# Zilog - Z8F4803FT020SC Datasheet

**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 | 60 |
| Program Memory Size | 48KB (48K x 8) |
| Program Memory Type | FLASH |
| EEPROM Size | - |
| RAM Size | 4K x 8 |
| Voltage - Supply (Vcc/Vdd) | 3V ~ 3.6V |
| Data Converters | A/D 12x10b |
| Oscillator Type | Internal |
| Operating Temperature | 0°C ~ 70°C (TA) |
| Mounting Type | Surface Mount |
| Package / Case | 80-BQFP |
| Supplier Device Package | - |
| Purchase URL | https://www.e-xfl.com/product-detail/zilog/z8f4803ft020sc |

out, first set the TPOL bit in the Timer Control Register to the start value before beginning One-Shot mode. Then, after starting the timer, set TPOL to the opposite bit value.

The steps for configuring a timer for One-Shot mode and initiating the count are as follows:

1. Write to the Timer Control register to:
   - Disable the timer
   - Configure the timer for One-Shot mode.
   - Set the prescale value.
   - If using the Timer Output alternate function, set the initial output level (High or Low).
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 Reload 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 register to enable the timer and initiate counting.

In One-Shot mode, the system clock always provides the timer input. The timer period is given by the following equation:

$$\textbf{One-Shot Mode Time-Out Period (s)} = \frac{(\textbf{Reload Value} - \textbf{Start Value}) \times \textbf{Prescale}}{\textbf{System Clock Frequency (Hz)}}$$

### Continuous Mode

In Continuous mode, 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. Upon 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. Also, if the Timer Output alternate function is enabled, the Timer Output pin changes state (from Low to High or from High to Low) upon timer Reload.

The steps for configuring a timer for Continuous mode and initiating the count are as follows:

1. Write to the Timer Control register to:
   - Disable the timer
   - Configure the timer for Continuous mode.
   - Set the prescale value.

5. Configure the associated GPIO port pin for the Timer Input alternate function.
6. Write to the Timer Control register to enable the timer.
7. Counting begins on the first appropriate transition of the Timer Input signal. No interrupt is generated by this first edge.

In Capture/Compare mode, the elapsed time from timer start to Capture event can be calculated using the following equation:

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

### Reading the Timer Count Values

The current count value in the timers can be read while counting (enabled). This capability has no effect on timer operation. When the timer is enabled and the Timer High Byte register is read, the contents of the Timer Low Byte register are placed in a holding register. A subsequent read from the Timer Low Byte register returns the value in the holding register. This operation allows accurate reads of the full 16-bit timer count value while enabled. When the timers are not enabled, a read from the Timer Low Byte register returns the actual value in the counter.

### Timer Output Signal Operation

Timer Output is a GPIO Port pin alternate function. Generally, the Timer Output is toggled every time the counter is reloaded.

## Timer Control Register Definitions

Timers 0–2 are available in all packages. Timer 3 is available only in the 64-, 68- and 80-pin packages.

### Timer 0-3 High and Low Byte Registers

The Timer 0-3 High and Low Byte (TxH and TxL) registers (Tables 38 and 39) contain the current 16-bit timer count value. When the timer is enabled, a read from TxH causes the value in TxL to be stored in a temporary holding register. A read from TMRL always returns this temporary register when the timers are enabled. When the timer is disabled, reads from the TMRL reads the register directly.

Writing to the Timer High and Low Byte registers while the timer is enabled is not recommended. There are no temporary holding registers available for write operations, so simultaneous 16-bit writes are not possible. If either the Timer High or Low Byte registers are

## Timer 0-3 Control Registers

The Timer 0-3 Control (TxCTL) registers enable/disable the timers, set the prescaler value, and determine the timer operating mode.

**Table 44. Timer 0-3 Control Register (TxCTL)**

| BITS | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| FIELD | TEN | TPOL | PRES | | | TMODE | | |
| 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 |
| ADDR | F07H, F0FH, F17H, F1FH | | | | | | | |

TEN—Timer Enable
0 = Timer is disabled.
1 = Timer enabled to count.

TPOL—Timer Input/Output Polarity
Operation of this bit is a function of the current operating mode of the timer.

**One-Shot 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.

**Continuous 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.

**Counter 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.

**PWM 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.

information on approximate time-out delays for the minimum and maximum WDT reload values.

**Table 45. Watch-Dog Timer Approximate Time-Out Delays**

| WDT Reload Value | WDT Reload Value | Approximate Time-Out Delay (with 50kHz typical WDT oscillator frequency) | |
|---|---|---|---|
| (Hex) | (Decimal) | Typical | Description |
| 000004 | 4 | 80µs | Minimum time-out delay |
| FFFFFF | 16,777,215 | 335.5s | Maximum time-out delay |

## Watch-Dog Timer Refresh

When first enabled, the Watch-Dog Timer is loaded with the value in the Watch-Dog Timer Reload registers. The Watch-Dog Timer then counts down to `000000H` unless a WDT instruction is executed by the eZ8 CPU. Execution of the WDT instruction causes the downcounter to be reloaded with the WDT Reload value stored in the Watch-Dog Timer Reload registers. Counting resumes following the reload operation.\

When the Z8F640x family device is operating in Debug Mode (via the On-Chip Debugger), the Watch-Dog Timer is continuously refreshed to prevent spurious Watch-Dog Timer time-outs.

## Watch-Dog Timer Time-Out Response

The Watch-Dog Timer times out when the counter reaches `000000H`. A time-out of the Watch-Dog Timer generates either an interrupt or a Short Reset. The `WDT_RES` Option Bit determines the time-out response of the Watch-Dog Timer. Refer to the **Option Bits** chapter for information regarding programming of the `WDT_RES` Option Bit.

### WDT Interrupt in Normal Operation

If configured to generate an interrupt when a time-out occurs, the Watch-Dog Timer issues an interrupt request to the interrupt controller and sets the `WDT` status bit in the Watch-Dog Timer Control register. If interrupts are enabled, the eZ8 CPU responds to the interrupt request by fetching the Watch-Dog Timer interrupt vector and executing code from the vector address. After time-out and interrupt generation, the Watch-Dog Timer counter rolls over to its maximum value of FFFFFH and continues counting. The Watch-Dog Timer counter is not automatically returned to its Reload Value.

### WDT Interrupt in Stop Mode

If configured to generate an interrupt when a time-out occurs and the Z8F640x family device is in STOP mode, the Watch-Dog Timer automatically initiates a STOP Mode Recovery and generates an interrupt request. Both the `WDT` status bit and the `STOP` bit in the Watch-Dog Timer Control register are set to 1 following WDT time-out in STOP mode. Refer to the **Reset and Stop Mode Recovery** chapter for more information on STOP Mode Recovery.

If interrupts are enabled, following completion of the Stop Mode Recovery the eZ8 CPU responds to the interrupt request by fetching the Watch-Dog 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 Watch-Dog Timer forces the Z8F640x family device into the Short Reset state. The WDT status bit in the Watch-Dog Timer Control register is set to 1. Refer to the **Reset and Stop Mode Recovery** chapter for more information on Short Reset.

### WDT Reset in Stop Mode

If configured to generate a Reset when a time-out occurs and the Z8F640x family device is in STOP mode, the Watch-Dog Timer initiates a Stop Mode Recovery. Both the `WDT` status bit and the `STOP` bit in the Watch-Dog Timer Control register are set to 1 following WDT time-out in STOP mode. Refer to the **Reset and Stop Mode Recovery** chapter for more information.

## Watch-Dog Timer Reload Unlock Sequence

Writing the unlock sequence to the Watch-Dog Timer Control register (WDTCTL) unlocks the three Watch-Dog 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. The follow sequence is required to unlock the Watch-Dog Timer Reload Byte registers (WDTU, WDTH, and WDTL) for write access.

1. Write 55H to the Watch-Dog Timer Control register (WDTCTL)
2. Write AAH to the Watch-Dog Timer Control register (WDTCTL)
3. Write the Watch-Dog Timer Reload Upper Byte register (WDTU)
4. Write the Watch-Dog Timer Reload High Byte register (WDTH)
5. Write the Watch-Dog Timer Reload Low Byte register (WDTL)

All three Watch-Dog Timer Reload registers must be written in the order just listed. There must be no other register writes between each of these operations. If a register write occurs, the lock state machine resets and no further writes can occur, unless the sequence is restarted. The value in the Watch-Dog Timer Reload registers is loaded into the counter when the Watch-Dog Timer is first enabled and every time a WDT instruction is executed.

0 = No parity error has occurred.
1 = A parity error has occurred.

OE—Overrun Error
This bit indicates that an overrun error has occurred. An overrun occurs when new data is received and the UART Receive Data register has not been read. If the `RDA` bit is reset to 0, then reading the UART Receive Data register clears this bit.
0 = No overrun error occurred.
1 = An overrun error occurred.

FE—Framing Error
This bit indicates that a framing error (no Stop bit following data reception) was detected. Reading the UART Receive Data register clears this bit.
0 = No framing error occurred.
1 = A framing error occurred.

BRKD—Break Detect
This bit indicates that a break occurred. If the data bits, parity/multiprocessor bit, and Stop bit(s) are all zeros then this bit is set to 1. Reading the UART Receive Data register clears this bit.
0 = No break occurred.
1 = A break occurred.

TDRE—Transmitter Data Register Empty
This bit indicates that the UART Transmit Data register is empty and ready for additional data. Writing to the UART Transmit Data register resets this bit.
0 = Do not write to the UART Transmit Data register.
1 = The UART Transmit Data register is ready to receive an additional byte to be transmitted.

TXE—Transmitter Empty
This bit indicates that the transmit shift register is empty and character transmission is finished.
0 = Data is currently transmitting.
1 = Transmission is complete.

CTS—$\overline{\text{CTS}}$ signal
When this bit is read it returns the level of the $\overline{\text{CTS}}$ signal.

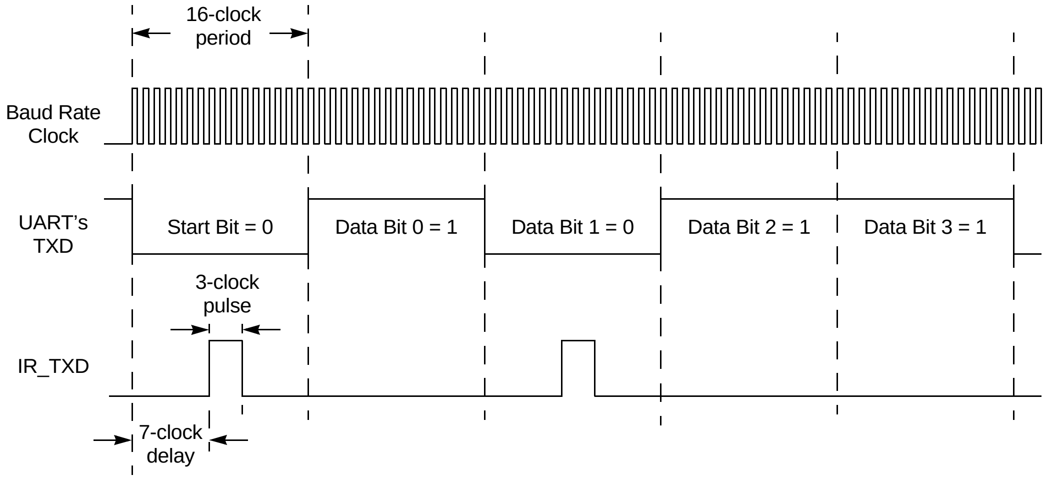

**Figure 72. Infrared Data Transmission**

## Receiving IrDA Data

Data received from the infrared transceiver via the IR_RXD signal through the RXD pin is decoded by the Infrared Endec and passed to the UART. The UART's baud rate clock is used by the Infrared Endec to generate the demodulated signal (RXD) that drives the UART. Each UART/Infrared data bit is 16-clocks wide. Figure 73 illustrates data reception. When the Infrared Endec is enabled, the UART's RXD signal is internal to the Z8F640x family device while the IR_RXD signal is received through the RXD pin.

# Serial Peripheral Interface

## Overview

The Serial Peripheral Interface™ (SPI) is a synchronous interface allowing several SPI-type devices to be interconnected. SPI-compatible devices include EEPROMs, Analog-to-Digital Converters, and ISDN devices. Features of the SPI include:

- Full-duplex, synchronous, character-oriented communication
- Four-wire interface
- Data transfers rates up to a maximum of one-fourth the system clock frequency
- Error detection
- Write and mode collision detection
- Dedicated Baud Rate Generator

## Architecture

The SPI may be configured as either a Master (in single or multi-master systems) or a Slave as illustrated in Figures 74 through 76.

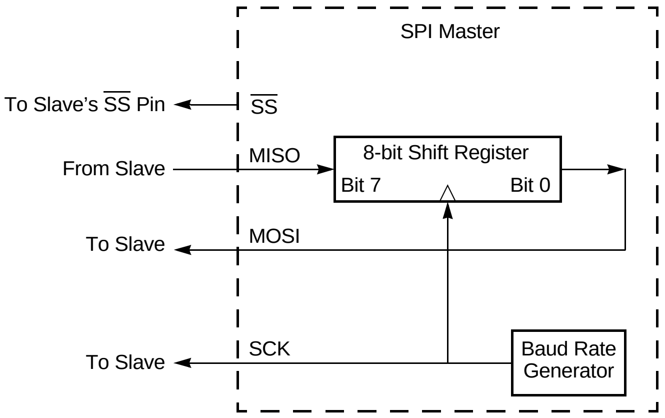

**Figure 74. SPI Configured as a Master in a Single Master, Single Slave System**

START—Send Start Condition
This bit sends the Start condition. Once asserted, it is cleared by the I$^2$C Controller after it sends the START condition or by deasserting the `IEN` bit. After this bit is set, the Start condition is sent if there is data in the I$^2$C Data or I$^2$C Shift register. If there is no data in one of these registers, the I$^2$C Controller waits until data is loaded. If this bit is set while the I$^2$C Controller is shifting out data, it generates a START condition after the byte shifts and the acknowledge phase completed. If the `STOP` bit is also set, it also waits until the STOP condition is sent before the START condition. If this bit is 1, it cannot be cleared to 0 by writing to the register.This bit clears when the I$^2$C is disabled.

STOP—Send Stop Condition
This bit causes the I$^2$C Controller to issue a Stop condition after the byte in the I$^2$C Shift register has completed transmission or after a byte has been received in a receive operation. Once set, this bit is reset by the I$^2$C Controller after a Stop condition has been sent or by deasserting the `IEN` bit. If this bit is 1, it cannot be cleared to 0 by writing to the register.This bit clears when the I$^2$C is disabled.

BIRQ—Baud Rate Generator Interrupt Request
This bit causes an interrupt to occur every time the baud rate generator counts down to zero. This bit allows the I$^2$C Controller to be used as an additional counter when it is not being used elsewhere. This bit must only be set when the I$^2$C Controller is disabled.

TXI—Enable TDRE interrupts
This bit enables interrupts when the I$^2$C Data register is empty on the I$^2$C Controller.

NAK—Send NAK
This bit sends a Not Acknowledge condition after the next byte of data has been read from the I$^2$C slave. Once asserted, it is deasserted after a Not Acknowledge is sent or the `IEN` bit is deasserted.

FLUSH—Flush Data
Setting this bit to 1 clears the I$^2$C Data register and sets the TDRE bit to 1. This bit allows flushing of the I$^2$C Data register when an NAK is received after the data has been sent to the I$^2$C Data register. Reading this bit always returns 0.

FILTEN—I$^2$C Signal Filter Enable
Setting this bit to 1 enables low-pass digital filters on the SDA and SCL input signals. These filters reject any input pulse with periods less than a full system clock cycle. The filters introduce a 3-system clock cycle latency on the inputs.

## Configuring DMA0 and DMA1 for Data Transfer

Follow these steps to configure and enable DMA0 or DMA1:

1. Write to the DMA*x* I/O Address register to set the Register File address identifying the on-chip peripheral control register. The upper nibble of the 12-bit address for on-chip peripheral control registers is always `FH`. The full address is {FH, DMA*x*_IO[7:0]}
2. Determine the 12-bit Start and End Register File addresses. The 12-bit Start Address is given by {DMA*x*_H[3:0], DMA_START[7:0]}. The 12-bit End Address is given by {DMA*x*_H[7:4], DMA_END[7:0]}.
3. Write the Start and End Register File address high nibbles to the DMA*x* End/Start Address High Nibble register.
4. Write the lower byte of the Start Address to the DMA*x* Start/Current Address register.
5. Write the lower byte of the End Address to the DMA*x* End Address register.
6. Write to the DMA*x* Control register to complete the following:
   – Select loop or single-pass mode operation
   – Select the data transfer direction (either from the Register File RAM to the on-chip peripheral control register; or from the on-chip peripheral control register to the Register File RAM)
   – Enable the DMA*x* interrupt request, if desired
   – Select Word or Byte mode
   – Select the DMA*x* request trigger
   – Enable the DMA*x* channel

## DMA_ADC Operation

DMA_ADC transfers data from the ADC to the Register File. The sequence of operations in a DMA_ADC data transfer is:

1. ADC completes conversion on the current ADC input channel and signals the DMA controller that two-bytes of ADC data are ready for transfer.
2. DMA_ADC requests control of the system bus (address and data) from the eZ8 CPU.
3. After the eZ8 CPU acknowledges the bus request, DMA_ADC transfers the two-byte ADC output value to the Register File and then returns system bus control back to the eZ8 CPU.
4. If the current ADC Analog Input is the highest numbered input to be converted:
   – DMA_ADC resets the ADC Analog Input number to 0 and initiates data conversion on ADC Analog Input 0.
   – If configured to generate an interrupt, DMA_ADC sends an interrupt request to the Interrupt Controller

## Flash Operation Timing Using the Flash Frequency Registers

Before performing either a program or erase operation on the Flash memory, the user must first configure the Flash Frequency High and Low Byte registers. The Flash Frequency registers allow programming and erasure of the Flash with system clock frequencies ranging from 32KHz (32768Hz) through 20MHz.

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

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

**Caution:** Flash programming and erasure are not supported for system clock frequencies below 32KHz (32768Hz) or above 20MHz. The Flash Frequency High and Low Byte registers must be loaded with the correct value to insure proper operation of the Z8F640x family device.

## Flash Code Protection Against External Access

The user code contained within the Z8F640x family device's Flash memory can be protected against external access via the On-Chip Debugger. Programming the `RP` Option Bit prevents reading of the user code through the On-Chip Debugger. Refer to the **Option Bits** chapter and the **On-Chip Debugger** chapter for more information.

## Flash Code Protection Against Accidental Program and Erasure

The Z8F640x family device provides several levels of protection against accidental program and erasure of the Flash memory contents. This protection is provided by a combination of the Option bits and the locking mechanism of the Flash Controller.

**Table 93. On-Chip Debugger Commands**

| Debug Command | Command Byte | Enabled when NOT in Debug mode? | Disabled by Read Protect Option Bit |
|---|---|---|---|
| Write Program Memory | 0AH | - | Disabled |
| Read Program Memory | 0BH | - | Disabled |
| Write Data Memory | 0CH | - | Yes |
| Read Data Memory | 0DH | - | - |
| Read Program Memory CRC | 0EH | - | - |
| Reserved | 0FH | - | - |
| Step Instruction | 10H | - | Disabled |
| Stuff Instruction | 11H | - | Disabled |
| Execute Instruction | 12H | - | Disabled |
| Reserved | 13H - 1FH | - | - |
| Write Watchpoint | 20H | - | Disabled |
| Read Watchpoint | 21H | - | - |
| Reserved | 22H - FFH | - | - |

In the following bulleted list of OCD Commands, data and commands sent from the host to the On-Chip Debugger are identified by ’`DBG <-- Command/Data`’. Data sent from the On-Chip Debugger back to the host is identified by ’`DBG --> Data`’

- **Read OCD Revision (00H)**—The Read OCD Revision command is used to determine the version of the On-Chip Debugger. If OCD commands are added, removed, or changed, this revision number changes.

```
DBG <-- 00H
DBG --> OCDREV[15:8] (Major revision number)
DBG --> OCDREV[7:0] (Minor revision number)
```

- **Read OCD Status Register (02H)**—The Read OCD Status Register command is used to read the OCDSTAT register.

```
DBG <-- 02H
DBG --> OCDSTAT[7:0]
```

- **Read Runtime Counter (03H)**—The Runtime Counter is used to count Z8 Encore! system clock cycles in between Breakpoints. The 16-bit Runtime Counter counts up from `0000H` and stops at the maximum count of `FFFFH`. The Runtime Counter is overwritten during the Write Memory, Read Memory, Write Register, Read Register, Read Memory CRC, Step Instruction, Stuff Instruction, and Execute Instruction commands.

# On-Chip Oscillator

The Z8F640x family devices feature an on-chip oscillator for use with an external 1-20MHz crystal. This oscillator generates the primary system clock for the internal eZ8 CPU and the majority of the on-chip peripherals. Alternatively, the $X_{IN}$ input pin can also accept a CMOS-level clock input signal (32kHz-20MHz). If an external clock generator is used, the $X_{OUT}$ pin must be left unconnected. The Z8F640x family device does ***not*** contain in internal clock divider. The frequency of the signal on the $X_{IN}$ input pin determines the frequency of the system clock. The Z8F640x family device on-chip oscillator does not support external RC networks or ceramic resonators.

## 20MHz Crystal Oscillator Operation

Figure 90 illustrates a recommended configuration for connection with an external 20MHz, fundamental-mode, parallel-resonant crystal. Recommended crystal specifications are provided in Table 99. Resistor $R_1$ limits total power dissipation by the crystal. Printed circuit board layout should add no more than 4pF of stray capacitance to either the $X_{IN}$ or $X_{OUT}$ pins. If oscillation does not occur, reduce the values of capacitors $C_1$ and $C_2$ to decrease loading.

## SPI Slave Mode Timing

Figure 97 and Table 111 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.

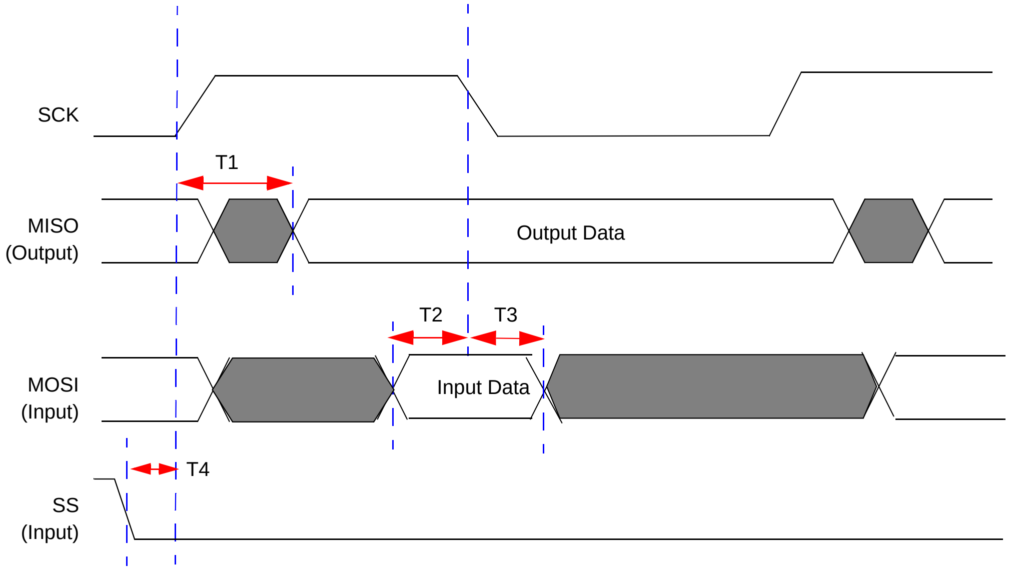

**Figure 97. SPI Slave Mode Timing**

**Table 111. SPI Slave Mode Timing**

| | | **Delay (ns)** | |
|---|---|---|---|
| **Parameter** | **Abbreviation** | **Minimum** | **Maximum** |
| $T_1$ | 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 | |

# *Opcode Maps*

Figures 101 and 102 provide information on each of the eZ8 CPU instructions. A description of the opcode map data and the abbreviations are provided in Figure 100 and Table 127.

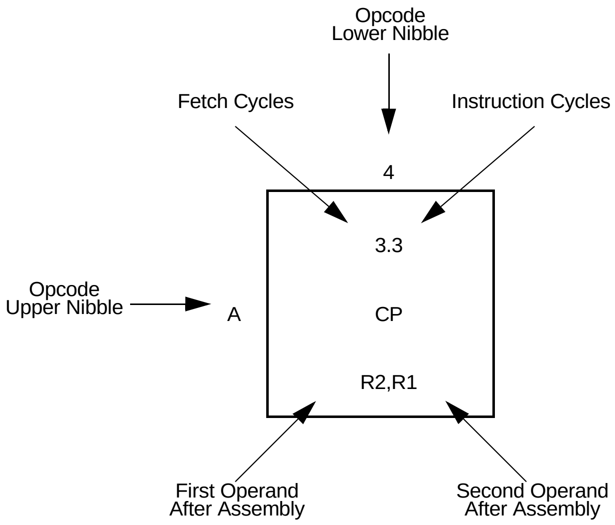

**Figure 100. Opcode Map Cell Description**

**Table 127. Opcode Map Abbreviations**

| Abbreviation | Description | Abbreviation | Description |
|---|---|---|---|
| b | Bit position | IRR | Indirect Register Pair |
| cc | Condition code | p | Polarity (0 or 1) |
| X | 8-bit signed index or displacement | r | 4-bit Working Register |
| DA | Destination address | R | 8-bit register |
| ER | Extended Addressing register | r1, R1, Ir1, Irr1, IR1, rr1, RR1, IRR1, ER1 | Destination address |
| IM | Immediate data value | r2, R2, Ir2, Irr2, IR2, rr2, RR2, IRR2, ER2 | Source address |
| Ir | Indirect Working Register | RA | Relative |
| IR | Indirect register | rr | Working Register Pair |
| Irr | Indirect Working Register Pair | RR | Register Pair |

Lower Nibble (Hex)

| Upper Nibble (Hex) | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1.2 **BRK** | 2.2 **SRP** IM | 2.3 **ADD** r1,r2 | 2.4 **ADD** r1,Ir2 | 3.3 **ADD** R2,R1 | 3.4 **ADD** IR2,R1 | 3.3 **ADD** R1,IM | 3.4 **ADD** IR1,IM | 4.3 **ADDX** ER2,ER1 | 4.3 **ADDX** IM,ER1 | 2.3 **DJNZ** r1,X | 2.2 **JR** cc,X | 2.2 **LD** r1,IM | 3.2 **JP** cc,DA | 1.2 **INC** r1 | 1.2 **NOP** |
| 1 | 2.2 **RLC** R1 | 2.3 **RLC** IR1 | 2.3 **ADC** r1,r2 | 2.4 **ADC** r1,Ir2 | 3.3 **ADC** R2,R1 | 3.4 **ADC** IR2,R1 | 3.3 **ADC** R1,IM | 3.4 **ADC** IR1,IM | 4.3 **ADCX** ER2,ER1 | 4.3 **ADCX** IM,ER1 | ↓ | ↓ | ↓ | ↓ | ↓ | See 2nd Opcode Map |
| 2 | 2.2 **INC** R1 | 2.3 **INC** IR1 | 2.3 **SUB** r1,r2 | 2.4 **SUB** r1,Ir2 | 3.3 **SUB** R2,R1 | 3.4 **SUB** IR2,R1 | 3.3 **SUB** R1,IM | 3.4 **SUB** IR1,IM | 4.3 **SUBX** ER2,ER1 | 4.3 **SUBX** IM,ER1 | ↓ | ↓ | ↓ | ↓ | ↓ | |
| 3 | 2.2 **DEC** R1 | 2.3 **DEC** IR1 | 2.3 **SBC** r1,r2 | 2.4 **SBC** r1,Ir2 | 3.3 **SBC** R2,R1 | 3.4 **SBC** IR2,R1 | 3.3 **SBC** R1,IM | 3.4 **SBC** IR1,IM | 4.3 **SBCX** ER2,ER1 | 4.3 **SBCX** IM,ER1 | ↓ | ↓ | ↓ | ↓ | ↓ | |
| 4 | 2.2 **DA** R1 | 2.3 **DA** IR1 | 2.3 **OR** r1,r2 | 2.4 **OR** r1,Ir2 | 3.3 **OR** R2,R1 | 3.4 **OR** IR2,R1 | 3.3 **OR** R1,IM | 3.4 **OR** IR1,IM | 4.3 **ORX** ER2,ER1 | 4.3 **ORX** IM,ER1 | ↓ | ↓ | ↓ | ↓ | ↓ | |
| 5 | 2.2 **POP** R1 | 2.3 **POP** IR1 | 2.3 **AND** r1,r2 | 2.4 **AND** r1,Ir2 | 3.3 **AND** R2,R1 | 3.4 **AND** IR2,R1 | 3.3 **AND** R1,IM | 3.4 **AND** IR1,IM | 4.3 **ANDX** ER2,ER1 | 4.3 **ANDX** IM,ER1 | ↓ | ↓ | ↓ | ↓ | ↓ | 1.2 **WDT** |
| 6 | 2.2 **COM** R1 | 2.3 **COM** IR1 | 2.3 **TCM** r1,r2 | 2.4 **TCM** r1,Ir2 | 3.3 **TCM** R2,R1 | 3.4 **TCM** IR2,R1 | 3.3 **TCM** R1,IM | 3.4 **TCM** IR1,IM | 4.3 **TCMX** ER2,ER1 | 4.3 **TCMX** IM,ER1 | ↓ | ↓ | ↓ | ↓ | ↓ | 1.2 **STOP** |
| 7 | 2.2 **PUSH** R2 | 2.3 **PUSH** IR2 | 2.3 **TM** r1,r2 | 2.4 **TM** r1,Ir2 | 3.3 **TM** R2,R1 | 3.4 **TM** IR2,R1 | 3.3 **TM** R1,IM | 3.4 **TM** IR1,IM | 4.3 **TMX** ER2,ER1 | 4.3 **TMX** IM,ER1 | ↓ | ↓ | ↓ | ↓ | ↓ | 1.2 **HALT** |
| 8 | 2.5 **DECW** RR1 | 2.6 **DECW** IRR1 | 2.5 **LDE** r1,Irr2 | 2.9 **LDEI** Ir1,Irr2 | 3.2 **LDX** r1,ER2 | 3.3 **LDX** Ir1,ER2 | 3.4 **LDX** IRR2,R1 | 3.5 **LDX** IRR2,IR1 | 3.4 **LDX** r1,rr2,X | 3.4 **LDX** rr1,r2,X | ↓ | ↓ | ↓ | ↓ | ↓ | 1.2 **DI** |
| 9 | 2.2 **RL** R1 | 2.3 **RL** IR1 | 2.5 **LDE** r2,Irr1 | 2.9 **LDEI** Ir2,Irr1 | 3.2 **LDX** r2,ER1 | 3.3 **LDX** Ir2,ER1 | 3.4 **LDX** R2,IRR1 | 3.5 **LDX** IR2,IRR1 | 3.3 **LEA** r1,r2,X | 3.5 **LEA** rr1,rr2,X | ↓ | ↓ | ↓ | ↓ | ↓ | 1.2 **EI** |
| A | 2.5 **INCW** RR1 | 2.6 **INCW** IRR1 | 2.3 **CP** r1,r2 | 2.4 **CP** r1,Ir2 | 3.3 **CP** R2,R1 | 3.4 **CP** IR2,R1 | 3.3 **CP** R1,IM | 3.4 **CP** IR1,IM | 4.3 **CPX** ER2,ER1 | 4.3 **CPX** IM,ER1 | ↓ | ↓ | ↓ | ↓ | ↓ | 1.4 **RET** |
| B | 2.2 **CLR** R1 | 2.3 **CLR** IR1 | 2.3 **XOR** r1,r2 | 2.4 **XOR** r1,Ir2 | 3.3 **XOR** R2,R1 | 3.4 **XOR** IR2,R1 | 3.3 **XOR** R1,IM | 3.4 **XOR** IR1,IM | 4.3 **XORX** ER2,ER1 | 4.3 **XORX** IM,ER1 | ↓ | ↓ | ↓ | ↓ | ↓ | 1.5 **IRET** |
| C | 2.2 **RRC** R1 | 2.3 **RRC** IR1 | 2.5 **LDC** r1,Irr2 | 2.9 **LDCI** Ir1,Irr2 | 2.3 **JP** IRR1 | 2.9 **LDC** Ir1,Irr2 | | 3.3 **LD** r1,r2,X | 3.2 **PUSHX** ER2 | | ↓ | ↓ | ↓ | ↓ | ↓ | 1.2 **RCF** |
| D | 2.2 **SRA** R1 | 2.3 **SRA** IR1 | 2.5 **LDC** r2,Irr1 | 2.9 **LDCI** Ir2,Irr1 | 2.6 **CALL** IRR1 | 2.2 **BSWAP** R1 | 3.3 **CALL** DA | 3.4 **LD** r2,r1,X | 3.2 **POPX** ER1 | | ↓ | ↓ | ↓ | ↓ | ↓ | 1.2 **SCF** |
| E | 2.2 **RR** R1 | 2.3 **RR** IR1 | 2.2 **BIT** p,b,r1 | 2.3 **LD** r1,Ir2 | 3.2 **LD** R2,R1 | 3.3 **LD** IR2,R1 | 3.2 **LD** R1,IM | 3.3 **LD** IR1,IM | 4.2 **LDX** ER2,ER1 | 4.2 **LDX** IM,ER1 | ↓ | ↓ | ↓ | ↓ | ↓ | 1.2 **CCF** |
| F | 2.2 **SWAP** R1 | 2.3 **SWAP** IR1 | 2.6 **TRAP** Vector | 2.3 **LD** Ir1,r2 | 2.8 **MULT** RR1 | 3.3 **LD** R2,IR1 | 3.3 **BTJ** p,b,r1,X | 3.4 **BTJ** p,b,Ir1,X | | | ↓ | ↓ | ↓ | ↓ | ↓ | |

**Figure 101. First Opcode Map**

# Ordering Information

**Table 128. Ordering Information**

| Part | Flash KB (Bytes) | RAM KB (Bytes) | Max. Speed (MHz) | Temp ($^0C$) | Voltage (V) | Package | Part Number |
|---|---|---|---|---|---|---|---|
| **Z8 Encore!® with 16KB Flash, Standard Temperature** | | | | | | | |
| Z8 Encore!® | 16 (16,384) | 2 (2048) | 20 | 0 to +70 | 3.0 - 3.6 | PDIP-40 | Z8F1601PM020SC |
| Z8 Encore!® | 16 (16,384) | 2 (2048) | 20 | 0 to +70 | 3.0 - 3.6 | LQFP-44 | Z8F1601AN020SC |
| Z8 Encore!® | 16 (16,384) | 2 (2048) | 20 | 0 to +70 | 3.0 - 3.6 | PLCC-44 | Z8F1601VN020SC |
| Z8 Encore!® | 16 (16,384) | 2 (2048) | 20 | 0 to +70 | 3.0 - 3.6 | LQFP-64 | Z8F1602AR020SC |
| Z8 Encore!® | 16 (16,384) | 2 (2048) | 20 | 0 to +70 | 3.0 - 3.6 | PLCC-68 | Z8F1602VS020SC |
| **Z8 Encore!® with 24KB Flash, Standard Temperature** | | | | | | | |
| Z8 Encore!® | 24 (24,576) | 2 (2048) | 20 | 0 to +70 | 3.0 - 3.6 | PDIP-40 | Z8F2401PM020SC |
| Z8 Encore!® | 24 (24,576) | 2 (2048) | 20 | 0 to +70 | 3.0 - 3.6 | LQFP-44 | Z8F2401AN020SC |
| Z8 Encore!® | 24 (24,576) | 2 (2048) | 20 | 0 to +70 | 3.0 - 3.6 | PLCC-44 | Z8F2401VN020SC |
| Z8 Encore!® | 24 (24,576) | 2 (2048) | 20 | 0 to +70 | 3.0 - 3.6 | LQFP-64 | Z8F2402AR020SC |
| Z8 Encore!® | 24 (24,576) | 2 (2048) | 20 | 0 to +70 | 3.0 - 3.6 | PLCC-68 | Z8F2402VS020SC |
| **Z8 Encore!® with 32KB Flash, Standard Temperature** | | | | | | | |
| Z8 Encore!® | 32 (32,768) | 2 (2048) | 20 | 0 to +70 | 3.0 - 3.6 | PDIP-40 | Z8F3201PM020SC |
| Z8 Encore!® | 32 (32,768) | 2 (2048) | 20 | 0 to +70 | 3.0 - 3.6 | LQFP-44 | Z8F3201AN020SC |
| Z8 Encore!® | 32 (32,768) | 2 (2048) | 20 | 0 to +70 | 3.0 - 3.6 | PLCC-44 | Z8F3201VN020SC |
| Z8 Encore!® | 32 (32,768) | 2 (2048) | 20 | 0 to +70 | 3.0 - 3.6 | LQFP-64 | Z8F3202AR020SC |
| Z8 Encore!® | 32 (32,768) | 2 (2048) | 20 | 0 to +70 | 3.0 - 3.6 | PLCC-68 | Z8F3202VS020SC |
| **Z8 Encore!®with 48KB Flash, Standard Temperature** | | | | | | | |
| Z8 Encore!® | 48 (49,152) | 4 (4096) | 20 | 0 to +70 | 3.0 - 3.6 | PDIP-40 | Z8F4801PM020SC |
| Z8 Encore!® | 48 (49,152) | 4 (4096) | 20 | 0 to +70 | 3.0 - 3.6 | LQFP-44 | Z8F4801AN020SC |
| Z8 Encore!® | 48 (49,152) | 4 (4096) | 20 | 0 to +70 | 3.0 - 3.6 | PLCC-44 | Z8F4801VN020SC |
| Z8 Encore!® | 48 (49,152) | 4 (4096) | 20 | 0 to +70 | 3.0 - 3.6 | LQFP-64 | Z8F4802AR020SC |
| Z8 Encore!® | 48 (49,152) | 4 (4096) | 20 | 0 to +70 | 3.0 - 3.6 | PLCC-68 | Z8F4802VS020SC |
| Z8 Encore!® | 48 (49,152) | 4 (4096) | 20 | 0 to +70 | 3.0 - 3.6 | QFP-80 | Z8F4803FT020SC |

## Problem Description or Suggestion

Provide a complete description of the problem or your suggestion. If you are reporting a specific problem, include all steps leading up to the occurrence of the problem. Attach additional pages as necessary.

___________________________________________________________________________
___________________________________________________________________________
___________________________________________________________________________
___________________________________________________________________________
___________________________________________________________________________
___________________________________________________________________________
___________________________________________________________________________
___________________________________________________________________________
___________________________________________________________________________
___________________________________________________________________________
___________________________________________________________________________