Zilog - Z8F4823FT020SC00TR Datasheet

Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Obsolete
Core Processor	eZ8
Core Size	8-Bit
Speed	20MHz
Connectivity	I²C, IrDA, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, DMA, POR, PWM, WDT
Number of I/O	60
Program Memory Size	48KB (48K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	4K x 8
Voltage - Supply (Vcc/Vdd)	3V ~ 3.6V
Data Converters	A/D 12x10b
Oscillator Type	Internal
Operating Temperature	0°C ~ 70°C (TA)
Mounting Type	Surface Mount
Package / Case	80-BQFP
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/zilog/z8f4823ft020sc00tr

Introduction

Zilog's Z8 Encore! XP MCU family of products are a line of Zilog® microcontroller products based upon the 8-bit eZ8 CPU. The Z8 Encore! XP® 64K Series Flash Microcontrollers, hereafter referred to collectively as the Z8 Encore! XP or the 64K Series adds Flash memory to Zilog's extensive line of 8-bit microcontrollers. The Flash in-circuit programming capability allows for faster development time and program changes in the field. The new eZ8™ CPU is upward compatible with existing Z8® instructions. The rich-peripheral set of the Z8 Encore! XP makes it suitable for a variety of applications including motor control, security systems, home appliances, personal electronic devices, and sensors.

Features

The features of Z8 Encore! XP 64K Series Flash Microcontrollers include:

- 20 MHz eZ8 CPU
- Up to 64 KB Flash with in-circuit programming capability
- Up to 4 KB register RAM
- 12-channel, 10-bit Analog-to-Digital Converter (ADC)
- Two full-duplex 9-bit UARTs with bus transceiver Driver Enable control
- Inter-integrated circuit (I^2C)
- Serial Peripheral Interface (SPI)
- Two Infrared Data Association (IrDA)-compliant infrared encoder/decoders
- Up to four 16-bit timers with capture, compare, and PWM capability
- Watchdog Timer (WDT) with internal RC oscillator
- Three-channel DMA
- Up to 60 input/output (I/O) pins
- 24 interrupts with configurable priority
- On-Chip Debugger
- Voltage Brownout (VBO) Protection
- Power-On Reset (POR)
- Operating voltage of 3.0 V to 3.6 V with 5 V-tolerant inputs
- 0 °C to +70 °C, –40 °C to +105 °C, and –40 °C to +125 °C operating temperature ranges

Table 6. Z8 Encore! XP 64K Series Flash Microcontrollers Information Area Map

Program Memory Address (Hex)	Function
FE00H-FE3FH	Reserved
FE40H-FE53H	Part Number 20-character ASCII alphanumeric code Left justified and filled with zeros (ASCII Null character)
FE54H-FFFFH	Reserved

Table 12. Port Alternate Function Mapping (Continued)

Port	Pin	Mnemonic	Alternate Function Description
Port C	PC0	T1IN	Timer 1 Input
	PC1	T1OUT	Timer 1 Output
	PC2	SS	SPI Slave Select
	PC3	SCK	SPI Serial Clock
	PC4	MOSI	SPI Master Out/Slave In
	PC5	MISO	SPI Master In/Slave Out
	PC6	T2IN	Timer 2 In
	PC7	T2OUT	Timer 2 Out
Port D	PD0	T3IN	Timer 3 In (unavailable in 44-pin packages)
	PD1	T3OUT	Timer 3 Out (unavailable in 44-pin packages)
	PD2	N/A	No alternate function
	PD3	DE1	UART 1 Driver Enable
	PD4	RXD1/IRRX1	UART 1/IrDA 1 Receive Data
	PD5	TXD1/IRTX1	UART 1/IrDA 1 Transmit Data
	PD6	CTS1	UART 1 Clear to Send
	PD7	RCOUT	Watchdog Timer RC Oscillator Output
Port E	PE[7:0]	N/A	No alternate functions
Port F	PF[7:0]	N/A	No alternate functions
Port G	PG[7:0]	N/A	No alternate functions
Port H	PH0	ANA8	ADC Analog Input 8
	PH1	ANA9	ADC Analog Input 9
	PH2	ANA10	ADC Analog Input 10
	PH3	ANA11	ADC Analog Input 11

GPIO Interrupts

Many of the GPIO port pins can be used as interrupt sources. Some port pins may be configured to generate an interrupt request on either the rising edge or falling edge of the pin input signal. Other port pin interrupts generate an interrupt when any edge occurs (both rising and falling). For more information on interrupts using the GPIO pins, see Interrupt Controller on page 67.

Table 19. Port A–H High Drive Enable Sub-Registers

BITS	7	6	5	4	3	2	1	0
FIELD	PHDE7	PHDE6	PHDE5	PHDE4	PHDE3	PHDE2	PHDE1	PHDE0
RESET	0							
R/W	R/W							
ADDR	If 04H in Port A-H Address Register, accessible through Port A-H Control Register							

PHDE[7:0]—Port High Drive Enabled
0 = The Port pin is configured for standard output current drive.
1 = The Port pin is configured for high output current drive.

Port A–H Stop Mode Recovery Source Enable Sub-Registers

The Port A–H Stop Mode Recovery Source Enable sub-register (Table 20) is accessed through the Port A–H Control register by writing `05H` to the Port A–H Address register. Setting the bits in the Port A–H Stop Mode Recovery Source Enable sub-registers to 1 configures the specified Port pins as a Stop Mode Recovery source. During STOP Mode, any logic transition on a Port pin enabled as a Stop Mode Recovery source initiates Stop Mode Recovery.

Table 20. Port A–H Stop Mode Recovery Source Enable Sub-Registers

BITS	7	6	5	4	3	2	1	0
FIELD	PSMRE7	PSMRE6	PSMRE5	PSMRE4	PSMRE3	PSMRE2	PSMRE1	PSMRE0
RESET	0							
R/W	R/W							
ADDR	If 05H in Port A–H Address Register, accessible through Port A–H Control Register							

PSMRE[7:0]—Port Stop Mode Recovery Source Enabled
0 = The Port pin is not configured as a Stop Mode Recovery source. Transitions on this pin during STOP mode do not initiate Stop Mode Recovery.

1 = The Port pin is configured as a Stop Mode Recovery source. Any logic transition on this pin during STOP mode initiates Stop Mode Recovery.

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.

POR—Power-On Reset Indicator
If this bit is set to 1, a Power-On Reset event occurred. This bit is reset to 0 if a WDT time-out or Stop Mode Recovery occurs. This bit is also reset to 0 when the register is read.

STOP—Stop Mode Recovery Indicator
If this bit is set to 1, a Stop Mode Recovery occurred. If the STOP and WDT bits are both set to 1, the Stop Mode Recovery occurred due to a WDT time-out. If the STOP bit is 1 and the WDT bit is 0, the Stop Mode Recovery was not caused by a WDT time-out. This bit is reset by a Power-On Reset or a WDT time-out that occurred while not in STOP mode. Reading this register also resets this bit.

WDT—Watchdog Timer Time-Out Indicator
If this bit is set to 1, a WDT time-out occurred. A Power-On Reset resets this pin. A Stop Mode Recovery from a change in an input pin also resets this bit. Reading this register resets this bit.

EXT—External Reset Indicator
If this bit is set to 1, a Reset initiated by the external $\overline{\text{RESET}}$ pin occurred. A Power-On Reset or a Stop Mode Recovery from a change in an input pin resets this bit. Reading this register resets this bit.

Reserved
These bits are reserved and must be 0.

SM—STOP Mode Configuration Indicator
0 = Watchdog Timer and its internal RC oscillator will continue to operate in STOP Mode.
1 = Watchdog Timer and its internal RC oscillator will be disabled in STOP Mode.

Watchdog Timer Reload Upper, High and Low Byte Registers

The Watchdog Timer Reload Upper, High and Low Byte (WDTU, WDTH, WDTL) registers (see Table 49 on page 102 through Table 51 on page 102) form the 24-bit reload value that is loaded into the Watchdog Timer when a WDT instruction executes. The 24-bit reload value is {WDTU[7:0], WDTH[7:0], WDTL[7:0]}. Writing to these registers sets the desired Reload Value. Reading from these registers returns the current Watchdog Timer count value.

3. Clear the UART Receiver interrupt in the applicable Interrupt Request register.
4. Execute the IRET instruction to return from the interrupt-service routine and await more data.

Clear To Send ($\overline{CTS}$) Operation

The CTS pin, if enabled by the CTSE bit of the UART Control 0 register, performs flow control on the outgoing transmit datastream. The Clear To Send ($\overline{CTS}$) input pin is sampled one system clock before beginning any new character transmission. To delay transmission of the next data character, an external receiver must deassert $\overline{CTS}$ at least one system clock cycle before a new data transmission begins. For multiple character transmissions, this would typically be done during Stop Bit transmission. If $\overline{CTS}$ deasserts in the middle of a character transmission, the current character is sent completely.

MULTIPROCESSOR (9-bit) Mode

The UART has a MULTIPROCESSOR (9-bit) mode that uses an extra (9th) bit for selective communication when a number of processors share a common UART bus. In MULTIPROCESSOR mode (also referred to as 9-Bit mode), the multiprocessor bit (MP) is transmitted immediately following the 8-bits of data and immediately preceding the Stop bit(s) as displayed in Figure 16. The character format is:

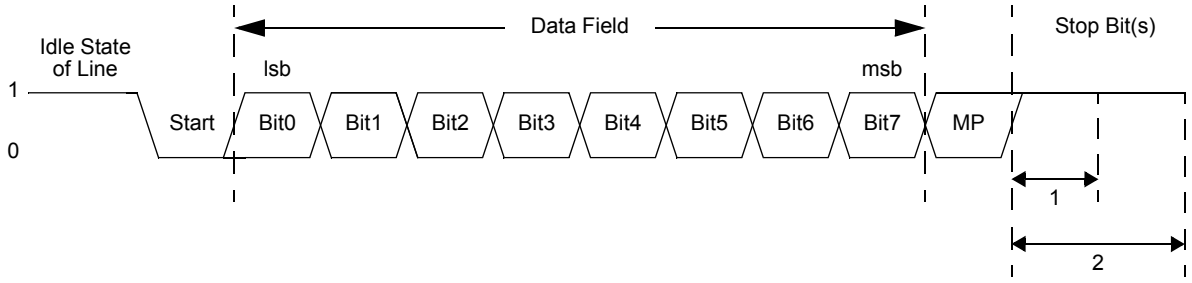

Figure 16. UART Asynchronous MULTIPROCESSOR Mode Data Format

In MULTIPROCESSOR (9-bit) mode, the Parity bit location (9th bit) becomes the MULTIPROCESSOR control bit. The UART Control 1 and Status 1 registers provide MULTIPROCESSOR (9-bit) mode control and status information. If an automatic address matching scheme is enabled, the UART Address Compare register holds the network address of the device.

MULTIPROCESSOR (9-bit) Mode Receive Interrupts

When MULTIPROCESSOR mode is enabled, the UART only processes frames addressed to it. The determination of whether a frame of data is addressed to the UART can be made in hardware, software or some combination of the two, depending on the multiprocessor

(BRG[15:0]) that sets the data transmission rate (baud rate) of the UART. The UART data rate is calculated using the following equation:

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

When the UART is disabled, the Baud Rate Generator can function as a basic 16-bit timer with interrupt on time-out. To configure the Baud Rate Generator as a timer with interrupt on time-out, complete the following procedure:

1. Disable the UART by clearing the REN and TEN bits in the UART Control 0 register to 0.
2. Load the desired 16-bit count value into the UART Baud Rate High and Low Byte registers.
3. Enable the Baud Rate Generator timer function and associated interrupt by setting the BRGCTL bit in the UART Control 1 register to 1.

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

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

UART Control Register Definitions

The UART control registers support the UART and the associated Infrared Encoder/Decoders. For more information on the infrared operation, see Infrared Encoder/Decoder on page 125.

UART Transmit Data Register

Data bytes written to the UART Transmit Data register (Table 52) are shifted out on the TXD*x* pin. The Write-only UART Transmit Data register shares a Register File address with the Read-only UART Receive Data register.

Table 52. UART Transmit Data Register (U*x*TXD)

BITS	7	6	5	4	3	2	1	0
FIELD	TXD							
RESET	X							
R/W	W							
ADDR	F40H and F48H							

SSIO—Slave Select I/O
0 = $\overline{SS}$ pin configured as an input.
1 = $\overline{SS}$ pin configured as an output (Master mode only).

SSV—Slave Select Value
If SSIO = 1 and SPI configured as a Master:
0 = $\overline{SS}$ pin driven Low (0).
1 = $\overline{SS}$ pin driven High (1).
This bit has no effect if SSIO = 0 or SPI configured as a Slave.

SPI Diagnostic State Register

The SPI Diagnostic State register (Table 67) provides observability of internal state. This is a read only register used for SPI diagnostics.

Table 67. SPI Diagnostic State Register (SPIDST)

BITS	7	6	5	4	3	2	1	0
FIELD	SCKEN	TCKEN	SPISTATE					
RESET	0							
R/W	R							
ADDR	F64H							

SCKEN—Shift Clock Enable
0 = The internal Shift Clock Enable signal is deasserted
1 = The internal Shift Clock Enable signal is asserted (shift register is updates on next system clock)

TCKEN—Transmit Clock Enable
0 = The internal Transmit Clock Enable signal is deasserted.
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).

SPISTATE—SPI State Machine
Defines the current state of the internal SPI State Machine.

Transmit interrupts occur when the TDRE bit of the I²C Status register sets and the TXI bit in the I²C Control register is set. Transmit interrupts occur under the following conditions when the transmit data register is empty:

- The I²C Controller is enabled.
- 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 until the Data register is written with the next value to send or the STOP or START bits are set indicating 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.*

16. If the I²C slave sends an acknowledge by pulling the SDA signal low during the next high period of SCL, the I²C Controller sets the ACK bit in the I²C Status register. Continue with step 17.

    If the slave does not acknowledge the second address byte or one of the data bytes, the I²C Controller sets the NCKI bit and clears the ACK bit in the I²C Status register. Software responds to the Not Acknowledge interrupt by setting the STOP and FLUSH bits and clearing the TXI bit. The I²C Controller sends the STOP condition on the bus and clears the STOP and NCKI bits. The transaction is complete (ignore the following steps).
17. The I²C Controller shifts the data out by the SDA signal. After the first bit is sent, the Transmit interrupt is asserted.
18. If more bytes remain to be sent, return to step 14.
19. If the last byte is currently being sent, software sets the STOP bit of the I²C Control register (or START bit to initiate a new transaction). In the STOP case, software also clears the TXI bit of the I²C Control register at the same time.
20. The I²C Controller completes transmission of the last data byte on the SDA signal.
21. The slave may either Acknowledge or Not Acknowledge the last byte. Because either the STOP or START bit is already set, the NCKI interrupt does not occur.
22. The I²C Controller sends the STOP (or RESTART) condition to the I²C bus and clears the STOP (or START) bit.

Read Transaction with a 7-Bit Address

Figure 32 displays the data transfer format for a read operation to a 7-bit addressed slave. The shaded regions indicate data transferred from the I²C Controller to slaves and unshaded regions indicate data transferred from the slaves to the I²C Controller.

S	Slave Address	R = 1	A	Data	A	Data	$\overline{A}$	P/S

Figure 32. Receive Data Transfer Format for a 7-Bit Addressed Slave

Follow the steps below for a read operation to a 7-bit addressed slave:

1. Software writes the I²C Data register with a 7-bit slave address plus the read bit (=1).
2. Software asserts the START bit of the I²C Control register.
3. If this is a single byte transfer, Software asserts the NAK bit of the I²C Control register so that after the first byte of data has been read by the I²C Controller, a Not Acknowledge is sent to the I²C slave.

0101 = ADC Analog Inputs 0-5 updated.
0110 = ADC Analog Inputs 0-6 updated.
0111 = ADC Analog Inputs 0-7 updated.
1000 = ADC Analog Inputs 0-8 updated.
1001 = ADC Analog Inputs 0-9 updated.
1010 = ADC Analog Inputs 0-10 updated.
1011 = ADC Analog Inputs 0-11 updated.
1100-1111 = Reserved.

DMA Status Register

The DMA Status register (Table 85 on page 173) indicates the DMA channel that generated the interrupt and the ADC Analog Input that is currently undergoing conversion. Reads from this register reset the Interrupt Request Indicator bits (IRQA, IRQ1, and IRQ0) to 0. Therefore, software interrupt service routines that read this register must process all three interrupt sources from the DMA.

Table 85. DMA_ADC Status Register (DMAA_STAT)

BITS	7	6	5	4	3	2	1	0
FIELD	CADC[3:0]				Reserved	IRQA	IRQ1	IRQ0
RESET	0							
R/W	R							
ADDR	FBFH							

CADC[3:0]—Current ADC Analog Input
This field identifies the Analog Input that the ADC is currently converting.

Reserved
This bit is reserved and must be 0.

IRQA—DMA_ADC Interrupt Request Indicator
This bit is automatically reset to 0 each time a read from this register occurs.
0 = DMA_ADC is not the source of the interrupt from the DMA Controller.
1 = DMA_ADC completed transfer of data from the last ADC Analog Input and generated an interrupt.

IRQ1—DMA1 Interrupt Request Indicator
This bit is automatically reset to 0 each time a read from this register occurs.
0 = DMA1 is not the source of the interrupt from the DMA Controller.
1 = DMA1 completed transfer of data to/from the End Address and generated an interrupt.

IRQ0—DMA0 Interrupt Request Indicator
This bit is automatically reset to 0 each time a read from this register occurs.

Timing Using the Flash Frequency Registers

Before performing a program or erase operation on the Flash memory, you must first configure the Flash Frequency High and Low Byte registers. The Flash Frequency registers allow programming and erasure of the Flash with system clock frequencies ranging from 20 kHz through 20 MHz (the valid range is limited to the device operating frequencies).

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

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

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

Flash Read Protection

The user code contained within the Flash memory can be protected from external access. Programming the Flash Read Protect Option Bit prevents reading of user code by the On-Chip Debugger or by using the Flash Controller Bypass mode. For more information, see Option Bits on page 195 and On-Chip Debugger on page 199.

Flash Write/Erase Protection

The 64K Series provides several levels of protection against accidental program and erasure of the Flash memory contents. This protection is provided by the Flash Controller unlock mechanism, the Flash Sector Protect register, and the Flash Write Protect option bit.

Flash Controller Unlock Mechanism

At Reset, the Flash Controller locks to prevent accidental program or erasure of the Flash memory. To program or erase the Flash memory, the Flash controller must be unlocked. After unlocking the Flash Controller, the Flash can be programmed or erased. Any value written by user code to the Flash Control register or Page Select Register out of sequence will lock the Flash Controller.

Follow the steps below to unlock the Flash Controller from user code:

1. Write 00H to the Flash Control register to reset the Flash Controller.
2. Write the page to be programmed or erased to the Page Select register.

On-Chip Debugger

Overview

The 64K Series products contain an integrated On-Chip Debugger (OCD) that provides advanced debugging features including:

- Reading and writing of the Register File
- Reading and writing of Program and Data Memory
- Setting of Breakpoints
- Execution of eZ8 CPU instructions

Architecture

The On-Chip Debugger consists of four primary functional blocks: transmitter, receiver, auto-baud generator, and debug controller. Figure 36 displays the architecture of the On-Chip Debugger.

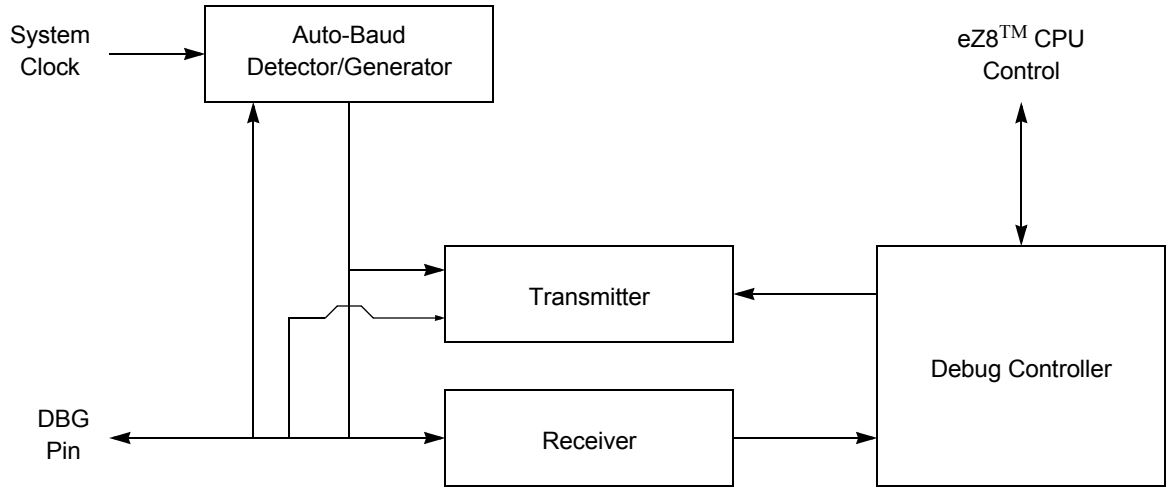

Figure 36. On-Chip Debugger Block Diagram

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	-

Electrical Characteristics

Absolute Maximum Ratings

Stresses greater than those listed in Table 105 may cause permanent damage to the device. These ratings are stress ratings only. Operation of the device at any condition outside those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability. For improved reliability, unused inputs must be tied to one of the supply voltages (V_{DD} or V_{SS}).

Table 105. Absolute Maximum Ratings

Parameter	Minimum	Maximum	Units	Notes
Ambient temperature under bias	-40	+125	C	
Storage temperature	-65	+150	C	
Voltage on any pin with respect to V_{SS}	-0.3	+5.5	V	1
Voltage on V_{DD} pin with respect to V_{SS}	-0.3	+3.6	V	
Maximum current on input and/or inactive output pin	-5	+5	µA	
Maximum output current from active output pin	-25	+25	mA	
80-Pin QFP Maximum Ratings at –40 °C to 70 °C				
Total power dissipation		550	mW	
Maximum current into V_{DD} or out of V_{SS}		150	mA	
80-Pin QFP Maximum Ratings at 70 °C to 125 °C				
Total power dissipation		200	mW	
Maximum current into V_{DD} or out of V_{SS}		56	mA	
68-Pin PLCC Maximum Ratings at –40 °C to 70 °C				
Total power dissipation		1000	mW	
Maximum current into V_{DD} or out of V_{SS}		275	mA	
68-Pin PLCC Maximum Ratings at 70 °C to 125 °C				
Total power dissipation		500	mW	

```
; value 01H, is the source. The value 01H is written into the
; Register at address 234H.
```

Assembly Language Syntax

For proper instruction execution, eZ8 CPU assembly language syntax requires that the operands be written as ‘destination, source’. After assembly, the object code usually has the operands in the order ‘source, destination’, but ordering is opcode-dependent. The following instruction examples illustrate the format of some basic assembly instructions and the resulting object code produced by the assembler. This binary format must be followed if you prefer manual program coding or intend to implement your own assembler.

Example 1: If the contents of Registers 43H and 08H are added and the result is stored in 43H, the assembly syntax and resulting object code is:

Assembly Language Syntax Example 1

Assembly Language Code	ADD	43H,	08H	(ADD dst, src)
Object Code	04	08	43	(OPC src, dst)

Example 2: In general, when an instruction format requires an 8-bit register address, that address can specify any register location in the range 0–255 or, using Escaped Mode Addressing, a Working Register R0 - R15. If the contents of Register 43H and Working Register R8 are added and the result is stored in 43H, the assembly syntax and resulting object code is:

Assembly Language Syntax Example 2

Assembly Language Code	ADD	43H,	R8	(ADD dst, src)
Object Code	04	E8	43	(OPC src, dst)

Refer to the device-specific Product Specification to determine the exact register file range available. The register file size varies, depending on the device type.

eZ8 CPU Instruction Notation

In the eZ8 CPU Instruction Summary and Description sections, the operands, condition codes, status Flags, and address modes are represented by a notational shorthand that is described in Table 122.

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.

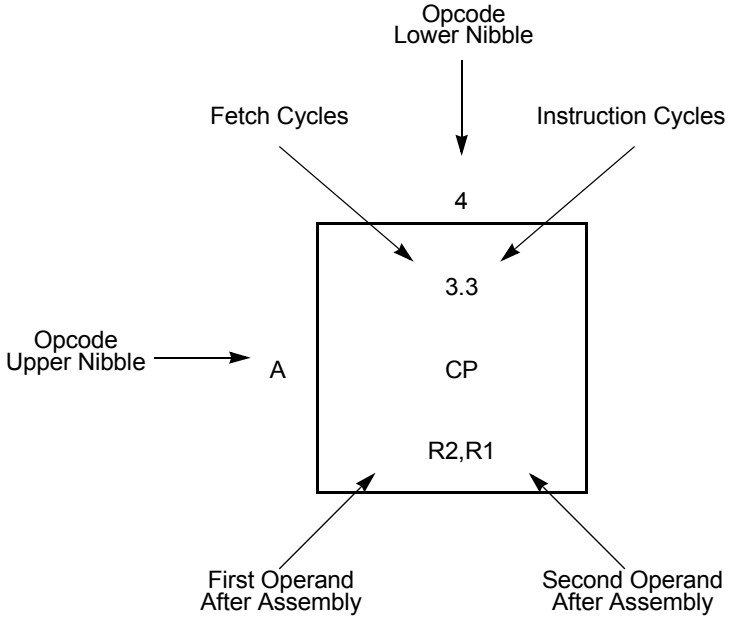

Figure 59. Opcode Map Cell Description