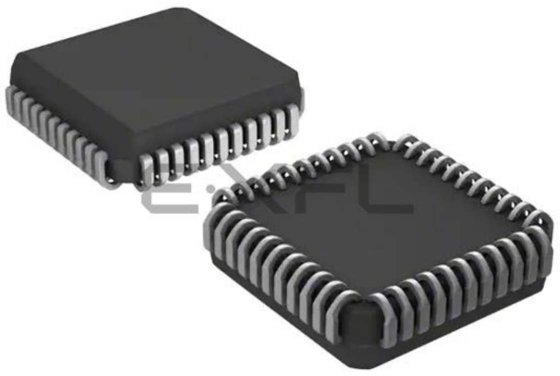
Zilog - Z8F6401VN020EC Datasheet



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, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, DMA, POR, PWM, WDT
Number of I/O	31
Program Memory Size	64KB (64K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	4K x 8
Voltage - Supply (Vcc/Vdd)	3V ~ 3.6V
Data Converters	A/D 8x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 105°C (TA)
Mounting Type	Surface Mount
Package / Case	44-LCC (J-Lead)
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/zilog/z8f6401vn020ec

- Power-On Reset (POR)
- 3.0-3.6V operating voltage with 5V-tolerant inputs
- 0° to +70°C standard temperature and -40° to +105°C extended temperature operating ranges

Part Selection Guide

Table 1 identifies the basic features and package styles available for each device within the Z8F640x family product line.

Table 1. Z8F640x Family Part Selection Guide

Part Number	Flash (KB)	RAM (KB)	I/O	16-bit Timers with PWM	ADC Inputs	UARTs with IrDA	I^2C	SPI	40/44-pin packages	64/68-pin packages	80-pin package
Z8F1601	16	2	31	3	8	2	1	1	X		
Z8F1602	16	2	46	4	12	2	1	1		X	
Z8F2401	24	2	31	3	8	2	1	1	X		
Z8F2402	24	2	46	4	12	2	1	1		X	
Z8F3201	32	2	31	3	8	2	1	1	X		
Z8F3202	32	2	46	4	12	2	1	1		X	
Z8F4801	48	4	31	3	8	2	1	1	X		
Z8F4802	48	4	46	4	12	2	1	1		X	
Z8F4803	48	4	60	4	12	2	1	1			X
Z8F6401	64	4	31	3	8	2	1	1	X		
Z8F6402	64	4	46	4	12	2	1	1		X	
Z8F6403	64	4	60	4	12	2	1	1			X

Signal and Pin Descriptions

Overview

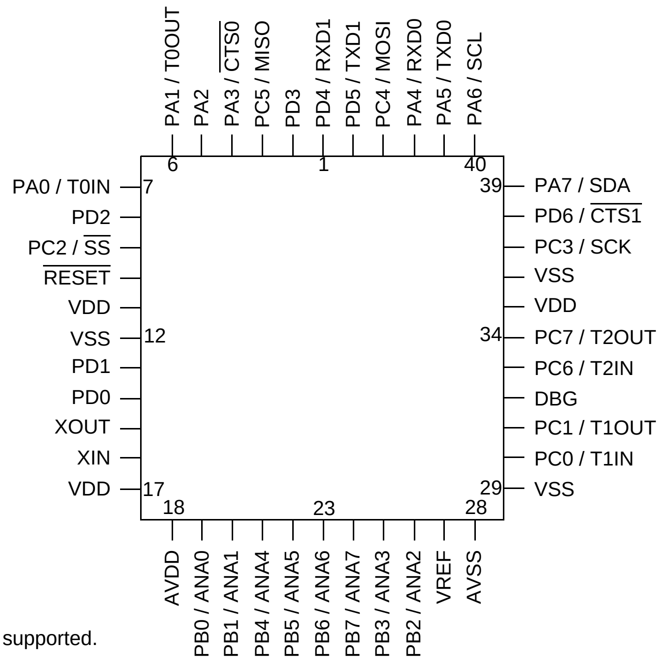
The Z8F640x family products are available in a variety of packages styles and pin configurations. This chapter describes the signals and available pin configurations for each of the package styles. For information regarding the physical package specifications, please refer to the chapter Packaging on page 206.

Available Packages

Table 2 identifies the package styles that are available for each device within the Z8F640x family product line.

Table 2. Z8F640x family Package Options

Part Number	40-pin PDIP	44-pin LQFP	44-pin PLCC	64-pin LQFP	68-pin PLCC	80-pin QFP
Z8F1601	X	X	X			
Z8F1602				X	X	
Z8F2401	X	X	X			
Z8F2402				X	X	
Z8F3201	X	X	X			
Z8F3202				X	X	
Z8F4801	X	X	X			
Z8F4802				X	X	
Z8F4803						X
Z8F6401	X	X	X			
Z8F6402				X	X	
Z8F6403						X



Note: Timer 3 is not supported.

Figure 57. Z8Fxx01 in 44-Pin Plastic Leaded Chip Carrier (PLCC)

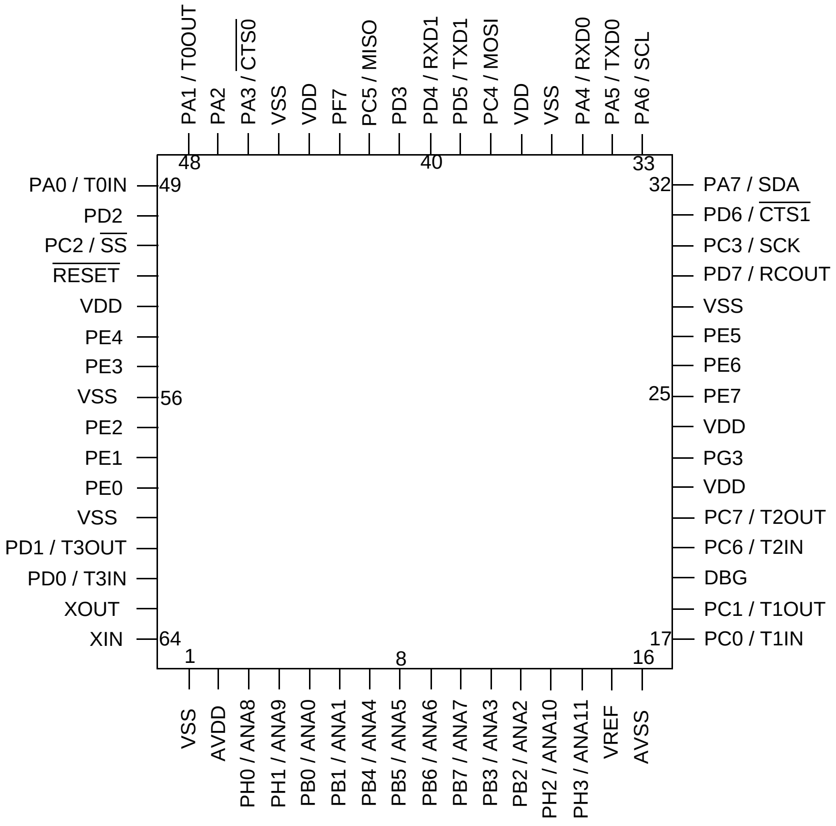


Figure 59. Z8Fxx02 in 64-Pin Low-Profile Quad Flat Package (LQFP)

Table 6. Register File Address Map (Continued)

Address (Hex)	Register Description	Mnemonic	Reset (Hex)	Page #
FCE	Interrupt Port Select	IRQPS	00	55
FCF	Interrupt Control	IRQCTL	00	56
GPIO Port A				
FD0	Port A Address	PAADDR	00	37
FD1	Port A Control	PACTL	00	38
FD2	Port A Input Data	PAIN	XX	42
FD3	Port A Output Data	PAOUT	00	43
GPIO Port B				
FD4	Port B Address	PBADDR	00	37
FD5	Port B Control	PBCTL	00	38
FD6	Port B Input Data	PBIN	XX	42
FD7	Port B Output Data	PBOUT	00	43
GPIO Port C				
FD8	Port C Address	PCADDR	00	37
FD9	Port C Control	PCCTL	00	38
FDA	Port C Input Data	PCIN	XX	42
FDB	Port C Output Data	PCOUT	00	43
GPIO Port D				
FDC	Port D Address	PDADDR	00	37
FDD	Port D Control	PDCTL	00	38
FDE	Port D Input Data	PDIN	XX	42
FDF	Port D Output Data	PDOUT	00	43
GPIO Port E				
FE0	Port E Address	PEADDR	00	37
FE1	Port E Control	PECTL	00	38
FE2	Port E Input Data	PEIN	XX	42
FE3	Port E Output Data	PEOUT	00	43
GPIO Port F				
FE4	Port F Address	PFADDR	00	37
FE5	Port F Control	PFCTL	00	38
FE6	Port F Input Data	PFIN	XX	42
FE7	Port F Output Data	PFOUT	00	43
GPIO Port G				
FE8	Port G Address	PGADDR	00	37
FE9	Port G Control	PGCTL	00	38
FEA	Port G Input Data	PGIN	XX	42
FEB	Port G Output Data	PGOUT	00	43
GPIO Port H				
FEC	Port H Address	PHADDR	00	37

XX=Undefined

If the Timer reaches `FFFFH`, the timer rolls over to `0000H` and continue counting.

The steps for configuring a timer for Compare mode and initiating the count are as follows:

1. Write to the Timer Control register to:
   - Disable the timer
   - Configure the timer for Compare mode.
   - Set the prescale value.
   - Set the initial logic level (High or Low) for the Timer Output alternate function, if desired.
2. Write to the Timer High and Low Byte registers to set the starting count value.
3. Write to the Timer Reload High and Low Byte registers to set the Compare value.
4. If desired, enable the timer interrupt and set the timer interrupt priority by writing to the relevant interrupt registers.
5. If using the Timer Output function, configure the associated GPIO port pin for the Timer Output alternate function.
6. Write to the Timer Control register to enable the timer and initiate counting.

In Compare mode, the system clock always provides the timer input. The Compare time is given by the following equation:

$$\textbf{Compare Mode Time (s)} = \frac{(\textbf{Compare Value} - \textbf{Start Value}) \times \textbf{Prescale}}{\textbf{System Clock Frequency (Hz)}}$$

Gated Mode

In Gated mode, the timer counts only when the Timer Input signal is in its active state (asserted), as determined by the `TPOL` bit in the Timer Control register. When the Timer Input signal is asserted, counting begins. A timer interrupt is generated when the Timer Input signal is deasserted or a timer reload occurs. To determine if a Timer Input signal deassertion generated the interrupt, read the associated GPIO input value and compare to the value stored in the `TPOL` bit.

The timer counts up to the 16-bit Reload value stored in the Timer Reload High and Low Byte registers. The timer input is the system clock. When 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 (assuming the Timer Input signal is still asserted). Also, if the Timer Output alternate function is enabled, the Timer Output pin changes state (from Low to High or from High to Low) at timer reset.

The steps for configuring a timer for Gated mode and initiating the count are as follows:

1. Write to the Timer Control register to:
   - Disable the timer

Timer 0-3 PWM High and Low Byte Registers

The Timer 0-3 PWM High and Low Byte (TxPWMH and TxPWML) registers (Tables 42 and 43) are used for Pulse-Width Modulator (PWM) operations. These registers also store the Capture values for the Capture and Capture/Compare modes.

Table 42. Timer 0-3 PWM High Byte Register (TxPWMH)

BITS	7	6	5	4	3	2	1	0
FIELD	PWMH							
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	F04H, F0CH, F14H, F1CH							

Table 43. Timer 0-3 PWM Low Byte Register (TxPWML)

BITS	7	6	5	4	3	2	1	0
FIELD	PWML							
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	F05H, F0DH, F15H, F1DH							

PWMH and PWML—Pulse-Width Modulator High and Low Bytes
These two bytes, {PWMH[7:0], PWML[7:0]}, form a 16-bit value that is compared to the current 16-bit timer count. When a match occurs, the PWM output changes state. The PWM output value is set by the `TPOL` bit in the Timer Control Register (TxCTL) register.

The TxPWMH and TxPWML registers also store the 16-bit captured timer value when operating in Capture or Capture/Compare modes.

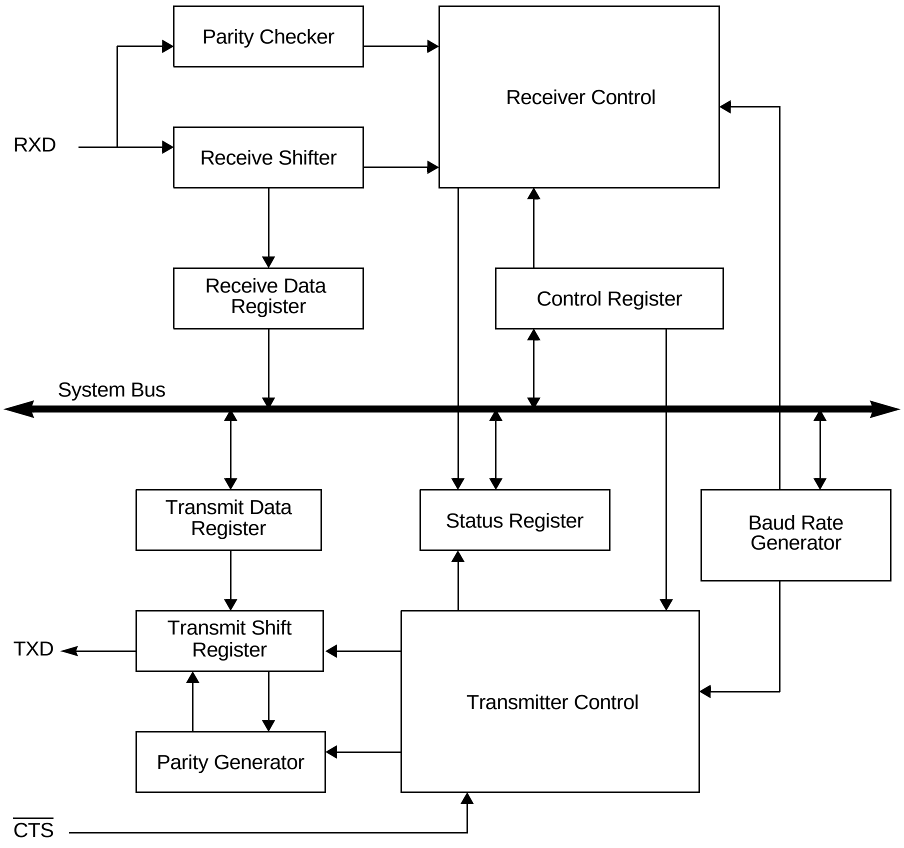


Figure 67. UART Block Diagram

Operation

Data Format

The UART always transmits and receives data in an 8-bit data format, least-significant bit first. An even or odd parity bit can be optionally added to the data stream. Each character begins with an active Low Start bit and ends with either 1 or 2 active High Stop bits.

Receiving Data using the Polled Method

Follow these steps to configure the UART for polled data reception:

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 receive enable bit (`REN`) to enable the UART for data reception
   - Enable parity, if desired, and select either even or odd parity.
5. Check the `RDA` bit in the UART Status 0 register to determine if the Receive Data register contains a valid data byte (indicated by a 1). If `RDA` is set to 1 to indicate available data, continue to Step 6. If the Receive Data register is empty (indicated by a 0), continue to monitor the RDA bit awaiting reception of the valid data.
6. Read data from the UART Receive Data register. If operating in Multiprocessor (9-bit) mode, first read the Multiprocessor Receive flag (`MPRX`) to determine if the data was directed to this UART before reading the data.
7. Return to Step 6 to receive additional data.

Receiving Data using the Interrupt-Driven Method

The UART Receiver interrupt indicates the availability of new data (as well as error conditions). Follow these steps to configure the UART receiver for interrupt-driven 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. Execute a DI instruction to disable interrupts.
4. Write to the Interrupt control registers to enable the UART Receiver interrupt and set the desired priority.
5. Clear the UART Receiver interrupt in the applicable Interrupt Request register.
6. Write to the UART Control 1 register to enable Multiprocessor (9-bit) mode functions, if desired.

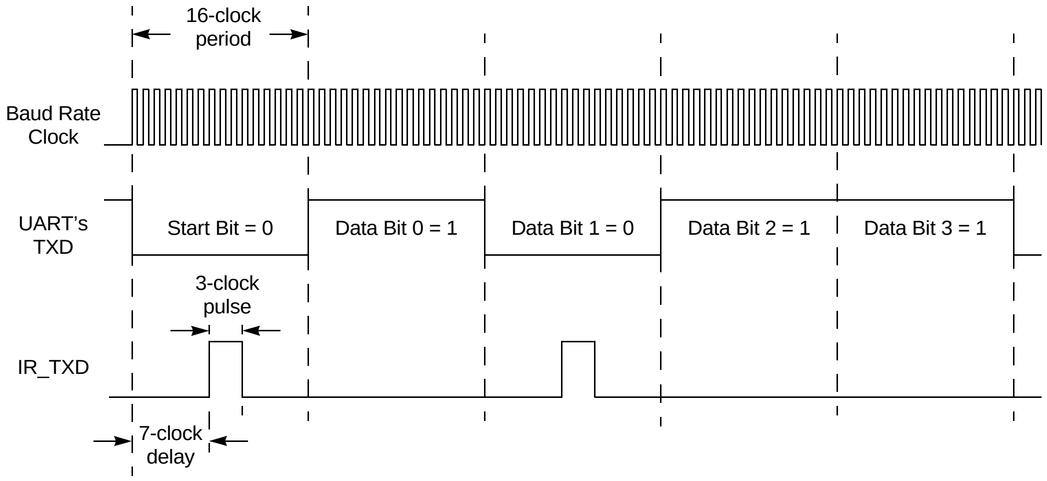


Figure 72. Infrared Data Transmission

Receiving IrDA Data

Data received from the infrared transceiver via the IR_RXD signal through the RXD pin is decoded by the Infrared Endec and passed to the UART. The UART's baud rate clock is used by the Infrared Endec to generate the demodulated signal (RXD) that drives the UART. Each UART/Infrared data bit is 16-clocks wide. Figure 73 illustrates data reception. When the Infrared Endec is enabled, the UART's RXD signal is internal to the Z8F640x family device while the IR_RXD signal is received through the RXD pin.

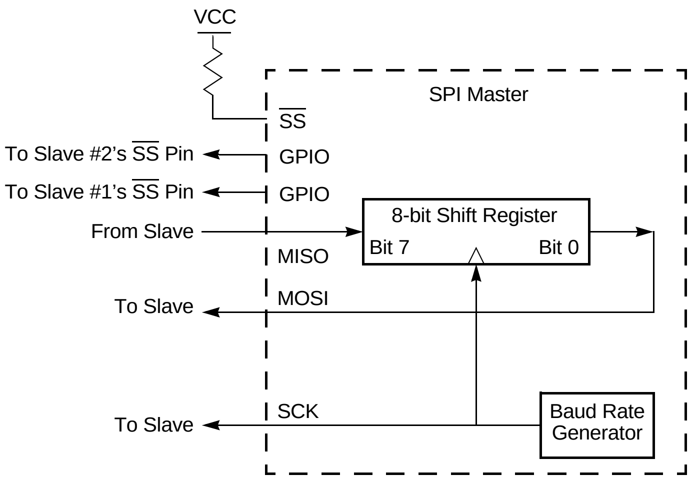


Figure 75. SPI Configured as a Master in a Single Master, Multiple Slave System

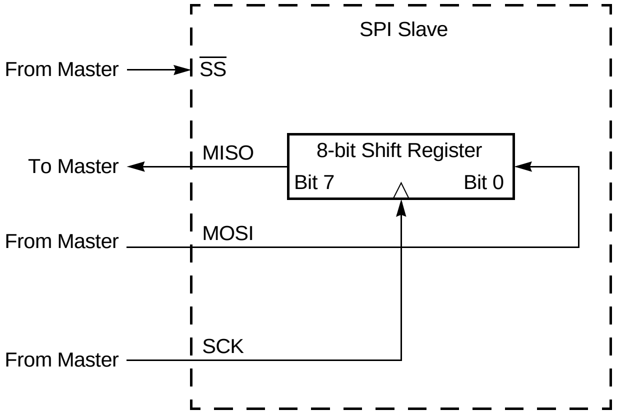


Figure 76. SPI Configured as a Slave

Operation

The SPI is a full-duplex, synchronous, character-oriented channel that supports a four-wire interface (serial clock, transmit, receive and Slave select). The SPI block consists of trans-

SPIEN—SPI Enable
0 = SPI disabled.
1 = SPI enabled.

SPI Status Register

The SPI Status register indicates the current state of the SPI.

Table 62. SPI Status Register (SPISTAT)

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

IRQ—Interrupt Request
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.

Reserved
These bits are reserved and 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.

2. The I^2C Controller waits for the slave to send an Acknowledge (by pulling the SDA signal Low). If the slave pulls the SDA signal High (Not-Acknowledge), the I^2C Controller sends a Stop signal.
3. If the slave needs to service an interrupt, it pulls the SCL signal Low, which halts I^2C operation.
4. If there is no other data in the I^2C Data register or the `STOP` bit in the I^2C Control register is set by software, then the Stop signal is sent.

Figure 79 illustrates the data transfer format for a 7-bit addressed slave. Shaded regions indicate data transferred from the I^2C Controller to slaves and unshaded regions indicate data transferred from the slaves to the I^2C Controller.

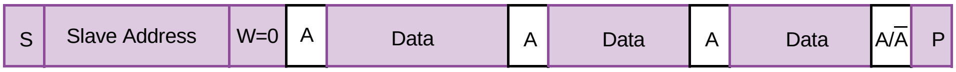


Figure 79. 7-Bit Addressed Slave Data Transfer Format

The data transfer format for a transmit operation on a 7-bit addressed slave is as follows:

1. Software asserts the `IEN` bit in the I^2C Control register.
2. Software asserts the `TXI` bit of the I^2C Control register to enable Transmit interrupts.
3. The I^2C interrupt asserts, because the I^2C Data register is empty
4. Software responds to the `TDRE` bit by writing a 7-bit slave address followed by a 0 (write) to the I^2C Data register.
5. Software asserts the START bit of the I^2C Control register.
6. The I^2C Controller sends the START condition to the I^2C slave.
7. The I^2C Controller loads the I^2C Shift register with the contents of the I^2C Data register.
8. After one bit of address has been shifted out by the SDA signal, the Transmit interrupt is asserted.
9. Software responds by writing the contents of the data into the I^2C Data register.
10. The I^2C Controller shifts the rest of the address and write bit out by the SDA signal.
11. The I^2C slave sends an acknowledge (by pulling the SDA signal low) during the next high period of SCL. The I^2C Controller sets the `ACK` bit in the I^2C Status register.
12. The I^2C Controller loads the contents of the I^2C Shift register with the contents of the I^2C Data register.
13. The I^2C Controller shifts the data out of via the SDA signal. After the first bit is sent, the Transmit interrupt is asserted.

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.

this bit to 0 when a conversion has been completed.
1 = Begin conversion. Writing a 1 to this bit starts a conversion. If a conversion is already in progress, the conversion restarts. This bit remains 1 until the conversion is complete.

Reserved
This bit is reserved and must be 0.

$\overline{\text{VREF}}$
0 = Internal voltage reference generator enabled. The VREF pin should be left unconnected (or capacitively coupled to analog ground).
1 = Internal voltage reference generator disabled. An external voltage reference must be provided through the VREF pin.

CONT
0 = Single-shot conversion. ADC data is output once at completion of the 5129 system clock cycles.
1 = Continuous conversion. ADC data updated every 256 system clock cycles.

ANAIN—Analog Input Select
These bits select the analog input for conversion. Not all Port pins in this list are available in all packages for the Z8F640x family of products. Refer to the **Signal and Pin Descriptions** chapter for information regarding the Port pins available with each package style. Do not enable unavailable analog inputs.
0000 = ANA0
0001 = ANA1
0010 = ANA2
0011 = ANA3
0100 = ANA4
0101 = ANA5
0110 = ANA6
0111 = ANA7
1000 = ANA8
1001 = ANA9
1010 = ANA10
1011 = ANA11
11XX = Reserved.

ADC Data High Byte Register

The ADC Data High Byte register contains the upper eight bits of the 10-bit ADC output. During a conversion, this value is invalid. Access to the ADC Data High Byte register is read-only. The full 10-bit ADC result is given by {ADCD_H[7:0], ADCD_L[7:6]}.

Table 81. ADC Data High Byte Register (ADCD_H)

BITS	7	6	5	4	3	2	1	0
FIELD	ADCD_H							
RESET	X							
R/W	R							
ADDR	F72H							

ADCD_H—ADC Data High Byte
This byte contains the upper eight bits of the 10-bit ADC output. These bits are not valid during a conversion. These bits are undefined after a Reset.

ADC Data Low Bits Register

The ADC Data Low Bits register contains the lower two bits of the conversion value. During a conversion this value is invalid. Access to the ADC Data Low Bits register is read-only. The full 10-bit ADC result is given by {ADCD_H[7:0], ADCD_L[7:6]}.

Table 82. ADC Data Low Bits Register (ADCD_L)

BITS	7	6	5	4	3	2	1	0
FIELD	ADCD_L		Reserved					
RESET	X		X					
R/W	R		R					
ADDR	F73H							

ADCD_L—ADC Data Low Bits
These are the least significant two bits of the 10-bit ADC output. During a conversion, this value is invalid. These bits are undefined after a Reset.

Reserved
These bits are reserved and are always undefined.

Flash Code Protection Using the Option Bits

The FHSWP and FWP Option Bits combine to provide three levels of Flash Program Memory protection as listed in Table 84. Refer to the **Option Bits** chapter for more information.

Table 84. Flash Code Protection Using the Option Bits

FHSWP	FWP	Flash Code Protection Description
0	0	Programming and erasure disabled for all of Flash Program Memory. In user code programming, Page Erase, and Mass Erase are all disabled. Mass Erase is available through the On-Chip Debugger.
1	0	Programming and Page Erase are enabled for the High Sector of the Flash 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 Flash 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 Flash Program Memory.

Flash Code Protection Using the Flash Controller

At Reset, the Flash Controller locks to prevent accidental program or erasure of the Flash memory. To program or erase the Flash memory, unlock the Flash Controller by making two consecutive writes to the Flash Control register with the values 73H and 8CH, sequentially. After unlocking the Flash Controller, the Flash can be programmed or erased. When the Flash Controller is unlocked, any value written to the Flash Control register locks the Flash Controller. Writing the Mass Erase or Page Erase commands executes the function before locking the Flash Controller.

Byte Programming

When the Flash Controller is unlocked, all writes to Program Memory program a byte into the Flash. An erased Flash byte contains all 1's (FFH). The programming operation can only be used to change bits from 1 to 0. To change a Flash bit (or multiple bits) from 0 to 1 requires execution of either the Page Erase or Mass Erase commands.

Byte Programming can be accomplished using the On-Chip Debugger's Write Memory command or eZ8 CPU execution of the LDC or LDCI instructions. Refer to the **eZ8 CPU User Manual** for a description of the LDC and LDCI instructions. While the Flash Controller programs the Flash memory, the eZ8 CPU idles but the system clock and on-chip peripherals continue to operate. To exit programming mode and lock the Flash, write any value to the Flash Control register, except the Mass Erase or Page Erase commands.

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).

Opcode Maps

Figures 101 and 102 provide information on each of the eZ8 CPU instructions. A description of the opcode map data and the abbreviations are provided in Figure 100 and Table 127.

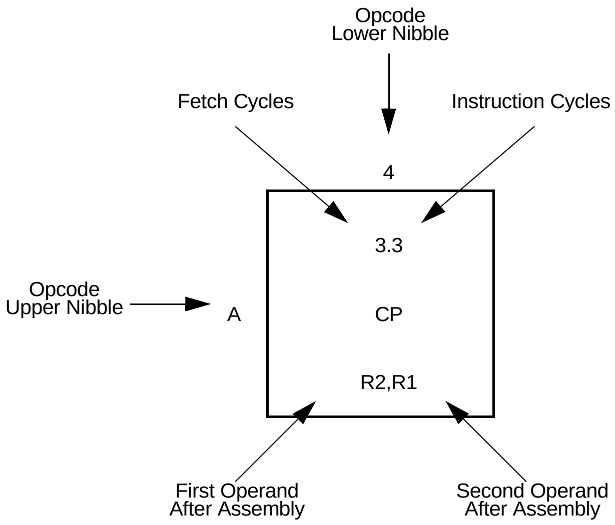


Figure 100. Opcode Map Cell Description

Lower Nibble (Hex)

Upper Nibble (Hex)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0																
1																
2																
3																
4																
5																
6																
7																
8																
9																
A			3.3 **CPC** r1,r2	3.4 **CPC** r1,Ir2	4.3 **CPC** R2,R1	4.4 **CPC** IR2,R1	4.3 **CPC** R1,IM	4.4 **CPC** IR1,IM	5.3 **CPCX** ER2,ER1	5.3 **CPCX** IM,ER1						
B																
C	3.2 **SRL** R1	3.3 **SRL** IR1														
D																
E																
F																

Figure 102. Second Opcode Map after 1FH