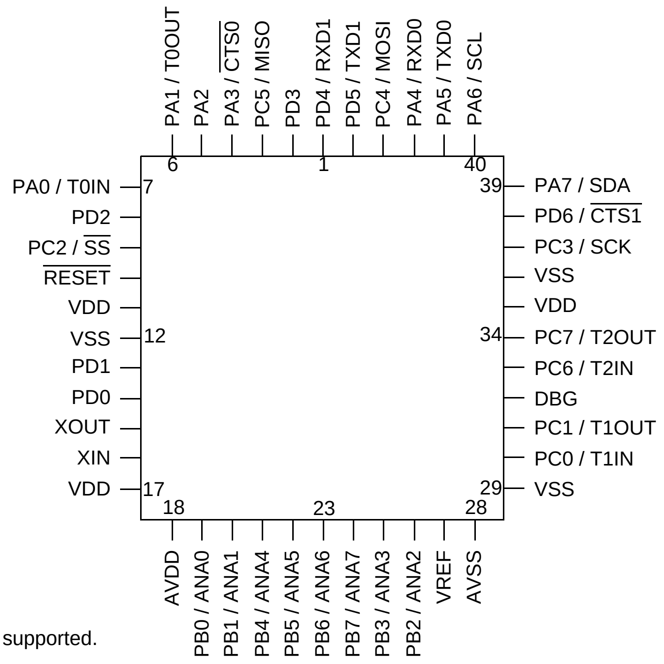
What is "Embedded - Microcontrollers"?

"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	46
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 12x10b
Oscillator Type	Internal
Operating Temperature	0°C ~ 70°C (TA)
Mounting Type	Surface Mount
Package / Case	64-LQFP
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/zilog/z8f3202ar020sc00tr



Note: Timer 3 is not supported.

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

Table 2. Signal Descriptions (Continued)

Signal Mnemonic	I/O	Description
Reset		
$\overline{\text{RESET}}$	I	RESET. Generates a Reset when asserted (driven Low).
Power Supply		
VDD	I	Power Supply.
AVDD	I	Analog Power Supply.
VSS	I	Ground.
AVSS	I	Analog Ground.

Pin Characteristics

Table 3 provides detailed information on the characteristics for each pin available on the Z8F640x family products. Data in Table 3 is sorted alphabetically by the pin symbol mnemonic.

Table 3. Pin Characteristics of the Z8F640x family

Symbol Mnemonic	Direction	Reset Direction	Active Low or Active High	Tri-State Output	Internal Pull-up or Pull-down	Schmitt Trigger Input	Open Drain Output
AVSS	N/A	N/A	N/A	N/A	No	No	N/A
AVDD	N/A	N/A	N/A	N/A	No	No	N/A
DBG	I/O	I	N/A	Yes	No	Yes	Yes
VSS	N/A	N/A	N/A	N/A	No	No	N/A
PA[7:0]	I/O	I	N/A	Yes	No	Yes	Yes, Programmable
PB[7:0]	I/O	I	N/A	Yes	No	Yes	Yes, Programmable
PC[7:0]	I/O	I	N/A	Yes	No	Yes	Yes, Programmable
PD[7:0]	I/O	I	N/A	Yes	No	Yes	Yes, Programmable
PE7:0]	I/O	I	N/A	Yes	No	Yes	Yes, Programmable

x represents integer 0, 1,... to indicate multiple pins with symbol mnemonics that differ only by the integer

Interrupt Controller

Overview

The interrupt controller on the Z8F640x family device prioritizes the interrupt requests from the on-chip peripherals and the GPIO port pins. The features of the interrupt controller on the Z8F640x family device include the following:

- 24 unique interrupt vectors:
  - 12 GPIO port pin interrupt sources
  - 12 on-chip peripheral interrupt sources
- Flexible GPIO interrupts
  - 8 selectable rising and falling edge GPIO interrupts
  - 4 dual-edge interrupts
- 3 levels of individually programmable interrupt priority
- Watch-Dog Timer can be configured to generate an interrupt

Interrupt requests (IRQs) allow peripheral devices to suspend CPU operation in an orderly manner and force the CPU to start an interrupt service routine (ISR). Usually this interrupt service routine is involved with the exchange of data, status information, or control information between the CPU and the interrupting peripheral. When the service routine is completed, the CPU returns to the operation from which it was interrupted.

The eZ8 CPU supports both vectored and polled interrupt handling. For polled interrupts, the interrupt control has no effect on operation. Refer to the *eZ8 CPU User Manual* for more information regarding interrupt servicing by the eZ8 CPU. The *eZ8 CPU User Manual* is available for download at www.zilog.com.

Interrupt Vector Listing

Table 22 lists all of the interrupts available on the Z8F640x family device in order of priority. The interrupt vector is stored with the most significant byte (MSB) at the even Program Memory address and the least significant byte (LSB) at the following odd Program Memory address.

Table 30. IRQ1 Enable High Bit Register (IRQ1ENH)

BITS	7	6	5	4	3	2	1	0
FIELD	PAD7ENH	PAD6ENH	PAD5ENH	PAD4ENH	PAD3ENH	PAD2ENH	PAD1ENH	PAD0ENH
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	FC4H							

PAD*x*ENH—Port A or Port D Bit[*x*] Interrupt Request Enable High Bit
Refer to the Interrupt Port Select register for selection of either Port A or Port D as the interrupt source.

Table 31. IRQ1 Enable Low Bit Register (IRQ1ENL)

BITS	7	6	5	4	3	2	1	0
FIELD	PAD7ENL	PAD6ENL	PAD5ENL	PAD4ENL	PAD3ENL	PAD2ENL	PAD1ENL	PAD0ENL
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	FC5H							

PAD*x*ENL—Port A or Port D Bit[*x*] Interrupt Request Enable Low Bit
Refer to the Interrupt Port Select register for selection of either Port A or Port D as the interrupt source.

IRQ2 Enable High and Low Bit Registers

The IRQ2 Enable High and Low Bit registers (Tables 33 and 34) form a priority encoded enabling for interrupts in the Interrupt Request 2 register. Priority is generated by setting bits in each register. Table 32 describes the priority control for IRQ2.

Table 32. IRQ2 Enable and Priority Encoding

IRQ2ENH[*x*]	IRQ2ENL[*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.

   – Configure the timer for Gated mode.
   – Set the prescale value.

2. Write to the Timer High and Low Byte registers to set the starting count value. This only affects the first pass in Gated mode. After the first timer reset in Gated mode, counting always begins at the reset value of `0001H`.
3. Write to the Timer Reload High and Low Byte registers to set the Reload value.
4. If desired, 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. Write to the Timer Control register to enable the timer.
7. Assert the Timer Input signal to initiate the counting.

Capture/Compare Mode

In Capture/Compare mode, the timer begins counting on the *first* external Timer Input transition. The desired transition (rising edge or falling edge) is set by the `TPOL` bit in the Timer Control Register. The timer input is the system clock.

Every subsequent desired transition (after the first) of the Timer Input signal captures the current count value. The Capture value is written to the Timer PWM High and Low Byte Registers. When the Capture event occurs, an interrupt is generated, the count value in the Timer High and Low Byte registers is reset to `0001H`, and counting resumes.

If no Capture event occurs, the timer counts up to the 16-bit Compare value stored in the Timer Reload High and Low Byte registers. Upon reaching the Compare value, the timer generates an interrupt, the count value in the Timer High and Low Byte registers is reset to 0001H and counting resumes.

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

1. Write to the Timer Control register to:
   – Disable the timer
   – Configure the timer for Capture/Compare mode.
   – Set the prescale value.
   – Set the Capture edge (rising or falling) for the Timer Input.
2. Write to the Timer High and Low Byte registers to set the starting count value (typically `0001H`).
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.

Watch-Dog Timer Control Register Definitions

Watch-Dog Timer Control Register

The Watch-Dog Timer Control (WDTCTL) register, detailed in Table 46, is a Read-Only register that indicates the source of the most recent Reset event, indicates a Stop Mode Recovery event, and indicates a Watch-Dog Timer time-out. Reading this register resets the upper four bits to 0.

Writing the `55H`, `AAH` unlock sequence to the Watch-Dog Timer Control (WDTCTL) register address unlocks the three Watch-Dog Timer Reload Byte registers (WDTU, WDTH, and WDTL) to allow changes to the time-out period. These write operations to the WDTCTL register address produce no effect on the bits in the WDTCTL register. The locking mechanism prevents spurious writes to the Reload registers.

Table 46. Watch-Dog Timer Control Register (WDTCTL)

BITS	7	6	5	4	3	2	1	0
FIELD	POR	STOP	WDT	EXT	Reserved			
RESET	X	X	X	0	0	0	0	0
R/W	R	R	R	R	R	R	R	R
ADDR	FF0							

POR—Power-On Reset Indicator
If this bit is set to 1, a Power-On Reset event occurred. This bit is reset to 0 if a WDT time-out or Stop Mode Recovery occurs. This bit is also reset to 0 when the register is read.

STOP—STOP Mode Recovery Indicator
If this bit is set to 1, a STOP Mode Recovery occurred. If the STOP and WDT bits are both set to 1, the STOP Mode Recovery occurred due to a WDT time-out. If the `STOP` bit is 1 and the `WDT` bit is 0, the STOP Mode Recovery was not caused by a WDT time-out. This bit is reset by a Power-On Reset or a WDT time-out that occurred while not in STOP mode. Reading this register also resets this bit.

WDT—Watch-Dog Timer Time-Out Indicator
If this bit is set to 1, a WDT time-out occurred. A Power-On Reset resets this pin. A Stop Mode Recovery from a change in an input pin also resets this bit. Reading this register resets this bit.

EXT—External Reset Indicator
If this bit is set to 1, a Reset initiated by the external $\overline{\text{RESET}}$ pin occurred. A Power-On Reset or a Stop Mode Recovery from a change in an input pin resets this bit. Reading this register resets this bit.

UART

Overview

The Universal Asynchronous Receiver/Transmitter (UART) is a full-duplex communication channel capable of handling asynchronous data transfers. The Z8F640x family device contains two fully independent UARTs. The UART uses a single 8-bit data mode with selectable parity. Features of the UART include:

- 8-bit asynchronous data transfer
- Selectable even- and odd-parity generation and checking
- Option of one or two Stop bits
- Separate transmit and receive interrupts
- Framing, parity, overrun and break detection
- Separate transmit and receive enables
- Selectable 9-bit multiprocessor (9-bit) mode
- 16-bit Baud Rate Generator (BRG)

Architecture

The UART consists of three primary functional blocks: transmitter, receiver, and baud rate generator. The UART's transmitter and receiver function independently, but employ the same baud rate and data format. Figure 67 illustrates the UART architecture.

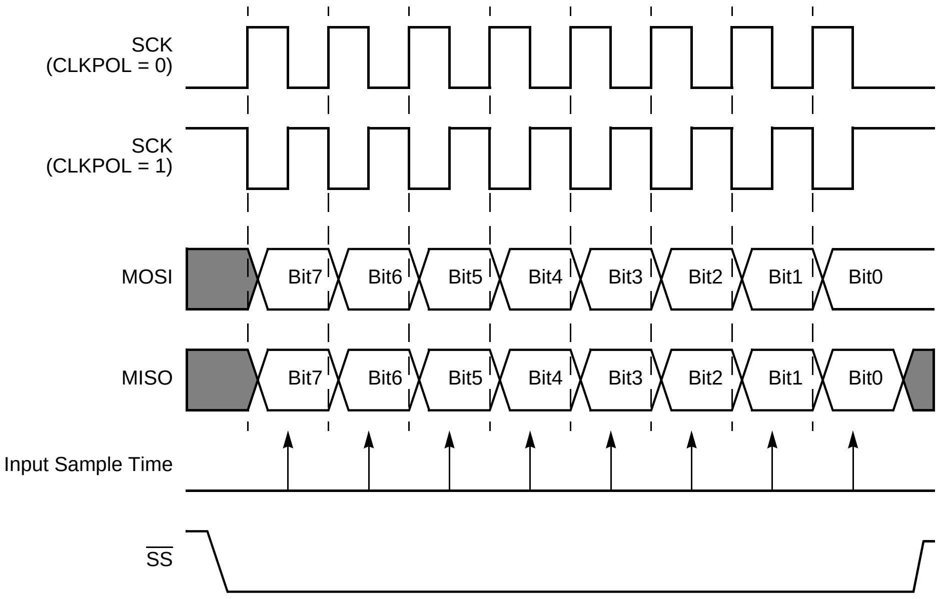


Figure 78. SPI Timing When `PHASE` is 1

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 $\overline{\text{SS}}$ 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 $\overline{\text{SS}}$ 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).

Error Detection

The SPI contains error detection logic to support SPI communication protocols and recognize when communication errors have occurred. The SPI Status register indicates when a data transmission error has been detected.

Overrun (Write Collision)

An overrun error (write collision) indicates a write to the SPI Data register was attempted while a data transfer is in progress. An overrun sets the `OVR` bit in the SPI Status register to 1. Writing a 1 to `OVR` clears this error flag.

Mode Fault (Multi-Master Collision)

A mode fault indicates when more than one Master is trying to communicate at the same time (a multi-master collision). The mode fault is detected when the enabled Master's $\overline{\text{SS}}$ pin is asserted. A mode fault sets the `COL` bit in the SPI Status register to 1. Writing a 1 to `COL` clears this error flag.

SPI Interrupts

When SPI interrupts are enabled, the SPI generates an interrupt after data transmission. The SPI in Master mode generates an interrupt after a character has been sent. A character can be defined to be 1 through 8 bits by the `NUMBITS` field in the SPI Mode register. The SPI in Slave mode generates an interrupt when the $\overline{\text{SS}}$ signal deasserts to indicate completion of the data transfer. Writing a 1 to the `IRQ` bit in the SPI Status Register clears the pending interrupt request. If the SPI is disabled, an SPI interrupt can be generated by a Baud Rate Generator time-out.

SPI Baud Rate Generator

In SPI Master mode, the Baud Rate Generator creates a lower frequency serial clock (SCK) for data transmission synchronization between the Master and the external Slave. The input to the Baud Rate Generator is the system clock. 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. The reload value must be greater than or equal to `0002H` for SPI operation (maximum baud rate is system clock frequency divided by 4). The SPI baud rate is calculated using the following equation:

$$\textbf{SPI Baud Rate (bits/s)} = \frac{\textbf{System Clock Frequency (Hz)}}{\textbf{2} \times \textbf{BRG[15:0]}}$$

When the SPI is disabled, the Baud Rate Generator can function as a basic 16-bit timer with interrupt on time-out. To configure the Baud Rate Generator as a timer with interrupt on time-out, complete the following procedure:

1. Disable the SPI by clearing the `SPIEN` bit in the SPI Control register to 0.
2. Load the desired 16-bit count value into the SPI Baud Rate High and Low Byte registers.
3. Enable the Baud Rate Generator timer function and associated interrupt by setting the `BIRQ` bit in the SPI Control register to 1.

SPI Control Register Definitions

SPI Data Register

The SPI Data register stores both the outgoing (transmit) data and the incoming (received) data. Reads from the SPI Data register always return the current contents of the 8-bit shift register.

With the SPI configured as a Master, writing a data byte to this register initiates the data transmission. With the SPI configured as a Slave, writing a data byte to this register loads the shift register in preparation for the next data transfer with the external Master. In either the Master or Slave modes, if a transmission is already in progress, writes to this register are ignored and the Overrun error flag, `OVR`, is set in the SPI Status register.

When the character length is less than 8 bits (as set by the `NUMBITS` field in the SPI Mode register), the transmit character must be left justified in the SPI Data register. A received character of less than 8 bits will be right justified. For example, if the SPI is configured for 4-bit characters, the transmit characters must be written to SPIDATA[7:4] and the received characters are read from SPIDATA[3:0].

Table 60. SPI Data Register (SPIDATA)

BITS	7	6	5	4	3	2	1	0
FIELD	DATA							
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	F60H							

DATA—Data
Transmit and/or receive data.

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.

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.

Option Bits

Overview

Option Bits allow user configuration of certain aspects of Z8F640x family device operation. The feature configuration data is stored in the Program Memory and read during Reset. The features available for control via the Option Bits are:

- Watch-Dog Timer time-out response selection–interrupt or Short Reset.
- Watch-Dog Timer enabled at Reset.
- The ability to prevent unwanted read access to user code in Program Memory.
- The ability to prevent accidental programming and erasure of all or a portion of the user code in Program Memory.

Operation

Option Bit Configuration By Reset

Each time the Option Bits are programmed or erased, the Z8F640x family device must be Reset for the change to take place. During any reset operation (System Reset, Short Reset, or Stop Mode Recovery), the Option Bits are automatically read from the Program Memory and written to Option Configuration registers. The Option Configuration registers control operation of the Z8F640x family device. Option Bit control of the Z8F640x family device is established before the device exits Reset and the eZ8 CPU begins code execution. The Option Configuration registers are not part of the Register File and are not accessible for read or write access.

Option Bit Address Space

The first two bytes of Program Memory at addresses `0000H` and `0001H` are reserved for the user Option Bits. The byte at Program Memory address `0000H` is used to configure user options. The byte at Program Memory address `0001H` is reserved for future use and must be left in its unprogrammed state.

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.

zero). If the Z8F640x family device is not in Debug mode or if the Read Protect Option Bit is enabled, this command returns `FFH` for all the data values.

```
DBG <-- 09H
DBG <-- {4'h0,Register Address[11:8]
DBG <-- Register Address[7:0]
DBG <-- Size[7:0]
DBG --> 1-256 data bytes
```

- **Write Program Memory (0AH)**—The Write Program Memory command writes data to Program Memory. This command is equivalent to the LDC and LDCI instructions. Data can be written 1-65536 bytes at a time (65536 bytes can be written by setting size to zero). The on-chip Flash Controller must be written to and unlocked for the programming operation to occur. If the Flash Controller is not unlocked, the data is discarded. If the Z8F640x family device is not in Debug mode or if the Read Protect Option Bit is enabled, the data is discarded.

```
DBG <-- 0AH
DBG <-- Program Memory Address[15:8]
DBG <-- Program Memory Address[7:0]
DBG <-- Size[15:8]
DBG <-- Size[7:0]
DBG <-- 1-65536 data bytes
```

- **Read Program Memory (0BH)**—The Read Program Memory command reads data from Program Memory. This command is equivalent to the LDC and LDCI instructions. Data can be read 1-65536 bytes at a time (65536 bytes can be read by setting size to zero). If the Z8F640x family device is not in Debug mode or if the Read Protect Option Bit is enabled, this command returns `FFH` for the data.

```
DBG <-- 0BH
DBG <-- Program Memory Address[15:8]
DBG <-- Program Memory Address[7:0]
DBG <-- Size[15:8]
DBG <-- Size[7:0]
DBG --> 1-65536 data bytes
```

- **Write Data Memory (0CH)**—The Write Data Memory command writes data to Data Memory. This command is equivalent to the LDE and LDEI instructions. Data can be written 1-65536 bytes at a time (65536 bytes can be written by setting size to zero). If the Z8F640x family device is not in Debug mode or if the Read Protect Option Bit is enabled, the data is discarded.

```
DBG <-- 0CH
DBG <-- Data Memory Address[15:8]
DBG <-- Data Memory Address[7:0]
DBG <-- Size[15:8]
DBG <-- Size[7:0]
DBG <-- 1-65536 data bytes
```

Table 121. CPU Control Instructions

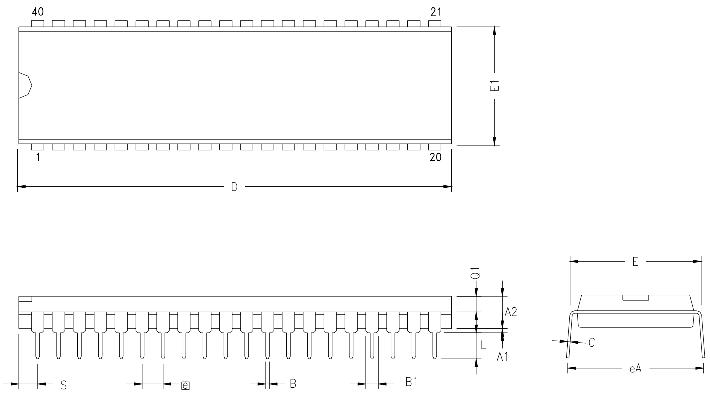
Mnemonic	Operands	Instruction
CCF	—	Complement Carry Flag
DI	—	Disable Interrupts
EI	—	Enable Interrupts
HALT	—	Halt Mode
NOP	—	No Operation
RCF	—	Reset Carry Flag
SCF	—	Set Carry Flag
SRP	src	Set Register Pointer
STOP	—	Stop Mode
WDT	—	Watch-Dog Timer Refresh

Table 122. Load Instructions

Mnemonic	Operands	Instruction
CLR	dst	Clear
LD	dst, src	Load
LDC	dst, src	Load Constant to/from Program Memory
LDCI	dst, src	Load Constant to/from Program Memory and Auto-Increment Addresses
LDE	dst, src	Load External Data to/from Data Memory
LDEI	dst, src	Load External Data to/from Data Memory and Auto-Increment Addresses
LDX	dst, src	Load using Extended Addressing
LEA	dst, X(src)	Load Effective Address
POP	dst	Pop
POPX	dst	Pop using Extended Addressing
PUSH	src	Push
PUSHX	src	Push using Extended Addressing

Packaging

Figure 103 illustrates the 40-pin PDIP (plastic dual-inline package) available for the Z8F1601, Z8F2401, Z8F3201, Z8F4801, and Z8F6401 devices.

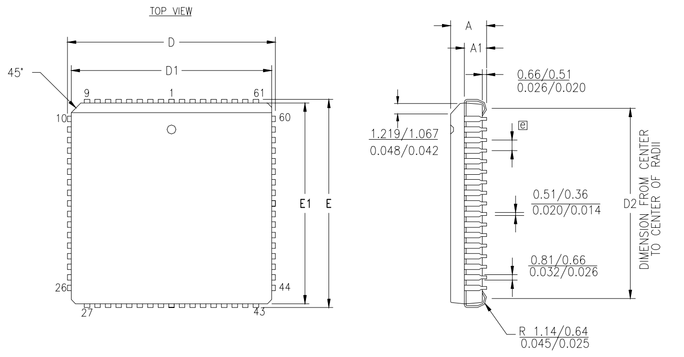


SYMBOL	MILLIMETER		INCH	
	MIN	MAX	MIN	MAX
A1	0.51	1.02	.020	.040
A2	3.18	3.94	.125	.155
B	0.38	0.53	.015	.021
B1	1.02	1.52	.040	.060
C	0.23	0.38	.009	.015
D	52.07	52.58	2.050	2.070
E	15.24	15.75	.600	.620
E1	13.59	14.22	.535	.560
$\boxed{e}$	2.54 TYP		.100 TYP	
eA	15.49	16.76	.610	.660
L	3.05	3.81	.120	.150
Q1	1.40	1.91	.055	.075
S	1.52	2.29	.060	.090

CONTROLLING DIMENSIONS : INCH

Figure 103. 40-Lead Plastic Dual-Inline Package (PDIP)

Figure 107 illustrates the 68-pin PLCC (plastic lead chip carrier) package available for the Z8F1602, Z8F2402, Z8F3202, Z8F4802, and Z8F6402 devices.



SYMBOL	MILLIMETER		INCH	
	MIN	MAX	MIN	MAX
A	4.32	4.57	.170	.180
A1	2.43	2.92	.095	.115
D/E	25.02	25.40	.985	1.000
D1/E1	24.13	24.33	.950	.958
D2	22.86	23.62	.900	.930
e	1.27 BSC		.050 BSC	

NOTE:
1. CONTROLLING DIMENSIONS : INCH.
2. LEADS ARE COPLANAR WITHIN 0.004 IN. RANGE.
3. DIMENSION : $\frac{\text{MM}}{\text{INCH.}}$

Figure 107. 68-Lead Plastic Lead Chip Carrier Package (PLCC)

extended addressing register 184
external pin reset 29
eZ8 CPU features 3
eZ8 CPU instruction classes 187
eZ8 CPU instruction notation 183
eZ8 CPU instruction set 182
eZ8 CPU instruction summary 191

F

FCTL register 144
features, Z8 Encore!® 1
first opcode map 204
FLAGS 185
flags register 185
flash
  controller 4
  option bit address space 148
  option bit configuration - reset 148
  program memory address 0000H 149
  program memory address 0001H 150
flash memory 138
  arrangement 139
  byte programming 142
  code protection 141
  configurations 138
  control register definitions 144
  controller bypass 143
  electrical characteristics and timing 173
  flash control register 144
  flash option bits 142
  flash status register 145
  flow chart 140
  frequency high and low byte registers 147
  mass erase 143
  operation 139
  operation timing 141
  page erase 143
  page select register 146
FPS register 146
FSTAT register 145

G

gated mode 71
general-purpose I/O 33
GPIO 4, 33
  alternate functions 34
  architecture 34
  control register definitions 36
  input data sample timing 176
  interrupts 36
  port A-H address registers 37
  port A-H alternate function sub-registers 39
  port A-H control registers 38
  port A-H data direction sub-registers 39
  port A-H high drive enable sub-registers 41
  port A-H input data registers 42
  port A-H output control sub-registers 40
  port A-H output data registers 43
  port A-H stop mode recovery sub-registers 41
  port availability by device 33
  port input timing 176
  port output timing 177

H

H 185
HALT 189
HALT mode 31, 189
hexadecimal number prefix/suffix 185

I

I^{2}C 4
  10-bit address read transaction 116
  10-bit address transaction 114
  10-bit addressed slave data transfer format 114
  10-bit receive data format 116
  7-bit address transaction 112
  7-bit address, reading a transaction 115
  7-bit addressed slave data transfer format 113
  7-bit receive data transfer format 115
  baud high and low byte registers 121
  C status register 118
  control register definitions 118

RRC 191
SBC 188
SCF 188, 189
SRA 191
SRL 191
SRP 189
STOP 189
SUB 188
SUBX 188
SWAP 191
TCM 188
TCMX 188
TM 188
TMX 188
TRAP 190
watch-dog timer refresh 189
XOR 190
XORX 190
instructions, eZ8 classes of 187
interrupt control register 56
interrupt controller 5, 44
architecture 44
interrupt assertion types 47
interrupt vectors and priority 47
operation 46
register definitions 48
interrupt edge select register 54
interrupt port select register 55
interrupt request 0 register 48
interrupt request 1 register 49
interrupt request 2 register 50
interrupt return 190
interrupt vector listing 44
interrupts
not acknowledge 112
receive 112
SPI 105
transmit 112
UART 85
introduction 1
IR 184
Ir 184
IrDA
architecture 95
block diagram 95
control register definitions 98
jitter 98
operation 96
receiving data 97
transmitting data 96
IRET 190
IRQ0 enable high and low bit registers 51
IRQ1 enable high and low bit registers 52
IRQ2 enable high and low bit registers 53
IRR 184
Irr 184

J
jitter 98
JP 190
jump, conditional, relative, and relative conditional 190

L
LD 189
LDC 189
LDCI 188, 189
LDE 189
LDEI 188, 189
LDX 189
LEA 189
load 189
load constant 188
load constant to/from program memory 189
load constant with auto-increment addresses 189
load effective address 189
load external data 189
load external data to/from data memory and auto-increment addresses 188
load external to/from data memory and auto-increment addresses 189
load instructions 189
load using extended addressing 189
logical AND 190
logical AND/extended addressing 190
logical exclusive OR 190