Product Specification

Super8™ MCU ROMless, ROM, and Prototyping Device with EPROM Interface

Z8800, Z8801, Z8820, Z8822

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.

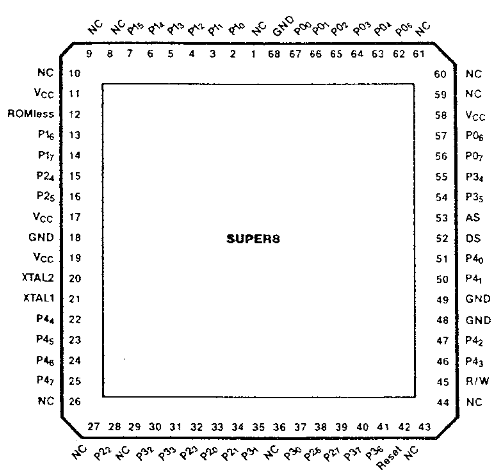
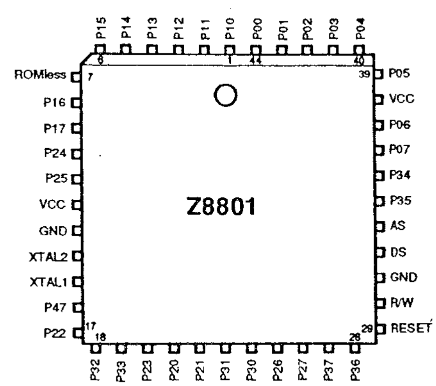


Figure 1a. Pin Assignments — 68-pin PLCC

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.



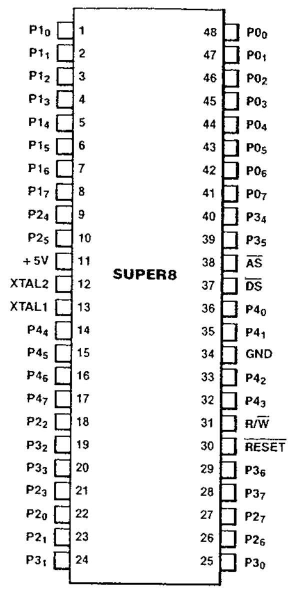


Figure 1b. Pin Assignments — 48-pin DIP

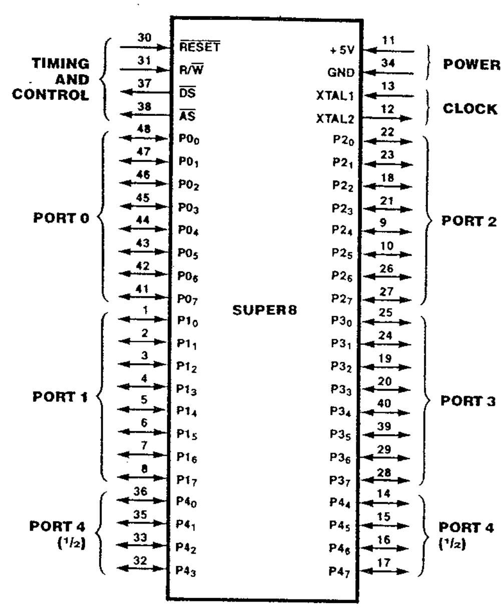


Figure 2. Pin Functions

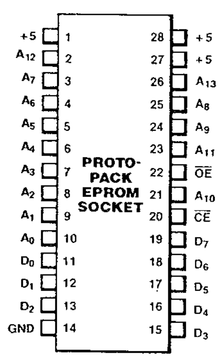


Figure 3. Pin Assignments—28-Pin Piggyback Socket

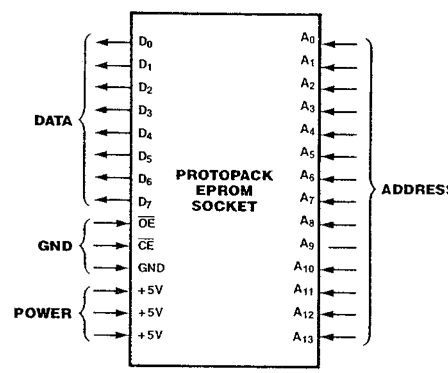


Figure 4. Pin Functions—28-Pin Piggyback Socket

Protopack

This part functions as an emulator for the basic microcomputer. It uses the same package and pin-out as the basic microcomputer but also has a 28-pin "piggy back" socket on the top into which a ROM or EPROM can be installed. The socket is designed to accept a type 2764 EPROM.

This package permits the protopack to be used in prototype and final PC boards while still permitting user program development. When a final program is developed, it can be mask-programmed into the production microcomputer device, directly replacing the emulator. The protopack part is also useful in situations where the cost of mask-programming is prohibitive or where program flexibility is desired.

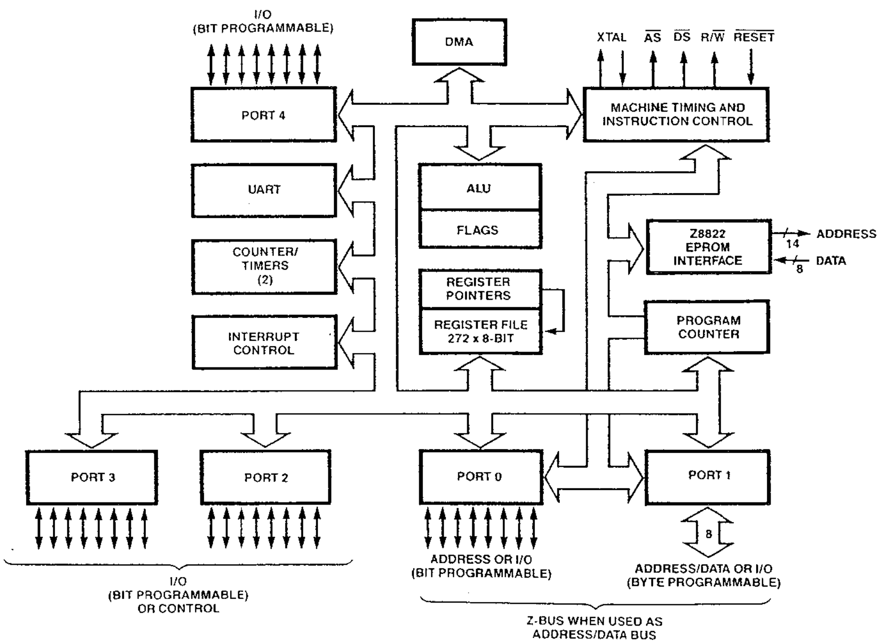


Figure 5. Functional Block Diagram

ARCHITECTURE

The Super8 architecture includes 325 byte-wide internal registers. 272 of these are available for general purpose use; the remaining 53 provide control and mode functions.

The instruction set is specially designed to deal with this large register set. It includes a full complement of 8-bit arithmetic and logical operations, including multiply and divide instructions and provisions for BCD operations. Addresses and counters can be incremented and decremented as 16-bit quantities. Rotate, shift, and bit manipulation instructions are provided. Three new instructions support threaded-code languages.

The UART is a full-function multipurpose asynchronous serial channel with many premium features.

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 5).

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 $P0_0$-$P0_4$ are configured as address out, while pins $P0_5$-$P0_7$ 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 0020_H.

ROMless. (input, active High). This input controls the operation mode of a 68-pin Super8. When connected to V_{CC}, the part will function as a ROMless Z8800. When connected to GND, the part will function as a Z8820 ROM part.

R/$\overline{\text{W}}$. *Read/Write* (output). R/$\overline{\text{W}}$ determines the direction of data transfer for external memory transactions. It is Low when writing to program memory or data memory, and High for everything else.

XTAL1, XTAL2. (Crystal oscillator input.) These pins connect a parallel resonant crystal or an external clock source to the on-board clock oscillator and buffer.

REGISTERS

The Super8 contains a 256-byte internal register space. However, by using the upper 64 bytes of the register space more than once, a total of 325 registers are available.

Registers from 00 to BF are used only once. They can be accessed by any register command. Register addresses C0 to FF contain two separate sets of 64 registers. One set, called control registers, can only be accessed by register direct commands. The other set can only be addressed by register indirect, indexed, stack, and DMA commands.

The uppermost 32 register direct registers (E0 to FF) are further divided into two banks (0 and 1), selected by the Bank Select bit in the Flag register. When a Register Direct command accesses a register between E0 and FF, it looks at the Bank Select bit in the Flag register to select one of the banks.

The register space is shown in Figure 6.

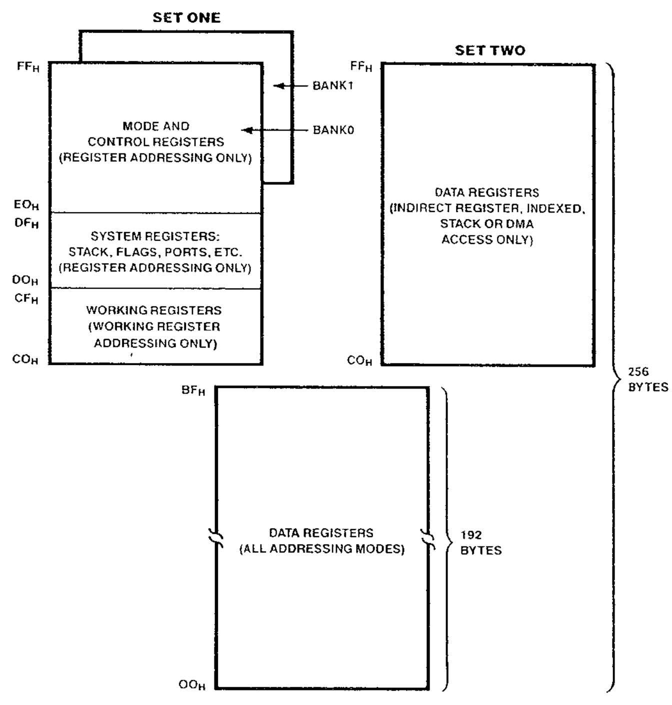


Figure 6. Super8 Registers

Working Register Window

Control registers R214 and R215 are the register pointers, RP0 and RP1. They each define a moveable, 8-register section of the register space. The registers within these spaces are called working registers.

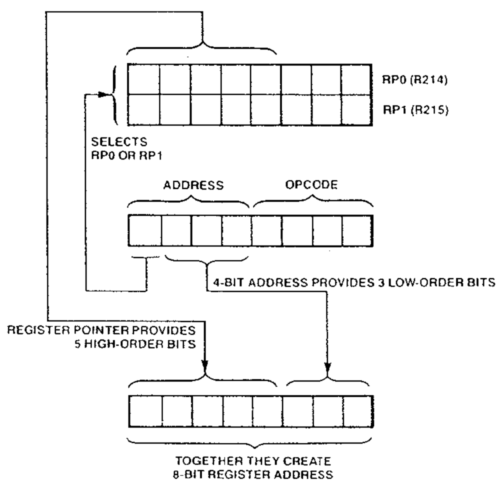
Working registers can be accessed using short 4-bit addresses. The process, shown in section a of Figure 4, works as follows:

- The high-order bit of the 4-bit address selects one of the two register pointers (0 selects RP0; 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.

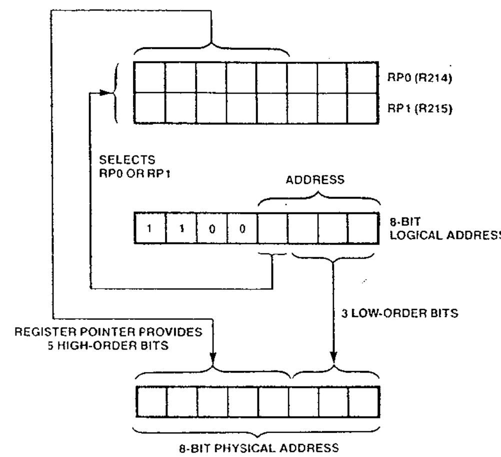
The net effect is to concatenate the five bits from the register pointer to the three bits from the address to form an 8-bit address. As long as the address in the register pointer remains unchanged, the three bits from the address will always point to an address within the same eight registers.

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 (C0 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 7.



a. 4-Bit Addressing



b. 8-Bit Addressing

Figure 7. 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, RP0 points to R192 and RP1 points to R200.

Register List

Table 1 lists the Super8 registers. For more details, see Figure 8.

Table 1. Super-8 Registers

Address Decimal	Hexadecimal		Mnemonic	Function
General-Purpose Registers				
000-192	00-BF		—	General purpose (all address modes)
192-207	C0-CF		—	Working register (direct only)
192-255	C0-FF		—	General purpose (indirect only)
Mode and Control Registers				
208	D0		P0	Port 0 I/O bits
209	D1		P1	Port 1 (I/O only)
210	D2		P2	Port 2
211	D3		P3	Port 3
212	D4		P4	Port 4
213	D5		FLAGS	System Flags Register
214	D6		RP0	Register Pointer 0
215	D7		RP1	Register Pointer 1
216	D8		SPH	Stack Pointer High Byte
217	D9		SPL	Stack Pointer Low Byte
218	DA		IPH	Instruction Pointer High Byte
219	DB		IPL	Instruction Pointer Low Byte
220	DC		IRQ	Interrupt Request
221	DD		IMR	Interrupt Mask Register
222	DE		SYM	System Mode
224	E0	Bank 0	C0CT	CTR 0 Control
		Bank 1	C0M	CTR 0 Mode
225	E1	Bank 0	C1CT	CTR 1 Control
		Bank 1	C1M	CTR 1 Mode
226	E2	Bank 0	C0CH	CTR 0 Capture Register, bits 8-15
		Bank 1	CTCH	CTR 0 Timer Constant, bits 8-15
227	E3	Bank 0	C0CL	CTR 0 Capture Register, bits 0-7
		Bank 1	CTCL	CTR 0 Time Constant, bits 0-7
228	E4	Bank 0	C1CH	CTR 1 Capture Register, bits 8-15
		Bank 1	C1TCH	CTR 1 Time Constant, bits 8-15
229	E5	Bank 0	C1CL	CTR 1 Capture Register, bits 0-7
		Bank 1	C1TCL	CTR 1 Time Constant, bits 0-7
235	EB	Bank 0	UTC	UART Transmit Control
236	EC	Bank 0	URC	UART Receive Control
237	ED	Bank 0	UIE	UART Interrupt Enable
239	EF	Bank 0	UIO	UART Data
240	F0	Bank 0	P0M	Port 0 Mode
		Bank 1	DCH	DMA Count, bits 8-15
241	F1	Bank 0	PM	Port Mode Register
		Bank 1	DCL	DMA Count, bits 0-7
244	F4	Bank 0	H0C	Handshake Channel 0 Control
245	F5	Bank 0	H1C	Handshake Channel 1 Control
246	F6	Bank 0	P4D	Port 4 Direction
247	F7	Bank 0	P4OD	Port 4 Open Drain
248	F8	Bank 0	P2AM	Port 2/3 A Mode
		Bank 1	UBGH	UART Baud Rate Generator, bits 8-15

Table 1. **Super-8 Registers** (Continued)

Address				
Decimal	**Hexadecimal**		**Mnemonic**	**Function**
Mode and Control Registers (Continued)				
249	F9	Bank 0	P2BM	Port 2/3 B Mode
		Bank 1	UBGL	UART Baud Rate Generator, bits 0-7
250	FA	Bank 0	P2CM	Port 2/3 C Mode
		Bank 1	UMA	UART Mode A
251	FB	Bank 0	P2DM	Port 2/3 D Mode
		Bank 1	UMB	UART Mode B
252	FC	Bank 0	P2AIP	Port 2/3 A Interrupt Pending
253	FD	Bank 0	P2BIP	Port 2/3 B Interrupt Pending
254	FE	Bank 0	EMT	External Memory Timing
		Bank 1	WUMCH	Wakeup Match Register
255	FF	Bank 0	IPR	Interrupt Priority Register
		Bank 1	WUMSK	Wakeup Mask Register

MODE AND CONTROL REGISTERS

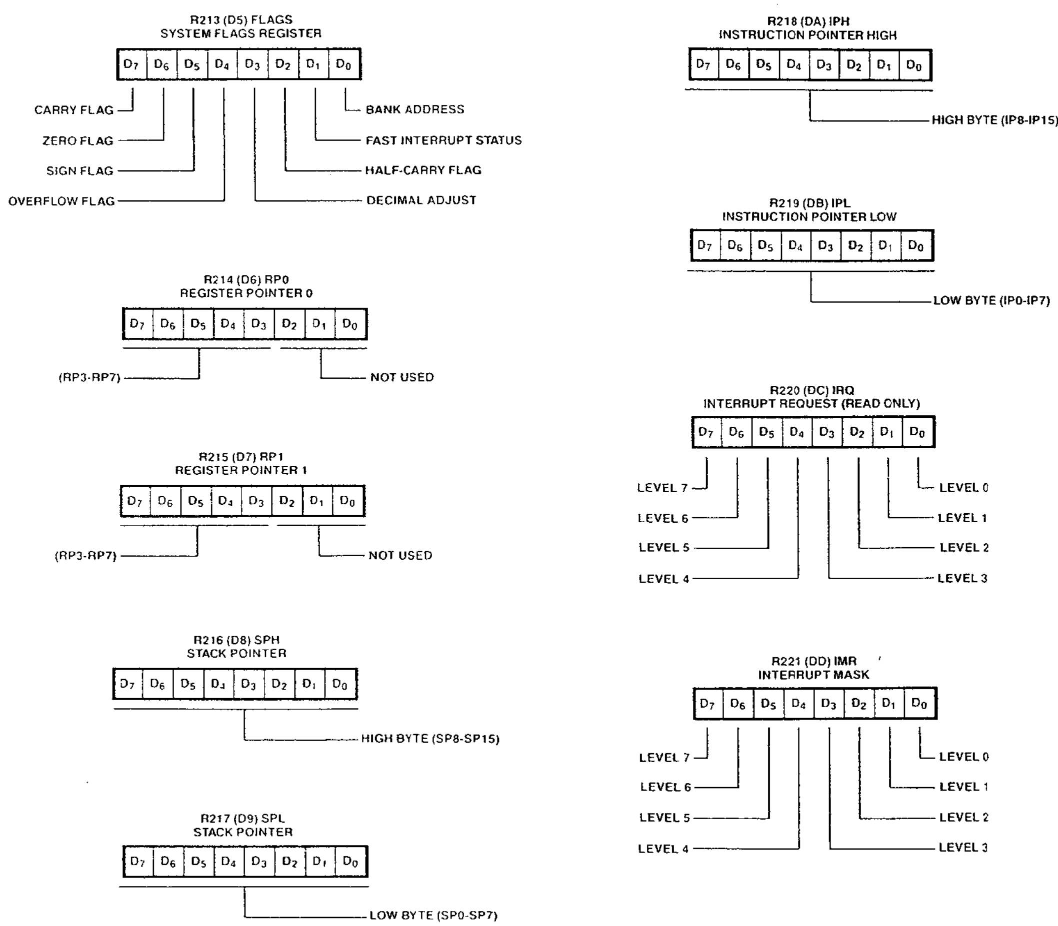
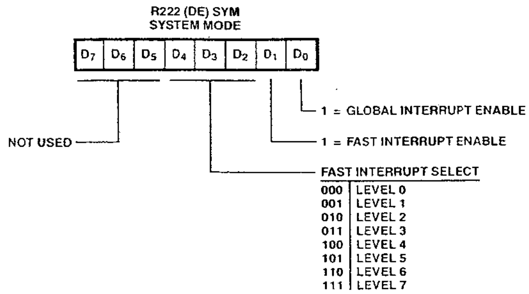
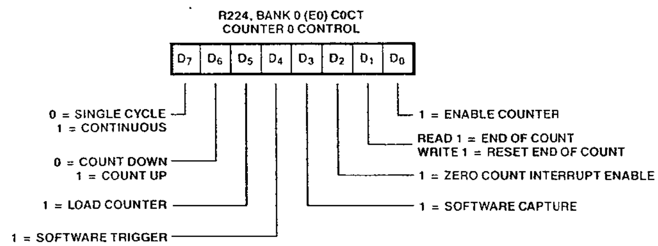
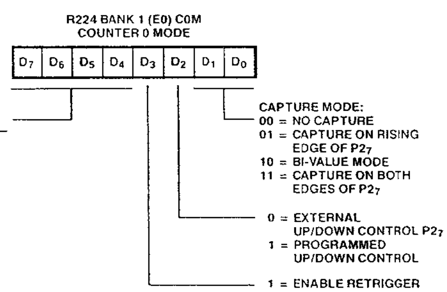


Figure 8. Mode and Control Registers







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 INPUT
0	1	0	1	TRIGGER	C0 INPUT
0	1	1	0	GATE	C0 INPUT
0	1	1	1	GATE/ TRIGGER	C0 INPUT
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 INPUT
1	1	0	1	UNDEFINED	
1	1	1	0	UNDEFINED	
1	1	1	1	CASCADE COUNTERS	

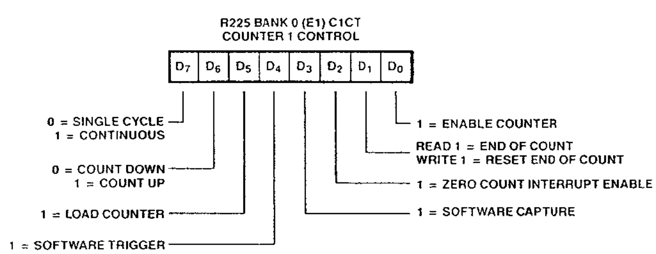


Figure 8. Mode and Control Registers (Continued)

R225 BANK 1 (E1) C1M
COUNTER 1 MODE

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

$D_1 D_0$ — CAPTURE MODE:
00 = NO CAPTURE
01 = CAPTURE ON RISING EDGE OF $P3_7$
10 = BI-VALUE MODE
11 = CAPTURE ON BOTH EDGES OF $P3_7$

D_2 — 0 = EXTERNAL UP/DOWN CONTROL $P3_7$
1 = PROGRAMMED UP/DOWN CONTROL

D_3 — 1 = ENABLE RETRIGGER

D_7–D_4 — INPUT PIN ASSIGNMENTS:

D_7	D_6	D_5	D_4	$P3_7$	$P3_6$
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 INPUT
0	1	0	1	TRIGGER	C0 INPUT
0	1	1	0	GATE	C0 INPUT
0	1	1	1	GATE/ TRIGGER	C0 INPUT
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 INPUT
1	1	0	1	UNDEFINED	
1	1	1	0	UNDEFINED	
1	1	1	1	UNDEFINED	

R226 BANK 0 (E2) C0CH
COUNTER 0 CAPTURE

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

HIGH BYTE ($C0C_8$-$C0C_{15}$)

R229 BANK 0 (E5) C1CL
COUNTER 1 CAPTURE

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

LOW BYTE ($C1C_0$-$C1C_7$)

R226 BANK 1 (E2) C0TCH
COUNTER 0 TIME CONSTANT

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

HIGH BYTE ($C0TC_8$-$C0TC_{15}$)

R229 BANK 1 (E5) C1TCL
COUNTER 1 TIME CONSTANT

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

LOW BYTE ($C1TC_0$-$C1TC_7$)

R227 BANK 0 (E3) C0CL
COUNTER 0 CAPTURE

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

LOW BYTE ($C0C_0$-$C0C_7$)

R235 BANK 0 (EB) UTC
UART TRANSMIT CONTROL

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

D_7 — TRANSMIT DATA SELECT:
0 = OUTPUT $P3_1$ DATA
1 = OUTPUT TRANSMIT DATA

D_6 — 1 = SEND BREAK

D_5 — STOP BITS:
0 = 1 STOP BIT
1 = 2 STOP BITS

D_4 — 1 = WAKE-UP ENABLE

D_3 — 1 = TRANSMIT ENABLE

D_2 — 1 = ZERO COUNT

D_1 — 1 = TRANSMIT BUFFER EMPTY

D_0 — 1 = TRANSMIT DMA ENABLE

R227 BANK 1 (E3) C0TCL
COUNTER 0 TIME CONSTANT

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

LOW BYTE ($C0TC_0$-$C0TC_7$)

R228 BANK 0 (E4) C1CH
COUNTER 1 CAPTURE

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

HIGH BYTE ($C1C_8$-$C1C_{15}$)

R236 BANK 0 (EC) URC
UART RECEIVE CONTROL

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

D_7 — 1 = WAKE-UP DETECT

D_6 — 1 = CONTROL CHARACTER DETECT

D_5 — 1 = BREAK DETECT

D_4 — 1 = FRAMING ERROR

D_3 — 1 = OVERRUN ERROR

D_2 — 1 = PARITY ERROR

D_1 — 1 = RECEIVE ENABLE

D_0 — 1 = RECEIVE CHARACTER AVAILABLE

R228 BANK 1 (E4) C1TCH
COUNTER 1 TIME CONSTANT

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

HIGH BYTE ($C1TC_8$-$C1TC_{15}$)

Figure 8. Mode and Control Registers (Continued)

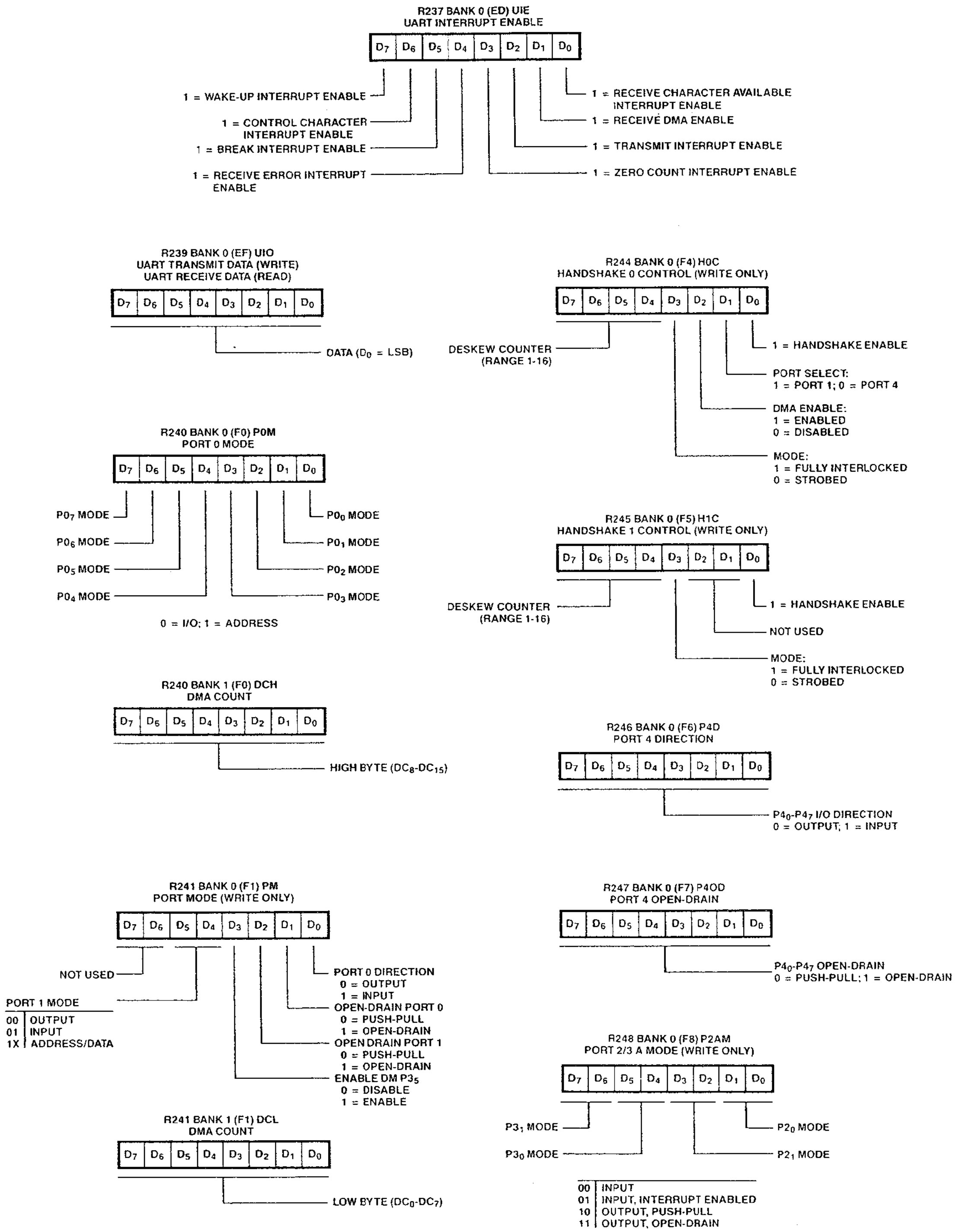


Figure 8. Mode and Control Registers (Continued)

MODE AND CONTROL REGISTERS (Continued)

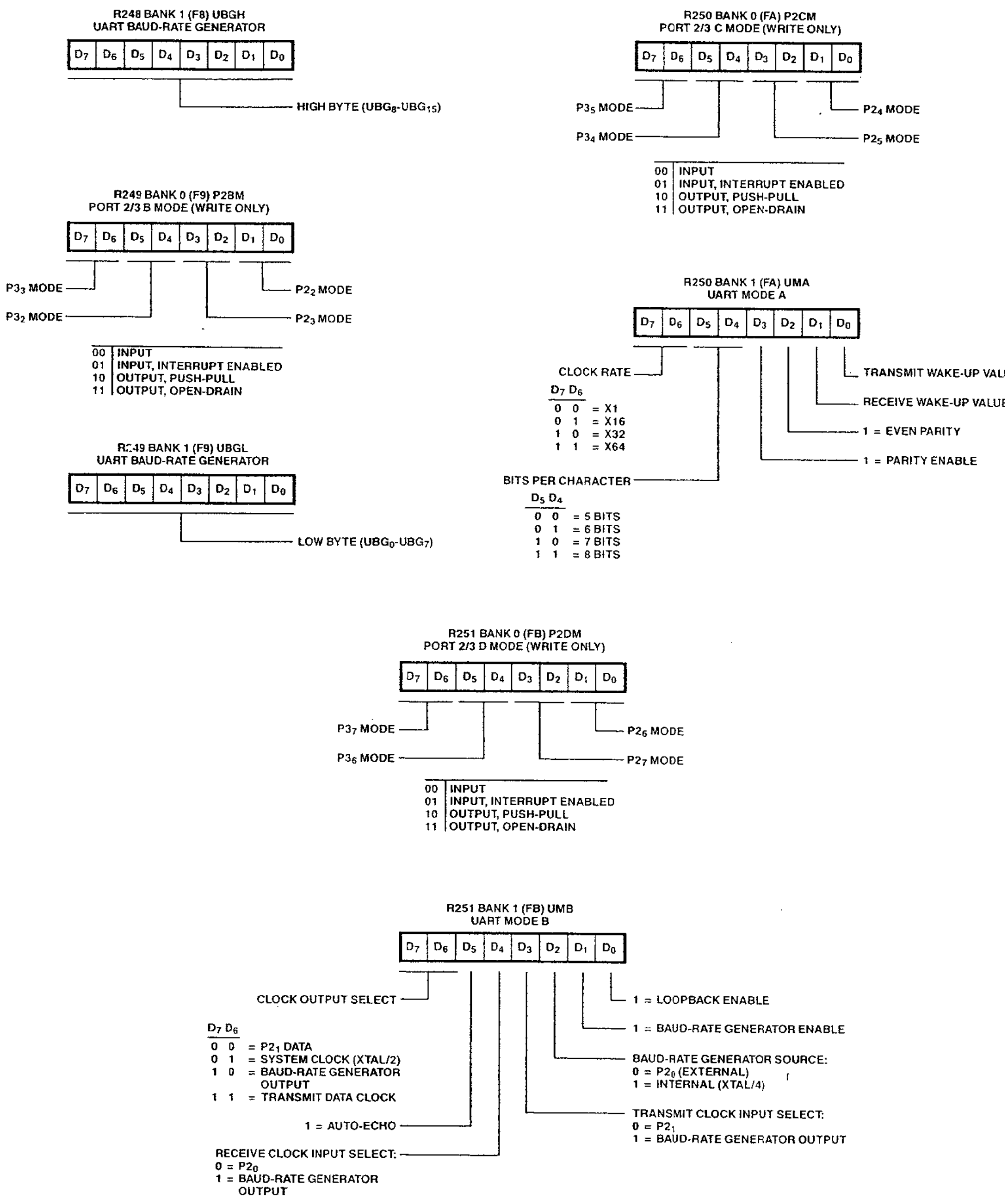


Figure 8. Mode and Control Registers (Continued)

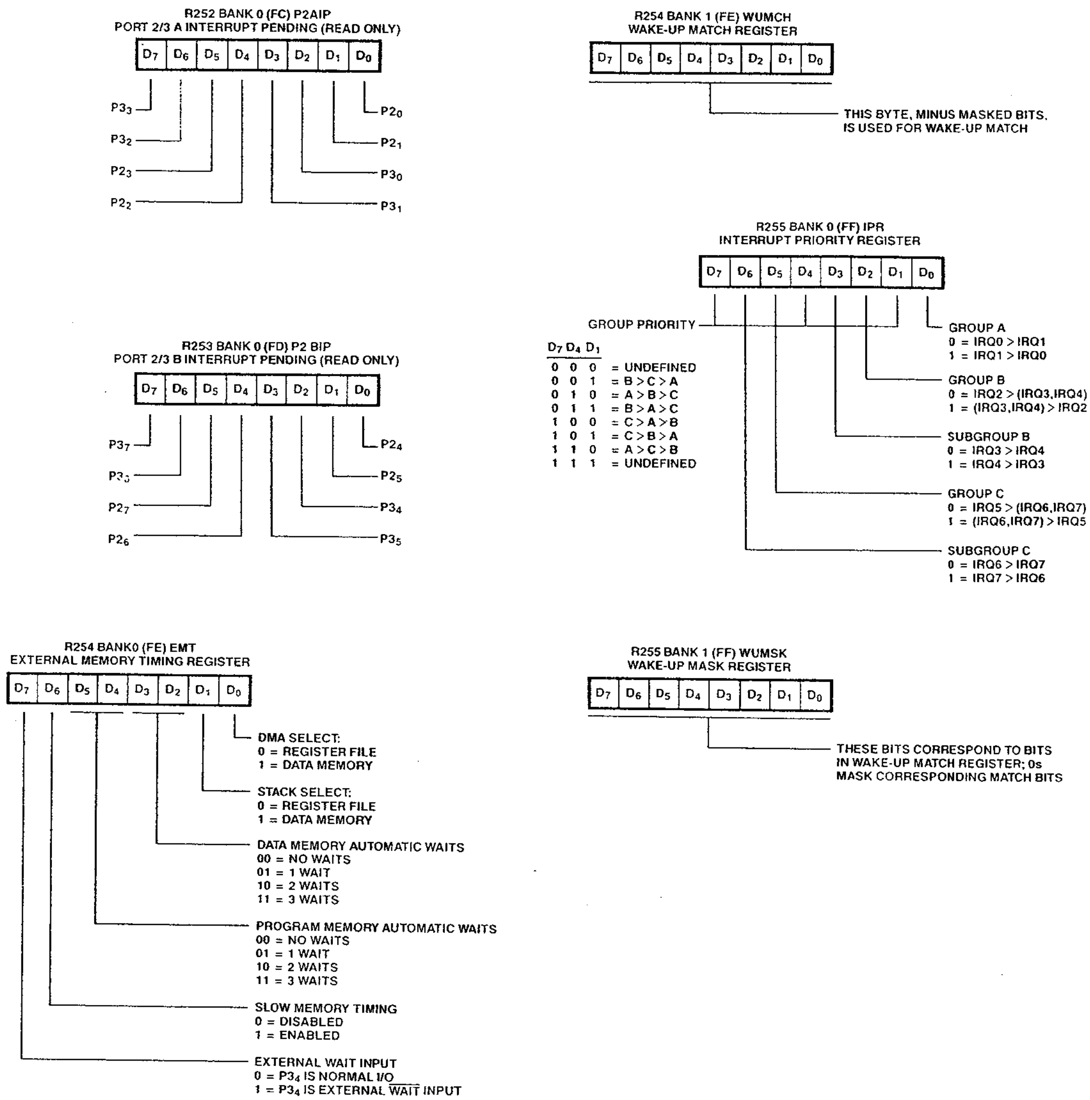


Figure 8. Mode and Control Registers (Continued)

I/O PORTS

The Super8 has 40 I/O lines arranged into five 8-bit ports. These lines are all TTL-compatible, and can be configured as inputs or outputs. Some can also be configured as address/data lines.

Each port has an input register, an output register, and a register address. Data coming into the port is stored in the input register, and data to be written to a port is stored in the output register. Reading a port's register address returns the value in the input register; writing a port's register address loads the value in the output register. If the port is configured for an output, this value will appear on the external pins.

When the CPU reads the bits configured as outputs, the data on the external pins is returned. Under normal output loading, this has the same effect as reading the output register, unless the bits are configured as open-drain outputs.

The ports can be configured as shown in Table 2.

Table 2. Port Configuration

Port	Configuration Choices
0	Address outputs and/or general I/O
1	Multiplexed address/data(or I/O, only for ROM and Protopack)
2 and 3	Control I/O for UART, handshake channels, and counter/timers; also general I/O and external interrupts
4	General I/O

Port 0

Port 0 can be configured as an I/O port or an output for addressing external memory, or it can be divided and used as both. The bits configured as I/O can be either all outputs or all inputs; they cannot be mixed. If configured for outputs, they can be push-pull or open-drain type.

Any bits configured for I/O can be accessed via R208. To write to the port, specify R208 as the destination (dst) of an instruction; to read the port, specify R208 as the source (src).

Port 0 bits configured as I/O can be placed under handshake control of handshake channel 1.

Port 0 bits configured as address outputs cannot be accessed via the register.

In ROMless devices, initially the four lower bits are configured as address eight through twelve.

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 3. 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/$\overline{WAIT}$
5	Handshake 0 output	5	Handshake 1 output/$\overline{DM}$
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 1X 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 controlled by the $\overline{DM}$ line (Port $P3_5$), which selects data memory when Low and program memory when High.

Figure 9 shows the system memory space.

CPU Program Memory

Program memory occupies addresses 0 to 64K. External program memory, if present, is accessed by configuring Ports 0 and 1 as a memory interface.

The address/data lines are controlled by $\overline{AS}$, $\overline{DS}$ and R/$\overline{W}$.

The first 32 program memory bytes are reserved for interrupt vectors; the lowest address available for user programs is 32 (decimal). This value is automatically loaded into the program counter after a hardware reset.

ROMless

Port 0 can be configured to provide from 0 to 8 additional address lines. Port 1 is always used as an 8-bit multiplexed address/data port.

ROM and Protopack

Port 1 is configured as multiplexed address/data or as I/O. When Port 1 is configured as address/data, Port 0 lines can be used as additional address lines, up to address 15. External program memory is mapped above internal program memory; that is, external program memory can occupy any space beginning at the top of the internal ROM space up to the 64K (16-bit address) limit.

CPU Data Memory

The external CPU data memory space, if separated from program memory by the $\overline{DM}$ optional output, can be mapped anywhere from 0 to 64K (full 16-bit address space). Data memory uses the same address/data bus (Port 1) and additional addresses (chosen from Port 0) as program memory. Data memory is distinguished from program memory by the DM pin ($P3_5$), and by the fact that data memory can begin at address 0000_H. This feature differs from the Z8.

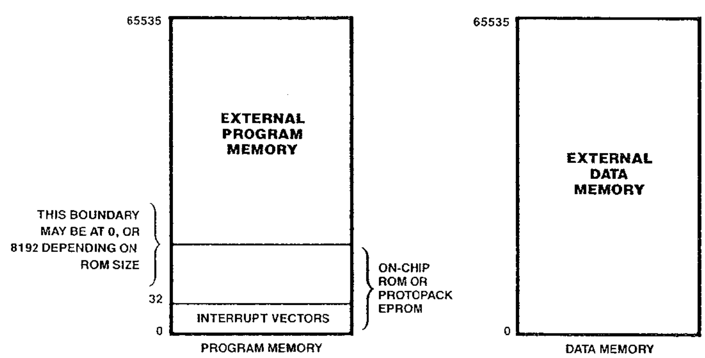


Figure 9. Program and Data Memory Address Spaces

INSTRUCTION SET

The Super8 instruction set is designed to handle its large register set. The instruction set provides a full complement of 8-bit arithmetic and logical operations, including multiply and divide. It supports BCD operations using a decimal adjustment of binary values, and it supports incrementing and decrementing 16-bit quantities for addresses and counters.

It provides extensive bit manipulation, and rotate and shift operations, and it requires no special I/O instructions—the I/O ports are mapped into the register file.

Instruction Pointer

A special register called the Instruction Pointer (IP) provides hardware support for threaded-code languages. It consists of register-pair R218 and R219, and it contains memory addresses. The MSB is R218.

Threaded-code languages deal with an imaginary higher-level machine within the existing hardware machine. The IP acts like the PC for that machine. The command NEXT passes control to or from the hardware machine to the imaginary machine, and the commands ENTER and EXIT are imaginary machine equivalents of (real machine) CALLS and RETURNS.

If the commands NEXT, ENTER, and EXIT are not used, the IP can be used by the fast interrupt processing, as described in the Interrupts section.

Flag Register

The Flag register (FLAGS) contains eight bits that describe the current status of the Super8. Four of these can be tested and used with conditional jump instructions; two others are used for BCD arithmetic. FLAGS also contains the Bank Address bit and the Fast Interrupt Status bit.

The flag bits can be set and reset by instructions.

> **CAUTION**
>
> Do not specify FLAGS as the destination of an instruction that normally affects the flag bits or the result will be unspecified.

The following paragraphs describe each flag bit:

Bank Address. This bit is used to select one of the register banks (0 or 1) between (decimal) addresses 224 and 255. It is cleared by the SB0 instruction and set by the SB1 instruction.

Fast Interrupt Status. This bit is set during a fast interrupt cycle and reset during the IRET following interrupt servicing. When set, this bit inhibits all interrupts and causes the fast interrupt return to be executed when the IRET instruction is fetched.

Half-Carry. This bit is set to 1 whenever an addition generates a carry out of bit 3, or when a subtraction borrows out of bit 4. This bit is used by the Decimal Adjust (DA) instruction to convert the binary result of a previous addition or subtraction into the correct decimal (BCD) result. This flag, and the Decimal Adjust flag, are not usually accessed by users.

Decimal Adjust. This bit is used to specify what type of instruction was executed last during BCD operations, so a subsequent Decimal Adjust operation can function correctly. This bit is not usually accessible to programmers, and cannot be used as a test condition.

Overflow Flag. This flag is set to 1 when the result of a twos-complement operation was greater than 127 or less than -128. It is also cleared to 0 during logical operations.

Sign Flag. Following arithmetic, logical, rotate, or shift operations, this bit identifies the state of the MSB of the result. A 0 indicates a positive number and a 1 indicates a negative number.

Zero Flag. For arithmetic and logical operations, this flag is set to 1 if the result of the operation is zero.

For operations that test bits in a register, the zero bit is set to 1 if the result is zero.

For rotate and shift operations, this bit is set to 1 if the result is zero.

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 Table 4.

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 4. Condition Codes and Meanings

Binary	Mnemonic	Flags	Meaning
0000	F	—	Always false
1000	—	—	Always true
0111*	C	C = 1	Carry
1111*	NC	C = 0	No carry
0110*	Z	Z = 1	Zero
1110*	NZ	Z = 0	Not zero
1101	PL	S = 0	Plus
0101	MI	S = 1	Minus
0100	OV	V = 1	Overflow
1100	NOV	V = 0	No overflow
0110*	EQ	Z = 1	Equal
1110*	NE	Z = 0	Not equal
1001	GE	(S XOR V) = 0	Greater than or equal
0001	LT	(S XOR V) = 1	Less than
1010	GT	(Z OR (S XOR V)) = 0	Greater than
0010	LE	(Z OR (S XOR V)) = 1	Less than or equal
1111*	UGE	C = 0	Unsigned greater than or equal
0111*	ULT	C = 1	Unsigned less than
1011	UGT	(C = 0 AND Z = 0) = 1	Unsigned greater than
0011	ULE	(C OR Z) = 1	Unsigned less than or equal

NOTE: Asterisks (*) indicate condition codes that relate to two different mnemonics but test the same flags. For example, Z and EQ are both True if the Zero flag is set, but after an ADD instruction, Z would probably be used, while after a CP instruction, EQ would probably be used.

Registers can be addressed by an 8-bit address in the range of 0 to 255. Working registers can also be addressed using 4-bit addresses, where five bits contained in a register pointer (R218 or R219) are concatenated with three bits from the 4-bit address to form an 8-bit address.

Registers can be used in pairs to generate 16-bit program or data memory addresses.

Notation and Encoding

The instruction set notations are described in Table 5.

Functional Summary of Commands

Figure 10 shows the formats followed by a quick reference guide to the commands.

Table 5. Instruction Set Notations

Notation	Meaning	Notation	Meaning
cc	Condition code (see Table 4)	DA	Direct address (between 0 and 65535)
r	Working régister (between 0 and 15)	RA	Relative address
rb	Bit of working register	IM	Immediate
r0	Bit 0 of working register	IML	Immediate long
R	Register or working register	dst	Destination operand
RR	Register pair or working register pair (Register pairs always start on an even-number boundary)	src	Source operand
		@	Indirect address prefix
IA	Indirect address	SP	Stack pointer
Ir	Indirect working register	PC	Program counter
IR	Indirect register or indirect working register	IP	Instruction pointer
Irr	Indirect working register pair	FLAGS	Flags register
IRR	Indirect register pair or indirect working register pair	RP	Register pointer
X	Indexed	#	Immediate operand prefix
XS	Indexed, short offset	%	Hexadecimal number prefix
XL	Indexed, long offset	OPC	Opcode

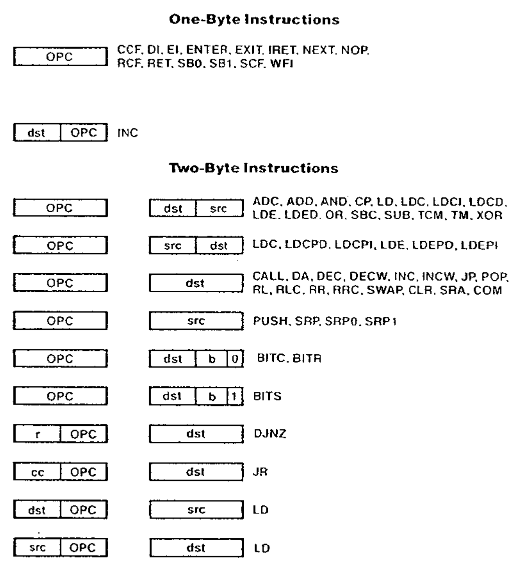


Figure 10. Instruction Formats

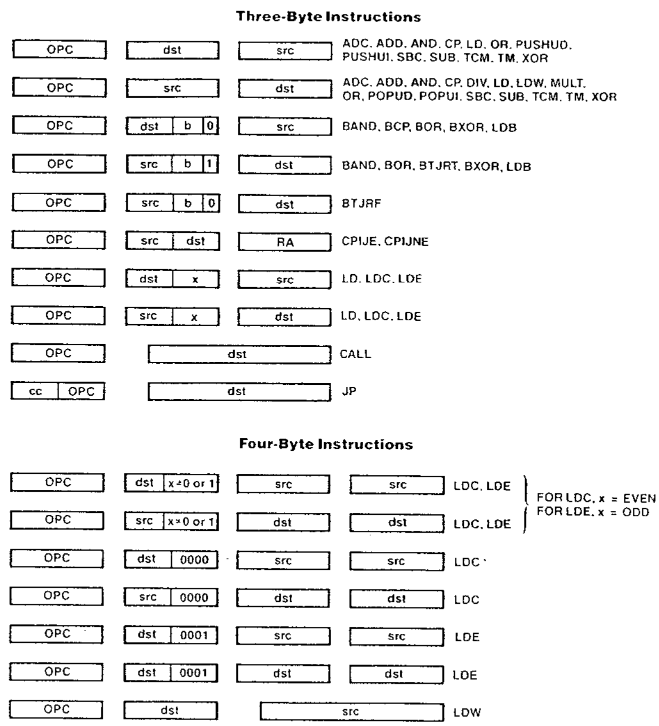


Figure 10. Instruction Formats (Continued)

INSTRUCTION SUMMARY

Instruction and Operation	Addr Mode dst	Addr Mode src	Opcode Byte (Hex)	C	Z	S	V	D	H
ADC dst,src dst ← dst + src + C	(Note 1)		1☐	*	*	*	—	0	*
ADD dst,src dst ← dst + src	(Note 1)		0☐	*	*	*	*	0	*
AND dst,src dst ← dst AND src	(Note 1)		5☐	—	*	*	0	—	—
BAND dst,src dst ← dst AND src	r0 Rb	Rb r0	67 67	—	*	0	U	—	—
BCP dst, src dst – src	r0	Rb	17	—	*	0	U	—	—
BITC dst dst ← NOT dst	rb		57	—	*	0	U	—	—
BITR dst dst ← 0	rb		77	—	—	—	—	—	—
BITS dst dst ← 1	rb		77	—	—	—	—	—	—

Instruction and Operation	Addr Mode dst	Addr Mode src	Opcode Byte (Hex)	C	Z	S	V	D	H
BOR dst, src dst ← dst OR src	r0 Rb	rB r0	07	—	*	0	U	—	—
BTJRF if src = 0, PC = PC + dst .	RA	rb	37	—	—	—	—	—	—
BTJRT if src = 1, PC = PC + dst	RA	rb	37	—	—	—	—	—	—
BXOR dst, src dst ← dst XOR src	r0 Rb	Rb r0	27 27	—	*	0	U	—	—
CALL dst SP ← SP – 2 @SP ← PC PC ← dst	DA IRR IA		F6 F4 D4	—	—	—	—	—	—
CCF C = NOT C			EF	*	—	—	—	—	—
CLR dst dst ← 0	R IR		B0 B1	—	—	—	—	—	—

INSTRUCTION SUMMARY (Continued)

Instruction and Operation	Addr Mode dst	Addr Mode src	Opcode Byte (Hex)	Flags Affected C Z S V D H
COM dst dst ← NOT dst	R IR		60 61	— * * 0 — —
CP dst,src dst – src	(Note 1)		A☐	* * * * — —
CPIJE if dst – src = 0,then PC ← PC + RA Ir ← Ir + 1	r	Ir	C2	— — — — — —
CPIJNE if dst – src = 0,then PC ← PC + RA Ir ← Ir + 1	r	Ir	D2	— — — — — —
DA dst dst ← DA dst	R IR		40 41	* * * U — —
DEC dst dst ← dst – 1	R IR		00 01	— * * * — —
DECW dst dst ← dst – 1	RR IR		80 81	— * * * — —
DI SMR (0) ← 0			8F	— — — — — —
DIV dst, src dst ÷ src dst (Upper) ← Quotient dst (Lower) ← Remainder	RR RR RR	R IR IM	94 95 96	* * * * — —
DJNZ r,dst r ← r – 1 if r = 0 PC ← PC + dst	RA	r	rA (r = 0 to F)	— — — — — —
EI SMR (0) ← 1			9F	— — — — — —
ENTER SP ← SP – 2 @ SP ← IP IP ← PC PC ← @ IP IP ← IP + 2			1F	— — — — — —
EXIT IP ← @SP SP ← SP + 2 PC ← @IP IP ← IP + 2			2F	— — — — — —
INC dst dst ← dst + 1	r R IR		rE (r = 0 to F) 20 21	— * * * — —

Instruction and Operation	Addr Mode dst	Addr Mode src	Opcode Byte (Hex)	Flags Affected C Z S V D H
INCW dst dst ← 1 + dst	RR IR		A0 A1	— * * * — —
IRET (Fast) PC ↔ IP FLAG ← FLAG' FIS ← 0			BF	Restored to before interrupt
IRET (Normal) FLAGS ← @SP; SP ← SP + 1 PC ← @SP; SP ← SP + 2; SMR (0) ← 1			BF	Restored to before interrupt
JP cc,dst if cc is true, PC ← dst	DA IRR		ccD (cc = 0 to F) 30	— — — — — —
JR cc,dst if cc is true, PC ← PC + d	RA		ccB (cc = 0 to F)	— — — — — —
LD dst,src dst ← src	r r R r IR R R R IR IR r x	IM R r IR r R IR IM IM R x r	rC r8 r9 (r = 0 to F) C7 D7 E4 E5 E6 D6 F5 87 97	— — — — — —
LDB dst, src dst ← src	r0 Rb	Rb r0	47 47	— — — — — —
LDC/LDE dst ← src	r Irr r xs r x1 r DA	Irr r xs r x1 r DA r	C3 D3 E7 F7 A7 B7 A7 B7	— — — — — —
LDCD/LDED dst, src dst ← src rr ← rr – 1	r	Irr	E2	— — — — — —
LDEI/LDCI dst, src dst ← src rr ← rr + 1	r	Irr	E3	— — — — — —
LDCPD/LDEPD dst,src rr ← rr – 1 dst ← src	Irr	r	F2	— — — — — —

Instruction and Operation	Addr Mode dst	Addr Mode src	Opcode Byte (Hex)	Flags Affected C Z S V D H
LDCPI/LDEPI dst, src rr ← rr + 1 dst ← src	Irr	r	F3	— — — — — —
LDW dst, src dst ← src	RR RR RR	RR IR IMM	C4 C5 C6	— — — — — —
MULT dst, src	RR RR RR	R IR IM	84 85 86	* 0 * * — —
NEXT PC ← @IP IP ← IP + 2			0F	— — — — — —
NOP			FF	— — — — — —
OR dst,src dst ← dst OR src	(Note 1)		4□	— * * 0 — —
POP dst dst ← @SP; SP ← SP + 1		R IR	50 51	— — — — — —
POPUD dst, src dst ← src IR ← IR − 1	R	IR	92	— — — — — —
POPUI dst, src dst ← src IR ← IR + 1	R	IR	93	— — — — — —
PUSH src SP ← SP − 1; @SP ← src		R IR	70 71	— — — — — —
PUSHUD dst, src IR ← IR − 1 dst ← src	IR	R	82	— — — — — —
PUSHUI dst, src IR ← IR + 1 dst ← src	IR	R	83	— — — — — —
RCF C ← 0			CF	0 — — — — —
RET PC ← @SP; SP ← SP + 2			AF	— — — — — —
RL dst C ← dst (7) dst (0) ← dst (7) dst (N + 1) ← dst (N) N = 0 to 6	R IR		90 91	* * * * — —

Instruction and Operation	Addr Mode dst	Addr Mode src	Opcode Byte (Hex)	Flags Affected C Z S V D H
RLC dst dst (0) ← C C ← dst (7) dst (N + 1) ← dst (N) N = 0 to 6	R IR		10 11	* * * * — —
RR dst C ← dst (0) dst (7) ← dst (0) dst (N) ← dst (N + 1) N = 0 to 6	R IR		E0 E1	* * * * — —
RRC dst C ← dst (0) dst (7) ← C dst (N) ← dst (N + 1) N = 0 to 6	R IR		C0 C1	* * * * — —
SB0 BANK ← 0			4F	— — — — — —
SB1 BANK ← 1			5F	— — — — — —
SBC dst,src dst ← dst − src − C	(Note 1)		3□	* * * * 1 *
SCF C ← 1			DF	1 — — — — —
SRA dst dst (7) ← dst (7) C ← dst (0) dst (N) ← dst (N + 1) N = 0 to 6	R IR		D0 D1	* * * 0 — —
SRP src RP0 ← IM RP1 ← IM + 8		IM	31	— — — — — —
SRP0 RP0 ← IM		IM	31	— — — — — —
SRP1 RP1 ← IM		IM	31	— — — — — —
SUB dst,src dst ← dst − src	(Note 1)		2□	* * * * 1 *

Instruction and Operation	Addr Mode dst	Addr Mode src	Opcode Byte (Hex)	Flags Affected C	Z	S	V	D	H
SWAP dst dst (0-3) ↔ dst (4-7)	R IR		F0 F1	—	*	*	U	—	—
TCM dst,src (NOT dst) AND src	(Note 1)		6□	—	*	*	0	—	—
TM dst,src dst AND src	(Note 1)		7□	—	*	*	0	—	—
WFI			3F	—	—	—	—	—	—
XOR dst,src dst ← dst XOR src	(Note 1)		B□	—	*	*	0	—	—

NOTE 1: These instructions have an identical set of addressing modes, which are encoded for brevity. The first opcode nibble identifies the command, and is found in the table above. The second nibble, represented by a □, defines the addressing mode as shown in Table 6.:

Table 6. Second Nibble

Addr Mode dst	Addr Mode src	Lower Opcode Nibble
r	r	2
r	Ir	3
R	R	4
R	IR	5
R	IM	6

For example, to use an opcode represented as x□ with an "RR" addressing mode, use the opcode "x4."

0 = Cleared to Zero
1 = Set to One
— = Unaffected
* = Set or reset, depending on result of operation.
U = Undefined

SUPER-8 OPCODE MAP

Lower Nibble (Hex) across; Upper Nibble (Hex) down.

	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,IM	12/10 JP cc,DA	6 INC r1	14 NEXT
1	6 RLC R_1	6 RLC IR_1	6 ADC r_1,r_2	6 ADC r_1,Ir_2	10 ADC 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-R_b	↓	↓	↓	↓	↓	↓	↓	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	8 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-R_b	↓	↓	↓	↓	↓	↓	↓	
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 RR_1	10 DECW IR_1	10 PUSHUD IR_1,R_2	10 PUSHUI IR_1,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	10 POPUD IR_2,R_1	10 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 R_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 Ir_1,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	10 LD R_2,IR_1	18 CALL DA_1	18 LDC* r_2,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_1,xL	20 LDC* r_2,DA_1

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

***Examples:**
BOR r_0-R_2
is BOR r_1,b,R_2
or BOR r_2,b,R_1
LDC r_1,Irr_2
is LDC r_1,Irr_2 = program
or LDE r_1,Irr_2 = data

Sequence:
Opcode, first, second, third operands

NOTE: The blank areas are not defined.

Figure 11. Opcode Map

Table 7. 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
DECW	dst	Decrement word
DIV	dst, src	Divide
INC	dst	Increment
INCW	dst	Increment word
MULT	dst, src	Multiply
SBC	dst, src	Subtract with carry
SUB	dst, src	Subtract
Logical Instructions		
AND	dst, src	Logical AND
COM	dst	Complement
OR	dst, src	Logical OR
XOR	dst, src	Logical exclusive
Program Control Instructions		
BTJRT	dst, src	Bit test jump relative on True
BTJRF	dst, src	Bit test jump relative on False
CALL	dst	Call procedure
CPIJE	dst, src	Compare, increment and jump on equal
CPIJNE	dst, src	Compare, increment and jump on non-equal
DJNZ	r, dst	Decrement and jump on non-zero
ENTER		Enter
EXIT		Exit
IRET		Return from interrupt
JP	cc, dst	Jump on condition code
JP	dst	Jump unconditional
JR	cc, dst	Jump relative on condition code
JR	dst	Jump relative unconditional
NEXT		Next
RET		Return
WFI		Wait for interrupt
Bit Manipulation Instructions		
BAND	dst, src	Bit AND
BCP	dst, src	Bit compare
BITC	dst	Bit complement
BITR	dst	Bit reset
BITS	dst	Bit set
BOR	dst, src	Bit OR
BXOR	dst, src	Bit exclusive OR
TCM	dst, src	Test complement under mask
TM	dst, src	Test under mask
Rotate and Shift Instructions		
RL	dst	Rotate left
RLC	dst	Rotate left through carry
RR	dst	Rotate right
RRC	dst	Rotate right through carry
SRA	dst	Shift right arithmetic
SWAP	dst	Swap nibbles
CPU Control Instructions		
CCF		Complement carry flag
DI		Disable interrupts
EI		Enable interrupts
NOP		Do nothing
RCF		Reset carry flag
SB0		Set bank 0
SB1		Set bank 1
SCF		Set carry flag
SRP	src	Set register pointers
SRP0	src	Set register pointer zero
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 10 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.
External interrupts are falling-edge triggered.

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 8 for bit details).

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

The levels are shown in Figure 12.

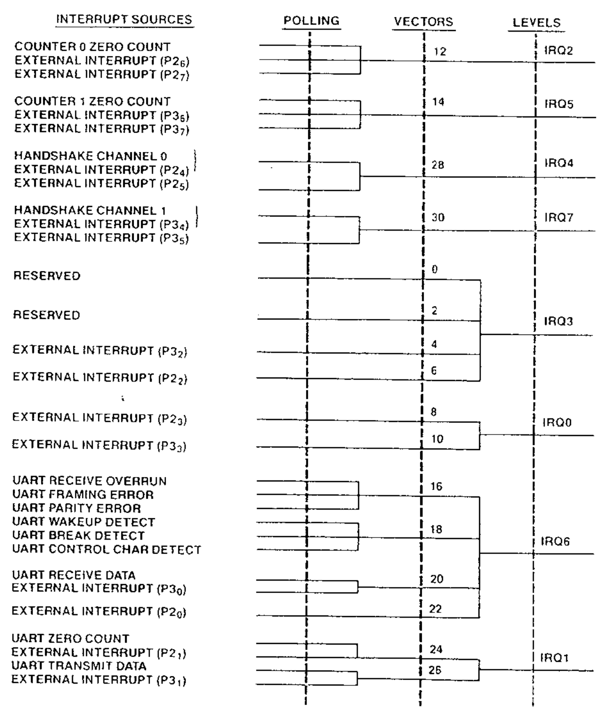


Figure 12. Interrupt Levels and Vectors

Enables

Interrupts can be enabled or disabled as follows:

- Interrupt enable/disable. The entire interrupt structure can be enabled or disabled by setting bit 0 in the System Mode register (R222).
- Level enable. Each level can be enabled or disabled by setting the appropriate bit in the Interrupt Mask register (R221).
- Level priority. The priority of each level can be controlled by the values in the Interrupt Priority register (R255, Bank 0).
- Source enable/disable. Each interrupt source can be enabled or disabled in the sources' Mode and Control register.

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 will be 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 addresses to high addresses in the register file, use PUSHUI and POPUD. For a stack that grows from high addresses to low addresses in data memory, use LDEI for pop and LDEPD for push.

COUNTER/TIMERS

The Super8 has two identical independently programmable 16-bit counter/timers that can be cascaded to produce a single 32-bit counter. They can be used to count external events, or they can obtain their input internally. The internal input is obtained by dividing the crystal frequency by four.

The counter/timers can be set to count up or down, by software or external events. They can be set for single or continuous cycle counting, and they can be set with a bi-value option, where two preset time constants alternate in loading the counter each time it reaches zero. This can be used to produce an output pulse train with a variable duty cycle.

The counter/timers can also be programmed to capture the count value at an external event or generate an interrupt whenever the count reaches zero. They can be turned on and off in response to external events by using a gate and/or a trigger option. The gate option enables counts only when the gate line is Low; the trigger option turns on the counter after a transient High. The gate and trigger options used together cause the counter/timer to work in gate mode after initially being triggered.

The control and status register bits for the counter/timers are shown in Figure 5.

DMA

The Super8 features an on-chip Direct Memory Access (DMA) channel to provide high bandwidth data transmission capabilities. The DMA channel can be used by the UART receiver, UART transmitter, or handshake channel 0. Data can be transferred between the peripheral and contiguous locations in either the register file or external data memory. A 16-bit count register determines the number of transactions to be performed; an interrupt can be generated when the count is exhausted. DMA transfers to or from the register file require six CPU clock cycles; DMA transfers to or from external memory take ten CPU clock cycles, excluding wait states.

ABSOLUTE MAXIMUM RATINGS

Voltage on all pins with respect
to ground . . . . $-0.3V$ to $+7.0V$
Ambient Operating
Temperature . . . . .See Ordering Information
Storage Temperature . . . . $-65°C$ to $+150°C$

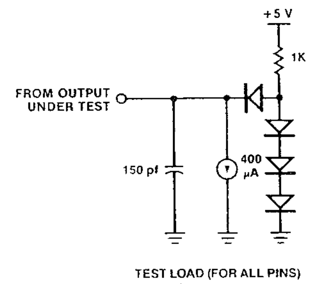
Stresses greater than these may cause permanent damage to the device. This is a stress rating only; operation of the device under conditions more severe than those listed for operating conditions may cause permanent damage to the device. Exposure to absolute maximum ratings for extended periods may also cause permanent damage.

STANDARD TEST CONDITIONS

Figure 14 shows the setup for standard test conditions. All voltages are referenced to ground, and positive current flows into the reference pin.

Standard conditions are:

- $+4.75V \leq V_{CC} \leq +5.25V$
- GND = 0V
- $0°C \leq T_A \leq +70°C$

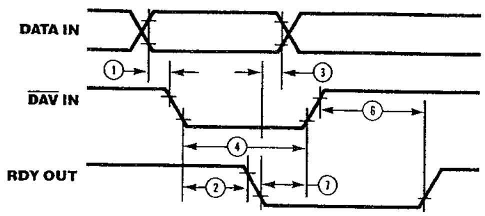


Standard Test Load

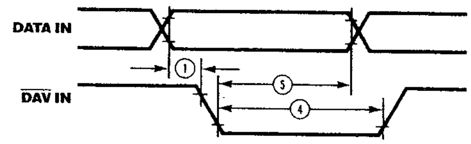
DC CHARACTERISTICS

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	$I_{OH} = -400\,\mu A$
V_{OL}	Output Low Voltage		0.4	V	$I_{CL} = +4.0\,mA$
I_{IL}	Input Leakage	-10	10	μA	
I_{OL}	Output Leakage	-10	10	μA	
I_{IR}	Reset Input Current		-50	μA	
I_{CC}	V_{CC} Supply Current		320	mA	

INPUT HANDSHAKE TIMING



Fully Interlocked Mode



Strobed Mode

AC CHARACTERISTICS (20 MHz)

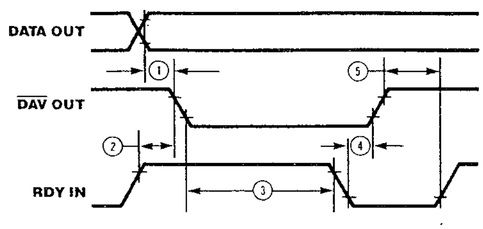
Input Handshake

Number	Symbol	Parameter	Min	Max	Notes*‡
1	TsDI(DAV)	Data In to Setup Time	0		
2	TdDAVIf(RDY)	$\overline{DAV}$ ↓ Input to RDY ↓ Delay		200	1
3	ThDI(RDY)	Data In Hold Time from RDY ↓	0		
4	TwDAV	$\overline{DAV}$ In Width	45		
5	ThDI(DAV)	Data In Hold Time from $\overline{DAV}$ ↓	130		
6	TdDAV(RDY)	$\overline{DAV}$ ↑ Input to RDY ↑ Delay		100	2
7	TdRDYf(DAV)	RDY ↓ Output to $\overline{DAV}$ ↑ Delay	0		

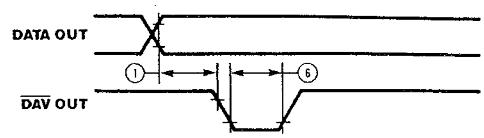
NOTES:
1. Standard Test Load
2. This time assumes user program reads data before $\overline{DAV}$ Input goes high. RDY will not go high before data is read.

‡Times given are in ns.
*Times are preliminary and subject to change.

OUTPUT HANDSHAKE TIMING



Fully Interlocked Mode



Strobed Mode

AC CHARACTERISTICS (12 MHz, 20 MHz)

Output Handshake

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

NOTES:
1. Standard Test Load
2. 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.

‡Times given are in ns.
*Times are preliminary and subject to change.

AC CHARACTERISTICS (12 MHz)

Read/Write

Number	Symbol	Parameter	Normal Timing Min	Normal Timing Max	Extended Timing Min	Extended Timing Max	Notes‡*
1	TdA(AS)	Address Valid to $\overline{\text{AS}}$ ↑ Delay	35		115		
2	TdAS(A)	$\overline{\text{AS}}$ ↑ to Address Float Delay	65		150		
3	TdAS(DR)	$\overline{\text{AS}}$ ↑ to Read Data Required Valid		270		600	1
4	TwAS	$\overline{\text{AS}}$ Low Width	65		150		
5	TdA(DS)	Address Float to $\overline{\text{DS}}$ ↓	20		20		
6a	TwDS(Read)	$\overline{\text{DS}}$ (Read) Low Width	225		470		1
6b	TwDS(Write)	$\overline{\text{DS}}$ (Write) Low Width	130		295		1
7	TdDS(DR)	$\overline{\text{DS}}$ ↓ to Read Data Required Valid		180		420	1
8	ThDS(DR)	Read Data to $\overline{\text{DS}}$ ↑ Hold Time	0		0		
9	TdDS(A)	$\overline{\text{DS}}$ ↑ to Address Active Delay	50		135		
10	TdDS(AS)	$\overline{\text{DS}}$ ↑ to $\overline{\text{AS}}$ ↓ Delay	60		145		
11	TdDO(DS)	Write Data Valid to $\overline{\text{DS}}$ (Write) ↓ Delay	35		115		
12	TdAS(W)	$\overline{\text{AS}}$ ↑ to Wait Delay		220		600	2
13	ThDS(W)	$\overline{\text{DS}}$ ↑ to Wait Hold Time	0		0		
14	TdRW(AS)	R/$\overline{\text{W}}$ Valid to $\overline{\text{AS}}$ ↑ Delay	50		135		

NOTES:
1. WAIT states add 167 ns to these times.
2. Auto-wait states add 167 ns to this time.

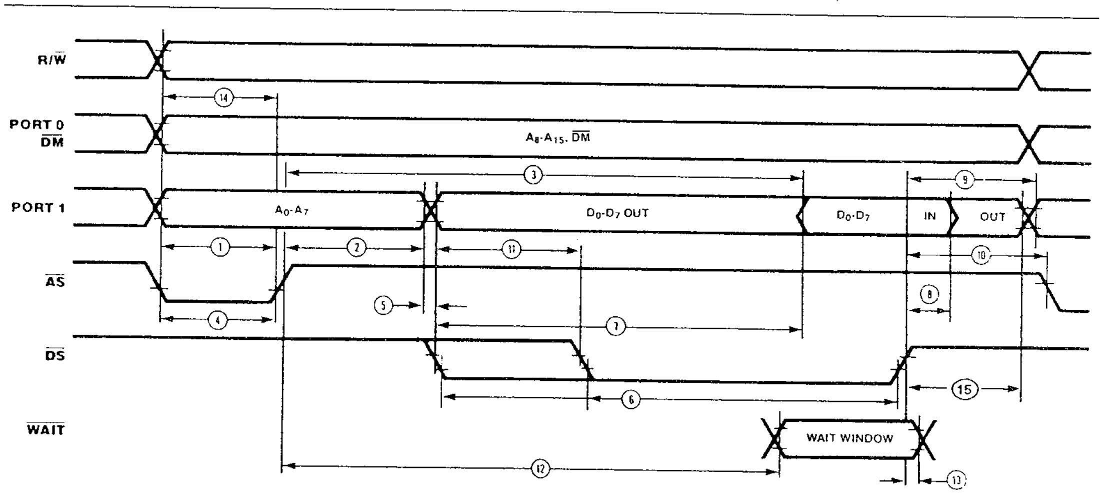
‡ All times are in ns and are for 12 MHz input frequency.
* Timings are preliminary and subject to change.

AC CHARACTERISTICS (20 MHz)

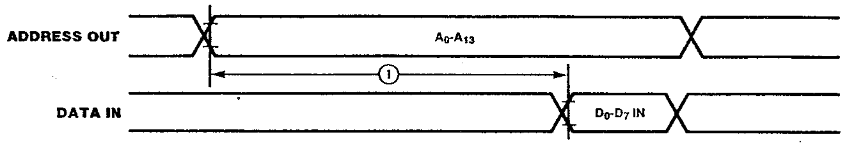
Read/Write

Number	Symbol	Parameter	Normal Timing Min	Normal Timing Max	Extended Timing Min	Extended Timing Max	Notes‡*
1	TdA(AS)	Address Valid to $\overline{AS}$ ↑ Delay	10		50		
2	TdAS(A)	$\overline{AS}$ ↑ to Address Float Delay	35		85		
3	TdAS(DR)	$\overline{AS}$ ↑ to Read Data Required Valid		140		335	1
4	TwAS	$\overline{AS}$ Low Width	35		85		
5	TdA(DS)	Address Float to $\overline{DS}$ ↓	0		0		
6a	TwDS(Read)	$\overline{DS}$ (Read) Low Width	125		275		1
6b	TwDS(Write)	$\overline{DS}$ (Write) Low Width	65		165		1
7	TdDS(DR)	$\overline{DS}$ ↓ to Read Data Required Valid		80		225	1
8	ThDS(DR)	Read Data to $\overline{DS}$ ↑ Hold Time	0		0		
9	TdDS(A)	$\overline{DS}$ ↑ to Address Active Delay	20		70		
10	TdDS(AS)	$\overline{DS}$ ↑ to $\overline{AS}$ ↓ Delay	30		80		
11	TdDO(DS)	Write Data Valid to $\overline{DS}$ (Write) ↓ Delay	10		50		
12	TdAS(W)	$\overline{AS}$ ↑ to Wait Delay		90		335	2
13	ThDS(W)	$\overline{DS}$ ↑ to Wait Hold Time	0		0		
14	TdRW(AS)	R/$\overline{W}$ Valid to $\overline{AS}$ ↑ Delay	20		70		
15	TdDS(DW)	$\overline{DS}$ ↑ to Write Data Not Valid Delay	20		70		

NOTES:
1. WAIT states add 100 ns to these times.
2. Auto-wait states add 100 ns to this time.
‡ All times are in ns and are for 20 MHz input frequency.
* Timings are preliminary and subject to change.



External Memory Read and Write Timing



EPROM Read Timing

AC CHARACTERISTICS (20 MHz)

EPROM Read Cycle

Number	Symbol	Parameter	Min	Max	Notes‡*
1	TdA(DR)	Address Valid to Read Data Required Valid		170	1

NOTES:
1. WAIT states add 167 ns to these times.
‡All times are in ns and are for 12 MHz input frequency.
*Timings are preliminary and subject to change.