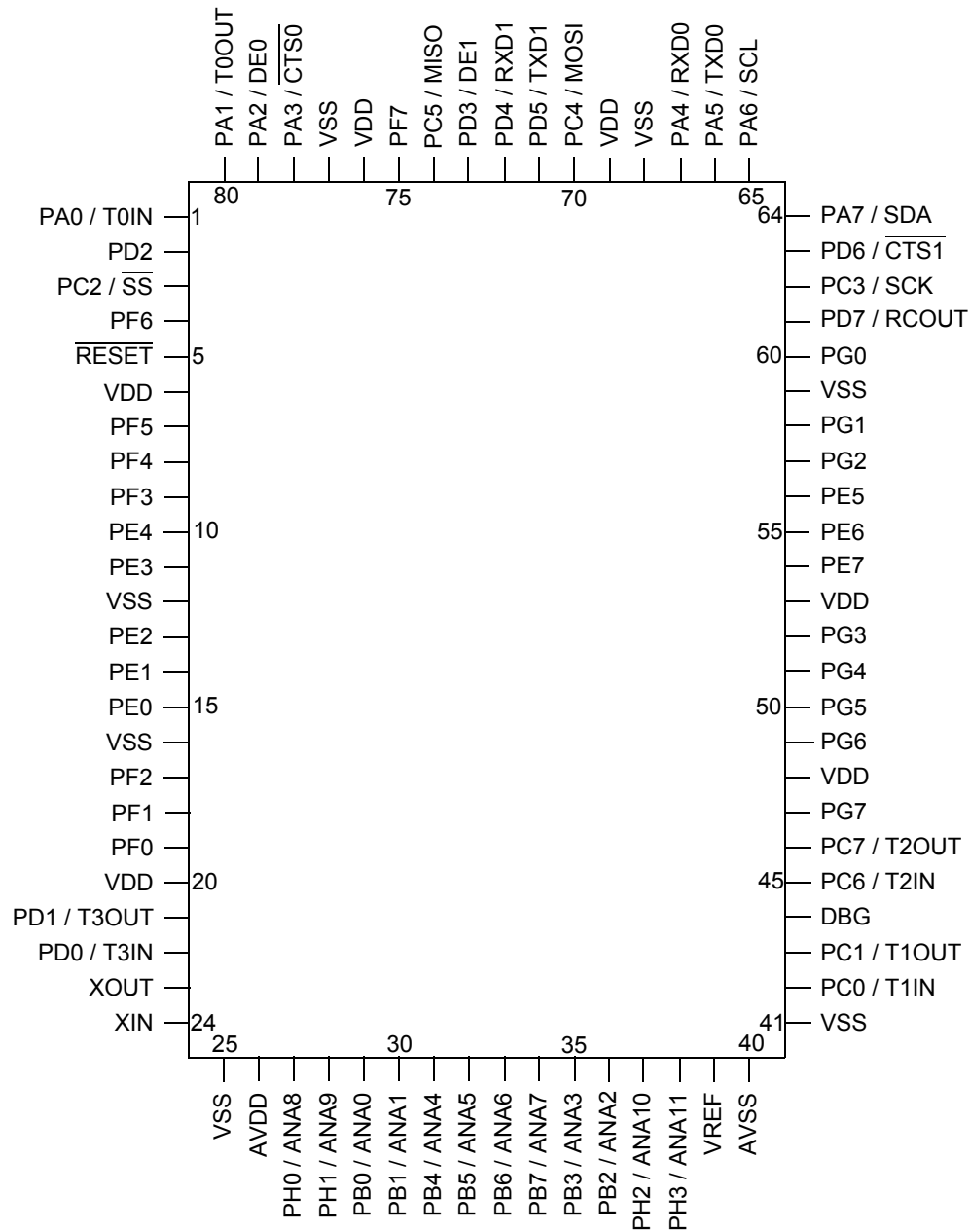
Figure 7. Z8 Encore! XP 64K Series Flash Microcontrollers in 80-Pin Quad Flat Package (QFP)

Z8 Encore! XP® 64K Series Flash Microcontrollers
Product Specification

28

Control Register Summary

Timer 0 High Byte
T0H   (F00H - Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0

└─── Timer 0 current count value [15:8]

Timer 0 Low Byte
T0L   (F01H - Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0

└─── Timer 0 current count value [7:0]

Timer 0 Reload High Byte
T0RH   (F02H - Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0

└─── Timer 0 reload value [15:8]

Timer 0 Reload Low Byte
T0RL   (HF03 - Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0

└─── Timer 0 reload value [7:0]

Timer 0 PWM High Byte
T0PWMH   (F04H - Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0

└─── Timer 0 PWM value [15:8]

Timer 0 Control 0
T0CTL0   (F06H - Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0

└─── Reserved

Cascade Timer
0 = Timer 0 Input signal is GPIO pin
1 = Timer 0 Input signal is Timer 3 out

Reserved

Timer 0 Control 1
T0CTL1   (F07H - Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0

Timer Mode
000 = One-Shot mode
001 = CONTINUOUS mode
010 = COUNTER mode
011 = PWM mode
100 = CAPTURE mode
101 = COMPARE mode
110 = GATED mode
111 = Capture/COMPARE mode

Prescale Value
000 = Divide by 1
001 = Divide by 2
010 = Divide by 4
011 = Divide by 8
100 = Divide by 16
101 = Divide by 32
110 = Divide by 64
111 = Divide by 128

Timer Input/Output Polarity
Operation of this bit is a function of the current operating mode of the timer

Timer Enable
0 = Timer is disabled
1 = Timer is enabled

Timer 1 High Byte
T1H   (F08H - Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0

└─── Timer 1 current count value [15:8]

Timer 1 Low Byte
T1L   (F09H - Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0

└─── Timer 1 current count value [7:0]

Timer 1 Reload High Byte
T1RH   (F0AH - Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0

└─── Timer 1 reload value [15:8]

Timer 1 Reload Low Byte
T1RL   (F0BH - Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0

└─── Timer 1 reload value [7:0]

Z8 Encore! XP® 64K Series Flash Microcontrollers
Product Specification

43

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

Z8 Encore! XP® 64K Series Flash Microcontrollers
Product Specification

61

GPIO Control Register Definitions

Four registers for each Port provide access to GPIO control, input data, and output data. Table 13 lists these Port registers. Use the Port A–H Address and Control registers together to provide access to sub-registers for Port configuration and control.

Table 13. GPIO Port Registers and Sub-Registers

Port Register Mnemonic	Port Register Name
PxADDR	Port A–H Address Register (Selects sub-registers)
PxCTL	Port A–H Control Register (Provides access to sub-registers)
PxIN	Port A–H Input Data Register
PxOUT	Port A–H Output Data Register

Port Sub-Register Mnemonic	Port Register Name
PxDD	Data Direction
PxAF	Alternate Function
PxOC	Output Control (Open-Drain)
PxDD	High Drive Enable
PxSMRE	Stop Mode Recovery Source Enable

Port A–H Address Registers

The Port A–H Address registers select the GPIO Port functionality accessible through the Port A–H Control registers. The Port A–H Address and Control registers combine to provide access to all GPIO Port control (Table 14).

Table 14. Port A–H GPIO Address Registers (PxADDR)

BITS	7	6	5	4	3	2	1	0
FIELD	PADDR[7:0]							
RESET	00H							
R/W	R/W							
ADDR	FD0H, FD4H, FD8H, FDCH, FE0H, FE4H, FE8H, FECH							

Z8 Encore! XP® 64K Series Flash Microcontrollers
Product Specification

zilog

108

Receiving Data using the Interrupt-Driven Method

The UART Receiver interrupt indicates the availability of new data (as well as error conditions). Follow the steps below to configure the UART receiver for interrupt-driven operation:

1. Write to the UART Baud Rate High and Low Byte registers to set the desired baud rate.

2. Enable the UART pin functions by configuring the associated GPIO Port pins for alternate function operation.

3. Execute a DI instruction to disable interrupts.

4. Write to the Interrupt control registers to enable the UART Receiver interrupt and set the desired priority.

5. Clear the UART Receiver interrupt in the applicable Interrupt Request register.

6. Write to the UART Control 1 Register to enable MULTIPROCESSOR (9-bit) mode functions, if desired.
   – Set the MULTIPROCESSOR Mode Select (MPEN) to Enable MULTIPROCESSOR mode.
   – Set the MULTIPROCESSOR Mode Bits, MPMD[1:0], to select the desired address matching scheme.
   – Configure the UART to interrupt on received data and errors or errors only (interrupt on errors only is unlikely to be useful for Z8 Encore! devices without a DMA block).

7. Write the device address to the Address Compare Register (automatic multiprocessor modes only).

8. Write to the UART Control 0 register to:
   – Set the receive enable bit (REN) to enable the UART for data reception.
   – Enable parity, if desired and if MULTIPROCESSOR mode is not enabled, and select either even or odd parity.

9. Execute an EI instruction to enable interrupts.

The UART is now configured for interrupt-driven data reception. When the UART Receiver interrupt is detected, the associated interrupt service routine performs the following:

1. Check the UART Status 0 register to determine the source of the interrupt - error, break, or received data.

2. If the interrupt was caused by data available, read the data from the UART Receive Data register. If operating in MULTIPROCESSOR (9-bit) mode, further actions may be required depending on the MULTIPROCESSOR Mode bits MPMD[1:0].

Z8 Encore! XP® 64K Series Flash Microcontrollers
Product Specification

131

During an SPI transfer, data is sent and received simultaneously by both the Master and the Slave SPI devices. Separate signals are required for data and the serial clock. When an SPI transfer occurs, a multi-bit (typically 8-bit) character is shifted out one data pin and an multi-bit character is simultaneously shifted in on a second data pin. An 8-bit shift register in the Master and another 8-bit shift register in the Slave are connected as a circular buffer. The SPI shift register is single-buffered in the transmit and receive directions. New data to be transmitted cannot be written into the shift register until the previous transmission is complete and receive data (if valid) has been read.

SPI Signals

The four basic SPI signals are:

- Master-In/Slave-Out
- Master-Out/Slave-In
- Serial Clock
- Slave Select

Each signal is described in both Master and Slave modes.

Master-In/Slave-Out

The Master-In/Slave-Out (MISO) pin is configured as an input in a Master device and as an output in a Slave device. It is one of the two lines that transfer serial data, with the most significant bit sent first. The MISO pin of a Slave device is placed in a high-impedance state if the Slave is not selected. When the SPI is not enabled, this signal is in a high-impedance state.

Master-Out/Slave-In

The Master-Out/Slave-In (MOSI) pin is configured as an output in a Master device and as an input in a Slave device. It is one of the two lines that transfer serial data, with the most significant bit sent first. When the SPI is not enabled, this signal is in a high-impedance state.

Serial Clock

The Serial Clock (SCK) synchronizes data movement both in and out of the device through its MOSI and MISO pins. In MASTER mode, the SPI's Baud Rate Generator creates the serial clock. The Master drives the serial clock out its own SCK pin to the Slave's SCK pin. When the SPI is configured as a Slave, the SCK pin is an input and the clock signal from the Master synchronizes the data transfer between the Master and Slave devices. Slave devices ignore the SCK signal, unless the $\overline{SS}$ pin is asserted. When configured as a slave, the SPI block requires a minimum SCK period of greater than or equal to 8 times the system (XIN) clock period.

**Z8 Encore! XP® 64K Series Flash Microcontrollers
Product Specification**

zilog | **133**

Transfer Format PHASE Equals Zero

Figure 25 displays the timing diagram for an SPI transfer in which PHASE is cleared to 0. The two SCK waveforms show polarity with CLKPOL reset to 0 and with CLKPOL set to one. The diagram may be interpreted as either a Master or Slave timing diagram because the SCK Master-In/Slave-Out (MISO) and Master-Out/Slave-In (MOSI) pins are directly connected between the Master and the Slave.
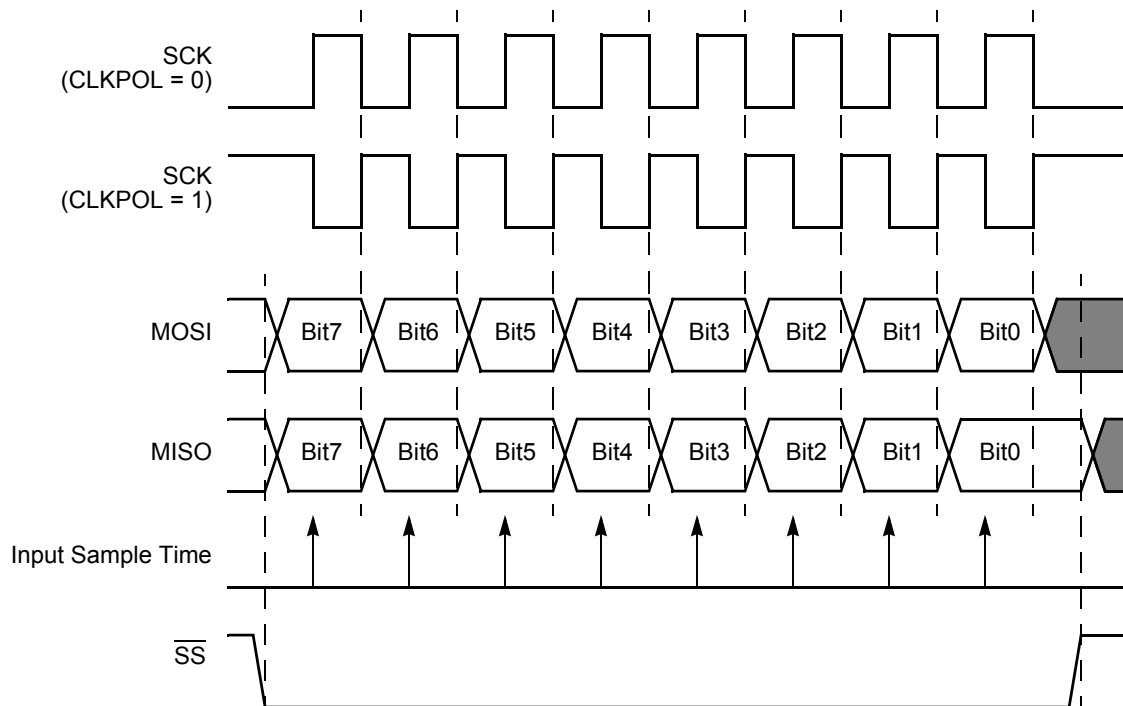
Figure 25. SPI Timing When PHASE is 0

Transfer Format PHASE Equals One

Figure 26 on page 134 displays the timing diagram for an SPI transfer in which PHASE is one. Two waveforms are depicted for SCK, one for CLKPOL reset to 0 and another for CLKPOL set to 1.

Z8 Encore! XP® 64K Series Flash Microcontrollers
Product Specification

142

SPI Baud Rate High and Low Byte Registers

The SPI Baud Rate High and Low Byte registers (Table 68 and Table 69) combine to form a 16-bit reload value, BRG[15:0], for the SPI Baud Rate Generator.

When configured as a general purpose timer, the SPI BRG interrupt interval is calculated using the following equation:

$$\text{SPI BRG Interrupt Interval (s)} = \text{System Clock Period (s)} \times \text{BRG[15:0]}$$

Table 68. SPI Baud Rate High Byte Register (SPIBRH)

BITS	7	6	5	4	3	2	1	0
FIELD	BRH							
RESET	1							
R/W	R/W							
ADDR	F66H							

BRH = SPI Baud Rate High Byte
Most significant byte, BRG[15:8], of the SPI Baud Rate Generator's reload value.

Table 69. SPI Baud Rate Low Byte Register (SPIBRL)

BITS	7	6	5	4	3	2	1	0
FIELD	BRL							
RESET	1							
R/W	R/W							
ADDR	F67H							

BRL = SPI Baud Rate Low Byte
Least significant byte, BRG[7:0], of the SPI Baud Rate Generator's reload value.

Z8 Encore! XP® 64K Series Flash Microcontrollers
Product Specification

zilog

150

13. The I²C Controller shifts the data out of using the SDA signal. After the first bit is sent, the Transmit interrupt is asserted.

14. If more bytes remain to be sent, return to step 9.

15. Software responds by setting the STOP bit of the I²C Control register (or START bit to initiate a new transaction). In the STOP case, software clears the TXI bit of the I²C Control register at the same time.

16. The I²C Controller completes transmission of the data on the SDA signal.

17. The slave may either Acknowledge or Not Acknowledge the last byte. Because either the STOP or START bit is already set, the NCKI interrupt does not occur.

18. The I²C Controller sends the STOP (or RESTART) condition to the I²C bus. The STOP or START bit is cleared.

Address Only Transaction with a 10-bit Address

In the situation where software wants to determine if a slave with a 10-bit address is responding without sending or receiving data, a transaction can be done which only consists of an address phase. Figure 30 displays this 'address only' transaction to determine if a slave with 10-bit address will acknowledge. As an example, this transaction can be used after a 'write' has been done to a EEPROM to determine when the EEPROM completes its internal write operation and is once again responding to I²C transactions. If the slave does not Acknowledge the transaction can be repeated until the slave is able to Acknowledge.

S	Slave Address 1st 7 bits	W = 0	A/$\overline{A}$	Slave Address 2nd Byte	A/$\overline{A}$	P

Figure 30. 10-Bit Address Only Transaction Format

Follow the steps below for an address only transaction to a 10-bit addressed slave:

1. Software asserts the IEN bit in the I²C Control register.

2. Software asserts the TXI bit of the I²C Control register to enable Transmit interrupts.

3. The I²C interrupt asserts, because the I²C Data register is empty (TDRE = 1)

4. Software responds to the TDRE interrupt by writing the first slave address byte. The least-significant bit must be 0 for the write operation.

5. Software asserts the START bit of the I²C Control register.

6. The I²C Controller sends the START condition to the I²C slave.

Z8 Encore! XP® 64K Series Flash Microcontrollers
Product Specification

zilog

152

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^{2}C Control register.

2. Software asserts the TXI bit of the I^{2}C Control register to enable Transmit interrupts.

3. The I^{2}C interrupt asserts because the I^{2}C Data register is empty.

4. Software responds to the TDRE interrupt by writing the first slave address byte to the I^{2}C Data register. The least-significant bit must be 0 for the write operation.

5. Software asserts the START bit of the I^{2}C Control register.

6. The I^{2}C Controller sends the START condition to the I^{2}C slave.

7. The I^{2}C Controller loads the I^{2}C Shift register with the contents of the I^{2}C 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^{2}C Data register.

10. The I^{2}C Controller shifts the rest of the first byte of address and write bit out the SDA signal.

11. If the I^{2}C slave acknowledges the first address byte by pulling the SDA signal low during the next high period of SCL, the I^{2}C Controller sets the ACK bit in the I^{2}C Status register. Continue with step 12.

    If the slave does not acknowledge the first address byte, the I^{2}C Controller sets the NCKI bit and clears the ACK bit in the I^{2}C Status register. Software responds to the Not Acknowledge interrupt by setting the STOP and FLUSH bits and clearing the TXI bit. The I^{2}C 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^{2}C Controller loads the I^{2}C Shift register with the contents of the I^{2}C Data register.

13. The I^{2}C 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^{2}C Data register.

15. The I^{2}C Controller completes shifting the contents of the shift register on the SDA signal.

Z8 Encore! XP® 64K Series Flash Microcontrollers
Product Specification

zilog

154

4. The I²C Controller sends the START condition.

5. The I²C Controller shifts the address and read bit out the SDA signal.

6. If the I²C slave acknowledges the address by pulling the SDA signal Low during the next high period of SCL, the I²C Controller sets the ACK bit in the I²C Status register. Continue with step 7.

    If the slave does not acknowledge, the Not Acknowledge interrupt occurs (NCKI bit is set in the Status register, ACK bit is cleared). Software responds to the Not Acknowledge interrupt by setting the STOP bit and clearing the TXI bit. The I²C Controller sends the STOP condition on the bus and clears the STOP and NCKI bits. The transaction is complete (ignore the following steps).

7. The I²C Controller shifts in the byte of data from the I²C slave on the SDA signal. The I²C Controller sends a Not Acknowledge to the I²C slave if the NAK bit is set (last byte), else it sends an Acknowledge.

8. The I²C Controller asserts the Receive interrupt (RDRF bit set in the Status register).

9. Software responds by reading the I²C Data register which clears the RDRF bit. If there is only one more byte to receive, set the NAK bit of the I²C Control register.

10. If there are more bytes to transfer, return to step 7.

11. After the last byte is shifted in, a Not Acknowledge interrupt is generated by the I²C Controller.

12. Software responds by setting the STOP bit of the I²C Control register.

13. A STOP condition is sent to the I²C slave, the STOP and NCKI bits are cleared.

Read Transaction with a 10-Bit Address

Figure 33 displays the read transaction format for a 10-bit addressed slave. The shaded regions indicate data transferred from the I²C Controller to slaves and unshaded regions indicate data transferred from the slaves to the I²C Controller.

S	Slave Address 1st 7 bits	W=0	A	Slave Address 2nd Byte	A	S	Slave Address 1st 7 bits	R=1	A	Data	A	Data	$\overline{A}$	P

Figure 33. Receive Data Format for a 10-Bit Addressed Slave

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 write control bit.

Z8 Encore! XP® 64K Series Flash Microcontrollers
Product Specification

166

Configuring DMA0 and DMA1 for Data Transfer

Follow the steps below to configure and enable DMA0 or DMA1:

1. Write to the DMA*x* I/O Address register to set the Register File address identifying the on-chip peripheral control register. The upper nibble of the 12-bit address for on-chip peripheral control registers is always FH. The full address is {FH, DMA*x*_IO[7:0]}.

2. Determine the 12-bit Start and End Register File addresses. The 12-bit Start Address is given by {DMA*x*_H[3:0], DMA_START[7:0]}. The 12-bit End Address is given by {DMA*x*_H[7:4], DMA_END[7:0]}.

3. Write the Start and End Register File address high nibbles to the DMA*x* End/Start Address High Nibble register.

4. Write the lower byte of the Start Address to the DMA*x* Start/Current Address register.

5. Write the lower byte of the End Address to the DMA*x* End Address register.

6. Write to the DMA*x* Control register to complete the following:
   – Select loop or single-pass mode operation
   – Select the data transfer direction (either from the Register File RAM to the on-chip peripheral control register; or from the on-chip peripheral control register to the Register File RAM)
   – Enable the DMA*x* interrupt request, if desired
   – Select Word or Byte mode
   – Select the DMA*x* request trigger
   – Enable the DMA*x* channel

DMA_ADC Operation

DMA_ADC transfers data from the ADC to the Register File. The sequence of operations in a DMA_ADC data transfer is:

1. ADC completes conversion on the current ADC input channel and signals the DMA controller that two-bytes of ADC data are ready for transfer.

2. DMA_ADC requests control of the system bus (address and data) from the eZ8 CPU.

3. After the eZ8 CPU acknowledges the bus request, DMA_ADC transfers the two-byte ADC output value to the Register File and then returns system bus control back to the eZ8 CPU.

4. If the current ADC Analog Input is the highest numbered input to be converted:
   – DMA_ADC resets the ADC Analog Input number to 0 and initiates data conversion on ADC Analog Input 0.
   – If configured to generate an interrupt, DMA_ADC sends an interrupt request to the Interrupt Controller

Z8 Encore! XP® 64K Series Flash Microcontrollers
Product Specification

169

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]}.

**Z8 Encore! XP® 64K Series Flash Microcontrollers
Product Specification**

190

Flash Control Register Definitions

Flash Control Register

The Flash Control register (Table 92) unlocks the Flash Controller for programming and erase operations, or to select the Flash Sector Protect register.

The Write-only Flash Control Register shares its Register File address with the Read-only Flash Status Register.

Table 92. Flash Control Register (FCTL)

BITS	7	6	5	4	3	2	1	0
FIELD	FCMD							
RESET	0							
R/W	W							
ADDR	FF8H							

FCMD—Flash Command
73H = First unlock command.
8CH = Second unlock command.
95H = Page erase command.
63H = Mass erase command
5EH = Flash Sector Protect register select.

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

Flash Status Register

The Flash Status register (Table 93) indicates the current state of the Flash Controller. This register can be read at any time. The Read-only Flash Status Register shares its Register File address with the Write-only Flash Control Register.

Table 93. Flash Status Register (FSTAT)

BITS	7	6	5	4	3	2	1	0
FIELD	Reserved		FSTAT					
RESET	0							
R/W	R							
ADDR	FF8H							

Z8 Encore! XP® 64K Series Flash Microcontrollers
Product Specification

198

Z8 Encore! XP® 64K Series Flash Microcontrollers
Product Specification

200

Operation

OCD Interface

The On-Chip Debugger uses the DBG pin for communication with an external host. This one-pin interface is a bi-directional open-drain interface that transmits and receives data. Data transmission is half-duplex, in that transmit and receive cannot occur simultaneously. The serial data on the DBG pin is sent using the standard asynchronous data format defined in RS-232. This pin can interface the 64K Series products to the serial port of a host PC using minimal external hardware.Two different methods for connecting the DBG pin to an RS-232 interface are depicted in Figure 37 and Figure 38 on page 201.

⚠️ **Caution:** *For operation of the On-Chip Debugger, **all** power pins (V_{DD} and AV_{DD}) must be supplied with power, and **all** ground pins (V_{SS} and AV_{SS}) must be properly grounded.*
The DBG pin is open-drain and must always be connected to V_{DD} through an external pull-up resistor to ensure proper operation.
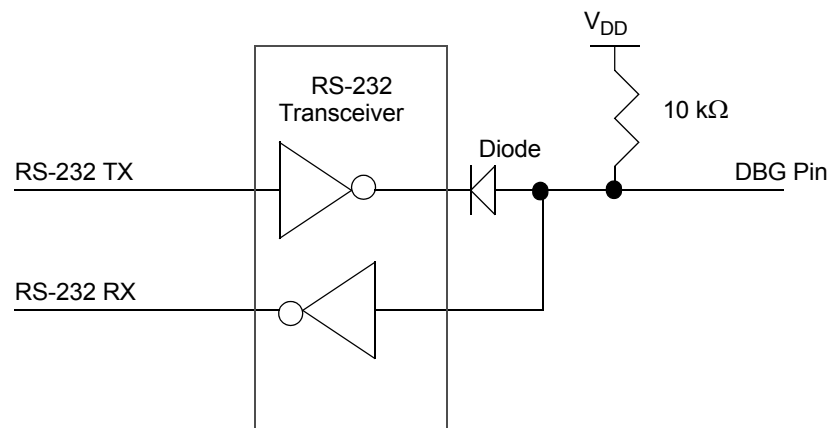

Figure 37. Interfacing the On-Chip Debugger's DBG Pin with an RS-232 Interface (1)

Z8 Encore! XP® 64K Series Flash Microcontrollers
Product Specification

zilog

204

finish the interrupt service routine it may be in and return the BRK instruction. When the CPU returns to the BRK instruction it was previously looping on, it automatically sets the DBGMODE bit and enter DEBUG mode.

Software detects that the majority of the OCD commands are still disabled when the eZ8™ CPU is looping on a BRK instruction. The eZ8 CPU must be stopped and the part must be in DEBUG mode before these commands can be issued.

Breakpoints in Flash Memory

The BRK instruction is opcode 00H, which corresponds to the fully programmed state of a byte in Flash memory. To implement a Breakpoint, write 00H to the desired address, over-writing the current instruction. To remove a Breakpoint, the corresponding page of Flash memory must be erased and reprogrammed with the original data.

On-Chip Debugger Commands

The host communicates to the On-Chip Debugger by sending OCD commands using the DBG interface. During normal operation, only a subset of the OCD commands are available. In DEBUG mode, all OCD commands become available unless the user code and control registers are protected by programming the Read Protect Option Bit (RP). The Read Protect Option Bit prevents the code in memory from being read out of the 64K Series products. When this option is enabled, several of the OCD commands are disabled. Table 101 contains a summary of the On-Chip Debugger commands. Each OCD command is described in detail in the bulleted list following Table 101.
Table 101 indicates those commands that operate when the device is not in DEBUG mode (normal operation) and those commands that are disabled by programming the Read Protect Option Bit.

Table 101. On-Chip Debugger Commands

Debug Command	Command Byte	Enabled when NOT in DEBUG mode?	Disabled by Read Protect Option Bit
Read OCD Revision	00H	Yes	-
Read OCD Status Register	02H	Yes	-
Read Runtime Counter	03H	-	-
Write OCD Control Register	04H	Yes	Cannot clear DBGMODE bit
Read OCD Control Register	05H	Yes	-

**Z8 Encore! XP® 64K Series Flash Microcontrollers
Product Specification**

zilog

238

UART Timing

Figure 56 and Table 120 provide timing information for UART pins for the case where the Clear To Send input pin ($\overline{\text{CTS}}$) is used for flow control. In this example, it is assumed that the Driver Enable polarity has been configured to be Active Low and is represented here by $\overline{\text{DE}}$. The $\overline{\text{CTS}}$ to $\overline{\text{DE}}$ assertion delay (T1) assumes the UART Transmit Data register has been loaded with data prior to $\overline{\text{CTS}}$ assertion.
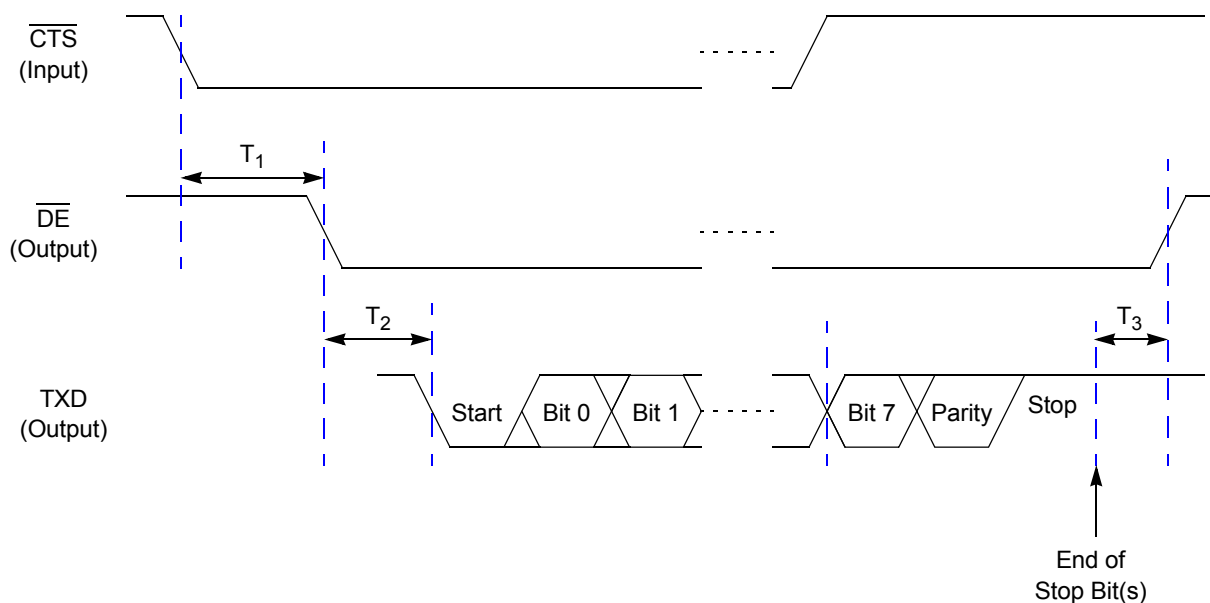

Figure 56. UART Timing with $\overline{\text{CTS}}$

Table 120. UART Timing with $\overline{\text{CTS}}$

Parameter	Abbreviation	Delay (ns) Minimum	Maximum
T_1	$\overline{\text{CTS}}$ Fall to $\overline{\text{DE}}$ Assertion Delay	2 * XIN period	2 * XIN period + 1 Bit period
T_2	$\overline{\text{DE}}$ Assertion to TXD Falling Edge (Start) Delay	1 Bit period	1 Bit period + 1 * XIN period
T_3	End of Stop Bit(s) to $\overline{\text{DE}}$ Deassertion Delay	1 * XIN period	2 * XIN period

Z8 Encore! XP® 64K Series Flash Microcontrollers
Product Specification

247

Table 126. Bit Manipulation Instructions (Continued)

Mnemonic	Operands	Instruction
BSWAP	dst	Bit Swap
CCF	—	Complement Carry Flag
RCF	—	Reset Carry Flag
SCF	—	Set Carry Flag
TCM	dst, src	Test Complement Under Mask
TCMX	dst, src	Test Complement Under Mask using Extended Addressing
TM	dst, src	Test Under Mask
TMX	dst, src	Test Under Mask using Extended Addressing

Table 127. Block Transfer Instructions

Mnemonic	Operands	Instruction
LDCI	dst, src	Load Constant to/from Program Memory and Auto-Increment Addresses
LDEI	dst, src	Load External Data to/from Data Memory and Auto-Increment Addresses

Table 128. CPU Control Instructions

Mnemonic	Operands	Instruction
ATM	—	Atomic Execution
CCF	—	Complement Carry Flag
DI	—	Disable Interrupts
EI	—	Enable Interrupts
HALT	—	HALT Mode
NOP	—	No Operation
RCF	—	Reset Carry Flag
SCF	—	Set Carry Flag
SRP	src	Set Register Pointer