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, POR, PWM, WDT
Number of I/O	19
Program Memory Size	8KB (8K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	1K x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 3.6V
Data Converters	-
Oscillator Type	Internal
Operating Temperature	0°C ~ 70°C (TA)
Mounting Type	Surface Mount
Package / Case	28-SOIC (0.295", 7.50mm Width)
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/zilog/z8f0812sj020sc00tr

remains below the POR voltage threshold (V_{POR}), the VBO block holds the device in the Reset state.

After the supply voltage again exceeds the POR voltage threshold, the device progresses through a full System Reset sequence as described in the POR section. Following POR, the POR status bit in the Watchdog Timer Control Register (WDTCTL) is set to 1. Figure 7 displays the VBO operation. See Electrical Characteristics on page 185 for the VBO and POR threshold voltages (V_{VBO} and V_{POR}).

The VBO circuit can be either enabled or disabled during STOP mode. Operation during STOP mode is set by the VBO_AO Option Bit. For information on configuring VBO_AO, see Option Bits on page 163.

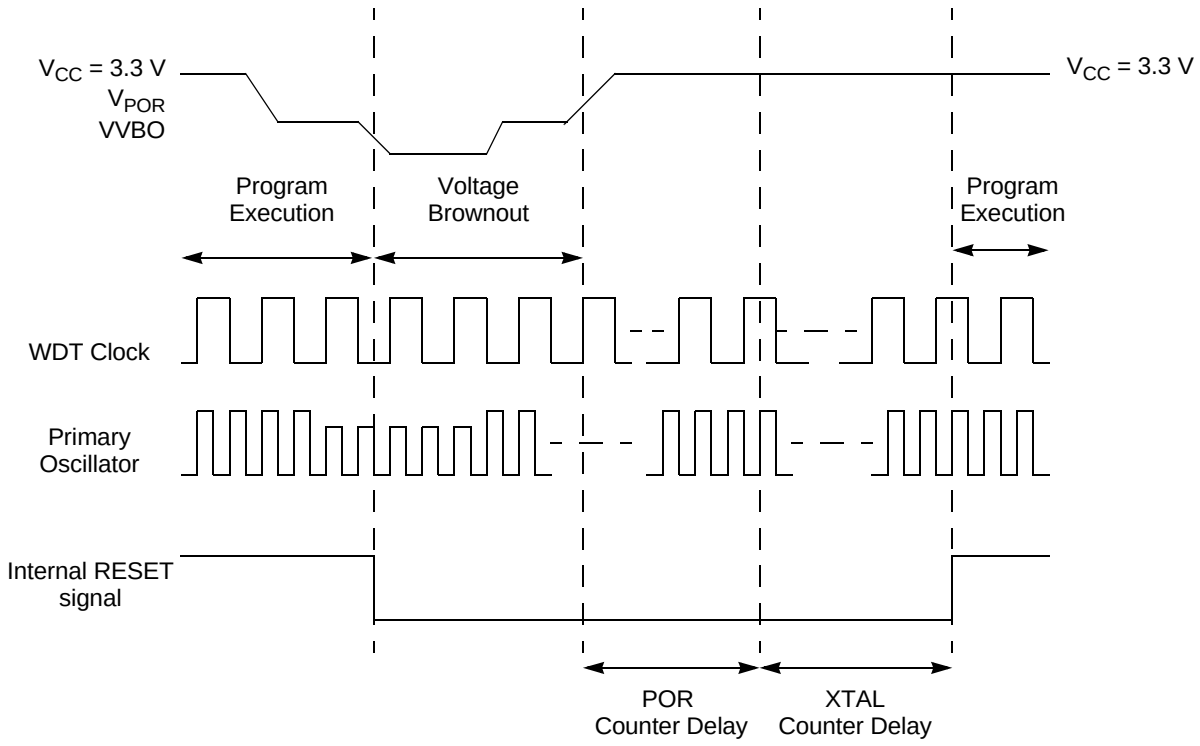


Figure 7. Voltage Brownout Reset Operation

Watchdog Timer Reset

If the device is in NORMAL or HALT mode, WDT initiates a System Reset at time-out, if the WDT_RES Option Bit is set to 1. This is the default (unprogrammed) setting of the WDT_RES Option Bit. The WDT status bit in the WDT Control Register is set to signify that the reset was initiated by the WDT.

Stop Mode Recovery Using WDT Time-Out

If the WDT times out during STOP mode, the device undergoes a Stop Mode Recovery sequence. In the WDT Control Register, the WDT and STOP bits are set to 1. If the WDT is configured to generate an interrupt upon time-out and the Z8 Encore! XP® F0822 Series device is configured to respond to interrupts, the eZ8 CPU services the WDT interrupt request following the normal Stop Mode Recovery sequence.

Stop Mode Recovery Using a GPIO Port Pin Transition

Each of the GPIO Port pins can be configured as a Stop Mode Recovery input source. On any GPIO pin enabled as a STOP Mode Recover source, a change in the input pin value (from High to Low or from Low to High) initiates Stop Mode Recovery. The GPIO Stop Mode Recovery signals are filtered to reject pulses less than 10 ns (typical) in duration. In the WDT Control Register, the STOP bit is set to 1.

Caution: *In STOP mode, the GPIO Port Input Data Registers (PxIN) are disabled. The Port Input Data Registers record the Port transition only if the signal stays on the Port pin through the end of the Stop Mode Recovery delay. Therefore, short pulses on the Port pin initiates Stop Mode Recovery without being written to the Port Input Data Register or without initiating an interrupt (if enabled for that pin).*

Interrupt Controller

The interrupt controller on Z8 Encore! XP® F0822 Series products prioritizes the interrupt requests from the on-chip peripherals and the GPIO port pins. The features of the interrupt controller include the following:

- 19 unique interrupt vectors:
  - 12 GPIO port pin interrupt sources.
  - 7 On-chip peripheral interrupt sources.
- Flexible GPIO interrupts:
  - 8 selectable rising and falling edge GPIO interrupts.
  - 4 dual-edge interrupts.
- Three levels of individually programmable interrupt priority.
- WDT is configured to generate an interrupt.

Interrupt Requests (IRQs) allow peripheral devices to suspend CPU operation in an orderly manner and force the CPU to start an Interrupt Service Routine (ISR). Usually this ISR is involved with the exchange of data, status information, or control information between the CPU and the interrupting peripheral. When the service routine is completed, the CPU returns to the operation from which it was interrupted.

The eZ8 CPU supports both vectored and polled interrupt handling. For polled interrupts, the interrupt control has no effect on operation. For more information on interrupt servicing, refer to *eZ8 CPU Core User Manual (UM0128)* available for download at www.zilog.com.

Interrupt Vector Listing

Table 24 lists all the interrupts available in order of priority. The interrupt vector is stored with the most significant byte (MSB) at the even Program Memory address and the least significant byte (LSB) at the following odd Program Memory address.

Table 24. Interrupt Vectors in Order of Priority

Priority	Program Memory Vector Address	Interrupt Source
Highest	0002H	Reset (not an interrupt)
	0004H	WDT (see Watchdog Timer on page 83)
	0006H	Illegal Instruction Trap (not an interrupt)

Poor coding style that resulting in lost interrupt requests:

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

Note: *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 WDT 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 25) 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 reads the IRQ0 Register to determine if any interrupt requests are pending.

Table 25. Interrupt Request 0 Register (IRQ0)

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

Reserved—Must be 0

T1I—Timer 1 Interrupt Request
0 = No interrupt request is pending for Timer 1.
1 = An interrupt request from Timer 1 is awaiting service.

T0I—Timer 0 Interrupt Request
0 = No interrupt request is pending for Timer 0.
1 = An interrupt request from Timer 0 is awaiting service.

U0RXI—UART 0 Receiver Interrupt Request
0 = No interrupt request is pending for the UART 0 receiver.
1 = An interrupt request from the UART 0 receiver is awaiting service.

U0TXI—UART 0 Transmitter Interrupt Request
0 = No interrupt request is pending for the UART 0 transmitter.
1 = An interrupt request from the UART 0 transmitter is awaiting service.

I2CI— I^2C Interrupt Request
0 = No interrupt request is pending for the I^2C.
1 = An interrupt request from the I^2C is awaiting service.

SPII—SPI Interrupt Request
0 = No interrupt request is pending for the SPI.
1 = An interrupt request from the SPI is awaiting service.

ADCI—ADC Interrupt Request
0 = No interrupt request is pending for the ADC.
1 = An interrupt request from the ADC is awaiting service.

Interrupt Request 1 Register

The Interrupt Request 1 (IRQ1) Register (Table 26) stores interrupt requests for both vectored and polled interrupts. When a request is presented to the interrupt controller, the corresponding bit in the IRQ1 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 reads the IRQ1 Register to determine if any interrupt requests are pending.

Table 26. Interrupt Request 1 Register (IRQ1)

BITS	7	6	5	4	3	2	1	0
FIELD	PA7I	PA6I	PA5I	PA4I	PA3I	PA2I	PA1I	PA0I
RESET	0							
R/W	R/W							
ADDR	FC3H							

PA*x*I—Port A Pin *x* Interrupt Request
0 = No interrupt request is pending for GPIO Port A pin *x*.
1 = An interrupt request from GPIO Port A pin *x* is awaiting service.
Where *x* indicates the specific GPIO Port pin number (0 through 7).

CAPTURE/COMPARE Mode

In CAPTURE/COMPARE mode, the timer begins counting on the *first* external Timer Input transition. The required transition (rising edge or falling edge) is set by the TPOL bit in the Timer Control Register. The timer input is the system clock.

Every subsequent desired transition (after the first) of the Timer Input signal captures the current count value. The Capture value is written to the Timer PWM High and Low Byte Registers. When the Capture event occurs, an interrupt is generated, the count value in the Timer High and Low Byte Registers is reset to 0001H and counting resumes.

If no Capture event occurs, the timer counts up to the 16-bit Compare value stored in the Timer Reload High and Low Byte registers. Upon reaching the Compare value, the timer generates an interrupt, the count value in the Timer High and Low Byte Registers is reset to 0001H and counting resumes.

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

1. Write to the Timer Control Register to:
   – Disable the timer
   – Configure the timer for CAPTURE/COMPARE mode
   – Set the prescale value
   – Set the Capture edge (rising or falling) for the Timer Input
2. Write to the Timer High and Low Byte registers to set the starting count value (typically 0001H)
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. Configure the associated GPIO port pin for the Timer Input alternate function
6. Write to the Timer Control Register to enable the timer
7. Counting begins on the first appropriate transition of the Timer Input signal. No interrupt is generated by this first edge

In CAPTURE/COMPARE mode, the elapsed time from timer start to Capture event is calculated using the following equation:

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

Reading the Timer Count Values

The current count value in the timers can be read while counting (enabled). This capability has no effect on timer operation. When the timer is enabled and the Timer High Byte

The UART is now configured for interrupt-driven data transmission. Because the UART Transmit Data Register is empty, an interrupt is generated immediately. When the UART Transmit Interrupt is detected, the associated ISR performs the following:

1. Write the UART Control 1 Register to select the outgoing address bit:
   – Set the Multiprocessor Bit Transmitter (MPBT) if sending an address byte, clear it if sending a data byte.
2. Write the data byte to the UART Transmit Data Register. The transmitter automatically transfers data to the Transmit Shift Register and then transmits the data.
3. Clear the UART Transmit Interrupt bit in the applicable Interrupt Request Register.
4. Execute the IRET instruction to return from the ISR and waits for the Transmit Data Register to again become empty.

Receiving Data using the Polled Method

Follow the steps below to configure the UART for polled data reception:

1. Write to the UART Baud Rate High and Low Byte Registers to set the required baud rate.
2. Enable the UART pin functions by configuring the associated GPIO Port pins for alternate function operation.
3. Write to the UART Control 1 Register to enable Multiprocessor mode functions, if desired.
4. Write to the UART Control 0 Register to:
   – Set the receive enable bit (REN) to enable the UART for data reception
   – Enable parity, if required, and if MULTIPROCESSOR mode is not enabled, and select either even or odd parity.
5. Check the RDA bit in the UART Status 0 Register to determine if the Receive Data Register contains a valid data byte (indicated by 1). If RDA is set to 1 to indicate available data, continue to step 6. If the Receive Data Register is empty (indicated by a 0), continue to monitor the RDA bit awaiting reception of the valid data.
6. Read 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].
7. Return to step 5 to receive additional data.

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.

Table 55. UART Status 1 Register (U0STAT1)

BITS	7	6	5	4	3	2	1	0
FIELD	Reserved						NEWFRM	MPRX
RESET	0							
R/W	R				R/W		R	
ADDR	F44H							

For a given UART data rate, the integer baud rate divisor value is calculated using the following equation:

$$\text{UART Baud Rate Divisor Value (BRG)} = \text{Round}\left(\frac{\text{System Clock Frequency (Hz)}}{16 \times \text{UART Data Rate (bits/s)}}\right)$$

The baud rate error relative to the desired baud rate is calculated using the following equation:

$$\text{UART Baud Rate Error (\%)} = 100 \times \left(\frac{\text{Actual Data Rate} - \text{Desired Data Rate}}{\text{Desired Data Rate}}\right)$$

For reliable communication, the UART baud rate error must never exceed 5 percent. Table 61 provides information on data rate errors for popular baud rates and commonly used crystal oscillator frequencies.

Table 61. UART Baud Rates

10.0 MHz System Clock				5.5296 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	1	625.0	0.00	625.0	N/A	N/A	N/A
250.0	3	208.33	-16.67	250.0	1	345.6	38.24
115.2	5	125.0	8.51	115.2	3	115.2	0.00
57.6	11	56.8	-1.36	57.6	6	57.6	0.00
38.4	16	39.1	1.73	38.4	9	38.4	0.00
19.2	33	18.9	0.16	19.2	18	19.2	0.00
9.60	65	9.62	0.16	9.60	36	9.60	0.00
4.80	130	4.81	0.16	4.80	72	4.80	0.00
2.40	260	2.40	-0.03	2.40	144	2.40	0.00
1.20	521	1.20	-0.03	1.20	288	1.20	0.00
0.60	1042	0.60	-0.03	0.60	576	0.60	0.00
0.30	2083	0.30	0.2	0.30	1152	0.30	0.00

The baud rate is set by the UART's Baud Rate Generator and supports IrDA standard baud rates from 9600 baud to 115.2 Kbaud. Higher baud rates are possible, but do not meet IrDA specifications. The UART must be enabled to use the Infrared Endec. The Infrared Endec data rate is calculated using the following equation.

$$\text{Infrared Data Rate (bits/s)} = \frac{\text{System Clock Frequency (Hz)}}{16 \times \text{UART Baud Rate Divisor Value}}$$

Transmitting IrDA Data

The data to be transmitted using the infrared transceiver is first sent to the UART. The UART's transmit signal (TXD) and baud rate clock are used by the IrDA to generate the modulation signal (IR_TXD) that drives the infrared transceiver. Each UART/Infrared data bit is 16-clocks wide. If the data to be transmitted is 1, the IR_TXD signal remains low for the full 16-clock period. If the data to be transmitted is 0, a 3-clock high pulse is output following a 7-clock low period. After the 3-clock high pulse, a 6-clock low pulse is output to complete the full 16-clock data period. Figure 18 displays IrDA data transmission. When the Infrared Endec is enabled, the UART's TXD signal is internal to the Z8 Encore! XP® F0822 Series products while the IR_TXD signal is output through the TXD pin.

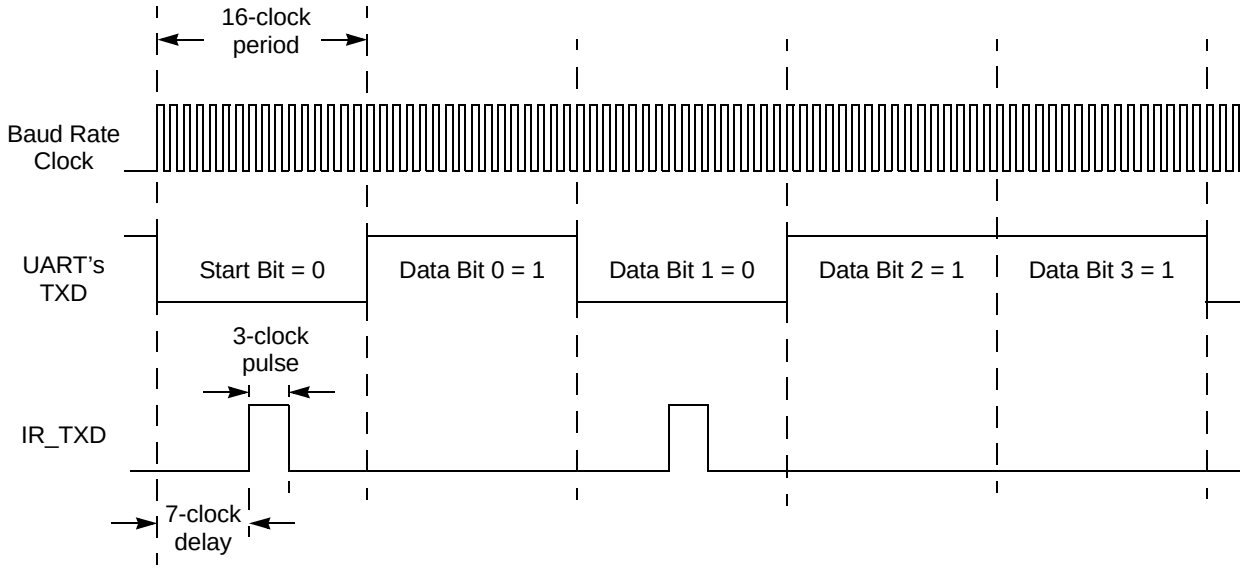


Figure 18. Infrared Data Transmission

Follow the steps below for a transmit operation on 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.
4. Software responds to the TDRE interrupt by writing the first slave address byte to the I2C Data Register. 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.
7. The I2C Controller loads the I2C Shift register with the contents of the I2C 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 I2C Data Register.
10. The I2C Controller shifts the rest of the first byte of address and write bit out the SDA signal.
11. If the I2C Slave acknowledges the first address byte 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 12.

    If the slave does not acknowledge the first address byte, the I2C Controller sets the NCKI bit and clears the ACK bit in the I2C Status register. Software responds to the Not Acknowledge interrupt by setting the STOP and FLUSH bits 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).
12. The I2C Controller loads the I2C Shift register with the contents of the I2C Data Register.
13. The I2C 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 I2C Data Register.
15. The I2C Controller completes shifting the contents of the shift register on the SDA signal.
16. If the I2C Slave sends an acknowledge 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 17.

    If the slave does not acknowledge the second address byte or one of the data bytes, the

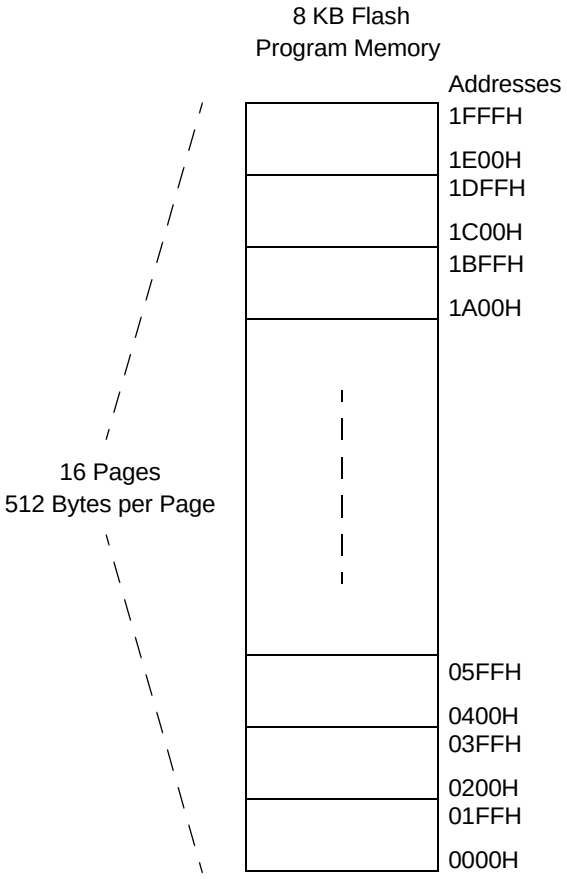


Figure 33. Flash Memory Arrangement

Information Area

Table 82 on page 155 describes the Z8 Encore! XP® F0822 Series Information Area. This 512-byte Information Area is accessed by setting bit 7 of the Page Select Register to 1. When access is enabled, the Information Area is mapped into Flash Memory and overlays the 512 bytes at addresses FE00H to FFFFH. When the Information Area access is enabled, LDC instructions return data from the Information Area. CPU instruction fetches always comes from Flash Memory regardless of the Information Area access bit. Access to the Information Area is read-only.

Table 82. Z8 Encore! XP® F0822 Series Information Area Map

Flash Memory Address (Hex)	Function
FE00H-FE3FH	Reserved
FE40H-FE53H	**Part Number** 20-character ASCII alphanumeric code Left justified and filled with zeros
FE54H-FFFFH	Reserved

Operation

The Flash Controller provides the proper signals and timing for Byte Programming, Page Erase, and Mass Erase of the Flash memory. The Flash Controller contains a protection mechanism, using the Flash Control Register (FCTL), to prevent accidental programming or erasure. The following subsections provide details on the various operations (Lock, Unlock, Sector Protect, Byte Programming, Page Erase, and Mass Erase).

Timing Using the Flash Frequency Registers

Before performing a program or erase operation on the Flash memory, you must first configure the Flash Frequency High and Low Byte registers. The Flash Frequency registers allow programming and erasure of the Flash with system clock frequencies ranging from 20 kHz through 20 MHz (the valid range is limited to the device operating frequencies).

The Flash Frequency High and Low Byte registers combine to form a 16-bit value, FFREQ, to control timing for Flash program and erase operations. The 16-bit Flash Frequency value must contain the system clock frequency in kHz. This value is calculated using the following equation:

$$\text{FFREQ[15:0]} = \frac{\text{System Clock Frequency (Hz)}}{1000}$$

Caution: *Flash programming and erasure are not supported for system clock frequencies below 20 kHz, above 20 MHz, or outside of the device operating frequency range. The Flash Frequency High and Low Byte registers must be loaded with the correct value to insure proper Flash programming and erase operations.*

Follow the steps below to setup the Flash Sector Protect Register from user code:

1. Write 00H to the Flash Control Register to reset the Flash Controller.
2. Write 5EH to the Flash Control Register to select the Flash Sector Protect Register.
3. Read and/or write the Flash Sector Protect Register which is now at Register File address FF9H.
4. Write 00H to the Flash Control Register to return the Flash Controller to its reset state.

Flash Write Protection Option Bit

The Flash Write Protect option bit can block all program and erase operations from user code. For more information, see Option Bits on page 163.

Byte Programming

When the Flash Controller is unlocked, writes to Flash Memory from user code programs a byte into the Flash if the address is located in the unlocked page. An erased Flash byte contains all 1s (FFH). The programming operation is used to change bits from 1 to 0. To change a Flash bit (or multiple bits) from zero to one requires a Page Erase or Mass Erase operation.

Byte Programming is accomplished using the eZ8 CPU's LDC or LDCI instructions. Refer to *eZ8 CPU Core User Manual (UM0128)* for a description of the LDC and LDCI instructions.

While the Flash Controller programs the Flash memory, the eZ8 CPU idles but the system clock and on-chip peripherals continue to operate. Interrupts that occur when a Programming operation is in progress are serviced once the Programming operation is complete. To exit Programming mode and lock the Flash Controller, write 00H to the Flash Control Register.

User code cannot program Flash Memory on a page that is located in a protected sector. When user code writes memory locations, only addresses located in the unlocked page are programmed. Memory writes outside of the unlocked page are ignored.

! Caution: *Each memory location must not be programmed more than twice before an erase occurs.*

Follow the steps below to program the Flash from user code:

1. Write 00H to the Flash Control Register to reset the Flash Controller.
2. Write the page of memory to be programmed to the Page Select Register.
3. Write the first unlock command 73H to the Flash Control Register.
4. Write the second unlock command 8CH to the Flash Control Register.

OCDCNTR Register

The OCD contains a multipurpose 16-bit Counter Register. It can be used for the following:

- Count system clock cycles between Breakpoints.
- Generate a BRK when it counts down to zero.
- Generate a BRK when its value matches the Program Counter.

When configured as a counter, the OCDCNTR register starts counting when the OCD leaves DEBUG mode and stops counting when it enters DEBUG mode again or when it reaches the maximum count of FFFFH. The OCDCNTR register automatically resets itself to 0000H when the OCD exits DEBUG mode if it is configured to count clock cycles between breakpoints.

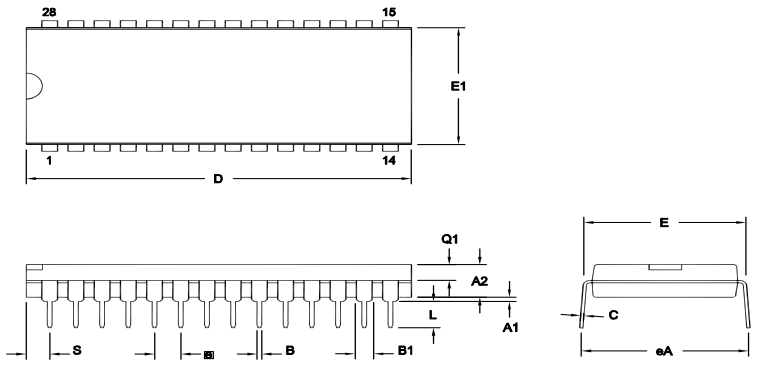
Caution: *The OCDCNTR register is used by many of the OCD commands. It counts the number of bytes for the register and memory read/write commands. It holds the residual value when generating the CRC. Therefore, if the OCDCNTR is being used to generate a BRK, its value should be written as a last step before leaving DEBUG mode.*

Since this register is overwritten by various OCD commands, it should only be used to generate temporary breakpoints, such as stepping over CALL instructions or running to a specific instruction and stopping.

On-Chip Debugger Commands

The host communicates to the OCD 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 Z8 Encore! XP F0822 Series products. When this option is enabled, several of the OCD commands are disabled. Table 93 on page 177 contains a summary of the OCD commands. Each OCD command is described further in the bulleted list. It also lists the 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.

Figure 63 displays the 28-pin PDIP package available for Z8 Encore! XP F0822 Series devices.



SYMBOL	OPT #	MILLIMETER		INCH	
		MIN	MAX	MIN	MAX
A1		0.38	1.02	.015	.040
A2		3.18	4.19	.125	.165
B		0.38	0.53	.015	.021
B1	01	1.40	1.65	.055	.065
	02	1.14	1.40	.045	.055
C		0.23	0.38	.009	.015
D	01	36.58	37.34	1.440	1.470
	02	35.31	35.94	1.390	1.415
E		15.24	15.75	.600	.620
E1	01	13.59	14.10	.535	.555
	02	12.83	13.08	.505	.515
e		2.54 TYP		.100 BSC	
eA		15.49	16.76	.610	.660
L		3.05	3.81	.120	.150
Q1	01	1.40	1.91	.055	.075
	02	1.40	1.78	.055	.070
S	01	1.52	2.29	.060	.090
	02	1.02	1.52	.040	.060

CONTROLLING DIMENSIONS : INCH

OPTION TABLE	
OPTION #	PACKAGE
01	STANDARD
02	IDF

Note: ZILOG supplies both options for production. Component layout PCB design should cover bigger option 01.

Figure 63. 28-Pin Plastic Dual-Inline Package (PDIP)

Index

Symbols

Numerics

A

B

C

D

E

F