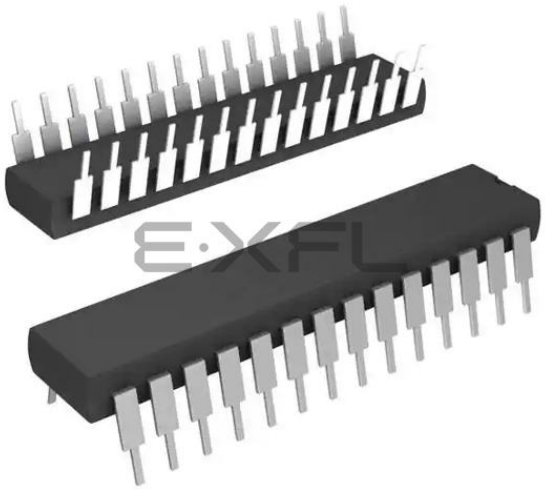


Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

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

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Obsolete
Core Processor	eZ8
Core Size	8-Bit
Speed	20MHz
Connectivity	I²C, IrDA, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	19
Program Memory Size	8KB (8K 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	28-DIP (0.600", 15.24mm)
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/zilog/z8f0812pj020sc

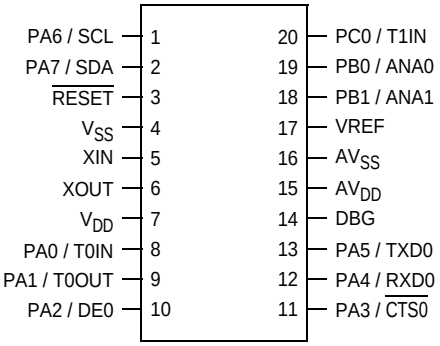


Figure 2. Z8F0821 and Z8F0421 in 20-Pin SSOP and PDIP Packages

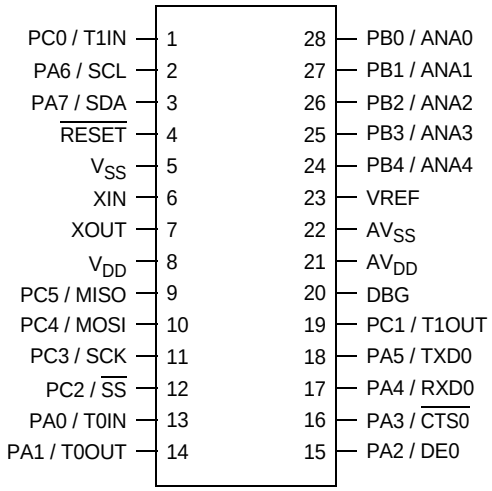


Figure 3. Z8F0822 and Z8F0422 in 28-Pin SOIC and PDIP Packages

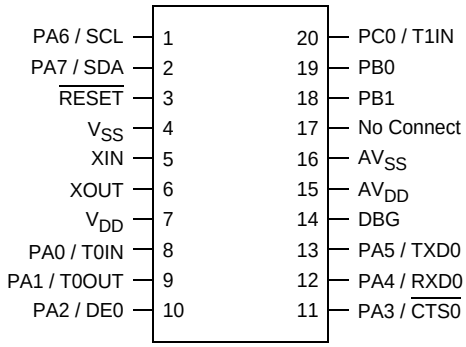


Figure 4. Z8F0811 and Z8F0411 in 20-Pin SSOP and PDIP Packages

Table 12. Port Alternate Function Mapping (Continued)

Port	Pin	Mnemonic	Alternate Function Description
Port C	PC0	T1IN	Timer 1 Input
	PC1	T1OUT	Timer 1 Output
	PC2	$\overline{SS}$	SPI Slave Select
	PC3	SCK	SPI Serial Clock
	PC4	MOSI	SPI Master Out Slave In
	PC5	MISO	SPI Master In Slave Out

GPIO Interrupts

Many of GPIO port pins are used as interrupt sources. Some port pins are configured to generate an interrupt request on either the rising edge or falling edge of the pin input signal. Other port pin interrupts generate an interrupt when any edge occurs (both rising and falling). For more details on interrupts using the GPIO pins, see GPIO Port Pin Block Diagram on page 48.

GPIO Control Register Definitions

Four registers for each port provide access to GPIO control, input data, and output data. Table 13 lists the GPIO Port Registers and Sub-Registers. Use the Port A–C Address and Control Registers together to provide access to sub-registers for Port configuration and control.

Table 13. GPIO Port Registers and Sub-Registers

Port Register Mnemonic	Port Register Name
P*x*ADDR	Port A–C Address Register (selects sub-registers)
P*x*CTL	Port A–C Control Register (provides access to sub-registers)
P*x*IN	Port A–C Input Data Register
P*x*OUT	Port A–C Output Data Register
Port Sub-Register Mnemonic	**Port Register Name**
P*x*DD	Data Direction
P*x*AF	Alternate Function

Port A–C Control Registers

The Port A–C Control Registers set the GPIO port operation. The value in the corresponding Port A–C Address Register determines the control sub-registers accessible using the Port A–C Control Register (Table 15).

Table 15. Port A–C Control Registers (P*x*CTL)

BITS	7	6	5	4	3	2	1	0
FIELD	PCTL							
RESET	00H							
R/W	R/W							
ADDR	FD1H, FD5H, FD9H							

PCTL[7:0]—Port Control
The Port Control Register provides access to all sub-registers that configure the GPIO Port operation.

Port A–C Data Direction Sub-Registers

The Port A–C Data Direction sub-register is accessed through the Port A–C Control register by writing 01H to the Port A–C Address Register (Table 16).

Table 16. Port A–C Data Direction Sub-Registers

BITS	7	6	5	4	3	2	1	0
FIELD	DD7	DD6	DD5	DD4	DD3	DD2	DD1	DD0
RESET	1							
R/W	R/W							
ADDR	If 01H in Port A–C Address Register, accessible through the Port A–C Control Register							

DD[7:0]—Data Direction

These bits control the direction of the associated port pin. Port Alternate Function operation overrides the Data Direction register setting.
0 = Output. Data in the Port A–C Output Data Register is driven onto the port pin.
1 = Input. The port pin is sampled and the value written into the Port A–C Input Data Register. The output driver is tri-stated.

Port A–C Alternate Function Sub-Registers

The Port A–C Alternate Function sub-register (Table 17) is accessed through the Port A–C Control Register by writing 02H to the Port A–C Address Register. The Port A–C Alternate Function sub-registers select the alternate functions for the selected

4. If required, enable the timer interrupt and set the timer interrupt priority by writing to the relevant interrupt registers.
5. Configure the associated GPIO port pin for the Timer Input alternate function.
6. If using the Timer Output function, configure the associated GPIO port pin for the Timer Output alternate function.
7. Write to the Timer Control Register to enable the timer.

In COUNTER mode, the number of Timer Input transitions since the timer start is given by the following equation:

COUNTER Mode Timer Input Transitions = Current Count Value – Start Value

PWM Mode

In PWM mode, the timer outputs a Pulse-Width Modulator output signal through a GPIO port pin. 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 then 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 then 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.

Follow the steps below for configuring a timer for PWM mode and initiating the PWM operation:

1. Write to the Timer Control Register to:
   – Disable the timer
   – Configure the timer for PWM mode.
   – Set the prescale value.
   – Set the initial logic level (High or Low) and PWM High/Low transition for the Timer Output alternate function.
2. Write to the Timer High and Low Byte Registers to set the starting count value (typically 0001H). This only affects the first pass in PWM mode. After the first timer reset in PWM mode, counting always begins at the reset value of 0001H.

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.

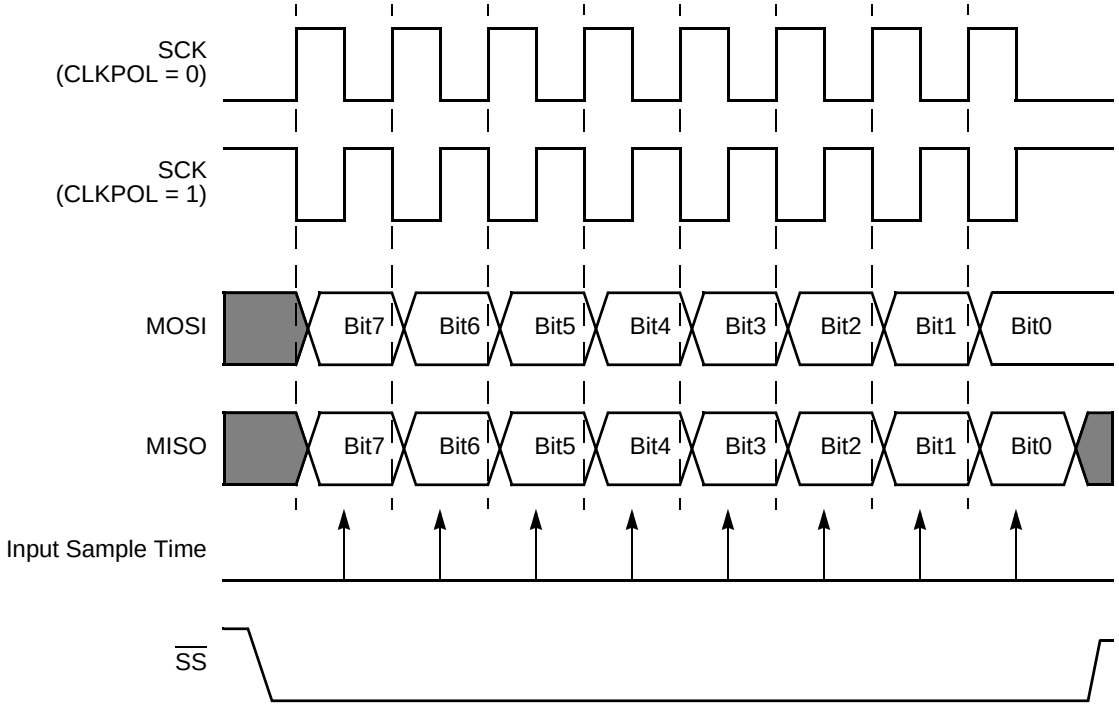


Figure 24. SPI Timing When PHASE is 1

Multi-Master Operation

In a multi-master SPI system, all SCK pins are tied together, all MOSI pins are tied together and all MISO pins are tied together. All SPI pins must then be configured in OPEN-DRAIN mode to prevent bus contention. At any one time, only one SPI device is configured as the Master and all other SPI devices on the bus are configured as Slaves. The Master enables a single Slave by asserting the $\overline{SS}$ pin on that Slave only. Then, the single Master drives data out its SCK and MOSI pins to the SCK and MOSI pins on the Slaves (including those which are not enabled). The enabled Slave drives data out its MISO pin to the MISO Master pin.

For a Master device operating in a multi-master system, if the $\overline{SS}$ pin is configured as an input and is driven Low by another Master, the COL bit is set to 1 in the SPI Status Register. The COL bit indicates the occurrence of a multi-master collision (mode fault error condition).

Slave Operation

The SPI block is configured for SLAVE mode operation by setting the SPIEN bit to 1 and the MMEN bit to 0 in the SPICTL Register and setting the SSIO bit to 0 in the SPIMODE Register. The IRQE, PHASE, CLKPOL, and WOR bits in the SPICTL Register and the

SPI Mode Register

The SPI Mode Register configures the character bit width and the direction and value of the $\overline{SS}$ pin.

Table 66. SPI Mode Register (SPIMODE)

BITS	7	6	5	4	3	2	1	0
FIELD	Reserved		DIAG	NUMBITS[2:0]			SSIO	SSV
RESET	0							
R/W	R		R/W					
ADDR	F63H							

Reserved—Must be 0

DIAG–Diagnostic Mode Control bit
This bit is for SPI diagnostics. Setting this bit allows the BRG value to be read using the SPIBRH and SPIBRL Register locations.
0 = Reading SPIBRH, SPIBRL returns the value in the SPIBRH and SPIBRL Registers
1 = Reading SPIBRH returns bits [15:8] of the SPI Baud Rate Generator; and reading SPIBRL returns bits [7:0] of the SPI Baud Rate Counter. The Baud Rate Counter High and Low byte values are not buffered.

Caution: *Take precautions if you are reading the values while BRG is counting.*

NUMBITS[2:0]—Number of Data Bits Per Character to Transfer
This field contains the number of bits to shift for each character transfer. See the SPI Data Register description for information on valid bit positions when the character length is less than 8-bits.
000 = 8 bits
001 = 1 bit
010 = 2 bits
011 = 3 bits
100 = 4 bits
101 = 5 bits
110 = 6 bits
111 = 7 bits

SSIO—Slave Select I/O
0 = $\overline{SS}$ pin configured as an input.
1 = $\overline{SS}$ pin configured as an output (MASTER mode only).

SSV—Slave Select Value
If SSIO = 1 and SPI configured as a Master:
0 = $\overline{SS}$ pin driven Low (0).

14. If more bytes remain to be sent, return to step 9.
15. Software responds by setting the STOP bit of the I2C Control Register (or START bit to initiate a new transaction). In the STOP case, software clears the TXI bit of the I2C Control Register at the same time.
16. The I2C 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 I2C Controller sends the STOP (or RESTART) condition to the I2C 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 I2C Control Register.
2. Software asserts the TXI bit of the I2C Control Register to enable Transmit interrupts.
3. The I2C interrupt asserts, because the I2C Data Register is empty (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 I2C Control Register.
6. The I2C Controller sends the START condition to the I2C Slave.
7. The I2C Controller loads the I2C Shift register with the contents of the I2C Data Register.
8. After one bit of address is shifted out by the SDA signal, the Transmit Interrupt is asserted.

5. After the first bit has been shifted out, a Transmit Interrupt is asserted.
6. Software responds by writing the lower eight bits of address to the I2C Data Register.
7. The I2C Controller completes shifting of the two address bits and a 0 (write).
8. If the I2C Slave acknowledges the first address byte by pulling the SDA signal low during the next high period of SCL, the I2C Controller sets the ACK bit in the I2C Status register. Continue with step 9.

   If the slave does not acknowledge the first address byte, the I2C Controller sets the NCKI bit and clears the ACK bit in the I2C Status register. Software responds to the Not Acknowledge interrupt by setting the STOP and FLUSH bits and clearing the TXI bit. The I2C Controller sends the STOP condition on the bus and clears the STOP and NCKI bits. The transaction is complete (ignore following steps).
9. The I2C Controller loads the I2C Shift register with the contents of the I2C Data Register (second address byte).
10. The I2C Controller shifts out the second address byte. After the first bit is shifted, the I2C Controller generates a Transmit Interrupt.
11. Software responds by setting the START bit of the I2C Control Register to generate a repeated START and by clearing the TXI bit.
12. Software responds by writing 11110B followed by the 2-bit Slave address and a 1 (read) to the I2C Data Register.
13. If only one byte is to be read, software sets the NAK bit of the I2C Control Register.
14. After the I2C Controller shifts out the 2nd address byte, the I2C Slave sends an acknowledge by pulling the SDA signal low during the next high period of SCL, the I2C Controller sets the ACK bit in the I2C Status register. Continue with step 15.

    If the slave does not acknowledge the second address byte, the I2C Controller sets the NCKI bit and clears the ACK bit in the I2C Status register. Software responds to the Not Acknowledge interrupt by setting the STOP and FLUSH bits and clearing the TXI bit. The I2C Controller sends the STOP condition on the bus and clears the STOP and NCKI bits. The transaction is complete (ignore the following steps).
15. The I2C Controller sends the repeated START condition.
16. The I2C Controller loads the I2C Shift register with the contents of the I2C Data Register (third address transfer).
17. The I2C Controller sends 11110B followed by the two most significant bits of the slave read address and a 1 (read).
18. The I2C Slave sends an acknowledge by pulling the SDA signal Low during the next high period of SCL.
    If the slave were to Not Acknowledge at this point (this should not happen because the slave did acknowledge the first two address bytes), software would respond by setting the STOP and FLUSH bits and clearing the TXI bit. The I2C Controller sends the

On-Chip Debugger

Z8 Encore! XP® F0822 Series products have an integrated On-Chip Debugger (OCD) that provides advanced debugging features including:

- Reading and writing of the Register File
- Reading and (Flash version only) writing of Program and Data Memory
- Setting of Breakpoints
- Executing eZ8 CPU instructions

Architecture

The OCD consists of four primary functional blocks: transmitter, receiver, autobaud generator, and debug controller. Figure 37 displays the architecture of the OCD.

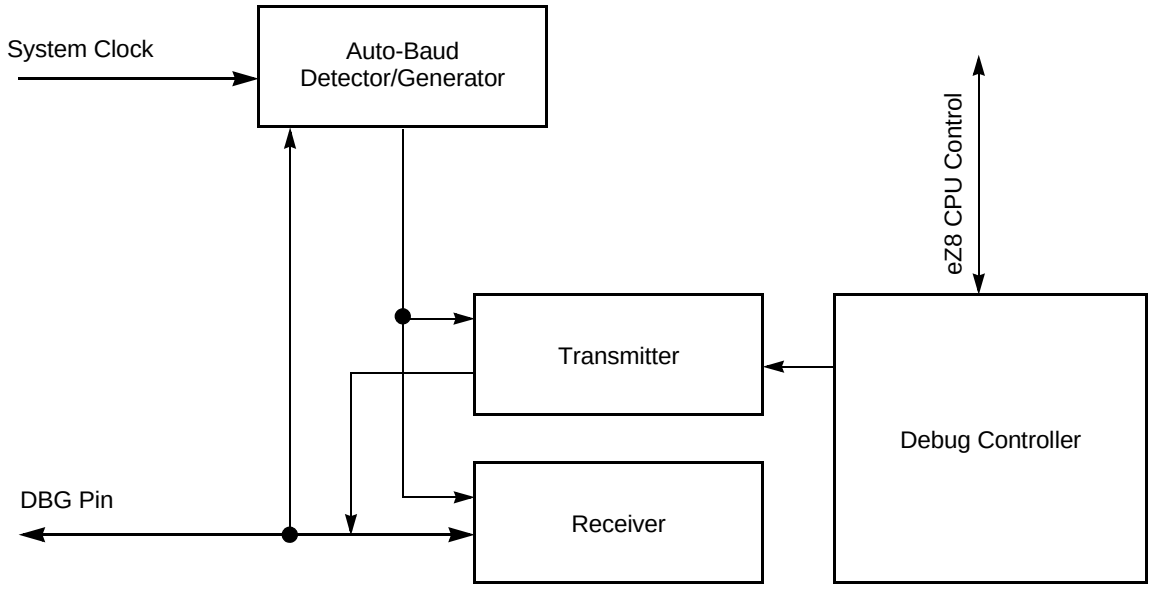


Figure 37. On-Chip Debugger Block Diagram

Operation

OCD Interface

The OCD uses the DBG pin for communication with an external host. This one-pin interface is a bi-directional open-drain interface that transmits and receives data. Data

Table 93. On-Chip Debugger Commands (Continued)

Debug Command	Command Byte	Enabled when NOT in DEBUG mode?	Disabled by Read Protect Option Bit
Execute Instruction	12H	-	Disabled
Reserved	13H - FFH	-	-

In the following bulleted list of OCD Commands, data and commands sent from the host to the OCD are identified by 'DBG ← Command/Data'. Data sent from the OCD back to the host is identified by 'DBG → Data'

- **Read OCD Revision (00H)**—The Read OCD Revision command determines the version of the OCD. 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)
```

- **Write OCD Counter Register (01H)**—The Write OCD Counter Register command writes the data that follows to the OCDCNTR register. If the device is not in DEBUG mode, the data is discarded.

```
DBG ← 01H
DBG ← OCDCNTR[15:8]
DBG ← OCDCNTR[7:0]
```

- **Read OCD Status Register (02H)**—The Read OCD Status Register command reads the OCDSTAT register.

```
DBG ← 02H
DBG → OCDSTAT[7:0]
```

- **Read OCD Counter Register (03H)**—The OCD Counter Register can be used to count system clock cycles in between Breakpoints, generate a BRK when it counts down to zero, or generate a BRK when its value matches the Program Counter. Since this register is really a down counter, the returned value is inverted when this register is read so the returned result appears to be an up counter. If the device is not in DEBUG mode, this command returns FFFFH.

```
DBG ← 03H
DBG → ~OCDCNTR[15:8]
DBG → ~OCDCNTR[7:0]
```

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

DC Characteristics

Table 97 lists the DC characteristics of the Z8 Encore! XP® F0822 Series products. All voltages are referenced to V_{SS}, the primary system ground.

Table 97. DC Characteristics

Symbol	Parameter	T_A = -40 °C to 105 °C			Units	Conditions
		Minimum	Typical	Maximum		
V_{DD}	Supply Voltage	2.7	–	3.6	V	
V_{IL1}	Low Level Input Voltage	-0.3	–	0.3*V_{DD}	V	For all input pins except $\overline{\text{RESET}}$, DBG, and XIN.
V_{IL2}	Low Level Input Voltage	-0.3	–	0.2*V_{DD}	V	For $\overline{\text{RESET}}$, DBG, and XIN.
V_{IH1}	High Level Input Voltage	0.7*V_{DD}	–	5.5	V	Ports A and C pins when their programmable pull-ups are disabled.
V_{IH2}	High Level Input Voltage	0.7*V_{DD}	–	V_{DD}+0.3	V	Port B pins. Ports A and C pins when their programmable pull-ups are enabled.
V_{IH3}	High Level Input Voltage	0.8*V_{DD}	–	V_{DD}+0.3	V	$\overline{\text{RESET}}$, DBG, and XIN pins.
V_{OL1}	Low Level Output Voltage	–	–	0.4	V	I_{OL} = 2 mA; VDD = 3.0 V High Output Drive disabled.
V_{OH1}	High Level Output Voltage	2.4	–	–	V	I_{OH} = -2 mA; VDD = 3.0 V High Output Drive disabled.
V_{OL2}	Low Level Output Voltage High Drive	–	–	0.6	V	I_{OL} = 20 mA; VDD = 3.3 V High Output Drive enabled T_A = -40 °C to +70 °C
V_{OH2}	High Level Output Voltage High Drive	2.4	–	–	V	I_{OH} = -20 mA; VDD = 3.3 V High Output Drive enabled; T_A = -40 °C to +70 °C
V_{OL3}	Low Level Output Voltage High Drive	–	–	0.6	V	I_{OL} = 15mA; VDD = 3.3 V High Output Drive enabled; T_A = +70 °C to +105 °C
V_{OH3}	High Level Output Voltage High Drive	2.4	–	–	V	I_{OH} = 15 mA; VDD = 3.3 V High Output Drive enabled; T_A = +70 °C to +105 °C

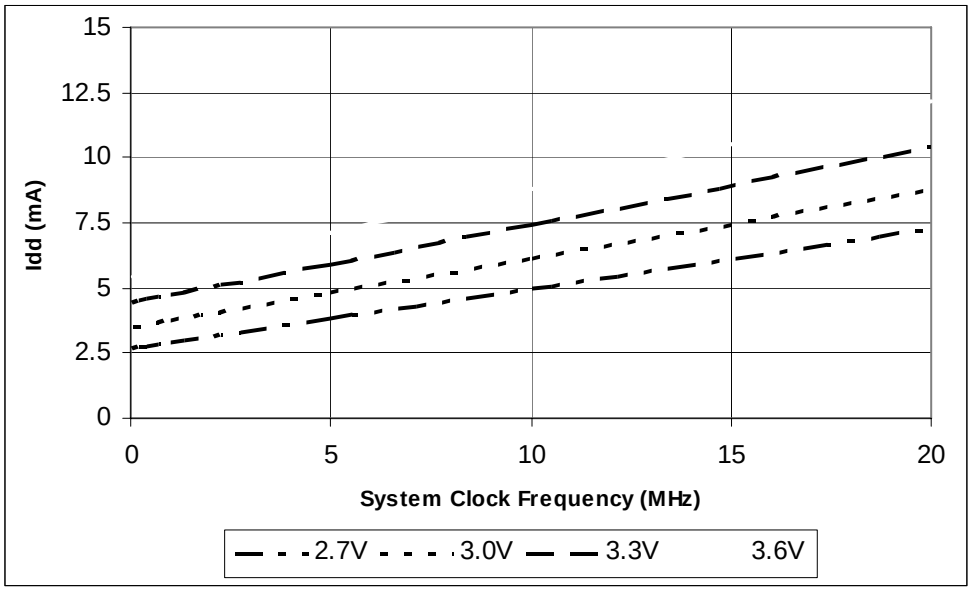


Figure 41. Typical Active Mode I_{DD} Versus System Clock Frequency

Figure 42 displays the maximum active mode current consumption across the full operating temperature range of the device and versus the system clock frequency. All GPIO pins are configured as outputs and driven High.

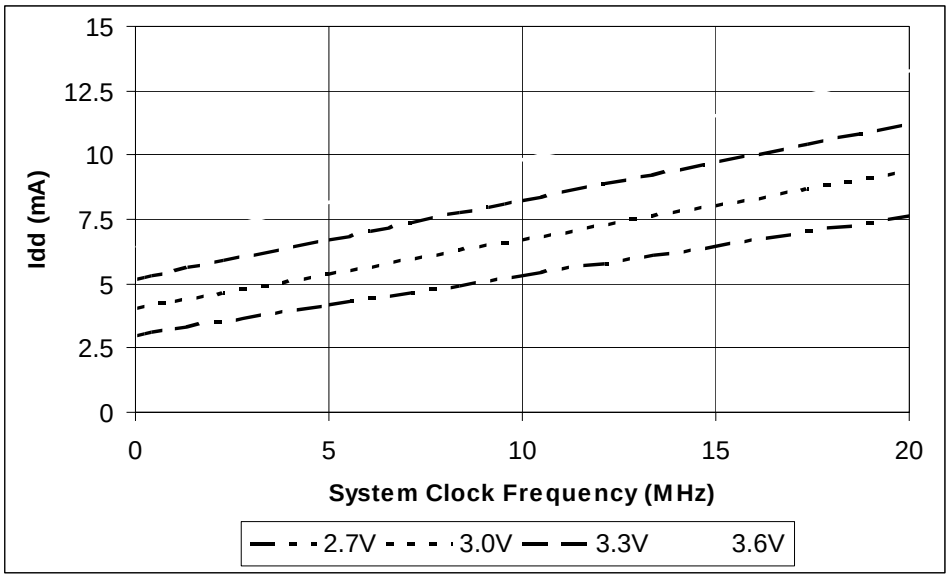


Figure 42. Maximum Active Mode I_{DD} Versus System Clock Frequency

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 117. 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 117. 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 126. eZ8 CPU Instruction Summary (Continued)

Assembly Mnemonic	Symbolic Operation	Address Mode dst	Address Mode src	Opcode(s) (Hex)	Flags C	Z	S	V	D	H	Fetch Cycles	Instr. Cycles
CALL dst	SP ← SP -2 @SP ← PC PC ← dst	IRR		D4	-	-	-	-	-	-	2	6
		DA		D6							3	3
CCF	C ← ~C			EF	*	-	-	-	-	-	1	2
CLR dst	dst ← 00H	R		B0	-	-	-	-	-	-	2	2
		IR		B1							2	3
COM dst	dst ← ~dst	R		60	-	*	*	0	-	-	2	2
		IR		61							2	3
CP dst, src	dst - src	r	r	A2	*	*	*	*	-	-	2	3
		r	Ir	A3							2	4
		R	R	A4							3	3
		R	IR	A5							3	4
		R	IM	A6							3	3
		IR	IM	A7							3	4
CPC dst, src	dst - src - C	r	r	1F A2	*	*	*	*	-	-	3	3
		r	Ir	1F A3							3	4
		R	R	1F A4							4	3
		R	IR	1F A5							4	4
		R	IM	1F A6							4	3
		IR	IM	1F A7							4	4
CPCX dst, src	dst - src - C	ER	ER	1F A8	*	*	*	*	-	-	5	3
		ER	IM	1F A9							5	3
CPX dst, src	dst - src	ER	ER	A8	*	*	*	*	-	-	4	3
		ER	IM	A9							4	3
DA dst	dst ← DA(dst)	R		40	*	*	*	X	-	-	2	2
		IR		41							2	3
DEC dst	dst ← dst - 1	R		30	-	*	*	*	-	-	2	2
		IR		31							2	3

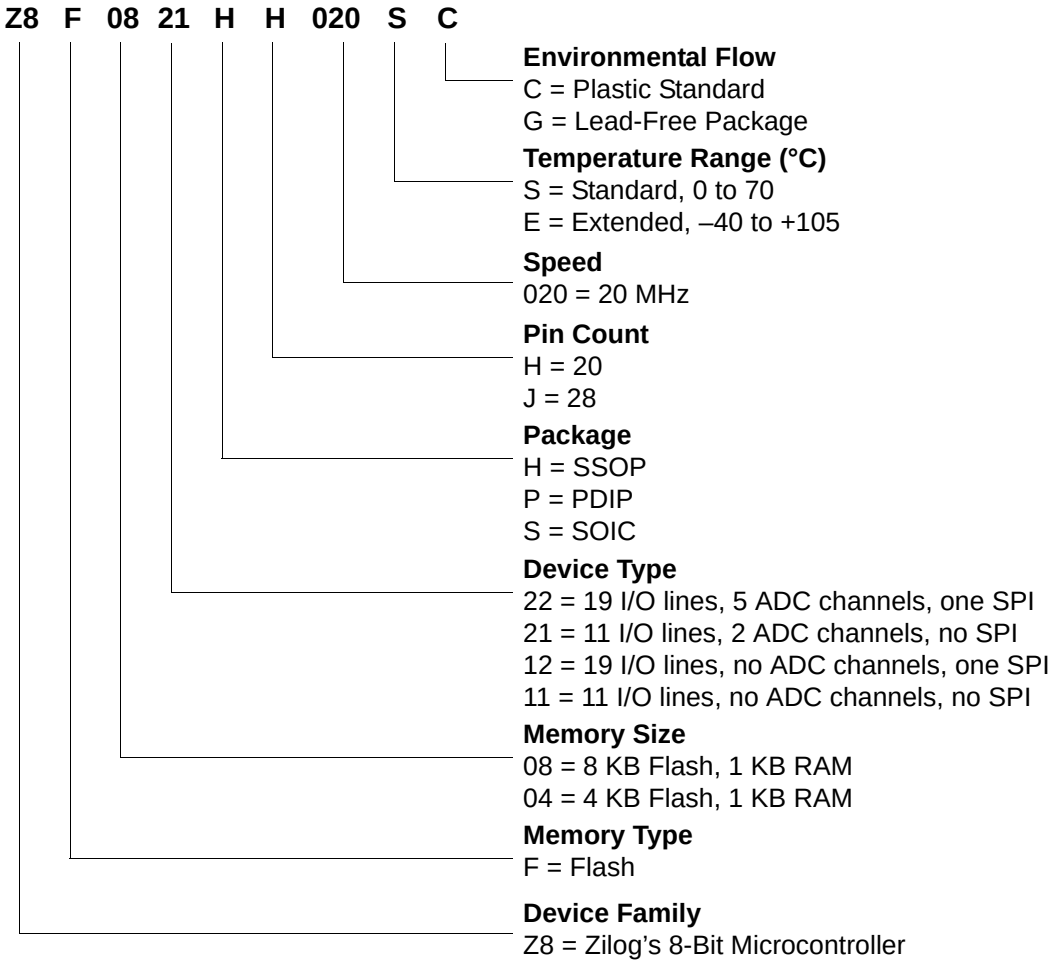
Table 126. eZ8 CPU Instruction Summary (Continued)

Assembly Mnemonic	Symbolic Operation	Address Mode dst	Address Mode src	Opcode(s) (Hex)	Flags C	Z	S	V	D	H	Fetch Cycles	Instr. Cycles
TM dst, src	dst AND src	r	r	72	-	*	*	0	-	-	2	3
		r	Ir	73							2	4
		R	R	74							3	3
		R	IR	75							3	4
		R	IM	76							3	3
		IR	IM	77							3	4
TMX dst, src	dst AND src	ER	ER	78	-	*	*	0	-	-	4	3
		ER	IM	79							4	3
TRAP Vector	SP ← SP – 2 @SP ← PC SP ← SP – 1 @SP ← FLAGS PC ← @Vector		Vector	F2	-	-	-	-	-	-	2	6
WDT				5F	-	-	-	-	-	-	1	2
XOR dst, src	dst ← dst XOR src	r	r	B2	-	*	*	0	-	-	2	3
		r	Ir	B3							2	4
		R	R	B4							3	3
		R	IR	B5							3	4
		R	IM	B6							3	3
		IR	IM	B7							3	4
XORX dst, src	dst ← dst XOR src	ER	ER	B8	-	*	*	0	-	-	4	3
		ER	IM	B9							4	3

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

Part Number	Flash	RAM	I/O Lines	Interrupts	16-Bit Timers w/PWM	10-Bit A/D Channels	I^2C	SPI	UARTs with IrDA	Description
Z8F08xx with 8 KB Flash										
Standard Temperature: 0 °C to 70 °C										
Z8F0811HH020SC	8 KB	1 KB	11	16	2	0	1	0	1	SSOP 20-pin package
Z8F0811PH020SC	8 KB	1 KB	11	16	2	0	1	0	1	PDIP 20-pin package
Z8F0812SJ020SC	8 KB	1 KB	19	19	2	0	1	1	1	SOIC 28-pin package
Z8F0812PJ020SC	8 KB	1 KB	19	19	2	0	1	1	1	PDIP 28-pin package
Extended Temperature: -40 °C to +105 °C										
Z8F0811HH020EC	8 KB	1 KB	11	16	2	0	1	0	1	SSOP 20-pin package
Z8F0811PH020EC	8 KB	1 KB	11	16	2	0	1	0	1	PDIP 20-pin package
Z8F0812SJ020EC	8 KB	1 KB	19	19	2	0	1	1	1	SOIC 28-pin package
Z8F0812PJ020EC	8 KB	1 KB	19	19	2	0	1	1	1	PDIP 28-pin package

Part Number Suffix Designations



For example, part number Z8F0821HH020SC is a Z8 Encore! XP Flash 8 KB microcontroller in a 20-pin SSOP package, operating with a maximum 20 MHz external clock frequency over a 0 ºC to +70 ºC temperature range and built using the Plastic-Standard environmental flow.

bit manipulation 215
block transfer 215
BRK 217
BSET 215
BSWAP 215, 218
BTJ 217
BTJNZ 217
BTJZ 217
CALL 217
CCF 215, 216
CLR 216
COM 217
CP 214
CPC 214
CPCX 214
CPU control 216
CPX 214
DA 214
DEC 214
DECW 214
DI 216
DJNZ 217
EI 216
HALT 216
INC 214
INCW 214
IRET 217
JP 217
LD 216
LDC 216
LDCI 215, 216
LDE 216
LDEI 215
LDX 216
LEA 216
load 216
logical 217
MULT 214
NOP 216
OR 217
ORX 217
POP 216
POPX 216
program control 217
PUSH 216
PUSHX 216
RCF 215, 216
RET 217
RL 218
RLC 218
rotate and shift 218
RR 218
RRC 218
SBC 215
SCF 215, 216
SRA 218
SRL 218
SRP 216
STOP 216
SUB 215
SUBX 215
SWAP 218
TCM 215
TCMX 215
TM 215
TMX 215
TRAP 217
watch-dog timer refresh 216
XOR 217
XORX 217
instructions, eZ8 classes of 214
interrupt control register 67
interrupt controller 5, 57
architecture 57
interrupt assertion types 60
interrupt vectors and priority 60
operation 59
register definitions 61
software interrupt assertion 60
interrupt edge select register 67
interrupt request 0 register 61
interrupt request 1 register 62
interrupt request 2 register 63
interrupt return 217
interrupt vector listing 57
interrupts
not acknowledge 128
receive 128

SPI 119
transmit 128
UART 97
introduction 1
IR 211
Ir 211
IrDA
architecture 109
block diagram 109
control register definitions 112
operation 109
receiving data 111
transmitting data 110
IRET 217
IRQ0 enable high and low bit registers 63
IRQ1 enable high and low bit registers 64
IRQ2 enable high and low bit registers 65
IRR 211
Irr 211

J

JP 217
jump, conditional, relative, and relative conditional 217

L

LD 216
LDC 216
LDCI 215, 216
LDE 216
LDEI 215, 216
LDX 216
LEA 216
load 216
load constant 215
load constant to/from program memory 216
load constant with auto-increment addresses 216
load effective address 216
load external data 216
load external data to/from data memory and auto-increment addresses 215
load external to/from data memory and auto-increment addresses 216
load instructions 216
load using extended addressing 216
logical AND 217
logical AND/extended addressing 217
logical exclusive OR 217
logical exclusive OR/extended addressing 217
logical instructions 217
logical OR 217
logical OR/extended addressing 217
low power modes 45

M

master interrupt enable 59
master-in, slave-out and-in 115
memory
program 13
MISO 115
mode
capture 81
capture/compare 82
continuous 81
counter 81
gated 82
one-shot 81
PWM 81
modes 82
MOSI 115
MULT 214
multiply 214
multiprocessor mode, UART 95

N

NOP (no operation) 216
not acknowledge interrupt 128
notation
b 211
cc 211
DA 211
ER 211
IM 211
IR 211