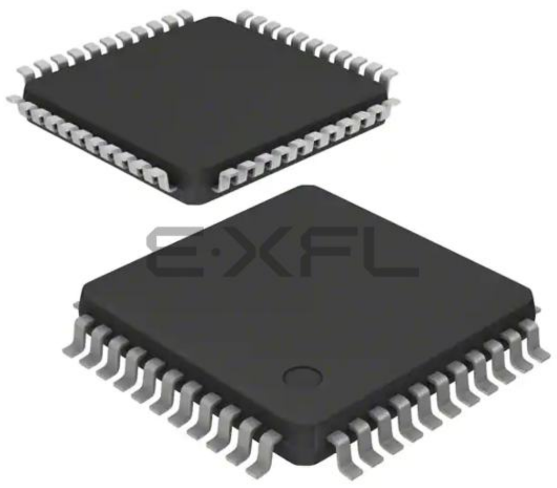


Welcome to E-XFL.COM

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	31
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 8x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 105°C (TA)
Mounting Type	Surface Mount
Package / Case	44-LQFP
Supplier Device Package	44-LQFP (10x10)
Purchase URL	https://www.e-xfl.com/product-detail/zilog/z8f2421an020eg

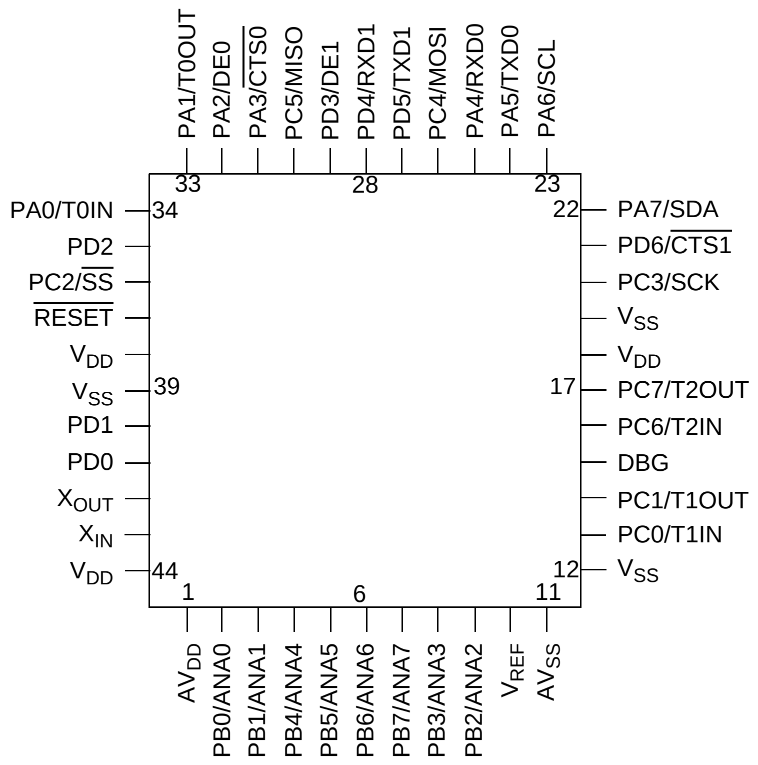


Figure 4. Z8 Encore! XP F64xx Series in 44-Pin Low-Profile Quad Flat Package (LQFP)

Note: Timer 3 is not available in the 44-pin LQFP package.

Port A–H Address Registers

The Port A–H Address registers, shown in Table 14, select the GPIO port functionality accessible through the Port A–H Control registers. The Port A–H Address and Control registers combine to provide access to all GPIO port control.

Table 14. 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							

Bit	**Description**
[7:0] PADDR	**Port Address** This port address selects one of the subregisters accessible through the Port A–H Control Registers. 00H = No function. Provides some protection against accidental port reconfiguration. 01H = Data Direction. 02H = Alternate Function. 03H = Output Control (Open-Drain). 04H = High Drive Enable. 05H = Stop Mode Recovery Source Enable. 06H–FFH = No function.

- Disable the timer
- Configure the timer for CONTINUOUS 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 (usually `0001H`), affecting only the first pass in CONTINUOUS Mode. After the first timer reload in CONTINUOUS 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 appropriate, 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 1 Register to enable the timer and initiate counting.

In CONTINUOUS Mode, the system clock always provides the timer input. The timer period is calculated using the following equation:

$$\text{CONTINUOUS Mode Time-Out Period (s)} = \frac{\text{Reload Value} \times \text{Prescale}}{\text{System Clock Frequency (Hz)}}$$

If an initial starting value other than `0001H` is loaded into the Timer High and Low Byte registers, the ONE-SHOT Mode equation must be used to determine the first time-out period.

COUNTER Mode

In COUNTER Mode, the timer counts input transitions from a GPIO port pin. The timer input is taken from the GPIO port pin timer input alternate function. The TPOL bit in the Timer Control 1 Register selects whether the count occurs on the rising edge or the falling edge of the timer input signal. In COUNTER Mode, the prescaler is disabled.

! **Caution:** The input frequency of the timer input signal must not exceed one-fourth the system clock frequency.

Upon reaching the reload value stored in the Timer Reload High and Low Byte registers, 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 ister. This operation allows accurate reads of the full 16-bit timer count value while enabled. When the timers are not enabled, a read from the Timer Low Byte Register returns the actual value in the counter.

Timer Output Signal Operation

A timer output is a GPIO port pin alternate function. Generally, the timer output is toggled every time the counter is reloaded.

Timer Control Register Definitions

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

Timer 0–3 High and Low Byte Registers: see page 72

Timer Reload High and Low Byte Registers: see page 74

Timer 0–3 PWM High and Low Byte Registers: see page 75

Timer 0–3 Control 0 Registers: see page 76

Timer 0–3 Control 1 Registers: see page 77

Timers 0–2 are available in all packages. Timer 3 is only available in 64-, 68- and 80-pin packages.

Timer 0–3 High and Low Byte Registers

The Timer 0–3 High and Low Byte (TxH and TxL) registers, shown in Tables 39 and 40, contain the current 16-bit timer count value. When the timer is enabled, a read from TxH causes the value in TxL to be stored in a temporary holding register. A read from TMRL always returns this temporary register when the timers are enabled. When the timer is disabled, reads from the TMRL read the register directly.

Writing to the Timer High and Low Byte registers while the timer is enabled is not recommended. There are no temporary holding registers available for write operations, so simultaneous 16-bit writes are not possible. If either the Timer High or Low Byte registers are written during counting, the 8-bit written value is placed in the counter (High or Low Byte) at the next clock edge. The counter continues counting from the new value.

Timer 3 is unavailable in 44-pin packages.

Operation

The SPI is a full-duplex, synchronous, character-oriented channel that supports a four-wire interface (serial clock, transmit, receive and Slave select). The SPI block consists of a transmit/receive shift register, a baud rate (clock) generator and a control unit.

During an SPI transfer, data is sent and received simultaneously by both the Master and the Slave SPI devices. Separate signals are required for data and the serial clock. When an SPI transfer occurs, a multibit (typically 8-bit) character is shifted out one data pin and an multibit character is simultaneously shifted in on a second data pin. An 8-bit shift register in the Master and another 8-bit shift register in the Slave are connected as a circular buffer. The SPI Shift Register is single-buffered in the transmit and receive directions. New data to be transmitted cannot be written into the shift register until the previous transmission is complete and receive data (if valid) has been read.

SPI Signals

The four basic SPI signals are:

- Master-In/Slave-Out
- Master-Out/Slave-In
- Serial Clock
- Slave Select

Each signal is described in both Master and Slave modes.

Master-In/Slave-Out

The Master-In/Slave-Out (MISO) pin is configured as an input in a Master device and as an output in a Slave device. It is one of the two lines that transfer serial data, with the most significant bit sent first. The MISO pin of a Slave device is placed in a high-impedance state if the Slave is not selected. When the SPI is not enabled, this signal is in a high-impedance state.

Master-Out/Slave-In

The Master-Out/Slave-In (MOSI) pin is configured as an output in a Master device and as an input in a Slave device. It is one of the two lines that transfer serial data, with the most significant bit sent first. When the SPI is not enabled, this signal is in a high-impedance state.

Bit	Description (Continued)
[5] COL	**Collision** 0 = A multimaster collision (mode fault) has not occurred. 1 = A multimaster collision (mode fault) has been detected.
[4] ABT	**Slave Mode Transaction Abort** This bit is set if the SPI is configured in slave mode, a transaction is occurring and $\overline{SS}$ deasserts before all bits of a character have been transferred as defined by the NUMBITS field of the SPIMODE Register. The IRQ bit also sets, indicating the transaction has completed. 0 = A slave mode transaction abort has not occurred. 1 = A slave mode transaction abort has been detected.
[3:2]	**Reserved** These bits are reserved and must be programmed to 00.
[1] TXST	**Transmit Status** 0 = No data transmission currently in progress. 1 = Data transmission currently in progress.
[0] SLAS	**Slave Select** If SPI enabled as a Slave, then the following conditions are true: 0 = $\overline{SS}$ input pin is asserted (Low). 1 = $\overline{SS}$ input is not asserted (High). If SPI enabled as a Master, this bit is not applicable.

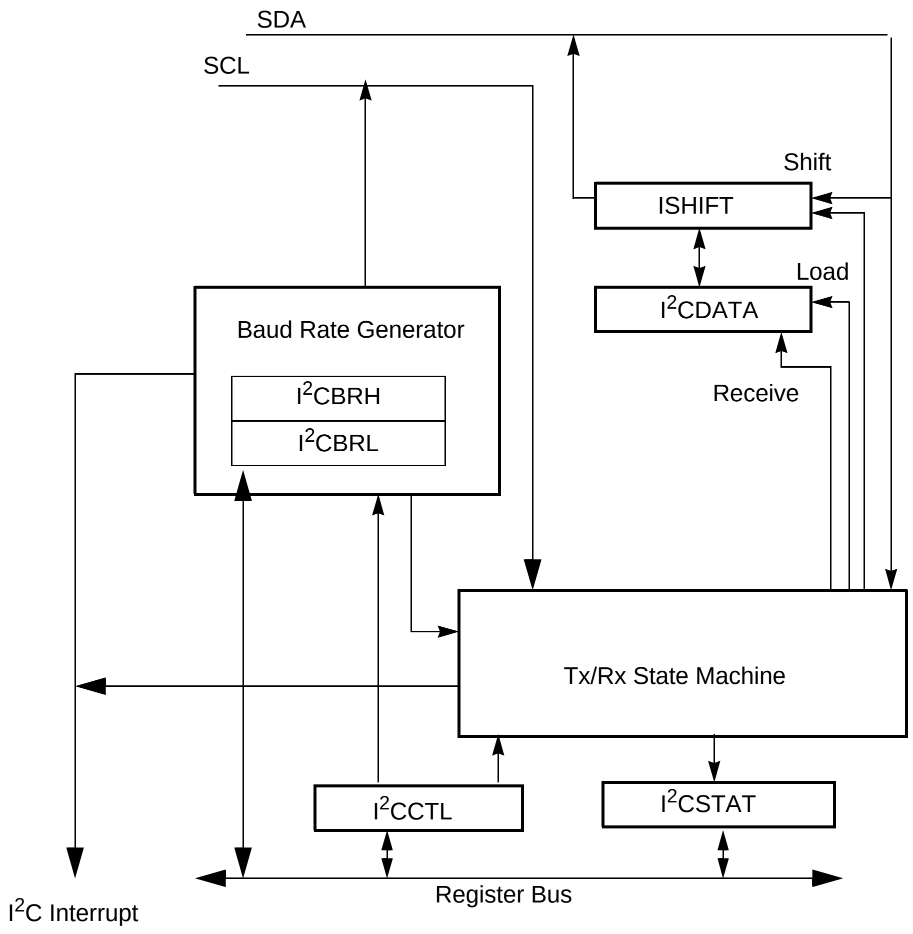


Figure 27. I^{2}C Controller Block Diagram

Operation

The I^{2}C Controller operates in MASTER Mode to transmit and receive data. Only a single master is supported. Arbitration between two masters must be accomplished in software. I^{2}C supports the following operations:

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

S	Slave Address	W = 0	A	Data	A	Data	A	Data	A/A	P/S

Figure 29. 7-Bit Addressed Slave Data Transfer Format

Observe the following procedure for a transmit operation to a 7-bit addressed slave:

1. Software asserts the IEN bit in the I^{2}C Control Register.
2. Software asserts the TXI bit of the I^{2}C Control Register to enable transmit interrupts.
3. The I^{2}C interrupt asserts, because the I^{2}C Data Register is empty
4. Software responds to the TDRE bit by writing a 7-bit slave address plus write bit (=0) to the I^{2}C Data Register.
5. Software asserts the start bit of the I^{2}C Control Register.
6. The I^{2}C Controller sends the start condition to the I^{2}C slave.
7. The I^{2}C Controller loads the I^{2}C Shift Register with the contents of the I^{2}C Data Register.
8. After one bit of address has been shifted out by the SDA signal, the transmit interrupt is asserted (TDRE = 1).
9. Software responds by writing the transmit data into the I^{2}C Data Register.
10. The I^{2}C Controller shifts the rest of the address and write bit out by the SDA signal.
11. If the I^{2}C slave sends an acknowledge (by pulling the SDA signal Low) during the next High period of SCL the I^{2}C Controller sets the ACK bit in the I^{2}C Status Register. Continue with Step 12.

    If the slave does not acknowledge, the Not Acknowledge interrupt occurs (NCKI bit is set in the Status Register, ACK bit is cleared). Software responds to the Not Acknowledge interrupt by setting the stop and flush bits and clearing the TXI bit. The I^{2}C Controller sends the stop condition on the bus and clears the stop and NCKI bits. The transaction is complete (ignore the following steps).
12. The I^{2}C Controller loads the contents of the I^{2}C Shift Register with the contents of the I^{2}C Data Register.
13. The I^{2}C Controller shifts the data out of using the SDA signal. After the first bit is sent, the transmit interrupt is asserted.
14. If more bytes remain to be sent, return to Step 9.
15. Software responds by setting the stop bit of the I^{2}C Control Register (or start bit to initiate a new transaction). In the stop case, software clears the TXI bit of the I^{2}C Control Register at the same time.

set in the Status Register, ACK bit is cleared). Software responds to the Not Acknowledge interrupt by setting the stop bit and clearing the TXI bit. The I^{2}C Controller sends the stop condition on the bus and clears the stop and NCKI bits. The transaction is complete (ignore the following steps).

7. The I^{2}C Controller shifts in the byte of data from the I^{2}C slave on the SDA signal. The I^{2}C Controller sends a Not Acknowledge to the I^{2}C slave if the NAK bit is set (last byte), else it sends an Acknowledge.
8. The I^{2}C Controller asserts the receive interrupt (RDRF bit set in the Status Register).
9. Software responds by reading the I^{2}C Data Register which clears the RDRF bit. If there is only one more byte to receive, set the NAK bit of the I^{2}C Control Register.
10. If there are more bytes to transfer, return to Step 7.
11. After the last byte is shifted in, a Not Acknowledge interrupt is generated by the I^{2}C Controller.
12. Software responds by setting the stop bit of the I^{2}C Control Register.
13. A stop condition is sent to the I^{2}C slave, the stop and NCKI bits are cleared.

Read Transaction with a 10-Bit Address

Figure 33 displays the read transaction format for a 10-bit addressed slave. The shaded regions indicate data transferred from the I^{2}C Controller to slaves and unshaded regions indicate data transferred from the slaves to the I^{2}C 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	A	P

Figure 33. Receive Data Format for a 10-Bit Addressed Slave

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

Observe the following procedure for the data transfer for a read operation to a 10-bit addressed slave:

1. Software writes `11110B` followed by the two address bits and a 0 (write) to the I^{2}C Data Register.
2. Software asserts the start and TXI bits of the I^{2}C Control Register.
3. The I^{2}C Controller sends the Start condition.
4. The I^{2}C Controller loads the I^{2}C Shift Register with the contents of the I^{2}C Data Register.

Table 71. I^{2}C Data Register (I2CDATA)

Bit	7	6	5	4	3	2	1	0
Field	DATA							
RESET	0							
R/W	R/W							
Address	F50H							

I^{2}C Status Register

The read-only I^{2}C Status Register, shown in Table 72, indicates the status of the I^{2}C Controller.

Table 72. I^{2}C Status Register (I2CSTAT)

Bit	7	6	5	4	3	2	1	0
Field	TDRE	RDRF	ACK	10B	RD	TAS	DSS	NCKI
RESET	1	0						
R/W	R							
Address	F51H							

Bit	Description
[7] TDRE	**Transmit Data Register Empty** When the I^{2}C Controller is enabled, this bit is 1 when the I^{2}C Data Register is empty. When this bit is set, an interrupt is generated if the TXI bit is set, except when the I^{2}C Controller is shifting in data during the reception of a byte or when shifting an address and the RD bit is set. This bit is cleared by writing to the I2CDATA Register.
[6] RDRF	**Receive Data Register Full** This bit is set = 1 when the I^{2}C Controller is enabled and the I^{2}C Controller has received a byte of data. When asserted, this bit causes the I^{2}C Controller to generate an interrupt. This bit is cleared by reading the I^{2}C Data Register (unless the read is performed using execution of the On-Chip Debugger’s Read Register command).

The write-only Flash Control Register shares its Register File address with the read-only Flash Status Register.

Table 93. Flash Control Register (FCTL)

Bit	7	6	5	4	3	2	1	0
Field	FCMD							
RESET	0							
R/W	W							
Address	FF8H							

Bit	Description
[7:0] FCMD	**Flash Command*** 73H = First unlock command. 8CH = Second unlock command. 95H = Page erase command. 63H = Mass erase command 5EH = Flash Sector Protect Register select.

Note: *All other commands, or any command out of sequence, lock the Flash Controller.

Table 95. Page Select Register (FPS)

Bit	7	6	5	4	3	2	1	0
Field	INFO_EN	PAGE						
RESET	0							
R/W	R/W							
Address	FF9H							

Bit	Description
[7] INFO_EN	**Information Area Enable** 0 = Information Area is not selected. 1 = Information Area is selected. The Information area is mapped into the Flash memory address space at addresses FE00H through FFFFH.
[6:0] PAGE	**Page Select** This 7-bit field selects the Flash memory page for programming and Page Erase operations. Flash Memory Address[15:9] = PAGE[6:0].

Flash Sector Protect Register

The Flash Sector Protect Register, shown in Table 96, protects Flash memory sectors from being programmed or erased from user code. The Flash Sector Protect Register shares its Register File address with the Page Select Register. The Flash Sector Protect Register can be accessed only after writing the Flash Control Register with `5EH`.

User code can only write bits in this register to 1 (bits cannot be cleared to 0 by user code). To determine the appropriate Flash memory sector address range and sector number for your Z8F64xx Series product, please refer to Table 91 on page 169.

Table 96. Flash Sector Protect Register (FPROT)

Bit	7	6	5	4	3	2	1	0
Field	SECT7	SECT6	SECT5	SECT4	SECT3	SECT2	SECT1	SECT0
RESET	0							
R/W	R/W*							
Address	FF9H							
Note: *R/W = This register is accessible for read operations; it can be written to 1 only via user code.								

Bit	Description
[7:0] SECT*n*	**Sector Protect**** 0 = Sector *n* can be programmed or erased from user code. 1 = Sector *n* is protected and cannot be programmed or erased from user code.
Note: **User code can only write bits from 0 to 1.	

If the OCD receives a serial break (nine or more continuous bits Low) the Autobaud Detector/Generator resets. The Autobaud Detector/Generator can then be reconfigured by sending `80H`.

OCD Serial Errors

The On-Chip Debugger can detect any of the following error conditions on the DBG pin:

- Serial break (a minimum of nine continuous bits Low)
- Framing error (received stop bit is Low)
- Transmit collision (OCD and host simultaneous transmission detected by the OCD)

When the OCD detects one of these errors, it aborts any command currently in progress, transmits a serial break 4096 system clock cycles long back to the host, and resets the Autobaud Detector/Generator. A framing error or transmit collision may be caused by the host sending a serial break to the OCD. Because of the open-drain nature of the interface, returning a serial break back to the host only extends the length of the serial break if the host releases the serial break early.

The host transmits a serial break on the DBG pin when first connecting to the Z8 Encore! XP F64xx Series devices or when recovering from an error. A serial break from the host resets the Autobaud Generator/Detector but does not reset the OCD Control Register. A serial break leaves the device in DEBUG Mode if that is the current mode. The OCD is held in Reset until the end of the serial break when the DBG pin returns High. Because of the open-drain nature of the DBG pin, the host can send a serial break to the OCD even if the OCD is transmitting a character.

Breakpoints

Execution breakpoints are generated using the BRK instruction (op code `00H`). When the eZ8 CPU decodes a BRK instruction, it signals the On-Chip Debugger. If breakpoints are enabled, the OCD idles the eZ8 CPU and enters DEBUG Mode. If breakpoints are not enabled, the OCD ignores the BRK signal and the BRK instruction operates as an NOP.

If breakpoints are enabled, the OCD can be configured to automatically enter DEBUG Mode, or to loop on the break instruction. If the OCD is configured to loop on the BRK instruction, then the CPU is still enabled to service DMA and interrupt requests.

The loop on BRK instruction can be used to service interrupts in the background. For interrupts to be serviced in the background, there cannot be any breakpoints in the interrupt service routine. Otherwise, the CPU stops on the breakpoint in the interrupt routine. For interrupts to be serviced in the background, interrupts must also be enabled. Debugging software should not automatically enable interrupts when using this feature, since

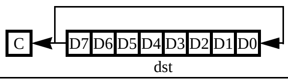
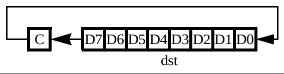
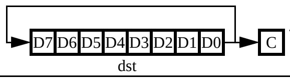
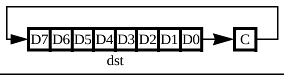
Condition Codes

The C, Z, S and V flags control the operation of the conditional jump (JP cc and JR cc) instructions. Sixteen frequently useful functions of the flag settings are encoded in a 4-bit field called the condition code (cc), which forms bits 7:4 of the conditional jump instructions. The condition codes are summarized in Table 127. Some binary condition codes can be created using more than one assembly code mnemonic. The result of the flag test operation decides if the conditional jump is executed.

Table 127. Condition Codes

Binary	Hex	Assembly Mnemonic	Definition	Flag Test Operation
0000	0	F	Always False	–
0001	1	LT	Less Than	(S XOR V) = 1
0010	2	LE	Less Than or Equal	(Z OR (S XOR V)) = 1
0011	3	ULE	Unsigned Less Than or Equal	(C OR Z) = 1
0100	4	OV	Overflow	V = 1
0101	5	MI	Minus	S = 1
0110	6	Z	Zero	Z = 1
0110	6	EQ	Equal	Z = 1
0111	7	C	Carry	C = 1
0111	7	ULT	Unsigned Less Than	C = 1
1000	8	T (or blank)	Always True	–
1001	9	GE	Greater Than or Equal	(S XOR V) = 0
1010	A	GT	Greater Than	(Z OR (S XOR V)) = 0
1011	B	UGT	Unsigned Greater Than	(C = 0 AND Z = 0) = 1
1100	C	NOV	No Overflow	V = 0
1101	D	PL	Plus	S = 0
1110	E	NZ	Non-Zero	Z = 0
1110	E	NE	Not Equal	Z = 0
1111	F	NC	No Carry	C = 0
1111	F	UGE	Unsigned Greater Than or Equal	C = 0

Table 136. eZ8 CPU Instruction Summary (Continued)

Assembly		Address Mode		Opcode(s)	Flags						Fetch	Instr.
Mnemonic	**Symbolic Operation**	**dst**	**src**	**(Hex)**	**C**	**Z**	**S**	**V**	**D**	**H**	**Cycles**	**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.

Hex Address: F11

Table 155. Timer 0–3 Low Byte Register (TxL)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	TL							
RESET	0							1
R/W	R/W							
Address	F01H, F09H, F11H, F19H							

Hex Address: F12

Table 156. Timer 0–3 Reload High Byte Register (TxRH)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	TRH							
RESET	1							
R/W	R/W							
Address	F02H, F0AH, F12H, F1AH							

Hex Address: F13

Table 157. Timer 0–3 Reload Low Byte Register (TxRL)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	TRL							
RESET	1							
R/W	R/W							
Address	F03H, F0BH, F13H, F1BH							

Hex Address: F14

Table 158. Timer 0–3 PWM High Byte Register (TxPWMH)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	PWMH							
RESET	0							
R/W	R/W							
Address	F04H, F0CH, F14H, F1CH							

Hex Address: FB4

Table 208. DMA*x* End Address Low Byte Register (DMA*x*END)

Bit	7	6	5	4	3	2	1	0
Field	DMA_END							
RESET	X							
R/W	R/W							
Address	FB4H, FBCH							

Hex Addresses: FB5–FB7

This address range is reserved.

Hex Address: FB8

Table 209. DMA*x* Control Register (DMA*x*CTL)

Bit	7	6	5	4	3	2	1	0
Field	DEN	DLE	DDIR	IRQEN	WSEL	RSS		
RESET	0							
R/W	R/W							
Address	FB0H, FB8H							

Hex Address: FB9

Table 210. DMA*x* I/O Address Register (DMA*x*IO)

Bit	7	6	5	4	3	2	1	0
Field	DMA_IO							
RESET	X							
R/W	R/W							
Address	FB1H, FB9H							

General-Purpose Input/Output (GPIO)

For more information about these GPIO Control registers, see the GPIO Control Register Definitions section on page 39.

Hex Address: FD0

Table 229. 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: FD1

Table 230. 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: FD2

Table 231. 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							