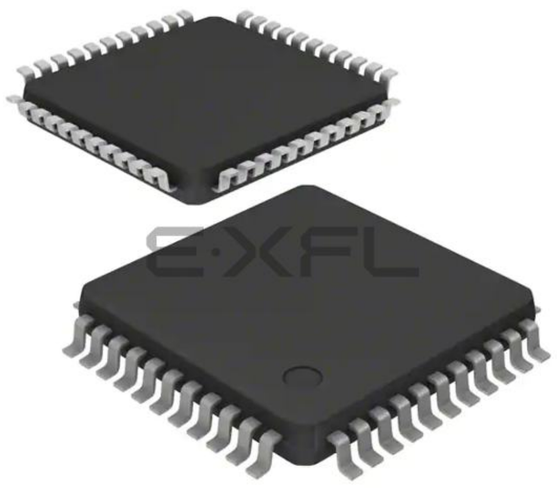
Zilog - Z8F6401AN020SC00TR Datasheet



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	0°C ~ 70°C (TA)
Mounting Type	Surface Mount
Package / Case	44-LQFP
Supplier Device Package	44-LQFP (10x10)
Purchase URL	https://www.e-xfl.com/product-detail/zilog/z8f6401an020sc00tr

Architecture

Figure 65 illustrates a block diagram of the interrupt controller.

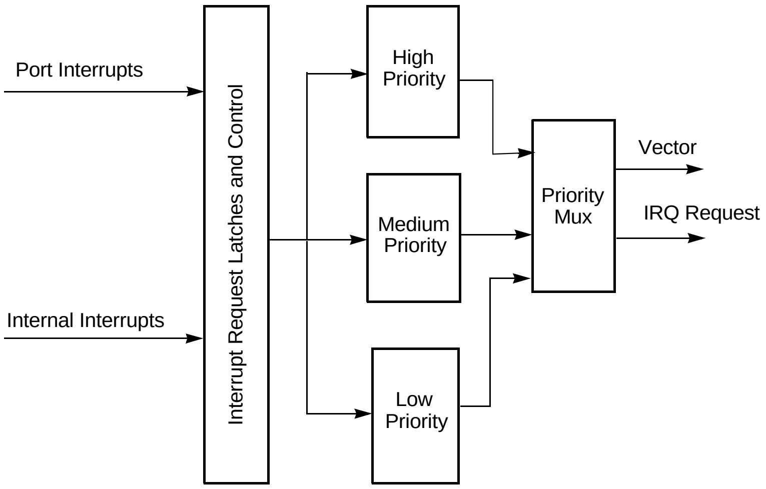


Figure 65. Interrupt Controller Block Diagram

Operation

Master Interrupt Enable

The master interrupt enable bit (`IRQE`) in the Interrupt Control register globally enables and disables interrupts.

Interrupts are globally enabled by any of the following actions:

- Execution of an EI (Enable Interrupt) instruction
- Execution of an IRET (Return from Interrupt) instruction
- Writing a 1 to the `IRQE` bit in the Interrupt Control register

Interrupts are globally disabled by any of the following actions:

- Execution of a DI (Disable Interrupt) instruction
- eZ8 CPU acknowledgement of an interrupt service request from the interrupt controller
- Writing a 0 to the `IRQE` bit in the Interrupt Control register
- Reset

Interrupt Control Register Definitions

For all interrupts other than the Watch-Dog Timer interrupt, the interrupt control registers enable individual interrupts, set interrupt priorities, and indicate interrupt requests.

Interrupt Request 0 Register

The Interrupt Request 0 (IRQ0) register (Table 23) stores the interrupt requests for both vectored and polled interrupts. When a request is presented to the interrupt controller, the corresponding bit in the IRQ0 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 can read the Interrupt Request 0 register to determine if any interrupt requests are pending

Table 23. Interrupt Request 0 Register (IRQ0)

BITS	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FIELD	T2I	T1I	T0I	U0RXI	U0TXI	I2CI	SPII	ADCI
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	FC0H							

T2I—Timer 2 Interrupt Request
0 = No interrupt request is pending for Timer 2.
1 = An interrupt request from Timer 2 is awaiting service.

T1I—Timer 1 Interrupt Request
0 = No interrupt request is pending for Timer 1.
1 = An interrupt request from Timer 1 is awaiting service.

T0I—Timer 0 Interrupt Request
0 = No interrupt request is pending for Timer 0.
1 = An interrupt request from Timer 0 is awaiting service.

U0RXI—UART 0 Receiver Interrupt Request
0 = No interrupt request is pending for the UART 0 receiver.
1 = An interrupt request from the UART 0 receiver is awaiting service.

U0TXI—UART 0 Transmitter Interrupt Request
0 = No interrupt request is pending for the UART 0 transmitter.
1 = An interrupt request from the UART 0 transmitter is awaiting service.

IRQ0 Enable High and Low Bit Registers

The IRQ0 Enable High and Low Bit registers (Tables 27 and 28) form a priority encoded enabling for interrupts in the Interrupt Request 0 register. Priority is generated by setting bits in each register. Table 26 describes the priority control for IRQ0.

Table 26. 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.

Table 27. IRQ0 Enable High Bit Register (IRQ0ENH)

BITS	7	6	5	4	3	2	1	0
FIELD	T2ENH	T1ENH	T0ENH	U0RENH	U0TENH	I2CENH	SPIENH	ADCENH
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	FC1H							

T2ENH—Timer 2 Interrupt Request Enable High Bit
T1ENH—Timer 1 Interrupt Request Enable High Bit
T0ENH—Timer 0 Interrupt Request Enable High Bit
U0RENH—UART 0 Receive Interrupt Request Enable High Bit
U0TENH—UART 0 Transmit Interrupt Request Enable High Bit
I2CENH—I^{2}C Interrupt Request Enable High Bit
SPIENH—SPI Interrupt Request Enable High Bit
ADCENH—ADC Interrupt Request Enable High Bit

Interrupt Port Select register selects between Port A and Port D for the individual interrupts.

Table 35. Interrupt Edge Select Register (IRQES)

BITS	7	6	5	4	3	2	1	0
FIELD	IES7	IES6	IES5	IES4	IES3	IES2	IES1	IES0
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	FCDH							

IES*x*—Interrupt Edge Select *x*
where *x* indicates the specific GPIO Port pin number (0 through 7). The pulse width should be greater than 1 system clock to guarantee capture of the edge triggered interrupt.
0 = An interrupt request is generated on the falling edge of the PA*x*/PD*x* input.
1 = An interrupt request is generated on the rising edge of the PA*x*/PD*x* input.

Interrupt Port Select Register

The Port Select (IRQPS) register (Table 36) determines the port pin that generates the PAx/PDx interrupts. This register allows either Port A or Port D pins to be used as interrupts. The Interrupt Edge Select register controls the active interrupt edge.

Table 36. Interrupt Port Select Register (IRQPS)

BITS	7	6	5	4	3	2	1	0
FIELD	PAD7S	PAD6S	PAD5S	PAD4S	PAD3S	PAD2S	PAD1S	PAD0S
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	FCEH							

PAD*x*S—PA*x*/PD*x* Selection
0 = PA*x* is used for the interrupt for PA*x*/PD*x* interrupt request.
1 = PD*x* is used for the interrupt for PA*x*/PD*x* interrupt request.
where *x* indicates the specific GPIO Port pin number (0 through 7).

Middle byte, Bits[15:8], of the 24-bit WDT reload value.

Table 49. Watch-Dog Timer Reload Low Byte Register (WDTL)

BITS	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FIELD	WDTL							
RESET	1	1	1	1	1	1	1	1
R/W	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*	R/W*
ADDR	FF3H							
R/W* - Read returns the current WDT count value. Write sets the desired Reload Value.								

WDTL—WDT Reload Low

Least significant byte (LSB), Bits[7:0], of the 24-bit WDT reload value.

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

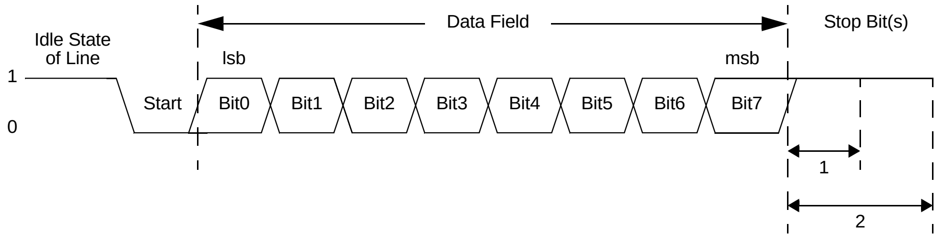


Figure 68. UART Asynchronous Data Format without Parity

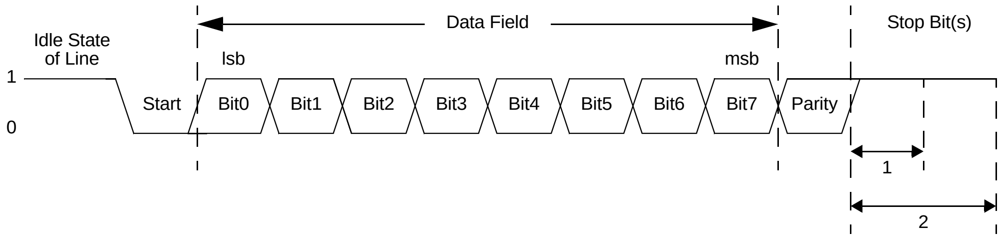


Figure 69. UART Asynchronous Data Format with Parity

Transmitting Data using the Polled Method

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

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

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.

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. The reload value must be greater than or equal to `0002H` for proper SPI operation (maximum baud rate is system clock frequency divided by 4). The SPI baud rate is calculated using the following equation:

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

Table 64. SPI Baud Rate High Byte Register (SPIBRH)

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

BRH = SPI Baud Rate High Byte
Most significant byte, BRG[15:8], of the SPI Baud Rate Generator's reload value.

Table 65. SPI Baud Rate Low Byte Register (SPIBRL)

BITS	7	6	5	4	3	2	1	0
FIELD	BRL							
RESET	1	1	1	1	1	1	1	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/w
ADDR	F67H							

BRL = SPI Baud Rate Low Byte
Least significant byte, BRG[7:0], of the SPI Baud Rate Generator's reload value.

2. The I^{2}C 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^{2}C Controller sends a Stop signal.
3. If the slave needs to service an interrupt, it pulls the SCL signal Low, which halts I^{2}C operation.
4. If there is no other data in the I^{2}C Data register or the `STOP` bit in the I^{2}C 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^{2}C Controller to slaves and unshaded regions indicate data transferred from the slaves to the I^{2}C Controller.

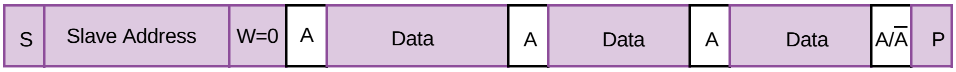


Figure 79. 7-Bit Addressed Slave Data Transfer Format

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^{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
4. Software responds to the `TDRE` bit by writing a 7-bit slave address followed by a 0 (write) to the I^{2}C Data register.
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 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^{2}C Data register.
10. The I^{2}C Controller shifts the rest of the address and write bit out by the SDA signal.
11. The I^{2}C slave sends an acknowledge (by pulling the SDA signal low) during the next high period of SCL. The I^{2}C Controller sets the `ACK` bit in the I^{2}C Status register.
12. The I^{2}C Controller loads the contents of the I^{2}C Shift register with the contents of the I^{2}C Data register.
13. The I^{2}C Controller shifts the data out of via the SDA signal. After the first bit is sent, the Transmit interrupt is asserted.

I²C Baud Rate High and Low Byte Registers

The I²C Baud Rate High and Low Byte registers combine to form a 16-bit reload value, BRG[15:0], for the I²C Baud Rate Generator. The I²C baud rate is calculated using the following equation:

$$\text{I2C Baud Rate (bits/s)} = \frac{\text{System Clock Frequency (Hz)}}{4 \times \text{BRG[15:0]}}$$

.

Table 69. I²C Baud Rate High Byte Register (I2CBRH)

BITS	7	6	5	4	3	2	1	0
FIELD	BRH							
RESET	1	1	1	1	1	1	1	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	F53H							

BRH = I²C Baud Rate High Byte
Most significant byte, BRG[15:8], of the I²C Baud Rate Generator's reload value.

Table 70. I²C Baud Rate Low Byte Register (I2CBRL)

BITS	7	6	5	4	3	2	1	0
FIELD	BRL							
RESET	1	1	1	1	1	1	1	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	F54H							

BRL = I²C Baud Rate Low Byte
Least significant byte, BRG[7:0], of the I²C Baud Rate Generator's reload value.

Flash Operation Timing Using the Flash Frequency Registers

Before performing either a program or erase operation on the Flash memory, the user must first configure the Flash Frequency High and Low Byte registers. The Flash Frequency registers allow programming and erasure of the Flash with system clock frequencies ranging from 32KHz (32768Hz) through 20MHz.

The Flash Frequency High and Low Byte registers combine to form a 16-bit value, `FFREQ`, to control timing for Flash program and erase operations. The 16-bit binary Flash Frequency value must contain the system clock frequency (in kHz). This value is calculated using the following equation:.

$$\mathbf{FFREQ[15{:}0]} = \frac{\textbf{System Clock Frequency (Hz)}}{\mathbf{1000}}$$

Caution: Flash programming and erasure are not supported for system clock frequencies below 32KHz (32768Hz) or above 20MHz. The Flash Frequency High and Low Byte registers must be loaded with the correct value to insure proper operation of the Z8F640x family device.

Flash Code Protection Against External Access

The user code contained within the Z8F640x family device's Flash memory can be protected against external access via the On-Chip Debugger. Programming the `RP` Option Bit prevents reading of the user code through the On-Chip Debugger. Refer to the **Option Bits** chapter and the **On-Chip Debugger** chapter for more information.

Flash Code Protection Against Accidental Program and Erasure

The Z8F640x family device provides several levels of protection against accidental program and erasure of the Flash memory contents. This protection is provided by a combination of the Option bits and the locking mechanism of the Flash Controller.

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

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

Program Memory Address 0001H

Table 91. Options Bits at Program Memory Address 0001H

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

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

Table 101. DC Characteristics

		T_A = -40⁰C to 105⁰C				
Symbol	**Parameter**	**Minimum**	**Typical**	**Maximum**	**Units**	**Conditions**
I_{PU}	Weak Pull-up Current	30	100	350	µA	V_{DD} = 3.0 - 3.6V
I_{CCS}	Supply Current in Stop Mode		600		µA	V_{DD} = 3.3V

[1] This condition excludes all pins that have on-chip pull-ups, when driven Low.
[2] These values are provided for design guidance only and are not tested in production.

Figure 91 illustrates the typical current consumption while operating at 25ºC, 3.3V, versus the system clock frequency.

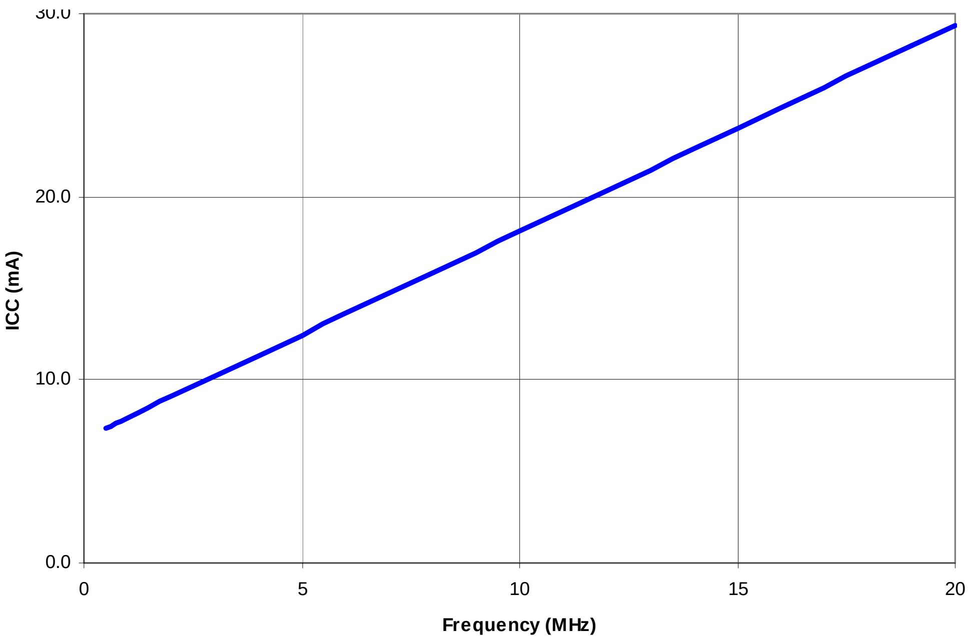


Figure 91. Nominal ICC Versus System Clock Frequency

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 **BRK**	2.2 **SRP** IM	2.3 **ADD** r1,r2	2.4 **ADD** r1,Ir2	3.3 **ADD** R2,R1	3.4 **ADD** IR2,R1	3.3 **ADD** R1,IM	3.4 **ADD** IR1,IM	4.3 **ADDX** ER2,ER1	4.3 **ADDX** IM,ER1	2.3 **DJNZ** r1,X	2.2 **JR** cc,X	2.2 **LD** r1,IM	3.2 **JP** cc,DA	1.2 **INC** r1	1.2 **NOP**
1	2.2 **RLC** R1	2.3 **RLC** IR1	2.3 **ADC** r1,r2	2.4 **ADC** r1,Ir2	3.3 **ADC** R2,R1	3.4 **ADC** IR2,R1	3.3 **ADC** R1,IM	3.4 **ADC** IR1,IM	4.3 **ADCX** ER2,ER1	4.3 **ADCX** IM,ER1	↓	↓	↓	↓	↓	See 2nd Opcode Map
2	2.2 **INC** R1	2.3 **INC** IR1	2.3 **SUB** r1,r2	2.4 **SUB** r1,Ir2	3.3 **SUB** R2,R1	3.4 **SUB** IR2,R1	3.3 **SUB** R1,IM	3.4 **SUB** IR1,IM	4.3 **SUBX** ER2,ER1	4.3 **SUBX** IM,ER1	↓	↓	↓	↓	↓	
3	2.2 **DEC** R1	2.3 **DEC** IR1	2.3 **SBC** r1,r2	2.4 **SBC** r1,Ir2	3.3 **SBC** R2,R1	3.4 **SBC** IR2,R1	3.3 **SBC** R1,IM	3.4 **SBC** IR1,IM	4.3 **SBCX** ER2,ER1	4.3 **SBCX** IM,ER1	↓	↓	↓	↓	↓	
4	2.2 **DA** R1	2.3 **DA** IR1	2.3 **OR** r1,r2	2.4 **OR** r1,Ir2	3.3 **OR** R2,R1	3.4 **OR** IR2,R1	3.3 **OR** R1,IM	3.4 **OR** IR1,IM	4.3 **ORX** ER2,ER1	4.3 **ORX** IM,ER1	↓	↓	↓	↓	↓	
5	2.2 **POP** R1	2.3 **POP** IR1	2.3 **AND** r1,r2	2.4 **AND** r1,Ir2	3.3 **AND** R2,R1	3.4 **AND** IR2,R1	3.3 **AND** R1,IM	3.4 **AND** IR1,IM	4.3 **ANDX** ER2,ER1	4.3 **ANDX** IM,ER1	↓	↓	↓	↓	↓	1.2 **WDT**
6	2.2 **COM** R1	2.3 **COM** IR1	2.3 **TCM** r1,r2	2.4 **TCM** r1,Ir2	3.3 **TCM** R2,R1	3.4 **TCM** IR2,R1	3.3 **TCM** R1,IM	3.4 **TCM** IR1,IM	4.3 **TCMX** ER2,ER1	4.3 **TCMX** IM,ER1	↓	↓	↓	↓	↓	1.2 **STOP**
7	2.2 **PUSH** R2	2.3 **PUSH** IR2	2.3 **TM** r1,r2	2.4 **TM** r1,Ir2	3.3 **TM** R2,R1	3.4 **TM** IR2,R1	3.3 **TM** R1,IM	3.4 **TM** IR1,IM	4.3 **TMX** ER2,ER1	4.3 **TMX** IM,ER1	↓	↓	↓	↓	↓	1.2 **HALT**
8	2.5 **DECW** RR1	2.6 **DECW** IRR1	2.5 **LDE** r1,Irr2	2.9 **LDEI** Ir1,Irr2	3.2 **LDX** r1,ER2	3.3 **LDX** Ir1,ER2	3.4 **LDX** IRR2,R1	3.5 **LDX** IRR2,IR1	3.4 **LDX** r1,rr2,X	3.4 **LDX** rr1,r2,X	↓	↓	↓	↓	↓	1.2 **DI**
9	2.2 **RL** R1	2.3 **RL** IR1	2.5 **LDE** r2,Irr1	2.9 **LDEI** Ir2,Irr1	3.2 **LDX** r2,ER1	3.3 **LDX** Ir2,ER1	3.4 **LDX** R2,IRR1	3.5 **LDX** IR2,IRR1	3.3 **LEA** r1,r2,X	3.5 **LEA** rr1,rr2,X	↓	↓	↓	↓	↓	1.2 **EI**
A	2.5 **INCW** RR1	2.6 **INCW** IRR1	2.3 **CP** r1,r2	2.4 **CP** r1,Ir2	3.3 **CP** R2,R1	3.4 **CP** IR2,R1	3.3 **CP** R1,IM	3.4 **CP** IR1,IM	4.3 **CPX** ER2,ER1	4.3 **CPX** IM,ER1	↓	↓	↓	↓	↓	1.4 **RET**
B	2.2 **CLR** R1	2.3 **CLR** IR1	2.3 **XOR** r1,r2	2.4 **XOR** r1,Ir2	3.3 **XOR** R2,R1	3.4 **XOR** IR2,R1	3.3 **XOR** R1,IM	3.4 **XOR** IR1,IM	4.3 **XORX** ER2,ER1	4.3 **XORX** IM,ER1	↓	↓	↓	↓	↓	1.5 **IRET**
C	2.2 **RRC** R1	2.3 **RRC** IR1	2.5 **LDC** r1,Irr2	2.9 **LDCI** Ir1,Irr2	2.3 **JP** IRR1	2.9 **LDC** Ir1,Irr2		3.3 **LD** r1,r2,X	3.2 **PUSHX** ER2		↓	↓	↓	↓	↓	1.2 **RCF**
D	2.2 **SRA** R1	2.3 **SRA** IR1	2.5 **LDC** r2,Irr1	2.9 **LDCI** Ir2,Irr1	2.6 **CALL** IRR1	2.2 **BSWAP** R1	3.3 **CALL** DA	3.4 **LD** r2,r1,X	3.2 **POPX** ER1		↓	↓	↓	↓	↓	1.2 **SCF**
E	2.2 **RR** R1	2.3 **RR** IR1	2.2 **BIT** p,b,r1	2.3 **LD** r1,Ir2	3.2 **LD** R2,R1	3.3 **LD** IR2,R1	3.2 **LD** R1,IM	3.3 **LD** IR1,IM	4.2 **LDX** ER2,ER1	4.2 **LDX** IM,ER1	↓	↓	↓	↓	↓	1.2 **CCF**
F	2.2 **SWAP** R1	2.3 **SWAP** IR1	2.6 **TRAP** Vector	2.3 **LD** Ir1,r2	2.8 **MULT** RR1	3.3 **LD** R2,IR1	3.3 **BTJ** p,b,r1,X	3.4 **BTJ** p,b,Ir1,X			↓	↓	↓	↓	↓	

Figure 101. First Opcode Map

Precharacterization Product

The product represented by this document is newly introduced and ZiLOG has not completed the full characterization of the product. The document states what ZiLOG knows about this product at this time, but additional features or nonconformance with some aspects of the document might be found, either by ZiLOG or its customers in the course of further application and characterization work. In addition, ZiLOG cautions that delivery might be uncertain at times, due to start-up yield issues.

ZiLOG, Inc.
532 Race Street
San Jose, CA 95126
Telephone (408) 558-8500
FAX 408 558-8300
Internet: www.zilog.com

Document Information

Document Number Description

The Document Control Number that appears in the footer on each page of this document contains unique identifying attributes, as indicated in the following table:

PS	Product Specification
0176	Unique Document Number
01	Revision Number
0702	Month and Year Published

Problem Description or Suggestion

Provide a complete description of the problem or your suggestion. If you are reporting a specific problem, include all steps leading up to the occurrence of the problem. Attach additional pages as necessary.