Zilog - Z8F1622AR020SC00TR 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	46
Program Memory Size	16KB (16K 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/z8f1622ar020sc00tr

Table of Contents

Use of All Uppercase Letters

The use of all uppercase letters designates the names of states, modes, and commands.

- Example 1: The bus is considered BUSY after the Start condition.
- Example 2: A START command triggers the processing of the initialization sequence.
- Example 3: STOP mode.

Bit Numbering

Bits are numbered from *0* to *n–1* where *n* indicates the total number of bits. For example, the 8 bits of a register are numbered from 0 to 7.

Safeguards

It is important that you understand the following safety terms, which are defined here.

Caution: *Indicates a procedure or file may become corrupted if you do not follow directions.*

Block Diagram

Figure 1 displays the block diagram of the architecture of the Z8 Encore! XP 64K Series Flash Microcontrollers.

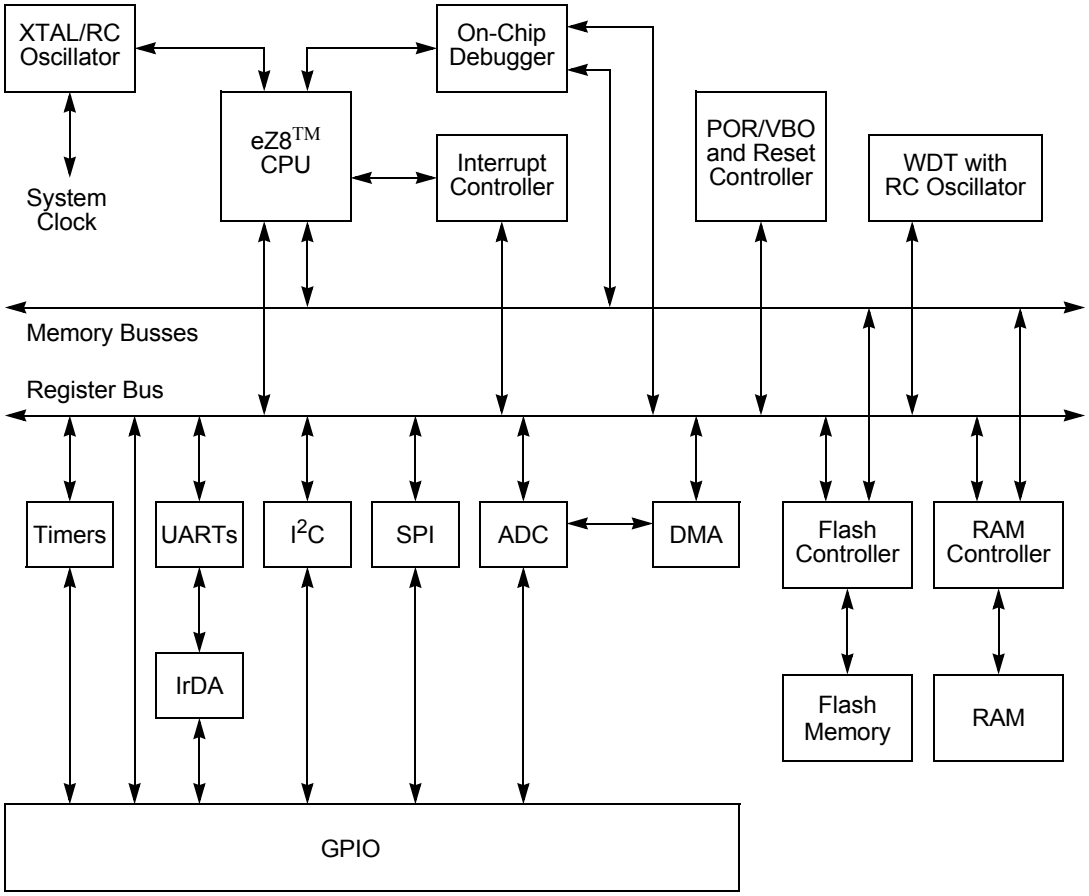


Figure 1. Z8 Encore! XP 64K Series Flash Microcontrollers Block Diagram

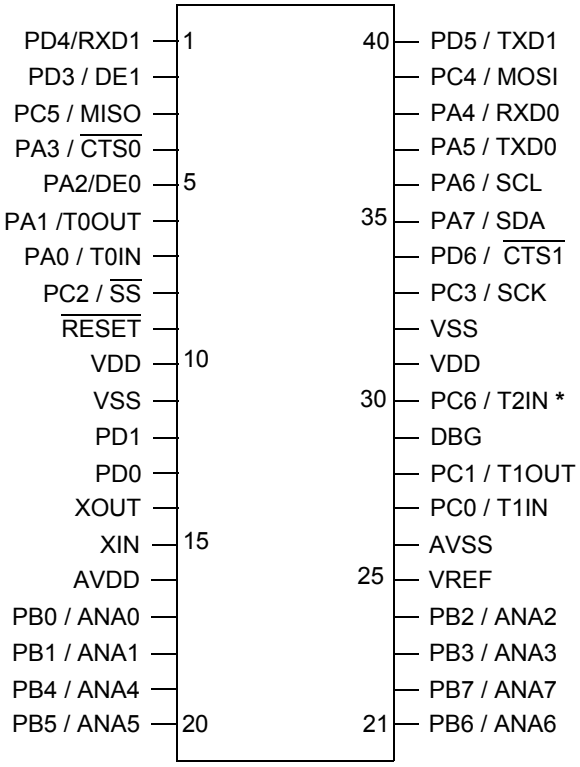
CPU and Peripheral Overview

eZ8™ CPU Features

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

Pin Configurations

Figure 2 through Figure 7 on page 13 display the pin configurations for all of the packages available in the Z8 Encore! XP 64K Series Flash Microcontrollers. For description of the signals, see Table 3 on page 14. Timer 3 is not available in the 40-pin and 44-pin packages.



Note: Timer 3 is not supported.

* T2OUT is not supported.

Figure 2. Z8 Encore! XP 64K Series Flash Microcontrollers in 40-Pin Dual Inline Package (PDIP)

Register File Address Map

Table 7 provides the address map for the Register File of the 64K Series products. Not all devices and package styles in the 64K Series support Timer 3 and all of the GPIO Ports. Consider registers for unimplemented peripherals as Reserved.

Table 7. Z8 Encore! XP 64K Series Flash Microcontrollers Register File Address Map

Address (Hex)	Register Description	Mnemonic	Reset (Hex)	Page No
General-Purpose RAM				
000-EFF	General-Purpose Register File RAM	—	XX	
Timer 0				
F00	Timer 0 High Byte	T0H	00	90
F01	Timer 0 Low Byte	T0L	01	90
F02	Timer 0 Reload High Byte	T0RH	FF	91
F03	Timer 0 Reload Low Byte	T0RL	FF	91
F04	Timer 0 PWM High Byte	T0PWMH	00	92
F05	Timer 0 PWM Low Byte	T0PWML	00	92
F06	Timer 0 Control 0	T0CTL0	00	93
F07	Timer 0 Control 1	T0CTL1	00	94
Timer 1				
F08	Timer 1 High Byte	T1H	00	90
F09	Timer 1 Low Byte	T1L	01	90
F0A	Timer 1 Reload High Byte	T1RH	FF	91
F0B	Timer 1 Reload Low Byte	T1RL	FF	91
F0C	Timer 1 PWM High Byte	T1PWMH	00	92
F0D	Timer 1 PWM Low Byte	T1PWML	00	92
F0E	Timer 1 Control 0	T1CTL0	00	93
F0F	Timer 1 Control 1	T1CTL1	00	94
Timer 2				
F10	Timer 2 High Byte	T2H	00	90
F11	Timer 2 Low Byte	T2L	01	90
F12	Timer 2 Reload High Byte	T2RH	FF	91
F13	Timer 2 Reload Low Byte	T2RL	FF	91
F14	Timer 2 PWM High Byte	T2PWMH	00	92
F15	Timer 2 PWM Low Byte	T2PWML	00	92
F16	Timer 2 Control 0	T2CTL0	00	93
F17	Timer 2 Control 1	T2CTL1	00	94

Timer 2 Control 1
T2CTL1 (F17H - Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0

Timer Mode
000 = One-Shot mode
001 = CONTINUOUS mode
010 = COUNTER mode
011 = PWM mode
100 = CAPTURE mode
101 = COMPARE mode
110 = GATED mode
111 = CAPTURE/COMPARE mode

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

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

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

Timer 3 High Byte
T3H (F18H - Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0

Timer 3 current count value [15:8]

Timer 3 Low Byte
T3L (F19H - Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0

Timer 3 current count value [7:0]

Timer 3 Reload High Byte
T3RH (F1AH - Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0

Timer 3 reload value [15:8]

Timer 3 Reload Low Byte
T3RL (F1BH - Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0

Timer 3 reload value [7:0]

Timer 3 PWM High Byte
T3PWMH (F1CH - Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0

Timer 3 PWM value [15:8]

Timer 3 PWM Low Byte
T3PWML (F1DH - Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0

Timer 3 PWM value [7:0]

Timer 3 Control 0
T3CTL0 (F1EH - Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0

Reserved

Cascade Timer
0 = Timer 3 Input signal is GPIO pin
1 = Timer 3 Input signal is Timer 2 out

Reserved

Timer 3 Control 1
T3CTL1 (F1FH - Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0

Timer Mode
000 = One-Shot mode
001 = CONTINUOUS mode
010 = COUNTER mode
011 = PWM mode
100 = CAPTURE mode
101 = COMPARE mode
110 = GATED mode
111 = Capture/COMPARE mode

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

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

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

HALT Mode

Execution of the eZ8 CPU's HALT instruction places the device into HALT mode. In HALT mode, the operating characteristics are:

- Primary crystal oscillator is enabled and continues to operate.
- System clock is enabled and continues to operate.
- eZ8 CPU is stopped.
- Program Counter stops incrementing.
- Watchdog Timer's internal RC oscillator continues to operate.
- The Watchdog Timer continues to operate, if enabled.
- All other on-chip peripherals continue to operate.

The eZ8 CPU can be brought out of HALT mode by any of the following operations:

- Interrupt
- Watchdog Timer time-out (interrupt or reset)
- Power-On Reset
- Voltage Brownout Reset
- External $\overline{\text{RESET}}$ pin assertion

To minimize current in HALT mode, all GPIO pins which are configured as inputs must be driven to one of the supply rails (V_{CC} or GND).

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).

Interrupt Control Register

The Interrupt Control (IRQCTL) register (Table 38) contains the master enable bit for all interrupts.

Table 38. Interrupt Control Register (IRQCTL)

BITS	7	6	5	4	3	2	1	0
FIELD	IRQE	Reserved						
RESET	0							
R/W	R/W	R						
ADDR	FCFH							

IRQE—Interrupt Request Enable
This bit is set to 1 by execution of an EI or IRET instruction, or by a direct register write of a 1 to this bit. It is reset to 0 by executing a DI instruction, eZ8 CPU acknowledgement of an interrupt request, or Reset.
0 = Interrupts are disabled
1 = Interrupts are enabled

Reserved—Must be 0.

If interrupts are enabled, following completion of the Stop Mode Recovery the eZ8 CPU responds to the interrupt request by fetching the Watchdog Timer interrupt vector and executing code from the vector address.

WDT Reset in Normal Operation

If configured to generate a Reset when a time-out occurs, the Watchdog Timer forces the device into the Reset state. The WDT status bit in the Watchdog Timer Control register is set to 1. For more information on Reset, see Reset and Stop Mode Recovery on page 47.

WDT Reset in STOP Mode

If enabled in STOP mode and configured to generate a Reset when a time-out occurs and the device is in STOP mode, the Watchdog Timer initiates a Stop Mode Recovery. Both the WDT status bit and the STOP bit in the Watchdog Timer Control register are set to 1 following WDT time-out in STOP mode. Default operation is for the WDT and its RC oscillator to be enabled during STOP mode.

WDT RC Disable in STOP Mode

To minimize power consumption in STOP Mode, the WDT and its RC oscillator can be disabled in STOP mode. The following sequence configures the WDT to be disabled when the 64K Series devices enter STOP Mode following execution of a STOP instruction:

1. Write 55H to the Watchdog Timer Control register (WDTCTL).
2. Write AAH to the Watchdog Timer Control register (WDTCTL).
3. Write 81H to the Watchdog Timer Control register (WDTCTL) to configure the WDT and its oscillator to be disabled during STOP Mode. Alternatively, write 00H to the Watchdog Timer Control register (WDTCTL) as the third step in this sequence to reconfigure the WDT and its oscillator to be enabled during STOP mode.

This sequence only affects WDT operation in STOP mode.

Watchdog Timer Reload Unlock Sequence

Writing the unlock sequence to the Watchdog Timer (WDTCTL) Control register address unlocks the three Watchdog 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. Follow the steps below to unlock the Watchdog Timer Reload Byte registers (WDTU, WDTH, and WDTL) for write access.

1. Write 55H to the Watchdog Timer Control register (WDTCTL).
2. Write AAH to the Watchdog Timer Control register (WDTCTL).
3. Write the Watchdog Timer Reload Upper Byte register (WDTU).
4. Write the Watchdog Timer Reload High Byte register (WDTH).

TXRXSTATE	State Description
0_0000	Idle State
0_0001	START State
0_0010	Send/Receive data bit 7
0_0011	Send/Receive data bit 6
0_0100	Send/Receive data bit 5
0_0101	Send/Receive data bit 4
0_0110	Send/Receive data bit 3
0_0111	Send/Receive data bit 2
0_1000	Send/Receive data bit 1
0_1001	Send/Receive data bit 0
0_1010	Data Acknowledge State
0_1011	Second half of data Acknowledge State used only for not acknowledge
0_1100	First part of STOP state
0_1101	Second part of STOP state
0_1110	10-bit addressing: Acknowledge State for 2nd address byte 7-bit addressing: Address Acknowledge State
0_1111	10-bit address: Bit 0 (Least significant bit) of 2nd address byte 7-bit address: Bit 0 (Least significant bit) (R/W) of address byte
1_0000	10-bit addressing: Bit 7 (Most significant bit) of 1st address byte
1_0001	10-bit addressing: Bit 6 of 1st address byte
1_0010	10-bit addressing: Bit 5 of 1st address byte
1_0011	10-bit addressing: Bit 4 of 1st address byte
1_0100	10-bit addressing: Bit 3 of 1st address byte
1_0101	10-bit addressing: Bit 2 of 1st address byte
1_0110	10-bit addressing: Bit 1 of 1st address byte
1_0111	10-bit addressing: Bit 0 (R/W) of 1st address byte
1_1000	10-bit addressing: Acknowledge state for 1st address byte
1_1001	10-bit addressing: Bit 7 of 2nd address byte 7-bit addressing: Bit 7 of address byte
1_1010	10-bit addressing: Bit 6 of 2nd address byte 7-bit addressing: Bit 6 of address byte
1_1011	10-bit addressing: Bit 5 of 2nd address byte 7-bit addressing: Bit 5 of address byte
1_1100	10-bit addressing: Bit 4 of 2nd address byte 7-bit addressing: Bit 4 of address byte

0101 = ADC Analog Inputs 0-5 updated.
0110 = ADC Analog Inputs 0-6 updated.
0111 = ADC Analog Inputs 0-7 updated.
1000 = ADC Analog Inputs 0-8 updated.
1001 = ADC Analog Inputs 0-9 updated.
1010 = ADC Analog Inputs 0-10 updated.
1011 = ADC Analog Inputs 0-11 updated.
1100-1111 = Reserved.

DMA Status Register

The DMA Status register (Table 85 on page 173) indicates the DMA channel that generated the interrupt and the ADC Analog Input that is currently undergoing conversion. Reads from this register reset the Interrupt Request Indicator bits (IRQA, IRQ1, and IRQ0) to 0. Therefore, software interrupt service routines that read this register must process all three interrupt sources from the DMA.

Table 85. DMA_ADC Status Register (DMAA_STAT)

BITS	7	6	5	4	3	2	1	0
FIELD	CADC[3:0]				Reserved	IRQA	IRQ1	IRQ0
RESET	0							
R/W	R							
ADDR	FBFH							

CADC[3:0]—Current ADC Analog Input
This field identifies the Analog Input that the ADC is currently converting.

Reserved
This bit is reserved and must be 0.

IRQA—DMA_ADC Interrupt Request Indicator
This bit is automatically reset to 0 each time a read from this register occurs.
0 = DMA_ADC is not the source of the interrupt from the DMA Controller.
1 = DMA_ADC completed transfer of data from the last ADC Analog Input and generated an interrupt.

IRQ1—DMA1 Interrupt Request Indicator
This bit is automatically reset to 0 each time a read from this register occurs.
0 = DMA1 is not the source of the interrupt from the DMA Controller.
1 = DMA1 completed transfer of data to/from the End Address and generated an interrupt.

IRQ0—DMA0 Interrupt Request Indicator
This bit is automatically reset to 0 each time a read from this register occurs.

0 = DMA0 is not the source of the interrupt from the DMA Controller.
1 = DMA0 completed transfer of data to/from the End Address and generated an interrupt.

Analog-to-Digital Converter

Overview

The Analog-to-Digital Converter (ADC) converts an analog input signal to a 10-bit binary number. The features of the sigma-delta ADC include:

- 12 analog input sources are multiplexed with general-purpose I/O ports
- Interrupt upon conversion complete
- Internal voltage reference generator
- Direct Memory Access (DMA) controller can automatically initiate data conversion and transfer of the data from 1 to 12 of the analog inputs

Architecture

Figure 34 displays the three major functional blocks (converter, analog multiplexer, and voltage reference generator) of the ADC. The ADC converts an analog input signal to its digital representation. The 12-input analog multiplexer selects one of the 12 analog input sources. The ADC requires an input reference voltage for the conversion. The voltage reference for the conversion may be input through the external VREF pin or generated internally by the voltage reference generator.

Flash Memory Address 0000H

Table 98. Flash Option Bits At Flash Memory Address 0000H

BITS	7	6	5	4	3	2	1	0
FIELD	WDT_RES	WDT_AO	OSC_SEL[1:0]		VBO_AO	RP	Reserved	FWP
RESET	U							
R/W	R/W							
ADDR	Program Memory 0000H							

Note: U = Unchanged by Reset. R/W = Read/Write.

WDT_RES—Watchdog Timer Reset
0 = Watchdog Timer time-out generates an interrupt request. Interrupts must be globally enabled for the eZ8 CPU to acknowledge the interrupt request.
1 = Watchdog Timer time-out causes a Short Reset. This setting is the default for unprogrammed (erased) Flash.

WDT_AO—Watchdog Timer Always On
0 = Watchdog Timer is automatically enabled upon application of system power. Watchdog Timer can not be disabled except during STOP Mode (if configured to power down during STOP Mode).
1 = Watchdog Timer is enabled upon execution of the WDT instruction. Once enabled, the Watchdog Timer can only be disabled by a Reset or Stop Mode Recovery. This setting is the default for unprogrammed (erased) Flash.

OSC_SEL[1:0]—Oscillator Mode Selection
00 = On-chip oscillator configured for use with external RC networks (<4 MHz).
01 = Minimum power for use with very low frequency crystals (32 kHz to 1.0 MHz).
10 = Medium power for use with medium frequency crystals or ceramic resonators (0.5 MHz to 10.0 MHz).
11 = Maximum power for use with high frequency crystals (8.0 MHz to 20.0 MHz). This setting is the default for unprogrammed (erased) Flash.

VBO_AO—Voltage Brownout Protection Always On
0 = Voltage Brownout Protection is disabled in STOP mode to reduce total power consumption.
1 = Voltage Brownout Protection is always enabled including during STOP mode. This setting is the default for unprogrammed (erased) Flash.

RP—Read Protect
0 = User program code is inaccessible. Limited control features are available through

eZ8™ CPU Instruction Set

Assembly Language Programming Introduction

The eZ8 CPU assembly language provides a means for writing an application program without having to be concerned with actual memory addresses or machine instruction formats. A program written in assembly language is called a source program. Assembly language allows the use of symbolic addresses to identify memory locations. It also allows mnemonic codes (opcodes and operands) to represent the instructions themselves. The opcodes identify the instruction while the operands represent memory locations, registers, or immediate data values.

Each assembly language program consists of a series of symbolic commands called statements. Each statement can contain labels, operations, operands and comments.

Labels can be assigned to a particular instruction step in a source program. The label identifies that step in the program as an entry point for use by other instructions.

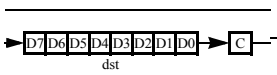
The assembly language also includes assembler directives that supplement the machine instruction. The assembler directives, or pseudo-ops, are not translated into a machine instruction. Rather, the pseudo-ops are interpreted as directives that control or assist the assembly process.

The source program is processed (assembled) by the assembler to obtain a machine language program called the object code. The object code is executed by the eZ8 CPU. An example segment of an assembly language program is detailed in the following example.

Assembly Language Source Program Example

```
JP START          ; Everything after the semicolon is a comment.
START:            ; A label called “START”. The first instruction (JP START) in this
                  ; example causes program execution to jump to the point within the
                  ; program where the START label occurs.
LD R4, R7         ; A Load (LD) instruction with two operands. The first operand,
                  ; Working Register R4, is the destination. The second operand,
                  ; Working Register R7, is the source. The contents of R7 is
                  ; written into R4.
LD 234H, #%01     ; Another Load (LD) instruction with two operands.
                  ; The first operand, Extended Mode Register Address 234H,
                  ; identifies the destination. The second operand, Immediate Data
```

Table 133. 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
RRC dst		R		C0	*	*	*	*	-	-	2	2
		IR		C1							2	3
SBC dst, src	dst ← dst – src - C	r	r	32	*	*	*	*	1	*	2	3
		r	Ir	33							2	4
		R	R	34							3	3
		R	IR	35							3	4
		R	IM	36							3	3
		IR	IM	37							3	4
SBCX dst, src	dst ← dst – src - C	ER	ER	38	*	*	*	*	1	*	4	3
		ER	IM	39							4	3
SCF	C ← 1			DF	1	-	-	-	-	-	1	2
SRA dst		R		D0	*	*	*	0	-	-	2	2
		IR		D1							2	3
SRL dst		R		1F C0	*	*	0	*	-	-	3	2
		IR		1F C1							3	3
SRP src	RP ← src		IM	01	-	-	-	-	-	-	2	2
STOP	STOP Mode			6F	-	-	-	-	-	-	1	2
SUB dst, src	dst ← dst – src	r	r	22	*	*	*	*	1	*	2	3
		r	Ir	23							2	4
		R	R	24							3	3
		R	IR	25							3	4
		R	IM	26							3	3
		IR	IM	27							3	4
SUBX dst, src	dst ← dst – src	ER	ER	28	*	*	*	*	1	*	4	3
		ER	IM	29							4	3

Opcode Maps

A description of the opcode map data and the abbreviations are provided in Figure 59 and Table 134 on page 262. Figure 60 on page 263 and Figure 61 on page 264 provide information on each of the eZ8™ CPU instructions.

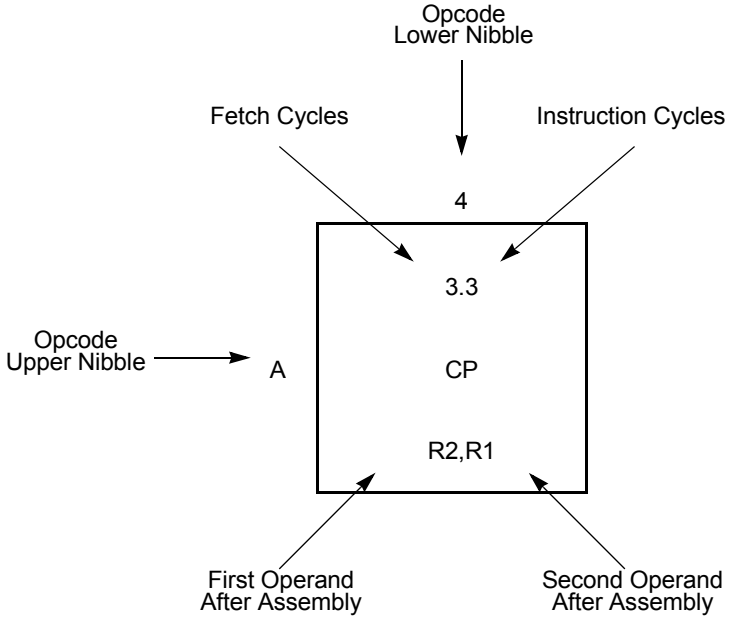


Figure 59. Opcode Map Cell Description

M

N

O