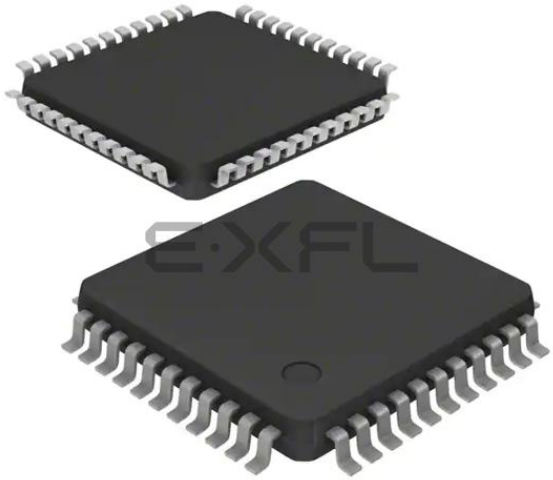

**Welcome to E-XFL.COM**

## 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 | Active |
| 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 | 31 |
| 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 8x10b |
| Oscillator Type | Internal |
| Operating Temperature | 0°C ~ 70°C (TA) |
| Mounting Type | Surface Mount |
| Package / Case | 44-LQFP |
| Supplier Device Package | 44-LQFP (10x10) |
| Purchase URL | https://www.e-xfl.com/product-detail/zilog/z8f4821an020sg |

# Revision History

Each instance in the Revision History table reflects a change to this document from its previous revision. For more details, refer to the corresponding pages or appropriate links listed in the table below.

| Date | Revision Level | Description | Page |
|---|---|---|---|
| Jan 2013 | 24 | Restored 40-pin PDIP package to Signal and Pin Descriptions and Packaging chapters. | 7, 286 |
| Feb 2012 | 23 | Corrected formatting of $I_{DDS}$ section, Table 107; corrected language in the General Purpose RAM section of Appendix A; | 202, 248 |
| Sep 2011 | 22 | Revised Flash Sector Protect Register description; revised Packaging chapter. | 178, 286 |
| Mar 2008 | 21 | Changed title to Z8 Encore! XP F64xx Series. | All |
| Feb 2008 | 20 | Changed Z8 Encore! XP 64K Series Flash Microcontrollers to Z8 Encore! XP F64xx Series Flash Microcontrollers. Deleted three sentences that mentioned Z8R642. Removed the 40 PDIP package. Added ZENETSC0100ZACG to the end of the Ordering Information table. Changed the flag status to unaffected for BIT, BSET, and BCLR in the eZ8 CPU Instruction Summary table. | 287, 234 |
| Dec 2007 | 19 | Updated Zilog logo, Disclaimer section, and implemented style guide. Updated Table 113. Changed Z8 Encore! 64K Series to Z8 Encore! XP 64K Series Flash Microcontrollers throughout the document. | All |
| Dec 2006 | 18 | Updated Flash Memory Electrical Characteristics and Timing table and Ordering Information chapter. | 213, 287 |
| Nov 2006 | 17 | Updated Part Number Suffix Designations section. | 292 |

# Register File Address Map

Table 7 provides the address map for the Register File of the Z8 Encore! XP F64xx Series products. Not all devices and package styles in the Z8 Encore! XP F64xx Series support Timer 3 and all of the GPIO ports. Consider registers for unimplemented peripherals to be reserved.

**Table 7. Z8 Encore! XP F64xx Series Register File Address Map**

| Address (Hex) | Register Description | Mnemonic | Reset (Hex) | Page |
|---|---|---|---|---|
| **General-Purpose RAM** | | | | |
| 000–EFF | General-Purpose Register File RAM | — | XX | |
| **Timer 0** | | | | |
| F00 | Timer 0 High Byte | T0H | 00 | 72 |
| F01 | Timer 0 Low Byte | T0L | 01 | 72 |
| F02 | Timer 0 Reload High Byte | T0RH | FF | 74 |
| F03 | Timer 0 Reload Low Byte | T0RL | FF | 74 |
| F04 | Timer 0 PWM High Byte | T0PWMH | 00 | 75 |
| F05 | Timer 0 PWM Low Byte | T0PWML | 00 | 75 |
| F06 | Timer 0 Control 0 | T0CTL0 | 00 | 76 |
| F07 | Timer 0 Control 1 | T0CTL1 | 00 | 77 |
| **Timer 1** | | | | |
| F08 | Timer 1 High Byte | T1H | 00 | 72 |
| F09 | Timer 1 Low Byte | T1L | 01 | 72 |
| F0A | Timer 1 Reload High Byte | T1RH | FF | 74 |
| F0B | Timer 1 Reload Low Byte | T1RL | FF | 74 |
| F0C | Timer 1 PWM High Byte | T1PWMH | 00 | 75 |
| F0D | Timer 1 PWM Low Byte | T1PWML | 00 | 75 |
| F0E | Timer 1 Control 0 | T1CTL0 | 00 | 76 |
| F0F | Timer 1 Control 1 | T1CTL1 | 00 | 77 |
| **Timer 2** | | | | |
| F10 | Timer 2 High Byte | T2H | 00 | 72 |
| F11 | Timer 2 Low Byte | T2L | 01 | 72 |
| F12 | Timer 2 Reload High Byte | T2RH | FF | 74 |
| F13 | Timer 2 Reload Low Byte | T2RL | FF | 74 |

Note: XX = Undefined.

# Interrupt Controller

The interrupt controller on the Z8 Encore! XP F64xx Series products prioritizes the interrupt requests from the on-chip peripherals and the GPIO port pins. The features of the interrupt controller include:

- 24 unique interrupt vectors:
  - 12 GPIO port pin interrupt sources
  - 12 on-chip peripheral interrupt sources
- Flexible GPIO interrupts
  - Eight selectable rising and falling edge GPIO interrupts
  - Four dual-edge interrupts
- Three levels of individually programmable interrupt priority
- Watchdog Timer can be configured to generate an interrupt

Interrupt requests (IRQs) allow peripheral devices to suspend CPU operation in an orderly manner and force the CPU to start an interrupt service routine (ISR). Usually this interrupt service routine is involved with the exchange of data, status information, or control information between the CPU and the interrupting peripheral. When the service routine is completed, the CPU returns to the operation from which it was interrupted.

The eZ8 CPU supports both vectored and polled interrupt handling. For polled interrupts, the interrupt control has no effect on operation. For more information about interrupt servicing by the eZ8 CPU, refer to the eZ8 CPU Core User Manual (UM0128), which is available for download on www.zilog.com.

## Interrupt Vector Listing

Table 23 lists all of the interrupts available in order of priority. The interrupt vector is stored with the most significant byte (MSB) at the even program memory address and the least significant byte (LSB) at the following odd program memory address.

6. Configure the associated GPIO port pin for the timer input alternate function.
7. 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:

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

## COMPARE Mode

In COMPARE Mode, the timer counts up to the 16-bit maximum compare value stored in the Timer Reload High and Low Byte registers. The timer input is the system clock. Upon reaching the compare value, the timer generates an interrupt and counting continues (the timer value is not reset to 0001H). 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 compare.

If the Timer reaches FFFFH, the timer rolls over to 0000H and continue counting.

Observe the following procedure for configuring a timer for COMPARE Mode and initiating the count:

1. Write to the Timer Control 1 Register to:
   - Disable the timer
   - Configure the timer for COMPARE Mode
   - Set the prescale value
   - Set the initial logic level (High or Low) for the timer output alternate function, if appropriate
2. Write to the Timer High and Low Byte registers to set the starting count value.
3. Write to the Timer Reload High and Low Byte registers to set the compare value.
4. If appropriate, enable the timer interrupt and set the timer interrupt priority by writing to the relevant interrupt registers.
5. If using the timer output function, configure the associated GPIO port pin for the timer output alternate function.
6. Write to the Timer Control 1 Register to enable the timer and initiate counting.

In COMPARE Mode, the system clock always provides the timer input. The compare time is calculated using the following equation:

ister. 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

A 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

This section defines the features of the following Timer Control registers.

Timer 0–3 High and Low Byte Registers: see page 72

Timer Reload High and Low Byte Registers: see page 74

Timer 0–3 PWM High and Low Byte Registers: see page 75

Timer 0–3 Control 0 Registers: see page 76

Timer 0–3 Control 1 Registers: see page 77

Timers 0–2 are available in all packages. Timer 3 is only available in 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, shown in Tables 39 and 40, 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 read 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 written during counting, the 8-bit written value is placed in the counter (High or Low Byte) at the next clock edge. The counter continues counting from the new value.

Timer 3 is unavailable in 44-pin packages.

## UART Control 0 and Control 1 Registers

The UART Control 0 and Control 1 Registers, shown in Tables 57 and 58, configure the properties of the UART's transmit and receive operations. The UART Control registers must not been written while the UART is enabled.

**Table 57. UART Control 0 Register (U*x*CTL0)**

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

| Bit | Description |
|---|---|
| [7]<br>TEN | **Transmit Enable**<br>This bit enables or disables the transmitter. The enable is also controlled by the $\overline{\text{CTS}}$ signal and the CTSE bit. If the $\overline{\text{CTS}}$ signal is low and the CTSE bit is 1, the transmitter is enabled.<br>0 = Transmitter disabled.<br>1 = Transmitter enabled. |
| [6]<br>REN | **Receive Enable**<br>This bit enables or disables the receiver.<br>0 = Receiver disabled.<br>1 = Receiver enabled. |
| [5]<br>CTSE | **CTS Enable**<br>0 = The $\overline{\text{CTS}}$ signal has no effect on the transmitter.<br>1 = The UART recognizes the $\overline{\text{CTS}}$ signal as an enable control from the transmitter. |
| [4]<br>PEN | **Parity Enable**<br>This bit enables or disables parity. Even or odd is determined by the PSEL bit. It is overridden by the MPEN bit.<br>0 = Parity is disabled.<br>1 = The transmitter sends data with an additional parity bit and the receiver receives an additional parity bit. |
| [3]<br>PSEL | **Parity Select**<br>0 = Even parity is transmitted and expected on all received data.<br>1 = Odd parity is transmitted and expected on all received data. |
| [2]<br>SBRK | **Send Break**<br>This bit pauses or breaks data transmission. Sending a break interrupts any transmission in progress, so ensure that the transmitter has finished sending data before setting this bit.<br>0 = No break is sent.<br>1 = The output of the transmitter is zero. |

**Table 62. UART Baud Rates (Continued)**

| 10.0 MHz System Clock | | | | 5.5296 MHz System Clock | | | |
|---|---|---|---|---|---|---|---|
| Desired Rate (kHz) | BRG Divisor (Decimal) | Actual Rate (kHz) | Error (%) | Desired Rate (kHz) | BRG Divisor (Decimal) | Actual Rate (kHz) | Error (%) |
| 1250.0 | N/A | N/A | N/A | 1250.0 | N/A | N/A | N/A |
| 625.0 | 1 | 625.0 | 0.00 | 625.0 | N/A | N/A | N/A |
| 250.0 | 3 | 208.33 | –16.67 | 250.0 | 1 | 345.6 | 38.24 |
| 115.2 | 5 | 125.0 | 8.51 | 115.2 | 3 | 115.2 | 0.00 |
| 57.6 | 11 | 56.8 | –1.36 | 57.6 | 6 | 57.6 | 0.00 |
| 38.4 | 16 | 39.1 | 1.73 | 38.4 | 9 | 38.4 | 0.00 |
| 19.2 | 33 | 18.9 | 0.16 | 19.2 | 18 | 19.2 | 0.00 |
| 9.60 | 65 | 9.62 | 0.16 | 9.60 | 36 | 9.60 | 0.00 |
| 4.80 | 130 | 4.81 | 0.16 | 4.80 | 72 | 4.80 | 0.00 |
| 2.40 | 260 | 2.40 | –0.03 | 2.40 | 144 | 2.40 | 0.00 |
| 1.20 | 521 | 1.20 | –0.03 | 1.20 | 288 | 1.20 | 0.00 |
| 0.60 | 1042 | 0.60 | –0.03 | 0.60 | 576 | 0.60 | 0.00 |
| 0.30 | 2083 | 0.30 | 0.2 | 0.30 | 1152 | 0.30 | 0.00 |
| **3.579545 MHz System Clock** | | | | **1.8432 MHz System Clock** | | | |
| Desired Rate (kHz) | BRG Divisor (Decimal) | Actual Rate (kHz) | Error (%) | Desired Rate (kHz) | BRG Divisor (Decimal) | Actual Rate (kHz) | Error (%) |
| 1250.0 | N/A | N/A | N/A | 1250.0 | N/A | N/A | N/A |
| 625.0 | N/A | N/A | N/A | 625.0 | N/A | N/A | N/A |
| 250.0 | 1 | 223.72 | –10.51 | 250.0 | N/A | N/A | N/A |
| 115.2 | 2 | 111.9 | –2.90 | 115.2 | 1 | 115.2 | 0.00 |
| 57.6 | 4 | 55.9 | –2.90 | 57.6 | 2 | 57.6 | 0.00 |
| 38.4 | 6 | 37.3 | –2.90 | 38.4 | 3 | 38.4 | 0.00 |
| 19.2 | 12 | 18.6 | –2.90 | 19.2 | 6 | 19.2 | 0.00 |
| 9.60 | 23 | 9.73 | 1.32 | 9.60 | 12 | 9.60 | 0.00 |
| 4.80 | 47 | 4.76 | –0.83 | 4.80 | 24 | 4.80 | 0.00 |
| 2.40 | 93 | 2.41 | 0.23 | 2.40 | 48 | 2.40 | 0.00 |
| 1.20 | 186 | 1.20 | 0.23 | 1.20 | 96 | 1.20 | 0.00 |
| 0.60 | 373 | 0.60 | –0.04 | 0.60 | 192 | 0.60 | 0.00 |
| 0.30 | 746 | 0.30 | –0.04 | 0.30 | 384 | 0.30 | 0.00 |

## SPI Diagnostic State Register

The SPI Diagnostic State Register, shown in Table 68, provides observability of internal state. This register is a read-only register that is used for SPI diagnostics.

**Table 68. SPI Diagnostic State Register (SPIDST)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | SCKEN | TCKEN | SPISTATE | | | | | |
| RESET | 0 | | | | | | | |
| R/W | R | | | | | | | |
| Address | F64H | | | | | | | |

| Bit | Description |
|---|---|
| [7]<br>SCKEN | **Shift Clock Enable**<br>0 = The internal Shift Clock Enable signal is deasserted.<br>1 = The internal Shift Clock Enable signal is asserted (shift register is updates on next system clock). |
| [6]<br>TCKEN | **Transmit Clock Enable**<br>0 = The internal Transmit Clock Enable signal is deasserted.<br>1 = The internal Transmit Clock Enable signal is asserted. When this is asserted the serial data out is updated on the next system clock (MOSI or MISO). |
| [5:0]<br>SPISTATE | **SPI State Machine**<br>Defines the current state of the internal SPI State Machine. |

## SPI Baud Rate High and Low Byte Registers

The SPI Baud Rate High and Low Byte registers, shown in Tables 69 and 70, combine to form a 16-bit reload value, BRG[15:0], for the SPI Baud Rate Generator.

When configured as a general purpose timer, the SPI BRG interrupt interval is calculated using the following equation:

SPI BRG Interrupt Interval (s) = System Clock Period (s) × BRG[15:0]

- The first bit of the byte of an address is shifting out and the RD bit of the I²C Status Register is deasserted.
- The first bit of a 10-bit address shifts out
- The first bit of write data shifts out

**Note:** Writing to the I²C Data Register always clears the TRDE bit to 0. When TDRE is asserted, the I²C Controller pauses at the beginning of the Acknowledge cycle of the byte currently shifting out. It does not resume until the Data Register is written with the next value to send or until the stop or start bits are set, indicating that the current byte is the last one to send.

The fourth interrupt source is the baud rate generator. If the I²C 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 counts down to 1. This allows the I²C baud rate generator to be used by software as a general purpose timer when IEN = 0.

## Software Control of I²C Transactions

Software can control I²C transactions by using the I²C Controller interrupt, by polling the I²C Status Register or by DMA. Note that not all products include a DMA Controller.

To use interrupts, the I²C interrupt must be enabled in the Interrupt Controller. The TXI bit in the I²C Control Register must be set to enable transmit interrupts.

To control transactions by polling, the interrupt bits (TDRE, RDRF and NCKI) in the I²C Status Register should be polled. The TDRE bit asserts regardless of the state of the TXI bit.

Either or both transmit and receive data movement can be controlled by the DMA Controller. The DMA Controller channel(s) must be initialized to select the I²C transmit and receive requests. Transmit DMA requests require that the TXI bit in the I²C Control Register be set.

**Caution:** A transmit (write) DMA operation hangs if the slave responds with a Not Acknowledge before the last byte has been sent. After receiving the Not Acknowledge, the I²C Controller sets the NCKI bit in the Status Register and pauses until either the stop or start bits in the Control Register are set.

For a receive (read) DMA transaction to send a Not Acknowledge on the last byte, the receive DMA must be set up to receive n-1 bytes, then software must set the NAK bit and receive the last (nth) byte directly.

2. Software asserts the TXI bit of the $I^2C$ Control Register to enable transmit interrupts.
3. The $I^2C$ interrupt asserts because the $I^2C$ Data Register is empty.
4. Software responds to the TDRE interrupt by writing the first slave address byte to the $I^2C$ Data Register. The least significant bit must be 0 for the write operation.
5. Software asserts the start bit of the $I^2C$ Control Register.
6. The $I^2C$ Controller sends the start condition to the $I^2C$ slave.
7. The $I^2C$ Controller loads the $I^2C$ Shift Register with the contents of the $I^2C$ Data Register.
8. After one bit of address is shifted out by the SDA signal, the transmit interrupt is asserted.
9. Software responds by writing the second byte of address into the contents of the $I^2C$ Data Register.
10. The $I^2C$ Controller shifts the rest of the first byte of address and write bit out the SDA signal.
11. If the $I^2C$ slave acknowledges the first address byte by pulling the SDA signal Low during the next High period of SCL, the $I^2C$ Controller sets the ACK bit in the $I^2C$ Status Register. Continue with Step 12.

    If the slave does not acknowledge the first address byte, the $I^2C$ Controller sets the NCKI bit and clears the ACK bit in the $I^2C$ Status Register. Software responds to the Not Acknowledge interrupt by setting the stop and flush bits and clearing the TXI bit. The $I^2C$ Controller sends the stop condition on the bus and clears the stop and NCKI bits. The transaction is complete (ignore the following steps).
12. The $I^2C$ Controller loads the $I^2C$ Shift Register with the contents of the $I^2C$ Data Register.
13. The $I^2C$ Controller shifts the second address byte out the SDA signal. After the first bit has been sent, the transmit interrupt is asserted.
14. Software responds by writing a data byte to the $I^2C$ Data Register.
15. The $I^2C$ Controller completes shifting the contents of the shift register on the SDA signal.
16. If the $I^2C$ slave sends an acknowledge by pulling the SDA signal Low during the next High period of SCL, the $I^2C$ Controller sets the ACK bit in the $I^2C$ Status Register. Continue with Step 17.

    If the slave does not acknowledge the second address byte or one of the data bytes, the $I^2C$ Controller sets the NCKI bit and clears the ACK bit in the $I^2C$ Status Register. Software responds to the Not Acknowledge interrupt by setting the stop and flush bits and clearing the TXI bit. The $I^2C$ Controller sends the stop condition on the bus and

For more information about bypassing the Flash Controller, refer to the Third Party Flash Programming Support for Z8 Encore! MCUs Application Note (AN0117), which is available for download at www.zilog.com.

### Flash Controller Behavior in Debug Mode

The following changes in Flash Controller behavior occur when the Flash Controller is accessed using the On-Chip Debugger:

- The Flash Write Protect option bit is ignored
- The Flash Sector Protect Register is ignored for programming and erase operations
- Programming operations are not limited to the page selected in the Page Select Register
- Bits in the Flash Sector Protect Register can be written to one or zero
- The second write of the Page Select Register to unlock the Flash Controller is not necessary
- The Page Select Register can be written when the Flash Controller is unlocked
- The Mass Erase command is enabled through the Flash Control Register

**!** **Caution:** For security reasons, the Flash Controller allows only a single page to be opened for write/erase operations. When writing multiple Flash pages, the Flash Controller must go through the unlock sequence again to select another page.

## Flash Control Register Definitions

This section defines the features of the following Flash Control registers.

Flash Control Register: see page 175

Flash Status Register: see page 177

Page Select Register: see page 177

Flash Sector Protect Register: see page 178

Flash Frequency High and Low Byte Registers: see page 179

### Flash Control Register

The Flash Control Register, shown in Table 93, unlocks the Flash Controller for programming and erase operations, or to select the Flash Sector Protect Register.

**Table 95. Page Select Register (FPS)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | INFO_EN | PAGE | | | | | | |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FF9H | | | | | | | |

| Bit | Description |
|---|---|
| [7]<br>INFO_EN | **Information Area Enable**<br>0 = Information Area is not selected.<br>1 = Information Area is selected. The Information area is mapped into the Flash memory address space at addresses FE00H through FFFFH. |
| [6:0]<br>PAGE | **Page Select**<br>This 7-bit field selects the Flash memory page for programming and Page Erase operations. Flash Memory Address[15:9] = PAGE[6:0]. |

## Flash Sector Protect Register

The Flash Sector Protect Register, shown in Table 96, protects Flash memory sectors from being programmed or erased from user code. The Flash Sector Protect Register shares its Register File address with the Page Select Register. The Flash Sector Protect Register can be accessed only after writing the Flash Control Register with 5EH.

User code can only write bits in this register to 1 (bits cannot be cleared to 0 by user code). To determine the appropriate Flash memory sector address range and sector number for your Z8F64xx Series product, please refer to Table 91 on page 169.

**Table 96. Flash Sector Protect Register (FPROT)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | SECT7 | SECT6 | SECT5 | SECT4 | SECT3 | SECT2 | SECT1 | SECT0 |
| RESET | 0 | | | | | | | |
| R/W | R/W* | | | | | | | |
| Address | FF9H | | | | | | | |

Note: *R/W = This register is accessible for read operations; it can be written to 1 only via user code.

| Bit | Description |
|---|---|
| [7:0]<br>SECT*n* | **Sector Protect****<br>0 = Sector *n* can be programmed or erased from user code.<br>1 = Sector *n* is protected and cannot be programmed or erased from user code. |

Note: **User code can only write bits from 0 to 1.

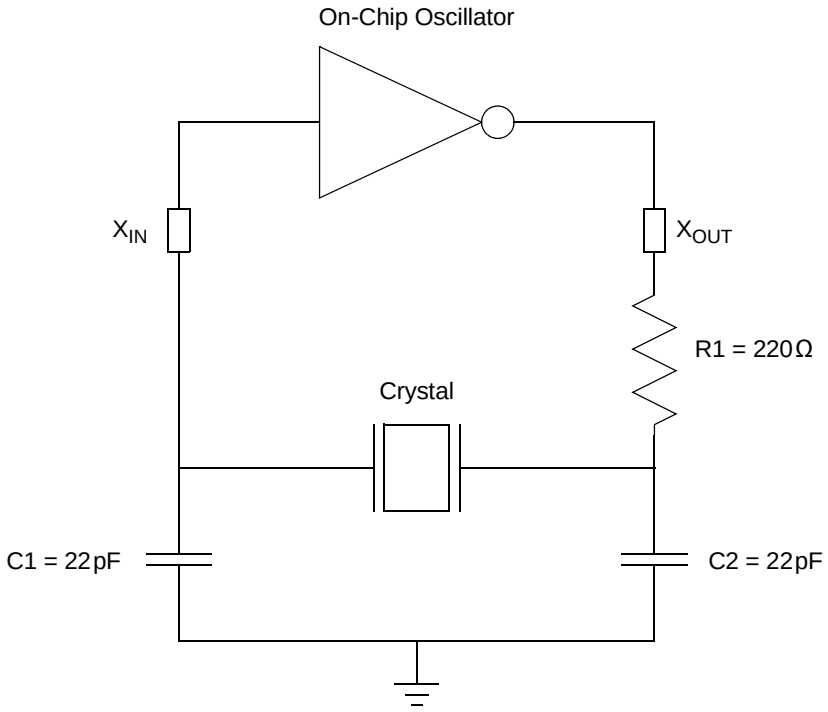


**Figure 40. Recommended 20MHz Crystal Oscillator Configuration**

**Table 105. Recommended Crystal Oscillator Specifications (20MHz Operation)**

| Parameter | Value | Units | Comments |
|---|---|---|---|
| Frequency | 20 | MHz | |
| Resonance | Parallel | | |
| Mode | Fundamental | | |
| Series Resistance ($R_S$) | 25 | W | Maximum |
| Load Capacitance ($C_L$) | 20 | pF | Maximum |
| Shunt Capacitance ($C_0$) | 7 | pF | Maximum |
| Drive Level | 1 | mW | Maximum |

# eZ8 CPU Instruction Classes

eZ8 CPU instructions can be divided functionally into the following groups:

- Arithmetic
- Bit Manipulation
- Block Transfer
- CPU Control
- Load
- Logical
- Program Control
- Rotate and Shift

Tables 128 through 135 contain the instructions belonging to each group and the number of operands required for each instruction. Some instructions appear in more than one table; these instructions can be considered to be a subset of more than one category. Within these tables, the source operand is identified as *src*, the destination operand is *dst* and a condition code is *cc*.

**Table 128. Arithmetic Instructions**

| Mnemonic | Operands | Instruction |
|---|---|---|
| ADC | dst, src | Add with Carry |
| ADCX | dst, src | Add with Carry using Extended Addressing |
| ADD | dst, src | Add |
| ADDX | dst, src | Add using Extended Addressing |
| CP | dst, src | Compare |
| CPC | dst, src | Compare with Carry |
| CPCX | dst, src | Compare with Carry using Extended Addressing |
| CPX | dst, src | Compare using Extended Addressing |
| DA | dst | Decimal Adjust |
| DEC | dst | Decrement |
| DECW | dst | Decrement Word |
| INC | dst | Increment |
| INCW | dst | Increment Word |

**Table 128. Arithmetic Instructions (Continued)**

| Mnemonic | Operands | Instruction |
|---|---|---|
| MULT | dst | Multiply |
| SBC | dst, src | Subtract with Carry |
| SBCX | dst, src | Subtract with Carry using Extended Addressing |
| SUB | dst, src | Subtract |
| SUBX | dst, src | Subtract using Extended Addressing |

**Table 129. Bit Manipulation Instructions**

| Mnemonic | Operands | Instruction |
|---|---|---|
| BCLR | bit, dst | Bit Clear |
| BIT | p, bit, dst | Bit Set or Clear |
| BSET | bit, dst | Bit Set |
| BSWAP | dst | Bit Swap |
| CCF | — | Complement Carry Flag |
| RCF | — | Reset Carry Flag |
| SCF | — | Set Carry Flag |
| TCM | dst, src | Test Complement Under Mask |
| TCMX | dst, src | Test Complement Under Mask using Extended Addressing |
| TM | dst, src | Test Under Mask |
| TMX | dst, src | Test Under Mask using Extended Addressing |

**Table 130. Block Transfer Instructions**

| Mnemonic | Operands | Instruction |
|---|---|---|
| LDCI | dst, src | Load Constant to/from program memory and Auto-Increment addresses |
| LDEI | dst, src | Load External Data to/from Data Memory and Auto-Increment addresses |

**Table 133. Logical Instructions**

| Mnemonic | Operands | Instruction |
|---|---|---|
| AND | dst, src | Logical AND |
| ANDX | dst, src | Logical AND using Extended Addressing |
| COM | dst | Complement |
| OR | dst, src | Logical OR |
| ORX | dst, src | Logical OR using Extended Addressing |
| XOR | dst, src | Logical Exclusive OR |
| XORX | dst, src | Logical Exclusive OR using Extended Addressing |

**Table 134. Program Control Instructions**

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

**Table 137. Op Code Map Abbreviations (Continued)**

| Abbreviation | Description | Abbreviation | Description |
|---|---|---|---|
| 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 |

### Hex Address: FD7

**Table 236. Port A–H Output Data Register (P*x*OUT)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | POUT7 | POUT6 | POUT5 | POUT4 | POUT3 | POUT2 | POUT1 | POUT0 |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FD3H, FD7H, FDBH, FDFH, FE3H, FE7H, FEBH, FEFH | | | | | | | |

### Hex Address: FD8

**Table 237. Port A–H GPIO Address Registers (P*x*ADDR)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | PADDR[7:0] | | | | | | | |
| RESET | 00H | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FD0H, FD4H, FD8H, FDCH, FE0H, FE4H, FE8H, FECH | | | | | | | |

### Hex Address: FD9

**Table 238. Port A–H Control Registers (P*x*CTL)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | PCTL | | | | | | | |
| RESET | 00H | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FD1H, FD5H, FD9H, FDDH, FE1H, FE5H, FE9H, FEDH | | | | | | | |

### Hex Address: FDA

**Table 239. Port A–H Input Data Registers (PxIN)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | PIN7 | PIN6 | PIN5 | PIN4 | PIN3 | PIN2 | PIN1 | PIN0 |
| RESET | X | | | | | | | |
| R/W | R | | | | | | | |
| Address | FD2H, FD6H, FDAH, FDEH, FE2H, FE6H, FEAH, FEEH | | | | | | | |

### Hex Address: FE7

**Table 252. Port A–H Output Data Register (P*x*OUT)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | POUT7 | POUT6 | POUT5 | POUT4 | POUT3 | POUT2 | POUT1 | POUT0 |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FD3H, FD7H, FDBH, FDFH, FE3H, FE7H, FEBH, FEFH | | | | | | | |

### Hex Address: FE8

**Table 253. Port A–H GPIO Address Registers (P*x*ADDR)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | PADDR[7:0] | | | | | | | |
| RESET | 00H | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FD0H, FD4H, FD8H, FDCH, FE0H, FE4H, FE8H, FECH | | | | | | | |

### Hex Address: FE9

**Table 254. Port A–H Control Registers (P*x*CTL)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | PCTL | | | | | | | |
| RESET | 00H | | | | | | | |
| R/W | R/W | | | | | | | |
| Address | FD1H, FD5H, FD9H, FDDH, FE1H, FE5H, FE9H, FEDH | | | | | | | |

### Hex Address: FEA

**Table 255. Port A–H Input Data Registers (PxIN)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | PIN7 | PIN6 | PIN5 | PIN4 | PIN3 | PIN2 | PIN1 | PIN0 |
| RESET | X | | | | | | | |
| R/W | R | | | | | | | |
| Address | FD2H, FD6H, FDAH, FDEH, FE2H, FE6H, FEAH, FEEH | | | | | | | |

## N



## O



## P