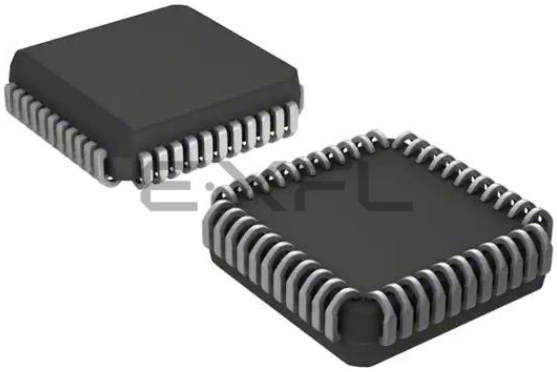


Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

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

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Obsolete
Core Processor	eZ8
Core Size	8-Bit
Speed	20MHz
Connectivity	I²C, IrDA, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, DMA, POR, PWM, WDT
Number of I/O	31
Program Memory Size	64KB (64K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	4K x 8
Voltage - Supply (Vcc/Vdd)	3V ~ 3.6V
Data Converters	A/D 8x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 105°C (TA)
Mounting Type	Surface Mount
Package / Case	44-LCC (J-Lead)
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/zilog/z8f6401vn020ec00tr

Figure 31. Flash Controller Operation Flow Chart 140
Figure 32. On-Chip Debugger Block Diagram 151
Figure 33. Interfacing the On-Chip Debugger's DBG Pin with an RS-232 Interface (1) 152
Figure 34. Interfacing the On-Chip Debugger's DBG Pin with an RS-232 Interface (2) 153
Figure 35. OCD Data Format 154
Figure 36. Recommended Crystal Oscillator Configuration (20MHz operation) 166
Figure 37. Nominal ICC Versus System Clock Frequency 170
Figure 38. Nominal Halt Mode ICC Versus System Clock Frequency 171
Figure 39. Port Input Sample Timing 176
Figure 40. GPIO Port Output Timing 177
Figure 41. On-Chip Debugger Timing 178
Figure 42. SPI Master Mode Timing 179
Figure 43. SPI Slave Mode Timing 180
Figure 44. I^2C Timing 181
Figure 45. Flags Register 201
Figure 46. Opcode Map Cell Description 202
Figure 47. First Opcode Map 204
Figure 48. Second Opcode Map after 1FH 205
Figure 49. 40-Lead Plastic Dual-Inline Package (PDIP) 206
Figure 50. 44-Lead Low-Profile Quad Flat Package (LQFP) 207
Figure 51. 44-Lead Plastic Lead Chip Carrier Package (PLCC) . . . 207
Figure 52. 64-Lead Low-Profile Quad Flat Package (LQFP) 208
Figure 53. 68-Lead Plastic Lead Chip Carrier Package (PLCC) . . . 209
Figure 54. 80-Lead Quad-Flat Package (QFP) 210

Table 32. IRQ2 Enable and Priority Encoding 53
Table 33. IRQ1 Enable High Bit Register (IRQ1ENH) 53
Table 34. IRQ2 Enable Low Bit Register (IRQ2ENL) 54
Table 35. IRQ2 Enable High Bit Register (IRQ2ENH) 54
Table 36. Interrupt Edge Select Register (IRQES) 55
Table 37. Interrupt Port Select Register (IRQPS) 55
Table 38. Interrupt Control Register (IRQCTL) 56
Table 39. Timer 0-3 High Byte Register (TxH) 67
Table 40. Timer 0-3 Low Byte Register (TxL) 67
Table 41. Timer 0-3 Reload High Byte Register (TxRH) 68
Table 42. Timer 0-3 Reload Low Byte Register (TxRL) 68
Table 43. Timer 0-3 PWM High Byte Register (TxPWMH) 69
Table 44. Timer 0-3 PWM Low Byte Register (TxPWML) 69
Table 45. Timer 0-3 Control Register (TxCTL) 70
Table 46. Watch-Dog Timer Approximate Time-Out Delays 73
Table 47. Watch-Dog Timer Control Register (WDTCTL) 75
Table 48. Watch-Dog Timer Reload Upper Byte Register (WDTU) 76
Table 49. Watch-Dog Timer Reload High Byte Register (WDTH) . 76
Table 50. Watch-Dog Timer Reload Low Byte Register (WDTL) . . 77
Table 51. UARTx Transmit Data Register (UxTXD) 86
Table 52. UARTx Receive Data Register (UxRXD) 87
Table 53. UARTx Status 0 Register (UxSTAT0) 87
Table 54. UARTx Control 0 Register (UxCTL0) 89
Table 55. UARTx Status 1 Register (UxSTAT1) 89
Table 56. UARTx Control 1 Register (UxCTL1) 90
Table 57. UARTx Baud Rate High Byte Register (UxBRH) 91
Table 58. UARTx Baud Rate Low Byte Register (UxBRL) 92
Table 59. UART Baud Rates 93
Table 60. SPI Clock Phase (PHASE) and Clock Polarity (CLKPOL) Operation 102
Table 61. SPI Data Register (SPIDATA) 106
Table 62. SPI Control Register (SPICTL) 107
Table 63. SPI Status Register (SPISTAT) 108
Table 64. SPI Mode Register (SPIMODE) 109
Table 65. SPI Baud Rate High Byte Register (SPIBRH) 110
Table 66. SPI Baud Rate Low Byte Register (SPIBRL) 110

Architecture

Figure 65 illustrates a block diagram of the interrupt controller.

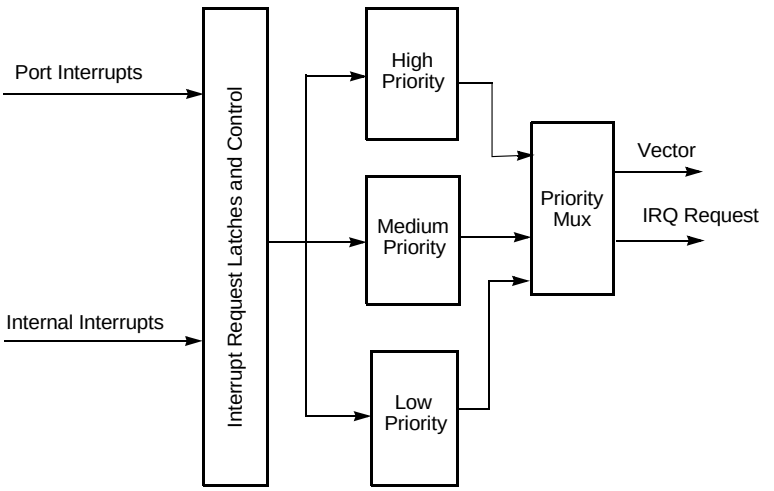


Figure 65. Interrupt Controller Block Diagram

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:

- Execution of an EI (Enable Interrupt) instruction
- Execution of an IRET (Return from Interrupt) 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 DI (Disable Interrupt) 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

out, first set the TPOL bit in the Timer Control Register to the start value before beginning One-Shot mode. Then, after starting the timer, set TPOL to the opposite bit value.

The steps for configuring a timer for One-Shot mode and initiating the count are as follows:

1. Write to the Timer Control register to:
   – Disable the timer
   – Configure the timer for One-Shot mode.
   – Set the prescale value.
   – If using the Timer Output alternate function, set the initial output level (High or Low).
2. Write to the Timer High and Low Byte registers to set the starting count value.
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. If using the Timer Output function, configure the associated GPIO port pin for the Timer Output alternate function.
6. Write to the Timer Control register to enable the timer and initiate counting.

In One-Shot mode, the system clock always provides the timer input. The timer period is given by the following equation:

$$\textbf{One-Shot Mode Time-Out Period (s)} = \frac{(\textbf{Reload Value} - \textbf{Start Value}) \times \textbf{Prescale}}{\textbf{System Clock Frequency (Hz)}}$$

Continuous Mode

In Continuous mode, the timer counts up to the 16-bit Reload value stored in the Timer Reload High and Low Byte registers. The timer input is the system clock. Upon reaching the Reload value, the timer generates an interrupt, the count value in the Timer High and Low Byte registers is reset to 0001H and counting resumes. 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 timer Reload.

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

1. Write to the Timer Control register to:
   – Disable the timer
   – Configure the timer for Continuous mode.
   – Set the prescale value.

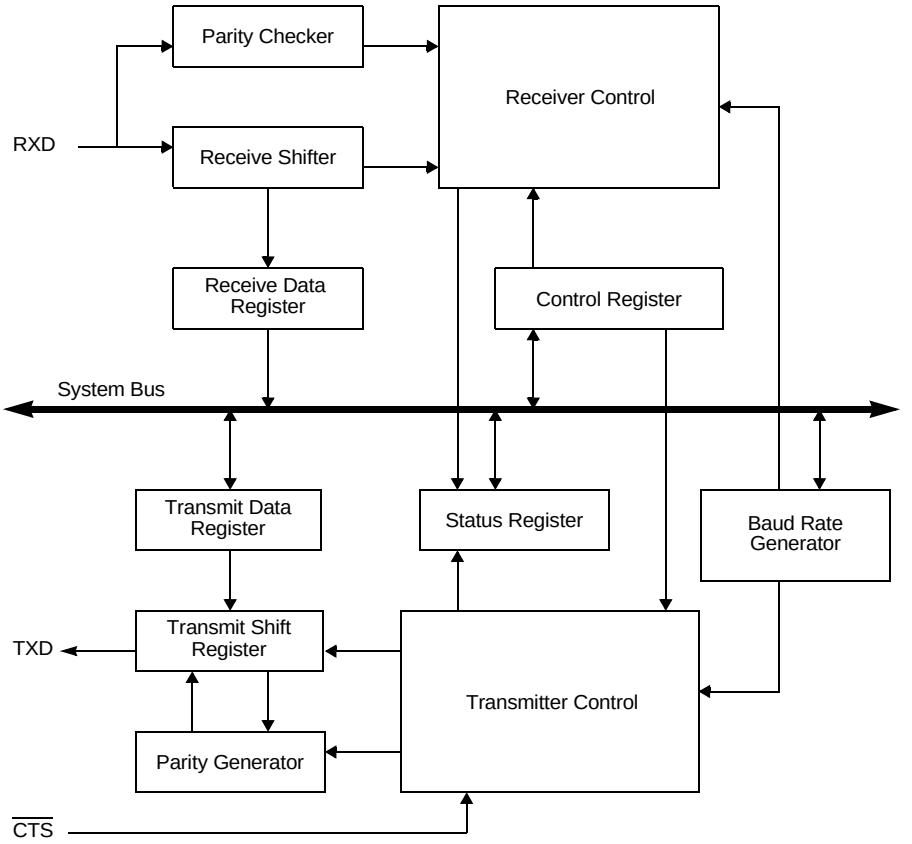


Figure 67. UART Block Diagram

Operation

Data Format

The UART always transmits and receives data in an 8-bit data format, least-significant bit first. An even or odd parity bit can be optionally added to the data stream. Each character begins with an active Low Start bit and ends with either 1 or 2 active High Stop bits.

Error Detection

The SPI contains error detection logic to support SPI communication protocols and recognize when communication errors have occurred. The SPI Status register indicates when a data transmission error has been detected.

Overrun (Write Collision)

An overrun error (write collision) indicates a write to the SPI Data register was attempted while a data transfer is in progress. An overrun sets the OVR bit in the SPI Status register to 1. Writing a 1 to OVR clears this error flag.

Mode Fault (Multi-Master Collision)

A mode fault indicates when more than one Master is trying to communicate at the same time (a multi-master collision). The mode fault is detected when the enabled Master's $\overline{SS}$ pin is asserted. A mode fault sets the COL bit in the SPI Status register to 1. Writing a 1 to COL clears this error flag.

SPI Interrupts

When SPI interrupts are enabled, the SPI generates an interrupt after data transmission. The SPI in Master mode generates an interrupt after a character has been sent. A character can be defined to be 1 through 8 bits by the NUMBITS field in the SPI Mode register. The SPI in Slave mode generates an interrupt when the $\overline{SS}$ signal deasserts to indicate completion of the data transfer. Writing a 1 to the IRQ bit in the SPI Status Register clears the pending interrupt request. If the SPI is disabled, an SPI interrupt can be generated by a Baud Rate Generator time-out.

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 reload value must be greater than or equal to 0002H for SPI operation (maximum baud rate is system clock frequency divided by 4). The SPI baud rate is calculated using the following equation:

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

When the SPI 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:

2. The I2C Controller waits for the slave to send an Acknowledge (by pulling the SDA signal Low). If the slave pulls the SDA signal High (Not-Acknowledge), the I2C Controller sends a Stop signal.
3. If the slave needs to service an interrupt, it pulls the SCL signal Low, which halts I2C operation.
4. If there is no other data in the I2C Data register or the STOP bit in the I2C Control register is set by software, then the Stop signal is sent.

Figure 79 illustrates the data transfer format for a 7-bit addressed slave. Shaded regions indicate data transferred from the I2C Controller to slaves and unshaded regions indicate data transferred from the slaves to the I2C Controller.

Figure 79. 7-Bit Addressed Slave Data Transfer Format

The data transfer format for a transmit operation on a 7-bit addressed slave is as follows:

1. Software asserts the IEN bit in the I2C Control register.
2. Software asserts the TXI bit of the I2C Control register to enable Transmit interrupts.
3. The I2C interrupt asserts, because the I2C Data register is empty
4. Software responds to the TDRE bit by writing a 7-bit slave address followed by a 0 (write) to the I2C Data register.
5. Software asserts the START bit of the I2C Control register.
6. The I2C Controller sends the START condition to the I2C slave.
7. The I2C Controller loads the I2C Shift register with the contents of the I2C Data register.
8. After one bit of address has been shifted out by the SDA signal, the Transmit interrupt is asserted.
9. Software responds by writing the contents of the data into the I2C Data register.
10. The I2C Controller shifts the rest of the address and write bit out by the SDA signal.
11. 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.
12. The I2C Controller loads the contents of the I2C Shift register with the contents of the I2C Data register.
13. The I2C Controller shifts the data out of via the SDA signal. After the first bit is sent, the Transmit interrupt is asserted.

14. Software responds by setting the STOP bit of the I²C Control register.
15. If no new data is to be sent or address is to be sent, software responds by clearing the TXI bit of the I²C Control register.
16. The I²C Controller completes transmission of the data on the SDA signal.
17. The I²C Controller sends the STOP condition to the I²C bus.

Writing a Transaction with a 10-Bit Address

1. The I²C Controller shifts the I²C Shift register out onto SDA signal.
2. The I²C Controller waits for the slave to send an Acknowledge (by pulling the SDA signal Low). If the slave pulls the SDA signal High (Not-Acknowledge), the I²C Controller sends a Stop signal.
3. If the slave needs to service an interrupt, it pulls the SCL signal low, which halts I²C operation.
4. If there is no other data in the I²C Data register or the STOP bit in the I²C Control register is set by software, then the Stop signal is sent.

The data transfer format for a 10-bit addressed slave is illustrated in the figure below. Shaded regions indicate data transferred from the I²C Controller to slaves and unshaded regions indicate data transferred from the slaves to the I²C Controller.

Figure 80. 10-Bit Addressed Slave Data Transfer Format

The first seven bits transmitted in the first byte are 11110XX. The two bits XX are the two most-significant bits of the 10-bit address. The lowest bit of the first byte transferred is the write signal. The transmit operation is carried out in the same manner as 7-bit addressing.

The data transfer format for a transmit operation on a 10-bit addressed slave is as follows:

1. Software asserts the IEN bit in the I²C Control register.
2. Software asserts the TXI bit of the I²C Control register to enable Transmit interrupts.
3. The I²C interrupt asserts because the I²C Data register is empty.
4. Software responds to the TDRE bit by writing the first slave address byte. The least-significant bit must be 0 for the write operation.
5. Software asserts the START bit of the I²C Control register.
6. The I²C Controller sends the START condition to the I²C slave.

1. Software writes the I2C Data register with a 7-bit slave address followed by a 1 (read).
2. Software asserts the START bit of the I2C Control register.
3. 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.
4. The I2C Controller sends the START condition.
5. The I2C Controller sends the address and read bit by the SDA signal.
6. The I2C slave sends an Acknowledge by pulling the SDA signal Low during the next high period of SCL.
7. The I2C Controller reads the first byte of data from the I2C slave.
8. The I2C Controller asserts the Receive interrupt.
9. Software responds by reading the I2C Data register.
10. The I2C Controller sends a NAK to the I2C slave.
11. A NAK interrupt is generated by the I2C Controller.
12. Software responds by setting the STOP bit of the I2C Control register.
13. A STOP condition is sent to the I2C slave.

Reading a Transaction with a 10-Bit Address

Figure 82 illustrates the receive format for a 10-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 1st 7 bits	W=0	A	Slave address 2nd Byte	A	S	Slave Address 1st 7 bits	R=1	A	Data	A	Data	$\overline{A}$	P

Figure 82. Receive Data Format for a 10-Bit Addressed Slave

The first seven bits transmitted in the first byte are 11110XX. The two bits XX are the two most-significant bits of the 10-bit address. The lowest bit of the first byte transferred is the write signal.

The data transfer format for a receive operation on a 10-bit addressed slave is as follows:

1. Software writes an address 11110B followed by the two address bits and a 0 (write).
2. Software asserts the START bit of the I2C Control register.
3. The I2C Controller sends the Start condition.

1 = DMA*x* is enabled and initiates a data transfer upon receipt of a request from the trigger source.

DLE—DMA*x* Loop Enable
0 = DMA*x* reloads the original Start Address and is then disabled after the End Address data is transferred.
1 = DMA*x*, after the End Address data is transferred, reloads the original Start Address and continues operating.

DDIR—DMA*x* Data Transfer Direction
0 = Register File → on-chip peripheral control register.
1 = on-chip peripheral control register → Register File.

IRQEN—DMA*x* Interrupt Enable
0 = DMA*x* does not generate any interrupts.
1 = DMA*x* generates an interrupt when the End Address data is transferred.

WSEL—Word Select
0 = DMA*x* transfers a single byte per request.
1 = DMA*x* transfers a two-byte word per request. The address for the on-chip peripheral control register must be an even address.

RSS—Request Trigger Source Select
The Request Trigger Source Select field determines the peripheral that can initiate a DMA request transfer. The corresponding interrupts do not need to be enabled within the Interrupt Controller to initiate a DMA transfer. However, if the Request Trigger Source can enable or disable the interrupt request sent to the Interrupt Controller, the interrupt request must be enabled within the Request Trigger Source block.
000 = Timer 0.
001 = Timer 1.
010 = Timer 2.
011 = Timer 3.
100 = DMA0 Control register: UART0 Received Data register contains valid data. DMA1 Control register: UART0 Transmit Data register empty.
101 = DMA0 Control register: UART1 Received Data register contains valid data. DMA1 Control register: UART1 Transmit Data register empty.
110 = DMA0 Control register: I^2C Receiver Interrupt. DMA1 Control register: I^2C Transmitter Interrupt register empty.
111 = Reserved.

DMA*x* I/O Address Register

The DMA*x* I/O Address register contains the low byte of the on-chip peripheral address for data transfer. The full 12-bit Register File address is given by {FH, DMA*x*_IO[7:0]}.

Flash Memory

Overview

The Z8F640x family features up to 64KB (65,536 bytes) of non-volatile Flash memory with read/write/erase capability. The Flash Memory can be programmed and erased in-circuit by either user code or through the On-Chip Debugger.

The Flash memory array is arranged in pages with 512 bytes per page. The 512-byte page is the minimum Flash block size that can be erased. Each page is divided into 8 rows of 64 bytes. The Flash memory also contains a High Sector that can be enabled for writes and erase separately from the rest of the Flash array. The first 2 bytes of the Flash Program memory are used as Option Bits. Refer to the **Option Bits** chapter for more information on their operation.

Table 83 describes the Flash memory configuration for each device in the Z8F640x family. Figure 84 illustrates the Flash memory arrangement.

Table 83. Z8F640x family Flash Memory Configurations

Part Number	Flash Size KB (Bytes)	Flash Pages	Program Memory Addresses	Flash High Sector Size KB (Bytes)	High Sector Addresses
Z8F160x	16 (16,384)	32	0000H - 3FFFH	1 (1024)	3C00H - 3FFFH
Z8F240x	24 (24,576)	48	0000H - 5FFFH	2 (2048)	5800H - 5FFFH
Z8F320x	32 (32,768)	64	0000H - 7FFFH	2 (2048)	7800H - 7FFFH
Z8F480x	48 (49,152)	96	0000H - BFFFH	4 (4096)	B000H - BFFFH
Z8F640x	64 (65,536)	128	0000H - FFFFH	8 (8192)	E000H - FFFFH

- Power-on reset
- Voltage Brownout reset
- Asserting the $\overline{\text{RESET}}$ pin Low to initiate a Reset.
- Driving the DBG pin Low while the Z8F640x family 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 1.5 Stop bits (Figure 89)

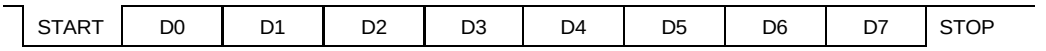


Figure 89. OCD Data Format

OCD Auto-Baud Detector/Generator

To run over a range of baud rates (data bits per second) with various system clock frequencies, the On-Chip Debugger has an Auto-Baud 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 Auto-Baud Detector measures this period and sets the OCD Baud Rate Generator accordingly.

The Auto-Baud Detector/Generator is clocked by the Z8F640x family device 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 92 lists minimum and recommended maximum baud rates for sample crystal frequencies.

Table 92. 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

```
DBG <-- 03H
DBG --> RuntimeCounter[15:8]
DBG --> RuntimeCounter[7:0]
```

- **Write OCD Control Register (04H)**—The Write OCD Control Register command writes the data that follows to the OCDCTL register. When the Read Protect Option Bit is enabled, the DBGMODE bit (OCDCTL[7]) can only be set to 1, it cannot be cleared to 0 and the only method of putting the Z8F640x family device back into normal operating mode is to reset the device.

```
DBG <-- 04H
DBG <-- OCDCTL[7:0]
```

- **Read OCD Control Register (05H)**—The Read OCD Control Register command reads the value of the OCDCTL register.

```
DBG <-- 05H
DBG --> OCDCTL[7:0]
```

- **Write Program Counter (06H)**—The Write Program Counter command writes the data that follows to the eZ8 CPU's Program Counter (PC). If the Z8F640x family device is not in Debug mode or if the Read Protect Option Bit is enabled, the Program Counter (PC) values are discarded.

```
DBG <-- 06H
DBG <-- ProgramCounter[15:8]
DBG <-- ProgramCounter[7:0]
```

- **Read Program Counter (07H)**—The Read Program Counter command reads the value in the eZ8 CPU's Program Counter (PC). If the Z8F640x family device is not in Debug mode or if the Read Protect Option Bit is enabled, this command returns FFFFH.

```
DBG <-- 07H
DBG --> ProgramCounter[15:8]
DBG --> ProgramCounter[7:0]
```

- **Write Register (08H)**—The Write Register command writes data to the Register File. Data can be written 1-256 bytes at a time (256 bytes can be written by setting size to zero). If the Z8F640x family device is not in Debug mode, the address and data values are discarded. If the Read Protect Option Bit is enabled, then only writes to the Flash Control Registers are allowed and all other register write data values are discarded.

```
DBG <-- 08H
DBG <-- {4'h0,Register Address[11:8]}
DBG <-- Register Address[7:0]
DBG <-- Size[7:0]
DBG <-- 1-256 data bytes
```

- **Read Register (09H)**—The Read Register command reads data from the Register File. Data can be read 1-256 bytes at a time (256 bytes can be read by setting size to

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.

```
; 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 by users that prefer manual program coding or intend to implement their 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:

Table 113. 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:

Table 114. Assembly Language Syntax Example 2

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

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

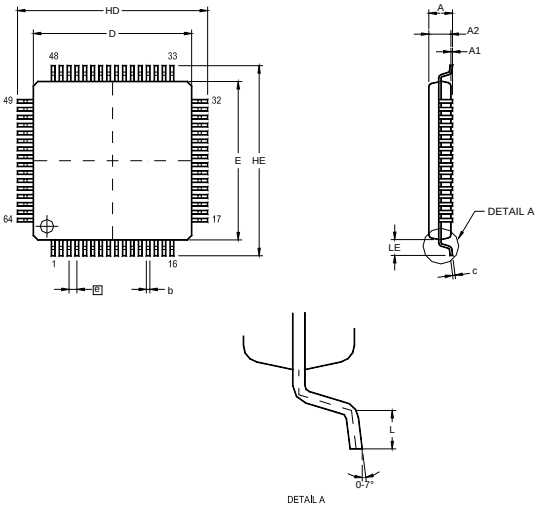
eZ8 CPU Instruction Notation

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 115

Table 126. 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
LD dst, rc	dst ← src	r	IM	0C-FC	-	-	-	-	-	-	2	2
		r	X(r)	C7							3	3
		X(r)	r	D7							3	4
		r	Ir	E3							2	3
		R	R	E4							3	2
		R	IR	E5							3	3
		R	IM	E6							3	3
		IR	IM	E7							3	3
		Ir	r	F3							2	3
		IR	R	F5							3	3
LDC dst, src	dst ← src	r	Irr	C2	-	-	-	-	-	-	2	5
		Ir	Irr	C5							2	9
		Irr	r	D2							2	5
LDCI dst, src	dst ← src r ← r + 1 rr ← rr + 1	Ir	Irr	C3	-	-	-	-	-	-	2	9
		Irr	Ir	D3							2	9
LDE dst, src	dst ← src	r	Irr	82	-	-	-	-	-	-	2	5
		Irr	r	92							2	5
LDEI dst, src	dst ← src r ← r + 1 rr ← rr + 1	Ir	Irr	83	-	-	-	-	-	-	2	9
		Irr	Ir	93							2	9
Flags Notation:	* = Value is a function of the result of the operation. - = Unaffected X = Undefined				0 = Reset to 0 1 = Set to 1							

Figure 105 illustrates the 64-pin LQFP (low-profile quad flat package) available for the Z8F1602, Z8F2402, Z8F3202, Z8F4802, and Z8F6402 devices.



SYMBOL	MILLIMETER		INCH	
	MIN	MAX	MIN	MAX
A	1.40	1.60	0.055	0.063
A1	0.05	0.15	0.002	0.006
A2	1.35	1.45	0.053	0.057
b	0.17	0.27	0.007	0.011
c	0.09	0.20	0.004	0.008
HD	11.75	12.25	0.463	0.482
D	9.90	10.10	0.390	0.398
HE	11.75	12.25	0.463	0.482
E	9.90	10.10	0.390	0.398
e	0.50 BSC		0.0197 BSC	
L	0.45	0.75	0.018	0.030
LE	1.00 REF		0.039 REF	

1. CONTROLLING DIMENSIONS : mm
2. MAX. COPLANARITY : 0.10mm / 0.004"

Figure 106. 64-Lead Low-Profile Quad Flat Package (LQFP)

Index

Symbols

185
% 185
@ 185

Numerics

10-bit ADC 4
40-lead plastic dual-inline package 206
44-lead low-profile quad flat package 207
44-lead plastic lead chip carrier package 207
64-lead low-profile quad flat package 208
68-lead plastic lead chip carrier package 209
80-lead quad flat package 210

A

absolute maximum ratings 167
AC characteristics 172
ADC 187
  architecture 132
  automatic power-down 133
  block diagram 133
  continuous conversion 134
  control register 135
  control register definitions 135
  data high byte register 137
  data low bits register 137
  DMA control 135
  electrical characteristics and timing 174
  operation 133
  single-shot conversion 133
ADCCTL register 135
ADCDH register 137
ADCDL register 137
ADCX 187
ADD 187
add - extended addressing 187
add with carry 187
add with carry - extended addressing 187
additional symbols 185
address space 17
ADDX 187
analog signals 14
analog-to-digital converter (ADC) 132
AND 190
ANDX 190
arithmetic instructions 187
assembly language programming 182
assembly language syntax 183

B

B 185
b 184
baud rate generator, UART 85
BCLR 188
binary number suffix 185
BIT 188
bit 184
  clear 188
  manipulation instructions 188
  set 188
  set or clear 188
  swap 188
  test and jump 190
  test and jump if non-zero 190
  test and jump if zero 190
bit jump and test if non-zero 190
bit swap 191
block diagram 3
block transfer instructions 188
BRK 190
BSET 188
BSWAP 188, 191
BTJ 190
BTJNZ 190
BTJZ 190

C

CALL procedure 190
capture mode 71
capture/compare mode 71

SDA and SCL (IrDA) signals 111
second opcode map after 1FH 205
serial clock 101
serial peripheral interface (SPI) 99
set carry flag 188, 189
set register pointer 189
shift right arithmetic 191
shift right logical 191
signal descriptions 13
single assertion (pulse) interrupt sources 47
single-shot conversion (ADC) 133
SIO 5
slave data transfer formats (I2C) 114
slave select 102
software trap 190
source operand 185
SP 185
SPI
  architecture 99
  baud rate generator 105
  baud rate high and low byte register 110
  clock phase 102
  configured as slave 100
  control register 107
  control register definitions 106
  data register 106
  error detection 105
  interrupts 105
  mode fault error 105
  mode register 109
  multi-master operation 104
  operation 100
  overrun error 105
  signals 101
  single master, multiple slave system 100
  single master, single slave system 99
  status register 108
  timing, PHASE = 0 103
  timing, PHASE=1 104
SPI controller signals 13
SPI mode (SPIMODE) 109
SPIBRH register 110
SPIBRL register 110
SPICTL register 107
SPIDATA register 106
SPIMODE register 109
SPISTAT register 108
SRA 191
src 185
SRL 191
SRP 189
SS, SPI signal 101
stack pointer 185
status register, I2C 118
STOP 189
stop mode 31, 189
stop mode recovery
  sources 29
  using a GPIO port pin transition 30
  using watch-dog timer time-out 29
SUB 188
subtract 188
subtract - extended addressing 188
subtract with carry 188
subtract with carry - extended addressing 188
SUBX 188
SWAP 191
swap nibbles 191
symbols, additional 185
system and short resets 26

T

TCM 188
TCMX 188
technical support 213
test complement under mask 188
test under mask 188
timer signals 14
timers 5, 57
  architecture 57
  block diagram 58
  capture mode 62, 71
  capture/compare mode 65, 71
  compare mode 63, 71
  continuous mode 59, 70
  counter mode 60
  counter modes 70

gated mode 64, 71
one-shot mode 58, 70
operating mode 58
PWM mode 61, 70
reading the timer count values 66
reload high and low byte registers 67
timer control register definitions 66
timer output signal operation 66
timers 0-3
control registers 70
high and low byte registers 66, 69
TM, TMX 188
tools, hardware and software 214
transmit
IrDA data 96
transmit interrupt 112
transmitting UART data-polled method 80
transmitting UART data-interrupt-driven method 81
TRAP 190

U

UART 4
architecture 78
asynchronous data format without/with parity 80
baud rate generator 85
baud rates table 93
control register definitions 86
controller signals 14
data format 79
interrupts 85
multiprocessor mode 84
receiving data using DMA controller 83
receiving data using interrupt-driven method 82
receiving data using the polled method 82
transmitting data using the interrupt-driven method 81
transmitting data using the polled method 80
x baud rate high and low registers 91
x control 0 and control 1 registers 89
x status 0 and status 1 registers 87
UxBRH register 91
UxBRL register 92
UxCTL0 register 89
UxCTL1 register 90
UxRXD register 87
UxSTAT0 register 87
UxSTAT1 register 89
UxTXD register 86

V

vector 184
voltage brown-out reset (VBR) 27

W

watch-dog timer
approximate time-out delays 72, 73
CNTL 28
control register 75
electrical characteristics and timing 174
interrupt in normal operation 73
interrupt in stop mode 73
operation 72
refresh 73, 189
reload unlock sequence 74
reload upper, high and low registers 76
reset 28
reset in normal operation 74
reset in stop mode 74
time-out response 73
WDTCTL register 75
WDTH register 76
WDTL register 77
working register 184
working register pair 184
WTDU register 76

X

X 184
XOR 190
XORX 190