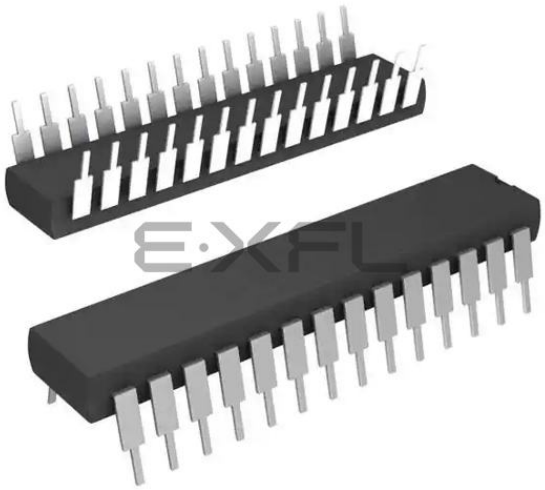

Welcome to E-XFL.COM

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	IrDA, UART/USART
Peripherals	Brown-out Detect/Reset, LED, LVD, POR, PWM, WDT
Number of I/O	25
Program Memory Size	4KB (4K x 8)
Program Memory Type	FLASH
EEPROM Size	128 x 8
RAM Size	1K x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 3.6V
Data Converters	-
Oscillator Type	Internal
Operating Temperature	0°C ~ 70°C (TA)
Mounting Type	Through Hole
Package / Case	28-DIP (0.600", 15.24mm)
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/zilog/z8f041apj020sg

Internal Precision Oscillator

The internal precision oscillator (IPO) is a trimmable clock source that requires no external components.

Temperature Sensor

The optional temperature sensor produces an analog output proportional to the device temperature. This signal can be sent to either the ADC or the analog comparator.

Analog Comparator

The analog comparator compares the signal at an input pin with either an internal programmable voltage reference or a second input pin. The comparator output can be used to drive either an output pin or to generate an interrupt.

External Crystal Oscillator

The crystal oscillator circuit provides highly accurate clock frequencies with the use of an external crystal, ceramic resonator or RC network.

Low Voltage Detector

The low voltage detector (LVD) is able to generate an interrupt when the supply voltage drops below a user-programmable level. The LVD is available on 8-pin devices only.

On-Chip Debugger

The Z8 Encore! XP F082A Series products feature an integrated on-chip debugger (OCD) accessed via a single-pin interface. The OCD provides a rich-set of debugging capabilities, such as reading and writing registers, programming Flash memory, setting breakpoints and executing code.

Universal Asynchronous Receiver/Transmitter

The full-duplex universal asynchronous receiver/transmitter (UART) is included in all Z8 Encore! XP package types. The UART supports 8- and 9-bit data modes and selectable parity. The UART also supports multi-drop address processing in hardware. The UART baud rate generator (BRG) can be configured and used as a basic 16-bit timer.

Timers

Two enhanced 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

Note: Asserting any power control bit disables the targeted block regardless of any enable bits contained in the target block's control registers.

Architecture

Figure 7 displays a simplified block diagram of a GPIO port pin. In this figure, the ability to accommodate alternate functions and variable port current drive strength is not displayed.

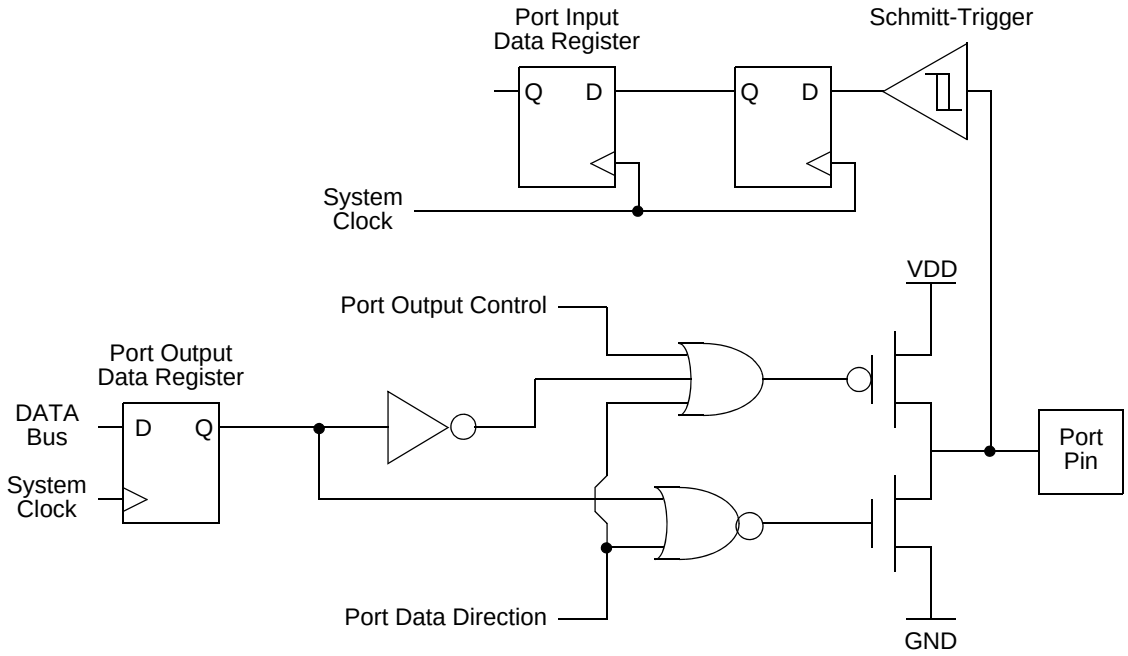

Figure 7. GPIO Port Pin Block Diagram

GPIO Alternate Functions

Many of the GPIO port pins can be used for general-purpose I/O and access to on-chip peripheral functions such as the timers and serial communication devices. The Port A–D Alternate Function subregisters configure these pins for either General-Purpose I/O or alternate function operation. When a pin is configured for alternate function, control of the port pin direction (input/output) is passed from the Port A–D Data Direction registers to the alternate function assigned to this pin. Table 15 on page 40 lists the alternate functions possible with each port pin. For those pins with more one alternate function, the alternate function is defined through Alternate Function Sets subregisters AFS1 and AFS2.

The crystal oscillator functionality is not controlled by the GPIO block. When the crystal oscillator is enabled in the oscillator control block, the GPIO functionality of PA0 and PA1 is overridden. In that case, those pins function as input and output for the crystal oscillator.

- 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). This action only affects 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. Enable the timer interrupt (if appropriate) and set the timer interrupt priority by writing to the relevant interrupt registers.
5. Configure the associated GPIO port pin (if using the Timer Output function) for the Timer Output alternate function.
6. Write to the Timer Control Register to enable the timer and initiate counting.

In CONTINUOUS Mode, the system clock always provides the timer input. The timer period is computed via 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, use the ONE-SHOT Mode equation 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 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. Further, the high or low state of the input signal pulse must be no less than twice the system clock period. A shorter pulse may not be captured.

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

The PWM period is represented by the following equation:

$$\text{PWM Period (s)} = \frac{\text{Reload Value} \times \text{Prescale}}{\text{System Clock Frequency (Hz)}}$$

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

If TPOL is set to 0, the ratio of the PWM output High time to the total period is represented by:

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

If TPOL is set to 1, the ratio of the PWM output High time to the total period is represented by:

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

PWM DUAL OUTPUT Mode

In PWM DUAL OUTPUT Mode, the timer outputs a Pulse-Width Modulated (PWM) output signal pair (basic PWM signal and its complement) through two GPIO port pins. The timer input is the system clock. The timer first counts up to the 16-bit PWM match value stored in the Timer PWM High and Low Byte registers. When the timer count value matches the PWM value, the Timer Output toggles. The timer continues counting until it reaches the reload value stored in the Timer Reload High and Low Byte registers. 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.

If the TPOL bit in the Timer Control Register is set to 1, the Timer Output signal begins as a High (1) and transitions to a Low (0) when the timer value matches the PWM value. The Timer Output signal returns to a High (1) after the timer reaches the reload value and is reset to 0001H.

If the TPOL bit in the Timer Control Register is set to 0, the Timer Output signal begins as a Low (0) and transitions to a High (1) when the timer value matches the PWM value. The Timer Output signal returns to a Low (0) after the timer reaches the reload value and is reset to 0001H.

The timer also generates a second PWM output signal Timer Output Complement. The Timer Output Complement is the complement of the Timer Output PWM signal. A programmable deadband delay can be configured to time delay (0 to 128 system clock cycles) PWM output transitions on these two pins from a low to a high (inactive to active). This

WDT Reset in Normal Operation

If configured to generate a Reset when a time-out occurs, the Watchdog Timer forces the device into the System Reset state. The WDT status bit in the Reset Status (RSTSTAT) Register is set to 1. For more information about system reset, see the Reset, Stop Mode Recovery and Low Voltage Detection chapter on page 22.

WDT Reset in STOP Mode

If configured to generate a Reset when a time-out occurs and the device is in STOP Mode, the Watchdog Timer initiates a Stop Mode Recovery. Both the WDT status bit and the STOP bit in the Reset Status (RSTSTAT) Register are set to 1 following WDT time-out in STOP Mode.

Watchdog Timer Reload Unlock Sequence

Writing the unlock sequence to the Watchdog Timer (WDTCTL) Control Register address unlocks the three Watchdog Timer Reload Byte registers (WDTU, WDTH and WDTL) to allow changes to the time-out period. These write operations to the WDTCTL Register address produce no effect on the bits in the WDTCTL Register. The locking mechanism prevents spurious writes to the Reload registers. Observe the following steps to unlock the Watchdog Timer Reload Byte registers (WDTU, WDTH and WDTL) for write access.

1. Write 55H to the Watchdog Timer Control Register (WDTCTL).
2. Write AAH to the Watchdog Timer Control Register (WDTCTL).
3. Write the Watchdog Timer Reload Upper Byte Register (WDTU) with the appropriate time-out value.
4. Write the Watchdog Timer Reload High Byte Register (WDTH) with the appropriate time-out value.
5. Write the Watchdog Timer Reload Low Byte Register (WDTL) with the appropriate time-out value.

All three Watchdog Timer Reload registers must be written in the order just listed. There must be no other register writes between each of these operations. If a register write occurs, the lock state machine resets and no further writes can occur unless the sequence is restarted. The value in the Watchdog Timer Reload registers is loaded into the counter when the Watchdog Timer is first enabled and every time a WDT instruction is executed.

Watchdog Timer Calibration

Due to its extremely low operating current, the Watchdog Timer oscillator is somewhat inaccurate. This variation can be corrected using the calibration data stored in the Flash Information Page; see Tables 100 and 101 on page 173 for details. Loading these values

Table 64. UART Control 1 Register (U0CTL1)

Bit	7	6	5	4	3	2	1	0
Field	MPMD[1]	MPEN	MPMD[0]	MPBT	DEPOL	BRGCTL	RDAIRQ	IREN
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
Address	F43H							

Bit	Description
[7,5] MPMD[1,0]	**MULTIPROCESSOR Mode** If MULTIPROCESSOR (9-bit) Mode is enabled: 00 = The UART generates an interrupt request on all received bytes (data and address). 01 = The UART generates an interrupt request only on received address bytes. 10 = The UART generates an interrupt request when a received address byte matches the value stored in the Address Compare Register and on all successive data bytes until an address mismatch occurs. 11 = The UART generates an interrupt request on all received data bytes for which the most recent address byte matched the value in the Address Compare Register.
[6] MPEN	**MULTIPROCESSOR (9-bit) Enable** This bit is used to enable MULTIPROCESSOR (9-bit) Mode. 0 = Disable MULTIPROCESSOR (9-bit) Mode. 1 = Enable MULTIPROCESSOR (9-bit) Mode.
[4] MPBT	**Multiprocessor Bit Transmit** This bit is applicable only when MULTIPROCESSOR (9-bit) Mode is enabled. The 9th bit is used by the receiving device to determine if the data byte contains address or data information. 0 = Send a 0 in the multiprocessor bit location of the data stream (data byte). 1 = Send a 1 in the multiprocessor bit location of the data stream (address byte).
[3] DEPOL	**Driver Enable Polarity** 0 = DE signal is Active High. 1 = DE signal is Active Low.

Table 69. UART Address Compare Register (U0ADDR)

Bit	7	6	5	4	3	2	1	0
Field	COMP_ADDR							
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
Address	F45H							

Bit	Description
[7:0] COMP_ADDR	**Compare Address** This 8-bit value is compared to incoming address bytes.

UART Baud Rate High and Low Byte Registers

The UART Baud Rate High (UxBRH) and Low Byte (UxBRL) registers, shown in Tables 70 and 71, 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 70. UART Baud Rate High Byte Register (U0BRH)

Bit	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
Address	F46H							

Bit	Description
[7:0] BRH	**UART Baud Rate High Byte**

Table 71. UART Baud Rate Low Byte Register (U0BRL)

Bit	7	6	5	4	3	2	1	0
Field	BRL							
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
Address	F47H							

Bit	Description
[7:0] BRL	**UART Baud Rate Low Byte**

Infrared Encoder/Decoder

Z8 Encore! XP F082A Series products contain a fully-functional, high-performance UART to Infrared Encoder/Decoder (endec). The infrared endec is integrated with an on-chip UART to allow easy communication between the Z8 Encore! XP MCU and IrDA Physical Layer Specification, Version 1.3-compliant infrared transceivers. Infrared communication provides secure, reliable, low-cost, point-to-point communication between PCs, PDAs, cell phones, printers and other infrared enabled devices.

Architecture

Figure 16 displays the architecture of the infrared endec.

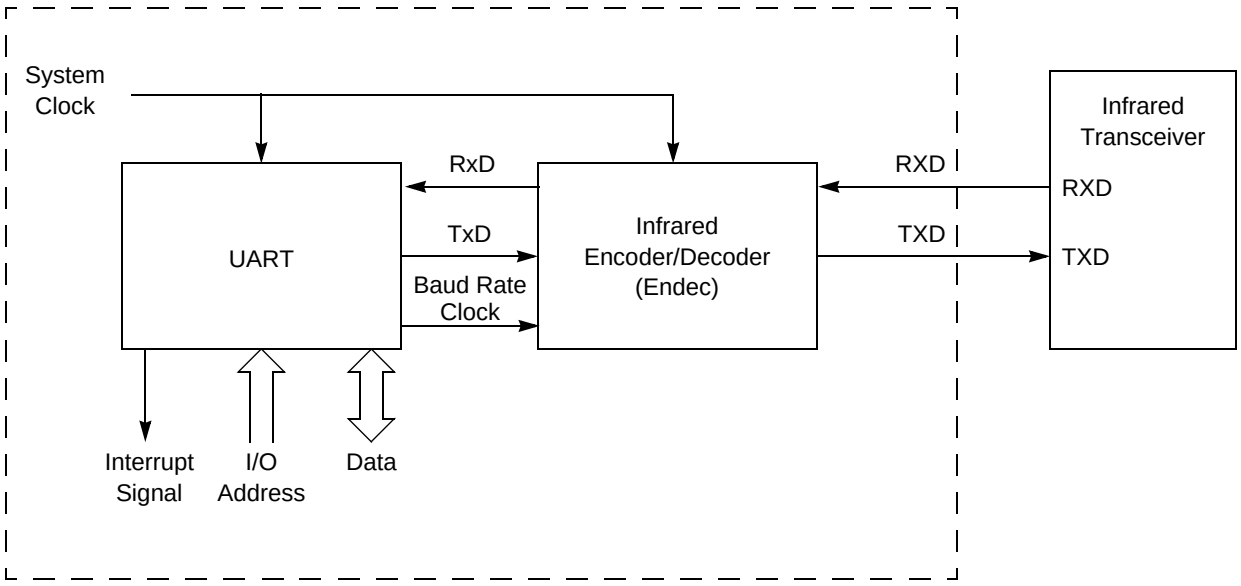

Figure 16. Infrared Data Communication System Block Diagram

Operation

When the infrared endec is enabled, the transmit data from the associated on-chip UART is encoded as digital signals in accordance with the IrDA standard and output to the infrared transceiver through the TXD pin. Likewise, data received from the infrared transceiver is passed to the infrared endec through the RXD pin, decoded by the infrared endec and passed to the UART. Communication is half-duplex, which means simultaneous data transmission and reception is not allowed.

- Write to BUFMODE[2:0] to select SINGLE-ENDED or DIFFERENTIAL mode, plus unbuffered or buffered mode.
- Write the REFSELH bit of the pair {REFSELH, REFSELL} to select the internal voltage reference level or to disable the internal reference. The REFSELL bit is contained in the ADC Control Register 0.

3. Write to the ADC Control Register 0 to configure the ADC for continuous conversion. The bit fields in the ADC Control Register may be written simultaneously:
   - Write to the ANAIN[3:0] field to select from the available analog input sources (different input pins available depending on the device).
   - Set CONT to 1 to select continuous conversion.
   - If the internal VREF must be output to a pin, set the REFEXT bit to 1. The internal voltage reference must be enabled in this case.
   - Write the REFSELL bit of the pair {REFSELH, REFSELL} to select the internal voltage reference level or to disable the internal reference. The REFSELH bit is contained in ADC Control/Status Register 1.
   - Set CEN to 1 to start the conversions.
4. When the first conversion in continuous operation is complete (after 5129 system clock cycles, plus the 40 cycles for power-up, if necessary), the ADC control logic performs the following operations:
   - CEN resets to 0 to indicate the first conversion is complete. CEN remains 0 for all subsequent conversions in continuous operation
   - An interrupt request is sent to the Interrupt Controller to indicate the conversion is complete
5. The ADC writes a new data result every 256 system clock cycles. For each completed conversion, the ADC control logic performs the following operations:
   - Writes the 13-bit two's complement result to {ADCD_H[7:0], ADCD_L[7:3]}
   - Sends an interrupt request to the Interrupt Controller denoting conversion complete
6. To disable continuous conversion, clear the CONT bit in the ADC Control Register to 0.

Interrupts

The ADC is able to interrupt the CPU when a conversion has been completed. When the ADC is disabled, no new interrupts are asserted; however, an interrupt pending when the ADC is disabled is not cleared.

Bit	Description (Continued)
[2:1]	**Reserved** These bits are reserved and must be undefined.
[0] OVF	**Overflow Status** 0 = A hardware overflow did not occur in the ADC for the current sample. 1= A hardware overflow did occur in the ADC for the current sample, therefore the current sample is invalid.

Comparator

The Z8 Encore! XP F082A Series devices feature a general purpose comparator that compares two analog input signals. These analog signals may be external stimulus from a pin (CINP and/or CINN) or internally generated signals. Both a programmable voltage reference and the temperature sensor output voltage are available internally. The output is available as an interrupt source or can be routed to an external pin.

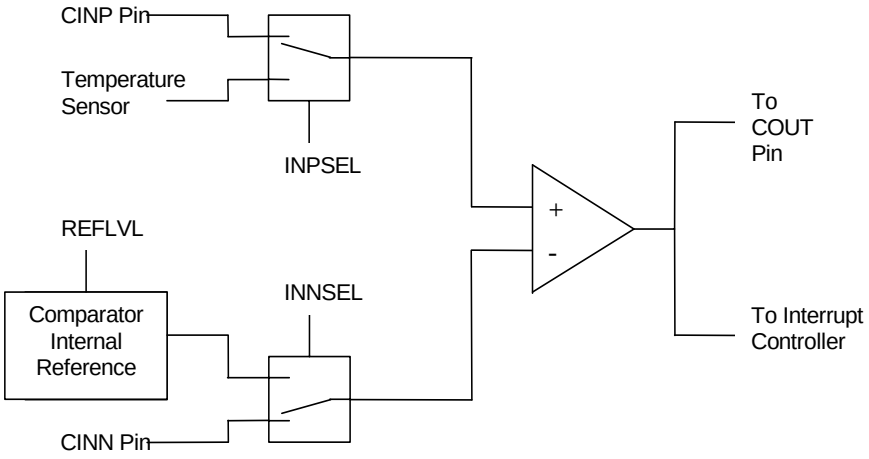

Figure 20. Comparator Block Diagram

Operation

When the positive comparator input exceeds the negative input by more than the specified hysteresis, the output is a logic High. When the negative input exceeds the positive by more than the hysteresis, the output is a logic Low. Otherwise, the comparator output retains its present value. See Table 141 on page 238 for details.

The comparator may be powered down to reduce supply current. See the Power Control Register 0 section on page 33 for details.

! Caution: Because of the propagation delay of the comparator, Zilog does not recommend enabling or reconfiguring the comparator without first disabling the interrupts and waiting for the comparator output to settle. Doing so can result in spurious interrupts.

Byte Write

To write a byte to the NVDS array, the user code must first push the address, then the data byte onto the stack. The user code issues a CALL instruction to the address of the byte-write routine (0x10B3). At the return from the sub-routine, the write status byte resides in working register R0. The bit fields of this status byte are defined in Table 106. The contents of the status byte are undefined for write operations to illegal addresses. Also, user code must pop the address and data bytes off the stack.

The write routine uses 13 bytes of stack space in addition to the two bytes of address and data pushed by the user. Sufficient memory must be available for this stack usage.

Because of the Flash memory architecture, NVDS writes exhibit a nonuniform execution time. In general, a write takes 251µs (assuming a 20MHz system clock). Every 400 to 500 writes, however, a maintenance operation is necessary. In this rare occurrence, the write takes up to 61ms to complete. Slower system clock speeds result in proportionally higher execution times.

NVDS byte writes to invalid addresses (those exceeding the NVDS array size) have no effect. Illegal write operations have a 2µs execution time.

Table 106. Write Status Byte

Bit	7	6	5	4	3	2	1	0
Field	Reserved				RCPY	PF	AWE	DWE
Default Value	0	0	0	0	0	0	0	0

Bit	Description
[7:4]	**Reserved** These bits are reserved and must be programmed to 0000.
[3] RCPY	**Recopy Subroutine Executed** A recopy subroutine was executed. These operations take significantly longer than a normal write operation.
[2] PF	**Power Failure Indicator** A power failure or system reset occurred during the most recent attempted write to the NVDS array.
[1] AWE	**Address Write Error** An address byte failure occurred during the most recent attempted write to the NVDS array.
[0] DWE	**Data Write Error** A data byte failure occurred during the most recent attempted write to the NVDS array.

```
DBG ← Size[15:8]
DBG ← Size[7:0]
DBG ← 1-65536 data bytes
```

Read Data Memory (0DH). The Read Data Memory command reads from Data Memory. This command is equivalent to the LDE and LDEI instructions. Data can be read 1 to 65536 bytes at a time (65536 bytes can be read by setting size to 0). If the device is not in DEBUG Mode, this command returns FFH for the data.

```
DBG ← 0DH
DBG ← Data Memory Address[15:8]
DBG ← Data Memory Address[7:0]
DBG ← Size[15:8]
DBG ← Size[7:0]
DBG → 1-65536 data bytes
```

Read Program Memory CRC (0EH). The Read Program Memory CRC command computes and returns the Cyclic Redundancy Check (CRC) of Program Memory using the 16-bit CRC-CCITT polynomial. If the device is not in DEBUG Mode, this command returns FFFFH for the CRC value. Unlike most other OCD Read commands, there is a delay from issuing of the command until the OCD returns the data. The OCD reads the Program Memory, calculates the CRC value and returns the result. The delay is a function of the Program Memory size and is approximately equal to the system clock period multiplied by the number of bytes in the Program Memory.

```
DBG ← 0EH
DBG → CRC[15:8]
DBG → CRC[7:0]
```

Step Instruction (10H). The Step Instruction command steps one assembly instruction at the current Program Counter (PC) location. If the device is not in DEBUG Mode or the Flash Read Protect Option bit is enabled, the OCD ignores this command.

```
DBG ← 10H
```

Stuff Instruction (11H). The Stuff Instruction command steps one assembly instruction and allows specification of the first byte of the instruction. The remaining 0-4 bytes of the instruction are read from Program Memory. This command is useful for stepping over instructions where the first byte of the instruction has been overwritten by a Breakpoint. If the device is not in DEBUG Mode or the Flash Read Protect Option bit is enabled, the OCD ignores this command.

```
DBG ← 11H
DBG ← opcode[7:0]
```

Execute Instruction (12H). The Execute Instruction command allows sending an entire instruction to be executed to the eZ8 CPU. This command can also step over Breakpoints. The number of bytes to send for the instruction depends on the opcode. If the device is not

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 has the operands in the order 'source, destination', but ordering is opcode-dependent. The following instruction examples illustrate the format of some basic assembly instructions and the resulting object code produced by the assembler. This binary format must be followed if manual program coding is preferred or if you intend to implement your own assembler.

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

Table 116. 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 117. Assembly Language Syntax Example 2

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

Table 128. 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
AND dst, src	dst ← dst AND src	r	r	52	–	*	*	0	–	–	2	3
		r	Ir	53							2	4
		R	R	54							3	3
		R	IR	55							3	4
		R	IM	56							3	3
		IR	IM	57							3	4
ANDX dst, src	dst ← dst AND src	ER	ER	58	–	*	*	0	–	–	4	3
		ER	IM	59							4	3
ATM	Block all interrupt and DMA requests during execution of the next 3 instructions			2F	–	–	–	–	–	–	1	2
BCLR bit, dst	dst[bit] ← 0	r		E2	–	–	–	–	–	–	2	2
BIT p, bit, dst	dst[bit] ← p	r		E2	–	–	–	–	–	–	2	2
BRK	Debugger Break			00	–	–	–	–	–	–	1	1
BSET bit, dst	dst[bit] ← 1	r		E2	–	–	–	–	–	–	2	2
BSWAP dst	dst[7:0] ← dst[0:7]	R		D5	X	*	*	0	–	–	2	2
BTJ p, bit, src, dst	if src[bit] = p PC ← PC + X		r	F6	–	–	–	–	–	–	3	3
			Ir	F7							3	4
BTJNZ bit, src, dst	if src[bit] = 1 PC ← PC + X		r	F6	–	–	–	–	–	–	3	3
			Ir	F7							3	4
BTJZ bit, src, dst	if src[bit] = 0 PC ← PC + X		r	F6	–	–	–	–	–	–	3	3
			Ir	F7							3	4

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

General Purpose I/O Port Output Timing

Figure 35 and Table 144 provide timing information for GPIO port pins.

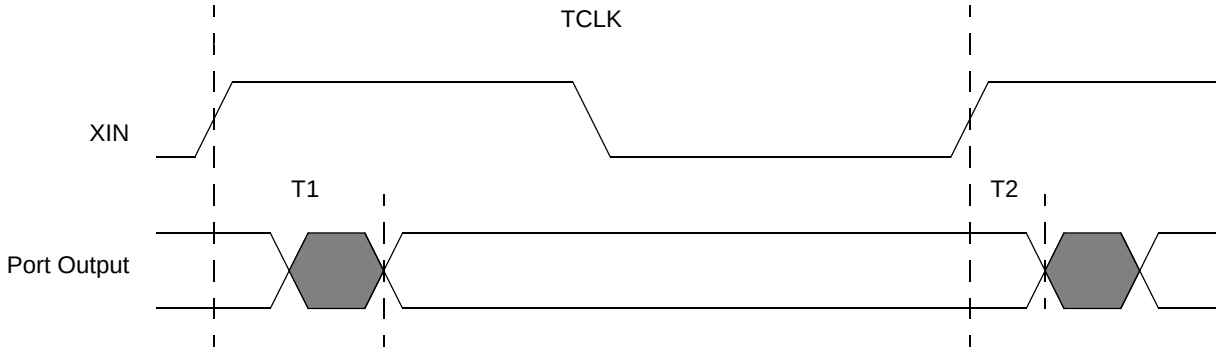

Figure 35. GPIO Port Output Timing

Table 144. GPIO Port Output Timing

		Delay (ns)	
Parameter	**Abbreviation**	**Minimum**	**Maximum**
GPIO port pins			
T_1	X_{IN} Rise to Port Output Valid Delay	–	15
T_2	X_{IN} Rise to Port Output Hold Time	2	–

D

E

F

U

V

W

X