Zilog - Z8F2422AR020SC00TR 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	46
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 12x10b
Oscillator Type	Internal
Operating Temperature	0°C ~ 70°C (TA)
Mounting Type	Surface Mount
Package / Case	64-LQFP
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/zilog/z8f2422ar020sc00tr

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^{2}C Controller		
SCL	O	Serial Clock. This is the output clock for the I^{2}C. 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^{2}C 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		
$\overline{\text{SS}}$	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.

DMA_ADC Control

DMAACTL (FBEH - Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0

ADC Analog Input Number
0000 = Analog input 0 updated
0001 = Analog input 0-1 updated
0010 = Analog input 0-2 updated
0011 = Analog input 0-3 updated
0100 = Analog input 0-4 updated
0101 = Analog input 0-5 updated
0100 = Analog input 0-6 updated
0101 = Analog input 0-7 updated
1000 = Analog input 0-8 updated
1001 = Analog input 0-9 updated
1010 = Analog input 0-10 updated
1011 = Analog inputs 0-11 updated
11xx = Reserved

Reserved

Interrupt request enable
0 = DMA_ADC does not generate interrupt requests
1 = DMA_ADC generates interrupt requests after last analog input

DMA_ADC Enable
0 = DMA_ADC is disabled
1 = DMA_ADC is enabled

DMA Status

DMAA_STAT (FBFH - Read Only)

D7	D6	D5	D4	D3	D2	D1	D0

DMA0 Interrupt Request Indicator
0 = DMA0 is not the source of the IRQ
1 = DMA0 is the source of the IRQ

DMA1 Interrupt Request Indicator
0 = DMA1 is not the source of the IRQ
1 = DMA1 is the source of the IRQ

DMA_ADC Interrupt Request
0 = DMA_ADC is not the source of the IRQ
1 = DMA_ADC is the source of the IRQ

Reserved

Current ADC analog input
Identifies the analog input the ADC is currently converting

Interrupt Request 0

IRQ0 (FC0H - Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0

ADC Interrupt Request

SPI Interrupt Request

I2C Interrupt Request

UART 0 Transmitter Interrupt

UART 0 Receiver Interrupt Request

Timer 0 Interrupt Request

Timer 1 Interrupt Request

Timer 2 Interrupt Request

For all of the above peripherals:
0 = Peripheral IRQ is not pending
1 = Peripheral IRQ is awaiting service

IRQ0 Enable High Bit

IRQ0ENH (FC1H - Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0

ADC IRQ Enable Hit Bit

SPI IRQ Enable High Bit

I2C IRQ Enable High Bit

UART 0 Transmitter IRQ Enable

UART 0 Receiver IRQ Enable High

Timer 0 IRQ Enable High Bit

Timer 1 IRQ Enable High Bit

Timer 2 IRQ Enable High Bit

Table 9. Reset Sources and Resulting Reset Type

Operating Mode	Reset Source	Reset Type
NORMAL or HALT modes	Power-On Reset/Voltage Brownout	system reset
	Watchdog Timer time-out when configured for Reset	system reset
	$\overline{\text{RESET}}$ pin assertion	system reset
	On-Chip Debugger initiated Reset (OCDCTL[0] set to 1)	system reset except the On-Chip Debugger is unaffected by the reset
STOP mode	Power-On Reset/Voltage Brownout	system reset
	$\overline{\text{RESET}}$ pin assertion	system reset
	DBG pin driven Low	system reset

Power-On Reset

Each device in the 64K Series contains an internal Power-On Reset circuit. The POR circuit monitors the supply voltage and holds the device in the Reset state until the supply voltage reaches a safe operating level. After the supply voltage exceeds the POR voltage threshold (V_{POR}), the POR Counter is enabled and counts 66 cycles of the Watchdog Timer oscillator. After the POR counter times out, the XTAL Counter is enabled to count a total of 16 system clock pulses. The devices are held in the Reset state until both the POR Counter and XTAL counter have timed out. After the 64K Series devices exit the Power-On Reset state, the eZ8 CPU fetches the Reset vector. Following Power-On Reset, the POR status bit in the Watchdog Timer Control (WDTCTL) register is set to 1.

Figure 8 displays Power-On Reset operation. For the POR threshold voltage (V_{POR}), see Electrical Characteristics on page 215.

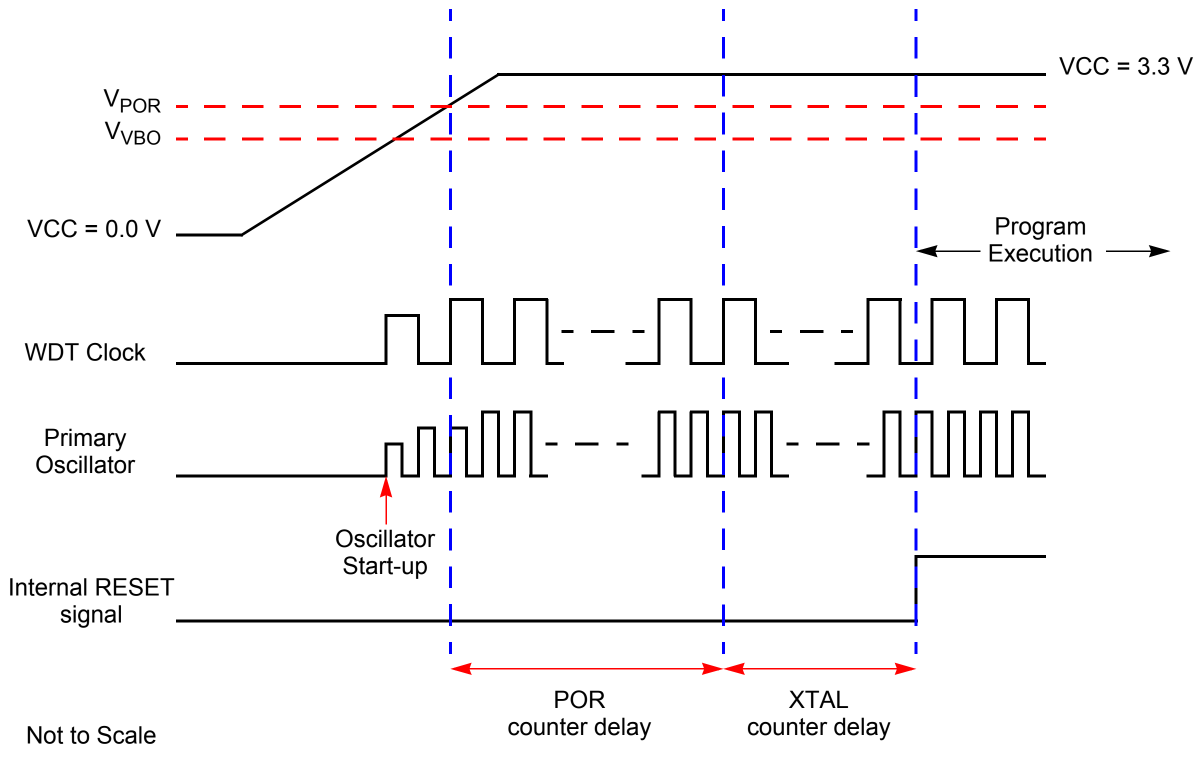

Figure 8. Power-On Reset Operation

Voltage Brownout Reset

The devices in the 64K Series provide low Voltage Brownout protection. The VBO circuit senses when the supply voltage drops to an unsafe level (below the VBO threshold voltage) and forces the device into the Reset state. While the supply voltage remains below the Power-On Reset voltage threshold (V_{POR}), the VBO block holds the device in the Reset state.

After the supply voltage again exceeds the Power-On Reset voltage threshold, the devices progress through a full system reset sequence, as described in the Power-On Reset section. Following Power-On Reset, the POR status bit in the Watchdog Timer Control (WDTCTL) register is set to 1. Figure 9 displays Voltage Brownout operation. For the VBO and POR threshold voltages (V_{VBO} and V_{POR}), see Electrical Characteristics on page 215.

The Voltage Brownout circuit can be either enabled or disabled during STOP mode. Operation during STOP mode is set by the VBO_AO Option Bit. For information on configuring VBO_AO, see Option Bits page 195.

Architecture

Figure 11 displays a block diagram of the interrupt controller.

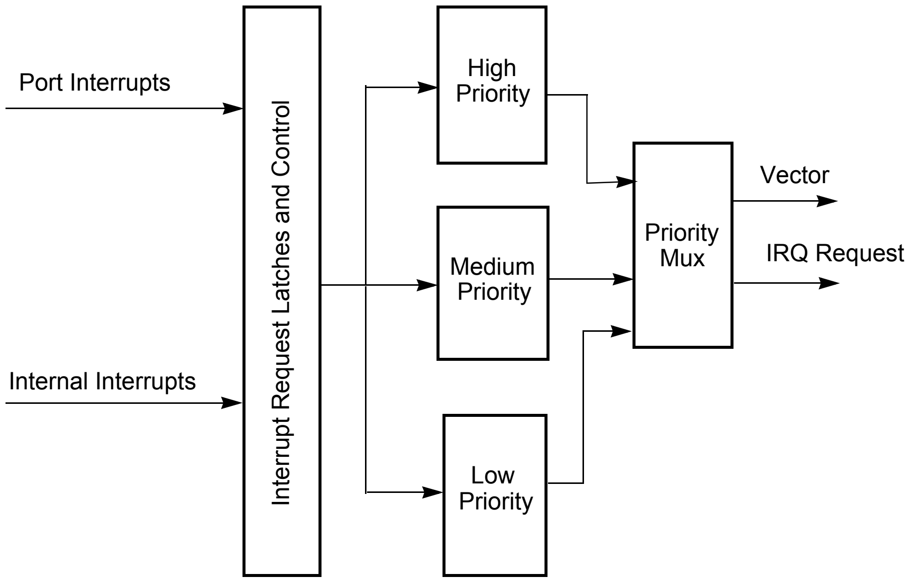

Figure 11. Interrupt Controller Block Diagram

Operation

Master Interrupt Enable

The master interrupt enable bit (IRQE) in the Interrupt Control register globally enables and disables interrupts.

Interrupts are globally enabled by any of the following actions:

- Executing an Enable Interrupt (EI) instruction.
- Executing an Return from Interrupt (IRET) instruction.
- Writing a 1 to the `IRQE` bit in the Interrupt Control register.

Interrupts are globally disabled by any of the following actions:

- Execution of a Disable Interrupt (DI) instruction.
- eZ8 CPU acknowledgement of an interrupt service request from the interrupt controller.
- Writing a 0 to the `IRQE` bit in the Interrupt Control register.
- Reset.

PAD*x*S—PA*x*/PD*x* Selection
0 = PA*x* is used for the interrupt for PA*x*/PD*x* interrupt request.
1 = PD*x* is used for the interrupt for PA*x*/PD*x* interrupt request.
where *x* indicates the specific GPIO Port pin number (0 through 7).

Interrupt Control Register

The Interrupt Control (IRQCTL) register (Table 38) contains the master enable bit for all interrupts.

Table 38. Interrupt Control Register (IRQCTL)

BITS	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FIELD	IRQE	Reserved						
RESET	0							
R/W	R/W	R						
ADDR	FCFH							

IRQE—Interrupt Request Enable
This bit is set to 1 by execution of an EI or IRET instruction, or by a direct register write of a 1 to this bit. It is reset to 0 by executing a DI instruction, eZ8 CPU acknowledgement of an interrupt request, or Reset.
0 = Interrupts are disabled
1 = Interrupts are enabled

Reserved—Must be 0.

Watchdog Timer

Overview

The Watchdog Timer (WDT) helps protect against corrupt or unreliable software, power faults, and other system-level problems which may place the Z8 Encore! XP into unsuitable operating states. The features of Watchdog Timer include:

- On-chip RC oscillator.
- A selectable time-out response.
- WDT Time-out response: Reset or interrupt.
- 24-bit programmable time-out value.

Operation

The Watchdog Timer (WDT) is a retriggerable one-shot timer that resets or interrupts the 64K Series devices when the WDT reaches its terminal count. The Watchdog Timer uses its own dedicated on-chip RC oscillator as its clock source. The Watchdog Timer has only two modes of operation—ON and OFF. Once enabled, it always counts and must be refreshed to prevent a time-out. An enable can be performed by executing the WDT instruction or by setting the `WDT_AO` Option Bit. The `WDT_AO` bit enables the Watchdog Timer to operate all the time, even if a WDT instruction has not been executed.

The Watchdog Timer is a 24-bit reloadable downcounter that uses three 8-bit registers in the eZ8™ CPU register space to set the reload value. The nominal WDT time-out period is given by the following equation:

$$\text{WDT Time-out Period (ms)} = \frac{\text{WDT Reload Value}}{10}$$

where the WDT reload value is the decimal value of the 24-bit value given by {WDTU[7:0], WDTH[7:0], WDTL[7:0]} and the typical Watchdog Timer RC oscillator frequency is 10 kHz. The Watchdog Timer cannot be refreshed once it reaches `000002H`. The WDT Reload Value must not be set to values below `000004H`. Table 47 provides information on approximate time-out delays for the minimum and maximum WDT reload values.

If interrupts are enabled, following completion of the Stop Mode Recovery the eZ8 CPU responds to the interrupt request by fetching the Watchdog 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 Watchdog Timer forces the device into the Reset state. The `WDT` status bit in the Watchdog Timer Control register is set to 1. For more information on Reset, see Reset and Stop Mode Recovery on page 47.

WDT Reset in STOP Mode

If enabled in STOP mode and configured to generate a Reset when a time-out occurs and the device is in STOP mode, the Watchdog Timer initiates a Stop Mode Recovery. Both the `WDT` status bit and the `STOP` bit in the Watchdog Timer Control register are set to 1 following WDT time-out in STOP mode. Default operation is for the WDT and its RC oscillator to be enabled during STOP mode.

WDT RC Disable in STOP Mode

To minimize power consumption in STOP Mode, the WDT and its RC oscillator can be disabled in STOP mode. The following sequence configures the WDT to be disabled when the 64K Series devices enter STOP Mode following execution of a STOP instruction:

1. Write `55H` to the Watchdog Timer Control register (WDTCTL).
2. Write `AAH` to the Watchdog Timer Control register (WDTCTL).
3. Write `81H` to the Watchdog Timer Control register (WDTCTL) to configure the WDT and its oscillator to be disabled during STOP Mode. Alternatively, write `00H` to the Watchdog Timer Control register (WDTCTL) as the third step in this sequence to reconfigure the WDT and its oscillator to be enabled during STOP mode.

This sequence only affects WDT operation in STOP mode.

Watchdog Timer Reload Unlock Sequence

Writing the unlock sequence to the Watchdog Timer (WDTCTL) Control register address unlocks the three Watchdog 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. Follow the steps below to unlock the Watchdog Timer Reload Byte registers (WDTU, WDTH, and WDTL) for write access.

1. Write `55H` to the Watchdog Timer Control register (WDTCTL).
2. Write `AAH` to the Watchdog Timer Control register (WDTCTL).
3. Write the Watchdog Timer Reload Upper Byte register (WDTU).
4. Write the Watchdog Timer Reload High Byte register (WDTH).

SPI Baud Rate High and Low Byte Registers

The SPI Baud Rate High and Low Byte registers (Table 68 and Table 69) 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:

$$\text{SPI BRG Interrupt Interval (s)} = \text{System Clock Period (s)} \times \text{BRG[15:0]}$$

Table 68. SPI Baud Rate High Byte Register (SPIBRH)

BITS	7	6	5	4	3	2	1	0
FIELD	BRH							
RESET	1							
R/W	R/W							
ADDR	F66H							

BRH = SPI Baud Rate High Byte
Most significant byte, BRG[15:8], of the SPI Baud Rate Generator’s reload value.

Table 69. SPI Baud Rate Low Byte Register (SPIBRL)

BITS	7	6	5	4	3	2	1	0
FIELD	BRL							
RESET	1							
R/W	R/W							
ADDR	F67H							

BRL = SPI Baud Rate Low Byte
Least significant byte, BRG[7:0], of the SPI Baud Rate Generator’s reload value.

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

SDA and SCL Signals

I^{2}C sends all addresses, data and acknowledge signals over the SDA line, most-significant bit first. SCL is the common clock for the I^{2}C 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 (I^{2}C) 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 I^{2}C 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.

I^{2}C Interrupts

The I^{2}C 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 I^{2}C Controller and neither the `START` or `STOP` bit is set. The Not Acknowledge event sets the NCKI bit of the I^{2}C Status register and can only be cleared by setting the `START` or `STOP` bit in the I^{2}C Control register. When this interrupt occurs, the I^{2}C 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 I^{2}C Controller (master reading data from slave). This procedure sets the RDRF bit of the I^{2}C Status register. The RDRF bit is cleared by reading the I^{2}C Data register. The RDRF bit is set during the acknowledge phase. The I^{2}C Controller pauses after the acknowledge phase until the receive interrupt is cleared before performing any other action.

reading the I^{2}C Data register. Once the I^{2}C data register has been read, the I^{2}C reads the next data byte.

Address Only Transaction with a 7-bit Address

In the situation where software determines if a slave with a 7-bit address is responding without sending or receiving data, a transaction can be done which only consists of an address phase. Figure 28 displays this ‘address only’ transaction to determine if a slave with a 7-bit address will acknowledge. As an example, this transaction can be used after a ‘write’ has been done to a EEPROM to determine when the EEPROM completes its internal write operation and is once again responding to I^{2}C transactions. If the slave does not Acknowledge, the transaction can be repeated until the slave does Acknowledge.

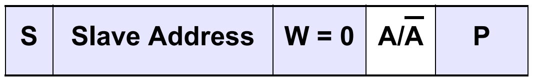

Figure 28. 7-Bit Address Only Transaction Format

Follow the steps below for an address only transaction to a 7-bit addressed slave:

1. Software asserts the `IEN` bit in the I^{2}C Control register.
2. Software asserts the `TXI` bit of the I^{2}C Control register to enable Transmit interrupts.
3. The I^{2}C interrupt asserts, because the I^{2}C Data register is empty (TDRE = 1)
4. Software responds to the `TDRE` bit by writing a 7-bit slave address plus write bit (=0) to the I^{2}C Data register. As an alternative this could be a read operation instead of a write operation.
5. Software sets the START and STOP bits of the I^{2}C Control register and clears the TXI bit.
6. The I^{2}C Controller sends the START condition to the I^{2}C slave.
7. The I^{2}C Controller loads the I^{2}C Shift register with the contents of the I^{2}C Data register.
8. Software polls the STOP bit of the I^{2}C Control register. Hardware deasserts the STOP bit when the address only transaction is completed.
9. Software checks the ACK bit of the I^{2}C Status register. If the slave acknowledged, the ACK bit is = 1. If the slave does not acknowledge, the ACK bit is = 0. The NCKI interrupt does not occur in the not acknowledge case because the STOP bit was set.

15. The I^{2}C Controller sends the repeated START condition.
16. The I^{2}C Controller loads the I^{2}C Shift register with the contents of the I^{2}C Data register (third address transfer).
17. The I^{2}C Controller sends `11110B` followed by the two most significant bits of the slave read address and a 1 (read).
18. The I^{2}C slave sends an acknowledge by pulling the SDA signal Low during the next high period of SCL

    If the slave were to Not Acknowledge at this point (this should not happen because the slave did acknowledge the first two address bytes), software would respond by setting the STOP and FLUSH bits and clearing the TXI bit. The I^{2}C Controller sends the STOP condition on the bus and clears the STOP and NCKI bits. The transaction is complete (ignore the following steps).
19. The I^{2}C Controller shifts in a byte of data from the I^{2}C slave on the SDA signal. The I^{2}C Controller sends a Not Acknowledge to the I^{2}C slave if the NAK bit is set (last byte), else it sends an Acknowledge.
20. The I^{2}C Controller asserts the Receive interrupt (RDRF bit set in the Status register).
21. Software responds by reading the I^{2}C Data register which clears the RDRF bit. If there is only one more byte to receive, set the NAK bit of the I^{2}C Control register.
22. If there are one or more bytes to transfer, return to step 19.
23. After the last byte is shifted in, a Not Acknowledge interrupt is generated by the I^{2}C Controller.
24. Software responds by setting the `STOP` bit of the I^{2}C Control register.
25. A STOP condition is sent to the I^{2}C slave and the STOP and NCKI bits are cleared.

I^{2}C Control Register Definitions

I^{2}C Data Register

The I^{2}C Data register (see Table 70 on page 157) holds the data that is to be loaded into the I^{2}C Shift register during a write to a slave. This register also holds data that is loaded from the I^{2}C Shift register during a read from a slave. The I^{2}C Shift Register is not accessible in the Register File address space, but is used only to buffer incoming and outgoing data.

IEN—I^{2}C Enable
1 = The I^{2}C transmitter and receiver are enabled.
0 = The I^{2}C transmitter and receiver are disabled.

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 if the `IEN` bit is deasserted. If this bit is 1, it cannot be cleared to 0 by writing to the register. 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 the Data register is written. 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 completes. If the `STOP` bit is also set, it also waits until the STOP condition is sent before the sending the START condition.

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.

BIRQ—Baud Rate Generator Interrupt Request
This bit allows the I^{2}C Controller to be used as an additional timer when the I^{2}C Controller is disabled. This bit is ignored when the I^{2}C Controller is enabled.
1 = An interrupt occurs every time the baud rate generator counts down to one.
0 = No baud rate generator interrupt occurs.

TXI—Enable TDRE interrupts
This bit enables the transmit interrupt when the I^{2}C Data register is empty (TDRE = 1).
1 = Transmit interrupt (and DMA transmit request) is enabled.
0 = Transmit interrupt (and DMA transmit request) is disabled.

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. If this bit is 1, it cannot be cleared to 0 by writing to the register.

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 a Not Acknowledge interrupt 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
This bit 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.
1 = low-pass filters are enabled.
0 = low-pass filters are disabled.

Flash Sector Protect Register

The Flash Sector Protect register (Table 95) 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).

Table 95. Flash Sector Protect Register (FPROT)

BITS	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FIELD	SECT7	SECT6	SECT5	SECT4	SECT3	SECT2	SECT1	SECT0
RESET	0							
R/W	R/W1							
ADDR	FF9H							
Note: R/W1 = Register is accessible for Read operations. Register can be written to 1 only (via user code).								

SECT*n*—Sector Protect
0 = Sector *n* can be programmed or erased from user code.
1 = Sector *n* is protected and cannot be programmed or erased from user code.

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

Flash Frequency High and Low Byte Registers

The Flash Frequency High and Low Byte registers (Table 96 and Table 97) combine to form a 16-bit value, FFREQ, to control timing for Flash program and erase operations. The 16-bit Flash Frequency registers must be written with the system clock frequency in kHz for Program and Erase operations. Calculate the Flash Frequency value using the following equation:

$$\text{FFREQ}[15:0] = \{\text{FFREQH}[7:0],\text{FFREQL}[7:0]\} = \frac{\text{System Clock Frequency}}{1000}$$

Caution: *Flash programming and erasure is not supported for system clock frequencies below 20 kHz, above 20 MHz, or outside of the valid operating frequency range for the device. The Flash Frequency High and Low Byte registers must be loaded with the correct value to insure proper program and erase times.*

finish the interrupt service routine it may be in and return the BRK instruction. When the CPU returns to the BRK instruction it was previously looping on, it automatically sets the DBGMODE bit and enter DEBUG mode.

Software detects that the majority of the OCD commands are still disabled when the eZ8™ CPU is looping on a BRK instruction. The eZ8 CPU must be stopped and the part must be in DEBUG mode before these commands can be issued.

Breakpoints in Flash Memory

The BRK instruction is opcode `00H`, which corresponds to the fully programmed state of a byte in Flash memory. To implement a Breakpoint, write `00H` to the desired address, overwriting the current instruction. To remove a Breakpoint, the corresponding page of Flash memory must be erased and reprogrammed with the original data.

On-Chip Debugger Commands

The host communicates to the On-Chip Debugger by sending OCD commands using the DBG interface. During normal operation, only a subset of the OCD commands are available. In DEBUG mode, all OCD commands become available unless the user code and control registers are protected by programming the Read Protect Option Bit (`RP`). The Read Protect Option Bit prevents the code in memory from being read out of the 64K Series products. When this option is enabled, several of the OCD commands are disabled. Table 101 contains a summary of the On-Chip Debugger commands. Each OCD command is described in detail in the bulleted list following Table 101.
Table 101 indicates those commands that operate when the device is not in DEBUG mode (normal operation) and those commands that are disabled by programming the Read Protect Option Bit.

Table 101. On-Chip Debugger Commands

Debug Command	Command Byte	Enabled when NOT in DEBUG mode?	Disabled by Read Protect Option Bit
Read OCD Revision	00H	Yes	-
Read OCD Status Register	02H	Yes	-
Read Runtime Counter	03H	-	-
Write OCD Control Register	04H	Yes	Cannot clear `DBGMODE` bit
Read OCD Control Register	05H	Yes	-

```
DBG ← Data Memory Address[7:0]
DBG ← Size[15:8]
DBG ← Size[7:0]
DBG → 1-65536 data bytes
```

- **Read Program Memory CRC (0EH)**—The Read Program Memory CRC command computes and returns the CRC (cyclic redundancy check) of Program Memory using the 16-bit CRC-CCITT polynomial. If the device is not in DEBUG mode, this command returns `FFFFH` for the CRC value. Unlike most other OCD Read commands, there is a delay from issuing of the command until the OCD returns the data. The OCD reads the Program Memory, calculates the CRC value, and returns the result. The delay is a function of the Program Memory size and is approximately equal to the system clock period multiplied by the number of bytes in the Program Memory.

```
DBG ← 0EH
DBG → CRC[15:8]
DBG → CRC[7:0]
```

- **Step Instruction (10H)**—The Step Instruction command steps one assembly instruction at the current Program Counter (PC) location. If the device is not in DEBUG mode or the Read Protect Option Bit is enabled, the OCD ignores this command.

```
DBG ← 10H
```

- **Stuff Instruction (11H)**—The Stuff Instruction command steps one assembly instruction and allows specification of the first byte of the instruction. The remaining 0-4 bytes of the instruction are read from Program Memory. This command is useful for stepping over instructions where the first byte of the instruction has been overwritten by a Breakpoint. If the device is not in DEBUG mode or the Read Protect Option Bit is enabled, the OCD ignores this command.

```
DBG ← 11H
DBG ← opcode[7:0]
```

- **Execute Instruction (12H)**—The Execute Instruction command allows sending an entire instruction to be executed to the eZ8 CPU. This command can also step over Breakpoints. The number of bytes to send for the instruction depends on the opcode. If the device is not in DEBUG mode or the Read Protect Option Bit is enabled, the OCD ignores this command

```
DBG ← 12H
DBG ← 1-5 byte opcode
```

Figure 44 displays the maximum active mode current consumption across the full operating temperature range of the device and versus the system clock frequency. All GPIO pins are configured as outputs and driven High.

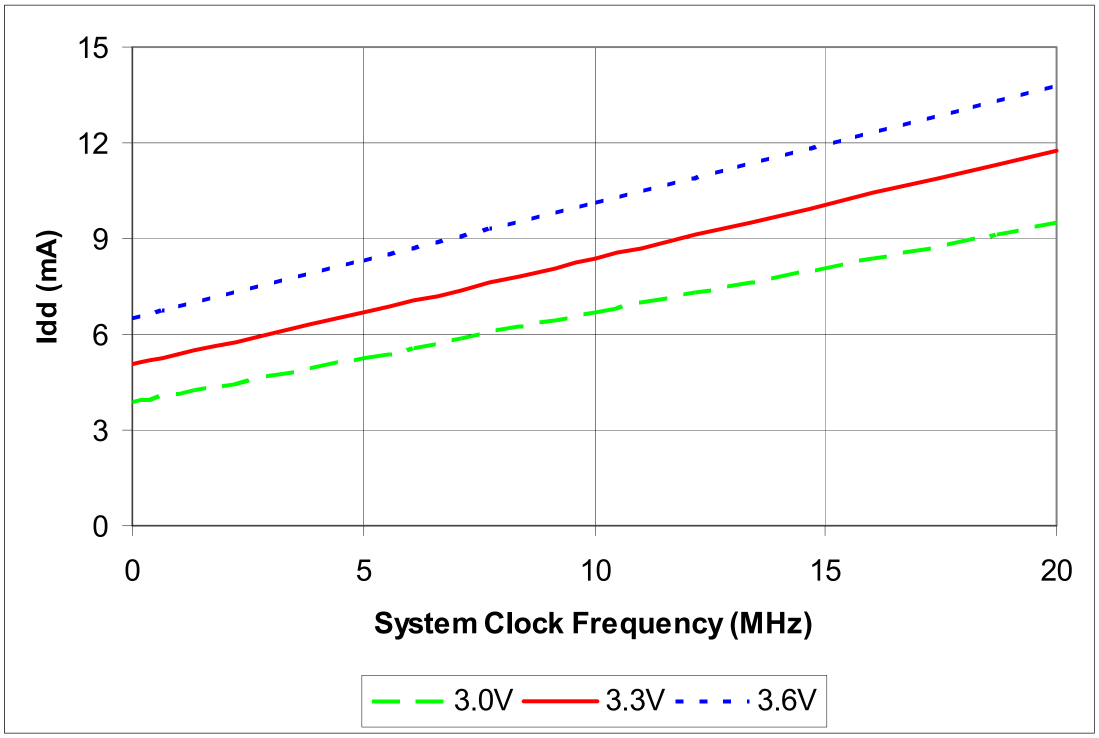

Figure 44. Maximum Active Mode Idd Versus System Clock Frequency

Table 123. Additional Symbols

Symbol	Definition
dst	Destination Operand
src	Source Operand
@	Indirect Address Prefix
SP	Stack Pointer
PC	Program Counter
FLAGS	Flags Register
RP	Register Pointer
#	Immediate Operand Prefix
B	Binary Number Suffix
%	Hexadecimal Number Prefix
H	Hexadecimal Number Suffix

Assignment of a value is indicated by an arrow. For example,

dst $\leftarrow$ dst + src

indicates the source data is added to the destination data and the result is stored in the destination location.

Condition Codes

The C, Z, S and V Flags control the operation of the conditional jump (JP cc and JR cc) instructions. Sixteen frequently useful functions of the Flag settings are encoded in a 4-bit field called the condition code (cc), which forms Bits 7:4 of the conditional jump instructions. The condition codes are summarized in Table 124. Some binary condition codes can be created using more than one assembly code mnemonic. The result of the Flag test operation decides if the conditional jump is executed.

Table 124. Condition Codes

Binary	Hex	Assembly Mnemonic	Definition	Flag Test Operation
0000	0	F	Always False	–
0001	1	LT	Less Than	(S XOR V) = 1
0010	2	LE	Less Than or Equal	(Z OR (S XOR V)) = 1

Opcode Maps

A description of the opcode map data and the abbreviations are provided in Figure 59 and Table 134 on page 262. Figure 60 on page 263 and Figure 61 on page 264 provide information on each of the eZ8™ CPU instructions.

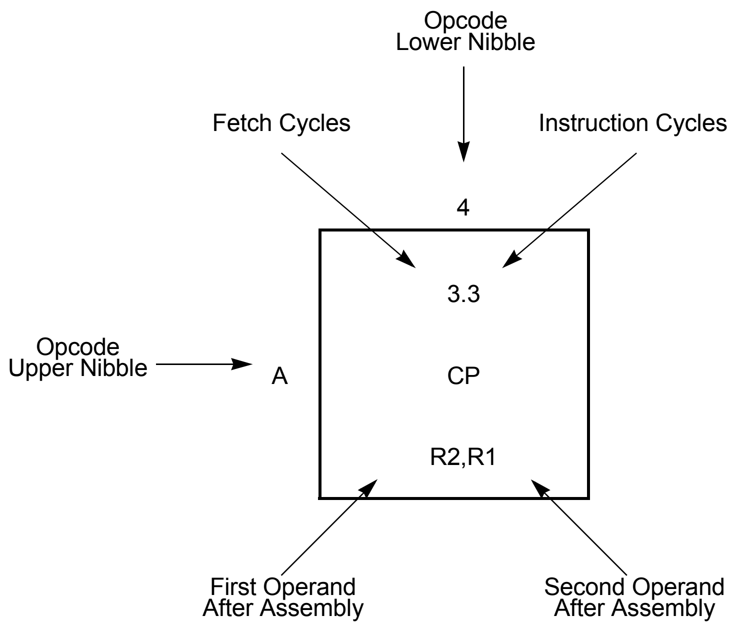

Figure 59. Opcode Map Cell Description