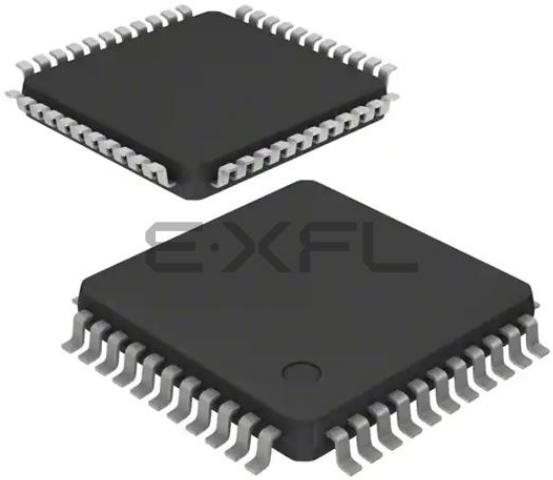

**Welcome to E-XFL.COM**

## 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 | 31 |
| Program Memory Size | 24KB (24K x 8) |
| Program Memory Type | FLASH |
| EEPROM Size | - |
| RAM Size | 2K x 8 |
| Voltage - Supply (Vcc/Vdd) | 3V ~ 3.6V |
| Data Converters | A/D 8x10b |
| Oscillator Type | Internal |
| Operating Temperature | -40°C ~ 105°C (TA) |
| Mounting Type | Surface Mount |
| Package / Case | 44-LQFP |
| Supplier Device Package | 44-LQFP (10x10) |
| Purchase URL | https://www.e-xfl.com/product-detail/zilog/z8f2421an020ec00tr |

## Signal Descriptions

Table 3 describes the Z8 Encore! XP signals. To determine the signals available for the specific package styles, see Pin Configurations on page 8.

### Table 3. Signal Descriptions

| Signal Mnemonic | I/O | Description |
|---|---|---|
| **General-Purpose I/O Ports A-H** | | |
| PA[7:0] | I/O | Port A[7:0]. These pins are used for general-purpose I/O and support 5 V-tolerant inputs. |
| PB[7:0] | I/O | Port B[7:0]. These pins are used for general-purpose I/O. |
| PC[7:0] | I/O | Port C[7:0]. These pins are used for general-purpose I/O. These pins are used for general-purpose I/O and support 5 V-tolerant inputs |
| PD[7:0] | I/O | Port D[7:0]. These pins are used for general-purpose I/O. These pins are used for general-purpose I/O and support 5 V-tolerant inputs |
| PE[7:0] | I/O | Port E[7:0]. These pins are used for general-purpose I/O. These pins are used for general-purpose I/O and support 5 V-tolerant inputs. |
| PF[7:0] | I/O | Port F[7:0]. These pins are used for general-purpose I/O. These pins are used for general-purpose I/O and support 5 V-tolerant inputs. |
| PG[7:0] | I/O | Port G[7:0]. These pins are used for general-purpose I/O. These pins are used for general-purpose I/O and support 5 V-tolerant inputs. |
| PH[3:0] | I/O | Port H[3:0]. These pins are used for general-purpose I/O. |
| **$I^2C$ Controller** | | |
| SCL | O | Serial Clock. This is the output clock for the $I^2C$. This pin is multiplexed with a general-purpose I/O pin. When the general-purpose I/O pin is configured for alternate function to enable the SCL function, this pin is open-drain. |
| SDA | I/O | Serial Data. This open-drain pin transfers data between the $I^2C$ and a slave. This pin is multiplexed with a general-purpose I/O pin. When the general-purpose I/O pin is configured for alternate function to enable the SDA function, this pin is open-drain. |
| **SPI Controller** | | |
| $\overline{SS}$ | I/O | Slave Select. This signal can be an output or an input. If the Z8 Encore! XP 64K Series Flash Microcontrollers is the SPI master, this pin may be configured as the Slave Select output. If the Z8 Encore! XP 64K Series Flash Microcontrollers is the SPI slave, this pin is the input slave select. It is multiplexed with a general-purpose I/O pin. |

**Table 7. Z8 Encore! XP 64K Series Flash Microcontrollers Register File Address Map (Continued)**

| Address (Hex) | Register Description | Mnemonic | Reset (Hex) | Page No |
|---|---|---|---|---|
| FEF | Port H Output Data | PHOUT | 00 | 66 |
| **Watchdog Timer** | | | | |
| FF0 | Watchdog Timer Control | WDTCTL | XXX00000b | 100 |
| FF1 | Watchdog Timer Reload Upper Byte | WDTU | FF | 101 |
| FF2 | Watchdog Timer Reload High Byte | WDTH | FF | 101 |
| FF3 | Watchdog Timer Reload Low Byte | WDTL | FF | 101 |
| FF4-FF7 | Reserved | — | XX | |
| **Flash Memory Controller** | | | | |
| FF8 | Flash Control | FCTL | 00 | 190 |
| FF8 | Flash Status | FSTAT | 00 | 190 |
| FF9 | Page Select | FPS | 00 | 191 |
| FF9 (if enabled) | Flash Sector Protect | FPROT | 00 | 192 |
| FFA | Flash Programming Frequency High Byte | FFREQH | 00 | 192 |
| FFB | Flash Programming Frequency Low Byte | FFREQL | 00 | 192 |
| FF4-FF8 | Reserved | — | XX | |
| **Read-Only Memory Controller** | | | | |
| FF9 | Page Select | RPS | 00 | |
| FFA-FFB | Reserved | — | XX | |
| **eZ8 CPU** | | | | |
| FFC | Flags | — | XX | Refer to *eZ8™ CPU Core User Manual (UM0128)* |
| FFD | Register Pointer | RP | XX | |
| FFE | Stack Pointer High Byte | SPH | XX | |
| FFF | Stack Pointer Low Byte | SPL | XX | |

**Note:** XX=Undefined

**UART0 Transmit Data**
U0TXD (F40H - Write Only)

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|

UART0 transmitter data byte [7:0]

**UART0 Receive Data**
U0RXD (F40H - Read Only)

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|

UART0 receiver data byte [7:0]

**UART0 Status 0**
U0STAT0 (F41H - Read Only)

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|

$\overline{\text{CTS}}$ signal
Returns the level of the $\overline{\text{CTS}}$ signal

Transmitter Empty
0 = Data is currently transmitting
1 = Transmission is complete

Transmitter Data Register Empty
0 = Transmit Data Register is full
1 = Transmit Data register is empty

Break Detect
0 = No break occurred
1 = A break occurred

Framing Error
0 = No framing error occurred
1 = A framing occurred

Overrun Error
0 = No overrun error occurred
1 = An overrun error occurred

Parity Error
0 = No parity error occurred
1 = A parity error occurred

Receive Data Available
0 = Receive Data Register is empty
1 = A byte is available in the Receive Data Register

**UART0 Control 0**
U0CTL0 (F42H - Read/Write)

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|

Loop Back Enable
0 = Normal operation
1 = Transmit data is looped back to the receiver

Stop Bit Select
0 = Transmitter sends 1 Stop bit
1 = Transmitter sends 2 Stop bits

Send Break
0 = No break is sent
1 = Output of the transmitter is zero

Parity Select
0 = Even parity
1 = Odd parity

Parity Enable
0 = Parity is disabled
1 = Parity is enabled

$\overline{\text{CTS}}$ Enable
0 = CTS signal has no effect on the transmitter
1 = UART recognizes $\overline{\text{CTS}}$ signal as a transmit enable control signal

Receive Enable
0 = Receiver disabled
1 = Receiver enabled

Transmit Enable
0 = Transmitter disabled
1 = Transmitter enabled

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

**Poor coding style that can result in lost interrupt requests:**

```
LDX r0, IRQ0
OR r0, MASK
LDX IRQ0, r0
```

*To avoid missing interrupts, the following style of coding to set bits in the Interrupt Request registers is recommended:*

**Good coding style that avoids lost interrupt requests:**

```
ORX IRQ0, MASK
```

## Interrupt Control Register Definitions

For all interrupts other than the Watchdog Timer interrupt, the interrupt control registers enable individual interrupts, set interrupt priorities, and indicate interrupt requests.

### Interrupt Request 0 Register

The Interrupt Request 0 (IRQ0) register (Table 24) stores the interrupt requests for both vectored and polled interrupts. When a request is presented to the interrupt controller, the corresponding bit in the IRQ0 register becomes 1. If interrupts are globally enabled (vectored interrupts), the interrupt controller passes an interrupt request to the eZ8™ CPU. If interrupts are globally disabled (polled interrupts), the eZ8 CPU can read the Interrupt Request 0 register to determine if any interrupt requests are pending

**Table 24. Interrupt Request 0 Register (IRQ0)**

| BITS | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| FIELD | T2I | T1I | T0I | U0RXI | U0TXI | I2CI | SPII | ADCI |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| ADDR | FC0H | | | | | | | |

T2I—Timer 2 Interrupt Request
0 = No interrupt request is pending for Timer 2.
1 = An interrupt request from Timer 2 is awaiting service.

T1I—Timer 1 Interrupt Request
0 = No interrupt request is pending for Timer 1.
1 = An interrupt request from Timer 1 is awaiting service.

T0I—Timer 0 Interrupt Request
0 = No interrupt request is pending for Timer 0.
1 = An interrupt request from Timer 0 is awaiting service.

U0RXI—UART 0 Receiver Interrupt Request
0 = No interrupt request is pending for the UART 0 receiver.
1 = An interrupt request from the UART 0 receiver is awaiting service.

U0TXI—UART 0 Transmitter Interrupt Request
0 = No interrupt request is pending for the UART 0 transmitter.
1 = An interrupt request from the UART 0 transmitter is awaiting service.

$I^2CI$— $I^2C$ Interrupt Request
0 = No interrupt request is pending for the $I^2C$.
1 = An interrupt request from the $I^2C$ is awaiting service.

SPII—SPI Interrupt Request
0 = No interrupt request is pending for the SPI.
1 = An interrupt request from the SPI is awaiting service.

ADCI—ADC Interrupt Request
0 = No interrupt request is pending for the Analog-to-Digital Converter.
1 = An interrupt request from the Analog-to-Digital Converter is awaiting service.

## Interrupt Request 1 Register

The Interrupt Request 1 (IRQ1) register (Table 25) stores interrupt requests for both vectored and polled interrupts. When a request is presented to the interrupt controller, the corresponding bit in the IRQ1 register becomes 1. If interrupts are globally enabled (vectored interrupts), the interrupt controller passes an interrupt request to the eZ8 CPU. If interrupts are globally disabled (polled interrupts), the eZ8 CPU can read the Interrupt Request 1 register to determine if any interrupt requests are pending.

**Table 25. Interrupt Request 1 Register (IRQ1)**

| BITS | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| FIELD | PAD7I | PAD6I | PAD5I | PAD4I | PAD3I | PAD2I | PAD1I | PAD0I |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| ADDR | FC3H | | | | | | | |

**Table 29. IRQ0 Enable Low Bit Register (IRQ0ENL)**

| BITS | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| FIELD | T2ENL | T1ENL | T0ENL | U0RENL | U0TENL | I2CENL | SPIENL | ADCENL |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| ADDR | FC2H | | | | | | | |

T2ENL—Timer 2 Interrupt Request Enable Low Bit
T1ENL—Timer 1 Interrupt Request Enable Low Bit
T0ENL—Timer 0 Interrupt Request Enable Low Bit
U0RENL—UART 0 Receive Interrupt Request Enable Low Bit
U0TENL—UART 0 Transmit Interrupt Request Enable Low Bit
I2CENL—$I^2C$ Interrupt Request Enable Low Bit
SPIENL—SPI Interrupt Request Enable Low Bit
ADCENL—ADC Interrupt Request Enable Low Bit

## IRQ1 Enable High and Low Bit Registers

The IRQ1 Enable High and Low Bit registers (see Table 31 and Table 32 on page 76) form a priority encoded enabling for interrupts in the Interrupt Request 1 register. Priority is generated by setting bits in each register. Table 30 describes the priority control for IRQ1.

**Table 30. IRQ1 Enable and Priority Encoding**

| IRQ1ENH[*x*] | IRQ1ENL[*x*] | Priority | Description |
|---|---|---|---|
| 0 | 0 | Disabled | Disabled |
| 0 | 1 | Level 1 | Low |
| 1 | 0 | Level 2 | Nominal |
| 1 | 1 | Level 3 | High |

**Note:** where *x* indicates the register bits from 0 through 7.

# Timers

## Overview

The 64K Series products contain up to four 16-bit reloadable timers that can be used for timing, event counting, or generation of pulse width modulated signals. The timers' features include:

- 16-bit reload counter
- Programmable prescaler with prescale values from 1 to 128
- PWM output generation
- Capture and compare capability
- External input pin for timer input, clock gating, or capture signal. External input pin signal frequency is limited to a maximum of one-fourth the system clock frequency.
- Timer output pin
- Timer interrupt

In addition to the timers described in this chapter, the Baud Rate Generators for any unused UART, SPI, or $I^2C$ peripherals may also be used to provide basic timing functionality. For information on using the Baud Rate Generators as timers, see the respective serial communication peripheral. Timer 3 is unavailable in the 44-pin package devices.

## Architecture

Figure 12 displays the architecture of the timers.

8. Execute an EI instruction to enable interrupts.

The UART is now configured for interrupt-driven data transmission. Because the UART Transmit Data register is empty, an interrupt is generated immediately. When the UART Transmit interrupt is detected, the associated interrupt service routine performs the following:

1. Write the UART Control 1 register to select the outgoing address bit:
   – Set the MULTIPROCESSOR Bit Transmitter (MPBT) if sending an address byte, clear it if sending a data byte.
2. Write the data byte to the UART Transmit Data register. The transmitter automatically transfers the data to the Transmit Shift register and transmits the data.
3. Clear the UART Transmit interrupt bit in the applicable Interrupt Request register.
4. Execute the IRET instruction to return from the interrupt-service routine and wait for the Transmit Data register to again become empty.

## Receiving Data using the Polled Method

Follow the steps below to configure the UART for polled data reception:

1. Write to the UART Baud Rate High and Low Byte registers to set the desired baud rate.
2. Enable the UART pin functions by configuring the associated GPIO Port pins for alternate function operation.
3. Write to the UART Control 1 register to enable MULTIPROCESSOR mode functions, if desired.
4. Write to the UART Control 0 register to:
   – Set the receive enable bit (REN) to enable the UART for data reception.
   – Enable parity, if desired and if MULTIPROCESSOR mode is not enabled, and select either even or odd parity.
5. Check the RDA bit in the UART Status 0 register to determine if the Receive Data register contains a valid data byte (indicated by a 1). If RDA is set to 1 to indicate available data, continue to step 6. If the Receive Data register is empty (indicated by a 0), continue to monitor the RDA bit awaiting reception of the valid data.
6. Read data from the UART Receive Data register. If operating in MULTIPROCESSOR (9-bit) mode, further actions may be required depending on the MULTIPROCESSOR Mode bits MPMD[1:0].
7. Return to step 5 to receive additional data.

## Receiving Data using the Interrupt-Driven Method

The UART Receiver interrupt indicates the availability of new data (as well as error conditions). Follow the steps below to configure the UART receiver for interrupt-driven operation:

1. Write to the UART Baud Rate High and Low Byte registers to set the desired baud rate.
2. Enable the UART pin functions by configuring the associated GPIO Port pins for alternate function operation.
3. Execute a `DI` instruction to disable interrupts.
4. Write to the Interrupt control registers to enable the UART Receiver interrupt and set the desired priority.
5. Clear the UART Receiver interrupt in the applicable Interrupt Request register.
6. Write to the UART Control 1 Register to enable MULTIPROCESSOR (9-bit) mode functions, if desired.
   - Set the MULTIPROCESSOR Mode Select (`MPEN`) to Enable MULTIPROCESSOR mode.
   - Set the MULTIPROCESSOR Mode Bits, `MPMD[1:0],` to select the desired address matching scheme.
   - Configure the UART to interrupt on received data and errors or errors only (interrupt on errors only is unlikely to be useful for Z8 Encore! devices without a DMA block).
7. Write the device address to the Address Compare Register (automatic multiprocessor modes only).
8. Write to the UART Control 0 register to:
   - Set the receive enable bit (`REN`) to enable the UART for data reception.
   - Enable parity, if desired and if MULTIPROCESSOR mode is not enabled, and select either even or odd parity.
9. Execute an `EI` instruction to enable interrupts.

The UART is now configured for interrupt-driven data reception. When the UART Receiver interrupt is detected, the associated interrupt service routine performs the following:

1. Check the UART Status 0 register to determine the source of the interrupt - error, break, or received data.
2. If the interrupt was caused by data available, read the data from the UART Receive Data register. If operating in MULTIPROCESSOR (9-bit) mode, further actions may be required depending on the MULTIPROCESSOR Mode bits `MPMD[1:0]`.

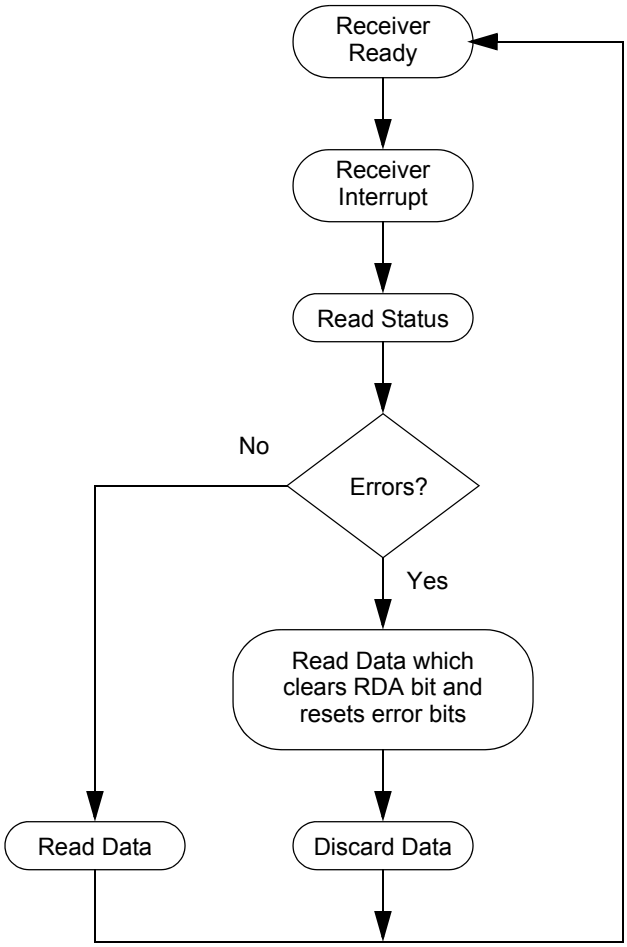


**Figure 18. UART Receiver Interrupt Service Routine Flow**

### Baud Rate Generator Interrupts

If the Baud Rate Generator interrupt enable is set, the UART Receiver interrupt asserts when the UART Baud Rate Generator reloads. This action allows the Baud Rate Generator to function as an additional counter if the UART functionality is not employed.

## UART Baud Rate Generator

The 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 UART Baud Rate High and Low Byte registers combine to create a 16-bit baud rate divisor value

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

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

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

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

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

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

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

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

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

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

- Master receives from a 7-bit slave
- Master receives from a 10-bit slave

## SDA and SCL Signals

I²C sends all addresses, data and acknowledge signals over the SDA line, most-significant bit first. SCL is the common clock for the I²C Controller. When the SDA and SCL pin alternate functions are selected for their respective GPIO ports, the pins are automatically configured for open-drain operation.

The master (I²C) is responsible for driving the SCL clock signal, although the clock signal can become skewed by a slow slave device. During the low period of the clock, the slave pulls the SCL signal Low to suspend the transaction. The master releases the clock at the end of the low period and notices that the clock remains low instead of returning to a high level. When the slave releases the clock, the I²C Controller continues the transaction. All data is transferred in bytes and there is no limit to the amount of data transferred in one operation. When transmitting data or acknowledging read data from the slave, the SDA signal changes in the middle of the low period of SCL and is sampled in the middle of the high period of SCL.

## I²C Interrupts

The I²C Controller contains four sources of interrupts—Transmit, Receive, Not Acknowledge and baud rate generator. These four interrupt sources are combined into a single interrupt request signal to the Interrupt Controller. The Transmit interrupt is enabled by the IEN and TXI bits of the Control register. The Receive and Not Acknowledge interrupts are enabled by the IEN bit of the Control register. The baud rate generator interrupt is enabled by the BIRQ and IEN bits of the Control register.

Not Acknowledge interrupts occur when a Not Acknowledge condition is received from the slave or sent by the I²C Controller and neither the START or STOP bit is set. The Not Acknowledge event sets the NCKI bit of the I²C Status register and can only be cleared by setting the START or STOP bit in the I²C Control register. When this interrupt occurs, the I²C Controller waits until either the STOP or START bit is set before performing any action. In an interrupt service routine, the NCKI bit should always be checked prior to servicing transmit or receive interrupt conditions because it indicates the transaction is being terminated.

Receive interrupts occur when a byte of data has been received by the I²C Controller (master reading data from slave). This procedure sets the RDRF bit of the I²C Status register. The RDRF bit is cleared by reading the I²C Data register. The RDRF bit is set during the acknowledge phase. The I²C Controller pauses after the acknowledge phase until the receive interrupt is cleared before performing any other action.

**Table 70. I²C Data Register (I2CDATA)**

| BITS | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| FIELD | DATA | | | | | | | |
| RESET | 0 | | | | | | | |
| R/W | R/W | | | | | | | |
| ADDR | F50H | | | | | | | |

## I²C Status Register

The Read-only I²C Status register (Table 71) indicates the status of the I²C Controller.

**Table 71. I²C Status Register (I2CSTAT)**

| BITS | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| FIELD | TDRE | RDRF | ACK | 10B | RD | TAS | DSS | NCKI |
| RESET | 1 | 0 | | | | | | |
| R/W | R | | | | | | | |
| ADDR | F51H | | | | | | | |

TDRE—Transmit Data Register Empty
When the I²C Controller is enabled, this bit is 1 when the I²C Data register is empty. When this bit is set, an interrupt is generated if the TXI bit is set, except when the I²C Controller is shifting in data during the reception of a byte or when shifting an address and the RD bit is set. This bit is cleared by writing to the I2CDATA register.

RDRF—Receive Data Register Full
This bit is set = 1 when the I²C Controller is enabled and the I²C Controller has received a byte of data. When asserted, this bit causes the I²C Controller to generate an interrupt. This bit is cleared by reading the I²C Data register (unless the read is performed using execution of the On-Chip Debugger's Read Register command).

ACK—Acknowledge
This bit indicates the status of the Acknowledge for the last byte transmitted or received. When set, this bit indicates that an Acknowledge occurred for the last byte transmitted or received. This bit is cleared when IEN = 0 or when a Not Acknowledge occurred for the last byte transmitted or received. It is not reset at the beginning of each transaction and is not reset when this register is read.

**Table 90. Flash Memory Sector Addresses**

| | Flash Sector Address Ranges | | | | |
|---|---|---|---|---|---|
| **Sector Number** | **Z8F162x** | **Z8F242x** | **Z8F322x** | **Z8F482x** | **Z8F642x** |
| 0 | 0000H-07FFH | 0000H-0FFFH | 0000H-0FFFH | 0000H-1FFFH | 0000H-1FFFH |
| 1 | 0800H-0FFFH | 1000H-1FFFH | 1000H-1FFFH | 2000H-3FFFH | 2000H-3FFFH |
| 2 | 1000H-17FFH | 2000H-2FFFH | 2000H-2FFFH | 4000H-5FFFH | 4000H-5FFFH |
| 3 | 1800H-1FFFH | 3000H-3FFFH | 3000H-3FFFH | 6000H-7FFFH | 6000H-7FFFH |
| 4 | 2000H-27FFH | 4000H-4FFFH | 4000H-4FFFH | 8000H-9FFFH | 8000H-9FFFH |
| 5 | 2800H-2FFFH | 5000H-5FFFH | 5000H-5FFFH | A000H-BFFFH | A000H-BFFFH |
| 6 | 3000H-37FFH | N/A | 6000H-6FFFH | N/A | C000H-DFFFH |
| 7 | 3800H-3FFFH | N/A | 7000H-7FFFH | N/A | E000H-FFFFH |

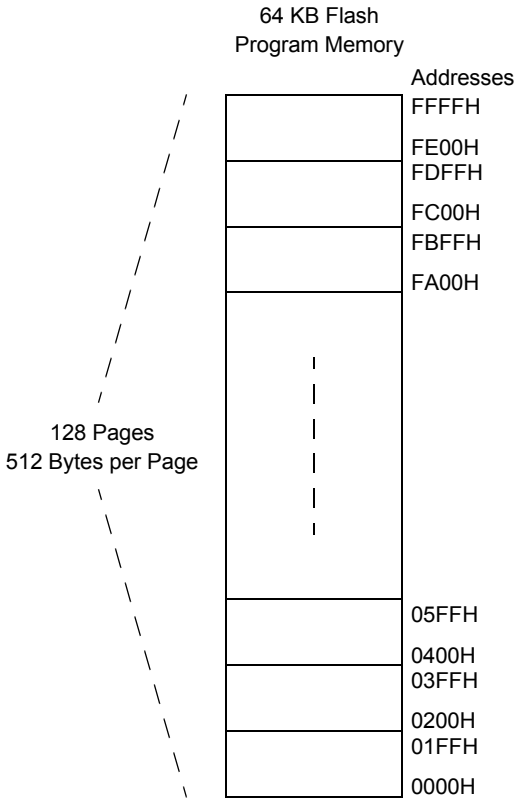


**Figure 35. Flash Memory Arrangement**

Figure 43 displays the typical active mode current consumption while operating at 25 ºC versus the system clock frequency. All GPIO pins are configured as outputs and driven High.

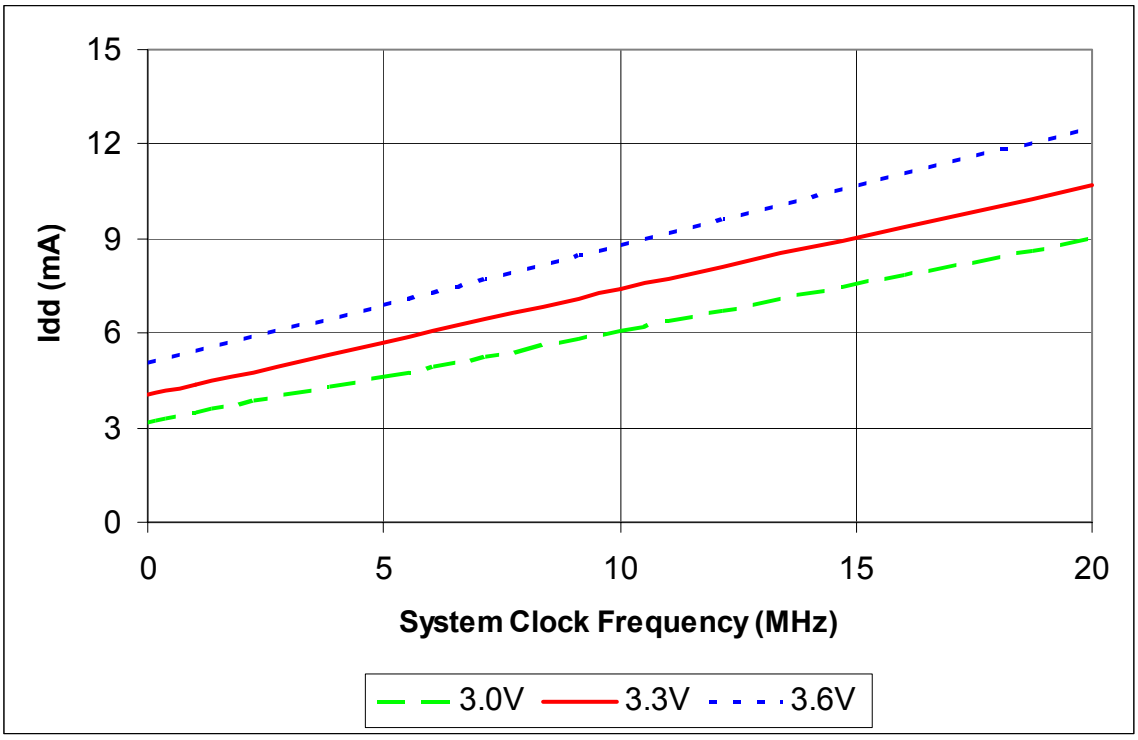


**Figure 43. Typical Active Mode Idd Versus System Clock Frequency**

Figure 45 displays the typical current consumption in HALT mode while operating at 25 ºC versus the system clock frequency. All GPIO pins are configured as outputs and driven High.

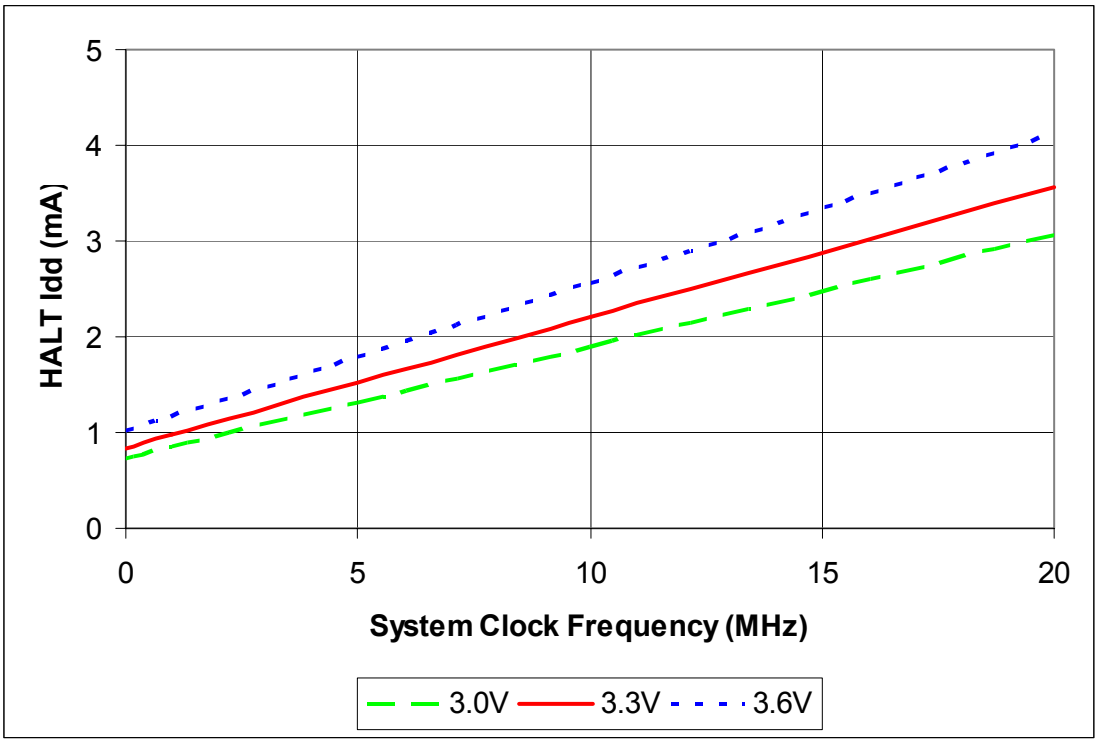


**Figure 45. Typical HALT Mode Idd Versus System Clock Frequency**

**Table 130. Logical Instructions (Continued)**

| Mnemonic | Operands | Instruction |
|---|---|---|
| XOR | dst, src | Logical Exclusive OR |
| XORX | dst, src | Logical Exclusive OR using Extended Addressing |

**Table 131. 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 132. Rotate and Shift Instructions**

| Mnemonic | Operands | Instruction |
|---|---|---|
| BSWAP | dst | Bit Swap |
| RL | dst | Rotate Left |
| RLC | dst | Rotate Left through Carry |
| RR | dst | Rotate Right |
| RRC | dst | Rotate Right through Carry |
| SRA | dst | Shift Right Arithmetic |

| Upper Nibble (Hex) \ Lower 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 | | | | | | 1,2 **ATM** |
| 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 | | 3.4 **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 60. First Opcode Map**