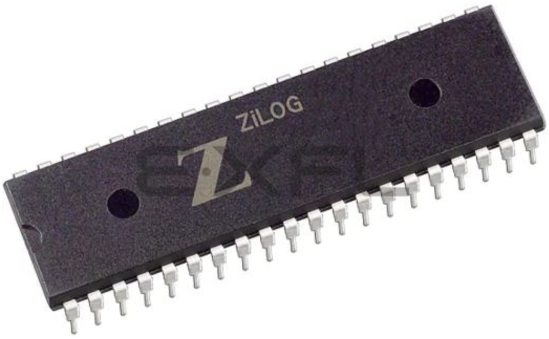
Zilog - Z88C0020PSG Datasheet



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	Z8
Core Size	8-Bit
Speed	20MHz
Connectivity	UART/USART
Peripherals	DMA
Number of I/O	32
Program Memory Size	-
Program Memory Type	ROMless
EEPROM Size	-
RAM Size	128K x 8
Voltage - Supply (Vcc/Vdd)	4.75V ~ 5.25V
Data Converters	-
Oscillator Type	Internal
Operating Temperature	0°C ~ 70°C (TA)
Mounting Type	Through Hole
Package / Case	40-DIP (0.620", 15.75mm)
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/zilog/z88c0020psg

Table of Contents

FEATURES

- Improved Z8® instruction set includes multiply and divide instructions, Boolean and BCD operations.
- Additional instructions support threaded-code languages, such as "Forth."
- 325 byte registers, including 272 general-purpose registers, and 53 mode and control registers.
- Addressing of up to 128K bytes of memory. Two register pointers allow use of short and fast instructions to access register groups within 600 nsec.
- Direct Memory Access controller (DMA).
- Two 16-bit counter/timers.
- Up to 32 bit-programmable and 8 byte-programmable I/O lines, with 2 handshake channels.
- Interrupt structure supports:
  - 27 interrupt sources
  - 16 interrupt vectors (2 reserved for future versions)
  - 8 interrupt levels
  - Servicing in 600 nsec. (1 level only)
- Full-duplex UART with special features.
- On-chip oscillator.
- 20 MHz clock.
- 8K byte ROM for Z8820

GENERAL DESCRIPTION

The Zilog Super8 single-chip MCU can be used for development and production. It can be used as I/O- or memory-intensive computers, or configured to address external memory while still supporting many I/O lines.

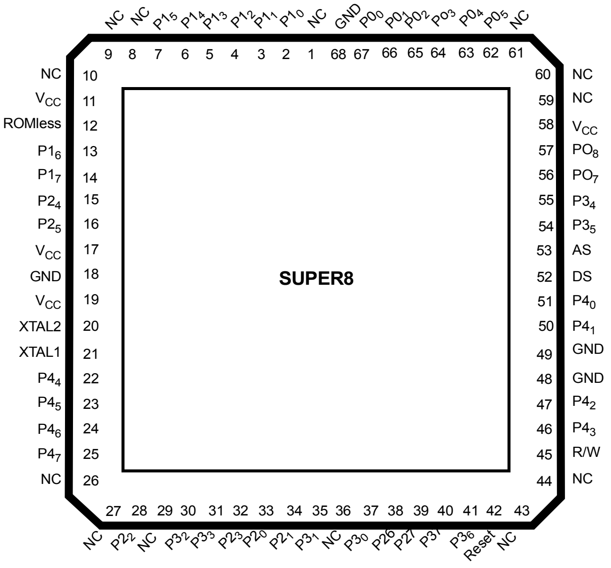


Figure 1.Pin Assignments 88-Pin PLCC

The Super8 features a full-duplex universal asynchronous receiver/ transmitter (UART) with on-chip baud rate generator, two programmable counter/timers, a direct memory access (DMA) controller, and an on-chip oscillator.

The Super8 is also available as a 48-pin and 68-pin ROMless microcomputer with four byte-wide I/O ports plus a byte-wide address/data bus. Additional address bits can be configured, up to a total of 16.

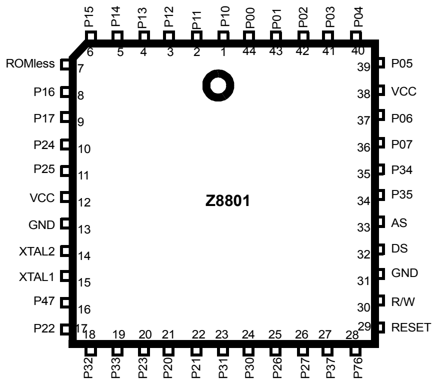


Figure 2.Pin Assignments 44-Pin PLCC

The 16-bit counters can operate independently or be cascaded to perform 32-bit counting and timing operations. The DMA controller handles transfers to and from the register file or memory. DMA can use the UART or one of two ports with handshake capability.

The architecture appears in the block diagram (Figure 7).

PIN DESCRIPTIONS

The Super8 connects to external devices via the following TTL-compatible pins:

$\overline{\text{AS}}$. *Address Strobe* (output, active Low). $\overline{\text{AS}}$ is pulsed Low once at the beginning of each machine cycle. The rising edge indicates that addresses R/$\overline{\text{W}}$ and $\overline{\text{DM}}$, when used, are valid.

$\overline{\text{DS}}$. *Data Strobe* (output, active Low). $\overline{\text{DS}}$ provides timing for data movement between the address/data bus and external memory. During write cycles, data output is valid at the leading edge of $\overline{\text{DS}}$. During read cycles, data input must be valid prior to the trailing edge of $\overline{\text{DS}}$.

P0_0-P0_7, P1_0-P1_7, P2_0-P2_7, P3_0-P3_7, P4_0-P4_7. *Port I/O Lines* (input/output). These 40 lines are divided into five 8-bit I/O ports that can be configured under program control for I/O or external memory interface.

In the ROMless devices, Port 1 is dedicated as a multiplexed address/data port, and Port 0 pins can be assigned as additional address lines; Port 0 non-address pins may be assigned as I/O. In the ROM and protopack, Port 1 can be assigned as input or output, and Port 0 can be assigned as input or output on a bit by bit basis.

Ports 2 and 3 can be assigned on a bit-for-bit basis as general I/O or interrupt lines. They can also be used as special-purpose I/O lines to support the UART, counter/timers, or handshake channels.

Port 4 is used for general I/O.

During reset, all port pins are configured as inputs (high impedance) except for Port 1 and Port 0 in the ROMless devices. In these, Port 1 is configured as a multiplexed address/data bus, and Port 0 pins PO_0-PO_4 are configured as address out, while pins P0_5-PO_7 are configured as inputs.

$\overline{\text{RESET}}$. *Reset* (input, active Low). Reset initializes and starts the Super8. When it is activated, it halts all processing; when it is deactivated, the Super8 begins processing at address 0020H.

ROMless. (input, active High). This input controls the operation mode of a 68-pin Super8. When connected to VCC, the part functions as a ROMless Z8800. When connected to GND, the part functions as a Z8820 ROM part.

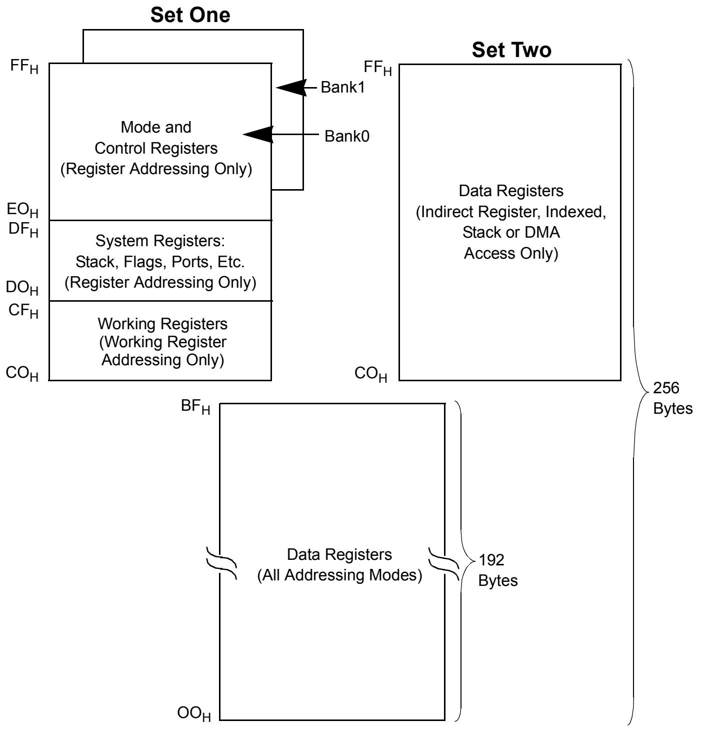


Figure 8.Super8 Registers

Working Register Window

Control registers R214 and R215 are the register pointers, RPO and RP1. They each define a moveable, 8-register section of the register space. The registers within these spaces are called working registers.

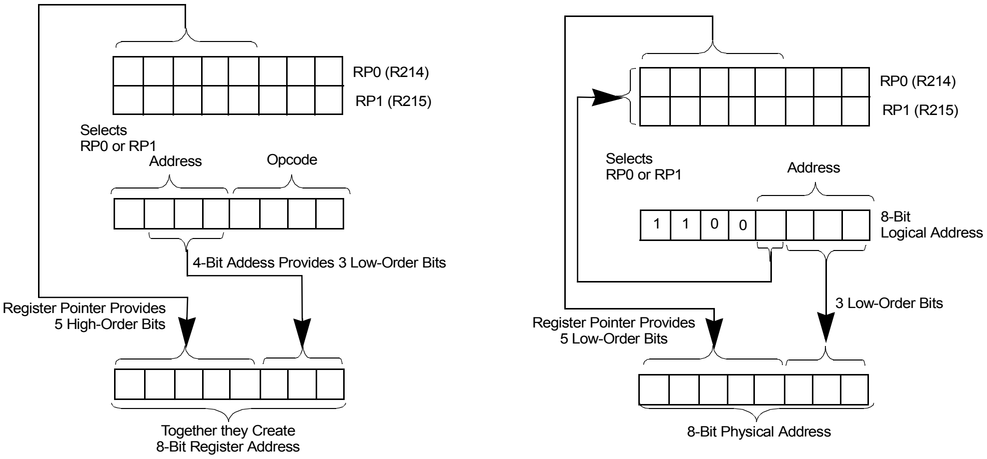
Working registers can be accessed using short 4-bit addresses. The process, shown in section a of Figure 9, works as follows:

- The high-order bit of the 4-bit address selects one of the two register pointers (0 selects RPO; 1 selects RP1).
- The five high-order bits in the register pointer select an 8-register (contiguous) slice of the register space.
- The three low-order bits of the 4-bit address select one of the eight registers in the slice.

The net effect is to concatenate the five bits from the register pointer to the three bits from the address to form an 8-bit address. As long as the address in the register pointer remains unchanged, the three bits from the address always point to an address within the same eight registers.

The register pointers can be moved by changing the five high bits in control registers R214 for RP0 and R215 for RP1.

The working registers can also be accessed by using full 8-bit addressing. When an 8-bit logical address in the range 192 to 207 (CO to CF) is specified, the lower nibble is used similarly to the 4-bit addressing described above. This is shown in section b of Figure 9.



a. 4-Bit Addressing

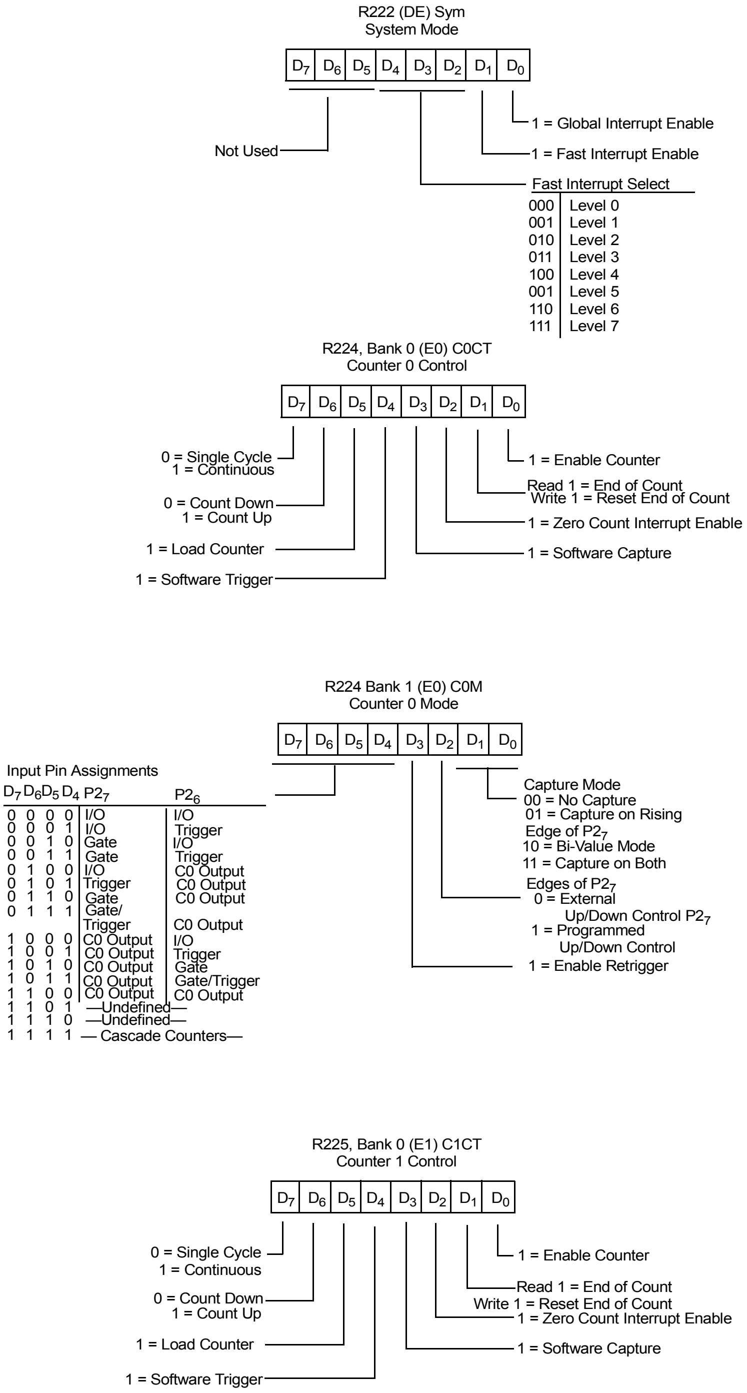
b. 8-Bit Addressing

Figure 9.Working Register Window

Since any direct access to logical addresses 192 to 207 involves the register pointers, the physical registers 192 to 207 can be accessed only when selected by a register pointer. After a reset, RPO points to R192 and RP1 points to R200.

Register List

Super-8 Registers lists the Super8 registers. For more details, see Figure 10.



Fast Interrupt Select	
000	Level 0
001	Level 1
010	Level 2
011	Level 3
100	Level 4
001	Level 5
110	Level 6
111	Level 7

Input Pin Assignments

D_7	D_6	D_5	D_4	$P2_7$	$P2_6$
0	0	0	0	I/O	I/O
0	0	0	1	I/O	Trigger
0	0	1	0	Gate	I/O
0	0	1	1	Gate	Trigger
0	1	0	0	I/O	C0 Output
0	1	0	1	Trigger	C0 Output
0	1	1	0	Gate	C0 Output
0	1	1	1	Gate/ Trigger	C0 Output
1	0	0	0	C0 Output	I/O
1	0	0	1	C0 Output	Trigger
1	0	1	0	C0 Output	Gate
1	0	1	1	C0 Output	Gate/Trigger
1	1	0	0	C0 Output	C0 Output
1	1	0	1	—Undefined—	
1	1	1	0	—Undefined—	
1	1	1	1	— Cascade Counters—	

Figure 11.Mode and Control Registers (Continued)

Port 1

In the ROMless device, Port 1 is configured as a byte-wide address/data port. It provides a byte-wide multiplexed address/data path. Additional address lines can be added by configuring Port 0.

The ROM and Protopack Port 1 can be configured as above or as an I/O port; it can be a byte-wide input, open-drain output, or push-pull output. It can be placed under handshake control or handshake channel 0.

Ports 2 and 3

Ports 2 and 3 provide external control inputs and outputs for the UART, handshake channels, and counter/timers. The pin assignments appear in Table 3.

Bits not used for control I/O can be configured as general-purpose I/O lines and/or external interrupt inputs.

Those bits configured for general I/O can be configured individually for input or output. Those configured for output can be individually configured for open-drain or push-pull output.

All Port 2 and 3 input pins are Schmitt-triggered.

The port address for Port 2 is R210, and for Port 3 is R211.

Table 17.Pin Assignments for Ports 2 and 3

Port 2		Port 3	
Bit	**Function**	**Bit**	**Function**
0	UART receive clock	0	UART receive data
1	UART transmit clock	1	UART transmit data
2	Reserved	2	Reserved
3	Reserved	3	Reserved
4	Handshake 0 input	4	Handshake 1 input/$\overline{\text{WAIT}}$
5	Handshake 0 output	5	Handshake 1 output/$\overline{\text{DM}}$
6	Counter 0 input	6	Counter 1 input
7	Counter 0 I/O	7	Counter 1 I/O

Port 4

Port 4 can be configured as I/O only. Each bit can be configured individually as input or output, with either push-pull or open-drain outputs. All Port 4 inputs are Schmitt-triggered.

Port 4 can be placed under handshake control of handshake channel 0. Its register address is R212.

UART

The UART is a full-duplex asynchronous channel. It transmits and receives independently with 5 to 8 bits per character, has options for even or odd bit parity, and a wake-up feature.

Data can be read into or out of the UART via R239, Bank 0. This single address is able to serve a full-duplex channel because it contains two complete 8-bit registers-one for the transmitter and the other for the receiver.

Pins

The UART uses the following Port 2 and 3 pins:

Port/Pin	UART Function
2/0	Receive Clock
3/0	Receive Data
2/1	Transmit Clock
3/1	Transmit Data

Transmitter

When the UART's register address is specified as the destination (dst) of an operation, the data is output on the UART, which automatically adds the start bit, the programmed parity bit, and the programmed number of stop bits. It can also add a wake-up bit if that option is selected.

If the UART is programmed for a 5-, 6-, or 7-bit character, the extra bits in R239 are ignored.

Serial data is transmitted at a rate equal to 1, 1/16, 1/32 or 1/64 of the transmitter clock rate, depending on the programmed data rate. All data is sent out on the falling edge of the clock input.

When the UART has no data to send, it holds the output marking (High). It may be programmed with the Send Break command to hold the output Low (Spacing), which it continues until the command is cleared.

Receiver

The UART begins receive operation when Receive Enable (URC, bit 0) is set High. After this, a Low on the receive input pin for longer than half a bit time is interpreted as a start bit. The UART samples the data on the input pin in the middle of each clock cycle until a complete byte is assembled. This is placed in the Receive Data register.

If the 1 X clock mode is selected, external bit synchronization must be provided, and the input data is sampled on the rising edge of the clock.

For character lengths of less than eight bits, the UART inserts ones into the unused bits, and, if parity is enabled, the parity bit is not stripped. The data bits, extra ones, and the parity bit are placed in the UART Data register (UIO).

While the UART is assembling a byte in its input shift register, the CPU has time to service an interrupt and manipulate the data character in UIO.

Once a complete character is assembled, the UART checks it and performs the following:

- If it is an-ASCII control character, the UART sets the Control Character status bit.
- It checks the wake-up settings and completes any indicated action.
- If parity is enabled, the UART checks to see if the calculated parity matches the programmed parity bit. If they do not match, it sets the Parity Error bit in URC (R236 Bank 0), which remains set until reset by software.
- It sets the Framing Error bit (URC, bit 4) if the character is assembled without any stop bits. This bit remains set until cleared by software.

Overrun errors occur when characters are received faster than they are read. That is, when the UART has assembled a complete character before the CPU has read the current character, the UART sets the Overrun Error bit (URC, bit 3), and the character currently in the receive buffer is lost.

The overrun bit remains set until cleared by software.

ADDRESS SPACE

The Super8 can access 64K bytes of program memory and 64K bytes of data memory. These spaces can be either combined or separate. If separate, they are

Carry Flag. This flag is set to 1 if the result from an arithmetic operation generates a carry out of, or a borrow into, bit 7.

After rotate and shift operations, it contains the last value shifted out of the specified register.

It can be set, cleared, or complemented by instructions.

Condition Codes

The flags C, Z, S, and V are used to control the operation of conditional jump instructions.

The opcode of a conditional jump contains a 4-bit field called the condition code (cc). This specifies under which conditions it is to execute the jump. For example, a conditional jump with the condition code for "equal" after a compare operation only jumps if the two operands are equal.

The condition codes and their meanings are given in Condition Codes and Meanings.

Addressing Modes

All operands except for immediate data and condition codes are expressed as register addresses, program memory addresses, or data memory addresses. The addressing modes and their designations are:

- Register (R)
- Indirect Register (IR)
- Indexed (X)
- Direct (DA)
- Relative (RA)
- Immediate (IM)
- Indirect (IA)

Table 18.Condition Codes and Meanings

Binary	Mnemonic	Flags	Meaning
0000	F	-	Always false
1000	-	-	Always true
0111[1]	C	C = 1	Carry
1111[1]	NC	C = 0	No carry

Table 20.Instruction Summary (Continued)

Instruction and Operation	Address Mode		Opcode Byte (Hex)	Flags Affected					
	dst	src		C	Z	S	V	D	H
SMR(0)←0									
DIV dst, src									
dst ÷ src	RR	R	94	*	*	*	*	-	-
dst (Upper)←Quotient	RR	IR	95						
dst (Lower)←Remainder	RR	IM	96						
DJNZ r, dst	RA	r	rA	-	-	-	-	-	-
r←r - 1			(r = 0 to F)						
if r = 0									
PC←PC + dst									
EI			9F	-	-	-	-	-	-
SMR(0)←1									
ENTER			1F	-	-	-	-	-	-
SP←SP - 2									
@SP←IP									
IP←PC									
PC←@IP									
IP←IP + 2									
EXIT			2F	-	-	-	-	-	-
IP←@SP									
SP←SP + 2									
PC←@IP									
IP←IP + 2									
INC dst	r		rE	-	*	*	*	-	-
dst←dst + 1			(r = 0 to F)						
	R		20						
	IR		21						

Table 20.Instruction Summary (Continued)

Instruction and Operation	Address Mode		Opcode Byte (Hex)	Flags Affected					
	dst	src		C	Z	S	V	D	H
INCW dst	RR		A0	-	*	*	*	-	-
dst←1 + dst	IR		A1						
IRET (Fast)			BF	Restored to before interrupt					
PC↔IP									
FLAG←FLAG‘									
FIS←0									
IRET (Normal)			BF	Restored to before interrupt					
FLAGS←@SP; SP←SP + 1									
PC←@SP; SP←SP + 2; SMR(0)←1									
JP cc, dst	DA		ccD	-	-	-	-	-	-
if cc is true,			(cc = 0 to F)						
PC←dst	IRR		30						
JR cc, dst	RA		ccB	-	-	-	-	-	-
if cc is true,									
PC←PC + d			(cc = 0 to F)						
LD dst, src	r	IM	rC	-	-	-	-	-	-
dst←src	r	R	r8						
	R	r	r9						
			(r = 0 to F)						
	r	IR	C7						
	IR	r	D7						
	R	R	E4						
	R	IR	E5						
	R	IM	E6						
	IR	IM	D6						
	IR	R	F5						
	r	x	87						
	x	r	97						

SUPER-8 OPCODE MAP

Lower Nibble (Hex)

Upper Nibble (Hex)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	6 **DEC** R_1	6 **DEC** IR_1	6 **ADD** r_1,r_2	6 **ADD** r_1,Ir_2	10 **ADD** R_2,R_1	10 **ADD** IR_2,R_1	10 **ADD** R_1,IM	10 **BOR*** r_0-R_b	6 **LD** r_1,R_2	6 **LD** r_2,R_1	12/10 **DJNZ** r_1,RA	12/10 **JR** cc,RA	6 **LD** r_1,RM	12/10 **JP** cc,DA	6 **INC** r_1	14 **NEXT**
1	6 **RLC** R_1	6 **RLC** IR_1	6 **ADC** r_1,r_2	6 **ADC** r_1,Ir_2	10 **ADD** R_2,R_1	10 **ADC** IR_2,R_1	10 **ADC** R_1,IM	10 **BCP** r_1,b,R_2	↓	↓	↓	↓	↓	↓	↓	20 **ENTER**
2	6 **INC** R_1	6 **INC** IR_1	6 **SUB** r_1,r_2	6 **SUB** r_1,Ir_2	10 **SUB** R_2,R_1	10 **SUB** IR_2,R_1	10 **SUB** R_1,IM	10 **BXOR*** r_0-R_b	↓	↓	↓	↓	↓	↓	↓	22 **EXIT**
3	10 **JP** IRR_1	**NOTE** **C**	6 **SBC** r_1,r_2	6 **SBC** r_1,Ir_2	10 **SBC** R_2,R_1	10 **SBC** IR_2,R_1	10 **SBC** R_1,IM	**NOTE** **A**	↓	↓	↓	↓	↓	↓	↓	6 **WFI**
4	6 **DA** R_1	6 **DA** IR_1	6 **OR** r_1,r_2	6 **OR** r_1,Ir_2	10 **OR** R_2,R_1	10 **OR** IR_2,R_1	10 **OR** R_1,IM	10 **LDB*** r_0-Rb	↓	↓	↓	↓	↓	↓	↓	6 **SBO**
5	10 **POP** R_1	10 **POP** IR_1	6 **AND** r_1,r_2	6 **AND** r_1,Ir_2	10 **AND** R_2,R_1	10 **AND** IR_2,R_1	10 **AND** R_1,IM	10 **BITC** r_1,b	↓	↓	↓	↓	↓	↓	↓	6 **SBI**
6	6 **COM** R_1	6 **COM** IR_1	6 **TCM** r_1,r_2	6 **TCM** r_1,Ir_2	10 **TCM** R_2,R_1	10 **TCM** IR_2,R_1	10 **TCM** R_1,IM	10 **BAND*** r_0-Rb	↓	↓	↓	↓	↓	↓	↓	
7	10/12 **PUSH** R_2	12/14 **PUSH** IR_2	6 **TM** r_1,r_2	6 **TM** r_1,Ir_2	10 **TM** R_2,R_1	10 **TM** IR_2,R_1	10 **TM** R_1,IM	**NOTE** **B**	↓	↓	↓	↓	↓	↓	↓	
8	10 **DECW** IRR_1	10 **DECW** IR_1	10 **PUSHUD** IR_1,R_2	10 **PUSHUI** IR_2,R_2	24 **MULT** R_2,RR_1	24 **MULT** IR_2,RR_1	24 **MULT** IM,RR_1	10 **LD** r_1,x,r_2	↓	↓	↓	↓	↓	↓	↓	6 **DI**
9	6 **RL** R_1	6 **RL** IR_1	6 **POPUD** IR_2,R_1	6 **POPUI** IR_2,R_1	28/12 **DIV** R_2,RR_1	28/12 **DIV** IR_2,RR_1	28/12 **DIV** IM,RR_1	10 **LD** r_2,x,r_1	↓	↓	↓	↓	↓	↓	↓	6 **EI**
A	10 **INCW** RR_1	10 **INCW** IR_1	6 **CP** r_1,r_2	6 **CP** r_1,Ir_2	10 **CP** R_2,R_1	10 **CP** IR_2,R_1	10 **CP** R_1,IM	**NOTE** **D**	↓	↓	↓	↓	↓	↓	↓	14 **RET**
B	6 **CLR** R_1	6 **CLR** IR_1	6 **XOR** r_1,r_2	6 **XOR** r_1,Ir_2	10 **XOR** R_2,R_1	10 **XOR** IR_2,R_1	10 **XOR** R_1,IM	**NOTE** **E**	↓	↓	↓	↓	↓	↓	↓	16/6 **IRET**
C	6 **RRC** IRR_1	6 **RRC** IR_1	16/18 **CPIJE** Ir,r_2,RA	12 **LDC*** r_1,Irr_2	10 **LDW** RR_2,RR_1	10 **LDW** IR_2,RR_1	12 **LDW** RR_1,IML	6 **LD** r_1,Ir_2	↓	↓	↓	↓	↓	↓	↓	6 **RCF**
D	6 **SRA** R_1	6 **SRA** IR_1	16/18 **CPIJNE** Ir_1,r_2,RA	12 **LDC*** r_2,Irr_1	20 **CALL** IA_1		10 **LD** IR_1,IM	6 **LD** $Ir1,r_2$	↓	↓	↓	↓	↓	↓	↓	6 **SCF**
E	6 **RR** R_1	6 **RR** IR_1	16 **LDCD*** r_1,Irr_2	16 **LDCI*** r_1,Irr_2	10 **LD** R_2,R_1	10 **LD** IR_2,R_1	10 **LD** R_1,IM	18 **LDC*** r_1,Irr_2,xs	↓	↓	↓	↓	↓	↓	↓	6 **CCF**
F	8 **SWAP** R_1	8 **SWAP** IR_1	16 **LDCPD*** r_2,Irr_1	16 **LDCPI*** r_2,Irr_1	18 **CALL** IRR_1	18 **CALL** R_2,IR_1	18 **CALL** DA_1	18 **LDC*** r_1,Irr_1,xs	↓	↓	↓	↓	↓	↓	↓	6 **NOP**

Note A

16/18 **BTJRF** r_2,b,RA	16/18 **BTJRT** r_2,b,RA

Note B

8 **BITR** r_1,b	8 **BITS** r_1,b

Note C

6 **SRP** IM	6 **SRP0** IM	6 **SRP1** IM

Note D

20 **LDC*** r_1,Irr_2,xL	20 **LDC*** r_1,DA_2

Note E

20 **LDC*** r_2,Irr_2,xL	20 **LDC*** r_2,DA_1

Legend:
r = 4-bit address
R = 8-bit address
b = bit number
R_1 or r_1 = dsts address
R_2 or r_2 = src address

***Examples:**
BORr_0-R_2
is BORr_1,b,R_2
or BORr_2,b,R_1
LDCr_1,Irr_2
is LDCr_1,Irr_2 = program
or LDEr_1,Irr_2 = data

Sequence:
Opcode, first, second, third operands

NOTE: The blank areas are not defined.

Figure 19.Opcode Map

INSTRUCTIONS

Table 22.Super8 Instructions

Mnemonic	Operands	Instruction
Load Instructions		
CLR	dst	Clear
LD	dst, src	Load
LDB	dst, src	Load bit
LDC	dst, src	Load program memory
LDE	dst, src	Load data memory
LDCD	dst, src	Load program memory and decrement
LDED	dst, src	Load data memory and decrement
LDCI	dst, src	Load program memory and increment
LDEI	dst, src	Load data memory and increment
LDCPD	dst, src	Load program memory with pre-decrement
LDEPD	dst, src	Load data memory with pre-decrement
LDCPI	dst, src	Load program memory with pre-increment
LDEPI	dst, src	Load data memory with pre-increment
LDW	dst, src	Load word
POP	dst	Pop stack
POPUD	dst, src	Pop user stack (decrement)
POPUI	dst, src	Pop user stack (increment)
PUSH	src	Push stack
PUSHUD	dst, src	Push user stack (decrement)
PUSHUI	dst, src	Push user stack (increment)
Arithmetic Instructions		
ADC	dst, src	Add with carry
ADD	dst, src	Add
CP	dst, src	Compare
DA	dst	Decimal adjust
DEC	dst	Decrement

Table 22.Super8 Instructions (Continued)

Mnemonic	Operands	Instruction
SRP1	src	Set register pointer one

INTERRUPTS

The Super8 interrupt structure contains 8 levels of interrupt, 16 vectors, and 27 sources.

Interrupt priority is assigned by level, controlled by the Interrupt Priority register (IPR). Each level is masked (or enabled) according to the bits in the Interrupt Mask register (IMR), and the entire interrupt structure can be disabled by clearing a bit in the System Mode register (R222).

The three major components of the interrupt structure are sources, vectors, and levels. These are shown in Figure 20 and discussed in the following paragraphs.

Sources

A source is anything that generates an interrupt. This can be internal or external to the Super8 MCU. Internal sources are hardwired to a particular vector and level, while external sources can be assigned to various external events.

Vectors

The 16 vectors are divided unequally among the eight levels. For example, vector 12 belongs to level 2, while level 3 contains vectors 0, 2, 4, and 6.

The vector number is used to generate the address of a particular interrupt servicing routine; therefore all interrupts using the same vector must use the same interrupt handling routine.

Levels

Levels provide the top level of priority assignment. While the sources and vectors are hardwired within each level, the priorities of the levels can be changed by using the Interrupt Priority register (see Figure 15 for bit details).

If more than one interrupt source is active, the source from the highest priority level is serviced first. If both sources are from the same level, the source with the lowest vector has priority. For example, if the UART Receive Data bit and UART Parity Error bit are both active, the UART Parity Error bit is serviced first because it is vector 16, and UART receive data is vector 20.

Service Routines

Before an interrupt request can be granted, a) interrupts must be enabled, b) the level must be enabled, c) it must be the highest priority interrupting level, d) it must be enabled at the interrupting source, and e) it must have the highest priority within the level.

If all this occurs, an interrupt request is granted.

The Super8 then enters an interrupt machine cycle that completes the following sequence:

- It resets the Interrupt Enable bit to disable all subsequent interrupts.
- It saves the Program Counter and status flags on the stack.
- It branches to the address contained within the vector location for the interrupt.
- It passes control to the interrupt servicing routine.

When the interrupt servicing routine has serviced the interrupt, it should issue an interrupt return (IRET) instruction. This restores the Program Counter and status flags and sets the Interrupt Enable bit in the System Mode register.

Fast Interrupt Processing

The Super8 provides a feature called fast interrupt processing, which completes the interrupt servicing in 6 clock periods instead of the usual 22.

Two hardware registers support fast interrupts. The Instruction Pointer (IP) holds the starting address of the service routine, and saves the PC value when a fast interrupt occurs. A dedicated register, FLAG', saves the contents of the FLAGS register when a fast interrupt occurs.

To use this feature, load the. address of the service routine in the Instruction Pointer, load the level number into the Fast Interrupt Select field, and turn on the Fast Interrupt Enable bit in the System Mode register.

When an interrupt occurs in the level selected for fast interrupt processing, the following occurs:

- The contents of the Instruction Pointer and Program Counter are swapped.
- The contents of the Flag register are copied into FLAG'.
- The Fast Interrupt Status Bit in FLAGS is set.
- The interrupt is serviced.
- When IRET is issued after the interrupt service outline is completed, the Instruction Pointer and Program Counter are swapped again.

- The contents of FLAG’ are copied back into the Flag register
- The Fast Interrupt Status bit in FLAGS is cleared.

The interrupt servicing routine selected for fast processing should be written so that the location after the IRET instruction is the entry point the next time the (same) routine is used.

Level or Edge Triggered

Because internal interrupt requests are levels and interrupt requests from the outside are (usually) edges, the hardware for external interrupts uses edge-triggered flip-flops to convert the edges to levels.

The level-activated system requires that interrupt-serving software perform some action to remove the interrupting source. The action involved in serving the interrupt may remove the source, or the software may have to actually reset the flip-flops by writing to the corresponding Interrupt Pending register.

STACK OPERATION

The Super8 architecture supports stack operations in the register file or in data memory. Bit 1 in the external Memory Timing register (R254 bank 0) selects between the two.

Register pair 216-217 forms the Stack Pointer used for all stack operations. R216 is the MSB and R217 is the LSB.

The Stack Pointer always points to data stored on the top of the stack. The address is decremented prior to a PUSH and incremented after a POP.

The stack is also used as a return stack for CALLs and interrupts. During a CALL, the contents of the PC are saved on the stack, to be restored later. Interrupts cause the contents of the PC and FLAGS to be saved on the stack, for recovery by IRET when the interrupt is finished.

When the Super8 is configured for an internal stack (using the register file), R217 contains the Stack Pointer. R216 may be used as a general-purpose register, but its contents are changed if an overflow or underflow, occurs as the result of incrementing or decrementing the stack address during normal stack operations.

User-Defined Stacks

The Super8 provides for user-defined stacks in both the register file and program or data memory. These can be made to increment or decrement on a push by the choice of opcodes. For example, to implement a stack that grows from low

DC CHARACTERISTICS

Table 23.DC Characteristics

Symbol	Parameter	Min	Max	Unit	Condition
V_{CH}	Clock Input High Voltage	3.8	V_{CC}	V	Driven by External Clock Generator
V_{CL}	Clock Input Low Voltage	-0.3	0.8	V	Driven by External Clock Generator
V_{IH}	Input High Voltage	2.2	V_{CC}	V	
V_{IL}	Input Low Voltage	-0.3	0.8	V	
V_{RH}	Reset Input High Voltage	3.8	V_{CC}	V	
V_{RL}	Reset Input Low Voltage	-0.3	0.8	V	
V_{OH}	Output High Voltage	2.4		V	I_{OH} = -400 µA
V_{OL}	Output Low Voltage		0.4	V	I_{OL} = +4.0 mA
I_{IL}	Input Leakage	-10	10	µA	
I_{OL}	Output Leakage	-10	10	µA	
I_{IR}	Reset Input Current		-50	µA	
I_{CC}	VCC Supply Current		320	rnA	

INPUT HANDSHAKE TIMING

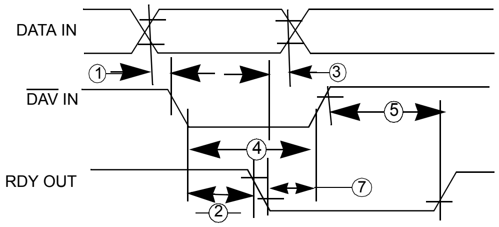


Figure 22.Fully Interlocked Mode

OUTPUT HANDSHAKE TIMING

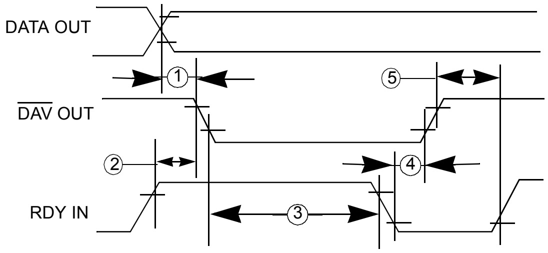


Figure 24.Fully Interlocked Mode

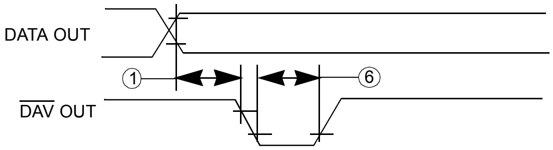


Figure 25.Strobed Mode

AC CHARACTERISTICS (12 MHz, 20 MHz)

Output Handshake

Table 25.AC Characteristics (12 MHz, 20 MHz) Output Handshake

Number	Symbol	Parameter	Min	Max	Notes[1,2]
1	TdDO(DAV)	Data Out to $\overline{\text{DAV}}$ ↓ Delay	90		Note[3,4]
2	TdRDYr(DAV)	RDY ↑ Input to $\overline{\text{DAV}}$ ↓ Delay	0	110	Note[3]
3	TdDAVOf(RDY)	$\overline{\text{DAV}}$ ↓ Output to RDY ↓ Delay	0		
4	TdRDYf(DAV)	RDY ↓ Input to $\overline{\text{DAV}}$ ↑ Delay	0	110	Note[3]
5	TdDAVOr(RDY)	$\overline{\text{DAV}}$ ↑ Output to RDY ↑ Delay	0		
6	TwDAVO	$\overline{\text{DAV}}$ Output Width	150		Note[4]

[1] Times are preliminary and subject to change.
[2] Times given are in ns.
[3] Standard Test Load
[4] Time given is for zero value in Deskew Counter. For nonzero value of n where n = 1, 2,. . . 15 add 2 x n x TpC to the given time.