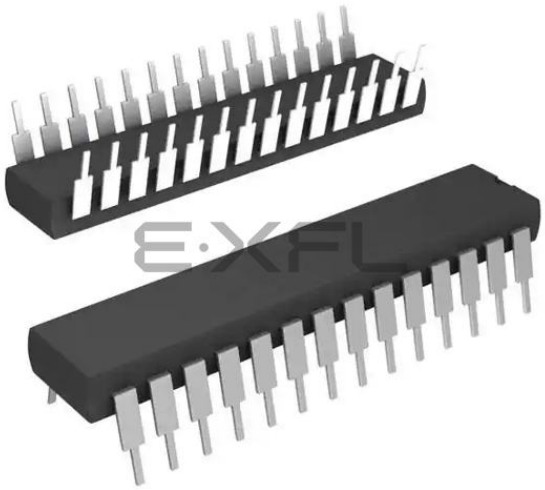

Welcome to E-XFL.COM

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	IrDA, UART/USART
Peripherals	Brown-out Detect/Reset, LED, LVD, POR, PWM, WDT
Number of I/O	25
Program Memory Size	2KB (2K x 8)
Program Memory Type	FLASH
EEPROM Size	64 x 8
RAM Size	512 x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 3.6V
Data Converters	-
Oscillator Type	Internal
Operating Temperature	0°C ~ 70°C (TA)
Mounting Type	Through Hole
Package / Case	28-DIP (0.600", 15.24mm)
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/zilog/z8f021apj020sc

Reset, Stop Mode Recovery, and Low Voltage Detection

The Reset Controller within the Z8 Encore! XP® F082A Series controls Reset and Stop Mode Recovery operation and provides indication of low supply voltage conditions. In typical operation, the following events cause a Reset:

- Power-On Reset (POR)
- Voltage Brownout (VBO)
- Watchdog Timer time-out (when configured by the WDT_RES Flash Option Bit to initiate a reset)
- External $\overline{\text{RESET}}$ pin assertion (when the alternate RESET function is enabled by the GPIO register)
- On-chip debugger initiated Reset (OCDCTL[0] set to 1)

When the device is in STOP mode, a Stop Mode Recovery is initiated by either of the following:

- Watchdog Timer time-out
- GPIO Port input pin transition on an enabled Stop Mode Recovery source

The low voltage detection circuitry on the device (available on the 8-pin product versions only) performs the following functions:

- Generates the VBO reset when the supply voltage drops below a minimum safe level.
- Generates an interrupt when the supply voltage drops below a user-defined level (8-pin devices only).

Reset Types

The Z8 Encore! XP F082A Series provides several different types of Reset operation. Stop Mode Recovery is considered as a form of Reset. Table 8 lists the types of Reset and their operating characteristics. The System Reset is longer if the external crystal oscillator is enabled by the Flash option bits, allowing additional time for oscillator start-up.

duration of the negative phase of the PWM signal (as defined by the difference between the PWM registers and the Timer Reload registers).

5. Write to the Timer Reload High and Low Byte registers to set the Reload value (PWM period). The Reload value must be greater than the PWM value.
6. If appropriate, enable the timer interrupt and set the timer interrupt priority by writing to the relevant interrupt registers.
7. Configure the associated GPIO port pin for the Timer Output and Timer Output Complement alternate functions. The Timer Output Complement function is shared with the Timer Input function for both timers. Setting the timer mode to Dual PWM automatically switches the function from Timer In to Timer Out Complement.
8. Write to the Timer Control register to enable the timer and initiate counting.

The PWM period is represented by the following equation:

$$\text{PWM Period (s)} = \frac{\text{Reload Value x Prescale}}{\text{System Clock Frequency (Hz)}}$$

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

If TPOL is set to 0, the ratio of the PWM output High time to the total period is represented by:

$$\text{PWM Output High Time Ratio (\%)} = \frac{\text{Reload Value} - \text{PWM Value}}{\text{Reload Value}} \times 100$$

If TPOL is set to 1, the ratio of the PWM output High time to the total period is represented by:

$$\text{PWM Output High Time Ratio (\%)} = \frac{\text{PWM Value}}{\text{Reload Value}} \times 100$$

CAPTURE Mode

In CAPTURE mode, the current timer count value is recorded when the appropriate external Timer Input transition occurs. The Capture count value is written to the Timer PWM High and Low Byte Registers. The timer input is the system clock. The TPOL bit in the Timer Control register determines if the Capture occurs on a rising edge or a falling edge of the Timer Input signal. When the Capture event occurs, an interrupt is generated and the timer continues counting. The INPCAP bit in TxCTL0 register is set to indicate the timer interrupt is because of an input capture event.

The timer continues counting up to the 16-bit Reload value stored in the Timer Reload High and Low Byte registers. Upon reaching the Reload value, the timer generates an interrupt and continues counting. The INPCAP bit in TxCTL0 register clears indicating the timer interrupt is not because of an input capture event.

PWM SINGLE OUTPUT mode
0 = Timer Output is forced Low (0) when the timer is disabled. When enabled, the Timer Output is forced High (1) upon PWM count match and forced Low (0) upon Reload.
1 = Timer Output is forced High (1) when the timer is disabled. When enabled, the Timer Output is forced Low (0) upon PWM count match and forced High (1) upon Reload.

CAPTURE mode
0 = Count is captured on the rising edge of the Timer Input signal.
1 = Count is captured on the falling edge of the Timer Input signal.

COMPARE mode
When the timer is disabled, the Timer Output signal is set to the value of this bit. When the timer is enabled, the Timer Output signal is complemented upon timer Reload.

GATED mode
0 = Timer counts when the Timer Input signal is High (1) and interrupts are generated on the falling edge of the Timer Input.
1 = Timer counts when the Timer Input signal is Low (0) and interrupts are generated on the rising edge of the Timer Input.

CAPTURE/COMPARE mode
0 = Counting is started on the first rising edge of the Timer Input signal. The current count is captured on subsequent rising edges of the Timer Input signal.
1 = Counting is started on the first falling edge of the Timer Input signal. The current count is captured on subsequent falling edges of the Timer Input signal.

PWM DUAL OUTPUT mode
0 = Timer Output is forced Low (0) and Timer Output Complement is forced High (1) when the timer is disabled. When enabled, the Timer Output is forced High (1) upon PWM count match and forced Low (0) upon Reload. When enabled, the Timer Output Complement is forced Low (0) upon PWM count match and forced High (1) upon Reload. The PWMD field in TxCTL0 register is a programmable delay to control the number of cycles time delay before the Timer Output and the Timer Output Complement is forced to High (1).
1 = Timer Output is forced High (1) and Timer Output Complement is forced Low (0) when the timer is disabled. When enabled, the Timer Output is forced Low (0) upon PWM count match and forced High (1) upon Reload.When enabled, the Timer Output Complement is forced High (1) upon PWM count match and forced Low (0) upon Reload. The PWMD field in TxCTL0 register is a programmable delay to control the number of cycles time delay before the Timer Output and the Timer Output Complement is forced to Low (0).

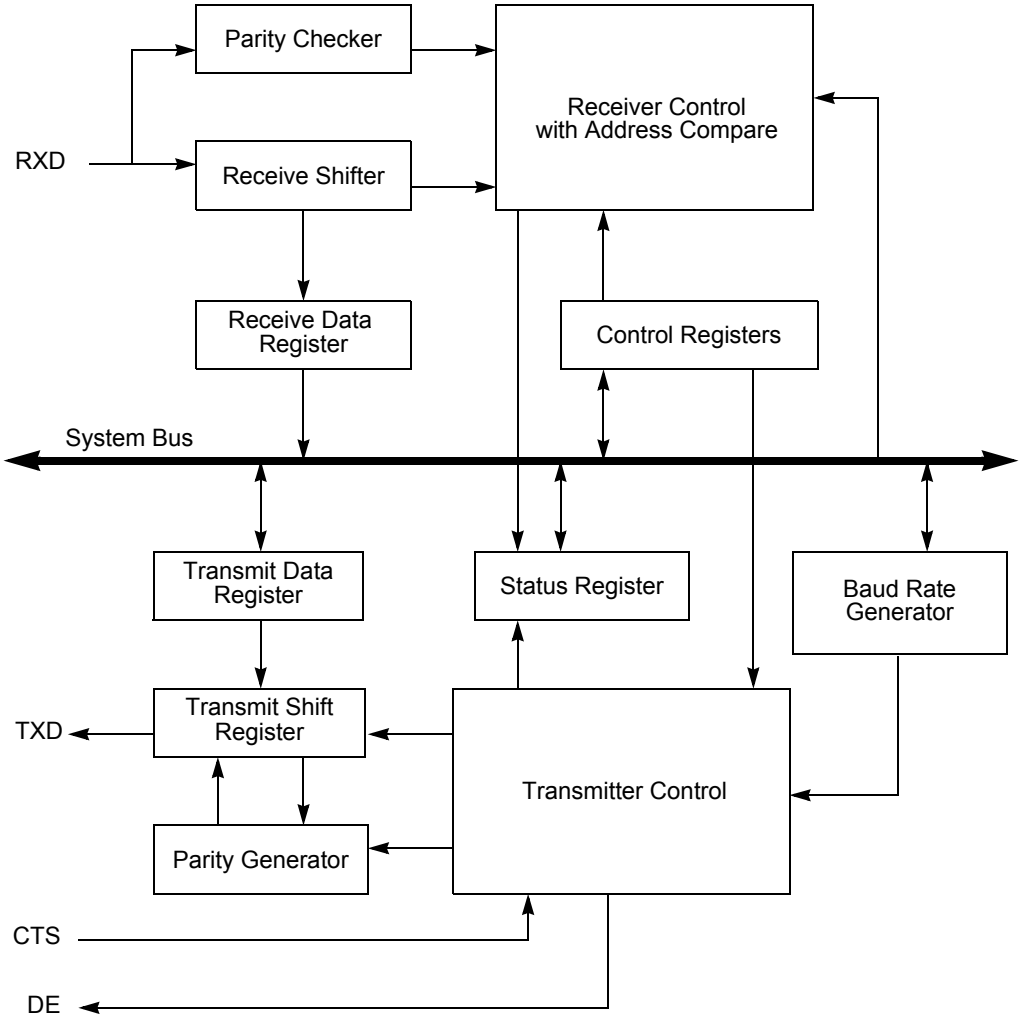

Figure 10. UART Block Diagram

Operation

Data Format

The UART always transmits and receives data in an 8-bit data format, least-significant bit first. An even or odd parity bit can be added to the data stream. Each character begins with an active Low START bit and ends with either 1 or 2 active High STOP bits. Figure 11 and Figure 12 display the asynchronous data format employed by the UART without parity and with parity, respectively.

6. Check the TDRE bit in the UART Status 0 register to determine if the Transmit Data register is empty (indicated by a 1). If empty, continue to Step 7. If the Transmit Data register is full (indicated by a 0), continue to monitor the TDRE bit until the Transmit Data register becomes available to receive new data.
7. Write the UART Control 1 register to select the outgoing address bit.
8. Set the Multiprocessor Bit Transmitter (MPBT) if sending an address byte, clear it if sending a data byte.
9. Write the data byte to the UART Transmit Data register. The transmitter automatically transfers the data to the Transmit Shift register and transmits the data.
10. Make any changes to the Multiprocessor Bit Transmitter (MPBT) value, if appropriate and MULTIPROCESSOR mode is enabled.
11. To transmit additional bytes, return to Step 5.

Transmitting Data using the Interrupt-Driven Method

The UART Transmitter interrupt indicates the availability of the Transmit Data register to accept new data for transmission. Follow the steps below to configure the UART for interrupt-driven data transmission:

1. Write to the UART Baud Rate High and Low Byte registers to set the appropriate 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 Transmitter interrupt and set the acceptable priority.
5. Write to the UART Control 1 register to enable MULTIPROCESSOR (9-bit) mode functions, if MULTIPROCESSOR mode is appropriate.
6. Set the MULTIPROCESSOR Mode Select (MPEN) to Enable MULTIPROCESSOR mode.
7. Write to the UART Control 0 register to:
   - Set the transmit enable bit (TEN) to enable the UART for data transmission.
   - Enable parity, if appropriate and if MULTIPROCESSOR mode is not enabled, and select either even or odd parity.
   - Set or clear CTSE to enable or disable control from the remote receiver using the $\overline{\text{CTS}}$ pin.
8. Execute an EI instruction to enable interrupts.

3. Clears the UART Receiver interrupt in the applicable Interrupt Request register.
4. Executes the IRET instruction to return from the interrupt-service routine and await more data.

Clear To Send ($\overline{\text{CTS}}$) Operation

The CTS pin, if enabled by the CTSE bit of the UART Control 0 register, performs flow control on the outgoing transmit datastream. The Clear To Send ($\overline{\text{CTS}}$) input pin is sampled one system clock before beginning any new character transmission. To delay transmission of the next data character, an external receiver must deassert $\overline{\text{CTS}}$ at least one system clock cycle before a new data transmission begins. For multiple character transmissions, this action is typically performed during Stop Bit transmission. If $\overline{\text{CTS}}$ deasserts in the middle of a character transmission, the current character is sent completely.

MULTIPROCESSOR (9-bit) Mode

The UART has a MULTIPROCESSOR (9-bit) mode that uses an extra (9th) bit for selective communication when a number of processors share a common UART bus. In MULTIPROCESSOR mode (also referred to as 9-bit mode), the multiprocessor bit (MP) is transmitted immediately following the 8-bits of data and immediately preceding the Stop bit(s) as displayed in Figure 13. The character format is:

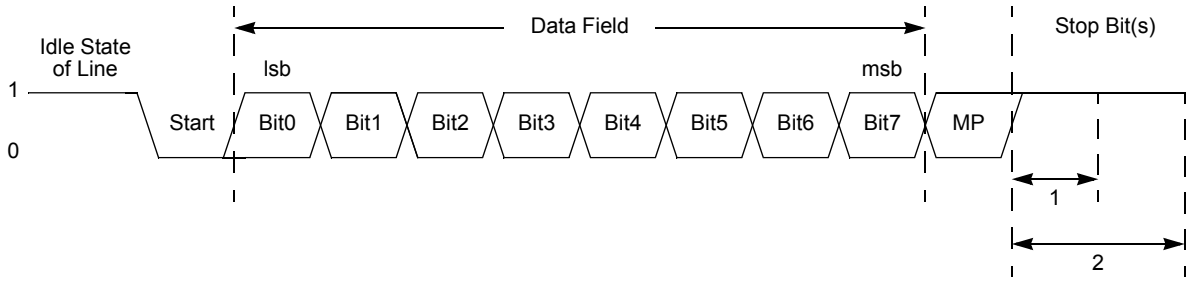

Figure 13. UART Asynchronous MULTIPROCESSOR Mode Data Format

In MULTIPROCESSOR (9-bit) mode, the Parity bit location (9th bit) becomes the Multiprocessor control bit. The UART Control 1 and Status 1 registers provide MULTIPROCESSOR (9-bit) mode control and status information. If an automatic address matching scheme is enabled, the UART Address Compare register holds the network address of the device.

MULTIPROCESSOR (9-bit) Mode Receive Interrupts

When MULTIPROCESSOR mode is enabled, the UART only processes frames addressed to it. The determination of whether a frame of data is addressed to the UART can be made in hardware, software or some combination of the two, depending on the multiprocessor

send. This action 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.

Receiver Interrupts

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

- A data byte is received and is available in the UART Receive Data register. This interrupt can be disabled independently of the other receiver interrupt sources. The received data interrupt occurs after the receive character has been received and placed in the Receive Data register. To avoid an overrun error, software must respond to this received data available condition before the next character is completely received.

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

- A break is received.
- An overrun is detected.
- A data framing error is detected.

UART Overrun Errors

When an overrun error condition occurs the UART prevents overwriting of the valid data currently in the Receive Data register. The Break Detect and Overrun status bits are not displayed until after the valid data has been read.

After the valid data has been read, the UART Status 0 register is updated to indicate the overrun condition (and Break Detect, if applicable). The RDA bit is set to 1 to indicate that the Receive Data register contains a data byte. However, because the overrun error occurred, this byte may not contain valid data and must be ignored. The BRKD bit indicates if the overrun was caused by a break condition on the line. After reading the status byte indicating an overrun error, the Receive Data register must be read again to clear the error bits is the UART Status 0 register. Updates to the Receive Data register occur only when the next data word is received.

UART Data and Error Handling Procedure

Figure 15 displays the recommended procedure for use in UART receiver interrupt service routines.

Analog-to-Digital Converter

The analog-to-digital converter (ADC) converts an analog input signal to its digital representation. The features of this sigma-delta ADC include:

- 11-bit resolution in DIFFERENTIAL mode.
- 10-bit resolution in SINGLE-ENDED mode.
- Eight single-ended analog input sources are multiplexed with general-purpose I/O ports.
- 9th analog input obtained from temperature sensor peripheral.
- 11 pairs of differential inputs also multiplexed with general-purpose I/O ports.
- Low-power operational amplifier (LPO).
- Interrupt on conversion complete.
- Bandgap generated internal voltage reference with two selectable levels.
- Manual in-circuit calibration is possible employing user code (offset calibration).
- Factory calibrated for in-circuit error compensation.

Architecture

Figure 19 displays the major functional blocks of the ADC. An analog multiplexer network selects the ADC input from the available analog pins, ANA0 through ANA7.

The input stage of the ADC allows both differential gain and buffering. The following input options are available:

- Unbuffered input (SINGLE-ENDED and DIFFERENTIAL modes).
- Buffered input with unity gain (SINGLE-ENDED and DIFFERENTIAL modes).
- LPO output with full pin access to the feedback path.

Comparator

The Z8 Encore! XP® F082A Series devices feature a general purpose comparator that compares two analog input signals. These analog signals may be external stimulus from a pin (CINP and/or CINN) or internally generated signals. Both a programmable voltage reference and the temperature sensor output voltage are available internally. The output is available as an interrupt source or can be routed to an external pin.

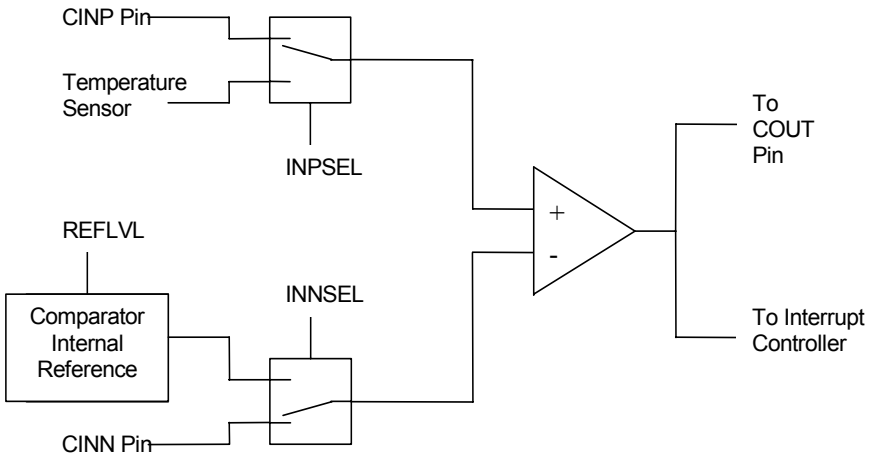

Figure 20. Comparator Block Diagram

Operation

When the positive comparator input exceeds the negative input by more than the specified hysteresis, the output is a logic HIGH. When the negative input exceeds the positive by more than the hysteresis, the output is a logic LOW. Otherwise, the comparator output retains its present value. See Table 137 on page 233 for details.

The comparator may be powered down to reduce supply current. See Power Control Register 0 on page 34 for details.

Caution: *Because of the propagation delay of the comparator, it is not recommended to enable or reconfigure the comparator without first disabling interrupts and waiting for the comparator output to settle. Doing so can result in spurious interrupts. The following example describes how to safely enable the comparator:*

```
di
ld cmp0, r0 ; load some new configuration
nop
```

Flash Memory

The products in the Z8 Encore! XP® F082A Series feature a non-volatile Flash memory of 8 KB (8192), 4 KB (4096), 2 KB (2048 bytes), or 1 KB (1024) with read/write/erase capability. The Flash Memory can be programmed and erased in-circuit by user code or through the On-Chip Debugger. The features include:

- User controlled read and write protect capability
- Sector-based write protection scheme
- Additional protection schemes against accidental program and erasure

Architecture

The Flash memory array is arranged in pages with 512 bytes per page. The 512 byte page is the minimum Flash block size that can be erased. Each page is divided into 8 rows of 64 bytes.

For program or data protection, the Flash memory is also divided into sectors. In the Z8 Encore! XP F082A Series, these sectors are either 1024 bytes (in the 8 KB devices) or 512 bytes (all other memory sizes) in size. Page and sector sizes are not generally equal.

The first 2 bytes of the Flash Program memory are used as Flash Option Bits. For more information about their operation, see Flash Option Bits on page 153.

Table 76 describes the Flash memory configuration for each device in the Z8 Encore! XP F082A Series. Figure 21 displays the Flash memory arrangement.

Table 76. Z8 Encore! XP F082A Series Flash Memory Configurations

Part Number	Flash Size KB (Bytes)	Flash Pages	Program Memory Addresses	Flash Sector Size (Bytes)
Z8F08xA	8 (8192)	16	0000H–1FFFH	1024
Z8F04xA	4 (4096)	8	0000H–0FFFH	512
Z8F02xA	2 (2048)	4	0000H–07FFH	512
Z8F01xA	1 (1024)	2	0000H–03FFH	512

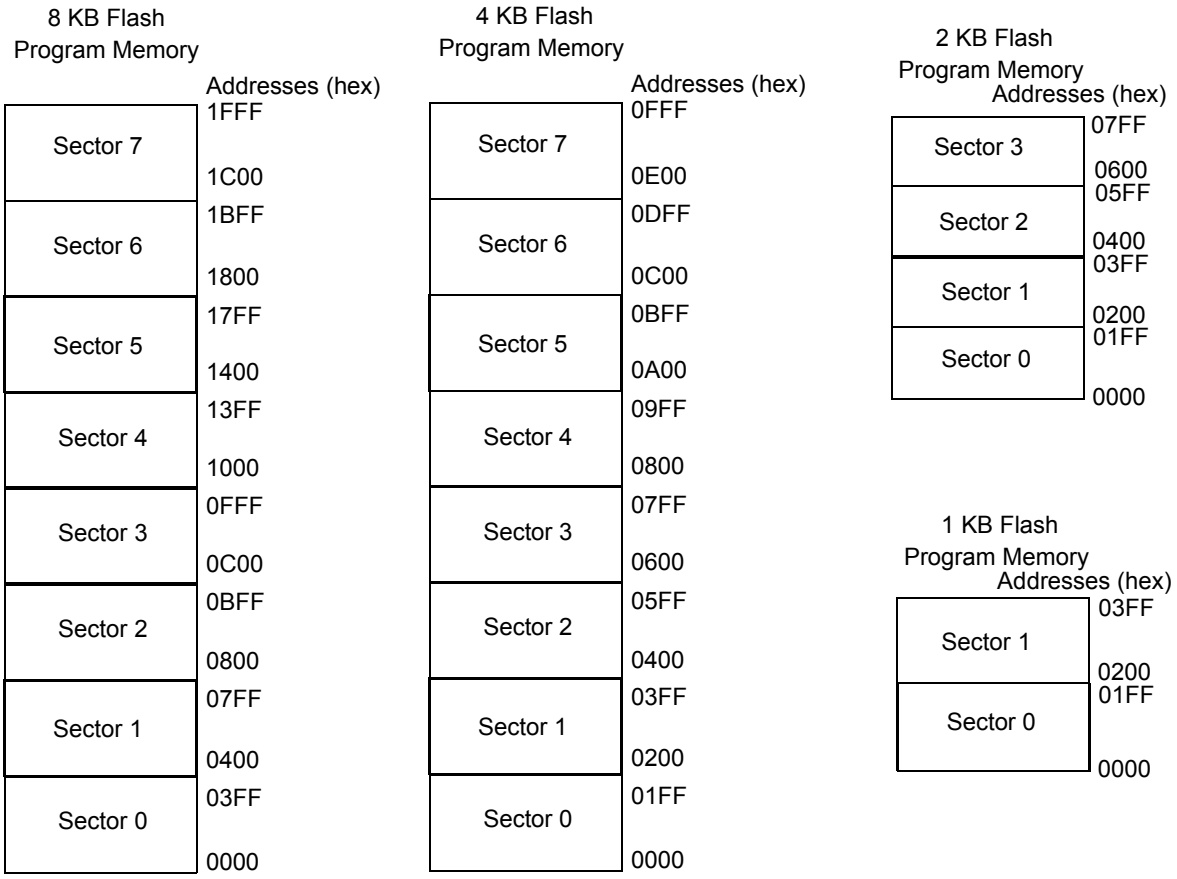

Figure 21. Flash Memory Arrangement

Flash Information Area

The Flash information area is separate from Program Memory and is mapped to the address range FE00H to FFFFH. This area is readable but cannot be erased or overwritten. Factory trim values for the analog peripherals are stored here. Factory calibration data for the ADC is also stored here.

Table 77. Flash Code Protection Using the Flash Option Bits

FWP	Flash Code Protection Description
0	Programming and erasing disabled for all of Flash Program Memory. In user code programming, Page Erase, and Mass Erase are all disabled. Mass Erase is available through the On-Chip Debugger.
1	Programming, Page Erase, and Mass Erase are enabled for all of Flash Program Memory.

Flash Code Protection Using the Flash Controller

At Reset, the Flash Controller locks to prevent accidental program or erasure of the Flash memory. To program or erase the Flash memory, first write the Page Select Register with the target page. Unlock the Flash Controller by making two consecutive writes to the Flash Control register with the values 73H and 8CH, sequentially. The Page Select Register must be rewritten with the target page. If the two Page Select writes do not match, the controller reverts to a locked state. If the two writes match, the selected page becomes active. See Figure 22 on page 144 for details.

After unlocking a specific page, you can enable either Page Program or Erase. Writing the value 95H causes a Page Erase only if the active page resides in a sector that is not protected. Any other value written to the Flash Control register locks the Flash Controller. Mass Erase is not allowed in the user code but only in through the Debug Port.

After unlocking a specific page, you can also write to any byte on that page. After a byte is written, the page remains unlocked, allowing for subsequent writes to other bytes on the same page. Further writes to the Flash Control Register cause the active page to revert to a locked state.

Sector Based Flash Protection

The final protection mechanism is implemented on a per-sector basis. The Flash memories of Z8 Encore!® devices are divided into at most 8 sectors. A sector is 1/8 of the total size of the Flash memory, unless this value is smaller than the page size, in which case the sector and page sizes are equal.

The Sector Protect register controls the protection state of each Flash sector. This register is shared with the Page Select Register. It is accessed by writing 73H followed by 5EH to the Flash controller. The next write to the Flash Control Register targets the Sector Protect Register.

The Sector Protect Register is initialized to 0 on reset, putting each sector into an unprotected state. When a bit in the Sector Protect Register is written to 1, the corresponding sector is no longer written or erased by the CPU. External Flash programming through

Non-Volatile Data Storage

The Z8 Encore! XP® F082A Series devices contain a non-volatile data storage (NVDS) element of up to 128 bytes. This memory can perform over 100,000 write cycles.

Operation

The NVDS is implemented by special purpose Zilog® software stored in areas of program memory, which are not user-accessible. These special-purpose routines use the Flash memory to store the data. The routines incorporate a dynamic addressing scheme to maximize the write/erase endurance of the Flash.

Note: *Different members of the Z8 Encore! XP F082A Series feature multiple NVDS array sizes. See Z8 Encore! XP® F082A Series Family Part Selection Guide on page 3 for details. Also the members containing 8 KB of Flash memory do not include the NVDS feature.*

NVDS Code Interface

Two routines are required to access the NVDS: a write routine and a read routine. Both of these routines are accessed with a CALL instruction to a pre-defined address outside of the user-accessible program memory. Both the NVDS address and data are single-byte values. Because these routines disturb the working register set, user code must ensure that any required working register values are preserved by pushing them onto the stack or by changing the working register pointer just prior to NVDS execution.

During both read and write accesses to the NVDS, interrupt service is NOT disabled. Any interrupts that occur during the NVDS execution must take care not to disturb the working register and existing stack contents or else the array may become corrupted. Disabling interrupts before executing NVDS operations is recommended.

Use of the NVDS requires 15 bytes of available stack space. Also, the contents of the working register set are overwritten.

For correct NVDS operation, the Flash Frequency Registers must be programmed based on the system clock frequency (see Flash Operation Timing Using the Flash Frequency Registers on page 145).

Byte Write

To write a byte to the NVDS array, the user code must first push the address, then the data byte onto the stack. The user code issues a `CALL` instruction to the address of the byte-write routine (0x10B3). At the return from the sub-routine, the write status byte

Table 104. NVDS Read Time (Continued)

Operation	Minimum Latency	Maximum Latency
Read (128 byte array)	883	7609
Write (16 byte array)	4973	5009
Write (64 byte array)	4971	5013
Write (128 byte array)	4984	5023
Illegal Read	43	43
Illegal Write	31	31

If NVDS read performance is critical to your software architecture, there are some things you can do to optimize your code for speed, listed in order from most helpful to least helpful:

- Periodically refresh all addresses that are used. The optimal use of NVDS in terms of speed is to rotate the writes evenly among all addresses planned to use, bringing all reads closer to the minimum read time. Because the minimum read time is much less than the write time, however, actual speed benefits are not always realized.
- Use as few unique addresses as possible: this helps to optimize the impact of refreshing as well as minimize the requirement for it.

Assembly Language Syntax

For proper instruction execution, eZ8 CPU assembly language syntax requires that the operands be written as ‘destination, source’. After assembly, the object code usually has the operands in the order ‘source, destination’, but ordering is opcode-dependent. The following instruction examples illustrate the format of some basic assembly instructions and the resulting object code produced by the assembler. This binary format must be followed if manual program coding is preferred or if you intend to implement your own assembler.

Example 1: If the contents of Registers 43H and 08H are added and the result is stored in 43H, the assembly syntax and resulting object code is:

Table 112. Assembly Language Syntax Example 1

Assembly Language Code	ADD	43H,	08H	(ADD dst, src)
Object Code	04	08	43	(OPC src, dst)

Example 2: In general, when an instruction format requires an 8-bit register address, that address can specify any register location in the range 0–255 or, using Escaped Mode Addressing, a Working Register R0–R15. If the contents of Register 43H and Working Register R8 are added and the result is stored in 43H, the assembly syntax and resulting object code is:

Table 113. Assembly Language Syntax Example 2

Assembly Language Code	ADD	43H,	R8	(ADD dst, src)
Object Code	04	E8	43	(OPC src, dst)

See the device-specific Product Specification to determine the exact register file range available. The register file size varies, depending on the device type.

eZ8 CPU Instruction Notation

In the eZ8 CPU Instruction Summary and Description sections, the operands, condition codes, status flags, and address modes are represented by a notational shorthand that is described in Table 114.

Table 121. Logical Instructions (Continued)

Mnemonic	Operands	Instruction
ORX	dst, src	Logical OR using Extended Addressing
XOR	dst, src	Logical Exclusive OR
XORX	dst, src	Logical Exclusive OR using Extended Addressing

Table 122. Program Control Instructions

Mnemonic	Operands	Instruction
BRK	—	On-Chip Debugger Break
BTJ	p, bit, src, DA	Bit Test and Jump
BTJNZ	bit, src, DA	Bit Test and Jump if Non-Zero
BTJZ	bit, src, DA	Bit Test and Jump if Zero
CALL	dst	Call Procedure
DJNZ	dst, src, RA	Decrement and Jump Non-Zero
IRET	—	Interrupt Return
JP	dst	Jump
JP cc	dst	Jump Conditional
JR	DA	Jump Relative
JR cc	DA	Jump Relative Conditional
RET	—	Return
TRAP	vector	Software Trap

Table 123. Rotate and Shift Instructions

Mnemonic	Operands	Instruction
BSWAP	dst	Bit Swap
RL	dst	Rotate Left
RLC	dst	Rotate Left through Carry
RR	dst	Rotate Right
RRC	dst	Rotate Right through Carry

Table 124. 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
XOR dst, src	dst ← dst XOR src	r	r	B2	–	*	*	0	–	–	2	3
		r	Ir	B3							2	4
		R	R	B4							3	3
		R	IR	B5							3	4
		R	IM	B6							3	3
		IR	IM	B7							3	4
XORX dst, src	dst ← dst XOR src	ER	ER	B8	–	*	*	0	–	–	4	3
		ER	IM	B9							4	3

Flags Notation:
* = Value is a function of the result of the operation.
– = Unaffected
X = Undefined
0 = Reset to 0
1 = Set to 1

Opcode Maps

A description of the opcode map data and the abbreviations are provided in Figure 30. Figure 31 and Figure 32 displays the eZ8 CPU instructions. Table 125 lists Opcode Map abbreviations.

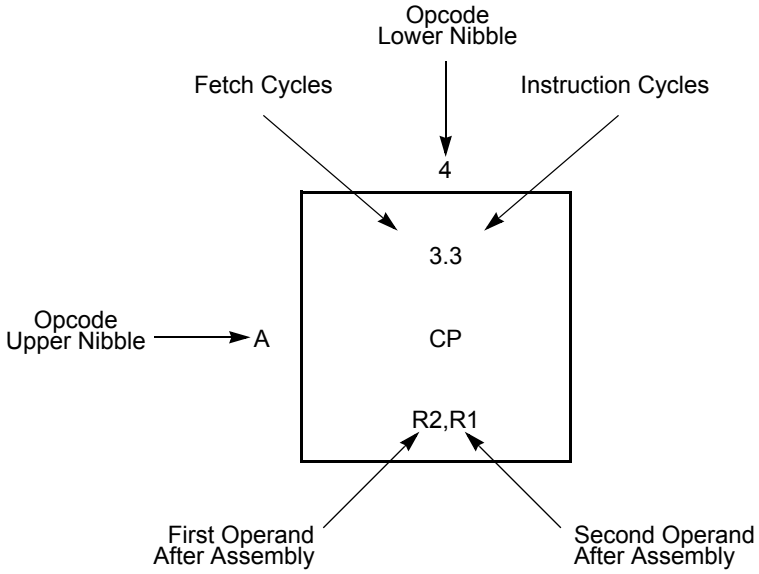

Figure 30. Opcode Map Cell Description

J