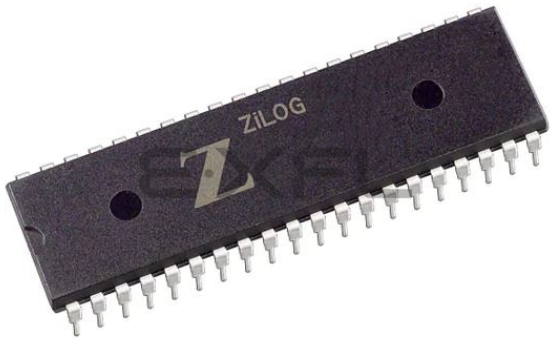


Welcome to E-XFL.COM

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	29
Program Memory Size	24KB (24K 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	Through Hole
Package / Case	40-DIP (0.620", 15.75mm)
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/zilog/z8f2421pm020ec

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.

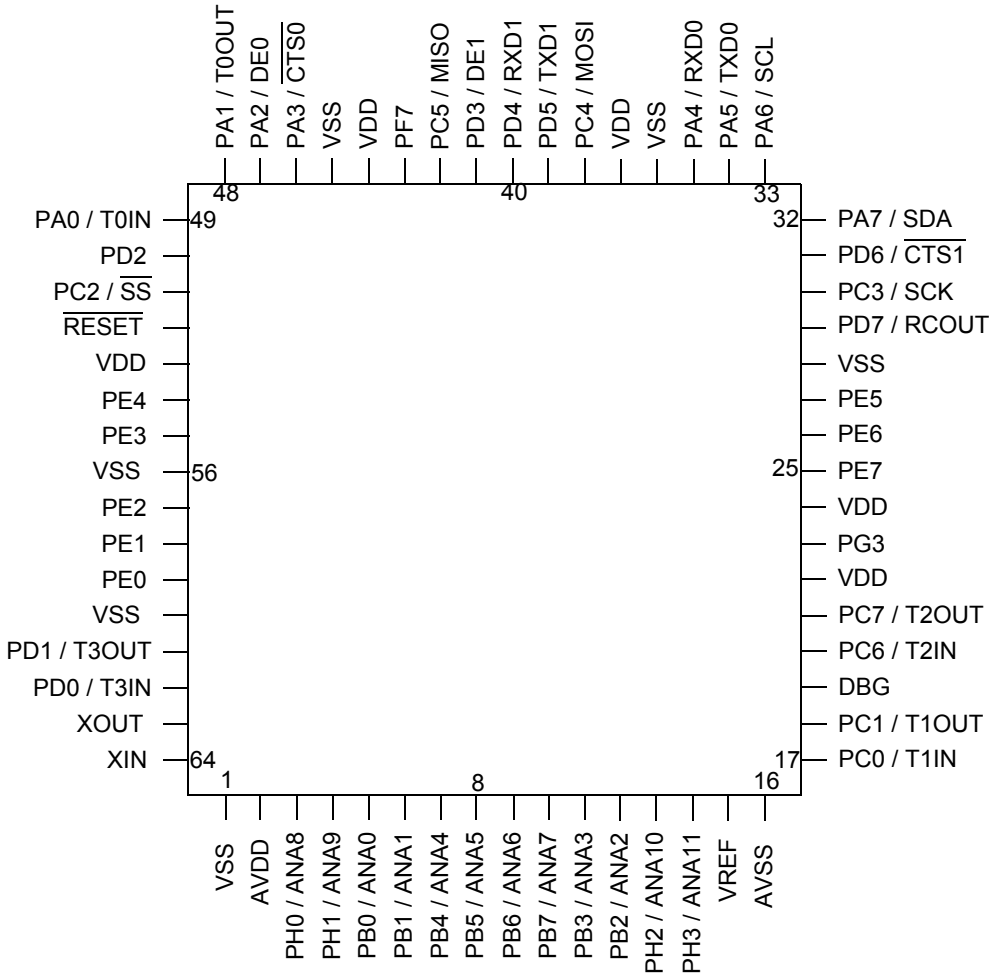
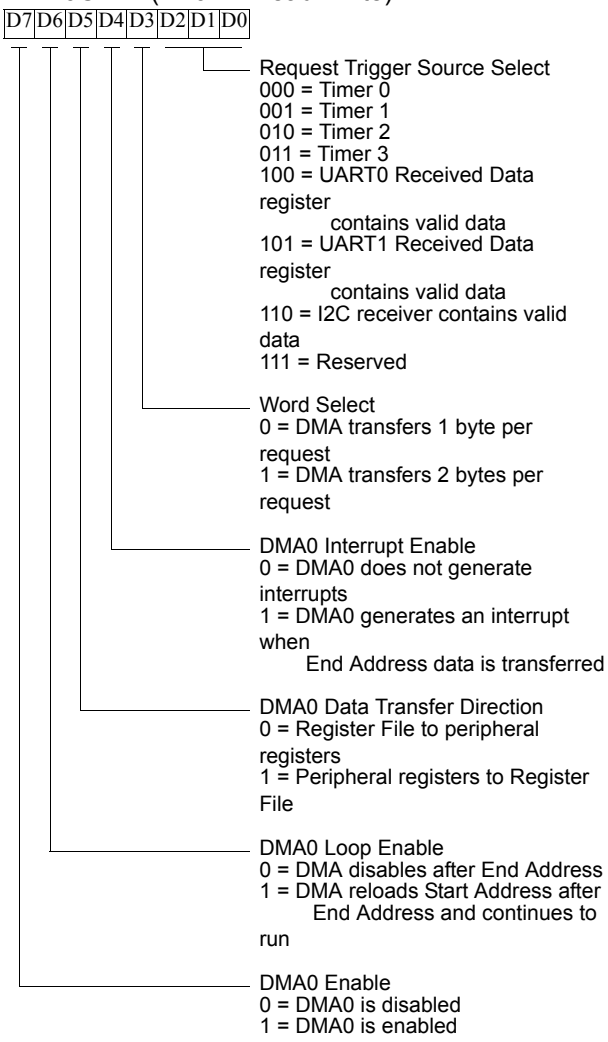


Figure 5. Z8 Encore! XP 64K Series Flash Microcontrollers in 64-Pin Low-Profile Quad Flat Package (LQFP)

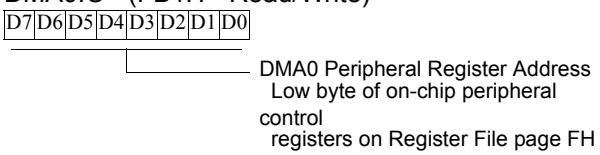
Table 6. Z8 Encore! XP 64K Series Flash Microcontrollers Information Area Map

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

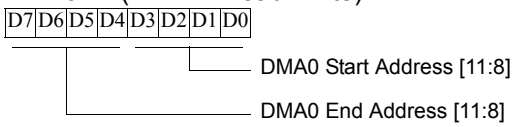
DMA0 Control
DMA0CTL (FB0H - Read/Write)



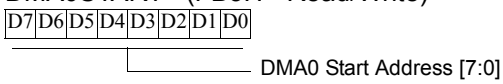
DMA0 I/O Address
DMA0IO (FB1H - Read/Write)



DMA0 Address High Nibble
DMA0H (FB2H - Read/Write)



DMA0 Start/Current Address Low Byte
DMA0START (FB3H - Read/Write)



DMA0 End Address Low Byte
DMA0END (FB4H - Read/Write)

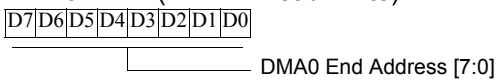


Table 11. Port Availability by Device and Package Type (Continued)

Device	Packages	Port A	Port B	Port C	Port D	Port E	Port F	Port G	Port H
Z8X4823	80-pin	[7:0]	[7:0]	[7:0]	[7:0]	[7:0]	[7:0]	[7:0]	[3:0]
Z8X6421	40-pin	[7:0]	[7:0]	[6:0]	[6:3, 1:0]	-	-	-	-
Z8X6421	44-pin	[7:0]	[7:0]	[7:0]	[6:0]	-	-	-	-
Z8X6422	64- and 68-pin	[7:0]	[7:0]	[7:0]	[7:0]	[7:0]	[7]	[3]	[3:0]
Z8X6423	80-pin	[7:0]	[7:0]	[7:0]	[7:0]	[7:0]	[7:0]	[7:0]	[3:0]

Architecture

Figure 10 displays a simplified block diagram of a GPIO port pin. In Figure 10, the ability to accommodate alternate functions and variable port current drive strength are not illustrated.

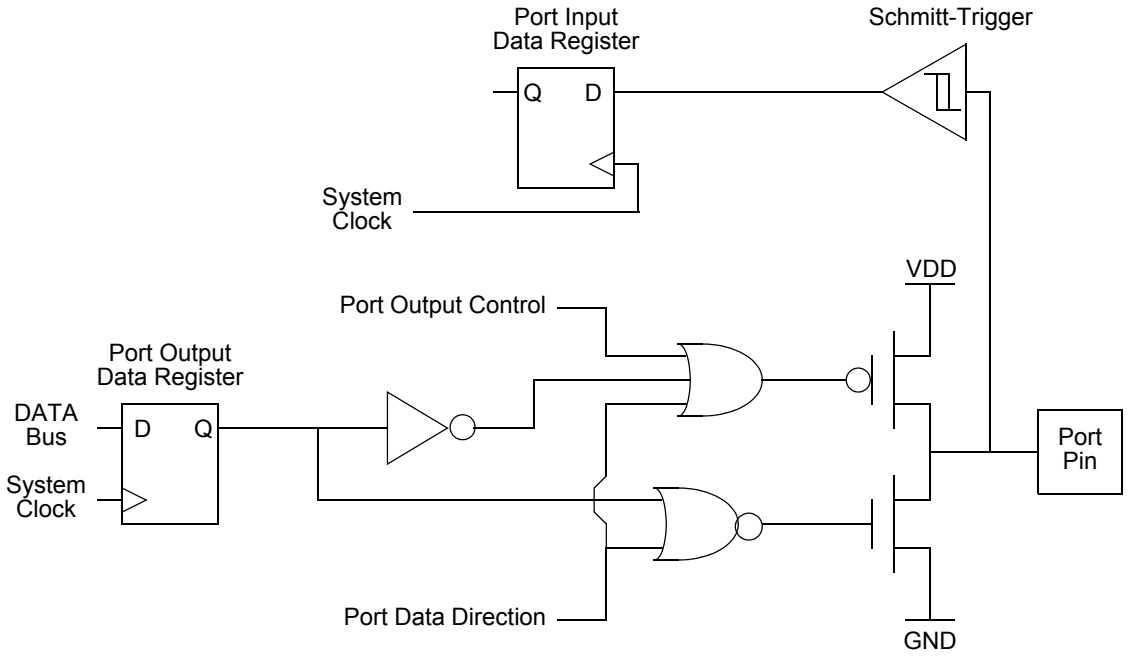


Figure 10. GPIO Port Pin Block Diagram

- Executing a Trap instruction.
- Illegal Instruction trap.

Interrupt Vectors and Priority

The interrupt controller supports three levels of interrupt priority. Level 3 is the highest priority, Level 2 is the second highest priority, and Level 1 is the lowest priority. If all of the interrupts were enabled with identical interrupt priority (all as Level 2 interrupts, for example), then interrupt priority would be assigned from highest to lowest as specified in Table 23 on page 68. Level 3 interrupts always have higher priority than Level 2 interrupts which, in turn, always have higher priority than Level 1 interrupts. Within each interrupt priority level (Level 1, Level 2, or Level 3), priority is assigned as specified in Table 23 on page 68. Reset, Watchdog Timer interrupt (if enabled), and Illegal Instruction Trap always have highest priority.

Interrupt Assertion

Interrupt sources assert their interrupt requests for only a single system clock period (single pulse). When the interrupt request is acknowledged by the eZ8 CPU, the corresponding bit in the Interrupt Request register is cleared until the next interrupt occurs. Writing a 0 to the corresponding bit in the Interrupt Request register likewise clears the interrupt request.

Caution: *The following style of coding to clear 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
AND r0, MASK
LDX IRQ0, r0
```

To avoid missing interrupts, the following style of coding to clear bits in the Interrupt Request 0 register is recommended:

Good coding style that avoids lost interrupt requests:

```
ANDX IRQ0, MASK
```

Software Interrupt Assertion

Program code can generate interrupts directly. Writing a 1 to the desired bit in the Interrupt Request register triggers an interrupt (assuming that interrupt is enabled). When the interrupt request is acknowledged by the eZ8 CPU, the bit in the Interrupt Request register is automatically cleared to 0.

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

Watchdog Timer

Overview

The Watchdog Timer (WDT) helps protect against corrupt or unreliable software, power faults, and other system-level problems which may place the Z8 Encore! XP into unsuitable operating states. The features of Watchdog Timer include:

- On-chip RC oscillator.
- A selectable time-out response.
- WDT Time-out response: Reset or interrupt.
- 24-bit programmable time-out value.

Operation

The Watchdog Timer (WDT) is a retriggerable one-shot timer that resets or interrupts the 64K Series devices when the WDT reaches its terminal count. The Watchdog Timer uses its own dedicated on-chip RC oscillator as its clock source. The Watchdog Timer has only two modes of operation—ON and OFF. Once enabled, it always counts and must be refreshed to prevent a time-out. An enable can be performed by executing the WDT instruction or by setting the WDT_AO Option Bit. The WDT_AO bit enables the Watchdog Timer to operate all the time, even if a WDT instruction has not been executed.

The Watchdog Timer is a 24-bit reloadable downcounter that uses three 8-bit registers in the eZ8™ CPU register space to set the reload value. The nominal WDT time-out period is given by the following equation:

$$\text{WDT Time-out Period (ms)} = \frac{\text{WDT Reload Value}}{10}$$

where the WDT reload value is the decimal value of the 24-bit value given by {WDTU[7:0], WDTH[7:0], WDTL[7:0]} and the typical Watchdog Timer RC oscillator frequency is 10 kHz. The Watchdog Timer cannot be refreshed once it reaches 000002H. The WDT Reload Value must not be set to values below 000004H. Table 47 provides information on approximate time-out delays for the minimum and maximum WDT reload values.

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).

REN—Receive Enable
This bit enables or disables the receiver.
0 = Receiver disabled.
1 = Receiver enabled.

CTSE—CTS Enable
0 = The $\overline{\text{CTS}}$ signal has no effect on the transmitter.
1 = The UART recognizes the $\overline{\text{CTS}}$ signal as an enable control from the transmitter.

PEN—Parity Enable
This bit enables or disables parity. Even or odd is determined by the PSEL bit. It is overridden by the MPEN bit.
0 = Parity is disabled.
1 = The transmitter sends data with an additional parity bit and the receiver receives an additional parity bit.

PSEL—Parity Select
0 = Even parity is transmitted and expected on all received data.
1 = Odd parity is transmitted and expected on all received data.

SBRK—Send Break
This bit pauses or breaks data transmission. Sending a break interrupts any transmission in progress, so ensure that the transmitter has finished sending data before setting this bit.
0 = No break is sent.
1 = The output of the transmitter is zero.

STOP—Stop Bit Select
0 = The transmitter sends one stop bit.
1 = The transmitter sends two stop bits.

LBEN—Loop Back Enable
0 = Normal operation.
1 = All transmitted data is looped back to the receiver.

Table 57. UART Control 1 Register (U*x*CTL1)

BITS	7	6	5	4	3	2	1	0
FIELD	MPMD[1]	MPEN	MPMD[0]	MPBT	DEPOL	BRGCTL	RDAIRQ	IREN
RESET	0							
R/W	R/W							
ADDR	F43H and F4BH							

MPMD[1:0]—MULTIPROCESSOR Mode
If MULTIPROCESSOR (9-bit) mode is enabled,
00 = The UART generates an interrupt request on all received bytes (data and address).

Table 59. UART Baud Rate High Byte Register (U*x*BRH)

BITS	7	6	5	4	3	2	1	0
FIELD	BRH							
RESET	1							
R/W	R/W							
ADDR	F46H and F4EH							

Table 60. UART Baud Rate Low Byte Register (U*x*BRL)

BITS	7	6	5	4	3	2	1	0
FIELD	BRL							
RESET	1							
R/W	R/W							
ADDR	F47H and F4FH							

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

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

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.

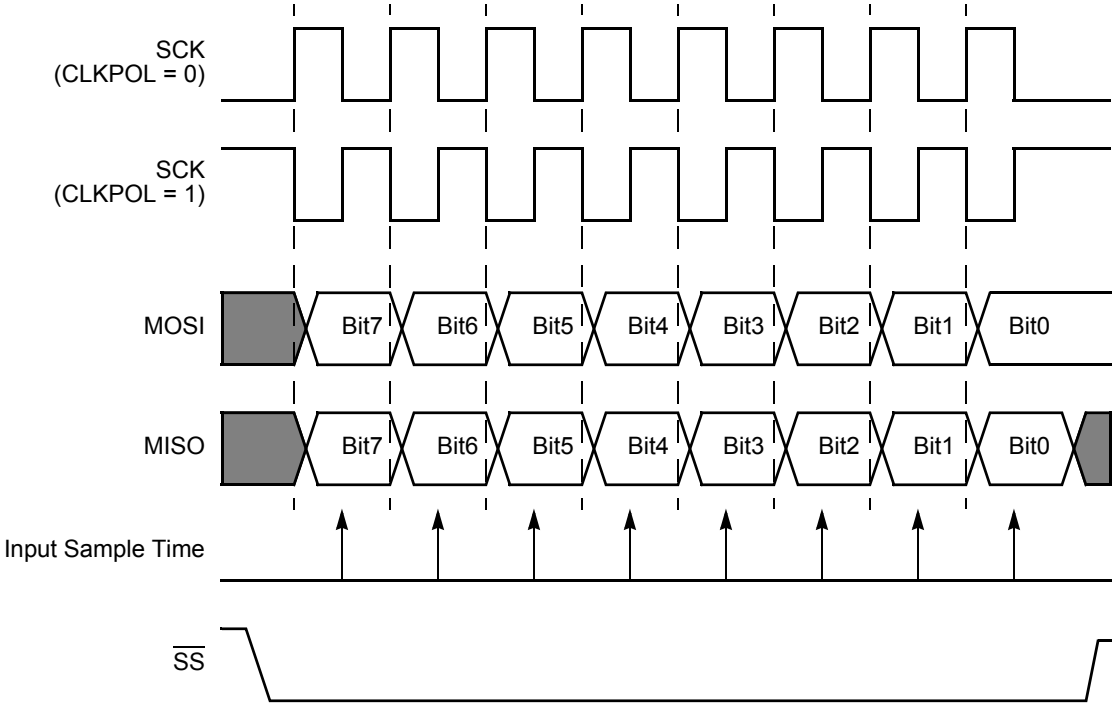


Figure 26. SPI Timing When PHASE **is 1**

Multi-Master Operation

In a multi-master SPI system, all SCK pins are tied together, all MOSI pins are tied together and all MISO pins are tied together. All SPI pins must then be configured in OPEN-DRAIN mode to prevent bus contention. At any one time, only one SPI device is configured as the Master and all other SPI devices on the bus are configured as Slaves. The Master enables a single Slave by asserting the $\overline{SS}$ pin on that Slave only. Then, the single Master drives data out its SCK and MOSI pins to the SCK and MOSI pins on the Slaves (including those which are not enabled). The enabled Slave drives data out its MISO pin to the MISO Master pin.

For a Master device operating in a multi-master system, if the $\overline{SS}$ pin is configured as an input and is driven Low by another Master, the COL bit is set to 1 in the SPI Status Register. The COL bit indicates the occurrence of a multi-master collision (mode fault error condition).

Slave Operation

The SPI block is configured for SLAVE mode operation by setting the SPIEN bit to 1 and the MMEN bit to 0 in the SPICTL register and setting the SSIO bit to 0 in the SPIMODE

register. The IRQE, PHASE, CLKPOL, WOR bits in the SPICTL register and the NUMBITS field in the SPIMODE register must be set to be consistent with the other SPI devices. The STR bit in the SPICTL register may be used if desired to force a "startup" interrupt. The BIRQ bit in the SPICTL register and the SSV bit in the SPIMODE register are not used in SLAVE mode. The SPI baud rate generator is not used in SLAVE mode so the SPIBRH and SPIBRL registers need not be initialized.

If the slave has data to send to the master, the data must be written to the SPIDAT register before the transaction starts (first edge of SCK when $\overline{SS}$ is asserted). If the SPIDAT register is not written prior to the slave transaction, the MISO pin outputs whatever value is currently in the SPIDAT register.

Due to the delay resulting from synchronization of the SPI input signals to the internal system clock, the maximum SPICLK baud rate that can be supported in SLAVE mode is the system clock frequency (XIN) divided by 8. This rate is controlled by the SPI master.

Error Detection

The SPI contains error detection logic to support SPI communication protocols and recognize when communication errors have occurred. The SPI Status register indicates when a data transmission error has been detected.

Overrun (Write Collision)

An overrun error (write collision) indicates a write to the SPI Data register was attempted while a data transfer is in progress (in either MASTER or SLAVE modes). An overrun sets the OVR bit in the SPI Status register to 1. Writing a 1 to OVR clears this error Flag. The data register is not altered when a write occurs while data transfer is in progress.

Mode Fault (Multi-Master Collision)

A mode fault indicates when more than one Master is trying to communicate at the same time (a multi-master collision). The mode fault is detected when the enabled Master's $\overline{SS}$ pin is asserted. A mode fault sets the COL bit in the SPI Status register to 1. Writing a 1 to COL clears this error Flag.

Slave Mode Abort

In SLAVE mode of operation if the $\overline{SS}$ pin deasserts before all bits in a character have been transferred, the transaction is aborted. When this condition occurs the ABT bit is set in the SPISTAT register as well as the IRQ bit (indicating the transaction is complete). The next time $\overline{SS}$ asserts, the MISO pin outputs SPIDAT[7], regardless of where the previous transaction left off. Writing a 1 to ABT clears this error Flag.

SPI Interrupts

When SPI interrupts are enabled, the SPI generates an interrupt after character transmission/reception completes in both MASTER and SLAVE modes. A character can be

DMAA_ADDR—DMA_ADC Address
These bits specify the seven most-significant bits of the 12-bit Register File addresses used for storing the ADC output data. The ADC Analog Input Number defines the five least-significant bits of the Register File address. Full 12-bit address is {DMAA_ADDR[7:1], 4-bit ADC Analog Input Number, 0}.

Reserved
This bit is reserved and must be 0.

DMA_ADC Control Register

The DMA_ADC Control register (Table 84 on page 172) enables and sets options (DMA enable and interrupt enable) for ADC operation.

Table 84. DMA_ADC Control Register (DMAACTL)

BITS	7	6	5	4	3	2	1	0
FIELD	DAEN	IRQEN	Reserved		ADC_IN			
RESET	0							
R/W	R/W							
ADDR	FBEH							

DAEN—DMA_ADC Enable
0 = DMA_ADC is disabled and the ADC Analog Input Number (ADC_IN) is reset to 0.
1 = DMA_ADC is enabled.

IRQEN—Interrupt Enable
0 = DMA_ADC does not generate any interrupts.
1 = DMA_ADC generates an interrupt after transferring data from the last ADC Analog Input specified by the ADC_IN field.

Reserved
These bits are reserved and must be 0.

ADC_IN—ADC Analog Input Number
These bits set the number of ADC Analog Inputs to be used in the continuous update (data conversion followed by DMA data transfer). The conversion always begins with ADC Analog Input 0 and then progresses sequentially through the other selected ADC Analog Inputs.
0000 = ADC Analog Input 0 updated.
0001 = ADC Analog Inputs 0-1 updated.
0010 = ADC Analog Inputs 0-2 updated.
0011 = ADC Analog Inputs 0-3 updated.
0100 = ADC Analog Inputs 0-4 updated.

Information Area

Table 91 describes the 64K 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 91. Z8 Encore! XP 64K Series Flash Microcontrollers 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, via 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).

On-Chip Debugger

Overview

The 64K Series products contain an integrated On-Chip Debugger (OCD) that provides advanced debugging features including:

- Reading and writing of the Register File
- Reading and writing of Program and Data Memory
- Setting of Breakpoints
- Execution of eZ8 CPU instructions

Architecture

The On-Chip Debugger consists of four primary functional blocks: transmitter, receiver, auto-baud generator, and debug controller. Figure 36 displays the architecture of the On-Chip Debugger.

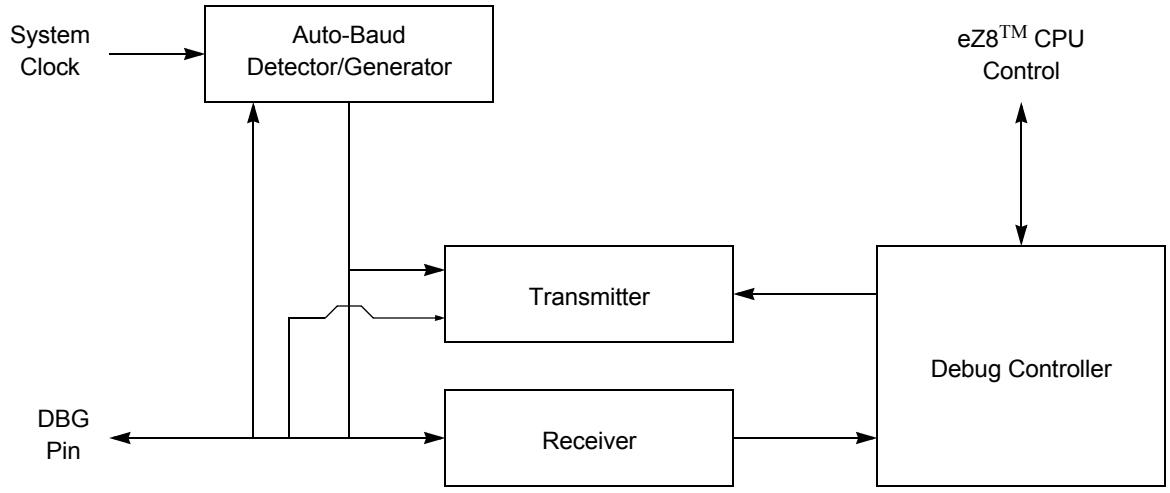


Figure 36. On-Chip Debugger Block Diagram

SPI Slave Mode Timing

Figure 54 and Table 118 provide timing information for the SPI slave mode pins. Timing is shown with SCK rising edge used to source MISO output data, SCK falling edge used to sample MOSI input data.

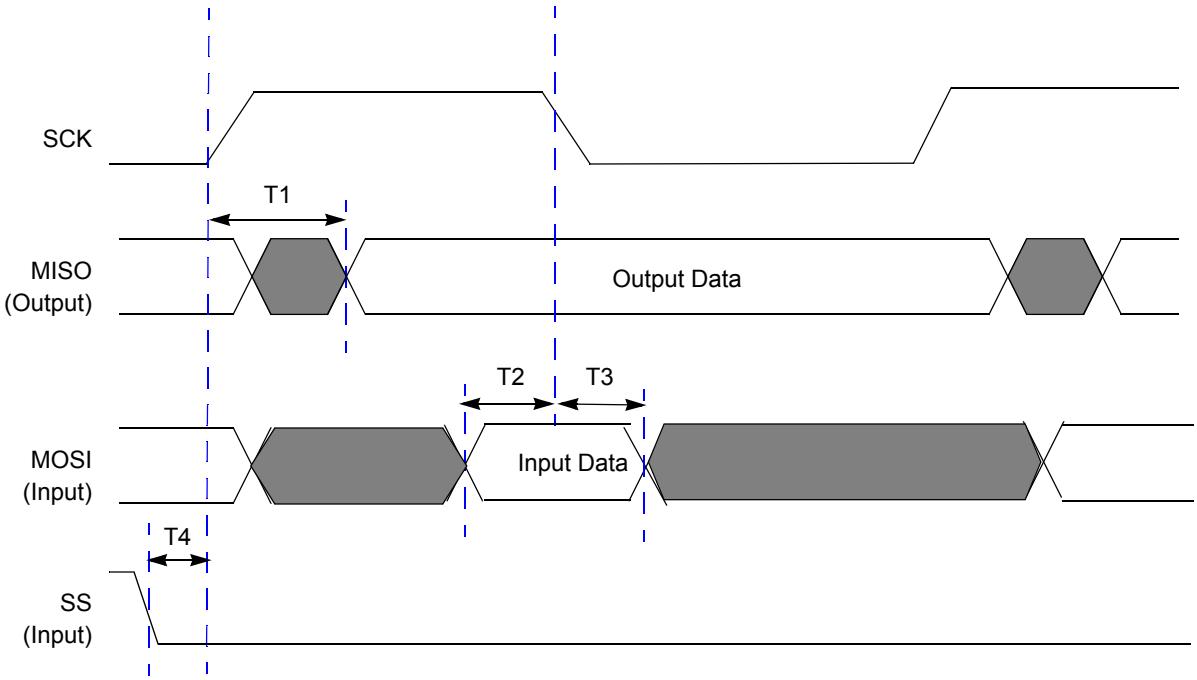


Figure 54. SPI Slave Mode Timing

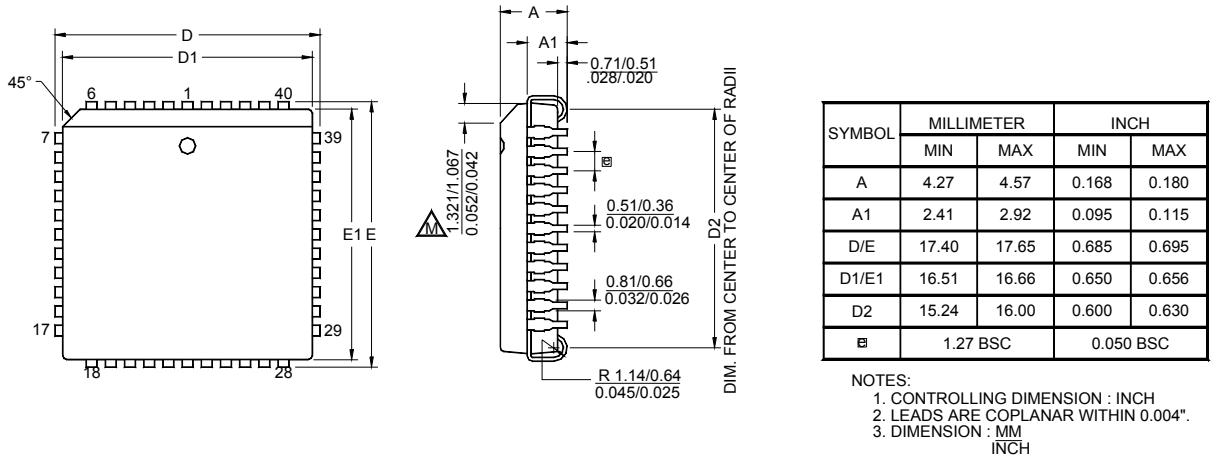
Table 118. SPI Slave Mode Timing

		Delay (ns)	
Parameter	**Abbreviation**	**Minimum**	**Maximum**
SPI Slave			
T_1	SCK (transmit edge) to MISO output Valid Delay	2 * Xin period	3 * Xin period + 20 nsec
T_2	MOSI input to SCK (receive edge) Setup Time	0	
T_3	MOSI input to SCK (receive edge) Hold Time	3 * Xin period	
T_4	SS input assertion to SCK setup	1 * Xin period	

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
SWAP dst	dst[7:4] ↔ dst[3:0]	R		F0	X	*	*	X	-	-	2	2
		IR		F1							2	3
TCM dst, src	(NOT dst) AND src	r	r	62	-	*	*	0	-	-	2	3
		r	Ir	63							2	4
		R	R	64							3	3
		R	IR	65							3	4
		R	IM	66							3	3
		IR	IM	67							3	4
TCMX dst, src	(NOT dst) AND src	ER	ER	68	-	*	*	0	-	-	4	3
		ER	IM	69							4	3
TM dst, src	dst AND src	r	r	72	-	*	*	0	-	-	2	3
		r	Ir	73							2	4
		R	R	74							3	3
		R	IR	75							3	4
		R	IM	76							3	3
		IR	IM	77							3	4
TMX dst, src	dst AND src	ER	ER	78	-	*	*	0	-	-	4	3
		ER	IM	79							4	3
TRAP Vector	SP ← SP – 2 @SP ← PC SP ← SP – 1 @SP ← FLAGS PC ← @Vector		Vector	F2	-	-	-	-	-	-	2	6
WDT				5F	-	-	-	-	-	-	1	2

Figure 64 displays the 44-pin Plastic Lead Chip Carrier (PLCC) package available for the Z8X1621, Z8X2421, Z8X3221, Z8X4821, and Z8X6421 devices.

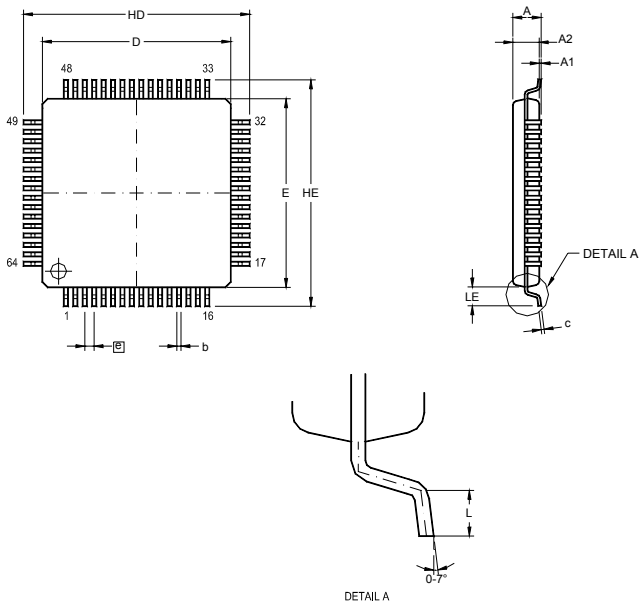


SYMBOL	MILLIMETER		INCH	
	MIN	MAX	MIN	MAX
A	4.27	4.57	0.168	0.180
A1	2.41	2.92	0.095	0.115
D/E	17.40	17.65	0.685	0.695
D1/E1	16.51	16.66	0.650	0.656
D2	15.24	16.00	0.600	0.630
e	1.27 BSC		0.050 BSC	

NOTES:
1. CONTROLLING DIMENSION : INCH
2. LEADS ARE COPLANAR WITHIN 0.004".
3. DIMENSION : MM / INCH

Figure 64. 44-Lead Plastic Lead Chip Carrier Package (PLCC)

Figure 64 displays the 64-pin Low-Profile Quad Flat Package (LQFP) available for the Z8X1622, Z8X2422, Z8X3222, Z8X4822, and Z8X6422 devices.



SYMBOL	MILLIMETER		INCH	
	MIN	MAX	MIN	MAX
A	1.40	1.60	0.055	0.063
A1	0.05	0.15	0.002	0.006
A2	1.35	1.45	0.053	0.057
b	0.17	0.27	0.007	0.011
c	0.09	0.20	0.004	0.008
HD	11.75	12.25	0.463	0.482
D	9.90	10.10	0.390	0.398
HE	11.75	12.25	0.463	0.482
E	9.90	10.10	0.390	0.398
e	0.50 BSC		0.0197 BSC	
L	0.45	0.75	0.018	0.030
LE	1.00 REF		0.039 REF	

1. CONTROLLING DIMENSIONS : mm
2. MAX. COPLANARITY : 0.10mm / 0.004"

Figure 65. 64-Lead Low-Profile Quad Flat Package (LQFP)

For technical and customer support, hardware and software development tools, refer to the Zilog® website at www.zilog.com. The latest released version of ZDS can be downloaded from this website.

Part Number Suffix Designations