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	-40°C ~ 105°C (TA)
Mounting Type	Surface Mount
Package / Case	68-LCC (J-Lead)
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/zilog/z8f2402vs020ec

- Software stack allows much greater depth in subroutine calls and interrupts than hardware stacks
- Compatible with existing Z8 code
- Expanded internal Register File allows access of up to 4KB
- New instructions improve execution efficiency for code developed using higher-level programming languages, including C
- Pipelined instruction fetch and execution
- New instructions for improved performance including BIT, BSWAP, BTJ, CPC, LDC, LDCI, LEA, MULT, and SRL
- New instructions support 12-bit linear addressing of the Register File
- Up to 10 MIPS operation
- C-Compiler friendly
- 2-9 clock cycles per instruction

For more information regarding the eZ8 CPU, refer to the *eZ8 CPU User Manual* available for download at www.zilog.com.

General Purpose I/O

The Z8 Encore!® features seven 8-bit ports (Ports A-G) and one 4-bit port (Port H) for general purpose I/O (GPIO). Each pin is individually programmable.

Flash Controller

The Flash Controller programs and erases the Flash memory.

10-Bit Analog-to-Digital Converter

The Analog-to-Digital Converter (ADC) converts an analog input signal to a 10-bit binary number. The ADC accepts inputs from up to 12 different analog input sources.

UARTs

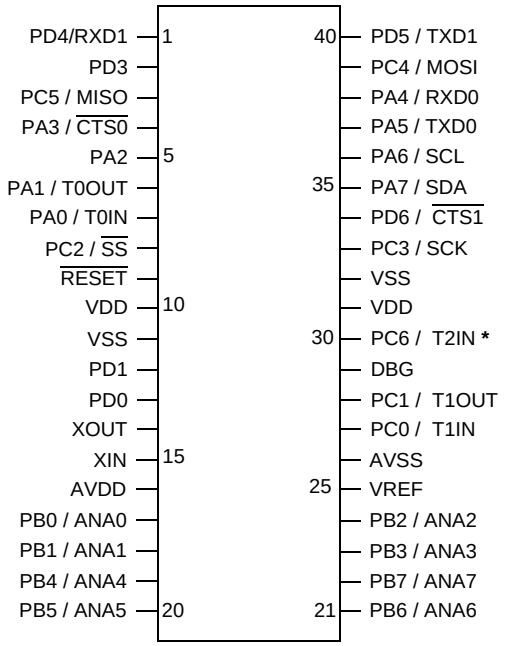
Each UART is full-duplex and capable of handling asynchronous data transfers. The UARTs support 8- and 9-bit data modes and selectable parity.

I^2C

The inter-integrated circuit (I^2C®) controller makes the Z8 Encore!® compatible with the I^2C protocol. The I^2C controller consists of two bidirectional bus lines, a serial data (SDA) line and a serial clock (SCL) line.

Pin Configurations

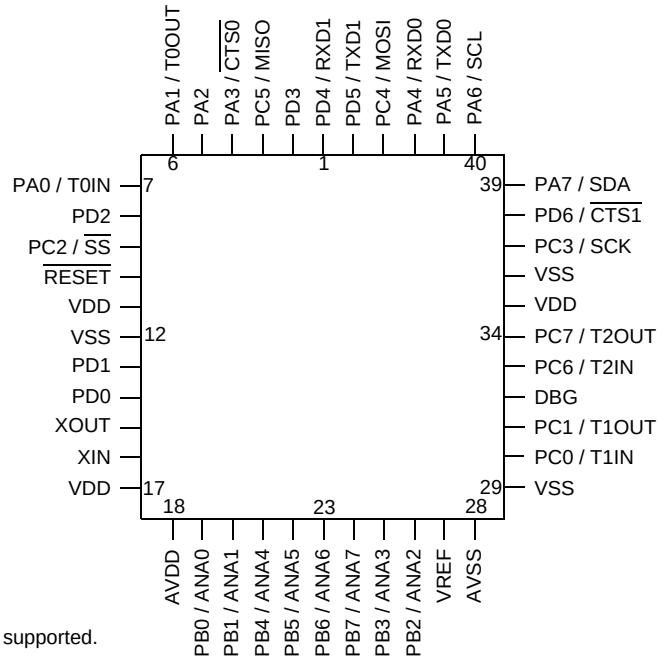
Figures 56 through 61 illustrate the pin configurations for all of the packages available in the Z8 Encore!® MCU family. Refer to Table 2 for a description of the signals.



Note: Timer 3 is not supported.

***** T2OUT is not supported.

Figure 56. Z8Fxx01 in 40-Pin Dual Inline Package (DIP)



Note: Timer 3 is not supported.

Figure 57. Z8Fxx01 in 44-Pin Plastic Leaded Chip Carrier (PLCC)

System and Short Resets

During a System Reset, the Z8F640x family device is held in Reset for 514 cycles of the Watch-Dog Timer oscillator followed by 16 cycles of the system clock (crystal oscillator). A Short Reset differs from a System Reset only in the number of Watch-Dog Timer oscillator cycles required to exit Reset. A Short Reset requires only 66 Watch-Dog Timer oscillator cycles. Unless specifically stated otherwise, System Reset and Short Reset are referred to collectively as Reset.

During Reset, the eZ8 CPU and on-chip peripherals are idle; however, the on-chip crystal oscillator and Watch-Dog Timer oscillator continue to run. The system clock begins operating following the Watch-Dog Timer oscillator cycle count. The eZ8 CPU and on-chip peripherals remain idle through the 16 cycles of the system clock.

Upon Reset, control registers within the Register File that have a defined Reset value are loaded with their reset values. Other control registers (including the Stack Pointer, Register Pointer, and Flags) and general-purpose RAM are undefined following Reset. The eZ8 CPU fetches the Reset vector at Program Memory addresses 0002H and 0003H and loads that value into the Program Counter. Program execution begins at the Reset vector address.

Reset Sources

Table 8 lists the reset sources and type of Reset as a function of the Z8F640x family device operating mode. The text following provides more detailed information on the individual Reset sources. Please note that Power-On Reset / Voltage Brown-Out events always have priority over all other possible reset sources to insure a full system reset occurs.

Table 8. Reset Sources and Resulting Reset Type

Operating Mode	Reset Source	Reset Type
Normal or Halt modes	Power-On Reset / Voltage Brown-Out	System Reset
	Watch-Dog Timer time-out when configured for Reset	Short Reset
	$\overline{\text{RESET}}$ pin assertion	Short Reset
	On-Chip Debugger initiated Reset (OCDCTL[1] set to 1)	System Reset except the On-Chip Debugger is unaffected by the reset
Stop mode	Power-On Reset / Voltage Brown-Out	System Reset
	$\overline{\text{RESET}}$ pin assertion	System Reset
	DBG pin driven Low	System Reset

General-Purpose I/O

Overview

The Z8F640x family products support a maximum of seven 8-bit ports (Ports A-G) and one 4-bit port (Port H) for general-purpose input/output (I/O) operations. Each port contains control and data registers. The GPIO control registers are used to determine data direction, open-drain, output drive current and alternate pin functions. Each port pin is individually programmable.

GPIO Port Availability By Device

Not all Z8F640x family products support all 8 ports (A-H). Table 10 lists the port pins available with each device and package type.

Table 10. Port Availability by Device and Package Type

Device	Packages	Port A	Port B	Port C	Port D	Port E	Port F	Port G	Port H
Z8F1601	40-pin	[7:0]	[7:0]	[6:0]	[6:3, 1:0]	-	-	-	-
Z8F1601	44-pin	[7:0]	[7:0]	[7:0]	[6:0]				
Z8F1602	64- and 68-pin	[7:0]	[7:0]	[7:0]	[7:0]	[7:0]	[7]	[3]	[3:0]
Z8F2401	40-pin	[7:0]	[7:0]	[6:0]	[6:3, 1:0]	-	-	-	-
Z8F2401	44-pin	[7:0]	[7:0]	[7:0]	[6:0]	-	-	-	-
Z8F2402	64- and 68-pin	[7:0]	[7:0]	[7:0]	[7:0]	[7:0]	[7]	[3]	[3:0]
Z8F3201	40-pin	[7:0]	[7:0]	[6:0]	[6:3, 1:0]	-	-	-	-
Z8F3201	44-pin	[7:0]	[7:0]	[7:0]	[6:0]	-	-	-	-
Z8F3202	64- and 68-pin	[7:0]	[7:0]	[7:0]	[7:0]	[7:0]	[7]	[3]	[3:0]
Z8F4801	40-pin	[7:0]	[7:0]	[6:0]	[6:3, 1:0]	-	-	-	-
Z8F4801	44-pin	[7:0]	[7:0]	[7:0]	[6:0]	-	-	-	-
Z8F4802	64- and 68-pin	[7:0]	[7:0]	[7:0]	[7:0]	[7:0]	[7]	[3]	[3:0]
Z8F4803	80-pin	[7:0]	[7:0]	[7:0]	[7:0]	[7:0]	[7:0]	[7:0]	[3:0]
Z8F6401	40-pin	[7:0]	[7:0]	[6:0]	[6:3, 1:0]	-	-	-	-

Table 22. Interrupt Vectors in Order of Priority

Priority	Program Memory Vector Address	Interrupt Source	Interrupt Assertion Type
Highest	0002h	Reset (not an interrupt)	Not applicable
	0004h	Watch-Dog Timer	Continuous assertion
	0006h	Illegal Instruction Trap (not an interrupt)	Not applicable
	0008h	Timer 2	Single assertion (pulse)
	000Ah	Timer 1	Single assertion (pulse)
	000Ch	Timer 0	Single assertion (pulse)
	000Eh	UART 0 receiver	Continuous assertion
	0010h	UART 0 transmitter	Continuous assertion
	0012h	I^2C	Continuous assertion
	0014h	SPI	Continuous assertion
	0016h	ADC	Single assertion (pulse)
	0018h	Port A7 or Port D7, rising or falling input edge	Single assertion (pulse)
	001Ah	Port A6 or Port D6, rising or falling input edge	Single assertion (pulse)
	001Ch	Port A5 or Port D5, rising or falling input edge	Single assertion (pulse)
	001Eh	Port A4 or Port D4, rising or falling input edge	Single assertion (pulse)
	0020h	Port A3 or Port D3, rising or falling input edge	Single assertion (pulse)
	0022h	Port A2 or Port D2, rising or falling input edge	Single assertion (pulse)
	0024h	Port A1 or Port D1, rising or falling input edge	Single assertion (pulse)
	0026h	Port A0 or Port D0, rising or falling input edge	Single assertion (pulse)
	0028h	Timer 3 *(not available in 40/44-pin packages)*	Single assertion (pulse)
	002Ah	UART 1 receiver	Continuous assertion
	002Ch	UART 1 transmitter	Continuous assertion
	002Eh	DMA	Single assertion (pulse)
	0030h	Port C3, both input edges	Single assertion (pulse)
	0032h	Port C2, both input edges	Single assertion (pulse)
	0034h	Port C1, both input edges	Single assertion (pulse)
Lowest	0036h	Port C0, both input edges	Single assertion (pulse)

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 24) 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 24. 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	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	FC3H							

PAD*x*I—Port A or Port D Pin *x* Interrupt Request
0 = No interrupt request is pending for GPIO Port A or Port D pin *x*.
1 = An interrupt request from GPIO Port A or Port D pin *x* is awaiting service.

where *x* indicates the specific GPIO Port pin number (0 through 7). For each pin, only 1 of either Port A or Port D can be enabled for interrupts at any one time. Port selection (A or D) is determined by the values in the Interrupt Port Select Register.

Timer Input signal. When the Capture event occurs, an interrupt is generated and the timer continues counting.

The timer continues counting up to the 16-bit Reload value stored in the Timer Reload High and Low Byte registers. Upon reaching the Reload value, the timer generates an interrupt and continues counting.

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

1. Write to the Timer Control register to:
   - Disable the timer
   - Configure the timer for Capture mode.
   - Set the prescale value.
   - Set the Capture edge (rising or falling) for the Timer Input.
2. Write to the Timer High and Low Byte registers to set the starting count value (typically 0001H).
3. Write to the Timer Reload High and Low Byte registers to set the Reload value.
4. Clear the Timer PWM High and Low Byte registers to 0000H. This allows user software to determine if interrupts were generated by either a capture event or a reload. If the PWM High and Low Byte registers still contain 0000H after the interrupt, then the interrupt was generated by a Reload.
5. If desired, enable the timer interrupt and set the timer interrupt priority by writing to the relevant interrupt registers.
6. Configure the associated GPIO port pin for the Timer Input alternate function.
7. Write to the Timer Control register to enable the timer and initiate counting.

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

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

Compare Mode

In Compare mode, the timer counts up to the 16-bit maximum Compare value stored in the Timer Reload High and Low Byte registers. The timer input is the system clock. Upon reaching the Compare value, the timer generates an interrupt and counting continues (the timer value is not reset to 0001H). Also, if the Timer Output alternate function is enabled, the Timer Output pin changes state (from Low to High or from High to Low) upon Compare.

– Configure the timer for Gated mode.
– Set the prescale value.

2. Write to the Timer High and Low Byte registers to set the starting count value. This only affects the first pass in Gated mode. After the first timer reset in Gated mode, counting always begins at the reset value of 0001H.
3. Write to the Timer Reload High and Low Byte registers to set the Reload value.
4. If desired, enable the timer interrupt and set the timer interrupt priority by writing to the relevant interrupt registers.
5. Configure the associated GPIO port pin for the Timer Input alternate function.
6. Write to the Timer Control register to enable the timer.
7. Assert the Timer Input signal to initiate the counting.

Capture/Compare Mode

In Capture/Compare mode, the timer begins counting on the *first* external Timer Input transition. The desired transition (rising edge or falling edge) is set by the TPOL bit in the Timer Control Register. The timer input is the system clock.

Every subsequent desired transition (after the first) of the Timer Input signal captures the current count value. The Capture value is written to the Timer PWM High and Low Byte Registers. When the Capture event occurs, an interrupt is generated, the count value in the Timer High and Low Byte registers is reset to 0001H, and counting resumes.

If no Capture event occurs, the timer counts up to the 16-bit Compare value stored in the Timer Reload High and Low Byte registers. Upon reaching the Compare value, the timer generates an interrupt, the count value in the Timer High and Low Byte registers is reset to 0001H and counting resumes.

The steps for configuring a timer for Capture/Compare mode and initiating the count are as follows:

1. Write to the Timer Control register to:
   – Disable the timer
   – Configure the timer for Capture/Compare mode.
   – Set the prescale value.
   – Set the Capture edge (rising or falling) for the Timer Input.
2. Write to the Timer High and Low Byte registers to set the starting count value (typically 0001H).
3. Write to the Timer Reload High and Low Byte registers to set the Compare value.
4. If desired, enable the timer interrupt and set the timer interrupt priority by writing to the relevant interrupt registers.

0 = Disable Multiprocessor mode.
1 = Enable Multiprocessor mode.

MPE—Multiprocessor Enable
0 = The UART processes all received data bytes.
1 = The UART processes only data bytes in which the multiprocessor data bit (9th bit) is set to 1.

MPBT—Multiprocessor Bit Transmitter
This bit is applicable only when Multiprocessor (9-bit) mode is enabled.
0 = Send a 0 in the multiprocessor bit location of the data stream (9th bit).
1 = Send a 1 in the multiprocessor bit location of the data stream (9th bit).

Reserved
These bits are reserved and must be 0.

$\overline{\text{RDAIRQ}}$—Receive Data Interrupt $\overline{\text{Enable}}$
0 = Received data and receiver errors generates an interrupt request to the Interrupt Controller.
1 = Received data does not generate an interrupt request to the Interrupt Controller. Only receiver errors generate an interrupt request. The associated DMA will still be notified that received data is available.

IREN—Infrared Encoder/Decoder Enable
0 = Infrared Encoder/Decoder is disabled. UART operates normally operation.
1 = Infrared Encoder/Decoder is enabled. The UART transmits and receives data through the Infrared Encoder/Decoder.

UART*x* Baud Rate High and Low Byte Registers

The UART*x* Baud Rate High and Low Byte registers (Tables 56 and 57) combine to create a 16-bit baud rate divisor value (BRG[15:0]) that sets the data transmission rate (baud rate) of the UART.

Table 56. UART*x* Baud Rate High Byte Register (U*x*BRH)

BITS	7	6	5	4	3	2	1	0
FIELD	BRH							
RESET	1	1	1	1	1	1	1	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	F46H and F4EH							

If the current ADC Analog Input is not the highest numbered input to be converted, DMA_ADC initiates data conversion in the next higher numbered ADC Analog Input.

Configuring DMA_ADC for Data Transfer

Follow these steps to configure and enable DMA_ADC:

1. Write the DMA_ADC Address register with the 7 most-significant bits of the Register File address for data transfers.
2. Write to the DMA_ADC Control register to complete the following:
   – Enable the DMA_ADC interrupt request, if desired
   – Select the number of ADC Analog Inputs to convert
   – Enable the DMA_ADC channel

Caution: When using the DMA_ADC to perform conversions on multiple ADC inputs and the ADC_IN field in the DMA_ADC Control Register is greater than 000b, the Analog-to-Digital Converter must be configured for Single-Shot mode.

Continuous mode operation of the ADC can **only** be used in conjunction with DMA_ADC if the ADC_IN field in the DMA_ADC Control Register is reset to 000b to enable conversion on ADC Analog Input 0 only.

DMA Control Register Definitions

DMA*x* Control Register

The DMA*x* Control register is used to enable and select the mode of operation for DMA*x*.

Table 71. DMA*x* Control Register (DMA*x*CTL)

BITS	7	6	5	4	3	2	1	0
FIELD	DEN	DLE	DDIR	IRQEN	WSEL	RSS		
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	FB0H, FB8H							

DEN—DMA*x* Enable
0 = DMA*x* is disabled and data transfer requests are disregarded.

DMA_ADC Control Register

The DMA_ADC Control register enables and sets options (DMA enable and interrupt enable) for ADC operation.

Table 78. DMA_ADC Control Register (DMAACTL)

BITS	7	6	5	4	3	2	1	0
FIELD	DAEN	IRQEN	Reserved		ADC_IN			
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	FBEH							

DAEN—DMA_ADC Enable
0 = DMA_ADC is disabled and the ADC Analog Input Number (ADC_IN) is reset to 0.
1 = DMA_ADC is enabled.

IRQEN—Interrupt Enable
0 = DMA_ADC does not generate any interrupts.
1 = DMA_ADC generates an interrupt after transferring data from the last ADC Analog Input specified by the ADC_IN field.

Reserved
These bits are reserved and must be 0.

ADC_IN—ADC Analog Input Number
These bits set the number of ADC Analog Inputs to be used in the continuous update (data conversion followed by DMA data transfer). The conversion always begins with ADC Analog Input 0 and then progresses sequentially through the other selected ADC Analog Inputs.
0000 = ADC Analog Input 0 updated.
0001 = ADC Analog Inputs 0-1 updated.
0010 = ADC Analog Inputs 0-2 updated.
0011 = ADC Analog Inputs 0-3 updated.
0100 = ADC Analog Inputs 0-4 updated.
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.

Table 93. On-Chip Debugger Commands

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

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

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

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

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

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

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

OCD Watchpoint Address Register

The OCD Watchpoint Address register specifies the lower 8 bits of the Register File address bus to match when generating Watchpoint Debug Breaks. The full 12-bit Register File address is given by {WPTCTL3:0], WPTADDR[7:0]}.

Table 97. OCD Watchpoint Address (WPTADDR)

BITS	7	6	5	4	3	2	1	0
FIELD	WPTADDR[7:0]							
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

WPTADDR[7:0]—Watchpoint Register File Address
These bits specify the lower eight bits of the register address to match when generating a Watchpoint Debug Break.

OCD Watchpoint Data Register

The OCD Watchpoint Data register specifies the data to match if Watchpoint data match is enabled.

Table 98. OCD Watchpoint Data (WPTDATA)

BITS	7	6	5	4	3	2	1	0
FIELD	WPTDATA[7:0]							
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

WPTDATA[7:0]—Watchpoint Register File Data
These bits specify the Register File data to match when generating Watchpoint Debug Breaks with the WPDM bit (WPTCTL[5]) is set to 1.

SPI Master Mode Timing

Figure 96 and Table 110 provide timing information for SPI Master mode pins. Timing is shown with SCK rising edge used to source MOSI output data, SCK falling edge used to sample MISO input data. Timing on the SS output pin(s) is controlled by software.

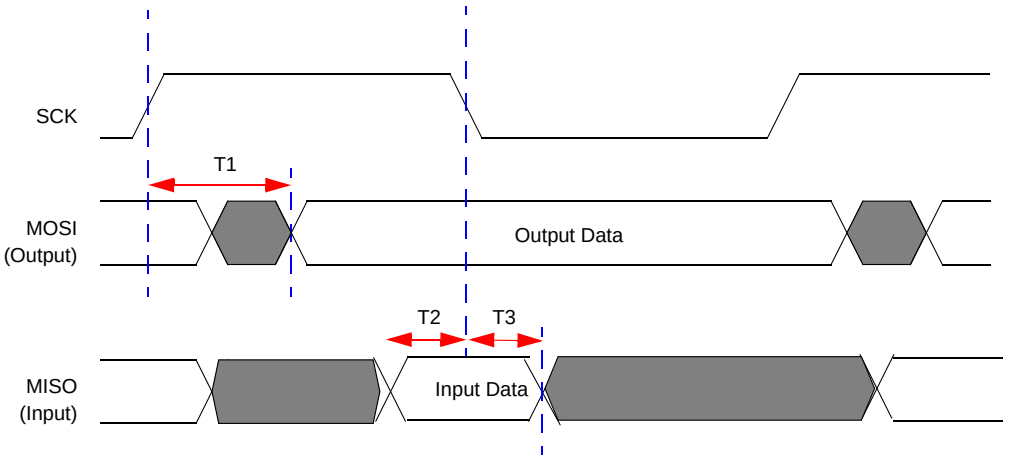


Figure 96. SPI Master Mode Timing

Table 110. SPI Master Mode Timing

		Delay (ns)	
Parameter	**Abbreviation**	**Minimum**	**Maximum**
T_1	SCK Rise to MOSI output Valid Delay	-5	+5
T_2	MISO input to SCK (receive edge) Setup Time	20	
T_3	MISO input to SCK (receive edge) Hold Time	0	

eZ8 CPU Instruction Set

Assembly Language Programming Introduction

The eZ8 CPU assembly language provides a means for writing an application program without having to be concerned with actual memory addresses or machine instruction formats. A program written in assembly language is called a source program. Assembly language allows the use of symbolic addresses to identify memory locations. It also allows mnemonic codes (opcodes and operands) to represent the instructions themselves. The opcodes identify the instruction while the operands represent memory locations, registers, or immediate data values.

Each assembly language program consists of a series of symbolic commands called statements. Each statement can contain labels, operations, operands and comments.

Labels can be assigned to a particular instruction step in a source program. The label identifies that step in the program as an entry point for use by other instructions.

The assembly language also includes assembler directives that supplement the machine instruction. The assembler directives, or pseudo-ops, are not translated into a machine instruction. Rather, the pseudo-ops are interpreted as directives that control or assist the assembly process.

The source program is processed (assembled) by the assembler to obtain a machine language program called the object code. The object code is executed by the eZ8 CPU. An example segment of an assembly language program is detailed in the following example.

Assembly Language Source Program Example

```
JP START          ; Everything after the semicolon is a comment.

START:            ; A label called “START”. The first instruction (JP START) in this
                  ; example causes program execution to jump to the point within the
                  ; program where the START label occurs.

LD R4, R7         ; A Load (LD) instruction with two operands. The first operand,
                  ; Working Register R4, is the destination. The second operand,
                  ; Working Register R7, is the source. The contents of R7 is
                  ; written into R4.

LD 234H, #%01     ; Another Load (LD) instruction with two operands.
                  ; The first operand, Extended Mode Register Address 234H,
                  ; identifies the destination. The second operand, Immediate Data
```

eZ8 CPU Instruction Classes

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

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

Tables 118 through 125 contain the instructions belonging to each group and the number of operands required for each instruction. Some instructions appear in more than one table as these instruction can be considered as a subset of more than one category. Within these tables, the source operand is identified as 'src', the destination operand is 'dst' and a condition code is 'cc'.

Table 118. Arithmetic Instructions

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

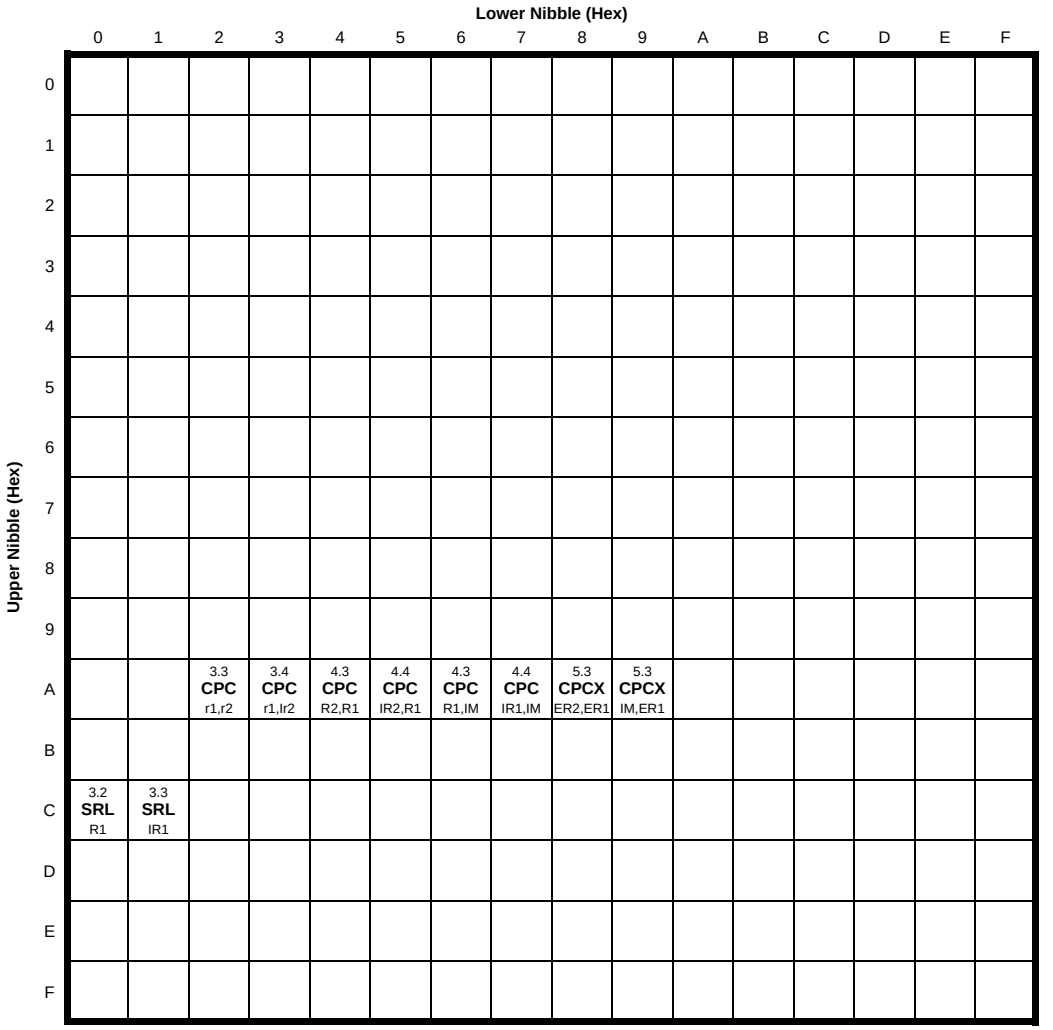


Figure 102. Second Opcode Map after 1FH

Precharacterization Product

The product represented by this document is newly introduced and ZiLOG has not completed the full characterization of the product. The document states what ZiLOG knows about this product at this time, but additional features or nonconformance with some aspects of the document might be found, either by ZiLOG or its customers in the course of further application and characterization work. In addition, ZiLOG cautions that delivery might be uncertain at times, due to start-up yield issues.

ZiLOG, Inc.
532 Race Street
San Jose, CA 95126
Telephone (408) 558-8500
FAX 408 558-8300
Internet: www.zilog.com

Document Information

Document Number Description

The Document Control Number that appears in the footer on each page of this document contains unique identifying attributes, as indicated in the following table:

PS	Product Specification
0176	Unique Document Number
01	Revision Number
0702	Month and Year Published

F

G

H

I