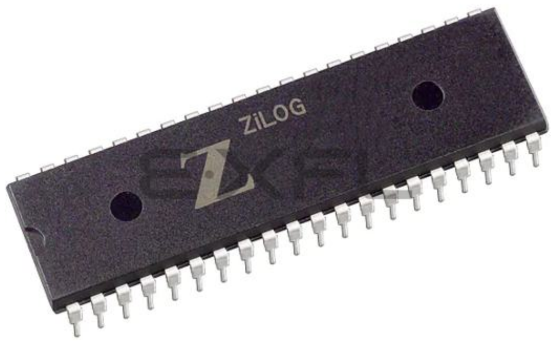

Details	
Product Status	Obsolete
Core Processor	eZ8
Core Size	8-Bit
Speed	20MHz
Connectivity	I²C, IrDA, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, DMA, POR, PWM, WDT
Number of I/O	31
Program Memory Size	32KB (32K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	2K x 8
Voltage - Supply (Vcc/Vdd)	3V ~ 3.6V
Data Converters	A/D 8x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 105°C (TA)
Mounting Type	Through Hole
Package / Case	40-DIP (0.620", 15.75mm)
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/zilog/z8f3201pm020ec

Address Space

Overview

The eZ8 CPU can access three distinct address spaces:

- The Register File contains addresses for the general-purpose registers and the eZ8 CPU, peripheral, and general-purpose I/O port control registers.
- The Program Memory contains addresses for all memory locations having executable code and/or data.
- The Data Memory contains addresses for all memory locations that hold data only.

These three address spaces are covered briefly in the following subsections. For more detailed information regarding the eZ8 CPU and its address space, refer to the *eZ8 CPU User Manual* available for download at www.zilog.com.

Register File

The Register File address space in the Z8 Encore!® is 4KB (4096 bytes). The Register File is composed of two sections—control registers and general-purpose registers. When instructions are executed, registers are read from when defined as sources and written to when defined as destinations. The architecture of the eZ8 CPU allows all general-purpose registers to function as accumulators, address pointers, index registers, stack areas, or scratch pad memory.

The upper 256 bytes of the 4KB Register File address space are reserved for control of the eZ8 CPU, the on-chip peripherals, and the I/O ports. These registers are located at addresses from `F00H` to `FFFH`. Some of the addresses within the 256-byte control register section are reserved (unavailable). Reading from an reserved Register File addresses returns an undefined value. Writing to reserved Register File addresses is not recommended and can produce unpredictable results.

The on-chip RAM always begins at address 000H in the Register File address space. The Z8F640x family products contain 2KB to 4KB of on-chip RAM depending upon the device. Reading from Register File addresses outside the available RAM addresses (and not within in the control register address space) returns an undefined value. Writing to these Register File addresses produces no effect. Refer to the **Part Selection Guide** section of the **Introduction** chapter to determine the amount of RAM available for the specific Z8F640x family device.

Table 6. Register File Address Map (Continued)

Address (Hex)	Register Description	Mnemonic	Reset (Hex)	Page #
FED	Port H Control	PHCTL	00	38
FEE	Port H Input Data	PHIN	XX	42
FEF	Port H Output Data	PHOUT	00	43
Watch-Dog Timer (WDT)				
FF0	Watch-Dog Timer Control	WDTCTL	XXX00000b	75
FF1	Watch-Dog Timer Reload Upper Byte	WDTU	FF	76
FF2	Watch-Dog Timer Reload High Byte	WDTH	FF	76
FF3	Watch-Dog Timer Reload Low Byte	WDTL	FF	76
FF4--FF7	Reserved	—	XX	
Flash Memory Controller				
FF8	Flash Control	FCTL	00	144
FF8	Flash Status	FSTAT	00	145
FF9	Flash Page Select	FPS	00	146
FFA	Flash Programming Frequency High Byte	FFREQH	00	147
FFB	Flash Programming Frequency Low Byte	FFREQL	00	147
eZ8 CPU				
FFC	Flags	—	XX	Refer to the *eZ8 CPU User Manual*
FFD	Register Pointer	RP	XX	
FFE	Stack Pointer High Byte	SPH	XX	
FFF	Stack Pointer Low Byte	SPL	XX	
XX=Undefined				

External Pin Reset

The $\overline{\text{RESET}}$ pin has a Schmitt-triggered input and an internal pull-up. Once the $\overline{\text{RESET}}$ pin is asserted, the device progresses through the Short Reset sequence. While the $\overline{\text{RESET}}$ input pin is asserted Low, the Z8F640x family device continues to be held in the Reset state. If the $\overline{\text{RESET}}$ pin is held Low beyond the Short Reset time-out, the device exits the Reset state immediately following $\overline{\text{RESET}}$ pin deassertion. Following a Short Reset initiated by the external $\overline{\text{RESET}}$ pin, the `EXT` status bit in the Watch-Dog Timer Control (WDTCTL) register is set to 1.

Stop Mode Recovery

Stop mode is entered by execution of a `STOP` instruction by the eZ8 CPU. Refer to the **Low-Power Modes** chapter for detailed Stop mode information. During Stop Mode Recovery, the Z8F640x family device is held in reset for 514 cycles of the Watch-Dog Timer oscillator followed by 16 cycles of the system clock (crystal oscillator). Stop Mode Recovery does not affect any values in the Register File, including the Stack Pointer, Register Pointer, Flags and general-purpose RAM.

The eZ8 CPU fetches the Reset vector at Program Memory addresses `0002H` and `0003H` and loads that value into the Program Counter. Program execution begins at the Reset vector address. Following Stop Mode Recovery, the `STOP` bit in the Watch-Dog Timer Control Register is set to 1. Table 9 lists the Stop Mode Recovery sources and resulting actions. The text following provides more detailed information on each of the Stop Mode Recovery sources.

Table 9. Stop Mode Recovery Sources and Resulting Action

Operating Mode	Stop Mode Recovery Source	Action
Stop mode	Watch-Dog Timer time-out when configured for Reset	Stop Mode Recovery
	Watch-Dog Timer time-out when configured for interrupt	Stop Mode Recovery followed by interrupt (if interrupts are enabled)
	Data transition on any GPIO Port pin enabled as a Stop Mode Recovery source	Stop Mode Recovery

Stop Mode Recovery Using Watch-Dog Timer Time-Out

If the Watch-Dog Timer times out during Stop mode, the Z8F640x family device undergoes a STOP Mode Recovery sequence. In the Watch-Dog Timer Control register, the `WDT` and STOP bits are set to 1. If the Watch-Dog Timer is configured to generate an interrupt upon time-out and the device is configured to respond to interrupts, the Z8F640x family device services the Watch-Dog Timer interrupt request following the normal Stop Mode Recovery sequence.

Timer 0-3 Control Registers

The Timer 0-3 Control (TxCTL) registers enable/disable the timers, set the prescaler value, and determine the timer operating mode.

Table 44. Timer 0-3 Control Register (TxCTL)

BITS	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FIELD	TEN	TPOL	PRES			TMODE		
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	F07H, F0FH, F17H, F1FH							

TEN—Timer Enable
0 = Timer is disabled.
1 = Timer enabled to count.

TPOL—Timer Input/Output Polarity
Operation of this bit is a function of the current operating mode of the timer.

One-Shot mode
When the timer is disabled, the Timer Output signal is set to the value of this bit. When the timer is enabled, the Timer Output signal is complemented upon timer Reload.

Continuous mode
When the timer is disabled, the Timer Output signal is set to the value of this bit. When the timer is enabled, the Timer Output signal is complemented upon timer Reload.

Counter mode
When the timer is disabled, the Timer Output signal is set to the value of this bit. When the timer is enabled, the Timer Output signal is complemented upon timer Reload.

PWM mode
0 = Timer Output is forced Low (0) when the timer is disabled. When enabled, the Timer Output is forced High (1) upon PWM count match and forced Low (0) upon Reload.
1 = Timer Output is forced High (1) when the timer is disabled. When enabled, the Timer Output is forced Low (0) upon PWM count match and forced High (1) upon Reload.

Figures 68 and 69 illustrates the asynchronous data format employed by the UART without parity and with parity, respectively.

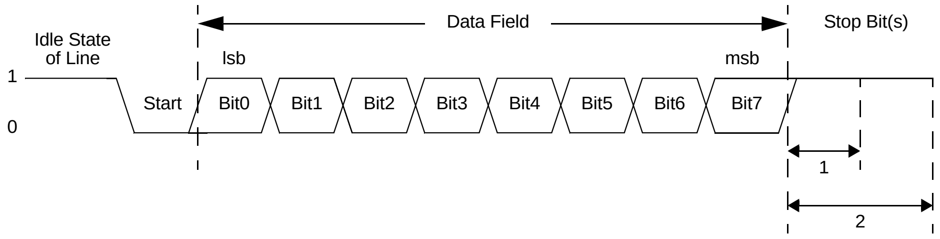

Figure 68. UART Asynchronous Data Format without Parity

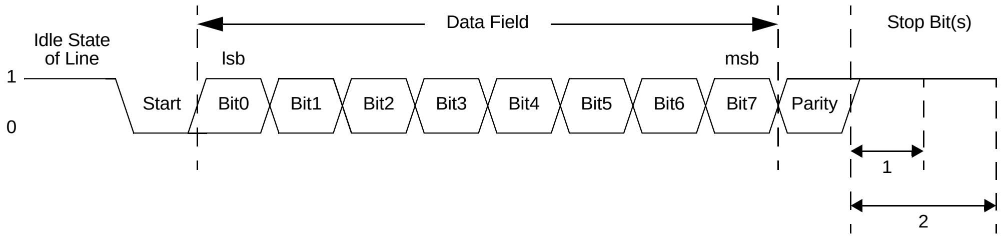

Figure 69. UART Asynchronous Data Format with Parity

Transmitting Data using the Polled Method

Follow these steps to transmit data using the polled method of operation:

1. Write to the UART Baud Rate High and Low Byte registers to set the desired baud rate.
2. Enable the UART pin functions by configuring the associated GPIO Port pins for alternate function operation.
3. Write to the UART Control 1 register to enable Multiprocessor (9-bit) mode functions, if desired.
4. Write to the UART Control 0 register to:
   - Set the transmit enable bit (`TEN`) to enable the UART for data transmission
   - Enable parity, if desired, and select either even or odd parity.
   - Set or clear the `CTSE` bit to enable or disable control from the receiver using the $\overline{\text{CTS}}$ pin.

Table 53. UARTx Status 1 Register (UxSTAT1)

BITS	7	6	5	4	3	2	1	0
FIELD	Reserved							MPRX
RESET	0	0	0	0	0	0	0	0
R/W	R	R	R	R	R	R	R	R
ADDR	F44H and F4CH							

Reserved
These bits are reserved and must be 0.

MPRX—Multiprocessor Receive
This status bit is for the receiver and reflects the actual status of the last multiprocessor bit received. Reading from the UART Data register resets this bit to 0.

UARTx Control 0 and Control 1 Registers

The UARTx Control 0 and Control 1 registers (Tables 54 and 55) configure the properties of the UART's transmit and receive operations. The UART Control registers must ben be written while the UART is enabled.

Table 54. UARTx Control 0 Register (UxCTL0)

BITS	7	6	5	4	3	2	1	0
FIELD	TEN	REN	CTSE	PEN	PSEL	SBRK	STOP	LBEN
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	F42H and F4AH							

TEN—Transmit Enable
This bit enables or disables the transmitter. The enable is also controlled by the $\overline{\text{CTS}}$ signal and the `CTSE` bit. If the $\overline{\text{CTS}}$ signal is low and the `CTSE` bit is 1, the transmitter is enabled.
0 = Transmitter disabled.
1 = Transmitter enabled.

REN—Receive Enable
This bit enables or disables the receiver.
0 = Receiver disabled.
1 = Receiver enabled.

SPI Control Register

The SPI Control register configures the SPI for transmit and receive operations.

Table 61. SPI Control Register (SPICTL)

BITS	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FIELD	IRQE	STR	BIRQ	PHASE	CLKPOL	WOR	MMEN	SPIEN
RESET	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	F61H							

IRQE—Interrupt Request Enable
0 = SPI interrupts are disabled. No interrupt requests are sent to the Interrupt Controller.
1 = SPI interrupts are enabled. Interrupt requests are sent to the Interrupt Controller.

STR—Start an SPI Interrupt Request
0 = No effect.
1 = Setting this bit to 1 also sets the `IRQ` bit in the SPI Status register to 1. Setting this bit forces the SPI to send an interrupt request to the Interrupt Control. This bit can be used by software for a function similar to transmit buffer empty in a UART.

BIRQ—BRG Timer Interrupt Request
If the SPI is enabled, this bit has no effect. If the SPI is disabled:
0 = The Baud Rate Generator timer function is disabled.
1 = The Baud Rate Generator timer function and time-out interrupt are enabled.

PHASE—Phase Select
Sets the phase relationship of the data to the clock. Refer to the **SPI Clock Phase and Polarity Control** section for more information on operation of the `PHASE` bit.

CLKPOL—Clock Polarity
0 = SCK idles Low (0).
1 = SCK idle High (1).

WOR—Wire-OR (Open-Drain) Mode Enabled
0 = SPI signal pins not configured for open-drain.
1 = All four SPI signal pins (SCK, $\overline{\text{SS}}$, MISO, MOSI) configured for open-drain function. This setting is typically used for multi-master and/or multi-slave configurations.

MMEN—SPI Master Mode Enable
0 = SPI configured in Slave mode.
1 = SPI configured in Master mode.

4. The I^{2}C Controller loads the I^{2}C Shift register with the contents of the I^{2}C Data register.
5. After the first bit has been shifted out, a Transmit interrupt is asserted.
6. Software responds by writing eight bits of address to the I^{2}C Data register.
7. The I^{2}C Controller completes shifting of the two address bits and a 0 (write).
8. The I^{2}C slave sends an acknowledge by pulling the SDA signal Low during the next high period of SCL.
9. The I^{2}C Controller loads the I^{2}C Shift register with the contents of the I^{2}C Data register.
10. The I^{2}C Controller shifts out the next eight bits of address. After the first bits are shifted, the I^{2}C Controller generates a Transmit interrupt.
11. Software responds by setting the `START` bit of the I^{2}C Control register to generate a repeated START.
12. Software responds by writing `11110B` followed by the 2-bit slave address and a 1 (read).
13. Software responds by setting the NAK bit of the I^{2}C Control register, so that a Not Acknowledge is sent after the first byte of data has been read. If you want to read only one byte, software responds by setting the NAK bit of the I^{2}C Control register.
14. After the I^{2}C Controller shifts out the address bits mentioned in step 9, the I^{2}C slave sends an acknowledge by pulling the SDA signal Low during the next high period of SCL.
15. The I^{2}C Controller sends the repeated START condition.
16. The I^{2}C Controller loads the I^{2}C Shift register with the contents of the I^{2}C Data register.
17. The I^{2}C Controller sends `11110B` followed by the 2-bit slave read and a 1 (read).
18. The I^{2}C slave sends an acknowledge by pulling the SDA signal Low during the next high period of SCL.
19. The I^{2}C slave sends a byte of data.
20. A Receive interrupt is generated.
21. Software responds by reading the I^{2}C Data register.
22. Software responds by setting the `STOP` bit of the I^{2}C Control register.
23. A NAK condition is sent to the I^{2}C slave.
24. A STOP condition is sent to the I^{2}C slave.

Table 74. DMA*x* Start/Current Address Low Byte Register (DMA*x*START)

BITS	7	6	5	4	3	2	1	0
FIELD	DMA_START							
RESET	X	X	X	X	X	X	X	X
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	FB3H, FHBH							

DMA_START—DMA*x* Start/Current Address Low
These bits, with the four lower bits of the DMA*x*_H register, form the 12-bit Start/Current address. The full 12-bit address is given by {DMA_START_H[3:0], DMA_START[7:0]}.

DMA*x* End Address Low Byte Register

The DMA*x* End Address Low Byte register, in conjunction with the DMA*x*_H register, forms a 12-bit End Address.

Table 75. DMA*x* End Address Low Byte Register (DMA*x*END)

BITS	7	6	5	4	3	2	1	0
FIELD	DMA_END							
RESET	X	X	X	X	X	X	X	X
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	FB4H, FBCH							

DMA_END—DMA*x* End Address Low
These bits, with the four upper bits of the DMA*x*_H register, form a 12-bit address. This address is the ending location of the DMA*x* transfer. The full 12-bit address is given by {DMA_END_H[3:0], DMA_END[7:0]}.

DMA_ADC Address Register

The DMA_ADC Address register points to a block of the Register File to store ADC conversion values as illustrated in Table 76. This register contains the seven most-significant bits of the 12-bit Register File addresses. The five least-significant bits are calculated from the ADC Analog Input number (5-bit base address is equal to twice the ADC Analog Input number). The 10-bit ADC conversion data is stored as two bytes with the most significant byte of the ADC data stored at the even numbered Register File address.

Analog-to-Digital Converter

Overview

The Analog-to-Digital Converter (ADC) converts an analog input signal to a 10-bit binary number. The features of the sigma-delta ADC include:

- 12 analog input sources are multiplexed with general-purpose I/O ports
- Interrupt upon conversion complete
- Internal voltage reference generator
- Direct Memory Access (DMA) controller can automatically initiate data conversion and transfer of the data from 1 to 12 of the analog inputs.

Architecture

Figure 83 illustrates the three major functional blocks (converter, analog multiplexer, and voltage reference generator) of the ADC. The ADC converts an analog input signal to its digital representation. The 12-input analog multiplexer selects one of the 12 analog input sources. The ADC requires an input reference voltage for the conversion. The voltage reference for the conversion may be input through the external VREF pin or generated internally by the voltage reference generator.

1. Enable the desired analog inputs by configuring the general-purpose I/O pins for alternate function. This configuration disables the digital input and output drivers.
2. Write to the ADC Control register to configure the ADC and begin the conversion. The bit fields in the ADC Control register can be written simultaneously:
   - Write to `ANAIN[3:0]` to select one of the 12 analog input sources.
   - Clear `CONT` to 0 to select a single-shot conversion.
   - Write to $\overline{\texttt{VREF}}$ to enable or disable the internal voltage reference generator.
   - Set `CEN` to 1 to start the conversion.
3. `CEN` remains 1 while the conversion is in progress. A single-shot conversion requires 5129 system clock cycles to complete. If a single-shot conversion is requested from an ADC powered-down state, the ADC uses 40 additional clock cycles to power-up before beginning the 5129 cycle conversion.
4. When the conversion is complete, the ADC control logic performs the following operations:
   - 10-bit data result written to {ADCD_H[7:0], ADCD_L[7:6]}.
   - `CEN` resets to 0 to indicate the conversion is complete.
   - An interrupt request is sent to the Interrupt Controller.
5. If the ADC remains idle for 160 consecutive system clock cycles, it is automatically powered-down.

Continuous Conversion

When configured for continuous conversion, the ADC continuously performs an analog-to-digital conversion on the selected analog input. Each new data value over-writes the previous value stored in the ADC Data registers. An interrupt is generated only at the end of the first conversion after enabling.

Caution: In Continuous mode, users must be aware that ADC updates are limited by the input signal bandwidth of the ADC and the latency of the ADC and its digital filter. Step changes at the input are not seen at the next output from the ADC. The response of the ADC (in all modes) is limited by the input signal bandwidth and the latency.

The steps for setting up the ADC and initiating continuous conversion are as follows:

1. Enable the desired analog input by configuring the general-purpose I/O pins for alternate function. This disables the digital input and output driver.
2. Write to the ADC Control register to configure the ADC for continuous conversion. The bit fields in the ADC Control register may be written simultaneously:
   - Write to `ANAIN[3:0]` to select one of the 12 analog input sources.

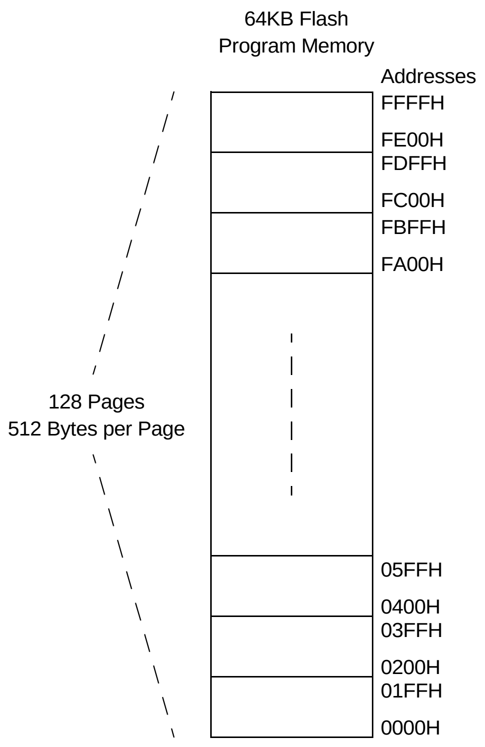

Figure 84. Flash Memory Arrangement

Operation

The Flash Controller programs and erases the Flash memory. The Flash Controller provides the proper Flash controls and timing for byte programming, Page Erase, and Mass Erase of the Flash memory. The Flash Controller contains a protection mechanism, via the Flash Control register (FCTL) to prevent accidental programming or erasure. The Flow Chart in Figure 85 illustrates basic Flash Controller operation. The following subsections provide details on the various operations (Lock, Unlock, Byte Programming, Page Erase, and Mass Erase) listed in Figure 85.

Flash Control Register Definitions

Flash Control Register

The Flash Controller must be unlocked via the Flash Control register before programming or erasing the Flash memory. Writing the sequence `73H 8CH`, sequentially, to the Flash Control register unlocks the Flash Controller. When the Flash Controller is unlocked, writing to the Flash Control register can initiate either Page Erase or Mass Erase of the Flash memory. Writing an invalid value or an invalid sequence returns the Flash Controller to its locked state. The Write-only Flash Control Register shares its Register File address with the Read-only Flash Status Register.

Table 85. Flash Control Register (FCTL)

BITS	7	6	5	4	3	2	1	0
FIELD	FCMD							
RESET	0	0	0	0	0	0	0	0
R/W	W	W	W	W	W	W	W	W
ADDR	FF8H							

FCMD—Flash Command
73H = First unlock command.
8CH = Second unlock command.
95H = Page erase command (must be third command in sequence to initiate Page Erase).
63H = Mass erase command (must be third command in sequence to initiate Mass Erase).

`FHSWP`—Flash High Sector Write Protect
`FWP`—Flash Write Protect
These two Option Bits combine to provide 3 levels of Program Memory protection:

`FHSWP`	**`FWP`**	**Description**
0	0	Programming and erasure disabled for all of Program Memory. Programming, Page Erase, and Mass Erase via User Code is disabled. Mass Erase is available through the On-Chip Debugger.
1	0	Programming and Page Erase are enabled for the High Sector of the Program Memory only. The High Sector on the Z8F640x family device contains 1KB to 4KB of Flash with addresses at the top of the available Flash memory. Programming and Page Erase are disabled for the other portions of the Program Memory. Mass erase through user code is disabled. Mass Erase is available through the On-Chip Debugger.
0 or 1	1	Programming, Page Erase, and Mass Erase are enabled for all of Program Memory.

Program Memory Address 0001H

Table 91. Options Bits at Program Memory Address 0001H

BITS	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FIELD	Reserved							
RESET	U	U	U	U	U	U	U	U
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	Program Memory 0001H							
Note: U = Unchanged by Reset. R/W = Read/Write.								

Reserved
These Option Bits are reserved for future use and must always be 1. This setting is the default for unprogrammed (erased) Flash.

Table 93. On-Chip Debugger Commands

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

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

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

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

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

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

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

General Purpose I/O Port Output Timing

Figure 94 and Table 108 provide timing information for GPIO Port pins.

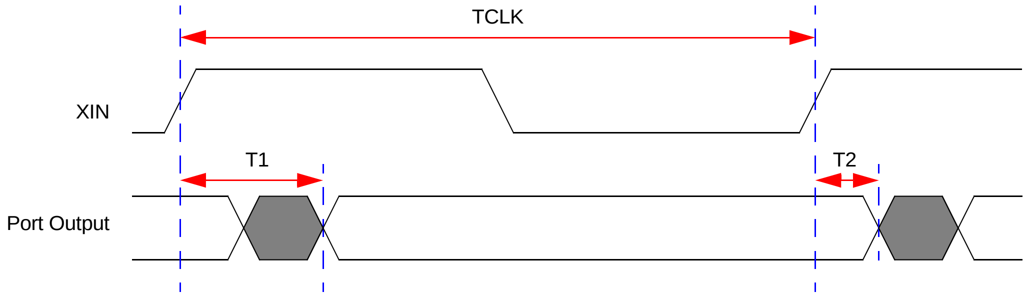

Figure 94. GPIO Port Output Timing

Table 108. GPIO Port Output Timing

		Delay (ns)	
Parameter	**Abbreviation**	**Minimum**	**Maximum**
T_1	XIN Rise to Port Output Valid Delay	–	15
T_2	XIN Rise to Port Output Hold Time	2	–

SPI Master Mode Timing

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

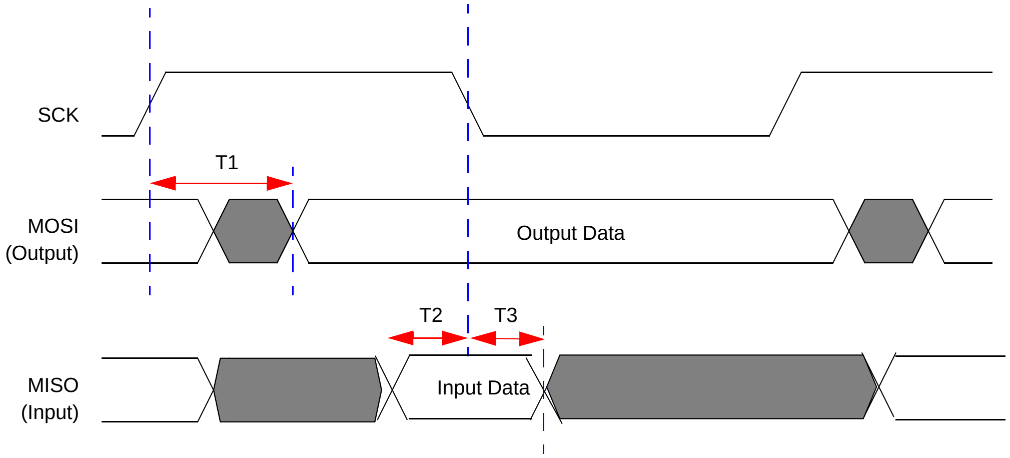

Figure 96. SPI Master Mode Timing

Table 110. SPI Master Mode Timing

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

Index