Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

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

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	eZ8
Core Size	8-Bit
Speed	20MHz
Connectivity	I²C, IrDA, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, DMA, POR, PWM, WDT
Number of I/O	46
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	-40°C ~ 105°C (TA)
Mounting Type	Surface Mount
Package / Case	64-LQFP
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/zilog/z8f4822ar020eg

List of Tables

Table 1. Z8 Encore! XP F64xx Series Part Selection Guide 2
Table 2. Z8 Encore! XP F64xx Series Package Options 7
Table 3. Signal Descriptions 14
Table 4. Pin Characteristics of the Z8 Encore! XP F64xx Series 17
Table 5. Z8 Encore! XP F64xx Series Program Memory Maps 19
Table 6. Z8 Encore! XP F64xx Series Information Area Map 21
Table 7. Z8 Encore! XP F64xx Series Register File Address Map 22
Table 8. Reset and Stop Mode Recovery Characteristics and Latency 28
Table 9. Reset Sources and Resulting Reset Type 29
Table 10. Stop Mode Recovery Sources and Resulting Action 33
Table 11. Port Availability by Device and Package Type 36
Table 12. Port Alternate Function Mapping 38
Table 13. GPIO Port Registers and Subregisters 39
Table 14. Port A–H GPIO Address Registers (PxADDR) 40
Table 15. Port A–H Control Registers (PxCTL) 41
Table 16. Port A–H Data Direction Subregisters 41
Table 17. Port A–H Alternate Function Subregisters 42
Table 18. Port A–H Output Control Subregisters 43
Table 19. Port A–H High Drive Enable Subregisters 44
Table 20. Port A–H Stop Mode Recovery Source Enable Subregisters 45
Table 21. Port A–H Input Data Registers (PxIN) 46
Table 22. Port A–H Output Data Register (PxOUT) 46
Table 23. Interrupt Vectors in Order of Priority 48
Table 24. Interrupt Request 0 Register (IRQ0) 52
Table 25. Interrupt Request 1 Register (IRQ1) 53
Table 26. Interrupt Request 2 Register (IRQ2) 54
Table 27. IRQ0 Enable and Priority Encoding 55
Table 28. IRQ0 Enable High Bit Register (IRQ0ENH) 55
Table 29. IRQ0 Enable Low Bit Register (IRQ0ENL) 56
Table 30. IRQ1 Enable and Priority Encoding 56
Table 31. IRQ1 Enable High Bit Register (IRQ1ENH) 57
Table 32. IRQ1 Enable Low Bit Register (IRQ1ENL) 57
Table 33. IRQ2 Enable and Priority Encoding 58

Table 70. SPI Baud Rate Low Byte Register (SPIBRL) 127
Table 71. I²C Data Register (I2CDATA) 142
Table 72. I2C Status Register (I2CSTAT) 142
Table 73. I2C Control Register (I2CCTL) 144
Table 74. I²C Baud Rate Low Byte Register (I2CBRL) 146
Table 75. I²C Baud Rate High Byte Register (I2CBRH) 146
Table 76. I²C Diagnostic State Register (I2CDST) 147
Table 77. I²C Diagnostic Control Register (I2CDIAG) 149
Table 78. DMAx Control Register (DMAxCTL) 153
Table 79. DMAx I/O Address Register (DMAxIO) 154
Table 80. DMAx Address High Nibble Register (DMAxH) 155
Table 81. DMAx Start/Current Address Low Byte Register (DMAxSTART) 156
Table 82. DMAx End Address Low Byte Register (DMAxEND) 156
Table 83. DMA_ADC Register File Address Example 157
Table 84. DMA_ADC Control Register (DMAACTL) 158
Table 85. DMA_ADC Address Register (DMAA_ADDR) 158
Table 86. DMA_ADC Status Register (DMAA_STAT) 159
Table 87. ADC Control Register (ADCCTL) 165
Table 88. ADC Data High Byte Register (ADCD_H) 167
Table 89. ADC Data Low Bits Register (ADCD_L) 168
Table 90. Flash Memory Configurations 169
Table 91. Flash Memory Sector Addresses 169
Table 92. Z8 Encore! XP F64xx Series Information Area Map 171
Table 93. Flash Control Register (FCTL) 176
Table 94. Flash Status Register (FSTAT) 177
Table 95. Flash Sector Protect Register (FPROT) 178
Table 96. Page Select Register (FPS) 178
Table 97. Flash Frequency High Byte Register (FFREQH) 179
Table 98. Flash Frequency Low Byte Register (FFREQL) 179
Table 99. Flash Option Bits At Flash Memory Address 0000H 181
Table 100. Options Bits at Flash Memory Address 0001H 182
Table 101. OCD Baud-Rate Limits 186
Table 102. On-Chip Debugger Commands 189
Table 103. OCD Control Register (OCDCTL) 193
Table 104. OCD Status Register (OCDSTAT) 194
Table 105. Recommended Crystal Oscillator Specifications (20 MHz Operation) . . . 197

I²C

The I²C controller makes the Z8 Encore! XP F64xx Series compatible with the I²C protocol. The I²C controller consists of two bidirectional bus lines, a serial data (SDA) line and a serial clock (SCL) line.

Serial Peripheral Interface

The serial peripheral interface allows the Z8 Encore! XP F64xx Series to exchange data between other peripheral devices such as EEPROMs, A/D converters and ISDN devices. The SPI is a full-duplex, synchronous, character-oriented channel that supports a four-wire interface.

Timers

Up to four 16-bit reloadable timers can be used for timing/counting events or for motor control operations. These timers provide a 16-bit programmable reload counter and operate in ONE-SHOT, CONTINUOUS, GATED, CAPTURE, COMPARE, CAPTURE AND COMPARE and PWM modes. Only 3 timers (Timer 0–2) are available in the 44-pin package.

Interrupt Controller

The Z8 Encore! XP F64xx Series products support up to 24 interrupts. These interrupts consist of 12 internal and 12 GPIO pins. The interrupts have 3 levels of programmable interrupt priority.

Reset Controller

The Z8 Encore! XP F64xx Series can be reset using the $\overline{\text{RESET}}$ pin, Power-On Reset, Watchdog Timer, STOP Mode exit, or Voltage Brown-Out (**VBO**) warning signal.

On-Chip Debugger

The Z8 Encore! XP F64xx Series features an integrated On-Chip Debugger. The OCD provides a rich set of debugging capabilities, such as reading and writing registers, programming the Flash, setting breakpoints and executing code. A single-pin interface provides communication to the OCD.

Table 7. Z8 Encore! XP F64xx Series Register File Address Map (Continued)

Address (Hex)	Register Description	Mnemonic	Reset (Hex)	Page
DMA 0 (continued)				
FB2	DMA0 End/Start Address High Nibble	DMA0H	XX	155
FB3	DMA0 Start Address Low Byte	DMA0START	XX	156
FB4	DMA0 End Address Low Byte	DMA0END	XX	156
DMA 1				
FB8	DMA1 Control	DMA1CTL	00	153
FB9	DMA1 I/O Address	DMA1IO	XX	154
FBA	DMA1 End/Start Address High Nibble	DMA1H	XX	155
FBB	DMA1 Start Address Low Byte	DMA1START	XX	156
FBC	DMA1 End Address Low Byte	DMA1END	XX	156
DMA ADC				
FBD	DMA_ADC Address	DMAA_ADDR	XX	157
FBE	DMA_ADC Control	DMAACTL	00	158
FBF	DMA_ADC Status	DMAASTAT	00	159
Interrupt Controller				
FC0	Interrupt Request 0	IRQ0	00	51
FC1	IRQ0 Enable High Bit	IRQ0ENH	00	55
FC2	IRQ0 Enable Low Bit	IRQ0ENL	00	55
FC3	Interrupt Request 1	IRQ1	00	53
FC4	IRQ1 Enable High Bit	IRQ1ENH	00	56
FC5	IRQ1 Enable Low Bit	IRQ1ENL	00	56
FC6	Interrupt Request 2	IRQ2	00	54
FC7	IRQ2 Enable High Bit	IRQ2ENH	00	58
FC8	IRQ2 Enable Low Bit	IRQ2ENL	00	58
FC9–FCC	Reserved	—	XX	
FCD	Interrupt Edge Select	IRQES	00	60
FCE	Interrupt Port Select	IRQPS	00	60
FCF	Interrupt Control	IRQCTL	00	61
GPIO Port A				
FD0	Port A Address	PAADDR	00	40
FD1	Port A Control	PACTL	00	41
FD2	Port A Input Data	PAIN	XX	46

Note: XX = Undefined.

Interrupt Controller

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

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

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

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

Interrupt Vector Listing

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

IRQ2 Enable High and Low Bit Registers

Table 33 describes the priority control for IRQ2. The IRQ2 Enable High and Low Bit registers, shown in Tables 34 and 35, form a priority-encoded enabling for interrupts in the Interrupt Request 2 Register. Priority is generated by setting bits in each register.

Table 33. IRQ2 Enable and Priority Encoding

IRQ2ENH[*x*]	IRQ2ENL[*x*]	Priority	Description
0	0	Disabled	Disabled
0	1	Level 1	Low
1	0	Level 2	Nominal
1	1	Level 3	High

Note: *x* indicates register bits in the range [7:0].

Table 34. IRQ2 Enable High Bit Register (IRQ2ENH)

Bit	7	6	5	4	3	2	1	0
Field	T3ENH	U1RENH	U1TENH	DMAENH	C3ENH	C2ENH	C1ENH	C0ENH
RESET	0							
R/W	R/W							
Address	FC7H							

Bit	Description
[7] T3ENH	**Timer 3 Interrupt Request Enable High Bit**
[6] U1RENH	**UART 1 Receive Interrupt Request Enable High Bit**
[5] U1TENH	**UART 1 Transmit Interrupt Request Enable High Bit**
[4] DMAENH	**DMA Interrupt Request Enable High Bit**
[3] C3ENH	**Port C3 Interrupt Request Enable High Bit**
[2] C2ENH	**Port C2 Interrupt Request Enable High Bit**
[1] C1ENH	**Port C1 Interrupt Request Enable High Bit**
[0] C0ENH	**Port C0 Interrupt Request Enable High Bit**

If TPOL is set to 0, the ratio of the PWM output High time to the total period is calculated using the following equation:

$$\text{PWM Output High Time Ratio (\%)} = \frac{\text{Reload Value} - \text{PWM Value}}{\text{Reload Value}} \times 100$$

If TPOL is set to 1, the ratio of the PWM output High time to the total period is calculated using the following equation:

$$\text{PWM Output High Time Ratio (\%)} = \frac{\text{PWM Value}}{\text{Reload Value}} \times 100$$

CAPTURE Mode

In CAPTURE Mode, the current timer count value is recorded when the appropriate external timer input transition occurs. The Capture count value is written to the Timer PWM High and Low Byte Registers. The timer input is the system clock. The TPOL bit in the Timer Control 1 Register determines if the Capture occurs on a rising edge or a falling edge of the 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.

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

1. Write to the Timer Control 1 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 the 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 appropriate, enable the timer interrupt and set the timer interrupt priority by writing to the relevant interrupt registers.

The next time $\overline{SS}$ asserts, the MISO pin outputs SPIDAT[7], regardless of where the previous transaction left off. Writing a 1 to ABT clears this error flag.

SPI Interrupts

When SPI interrupts are enabled, the SPI generates an interrupt after character transmission/reception completes in both MASTER and SLAVE modes. A character can be defined to be 1 through 8 bits by the NUMBITS field in the SPI Mode Register. In slave mode it is not necessary for $\overline{SS}$ to deassert between characters to generate the interrupt. The SPI in Slave mode can also generate an interrupt if the $\overline{SS}$ signal deasserts prior to transfer of all the bits in a character (see description of slave abort error above). Writing a 1 to the IRQ bit in the SPI Status Register clears the pending SPI interrupt request. The IRQ bit must be cleared to 0 by the Interrupt Service Routine to generate future interrupts. To start the transfer process, an SPI interrupt may be forced by software writing a 1 to the STR bit in the SPICTL Register.

If the SPI is disabled, an SPI interrupt can be generated by a Baud Rate Generator time-out. This timer function must be enabled by setting the BIRQ bit in the SPICTL Register. This Baud Rate Generator time-out does not set the IRQ bit in the SPISTAT Register, just the SPI interrupt bit in the interrupt controller.

SPI Baud Rate Generator

In SPI Master Mode, the Baud Rate Generator creates a lower frequency serial clock (SCK) for data transmission synchronization between the Master and the external Slave. The input to the Baud Rate Generator is the system clock. The SPI Baud Rate High and Low Byte registers combine to form a 16-bit reload value, BRG[15:0], for the SPI Baud Rate Generator. The SPI baud rate is calculated using the following equation:

$$\text{SPI Baud Rate (bits/s)} = \frac{\text{System Clock Frequency (Hz)}}{2 \times \text{BRG[15:0]}}$$

Minimum baud rate is obtained by setting BRG[15:0] to 0000H for a clock divisor value of (2 X 65536 = 131072).

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

1. Disable the SPI by clearing the SPIEN bit in the SPI Control Register to 0.
2. Load the appropriate 16-bit count value into the SPI Baud Rate High and Low Byte registers.
3. Enable the Baud Rate Generator timer function and associated interrupt by setting the BIRQ bit in the SPI Control Register to 1.

DMA*x* Address High Nibble Register

The DMA*x* Address High Register, shown in Table 80, specifies the upper four bits of address for the Start/Current and End addresses of DMA*x*.

Table 80. DMA*x* Address High Nibble Register (DMA*x*H)

Bit	7	6	5	4	3	2	1	0
Field	DMA_END_H				DMA_START_H			
RESET	X							
R/W	R/W							
Address	FB2H, FBAH							

Bit	Description
[7:4] DMA_END_H	**DMA*x* End Address High Nibble** These bits, used with the DMA*x* End Address Low Register, form a 12-bit End Address. The full 12-bit address is provided by {DMA_END_H[3:0], DMA_END[7:0]}.
[3:0] DMA_START_H	**DMA*x* Start/Current Address High Nibble** These bits, used with the DMA*x* Start/Current Address Low Register, form a 12-bit Start/Current Address. The full 12-bit address is provided by {DMA_START_H[3:0], DMA_START[7:0]}.

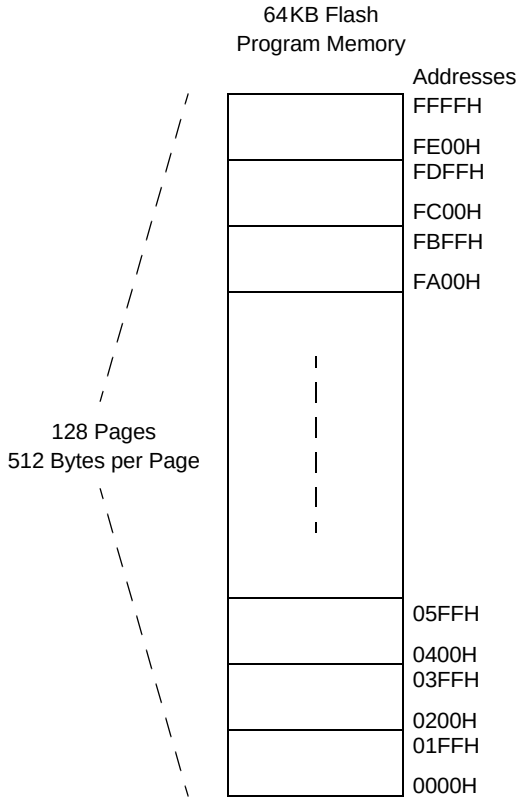


Figure 35. Flash Memory Arrangement

Information Area

Table 92 describes the Z8 Encore! XP F64xx Series Information Area. This 512-byte Information Area is accessed by setting bit 7 of the Page Select Register to 1. When access is enabled, the Information Area is mapped into Flash memory and overlays the 512 bytes at addresses FE00H to FFFFH. When the Information Area access is enabled, LDC instructions return data from the Information Area. CPU instruction fetches always comes from Flash memory regardless of the Information Area access bit. Access to the Information Area is read-only.

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

Flash Controller Behavior in Debug Mode

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

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

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

Flash Control Register Definitions

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

Flash Control Register: see page 175

Flash Status Register: see page 177

Page Select Register: see page 177

Flash Sector Protect Register: see page 178

Flash Frequency High and Low Byte Registers: see page 179

Flash Control Register

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

- Voltage Brown-Out reset
- Asserting the $\overline{\text{RESET}}$ pin Low to initiate a Reset
- Driving the DBG pin Low while the device is in STOP Mode initiates a system reset

OCD Data Format

The OCD interface uses the asynchronous data format defined for RS-232. Each character is transmitted as 1 start bit, 8 data bits (least significant bit first), and 1 stop bit, as shown in Figure 39.

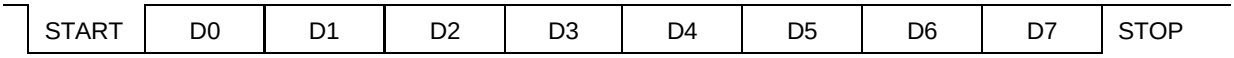


Figure 39. OCD Data Format

OCD Autobaud Detector/Generator

To run over a range of baud rates (bits per second) with various system clock frequencies, the On-Chip Debugger has an Autobaud Detector/Generator. After a reset, the OCD is idle until it receives data. The OCD requires that the first character sent from the host is the character 80H. The character 80H has eight continuous bits Low (one start bit plus 7 data bits). The Autobaud Detector measures this period and sets the OCD Baud Rate Generator accordingly.

The Autobaud Detector/Generator is clocked by the system clock. The minimum baud rate is the system clock frequency divided by 512. For optimal operation, the maximum recommended baud rate is the system clock frequency divided by 8. The theoretical maximum baud rate is the system clock frequency divided by 4. This theoretical maximum is possible for low noise designs with clean signals. Table 101 lists minimum and recommended maximum baud rates for sample crystal frequencies.

Table 101. OCD Baud-Rate Limits

System Clock Frequency (MHz)	Recommended Maximum Baud Rate (kbits/s)	Minimum Baud Rate (kbits/s)
20.0	2500	39.1
1.0	125.0	1.96
0.032768 (32kHz)	4.096	0.064

On-Chip Debugger Timing

Figure 52 and Table 117 provide timing information for the DBG pin. The DBG pin timing specifications assume a 4µs maximum rise and fall time.

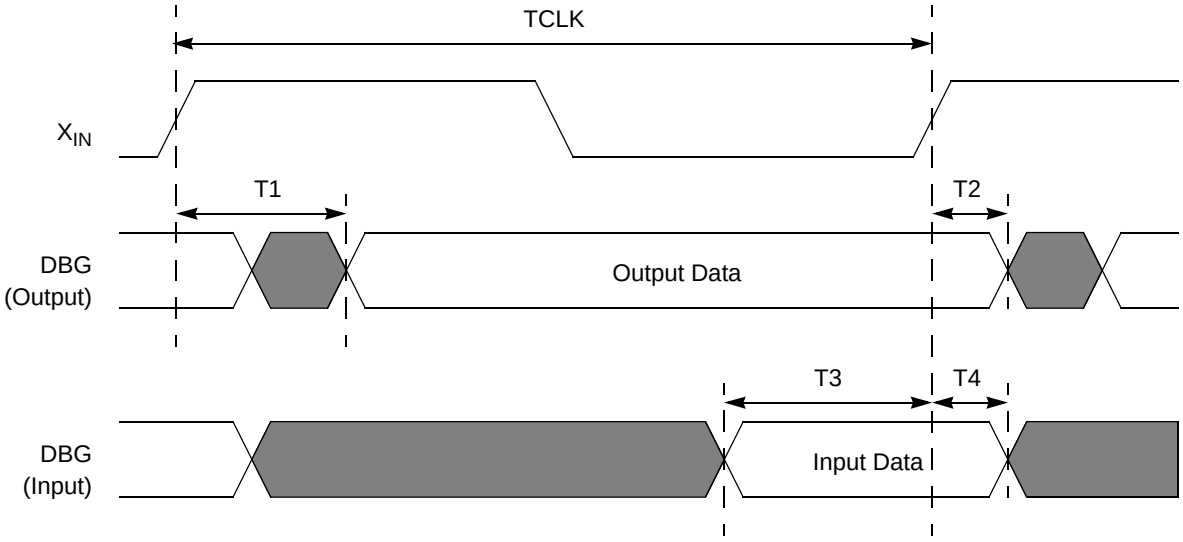


Figure 52. On-Chip Debugger Timing

Table 117. On-Chip Debugger Timing

		Delay (ns)	
Parameter	**Abbreviation**	**Minimum**	**Maximum**
DBG			
T_1	X_{IN} Rise to DBG Valid Delay	–	30
T_2	X_{IN} Rise to DBG Output Hold Time	2	–
T_3	DBG to X_{IN} Rise Input Setup Time	10	–
T_4	DBG to X_{IN} Rise Input Hold Time	5	–
	DBG frequency		System Clock/4

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 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 presents the operands in the *source, destination* order; however, ordering is op code-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 result is shown in Table 123.

Table 123. 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 result is shown in Table 124.

Table 135. 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
SRL	dst	Shift Right Logical
SWAP	dst	Swap Nibbles

eZ8 CPU Instruction Summary

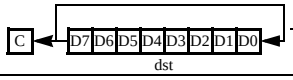
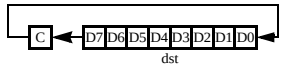
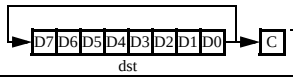
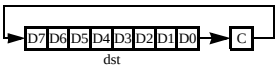
Table 136 summarizes the eZ8 CPU instructions. The table identifies the addressing modes employed by the instruction, the effect upon the Flags Register, the number of CPU clock cycles required for the instruction fetch, and the number of CPU clock cycles required for the instruction execution.

Table 136. eZ8 CPU Instruction Summary

Assembly Mnemonic	Symbolic Operation	Address Mode dst	Address Mode src	Opcode(s) (Hex)	Flags C	Z	S	V	D	H	Fetch Cycles	Instr. Cycles
ADC dst, src	dst ← dst + src + C	r	r	12	*	*	*	*	0	*	2	3
		r	Ir	13							2	4
		R	R	14							3	3
		R	IR	15							3	4
		R	IM	16							3	3
		IR	IM	17							3	4
ADCX dst, src	dst ← dst + src + C	ER	ER	18	*	*	*	*	0	*	4	3
		ER	IM	19							4	3

Note: Flags Notation:
* = Value is a function of the result of the operation.
– = Unaffected.
X = Undefined.
0 = Reset to 0.
1 = Set to 1.

Table 136. eZ8 CPU Instruction Summary (Continued)

Assembly Mnemonic	Symbolic Operation	Address Mode dst	Address Mode src	Opcode(s) (Hex)	Flags C	Z	S	V	D	H	Fetch Cycles	Instr. Cycles
POPX dst	dst ← @SP SP ← SP + 1	ER		D8	–	–	–	–	–	–	3	2
PUSH src	SP ← SP – 1 @SP ← src	R		70	–	–	–	–	–	–	2	2
		IR		71							2	3
		IM		1F 70							3	2
PUSHX src	SP ← SP – 1 @SP ← src	ER		C8	–	–	–	–	–	–	3	2
RCF	C ← 0			CF	0	–	–	–	–	–	1	2
RET	PC ← @SP SP ← SP + 2			AF	–	–	–	–	–	–	1	4
RL dst		R		90	*	*	*	*	–	–	2	2
		IR		91							2	3
RLC dst		R		10	*	*	*	*	–	–	2	2
		IR		11							2	3
RR dst		R		E0	*	*	*	*	–	–	2	2
		IR		E1							2	3
RRC dst		R		C0	*	*	*	*	–	–	2	2
		IR		C1							2	3
SBC dst, src	dst ← dst – src – C	r	r	32	*	*	*	*	1	*	2	3
		r	Ir	33							2	4
		R	R	34							3	3
		R	IR	35							3	4
		R	IM	36							3	3
		IR	IM	37							3	4
SBCX dst, src	dst ← dst – src – C	ER	ER	38	*	*	*	*	1	*	4	3
		ER	IM	39							4	3
SCF	C ← 1			DF	1	–	–	–	–	–	1	2

Note: Flags Notation:
* = Value is a function of the result of the operation.
– = Unaffected.
X = Undefined.
0 = Reset to 0.
1 = Set to 1.

Op Code Maps

A description of the op code map data and the abbreviations are provided in Figure 59 and Table 137. Figures 60 and 61 provide information about each of the eZ8 CPU instructions.

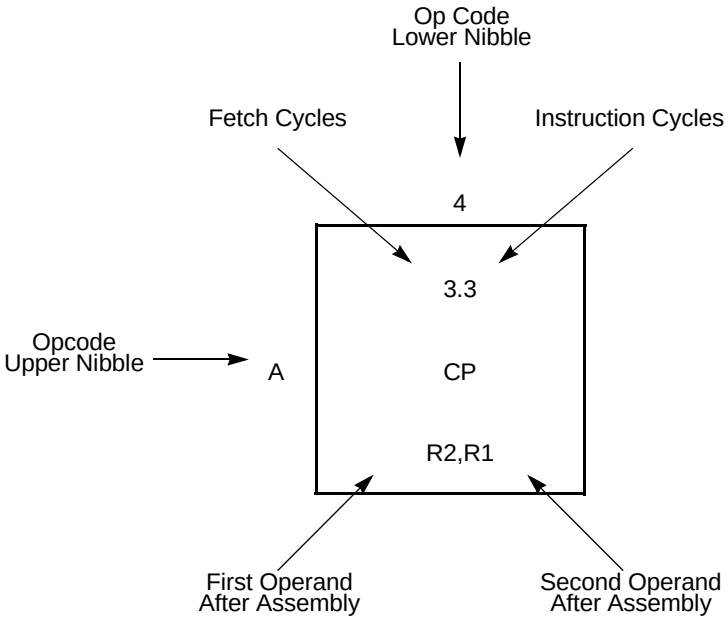


Figure 59. Op Code Map Cell Description

Table 137. Op Code Map Abbreviations

Abbreviation	Description	Abbreviation	Description
b	Bit position	IRR	Indirect register pair
cc	Condition code	p	Polarity (0 or 1)
X	8-bit signed index or displacement	r	4-bit working register
DA	Destination address	R	8-bit register
ER	Extended addressing register	r1, R1, Ir1, Irr1, IR1, rr1, RR1, IRR1, ER1	Destination address

Hex Address: FEB

Table 256. Port A–H Output Data Register (P*x*OUT)

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

Hex Address: FEC

Table 257. Port A–H GPIO Address Registers (P*x*ADDR)

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

Hex Address: FED

Table 258. Port A–H Control Registers (P*x*CTL)

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

Hex Address: FEE

Table 259. Port A–H Input Data Registers (PxIN)

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

increment word 231
INCW 231
indexed 229
indirect address prefix 229
indirect register 228
indirect register pair 228
indirect working register 228
indirect working register pair 228
infrared encoder/decoder (IrDA) 110
instruction set, ez8 CPU 226
instructions
ADC 231
ADCX 231
ADD 231
ADDX 231
AND 234
ANDX 234
arithmetic 231
BCLR 232
BIT 232
bit manipulation 232
block transfer 232
BRK 234
BSET 232
BSWAP 232, 235
BTJ 234
BTJNZ 234
BTJZ 234
CALL 234
CCF 232, 233
CLR 233
COM 234
CP 231
CPC 231
CPCX 231
CPU control 233
CPX 231
DA 231
DEC 231
DECW 231
DI 233
DJNZ 234
EI 233
HALT 233
INC 231
INCW 231
IRET 234
JP 234
LD 233
LDC 233
LDCI 232, 233
LDE 233
LDEI 232
LDX 233
LEA 233
load 233
logical 234
MULT 232
NOP 233
OR 234
ORX 234
POP 233
POPX 233
program control 234
PUSH 233
PUSHX 233
RCF 232, 233
RET 234
RL 235
RLC 235
rotate and shift 235
RR 235
RRC 235
SBC 232
SCF 232, 233
SRA 235
SRL 235
SRP 233
STOP 233
SUB 232
SUBX 232
SWAP 235
TCM 232
TCMX 232
TM 232
TMX 232
TRAP 234
watch-dog timer refresh 233

SPI status (SPISTAT) 124, 265
status, I2C 143
status, SPI 124
UARTx baud rate high byte (UxBRH) 107, 259, 262
UARTx baud rate low byte (UxBRL) 107, 260, 262
UARTx Control 0 (UxCTL0) 103, 106, 258, 259, 261
UARTx control 1 (UxCTL1) 104, 259, 261
UARTx receive data (UxRXD) 100, 258, 260
UARTx status 0 (UxSTAT0) 101, 258, 260
UARTx status 1 (UxSTAT1) 102, 259, 261
UARTx transmit data (UxTXD) 100, 258, 260
watchdog timer control (WDTCTL) 85, 283
watchdog timer reload high byte (WDTH) 87, 284
watchdog timer reload low byte (WDTL) 87, 284
watchdog timer reload upper byte (WDTU) 86, 283
register file 19
register file address map 23
register pair 229
register pointer 229
reset
and STOP mode characteristics 29
carry flag 232
controller 6
sources 30
RET 234
return 234
RL 235
RLC 235
rotate and shift instructions 235
rotate left 235
rotate left through carry 235
rotate right 235
rotate right through carry 235
RP 229
RR 229, 235
rr 229
RRC 235

S

SBC 232
SCF 232, 233
SDA and SCL (IrDA) signals 131
second opcode map after 1FH 248
serial clock 117
serial peripheral interface (SPI) 114
set carry flag 232, 233
set register pointer 233
shift right arithmetic 235
shift right logical 235
signal descriptions 15
single-shot conversion (ADC) 164
SIO 6
slave data transfer formats (I2C) 137
slave select 117
software trap 234
source operand 229
SP 229
SPI
architecture 114
baud rate generator 121
baud rate high and low byte register 127
clock phase 117
configured as slave 115
control register 123
control register definitions 122
data register 122
error detection 120
interrupts 121
mode fault error 120
mode register 126
multi-master operation 119
operation 116
overrun error 120
signals 116
single master, multiple slave system 115
single master, single slave system 114
status register 124
timing, PHASE = 0 118
timing, PHASE=1 119
SPI controller signals 15
SPI mode (SPIMODE) 126, 265
SPIBRH register 128, 266