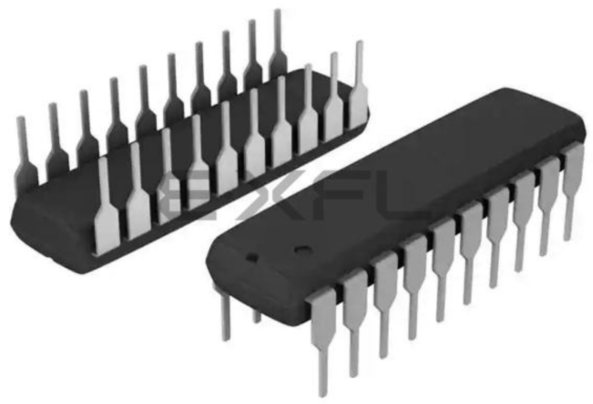
Zilog - Z8F0411PH020SC Datasheet



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, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	11
Program Memory Size	4KB (4K x 8)
Program Memory Type	FLASH
EEPROM Size	-
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	20-DIP (0.300", 7.62mm)
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/zilog/z8f0411ph020sc

Introduction

This Product Specification provides detailed operating information for Z8 Encore! XP® F0822 Series devices within the Z8 Encore! XP Microcontroller (MCU) family of products. Within this document, Z8 Encore! XP® F0822 Series is referred as Z8 Encore! XP or the F0822 Series unless specifically stated otherwise.

About This Manual

Zilog recommends that you read and understand everything in this manual before setting up and using the product. We have designed this Product Specification to be used either as a *how to* procedural manual or a reference guide to important data.

Intended Audience

This document is written for Zilog customers who are experienced at working with microcontrollers, integrated circuits, or printed circuit assemblies.

Manual Conventions

The following assumptions and conventions are adopted to provide clarity and ease of use:

Courier Typeface

Commands, code lines and fragments, bits, equations, hexadecimal addresses, and various executable items are distinguished from general text by the use of the `Courier` typeface. Where the use of the font is not indicated, as in the Index, the name of the entity is presented in upper case.

- **Example:** FLAGS[1] is `smrf`.

Hexadecimal Values

Hexadecimal values are designated by uppercase *H* suffix and appear in the `Courier` typeface.

- **Example:** R1 is set to `F8H`.

Brackets

The square brackets [ ], indicate a register or bus.

- **Example:** For the register R1[7:0], R1 is an 8-bit register, R1[7] is the most significant bit, and R1[0] is the least significant bit.

Block Diagram

Figure 1 displays the block diagram of the architecture of Z8 Encore! XP® F0822 Series devices.

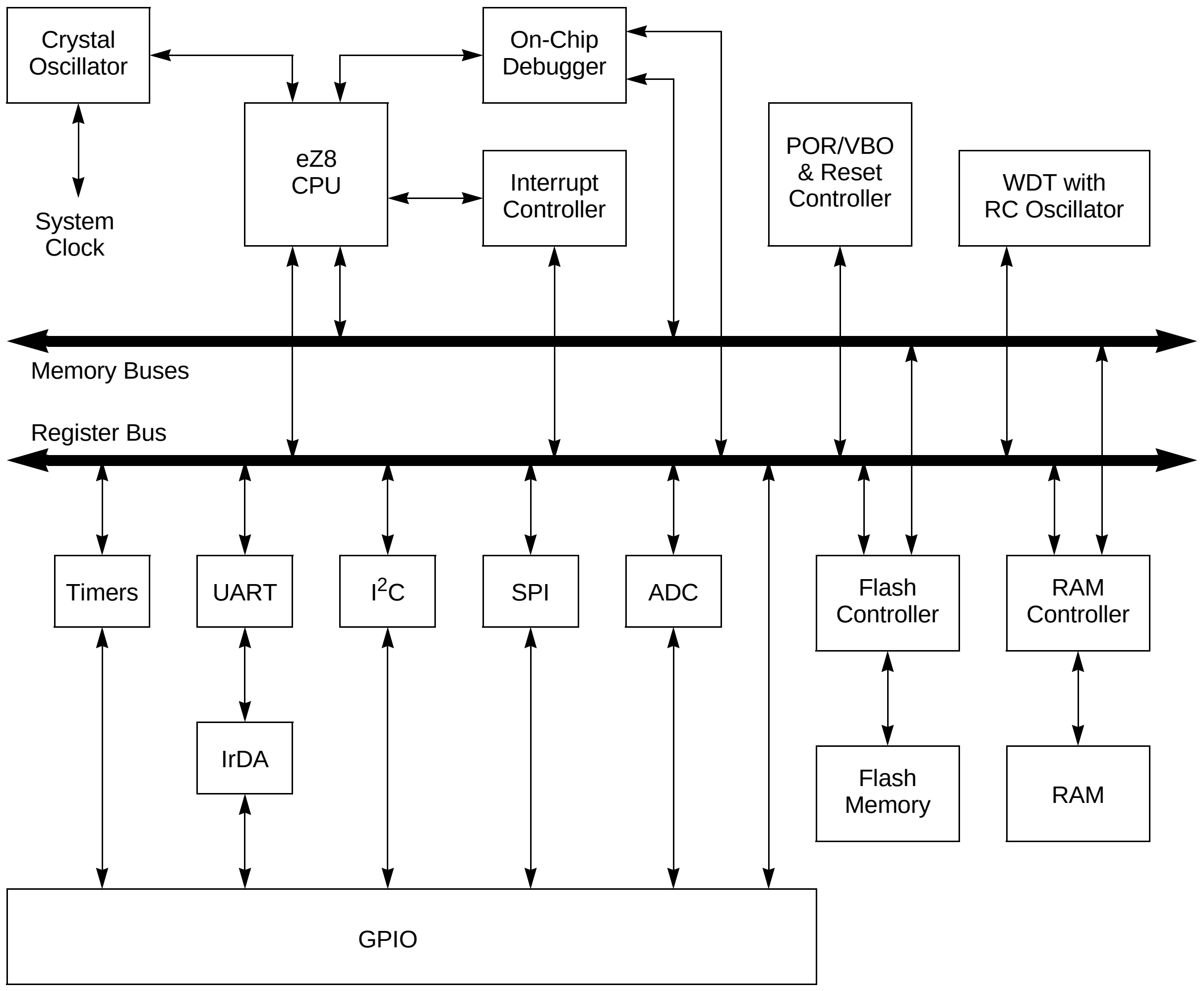


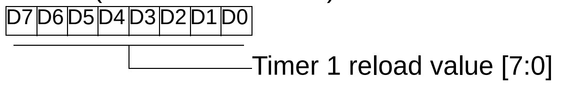
Figure 1. Z8 Encore! XP® F0822 Series Block Diagram

CPU and Peripheral Overview

eZ8 CPU Features

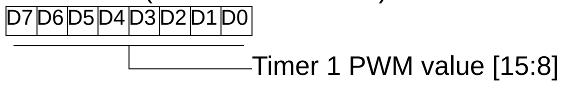
Zilog's latest eZ8 8-bit CPU, meets the continuing demand for faster and more code-efficient microcontrollers. The eZ8 CPU executes a superset of the original Z8® instruction set.

Timer 1 Reload Low Byte
T1RL (F0BH - Read/Write)


D7 D6 D5 D4 D3 D2 D1 D0

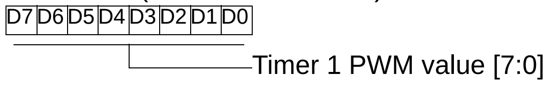
Timer 1 reload value [7:0]

Timer 1 PWM High Byte
T1PWMH (F0CH - Read/Write)


D7 D6 D5 D4 D3 D2 D1 D0

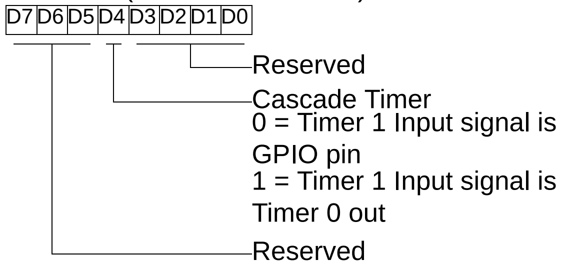
Timer 1 PWM value [15:8]

Timer 1 PWM Low Byte
T1PWML (F0DH - Read/Write)


D7 D6 D5 D4 D3 D2 D1 D0

Timer 1 PWM value [7:0]

Timer 1 Control 0
T1CTL0 (F0EH - Read/Write)

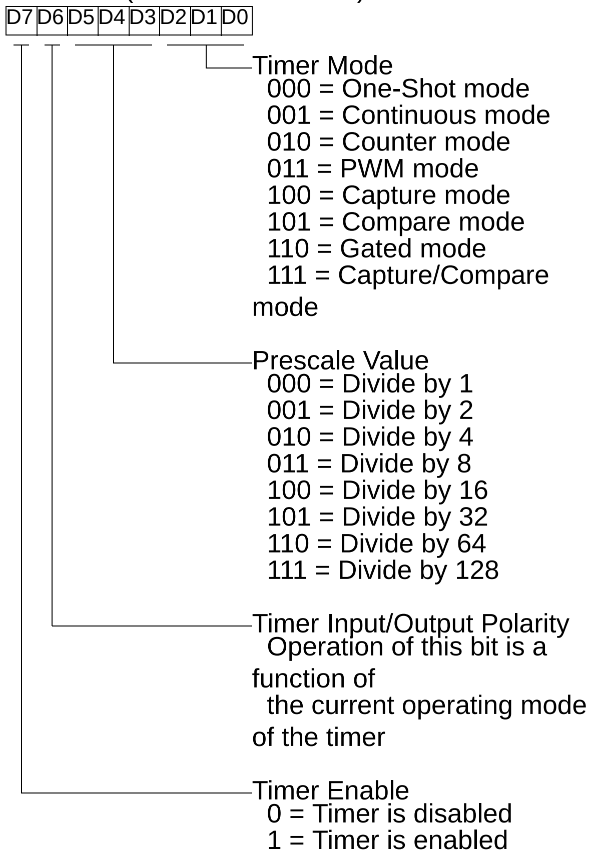

D7 D6 D5 D4 D3 D2 D1 D0

Reserved

Cascade Timer
0 = Timer 1 Input signal is GPIO pin
1 = Timer 1 Input signal is Timer 0 out

Reserved

Timer 1 Control 1
T1CTL1 (F0FH - Read/Write)


D7 D6 D5 D4 D3 D2 D1 D0

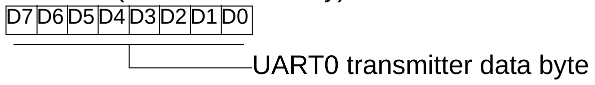
Timer Mode
000 = One-Shot mode
001 = Continuous mode
010 = Counter mode
011 = PWM mode
100 = Capture mode
101 = Compare mode
110 = Gated mode
111 = Capture/Compare mode

Prescale Value
000 = Divide by 1
001 = Divide by 2
010 = Divide by 4
011 = Divide by 8
100 = Divide by 16
101 = Divide by 32
110 = Divide by 64
111 = Divide by 128

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

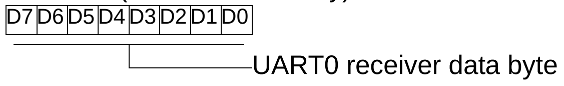
Timer Enable
0 = Timer is disabled
1 = Timer is enabled

UART0 Transmit Data
U0TXD (F40H - Write Only)


D7 D6 D5 D4 D3 D2 D1 D0

UART0 transmitter data byte

UART0 Receive Data
U0RXD (F40H - Read Only)


D7 D6 D5 D4 D3 D2 D1 D0

UART0 receiver data byte

Table 13. GPIO Port Registers and Sub-Registers (Continued)

Port Register Mnemonic	Port Register Name
P*x*OC	Output Control (Open-Drain)
P*x*HDE	High Drive Enable
P*x*SMRE	Stop Mode Recovery Source Enable
P*x*PUE	Pull-up Enable

Port A–C Address Registers

The Port A–C Address Registers select the GPIO Port functionality accessible through the Port A–C Control Registers. The Port A–C Address and Control Registers combine to provide access to all GPIO Port control (Table 14).

Table 14. Port A–C GPIO Address Registers (P*x*ADDR)

BITS	7	6	5	4	3	2	1	0
FIELD	PADDR[7:0]							
RESET	00H							
R/W	R/W							
ADDR	FD0H, FD4H, FD8H							

PADDR[7:0]—Port Address
The Port Address selects one of the sub-registers accessible through the Port Control register.

PADDR[7:0]	Port Control Sub-Register Accessible Using the Port A–C 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	Pull-up Enable
07H–FFH	No Function

Interrupt Request 2 Register

The Interrupt Request 2 (IRQ2) Register (Table 27) stores interrupt requests for both vectored and polled interrupts. When a request is presented to the interrupt controller, the corresponding bit in the IRQ2 register becomes 1. If interrupts are globally enabled (vectored interrupts), the interrupt controller passes an interrupt request to the eZ8 CPU. If interrupts are globally disabled (polled interrupts), the eZ8 CPU reads the IRQ2 Register to determine if any interrupt requests are pending.

Table 27. Interrupt Request 2 Register (IRQ2)

BITS	7	6	5	4	3	2	1	0
FIELD	Reserved				PC3I	PC2I	PC1I	PC0I
RESET	0							
R/W	R/W							
ADDR	FC6H							

Reserved—Must be 0

PC*x*I—Port C Pin *x* Interrupt Request
0 = No interrupt request is pending for GPIO Port C pin *x*.
1 = An interrupt request from GPIO Port C pin *x* is awaiting service.
Where *x* indicates the specific GPIO Port C pin number (0 through 3).

IRQ0 Enable High and Low Bit Registers

The IRQ0 Enable High and Low Bit Registers (Table 29 and Table 30) form a priority encoded enabling for interrupts in the Interrupt Request 0 Register. Priority is generated by setting bits in each register. Table 28 describes the priority control for IRQ0.

Table 28. IRQ0 Enable and Priority Encoding

IRQ0ENH[*x*]	IRQ0ENL[*x*]	Priority	Description
0	0	Disabled	Disabled
0	1	Level 1	Low
1	0	Level 2	Nominal
1	1	Level 3	High

where *x* indicates the register bits from 0 through 7.

Register is read, the contents of the Timer Low Byte Register are placed in a holding register. A subsequent read from the Timer Low Byte Register returns the value in the holding register. 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

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

Timer 0–1 High and Low Byte Registers

The Timer 0–1 High and Low Byte (TxH and TxL) Registers (Table 39) 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 reads 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.

Table 39. Timer 0–1 High Byte Register (TxH)

BITS	7	6	5	4	3	2	1	0
FIELD	TH							
RESET	0							
R/W	R/W							
ADDR	F00H, F08H							

Table 40. Timer 0–1 Low Byte Register (TxL)

BITS	7	6	5	4	3	2	1	0
FIELD	TL							
RESET	0							1
R/W	R/W							
ADDR	F01H, F09H							

5. Check the TDRE bit in the UART Status 0 Register to determine if the Transmit Data Register is empty (indicated by a 1). If empty, continue to step 6. If the Transmit Data Register is full (indicated by a 0), continue to monitor the TDRE bit until the Transmit Data Register becomes available to receive new data.
6. Write the UART Control 1 Register to select the outgoing address bit:
   – Set the Multiprocessor Bit Transmitter (MPBT) if sending an address byte, clear it if sending a data byte.
7. Write data byte to the UART Transmit Data Register. The transmitter automatically transfers data to the Transmit Shift Register and then transmits the data.
8. If required, and multiprocessor mode is enabled, make any changes to the Multiprocessor Bit Transmitter (MPBT) value.
9. To transmit additional bytes, return to step 5.

Transmitting Data Using Interrupt-Driven Method

The UART Transmitter interrupt indicates the availability of the Transmit Data Register to accept new data for transmission. Follow the below steps to configure the UART for interrupt-driven data transmission:

1. Write to the UART Baud Rate High and Low Byte Registers to set the required baud rate.
2. Enable the UART pin functions by configuring the associated GPIO Port pins for alternate function operation.
3. Execute a DI instruction to disable interrupts.
4. Write to the Interrupt Control Registers to enable the UART Transmitter interrupt and set the required priority.
5. If MULTIPROCESSOR mode is required, write to the UART Control 1 Register to enable Multiprocessor (9-bit) mode functions:
   – Set the Multiprocessor Mode Select (MPEN) to enable MULTIPROCESSOR mode.
6. Write to the UART Control 0 Register to:
   – Set the transmit enable (TEN) bit to enable the UART for data transmission
   – Enable parity, if required, and if MULTIPROCESSOR mode is not enabled, and select either even or odd parity.
   – Set or clear the CTSE bit to enable or disable control from the remote receiver through the $\overline{\text{CTS}}$ pin.
7. Execute an EI instruction to enable interrupts.

Reserved—Must be 0

NEWFRM—Status bit denoting the start of a new frame. Reading the UART Receive Data Register resets this bit to 0.
0 = The current byte is not the first data byte of a new frame.
1 = The current byte is the first data byte of a new frame.

MPRX—Multiprocessor Receive
Returns the value of the last multiprocessor bit received. Reading from the UART Receive Data Register resets this bit to 0.

UART Control 0 and Control 1 Registers

The UART Control 0 and Control 1 registers (Table 56 and Table 57 on page 104) configure the properties of the UART's transmit and receive operations. The UART Control Registers must not been written while the UART is enabled.

Table 56. UART Control 0 Register (U0CTL0)

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

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.

CTSE—$\overline{\text{CTS}}$ Enable
0 = The $\overline{\text{CTS}}$ signal has no effect on the transmitter.
1 = The UART recognizes the $\overline{\text{CTS}}$ signal as an enable control from the transmitter.

PEN—Parity Enable
This bit enables or disables parity. Even or odd is determined by the `PSEL` bit. This bit is overridden by the `MPEN` bit.
0 = Parity is disabled.
1 = The transmitter sends data with an additional parity bit and the receiver receives an additional parity bit.

Serial Peripheral Interface

The Serial Peripheral Interface (SPI) is a synchronous interface allowing several SPI-type devices to be interconnected. SPI-compatible devices include EEPROMs, Analog-to-Digital Converters, and ISDN devices. Features of the SPI include:

- Full-duplex, synchronous, and character-oriented communication
- Four-wire interface
- Data transfers rates up to a maximum of one-half the system clock frequency
- Error detection
- Dedicated Baud Rate Generator

The SPI is not available in 20-pin package devices.

Architecture

The SPI is be configured as either a Master (in single or multi-master systems) or a Slave as displayed in Figure 20 through Figure 22.

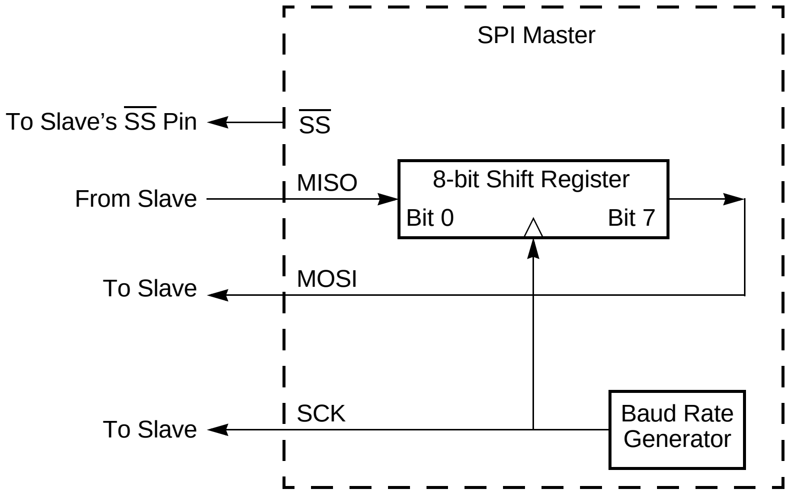


Figure 20. SPI Configured as a Master in a Single Master, Single Slave System

SPI Status Register

The SPI Status Register indicates the current state of the SPI. All bits revert to their reset state if the `SPIEN` bit in the SPICTL Register equals 0.

Table 65. SPI Status Register (SPISTAT)

BITS	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FIELD	IRQ	OVR	COL	ABT	Reserved		TXST	SLAS
RESET	0							1
R/W	R/W*				R			
ADDR	F62H							
R/W* = Read access. Write a 1 to clear the bit to 0.								

IRQ—Interrupt Request
If SPIEN = 1, this bit is set if the STR bit in the SPICTL Register is set, or upon completion of an SPI Master or Slave transaction. This bit does not set if SPIEN = 0 and the SPI Baud Rate Generator is used as a timer to generate the SPI interrupt.
0 = No SPI interrupt request pending.
1 = SPI interrupt request is pending.

OVR—Overrun
0 = An overrun error has not occurred.
1 = An overrun error has been detected.

COL—Collision
0 = A multi-master collision (mode fault) has not occurred.
1 = A multi-master collision (mode fault) has been detected.

ABT—SLAVE mode transaction abort
This bit is set if the SPI is configured in SLAVE mode, a transaction is occurring and $\overline{\text{SS}}$ 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.

Reserved—Must be 0

TXST—Transmit Status
0 = No data transmission currently in progress.
1 = Data transmission currently in progress.

SLAS—Slave Select
If SPI enabled as a Slave
0 = $\overline{\text{SS}}$ input pin is asserted (Low)
1 = $\overline{\text{SS}}$ input is not asserted (High).
If SPI enabled as a Master, this bit is not applicable.

1 = $\overline{\text{SS}}$ pin driven High (1).
This bit has no effect if `SSIO` = 0 or SPI configured as a Slave

SPI Diagnostic State Register

The SPI Diagnostic State Register provides observability of internal state. This is a read only register used for SPI diagnostics.

Table 67. SPI Diagnostic State Register (SPIDST)

BITS	7	6	5	4	3	2	1	0
FIELD	SCKEN	TCKEN	SPISTATE					
RESET	0							
R/W	R							
ADDR	F64H							

SCKEN–Shift Clock Enable
0 = The internal Shift Clock Enable signal is deasserted
1 = The internal Shift Clock Enable signal is asserted (shift register is updates on next system clock)

TCKEN–Transmit Clock Enable
0 = The internal Transmit Clock Enable signal is deasserted.
1 = The internal Transmit Clock Enable signal is asserted. When this is asserted the serial data out is updated on the next system clock (MOSI or MISO).

SPISTATE–SPI State Machine
Defines the current state of the internal SPI State Machine.

SPI Baud Rate High and Low Byte Registers

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. When configured as a general purpose timer, the interrupt interval is calculated using the following equation:

Interrupt Interval (s) = System Clock Period (s) × BRG[15:0]

Table 68. SPI Baud Rate High Byte Register (SPIBRH)

BITS	7	6	5	4	3	2	1	0
FIELD	BRH							
RESET	1							
R/W	R/W							
ADDR	F66H							

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.
16. The I^{2}C Controller completes transmission of the data on the SDA signal.
17. The slave can either Acknowledge or Not Acknowledge the last byte. Because either the STOP or START bit is already set, the NCKI interrupt does not occur.
18. The I^{2}C Controller sends the STOP (or RESTART) condition to the I^{2}C bus. The STOP or START bit is cleared.

Address Only Transaction with a 10-bit Address

In the situation where software wants to determine if a slave with a 10-bit address is responding without sending or receiving data, a transaction is done which only consists of an address phase. Figure 28 displays this "address only" transaction to determine if a slave with 10-bit address will acknowledge. As an example, this transaction is used after a "write" has been done to a EEPROM to determine when the EEPROM completes its internal write operation and is once again responding to I2C transactions. If the slave does not Acknowledge the transaction is repeated until the slave is able to Acknowledge.

S	Slave Address 1st 7 bits	W = 0	$A/\overline{A}$	Slave Address 2nd Byte	$A/\overline{A}$	P

Figure 28. 10-Bit Address Only Transaction Format

Follow the steps below for an address only transaction to a 10-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 (TDRE = 1)
4. Software responds to the `TDRE` interrupt 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 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 is shifted out by the SDA signal, the Transmit Interrupt is asserted.

Reserved
These Option Bits are reserved for future use and must always be 1.

The following information applies only to the Flash versions of the F0822 Series devices:

FWP—Flash Write Protect
These two Option Bits combine to provide three levels of Program Memory protection:

FWP	Description
0	Programming, Page Erase, and Mass Erase using User Code is disabled. Mass Erase is available through the OCD.
1	Programming and Page Erase are enabled for all of Flash Program Memory.

Flash Memory Address 0001H

Table 90. Options Bits at Flash Memory Address 0001H

BITS	7	6	5	4	3	2	1	0
FIELD	Reserved							
RESET	U							
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.

Figure 45 displays the maximum current consumption in STOP mode with the VBO and Watchdog Timer enabled versus the power supply voltage. All GPIO pins are configured as outputs and driven High.

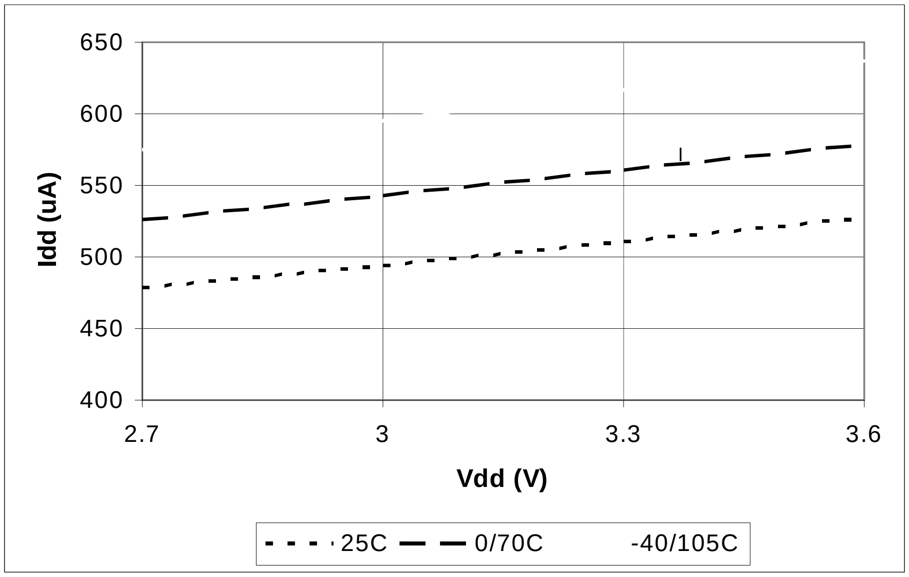


Figure 45. Maximum STOP Mode I_{DD} with VBO Enabled versus Power Supply Voltage

Figure 46 on page 193 displays the maximum current consumption in STOP mode with the VBO disabled and Watchdog Timer enabled versus the power supply voltage. All GPIO pins are configured as outputs and driven High. Disabling the Watchdog Timer and its internal RC oscillator in STOP mode will provide some additional reduction in STOP mode current consumption. This small current reduction is indistinquishable on the scale of Figure 46 on page 193.

AC Characteristics

Table 98 provides information on the AC characteristics and timing. All AC timing information assumes a standard load of 50 pF on all outputs.

Table 98. AC Characteristics

		V_{DD} = 2.7 - 3.6 V T_A = -40 °C to 105 °C			
Symbol	**Parameter**	**Minimum**	**Maximum**	**Units**	**Conditions**
F_{SYSCLK}	System Clock Frequency (ROM)	–	20.0	MHz	
F_{SYSCLK}	System Clock Frequency (Flash)	–	20.0	MHz	Read-only from Flash memory.
		0.032768	20.0	MHz	Program or erasure of the Flash memory.
F_{XTAL}	Crystal Oscillator Frequency	0.032768	20.0	MHz	System clock frequencies below the crystal oscillator minimum require an external clock driver.
T_{XIN}	System Clock Period	50	–	ns	$T_{CLK} = 1/F_{sysclk}$
T_{XINH}	System Clock High Time	20	30	ns	T_{CLK} = 50 ns
T_{XINL}	System Clock Low Time	20	30	ns	T_{CLK} = 50 ns

I²C Timing

Figure 53 and Table 110 provide timing information for I²C pins.

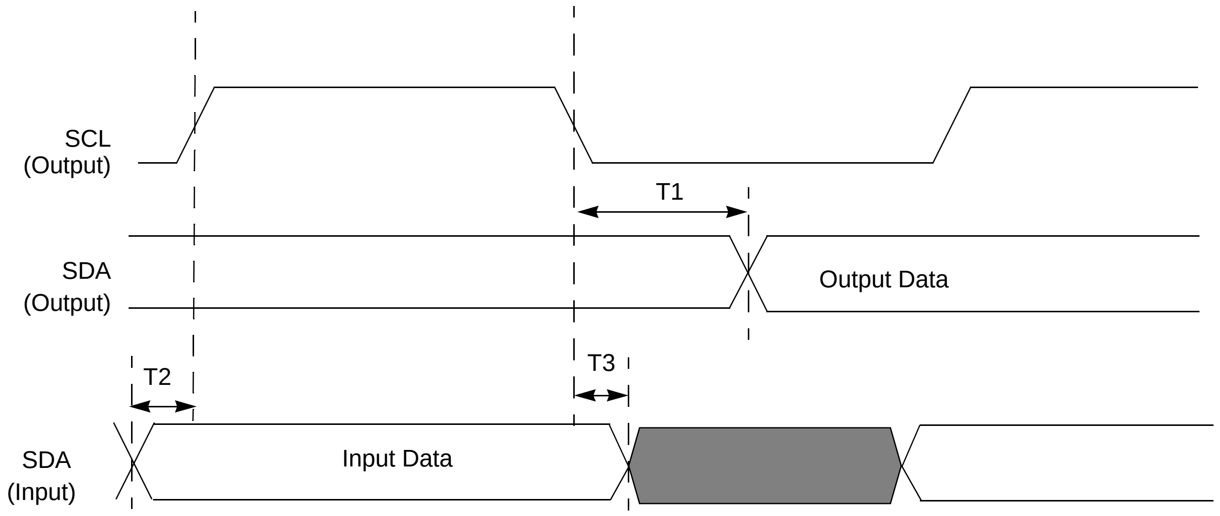


Figure 53. I²C Timing

Table 110. I²C Timing

		Delay (ns)	
Parameter	**Abbreviation**	**Minimum**	**Maximum**
I²C			
T_1	SCL Fall to SDA output delay	SCL period/4	
T_2	SDA Input to SCL rising edge Setup Time	0	
T_3	SDA Input to SCL falling edge Hold Time	0	

eZ8 CPU Instruction Classes

eZ8 CPU instructions are divided functionally into the following groups:

- Arithmetic
- Bit Manipulation
- Block Transfer
- CPU Control
- Load
- Logical
- Program Control
- Rotate and Shift

Tables 118 through Table 125 on page 218 contain the instructions belonging to each group and the number of operands required for each instruction. Some instructions appear in more than one table as these instruction can be considered as a subset of more than one category. Within these tables, the source operand is identified as ’src’, the destination operand is ’dst’ and a condition code is ’cc’.

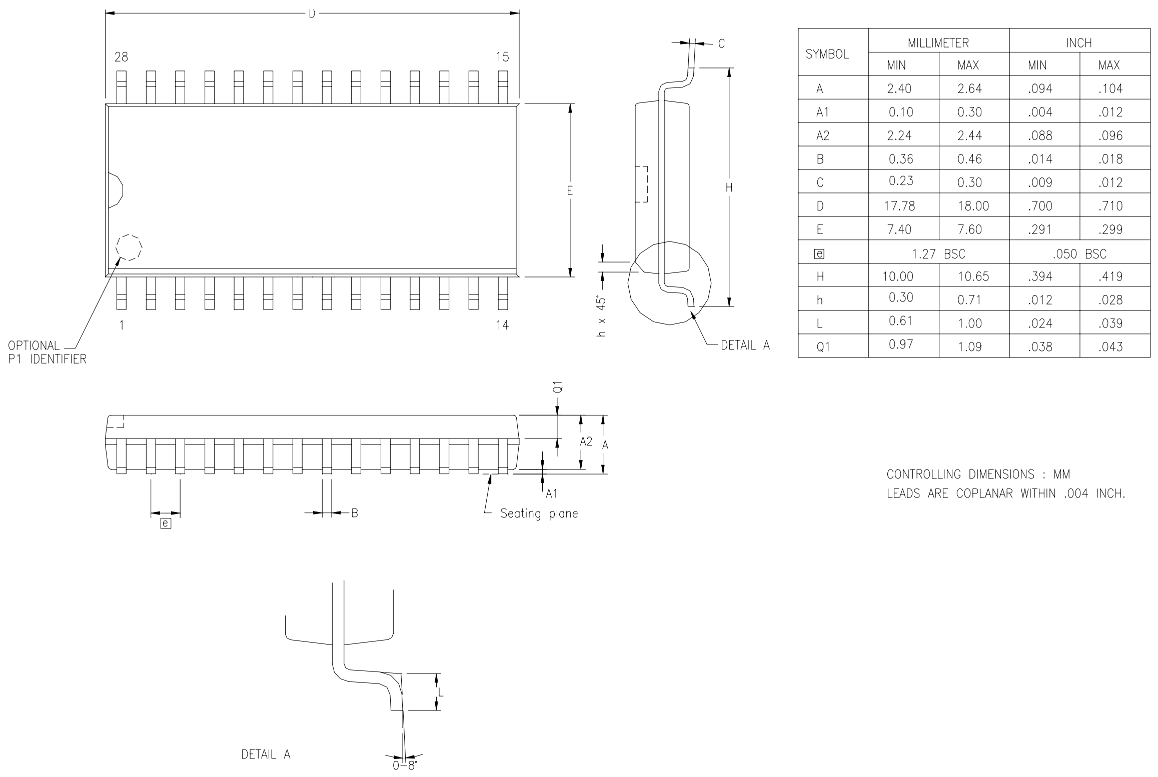
Table 118. Arithmetic Instructions

Mnemonic	Operands	Instruction
ADC	dst, src	Add with Carry
ADCX	dst, src	Add with Carry using Extended Addressing
ADD	dst, src	Add
ADDX	dst, src	Add using Extended Addressing
CP	dst, src	Compare
CPC	dst, src	Compare with Carry
CPCX	dst, src	Compare with Carry using Extended Addressing
CPX	dst, src	Compare using Extended Addressing
DA	dst	Decimal Adjust
DEC	dst	Decrement
DECW	dst	Decrement Word
INC	dst	Increment
INCW	dst	Increment Word
MULT	dst	Multiply

Table 126. eZ8 CPU Instruction Summary (Continued)

Assembly Mnemonic	Symbolic Operation	Address Mode dst	Address Mode src	Opcode(s) (Hex)	Flags C	Flags Z	Flags S	Flags V	Flags D	Flags H	Fetch Cycles	Instr. Cycles
LDX dst, src	dst ← src	r	ER	84	-	-	-	-	-	-	3	2
		Ir	ER	85							3	3
		R	IRR	86							3	4
		IR	IRR	87							3	5
		r	X(rr)	88							3	4
		X(rr)	r	89							3	4
		ER	r	94							3	2
		ER	Ir	95							3	3
		IRR	R	96							3	4
		IRR	IR	97							3	5
		ER	ER	E8							4	2
		ER	IM	E9							4	2
LEA dst, X(src)	dst ← src + X	r	X(r)	98	-	-	-	-	-	-	3	3
		rr	X(rr)	99							3	5
MULT dst	dst[15:0] ← dst[15:8] * dst[7:0]	RR		F4	-	-	-	-	-	-	2	8
NOP	No operation			0F	-	-	-	-	-	-	1	2
OR dst, src	dst ← dst OR src	r	r	42	-	*	*	0	-	-	2	3
		r	Ir	43							2	4
		R	R	44							3	3
		R	IR	45							3	4
		R	IM	46							3	3
		IR	IM	47							3	4
ORX dst, src	dst ← dst OR src	ER	ER	48	-	*	*	0	-	-	4	3
		ER	IM	49							4	3
POP dst	dst ← @SP SP ← SP + 1	R		50	-	-	-	-	-	-	2	2
		IR		51							2	3

Figure 62 displays the 28-pin SOIC package available for Z8 Encore! XP F0822 Series devices.



SYMBOL	MILLIMETER		INCH	
	MIN	MAX	MIN	MAX
A	2.40	2.64	.094	.104
A1	0.10	0.30	.004	.012
A2	2.24	2.44	.088	.096
B	0.36	0.46	.014	.018
C	0.23	0.30	.009	.012
D	17.78	18.00	.700	.710
E	7.40	7.60	.291	.299
e	1.27 BSC		.050 BSC	
H	10.00	10.65	.394	.419
h	0.30	0.71	.012	.028
L	0.61	1.00	.024	.039
Q1	0.97	1.09	.038	.043

CONTROLLING DIMENSIONS : MM
LEADS ARE COPLANAR WITHIN .004 INCH.

Figure 62. 28-Pin Small Outline Integrated Circuit Package (SOIC)