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, LINbus, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, LED, LVD, POR, PWM, Temp Sensor, WDT
Number of I/O	37
Program Memory Size	8KB (8K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	2K x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 3.6V
Data Converters	A/D 8x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 105°C (TA)
Mounting Type	Surface Mount
Package / Case	44-VFQFN Exposed Pad
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/zilog/z8f0880qn020eg

- Optional 16-bit Multi-Channel Timer which supports four Capture/Compare/PWM modules (44-pin packages only)
- Watchdog Timer (WDT) with dedicated internal RC oscillator
- 17 to 37 General-Purpose Input/Output (GPIO) pins depending upon package
- Up to 8 direct LED drives with programmable drive current capability
- Up to 31 interrupt sources with up to 24 interrupt vectors
- On-Chip Debugger (OCD)
- Power-On Reset (POR) and Voltage Brown-Out (VBO) protection
- Built-in Low-Voltage Detection (LVD) with programmable voltage threshold
- 32kHz secondary oscillator for Timers
- Internal Precision Oscillator (IPO) with output frequency in the range of 43.2kHz to 11.0592MHz
- Crystal oscillator with three power settings and external RC network option
- Wide operation voltage range: 1.8V–3.6V
- 20-, 28-, 40- and 44-pin packages
- 0°C to +70°C (standard) and –40°C to +105°C (extended) operating temperature ranges

1.2. Part Selection Guide

Table 1 displays basic features and package styles available for each of the F1680 Series MCUs.

Table 1. Z8 Encore! XP F1680 Series Part Selection Guide

Part Number	Flash (KB)	RAM (B)	Program RAM (B)	NVDS (B)	I/O	ADC Inputs	SPI	I^2C	UARTs	Packages
Z8F2480	24	2048	1024	—	17–37	7–8	0–1	1	1–2	20-, 28-, 40- and 44-pin
Z8F1680	16	2048	1024	256	17–37	7–8	0–1	1	1–2	20-, 28-, 40- and 44-pin
Z8F0880	8	1024	1024	128	17–37	7–8	0–1	1	1–2	20-, 28-, 40- and 44-pin

2.4. Pin Characteristics

Table 5 provides detailed information about the characteristics of each pin available on the F1680 Series MCU 20-, 28-, 40- and 44-pin devices. Data provided in Table 5 is sorted alphabetically by the pin symbol mnemonic.

Table 5. Pin Characteristics (20-, 28-, 40- and 44-pin Devices)

Symbol Mnemonic	Direction	Reset Direction	Active Low or Active High	Tristate Output	Internal Pull-up or Pull-down	Schmitt Trigger Input	Open Drain Output	5V Tolerance
AV_{DD}	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
AV_{SS}	N/A	N/A	N/A	N/A	N/A	N/A	N/A	NA
DBG	I/O	I	N/A	Yes	Yes	Yes	Yes	No
PA[7:0]	I/O	I	N/A	Yes	Programmable pull-up	Yes	Yes, programmable	Yes, 5V tolerant inputs unless pull-ups are enabled
PB[5:0]	I/O	I	N/A	Yes	Programmable pull-up	Yes	Yes, programmable	Yes, 5V tolerant inputs unless pull-ups are enabled
PC[7:0]	I/O	I	N/A	Yes	Programmable pull-up	Yes	Yes, programmable	Yes, 5V tolerant inputs unless pull-ups are enabled
PD[7:1]	I/O	I	N/A	Yes	Programmable pull-up	Yes	Yes, programmable	Yes, 5V tolerant inputs unless pull-ups are enabled

Table 7. F1680 Series MCU Flash Memory Information Area Map

Program Memory Address (Hex)	Function
FE00–FE3F	Zilog option bits
FE40–FE53	Part Number: 20-character ASCII alphanumeric code Left-justified and filled with FH
FE54–FE5F	Reserved
FE60–FE7F	Zilog calibration data (only use the first two bytes FE60 and FE61)
FE80–FFFF	Reserved

Chapter 4. Register Map

Table 8 provides an address map to the register file contained in all Z8 Encore! XP F1680 Series devices. Not all devices and package styles in this product series support the ADC, nor all of the GPIO ports. Therefore, consider the registers for unimplemented peripherals to be reserved.

Table 8. Register File Address Map

Address (Hex)	Register Description	Mnemonic	Reset (Hex)[1]	Page #
General Purpose RAM				
Z8F2480 Device				
000–7FF	General-Purpose Register File RAM	—	XX	
800–EFF	Reserved[2]	—	XX	
Z8F1680 Device				
000–7FF	General-Purpose Register File RAM	—	XX	
800–EFF	Reserved[2]	—	XX	
Z8F0880 Device				
000–3FF	General-Purpose Register File RAM	—	XX	
400–EFF	Reserved[2]	—	XX	
Special Purpose Registers				
Timer 0				
F00	Timer 0 High Byte	T0H	00	109
F01	Timer 0 Low Byte	T0L	01	109
F02	Timer 0 Reload High Byte	T0RH	FF	110
F03	Timer 0 Reload Low Byte	T0RL	FF	110
F04	Timer 0 PWM0 High Byte	T0PWM0H	00	110
F05	Timer 0 PWM0 Low Byte	T0PWM0L	00	111
F06	Timer 0 Control 0	T0CTL0	00	112
F07	Timer 0 Control 1	T0CTL1	00	113

Notes:
1. XX=Undefined.
2. The Reserved space can be configured as General-Purpose Register File RAM depending on the user option bits (see the User Option Bits chapter on page 277) and the on-chip PRAM size (see the Ordering Information chapter on page 372). If the PRAM is programmed as General-Purpose Register File RAM on Reserved space, the starting address always begins immediately after the end of General-Purpose Register File RAM.

Table 8. Register File Address Map (Continued)

Address (Hex)	Register Description	Mnemonic	Reset (Hex)[1]	Page #
Multi-Channel Timer				
FA0	MCT High Byte	MCTH	00	130
FA1	MCT Low Byte	MCTL	00	130
FA2	MCT Reload High Byte	MCTRH	FF	131
FA3	MCT Reload Low Byte	MCTRL	FF	131
FA4	MCT Subaddress	MCTSA	XX	132
FA5	MCT Subregister 0	MCTSR0	XX	132
FA6	MCT Subregister 1	MCTSR1	XX	132
FA7	MCT Subregister 2	MCTSR2	XX	132
FA8–FBF	Reserved	—	XX	
Interrupt Controller				
FC0	Interrupt Request 0	IRQ0	00	73
FC1	IRQ0 Enable High Bit	IRQ0ENH	00	76
FC2	IRQ0 Enable Low Bit	IRQ0ENL	00	77
FC3	Interrupt Request 1	IRQ1	00	74
FC4	IRQ1 Enable High Bit	IRQ1ENH	00	78
FC5	IRQ1 Enable Low Bit	IRQ1ENL	00	79
FC6	Interrupt Request 2	IRQ2	00	75
FC7	IRQ2 Enable High Bit	IRQ2ENH	00	80
FC8	IRQ2 Enable Low Bit	IRQ2ENL	00	81
FC9–FCC	Reserved	—	XX	
FCD	Interrupt Edge Select	IRQES	00	82
FCE	Shared Interrupt Select	IRQSS	00	82
FCF	Interrupt Control	IRQCTL	00	83
GPIO Port A				
FD0	Port A Address	PAADDR	00	58
FD1	Port A Control	PACTL	00	60

Notes:
1. XX=Undefined.
2. The Reserved space can be configured as General-Purpose Register File RAM depending on the user option bits (see the User Option Bits chapter on page 277) and the on-chip PRAM size (see the Ordering Information chapter on page 372). If the PRAM is programmed as General-Purpose Register File RAM on Reserved space, the starting address always begins immediately after the end of General-Purpose Register File RAM.

7.11.4. Port A–E Alternate Function Subregisters

The Port A–E Alternate Function Subregister, shown in Table 24, is accessed through the Port A–E Control Register by writing 02H to the Port A–E Address Register. The Port A–E Alternate Function subregisters enable the alternate function selection on the pins; if disabled, the pins function as GPIO. If enabled, select one of the four alternate functions using Alternate Function Set Subregisters 1 and 2 as described in the Port A–E Alternate Function Set 1 Subregisters section on page 64 and the Port A–E Alternate Function Set 2 Subregisters section on page 64. To determine the alternate function associated with each port pin, see the GPIO Alternate Functions section on page 47.

Caution: Do not enable alternate functions for GPIO port pins for which there is no associated alternate function. Failure to follow this guideline results in unpredictable operation.

Table 24. Port A–E Alternate Function Subregisters (PxAF)

Bits	7	6	5	4	3	2	1	0
Field	AF7	AF6	AF5	AF4	AF3	AF2	AF1	AF0
Reset	00H (Ports A–C); 01H (Port D); 00H (Port E);							
R/W	R/W							
Address	If 02H in Port A–D Address Register, accessible through the Port A–E Control Register.							

Bit	Description
[7:0] AF	**Port Alternate Function enabled** 0 = The port pin is in NORMAL Mode and the DDx bit in the Port A–E Data Direction subregister determines the direction of the pin. 1 = The alternate function selected through Alternate Function set subregisters are enabled. Port-pin operation is controlled by the alternate function.

If the TPOL bit in the Timer Control 1 Register is set to 1, the Timer Output signal begins as High (1) and then transitions to Low (0) when the timer value matches the PWM value. The Timer Output signal returns to High (1) after the timer reaches the reload value and is reset to 0001H.

If the TPOL bit in the Timer Control 1 Register is set to 0, the Timer Output signal begins as Low (0) and then transitions to High (1) when the timer value matches the PWM value. The Timer Output signal returns to Low (0) after the timer reaches the reload value and is reset to 0001H.

Observe the following steps to configure a timer for PWM SINGLE OUTPUT Mode and initiate PWM operation:

1. Write to the Timer Control 1 Register to:
   - Disable the timer
   - Configure the timer for PWM mode
   - Set the prescale value
   - Set the initial logic level (High or Low) and PWM High/Low transition for the Timer Output Alternate Function
2. Write to the Timer Control 2 Register to choose the timer clock source.
3. Write to the Timer Control 0 Register to set the timer interrupt configuration field TICONFIG.
4. Write to the Timer High and Low Byte registers to set the starting count value (typically 0001H). This value only affects the first pass in PWM mode. After the first timer reset in PWM mode, counting always begins at the reset value of 0001H.
5. Write to the Timer PWM0 High and Low Byte registers to set the PWM value.
6. 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.
7. If appropriate, enable the timer interrupt and set the timer interrupt priority by writing to the relevant interrupt registers.
8. Configure the associated GPIO port pin for the Timer Output alternate function.
9. Write to the Timer Control 1 Register to enable the timer and initiate counting.

The PWM period is calculated using the following equation:

If an initial starting value other than 0001H is loaded into the Timer High and Low Byte registers, the ONE-SHOT Mode equation must be used to determine the first PWM time-out period.

Observe the following steps to configure a timer for CAPTURE RESTART Mode and initiate the count:

1. Write to the Timer Control 1 Register to:
   - Disable the timer
   - Configure the timer for CAPTURE RESTART Mode. Setting the mode also involves writing to TMODE[3] bit in the TxCTL0 Register
   - Set the prescale value
   - Set the Capture edge (rising or falling) for the Timer Input
2. Write to the Timer Control 2 Register to choose the timer clock source.
3. Write to the Timer Control 0 Register to set the timer interrupt configuration field TICONFIG.
4. Write to the Timer High and Low Byte registers to set the starting count value (typically `0001H`).
5. Write to the Timer Reload High and Low Byte registers to set the reload value.
6. Clear the Timer PWM High and Low Byte registers to `0000H`. This allows user software to determine if interrupts are generated by either a Capture Event or a Reload. If the PWM High and Low Byte registers still contain `0000H` after the interrupt, then the interrupt is generated by a Reload.
7. If required, enable the timer interrupt and set the timer interrupt priority by writing to the relevant interrupt registers. By default, the timer interrupt will be generated for both input capture and reload events. If required, configure the timer interrupt to be generated only at the Input Capture event or the reload event by setting TICONFIG field of the Timer Control 0 Register.
8. Configure the associated GPIO port pin for the Timer Input alternate function.
9. Write to the Timer Control 1 Register to enable the timer and initiate counting.

In CAPTURE Mode, the elapsed time from Timer start to Capture event can be calculated using the following equation:

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

Table 60. Timer 0–2 PWM0 Low Byte Register (TxPWM0L)

Bit	7	6	5	4	3	2	1	0
Field	PWM0L							
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	F05H, F0DH, F15H							

Bit	Description
[7:0] PWM0H, PWM0L	**Pulse Width Modulator 0 High and Low Bytes** These two bytes, {PWM0H[7:0], PWM0L[7:0]}, form a 16-bit value that is compared to the current 16-bit timer count. When a match occurs, the PWM output changes state. The PWM output value is set by the TPOL bit in the Timer Control 1 Register (TxCTL1). The TxPWM0H and TxPWM0L registers also store the 16-bit captured timer value when operating in CAPTURE, CAPTURE/COMPARE and DEMODULATION Modes.

9.3.4. Timer 0–2 PWM1 High and Low Byte Registers

The Timer 0–2 PWM1 High and Low Byte (TxPWM1H and TxPWM1L) registers, shown in Tables 61 and 62, store Capture values for DEMODULATION Mode.

Table 61. Timer 0-2 PWM1 High Byte Register (TxPWM1H)

Bit	7	6	5	4	3	2	1	0
Field	PWM1H							
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	F20H, F24H, F28H							

Table 62. Timer 0–2 PWM1 Low Byte Register (TxPWM1L)

Bit	7	6	5	4	3	2	1	0
Field	PWM1L							
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	F21H, F25H, F29H							

Bit	Description
[7:0] PWM1H, PWM1L	**Pulse Width Modulator 1 High and Low Bytes** These two bytes, {PWM1H[7:0], PWM1L[7:0]}, store the 16-bit captured timer value for DEMODULATION Mode.

rate. To avoid an autobaud overrun error, the system clock must not be greater than 2^{19} times the baud rate (16 bit counter following 3-bit prescaler when counting the 8 bit times of the Autobaud sequence).

Following the Synch character, the LIN-UART hardware transits to the Active state, in which the identifier character is received and the characters of the response section of the message are sent or received. The slave remains in this Active state until a break is received or software forces a state change. After it is in an Active state (i.e., autobaud has completed), a break of 10 or more bit times is recognized and causes a transition to the Autobaud state.

If the identifier character indicates that this slave device is not participating in the message, software sets the LinState[1:0] = 01b (Wait for Break state) to ignore the rest of the message. No further receive interrupts will occur until the next break.

12.1.11. LIN-UART Interrupts

The LIN-UART features separate interrupts for the transmitter and receiver. In addition, when the LIN-UART primary functionality is disabled, the Baud Rate Generator can also function as a basic timer with interrupt capability.

12.1.11.1. Transmitter Interrupts

The transmitter generates a single interrupt when the Transmit Data Register Empty bit (TDRE) is set to 1. This indicates that the transmitter is ready to accept new data for transmission. The TDRE interrupt occurs when the transmitter is initially enabled and after the Transmit Shift Register has shifted out the first bit of a character. At this point, the Transmit Data Register can be written with the next character to send. This 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 LIN-UART Transmit Data Register clears the TDRE bit to 0.

12.1.11.2. Receiver Interrupts

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

- A data byte has been received and is available in the LIN-UART Receive Data Register. This interrupt can be disabled independent of the other receiver interrupt sources via the RDAIRQ bit (this feature is useful in devices which support DMA). The received data interrupt occurs after the receive character has been placed in the Receive Data Register. Software must respond to this received data available condition before the next character is completely received to avoid an overrun error.

12.1.12. LIN-UART Baud Rate Generator

The LIN-UART Baud Rate Generator creates a lower frequency baud rate clock for data transmission. The input to the Baud Rate Generator is the system clock. The LIN-UART Baud Rate High and Low Byte registers combine to create a 16-bit baud rate divisor value (BRG[15:0]) that sets the data-transmission rate (baud rate) of the LIN-UART. The LIN-UART data rate for normal UART operation is calculated using the following equation:

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

The LIN-UART data rate for LIN mode UART operation is calculated using the following equation:

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

When the LIN-UART is disabled, the BRG functions as a basic 16-bit timer with interrupt on time-out. To configure the BRG as a timer with interrupt on time-out, follow the procedure below:

1. Disable the LIN-UART receiver by clearing the REN bit in the LIN-UART Control 0 Register to 0 (i.e., the TEN bit can be asserted; transmit activity can occur).
2. Load the appropriate 16-bit count value into the LIN-UART Baud Rate High and Low Byte registers.
3. Enable the BRG timer function and the associated interrupt by setting the BRGCTL bit in the LIN-UART Control 1 Register to 1.

12.2. Noise Filter

A noise filter circuit is included which filters noise on a digital input signal (such as UART Receive Data) before the data is sampled by the block. This noise filter is likely to be a requirement for protocols with a noisy environment.

The noise filter contains the following features:

- Synchronizes the receive input data to the System Clock
- Noise Filter Enable (NFEN) input selects whether the noise filter is bypassed (NFEN = 0) or included (NFEN=1) in the receive data path

12.3.5. LIN-UART Control 0 Register

The LIN-UART Control 0 Register, shown in Table 89, configures the basic properties of LIN-UART's transmit and receive operations. A more detailed discussion of each bit follows the table.

Table 89. LIN-UART Control 0 Register (U0CTL0 = F42H)

Bit	7	6	5	4	3	2	1	0
Field	TEN	REN	CTSE	PEN	PSEL	SBRK	STOP	LBEN
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	F42H, F4AH							

Note: R/W = Read/Write.

Bit	Description
[7] TEN	**Transmit Enable** This bit enables or disables the transmitter. The enable is also controlled by the $\overline{CTS}$ signal and the CTSE bit. If the $\overline{CTS}$ signal is Low and the CTSE bit is 1, the transmitter is enabled. 0 = Transmitter disabled. 1 = Transmitter enabled.
[6] REN	**Receive Enable** This bit enables or disables the receiver. 0 = Receiver disabled. 1 = Receiver enabled.
[5] CTSE	**Clear To Send Enable** 0 = The $\overline{CTS}$ signal has no effect on the transmitter. 1 = The LIN-UART recognizes the $\overline{CTS}$ signal as an enable control for the transmitter.
[4] PEN	**Parity Enable** This bit enables or disables parity. Even or odd is determined by the PSEL bit. 0 = Parity is disabled. This bit is overridden by the MPEN bit. 1 = The transmitter sends data with an additional parity bit and the receiver receives an additional parity bit.
[3] PSEL	**Parity Select** 0 = Even parity is sent as an additional parity bit for the transmitter/receiver. 1 = Odd parity is sent as an additional parity bit for the transmitter/receiver.

The window remains open until the count again reaches 8 (in other words, 24 baud clock periods since the previous pulse is detected), giving the endec a sampling window of minus 4 baud rate clocks to plus 8 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, allowing the endec to tolerate jitter and baud rate errors in the incoming datastream. 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.

13.3. Infrared Encoder/Decoder Control Register Definitions

All infrared endec configuration and status information is set by the UART control registers as defined beginning on page 163.

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 encoder/decoder before enabling the GPIO Port alternate function for the corresponding pin of UART. See Tables 17 through 19 on pages 49–54 for details.

the V_{REF} pin. When RBUF is disabled, the ADC must have the reference voltage supplied externally through the V_{REF} pin. RBUF is controlled by the REFEN bit in the ADC Control Register.

14.2.4. Internal Voltage Reference Generator

The Internal Voltage Reference Generator provides the voltage, VR2, for the RBUF. VR2 is 1.6V.

14.2.5. Calibration and Compensation

You can calibrate and store the values into Flash, or the user code can perform a manual offset calibration. There is no provision for manual gain calibration.

14.3. ADC Control Register Definitions

The registers that control analog-to-digital conversion functions are defined in this section.

14.3.1. ADC Control Register 0

The ADC Control Register 0, shown in Table 101, initiates the A/D conversion and provides ADC status information.

Table 101. ADC Control Register 0 (ADCCTL0)

Bits	7	6	5	4	3	2	1	0
Field	START	INTREF_SEL	REFEN	ADCEN	ANAIN[3:0]			
Reset	0	0	0	0	0	0	0	0
R/W	R/W1	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	F70h							

Bit Position	Value (H)	Description
[7] START	0	**ADC Start/Busy** Writing a 0 has no effect. Reading a 0 indicates the ADC is available to begin a conversion.
	1	Writing a 1 starts a conversion. Reading a 1 indicates that a conversion is currently in progress.
[6] INTREF_SEL	0	Select 1.6 V as internal reference.
	1	Select AVDD as internal reference.

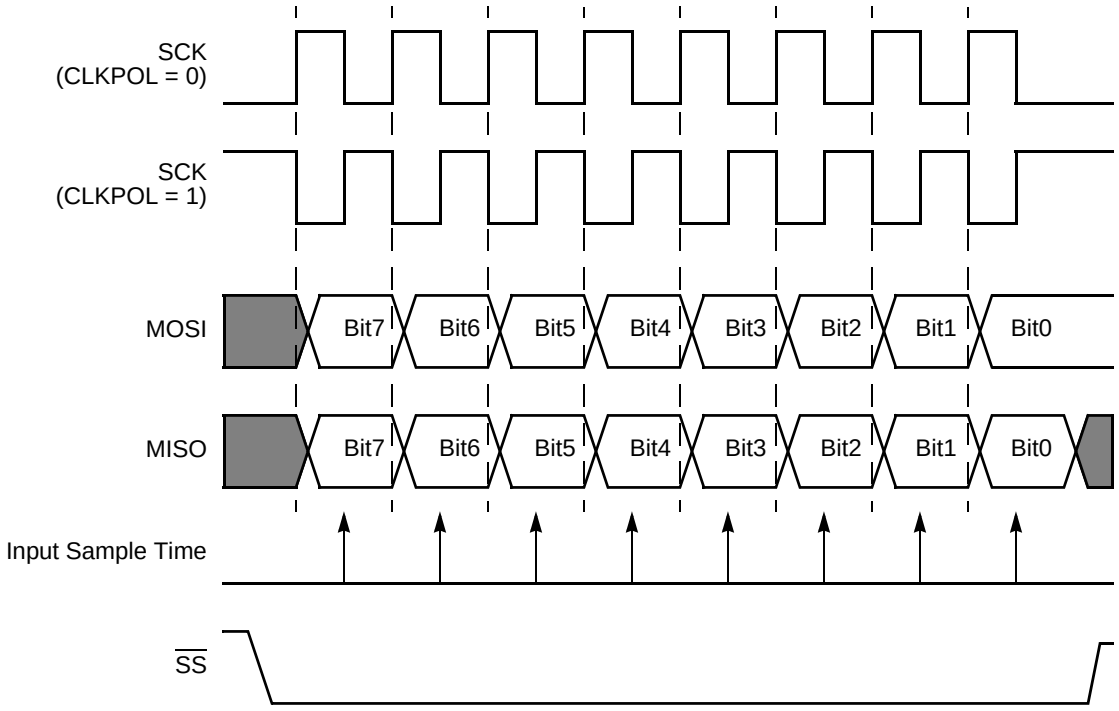

Figure 35. ESPI Timing when PHASE = 1

16.3.3. Slave Select Modes of Operation

This section describes the different modes of data transfer supported by the ESPI block. The mode is selected by the Slave Select Mode (SSMD) field of the Mode Register.

16.3.3.1. SPI Mode

This mode is selected by setting the SSMD field of the Mode Register to 00. In this mode software controls the assertion of the $\overline{SS}$ signal directly via the SSV bit of the SPI Transmit Data Command register. Software can be used to control an SPI mode transaction. Prior to or simultaneously with writing the first transmit data byte; software sets the SSV bit. Software sets the SSV bit either by performing a byte write to the Transmit Data Command register prior to writing the first transmit character to the Data Register or by performing a word write to the Data Register address which loads the first transmit character and simultaneously sets the SSV bit. $\overline{SS}$ will remain asserted when one or more characters are transferred. There are two mechanisms for deasserting $\overline{SS}$ at the end of the transaction. One method used by software is to set the TEOF bit of the Transmit Data Command register, when the last TDRE interrupt is being serviced (set TEOF before or simultaneously with writing the last data byte). After the last bit of the last character is

GCE bit = 1 in the I2CMODE Register. The software checks the RD bit in the I2CISTAT Register to determine if the transaction is a Read or Write transaction. The General Call Address and STARTBYTE address are also distinguished by the RD bit. The General Call Address (GCA) bit of the I2CISTAT Register indicates whether the address match occurred on the unique slave address or the General Call/STARTBYTE address. The SAM bit clears automatically when the I2CISTAT Register is read.

If configured via the MODE[1:0] field of the I^2C Mode Register for 7-bit slave addressing, the most significant 7 bits of the first byte of the transaction are compared against the SLA[6:0] bits of the Slave Address Register. If configured for 10-bit slave addressing, the first byte of the transaction is compared against {11110,SLA[9:8], R/$\overline{W}$} and the second byte is compared against SLA[7:0].

17.2.2.4. Arbitration Lost Interrupts

Arbitration Lost interrupts (ARBLST bit = 1 in I2CISTAT) occur when the I^2C controller is in MASTER Mode and loses arbitration (outputs 1 on SDA and receives 0 on SDA). The I^2C controller switches to SLAVE Mode when this instance occurs. This bit clears automatically when the I2CISTAT Register is read.

17.2.2.5. Stop/Restart Interrupts

A Stop/Restart event interrupt (SPRS bit = 1 in I2CISTAT) occurs when the I^2C controller is operating in SLAVE Mode and a stop or restart condition is received, indicating the end of the transaction. The RSTR bit in the I^2C State Register indicates whether the bit is set due to a stop or restart condition. When a restart occurs, a new transaction by the same master is expected to follow. This bit is cleared automatically when the I2CISTAT Register is read. The Stop/Restart interrupt occurs only on a selected (address match) slave.

17.2.2.6. Not Acknowledge Interrupts

Not Acknowledge interrupts (NCKI bit = 1 in I2CISTAT) occur in MASTER Mode when Not Acknowledge is received or sent by the I^2C controller and the start or stop bit is not set in the I^2C Control Register. In MASTER Mode, the Not Acknowledge interrupt clears by setting the start or stop bit. When this interrupt occurs in MASTER Mode, the I^2C controller waits until it is cleared before performing any action. In SLAVE Mode, the Not Acknowledge interrupt occurs when a Not Acknowledge is received in response to data sent. The NCKI bit clears in SLAVE Mode when software reads the I2CISTAT Register.

17.2.2.7. General Purpose Timer Interrupt from Baud Rate Generator

If the I^2C controller is disabled (IEN bit in the I2CCTL Register = 0) and the BIRQ bit in the I2CCTL Register = 1, an interrupt is generated when the baud rate generator (BRG) counts down to 1. The baud rate generator reloads and continues counting, providing a periodic interrupt. None of the bits in the I2CISTAT Register are set, allowing the BRG in the I^2C Controller to be used as a general-purpose timer when the I^2C Controller is disabled.

17.2.3. Start and Stop Conditions

The Master generates the start and stop conditions to start or end a transaction. To start a transaction, the I^2C controller generates a start condition by pulling the SDA signal Low while SCL is High. To complete a transaction, the I^2C controller generates a stop condition by creating a Low-to-High transition of the SDA signal while the SCL signal is High. These start and stop events occur when the start and stop bits in the I^2C Control Register are written by software to begin or end a transaction. Any byte transfer currently under way including the Acknowledge phase finishes before the start or stop condition occurs.

17.2.4. Software Control of I^2C Transactions

The I^2C controller is configured via the I^2C Control and I^2C Mode registers. The MODE[1:0] field of the I^2C Mode Register allows the configuration of the I^2C controller for MASTER/SLAVE or SLAVE ONLY mode and configures the slave for 7-bit or 10-bit addressing recognition.

MASTER/SLAVE Mode can be used for:

- MASTER ONLY operation in a Single Master/One or More Slave I^2C system
- MASTER/SLAVE in a Multimaster/multislave I^2C system
- SLAVE ONLY operation in an I^2C system

In SLAVE ONLY mode, the start bit of the I^2C Control Register is ignored (software cannot initiate a master transaction by accident) and operation to SLAVE ONLY Mode is restricted thereby preventing accidental operation in MASTER Mode. The software controls I^2C transactions by enabling the I^2C controller interrupt in the interrupt controller or by polling the I^2C Status Register.

To use interrupts, the I^2C interrupt must be enabled in the interrupt controller and followed by executing an EI instruction. The TXI bit in the I^2C Control Register must be set to enable transmit interrupts. An I^2C interrupt service routine then checks the I^2C Status Register to determine the cause of the interrupt.

To control transactions by polling, the TDRE, RDRF, SAM, ARBLST, SPRS and NCKI interrupt bits in the I^2C Status Register should be polled. The TDRE bit asserts regardless of the state of the TXI bit.

17.2.5. Master Transactions

The following sections describe Master Read and Write transactions to both 7-bit and 10-bit slaves.

I2CISTAT Register is set to 1, thereby causing an interrupt. The RD bit is cleared to 0, indicating a Write to the slave. The I^2C controller acknowledges, indicating it is available to accept the data.

4. The software responds to the interrupt by reading the I2CISTAT Register, which clears the SAM bit. Because RD = 0, no immediate action is taken by the software until the first byte of data is received. If the software is only able to accept a single byte, it sets the NAK bit in the I2CCTL Register.
5. The Master detects the Acknowledge and sends the first byte of data.
6. The I^2C controller receives the first byte and responds with Acknowledge or Not Acknowledge, depending on the state of the NAK bit in the I2CCTL Register. The I^2C controller generates the receive data interrupt by setting the RDRF bit in the I2CISTAT Register.
7. The software responds by reading the I2CISTAT Register, finding the RDRF bit = 1 and then reading the I2CDATA Register, which clears the RDRF bit. If the software can accept only one more data byte, it sets the NAK bit in the I2CCTL Register.
8. The Master and Slave loops through Step 5 to Step 7 until the Master detects a Not Acknowledge instruction or runs out of data to send.
9. The Master sends the stop or restart signal on the bus. Either of these signals can cause the I^2C controller to assert the stop interrupt (the stop bit = 1 in the I2CISTAT Register). Because the slave received data from the master, the software takes no action in response to the STOP interrupt other than reading the I2CISTAT Register to clear the stop bit.

17.2.6.7. Slave Transmit Transaction With 7-bit Address

The data transfer format for a master reading data from a slave in 7-bit address mode is displayed in Figure 49. The procedure that follows describes the I^2C Master/Slave Controller operating as a slave in 7-bit addressing mode and transmitting data to the bus master.

S	Slave Address	R = 1	A	Data	A	Data	A	P/S

Figure 49. Data Transfer Format—Slave Transmit Transaction with 7-bit Address

1. The software configures the controller for operation as a slave in 7-bit addressing mode, as follows:
   a. Initialize the MODE field in the I^2C Mode Register for either SLAVE ONLY mode or MASTER/SLAVE Mode with 7-bit addressing.
   b. Optionally set the GCE bit.
   c. Initialize the SLA[6:0] bits in the I^2C Slave Address Register.

Table 186. eZ8 CPU Instruction Summary (Continued)

Assembly Mnemonic	Symbolic Operation	Address Mode dst	Address Mode src	Op Code(s) (Hex)	Flags C	Z	S	V	D	H	Fetch Cycles	Instr. Cycles
LDX dst, src	dst ← src	r	ER	84	–	–	–	–	–	–	3	2
		Ir	ER	85							3	3
		R	IRR	86							3	4
		IR	IRR	87							3	5
		r	X(rr)	88							3	4
		X(rr)	r	89							3	4
		ER	r	94							3	2
		ER	Ir	95							3	3
		IRR	R	96							3	4
		IRR	IR	97							3	5
		ER	ER	E8							4	2
		ER	IM	E9							4	2
LEA dst, X(src)	dst ← src + X	r	X(r)	98	–	–	–	–	–	–	3	3
		rr	X(rr)	99							3	5
MULT dst	dst[15:0] ← dst[15:8] * dst[7:0]	RR		F4	–	–	–	–	–	–	2	8
NOP	No operation			0F	–	–	–	–	–	–	1	2
OR dst, src	dst ← dst OR src	r	r	42	–	*	*	0	–	–	2	3
		r	Ir	43							2	4
		R	R	44							3	3
		R	IR	45							3	4
		R	IM	46							3	3
		IR	IM	47							3	4
ORX dst, src	dst ← dst OR src	ER	ER	48	–	*	*	0	–	–	4	3
		ER	IM	49							4	3
POP dst	dst ← @SP SP ← SP + 1	R		50	–	–	–	–	–	–	2	2
		IR		51							2	3

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