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	4KB (4K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	1K x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 3.6V
Data Converters	A/D 5x10b
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/z8f0422sj020sc00tr

Introduction

Zilog's Z8 Encore! XP® MCU product family is a line of Zilog microcontrollers based on the 8-bit eZ8 CPU. Z8 Encore! XP® F0822 Series, hereafter referred as Z8 Encore! XP or the 8K Series adds Flash memory to Zilog's extensive line of 8-bit microcontrollers. The Flash in-circuit programming allows faster development time and program changes in the field. The new eZ8 CPU is upward-compatible with the existing Z8® instructions. The rich peripheral set of Z8 Encore! XP makes it suitable for a variety of applications including motor control, security systems, home appliances, personal electronic devices, and sensors.

Features

The features of Z8 Encore! XP MCU product family include:

- 20 MHz eZ8 CPU core
- Up to 8 KB Flash with in-circuit programming capability
- 1 KB Register RAM
- Optional 2- to 5-channel, 10-bit Analog-to-Digital Converter (ADC)
- Full-duplex 9-bit Universal Asynchronous Receiver/Transmitter (UART) with bus transceiver Driver Enable Control
- Inter-Integrated Circuit (I^2C)
- Serial Peripheral Interface (SPI)
- Infrared Data Association (IrDA)-compliant infrared encoder/decoders
- Two 16-bit timers with Capture, Compare, and PWM capability
- Watchdog Timer (WDT) with internal RC oscillator
- 11 to 19 Input/Output pins depending upon package
- Up to 19 interrupts with configurable priority
- On-Chip Debugger (OCD)
- Voltage Brownout (VBO) protection
- Power-On Reset (POR)
- Crystal oscillator with three power settings and RC oscillator option

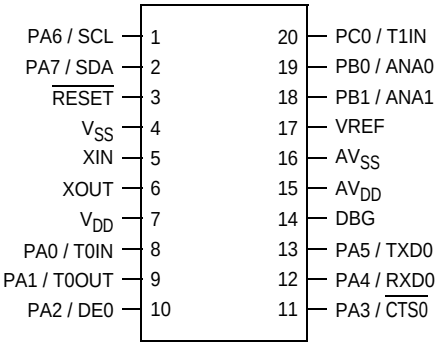


Figure 2. Z8F0821 and Z8F0421 in 20-Pin SSOP and PDIP Packages

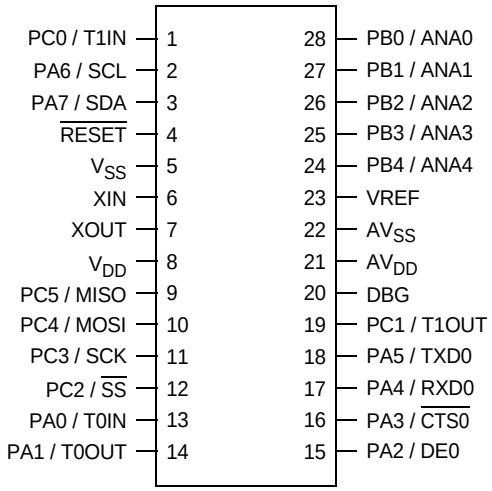


Figure 3. Z8F0822 and Z8F0422 in 28-Pin SOIC and PDIP Packages

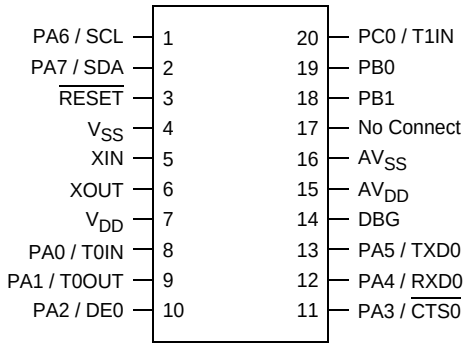


Figure 4. Z8F0811 and Z8F0411 in 20-Pin SSOP and PDIP Packages

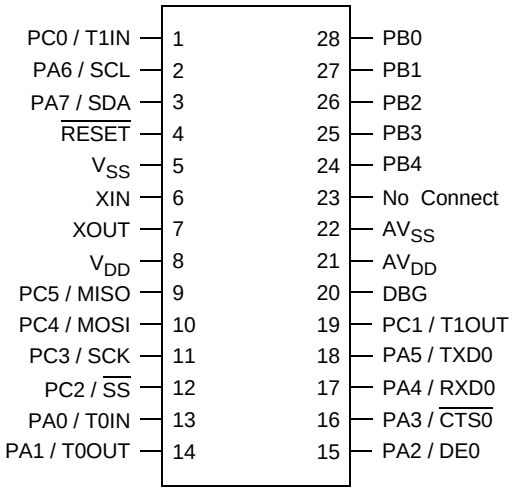


Figure 5. Z8F0812 and Z8F0412 in 28-Pin SOIC and PDIP Packages

Signal Descriptions

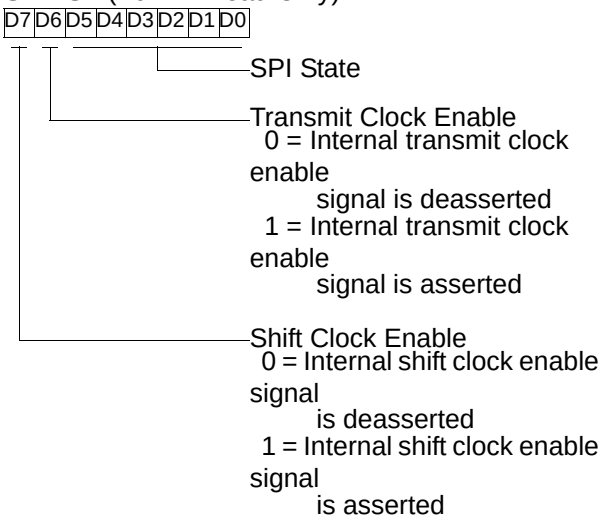
Table 3 describes Z8 Encore! XP® F0822 Series signals. See Pin Configurations on page 7 to determine the signals available for the specific package styles

.

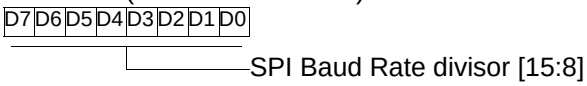
Table 3. Signal Descriptions

Signal Mnemonic	I/O	Description
General-Purpose I/O Ports A-H		
PA[7:0]	I/O	**Port C**—These pins are used for general-purpose I/O and supports 5 V-tolerant inputs.
PB[4:0]	I/O	**Port B**—These pins are used for general-purpose I/O.
PC[5:0]	I/O	**Port C**—These pins are used for general-purpose I/O and support 5 V-tolerant inputs.
I^2C Controller		
SCL	I/O	**Serial Clock**—This open-drain pin clocks data transfers in accordance with the I^2C standard protocol. This pin is multiplexed with a GPIO pin. When the GPIO pin is configured for alternate function to enable the SCL function, this pin is open-drain.
SDA	I/O	**Serial Data**—This open-drain pin transfers data between the I^2C and a slave. This pin is multiplexed with a GPIO pin. When the GPIO pin is configured for alternate function to enable the SDA function, this pin is open-drain.

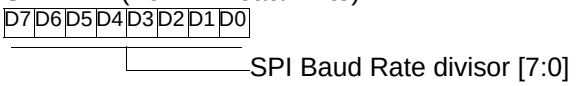
SPI Diagnostic State
SPIDST (F64H - Read Only)



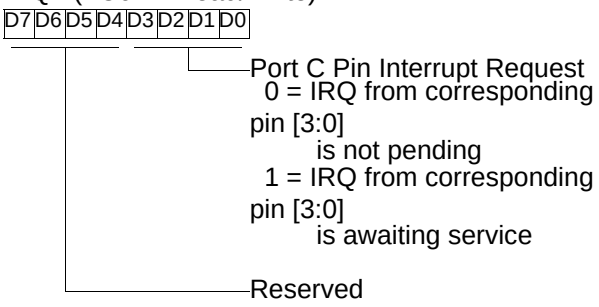
SPI Baud Rate Generator High Byte
SPIBRH (F66H - Read/Write)



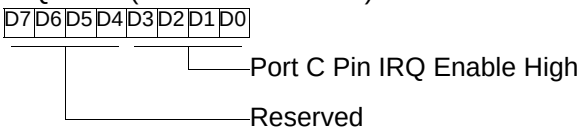
SPI Baud Rate Generator Low Byte
SPIBRL (F67H - Read/Write)



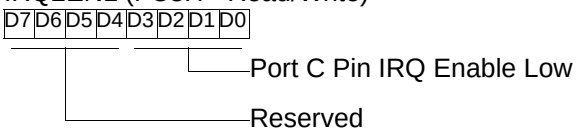
Interrupt Request 2
IRQ2 (FC6H - Read/Write)



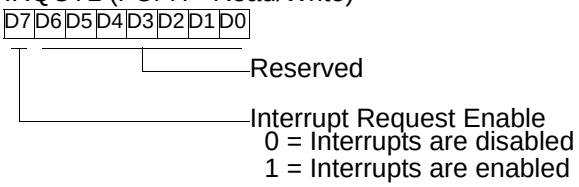
IRQ2 Enable High Bit
IRQ2ENH (FC7H - Read/Write)



IRQ2 Enable Low Bit
IRQ2ENL (FC8H - Read/Write)



Interrupt Control
IRQCTL (FCFH - Read/Write)



Reset and Stop Mode Recovery

The Reset Controller within the Z8 Encore! XP® F0822 Series controls Reset and Stop Mode Recovery operation. In typical operation, the following events cause a Reset to occur:

- Power-On Reset (POR)
- Voltage Brownout
- WDT time-out (when configured through the WDT_RES Option Bit to initiate a Reset)
- External $\overline{\text{RESET}}$ pin assertion
- On-Chip Debugger initiated Reset (OCDCTL[0] set to 1)

When the Z8 Encore! XP F0822 Series device is in STOP mode, a Stop Mode Recovery is initiated by any of the following events:

- WDT time-out
- GPIO Port input pin transition on an enabled Stop Mode Recovery source
- DBG pin driven Low

Reset Types

Z8 Encore! XP F0822 Series provides two types of reset operation (System Reset and Stop Mode Recovery). The type of reset is a function of both the current operating mode of the Z8 Encore! XP F0822 Series device and the source of the Reset. Table 8 lists the types of Resets and their operating characteristics.

Table 8. Reset and Stop Mode Recovery Characteristics and Latency

	Reset Characteristics and Latency		
Reset Type	**Control Registers**	**eZ8 CPU**	**Reset Latency (Delay)**
System Reset	Reset (as applicable)	Reset	66 WDT Oscillator cycles + 16 System Clock cycles
Stop Mode Recovery	Unaffected, except WDT_CTL register	Reset	66 WDT Oscillator cycles + 16 System Clock cycles

1 = The port pin is configured as a Stop Mode Recovery source. Any logic transition on this pin during STOP mode initiates Stop Mode Recovery.

Port A–C Pull-up Enable Sub-Registers

The Port A–C Pull-Up Enable sub-register (Table 21) is accessed through the Port A–C Control Register by writing 06H to the Port A–C Address Register. Setting the bits in the Port A–C Pull-Up Enable sub-registers enables a weak internal resistive pull-up on the specified Port pins.

Table 21. Port A–C Pull-Up Enable Sub-Registers

BITS	7	6	5	4	3	2	1	0
FIELD	PPUE7	PPUE6	PPUE5	PPUE4	PPUE3	PPUE2	PPUE1	PPUE0
RESET	0							
R/W	R/W							
ADDR	If 06H in Port A–C Address Register, accessible through the Port A–C Control Register							

PPUE[7:0]—Port Pull-up Enabled

0 = The weak pull-up on the Port pin is disabled.
1 = The weak pull-up on the Port pin is enabled.

Port A–C Input Data Registers

Reading from the Port A–C Input Data Registers (Table 22) returns the sampled values from the corresponding port pins. The Port A–C Input Data Registers are Read-only.

Table 22. Port A–C Input Data Registers (PxIN)

BITS	7	6	5	4	3	2	1	0
FIELD	PIN7	PIN6	PIN5	PIN4	PIN3	PIN2	PIN1	PIN0
RESET	X							
R/W	R							
ADDR	FD2H, FD6H, FDAH							

PIN[7:0]—Port Input Data

Sampled data from the corresponding port pin input.
0 = Input data is logical 0 (Low).
1 = Input data is logical 1 (High).

Port A–C Output Data Register

The Port A–C Output Data Register (Table 23) controls the output data to the pins.

Table 23. Port A–C Output Data Register (P*x*OUT)

BITS	7	6	5	4	3	2	1	0
FIELD	POUT7	POUT6	POUT5	POUT4	POUT3	POUT2	POUT1	POUT0
RESET	0							
R/W	R/W							
ADDR	FD3H, FD7H, FDBH							

POUT[7:0]—Port Output Data

These bits contain the data to be driven to the port pins. The values are only driven if the corresponding pin is configured as an output and the pin is not configured for alternate function operation.

0 = Drive a logical 0 (Low).

1 = Drive a logical 1 (High). High value is not driven if the drain has been disabled by setting the corresponding Port Output Control Register bit to 1.

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 the interrupts were enabled with identical interrupt priority (all as Level 2 interrupts), then interrupt priority would be assigned from highest to lowest as specified in Table 24. 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 24. Reset, WDT 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 received between execution of the first LDX command and the last LDX command is lost.*

Poor coding style resulting in lost interrupt requests:

```
LDX r0, IRQ0
AND r0, MASK
LDX IRQ0, r0
```

Note: *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 generates interrupts directly. Writing 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.

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

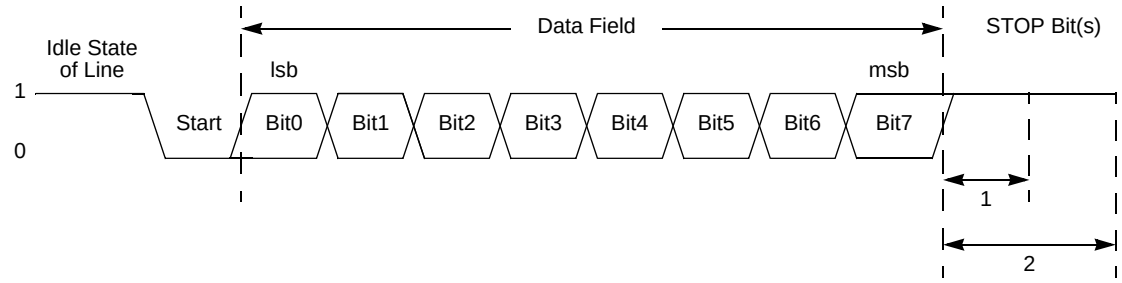


Figure 12. UART Asynchronous Data Format without Parity

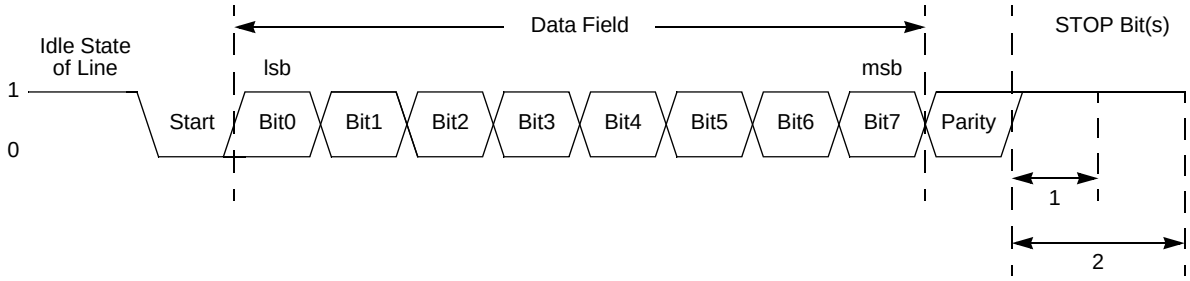


Figure 13. UART Asynchronous Data Format with Parity

Transmitting Data using Polled Method

Follow the steps below to transmit data using polled method of operation:

1. Write to the UART Baud Rate High Byte 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. If MULTIPROCESSOR mode is required, write to the UART Control 1 Register to enable multiprocessor (9-bit) mode functions.
   – Set the Multiprocessor Mode Select (MPEN) to enable MULTIPROCESSOR mode.
4. Write to the UART Control 0 Register to:
   – Set the transmit enable bit (TEN) to enable the UART for data transmission
   – If parity is required, and MULTIPROCESSOR mode is not enabled, set the parity enable bit (PEN) and select either even or odd parity (PSEL).
   – Set or clear the CTSE bit to enable or disable control from the remote receiver using the $\overline{\text{CTS}}$ pin.

Receiving Data Using 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 required 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 required 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 required 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! XP 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 required, 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 ISR 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 due to 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].
3. Clear the UART Receiver Interrupt in the applicable Interrupt Request Register.
4. Execute the IRET instruction to return from the ISR and await more data.

multi-node network. The following MULTIPROCESSOR modes are available in hardware:

- Interrupt on all address bytes.
- Interrupt on matched address bytes and correctly framed data bytes.
- Interrupt only on correctly framed data bytes.

These modes are selected with MPMD[1:0] in the UART Control 1 Register. For all MULTIPROCESSOR modes, bit MPEN of the UART Control 1 Register must be set to 1.

The first scheme is enabled by writing 01b to MPMD[1:0]. In this mode, all incoming address bytes cause an interrupt, while data bytes never cause an interrupt. The ISR must manually check the address byte that caused triggered the interrupt. If it matches the UART address, the software should clear MPMD[0]. At this point, each new incoming byte interrupts the CPU. The software is then responsible for determining the end-of-frame. It checks for the end-of-frame by reading the MPRX bit of the UART Status 1 Register for each incoming byte. If MPRX=1, then a new frame begins. If the address of this new frame is different from the UART's address, then MPMD[0] must be set to 1 causing the UART interrupts to go inactive until the next address byte. If the new frame's address matches the UART's address, then the data in the new frame should be processed as well.

The second scheme is enabled by setting MPMD[1:0] to 10b and writing the UART's address into the UART Address Compare Register. This mode introduces more hardware control, interrupting only on frames that match the UART's address. When an incoming address byte does not match the UART's address, it is ignored. All successive data bytes in this frame are also ignored. When a matching address byte occurs, an interrupt is issued and further interrupts occur on each successive data byte. The first data byte in the frame contains the NEWFRM=1 in the UART Status 1 Register. When the next address byte occurs, the hardware compares it to the UART's address. If there is a match, the interrupts continue and the NEWFRM bit is set for the first byte of the new frame. If there is no match, then the UART ignores all incoming bytes until the next address match.

The third scheme is enabled by setting MPMD[1:0] to 11b and by writing the UART's address into the UART Address Compare Register. This mode is identical to the second scheme, except that there are no interrupts on address bytes. The first data byte of each frame is still accompanied by a NEWFRM assertion.

External Driver Enable

The UART provides a Driver Enable (DE) signal for off-chip bus transceivers. This feature reduces the software overhead associated with using a GPIO pin to control the transceiver when communicating on a multi-transceiver bus, such as RS-485.

Driver Enable is an active High signal that envelopes the entire transmitted data frame including parity and STOP bits as displayed in Figure15 on page 97. The Driver Enable signal asserts when a byte is written to the UART Transmit Data Register. The Driver

Table 58. UART Address Compare Register (U0ADDR)

BITS	7	6	5	4	3	2	1	0
FIELD	COMP_ADDR							
RESET	0							
R/W	R/W							
ADDR	F45H							

COMP_ADDR—Compare Address
This 8-bit value is compared to the incoming address bytes.

UART Baud Rate High and Low Byte Registers

The UART Baud Rate High and Low Byte registers (Table 59 and Table 60) combine to create a 16-bit baud rate divisor value (BRG[15:0]) that sets the data transmission rate (baud rate) of the UART.

Table 59. UART Baud Rate High Byte Register (U0BRH)

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

Table 60. UART Baud Rate Low Byte Register (U0BRL)

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

The UART data rate is calculated using the following equation:

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

of minus four baud rate clocks to plus eight baud rate clocks around the expected time of an incoming pulse. If an incoming pulse is detected inside this window this process is repeated. If the incoming data is a logical 1 (no pulse), the Endec returns to the initial state and waits for the next falling edge. As each falling edge is detected, the Endec clock counter is reset, resynchronizing the Endec to the incoming signal. This procedure allows the Endec to tolerate jitter and baud rate errors in the incoming data stream. Resynchronizing the Endec does not alter the operation of the UART, which ultimately receives the data. The UART is only synchronized to the incoming data stream when a Start bit is received.

Infrared Endec Control Register Definitions

All Infrared Endec configuration and status information is set by the UART control registers as defined in UART Control Register Definitions on page 100.

Caution: *To prevent spurious signals during IrDA data transmission, set the IREN bit in the UART Control 1 register to 1 to enable the Infrared Endec before enabling the GPIO Port alternate function for the corresponding pin.*

When reading data from the slave, the I2C pauses after the data Acknowledge cycle until the receive interrupt is serviced and the RDRF bit of the status register is cleared by reading the I2C Data Register. Once the I2C Data Register has been read, the I2C reads the next data byte.

Address Only Transaction with a 7-bit Address

In the situation where software determines if a slave with a 7-bit address is responding without sending or receiving data, a transaction can be done which only consists of an address phase. Figure 26 on page 131 displays this "address only" transaction to determine if a slave with a 7-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 is repeated until the slave does Acknowledge.

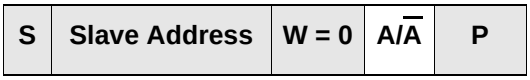


Figure 26. 7-Bit Address Only Transaction Format

Follow the steps below for an address only transaction to a 7-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 bit by writing a 7-bit Slave address plus write bit (=0) to the I2C Data Register. As an alternative this could be a read operation instead of a write operation.
5. Software sets the START and STOP bits of the I2C Control Register and clears the TXI bit.
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. Software polls the STOP bit of the I2C Control Register. Hardware deasserts the STOP bit when the address only transaction is completed.
9. Software checks the ACK bit of the I2C Status Register. If the slave acknowledged, the ACK bit is equal to 1. If the slave does not acknowledge, the ACK bit is equal to 0. The NCKI interrupt does not occur in the not acknowledge case because the STOP bit was set.

5. Re-write the page written in step 2 to the Page Select Register.
6. Write Flash Memory using LDC or LDCI instructions to program the Flash.
7. Repeat step 6 to program additional memory locations on the same page.
8. Write 00H to the Flash Control Register to lock the Flash Controller.

Page Erase

Flash memory can be erased one page (512 bytes) at a time. Page Erasing the Flash memory sets all bytes in that page to the value FFH. The Page Select Register identifies the page to be erased. While the Flash Controller executes the Page Erase operation, the eZ8 CPU idles but the system clock and on-chip peripherals continue to operate. The eZ8 CPU resumes operation after the Page Erase operation completes. Interrupts that occur when the Page Erase operation is in progress are serviced once the Page Erase operation is complete. When the Page Erase operation is complete, the Flash Controller returns to its locked state. Only pages located in unprotected sectors can be erased.

Follow the steps below to perform a Page Erase operation:

1. Write 00H to the Flash Control Register to reset the Flash Controller.
2. Write the page to be erased 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.
5. Re-write the page written in step 2 to the Page Select Register.
6. Write the Page Erase command 95H to the Flash Control Register.

Mass Erase

The Flash memory cannot be Mass Erased by user code.

Flash Controller Bypass

The Flash Controller can be bypassed and the control signals for the Flash memory brought out to the GPIO pins. Bypassing the Flash Controller allows faster Programming algorithms by controlling the Flash programming signals directly.

Flash Controller Bypass is recommended for gang programming applications and large volume customers who do not require in-circuit programming of the Flash memory.

For more information on bypassing the Flash Controller, refer to *Third-Party Flash Programming Support for Z8 Encore! XP*, available for download at www.zilog.com.

General Purpose I/O Port Input Data Sample Timing

Figure 48 displays timing of the GPIO Port input sampling. Table 105 lists the GPIO port input timing.

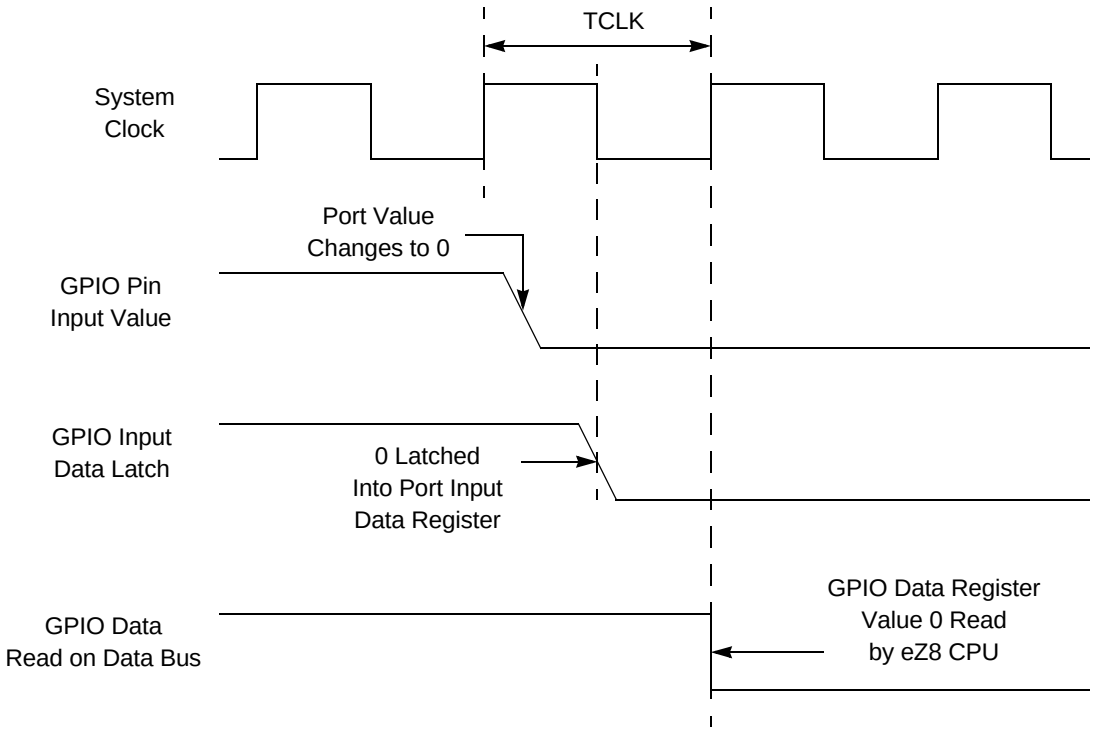


Figure 48. Port Input Sample Timing

Table 105. GPIO Port Input Timing

		Delay (ns)	
Parameter	**Abbreviation**	**Minimum**	**Maximum**
T_{S_PORT}	Port Input Transition to XIN Fall Setup Time (Not pictured)	5	–
T_{H_PORT}	XIN Fall to Port Input Transition Hold Time (Not pictured)	5	–
T_{SMR}	GPIO Port Pin Pulse Width to Insure Stop Mode Recovery (for GPIO Port Pins enabled as SMR sources)	1µs	

Figure 55 and Table 112 provide timing information for UART pins for the case where the Clear To Send input signal ($\overline{CTS}$) is not 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 $\overline{DE}$. $\overline{DE}$ asserts after the UART Transmit Data Register has been written. $\overline{DE}$ remains asserted for multiple characters as long as the Transmit Data Register is written with the next character before the current character has completed.

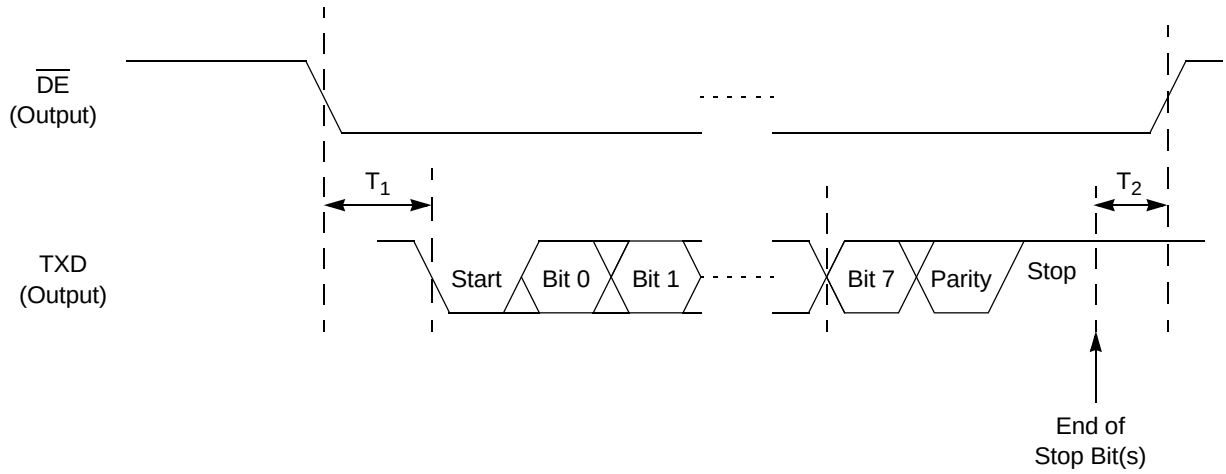


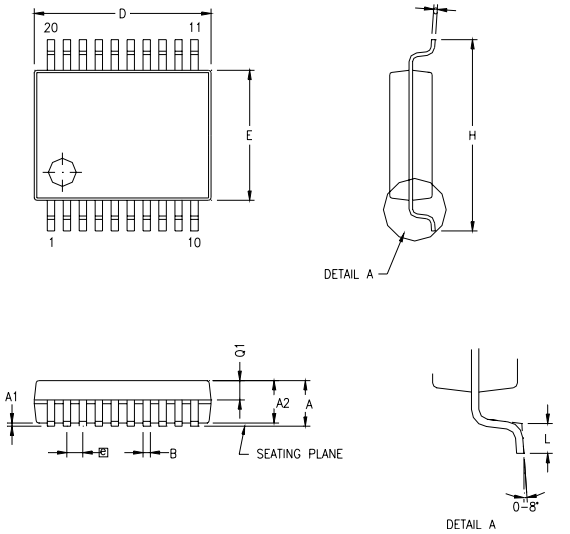
Figure 55. UART Timing without $\overline{CTS}$

Table 112. UART Timing without $\overline{CTS}$

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

Packaging

Figure 60 displays the 20-pin SSOP package available for Z8 Encore! XP® F0822 Series devices.

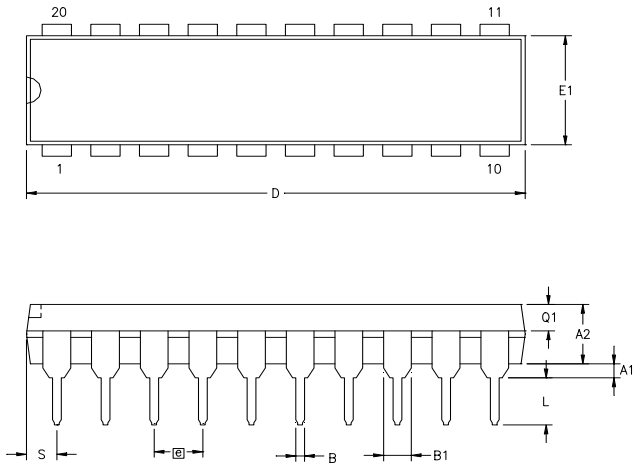


SYMBOL	MILLIMETER			INCH		
	MIN	NOM	MAX	MIN	NOM	MAX
A	1.73	1.85	1.98	0.068	0.073	0.078
A1	0.05	0.13	0.21	0.002	0.005	0.008
A2	1.68	1.73	1.83	0.066	0.068	0.072
B	0.25	0.30	0.38	0.010	0.012	0.015
C	0.13	0.15	0.22	0.005	0.006	0.009
D	7.07	7.20	7.33	0.278	0.283	0.289
E	5.20	5.30	5.38	0.205	0.209	0.212
e	0.65 BSC			0.0256 BSC		
H	7.65	7.80	7.90	0.301	0.307	0.311
L	0.56	0.75	0.94	0.022	0.030	0.037
Q1	0.74	0.78	0.82	0.029	0.031	0.032

CONTROLLING DIMENSIONS : MM
LEADS ARE COPLANAR WITHIN .004 INCH.

Figure 60. 20-Pin Small Shrink Outline Package (SSOP)

Figure 61 displays the 20-pin PDIP package available for Z8 Encore! XP F0822 Series devices.



SYMBOL	MILLIMETER		INCH	
	MIN	MAX	MIN	MAX
A1	0.38	0.81	.015	.032
A2	3.25	3.68	.128	.145
B	0.41	0.51	.016	.020
B1	1.47	1.57	.058	.062
C	0.20	0.30	.008	.012
D	25.65	26.16	1.010	1.030
E	7.49	8.26	.295	.325
E1	6.10	6.65	.240	.262
e	2.54 BSC		.100 BSC	
eA	7.87	9.14	.310	.360
L	3.18	3.43	.125	.135
Q1	1.42	1.65	.056	.065
S	1.52	1.65	.060	.065

CONTROLLING DIMENSIONS : INCH

Figure 61. 20-Pin Plastic Dual-Inline Package (PDIP)

X

Z