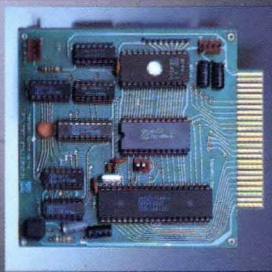
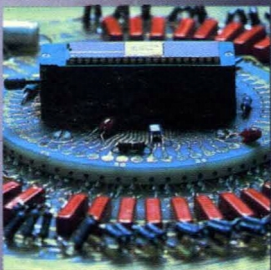
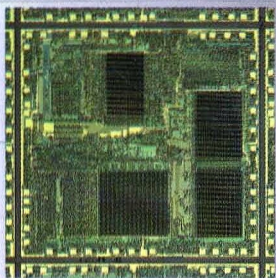


# DATABOOK

## Z8 MICROCOMPUTER FAMILY

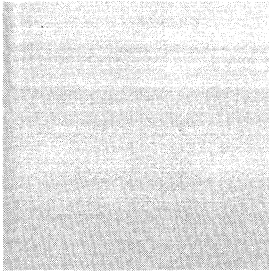
3<sup>rd</sup> EDITION



Technology  
and Service

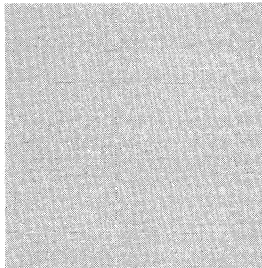
# Z8 MICROCOMPUTER FAMILY

3<sup>rd</sup> EDITION  
MAY 1986

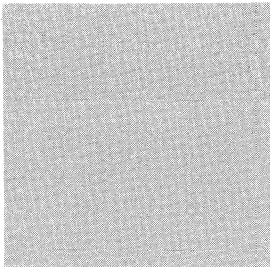


*The Z8 2K ROM single chip  
microcomputer produced  
by SGS using NMOS technology*

*Automatic electrical  
test of a VLSI device  
in the SGS Agrate facility*



*Application board  
using Z8671 tiny  
Basic microcomputer*





# Contents

	<b>Page</b>
<b>SGS: an introduction</b> .....	3
<b>Overview</b> .....	7
Cross Reference .....	9
Part Number Identification .....	9
<b>Datasheets</b> .....	13
<i>NMOS Family</i>	
— Z8601/L .....	13
— Z8611/L .....	35
— Z8621/L .....	57
— Z8671 .....	79
— Z8681/L .....	103
— Z8682/L .....	103
— Z8684/L .....	103
— Z86E11 .....	129
— Z86E21 .....	155
<b>Packages</b> .....	183
<b>Reliability Informations</b> .....	187
<b>Development Products</b> .....	199
— General Purpose Fundation Module for MCU Emulation .....	199
— Z8 Emulation and Development Package .....	203
— Z8 Emulation and Development Package for IBM compatible Personal Computers .....	207
<b>Technical and Application Notes</b> .....	213
— Double Layer P-Vapox and $Si_3N_4$ Glass Passivation .....	213
— PMZ8: Z8681 in Single Board Computer Application .....	215
— Single Board Computer Using Z8671 .....	219
— Z8 in Electronic Private Automated Branch Exchange (EPABX 2 Ext./8 Int. Lines) .....	227
— Using Z8 MCU in Keyboard Controller .....	237
— Z8 MCU in Dynamic Keyboard .....	241
— Comparison of Z8611, 8051 and MC6801 Microcomputer .....	251
— A Programmer's Guide to the Z8 Microcomputer .....	265



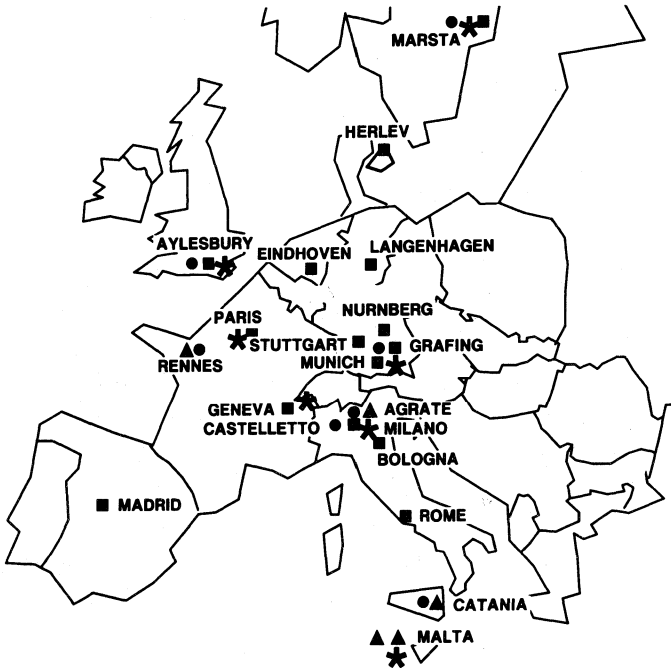
# Identity

Late in 1957, SGS was founded around a team of researchers who were already carrying out pioneer work in the field of semiconductors. From that small nucleus, the company has evolved into a Group of Companies, operating on a worldwide basis as a broad range semiconductor producer, with billings over 300 million dollars and employing over 9500 people.

The SGS Group of Companies has now reached a total of 11 subsidiaries, located in Brazil, France, Germany, Italy, Malta, Malaysia, Singapore, Sweden, Switzerland, United Kingdom and the USA.

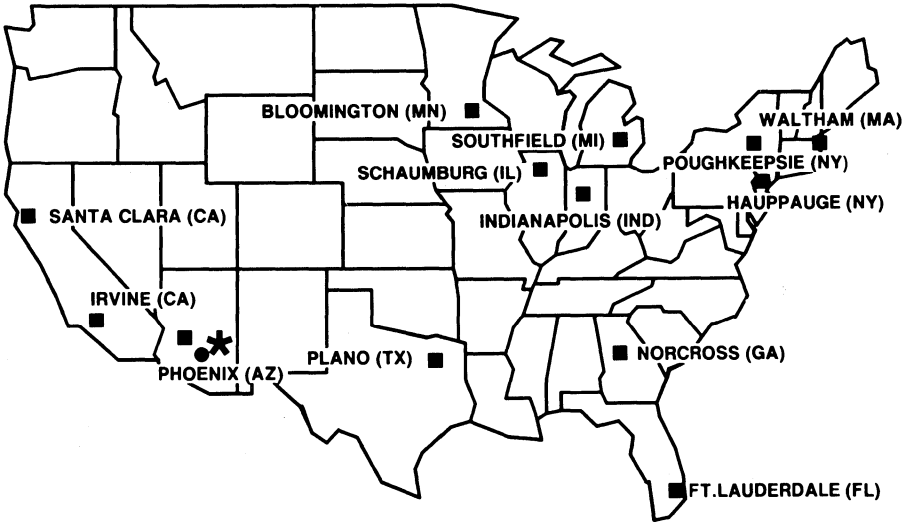
To go with its logo, the company takes the motto "Technology and Service", underlining the accent given to the development of state-of-the-art technologies and the corporate commitment to offer customers the best quality and service in the industry.

# SGS Locations - Europe



- \* HEADQUARTERS
- ▲ FACTORIES
- SALES OFFICES
- DESIGN CENTERS

# SGS Locations - North America



- \* HEADQUARTERS
- SALES OFFICES
- DESIGN CENTERS



# SGS Locations - Asia/Pacific



- \* HEADQUARTERS
- ▲ FACTORIES
- SALES OFFICES
- DESIGN CENTERS

# Overview

## Z8\* Family

The Z8 microcomputer family offers the most sophisticated processing capability available on a single chip. As an extension of earlier generations of microcomputers, the Z8 family provides standard on-chip functions, such as:

- 2K, 4K or 8K bytes of ROM
- 144 or 256 8-bit registers
- 32 lines of programmable I/O
- Clock oscillator

In addition, the Z8 Family offers advanced on-chip features, including:

- Two counter/timers
- Six vectored interrupts
- UART for serial I/O communication
- Stack functions
- Power-down option
- TTL compatibility

The Z8 microcomputer family is expandable off-chip to provide an additional 62K bytes of program memory and 62K bytes of data memory for the 2K-byte ROM version, an additional 60K bytes of program memory and 60K bytes of data memory for the 4K-byte ROM version and an additional 56K bytes of program memory and 56K bytes of data memory for the 8K-byte ROM version. The interface to external memory is accomplished through one, one and one-half, or two of the 8-bit I/O ports, depending on the number of address bits required for the external functions. The Z-BUS\* protocol allows easy interface to external functions including peripheral chips.

The Z8 family challenges the "multi-chip solution" design currently implemented by general-purpose microprocessors. Designs based on Z8 family microcomputers offer a minimum chip-count configuration that can easily be expanded to meet requirements for enhancement options and for future improvements.

**Optimized Instruction Set.** The instruction set of the Z8 family is optimized for high-code density and reduced execution time. This feature is supported by a "working register area" concept that uses short (4-bit) register addresses. The general-purpose registers can be used as accumulators, as address pointers for indirect addressing, as index registers, or for implementing an on-chip stack.

The 47 instruction types and six addressing modes—together with the ability to operate on bits, 4-bit BCD digits, 8-bit bytes, and 16-bit words—offer unique programming capability and flexibility.

**Growing Family.** The Z8 microcomputer family is growing to meet the needs of more complex designs. The 8K ROM version, Z8621 completed developed by SGS, offers all the features of the Z8 Family, plus 8K bytes of on-chip ROM and 256-byte register file. The increased ROM and bytes register file allows the designer to take advantage of the code optimization inherent in the Z8 instruction set when using between 2K and 8K bytes at program memory.

The ROMless microcomputers provide an alternative for designers seeking to take advantage of the on-chip features of the Z8681, Z8682 and Z8684 in applications that require external program memory. A Z8681 microcomputer can be used to control a system that addresses up to 128K bytes of on-chip memory, a Z8682 up to 124K bytes and Z8684 up to 120K bytes.

Newly in the Z8 family the 4K and 8K bytes on-chip Eprom Z86E11 and Z86E21, that perform different programming modes, like: Eprom-like, using standard eprom programmer; Self-programming, during normal microcomputer operation and time-efficient self-program facility; and integrated programmable Eprom read-out protection.

For these characteristics the Z86E11 and Z86E21 can be considered as low cost development tools for the Z8 microcomputer family.

Low Power version 80mA current consumption, on all the family will be available.

**Expanded Applications.** The Z8 microcomputer family is finding its way into increasingly sophisticated designs. In addition to the low-end capability applications commonly used with microcomputers, the Z8 family can be used effectively in such applications as:

- Computer peripheral controllers
- Smart terminals
- Dumb terminals
- Telephone switching systems

- Arcade games and intelligent home games
- Process control
- Intelligent instrumentation
- Automotive mechanisms

An example of how a Z8 might be used in the design of an intelligent terminal is shown in Figure 1. The features of such a terminal depend on its specific requirements, but it is clear that the Z8 microcomputers offer unprecedented capability and flexibility to the microcomputer designer.

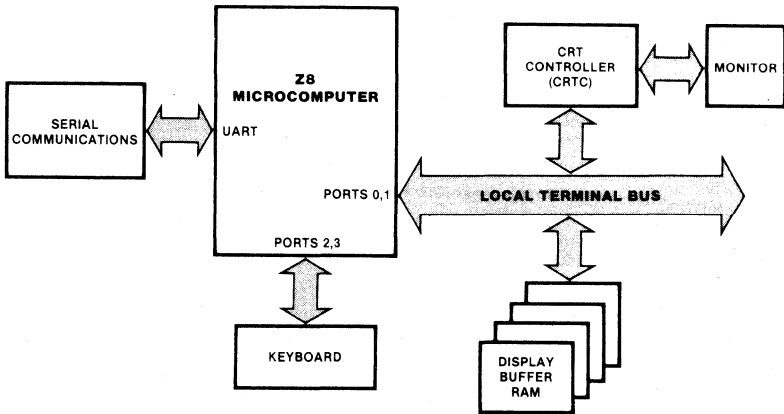


Figure 1. Z8 Based Intelligent Terminal

# Z8 Cross Reference

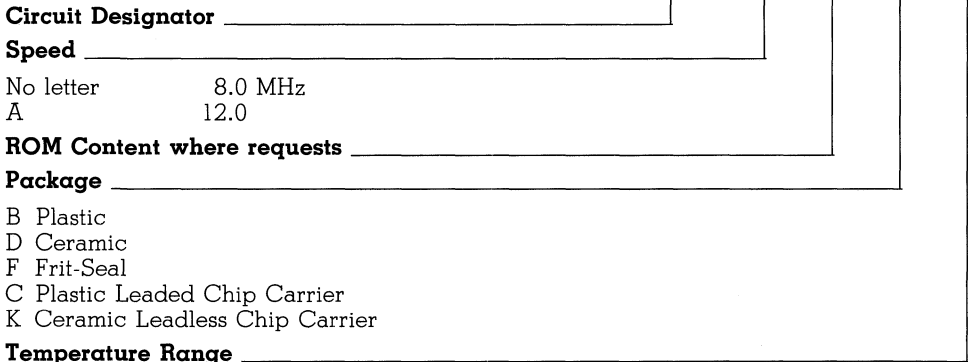
		SUFFIX DESCRIPTION										
		BASE PART NUMBER		SPEED SUFFIX (MHz)		PACKAGE SUFFIX					TEMP. SUFFIX	
				8.0	12.0	PL	FR	CER	PLCC	LCC	0/+70°C	-40/+85°C
<b>SGS</b>	<del>Z86XX</del>	*	A			B	F	D	C	K	1	6
ZILOG	Z86XX	*	A			P	na	C	V	Q	S	E

		DEVICE TYPE								
		2K ROM	4K ROM	8K ROM	TINY BASIC	ROMLESS	4K EPROM	8K EPROM	2K PIGGY-BACK	4K PIGGY-BACK
<b>SGS</b>	Z8601	Z8611	Z8621	Z8671	Z8681/2/4	Z86E11	Z86E21	—	—	—
ZILOG	Z8601	Z8611	—	Z8671	Z8681/2/4	—	—	Z8603RS	Z8613RS	

Notes: \* Standard Version no suffix required  
na: Not Available

## SGS Part Number Identification

example:



1    0 to + 70°C  
6    -40 to + 85°C



---

# **Datasheets**

---

---





# Z8601/L

## Z8 2K ROM Microcomputer

- Complete microcomputer, 2K bytes of ROM, 128 bytes of RAM, 32 I/O lines, and up to 62K bytes addressable external space each for program and data memory.
- 144-byte register file, including 124 general-purpose registers, four I/O port registers, and 16 status and control registers.
- Minimum instruction execution time 1  $\mu$ s, at 12 MHz.
- Vectored, priority interrupts for I/O, counter/timers, and UART.
- Full-duplex UART and two programmable 8-bit counter/timers, each with a 6-bit programmable prescaler.
- Register Pointer so that short, fast instructions can access any of nine working register groups in 1  $\mu$ s.
- On-chip oscillator that accepts crystal or external clock drive.
- Low-power standby option which retains contents of general-purpose registers.
- Single +5 V power supply—all pins TTL-compatible.
- Low Power version (Z8601L):
  - Available 8 MHz
  - Current consumption 80 mA
- Available in 8 and 12 MHz versions.

### General Description

The Z8601 microcomputer introduces a new level of sophistication to single-chip architecture. Compared to earlier single-chip microcomputers, the Z8601 offers faster execution; more efficient use of memory; more sophisticated interrupt, input/output and bit-manipulation capabilities; and easier system expansion.

Under program control, the Z8601 can be

tailored to the needs of its user. It can be configured as a stand-alone microcomputer with 2K bytes of internal ROM, a traditional microprocessor that manages up to 124K bytes of external memory, or a parallel-processing element in a system with other processors and peripheral controllers linked by the Z-BUS. In all configurations, a large number of pins remain available for I/O.

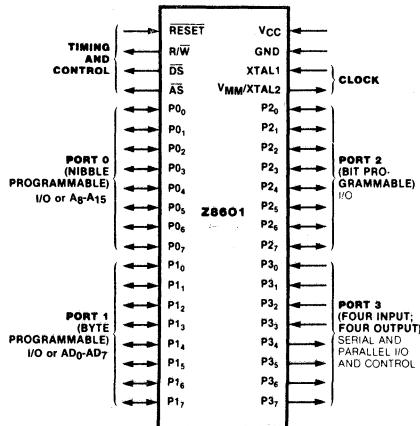


Figure 1. Logic Functions





# Z8601/L

## General Description (Continued)

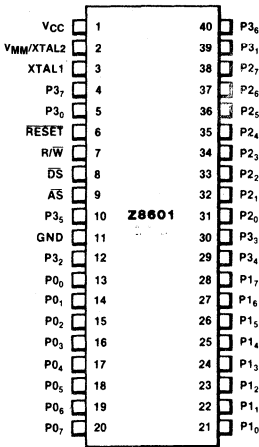
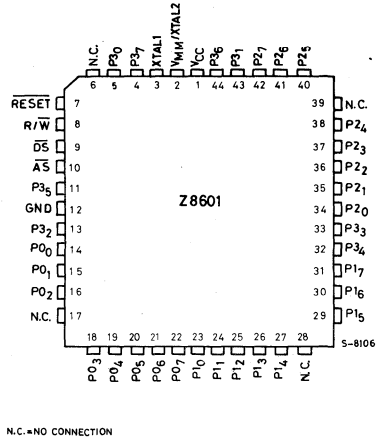


Figure 2. Pin Configuration



N.C. = NO CONNECTION

Figure 2a. Chip Carrier Pin Configuration

## Architecture

Z8601 architecture is characterized by a flexible I/O scheme, an efficient register and address space structure and a number of ancillary features that are helpful in many applications.

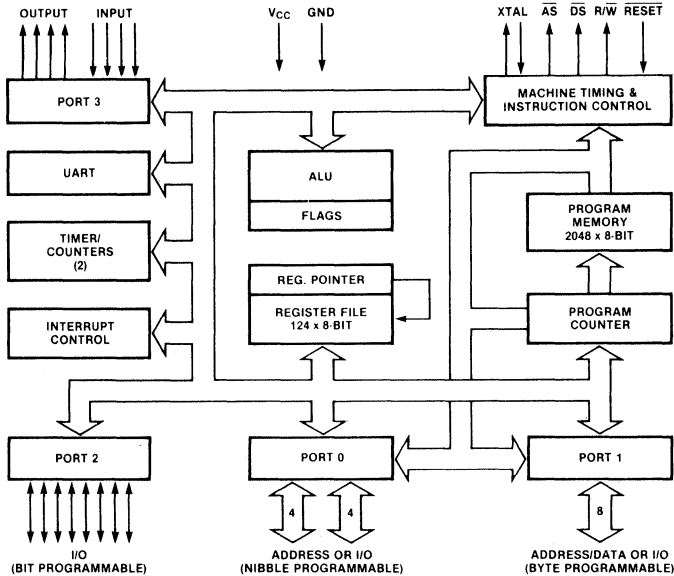
Microcomputer applications demand powerful I/O capabilities. The Z8601 fulfills this with 32 pins dedicated to input and output. These lines are grouped into four ports of eight lines each and are configurable under software control to provide timing, status signals, serial or parallel I/O with or without handshake, and an address/data bus for interfacing external memory.

Because the multiplexed address/data bus is merged with the I/O-oriented ports, the Z8601 can assume many different memory and I/O configurations. These configurations range from a self-contained microcomputer

to a microprocessor that can address 124K bytes of external memory.

Three basic address spaces are available to support this wide range of configurations: program memory (internal and external), data memory (external) and the register file (internal). The 144-byte random-access register file is composed of 124 general-purpose registers, four I/O port registers, and 16 control and status registers.

To unburden the program from coping with real-time problems such as serial data communication and counting/timing, an asynchronous receiver/transmitter (UART) and two counter/timers with a large number of user-selectable modes are offered on-chip. Hardware support for the UART is minimized because one of the on-chip timers supplies the bit rate.

**Architecture (Continued)**

**Figure 3. Block Diagram**
**Pin Description**

**AS.** *Address Strobe* (output, active Low). Address Strobe is pulsed once at the beginning of each machine cycle. Addresses output via Port 1 for all external program or data memory transfers are valid at the trailing edge of AS. Under program control, AS can be placed in the high-impedance state along with Ports 0 and 1, Data Strobe and Read/Write.

**DS.** *Data Strobe* (output, active Low). Data Strobe is activated once for each external memory transfer.

**P0-P7.** *I/O Port Lines* (input/output, TTL compatible). 8 lines Nibble Programmable that can be configured under program control for I/O or external memory interface.

**P10-P17.** *I/O Port Lines* (input/output, TTL compatible). 8 lines Byte Programmable that can be configured under program control for I/O or multiplexed address (A<sub>0</sub>-A<sub>7</sub>) and data (D<sub>0</sub>-D<sub>7</sub>) lines used to interface with program/data memory.

**P20-P27.** *I/O Port Lines* (input/output, TTL compatible). 8 lines Bit Programmable. In addition they can be configured to provide open-drain outputs.

**P30-P34.** *Input Port Lines* (TTL compatible). They can also be configured as control lines.

**P35-P37.** *Output Port Lines* (TTL compatible). They can also be configured as control lines.



---

**Pin Descriptions** (Continued)

**RESET.** *Reset* (input, active Low).  $\overline{\text{RESET}}$  initializes the Z8601. When  $\overline{\text{RESET}}$  is deactivated, program execution begins from internal program location 000CH.

**R/W.** *Read/Write* (output).  $\overline{\text{R/W}}$  is Low when the Z8601 is writing to external program or data memory.

**XTAL1, XTAL2.** *Crystal 1, Crystal 2* (time-base input and output). These pins connect a parallel-resonant crystal (8 or 12 MHz maximum) or an external single-phase clock (8 or 12 MHz maximum) to the on-chip clock oscillator and buffer.

---

**Address Spaces**

**Program Memory.** The 16-bit program counter addresses 64K bytes of program memory space. Program memory can be located in two areas: one internal and the other external (Figure 4). The first 2048 bytes consist of on-chip mask-programmed ROM. At addresses 2048 and greater, the Z8601 executes external program memory fetches.

The first 12 bytes of program memory are reserved for the interrupt vectors. These locations contain six 16-bit vectors that correspond to the six available interrupts.

**Data Memory.** The Z8601 can address 62K bytes of external data memory beginning at locations 2048 (Figure 5). External data memory may be included with or separated from the external program memory space.  $\overline{\text{DM}}$ , an optional I/O function that can be programmed to appear on pin P34, is used to distinguish between data and program memory space.

**Register File.** The 144-byte register file includes four I/O port registers (R0-R3), 124

general-purpose registers (R4-R127) and 16 control and status registers (R240-R255). These registers are assigned the address locations shown in Figure 6.

Z8601 instructions can access registers directly or indirectly with an 8-bit address field. The Z8601 also allows short 4-bit register addressing using the Register Pointer (one of the control registers). In the 4-bit mode, the register file is divided into nine working-register groups, each occupying 16 contiguous locations (Figure 7). The Register Pointer addresses the starting location of the active working-register group.

**Stacks.** Either the internal register file or the external data memory can be used for the stack. A 16-bit Stack Pointer (R254 and R255) is used for the external stack, which can reside anywhere in data memory between locations 2048 and 65535. An 8-bit Stack Pointer (R255) is used for the internal stack that resides within the 124 general-purpose registers (R4-R127).



Address Spaces (Continued)

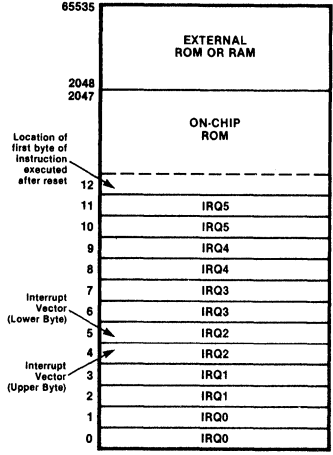


Figure 4. Program Memory Map

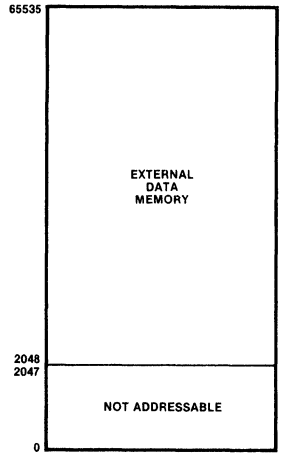


Figure 5. Data Memory Map

LOCATION	REGISTER	IDENTIFIERS
255	STACK POINTER (BITS 7-0)	SPL
254	STACK POINTER (BITS 15-8)	SPH
253	REGISTER POINTER	RP
252	PROGRAM CONTROL FLAGS	FLAGS
251	INTERRUPT MASK REGISTER	IMR
250	INTERRUPT REQUEST REGISTER	IRQ
249	INTERRUPT PRIORITY REGISTER	IPR
248	PORTS 0-1 MODE	P01M
247	PORT 3 MODE	P3M
246	PORT 2 MODE	P2M
245	T0 PRESCALER	PRE0
244	TIMER/COUNTER 0	T0
243	T1 PRESCALER	PRE1
242	TIMER/COUNTER 1	T1
241	TIMER MODE	TMR
240	SERIAL I/O	SIO
	NOT IMPLEMENTED	
127	GENERAL-PURPOSE REGISTERS	
4		
3	PORT 3	P3
2	PORT 2	P2
1	PORT 1	P1
0	PORT 0	P0

Figure 6. The Register File

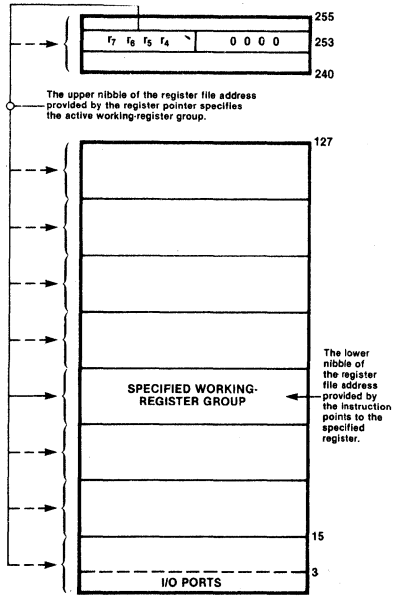


Figure 7. The Register Pointer



## Serial Input/Output

Port 3 lines P3<sub>0</sub> and P3<sub>7</sub> can be programmed as serial I/O lines for full-duplex serial asynchronous receiver/transmitter operation. The bit rate is controlled by Counter/Timer 0, with a maximum rate of 62.5K bits/second for 8 MHz and 94.8K bits/second for 12 MHz.

The Z8601 automatically adds a start bit and two stop bits to transmitted data (Figure 8). Odd parity is also available as an option.

Eight data bits are always transmitted, regardless of parity selection. If parity is enabled, the eighth bit is the odd parity bit. An interrupt request (IRQ<sub>4</sub>) is generated on all transmitted characters.

Received data must have a start bit, eight data bits and at least one stop bit. If parity is on, bit 7 of the received data is replaced by a parity error flag. Received characters generate the IRQ<sub>3</sub> interrupt request.

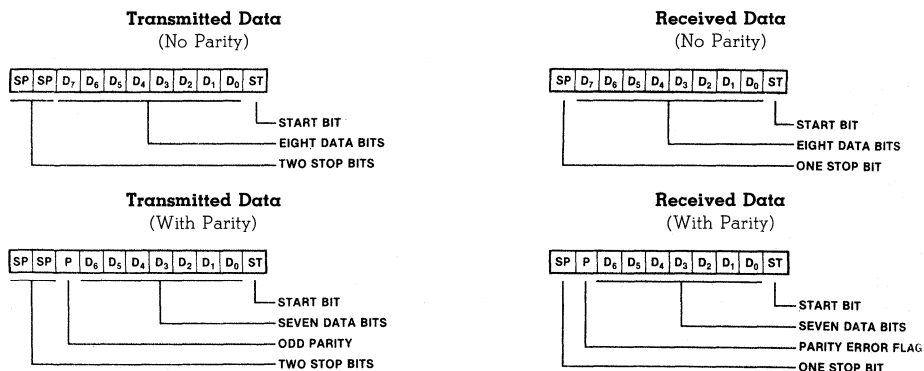


Figure 8. Serial Data Formats

## Counter/Timers

The Z8601 contains two 8-bit programmable counter/timers (T<sub>0</sub> and T<sub>1</sub>), each driven by its own 6-bit programmable prescaler. The T<sub>1</sub> prescaler can be driven by internal or external clock sources; however, the T<sub>0</sub> prescaler is driven by the internal clock only.

The 6-bit prescalers can divide the input frequency of the clock source by any number from 1 to 64. Each prescaler drives its counter, which decrements the value (1 to 256) that has been loaded into the counter. When the counter reaches the end of count, a timer interrupt request—IRQ<sub>4</sub> (T<sub>0</sub>) or IRQ<sub>5</sub> (T<sub>1</sub>)—is generated.

The counters can be started, stopped, restarted to continue, or restarted from the initial value. The counters can also be programmed to stop upon reaching zero (single-pass mode) or to automatically reload

the initial value and continue counting (modulo-n continuous mode). The counters, but not the prescalers, can be read any time without disturbing their value or count mode.

The clock source for T<sub>1</sub> is user-definable and can be the internal microprocessor clock (4 MHz maximum for the 8 MHz device and 6 MHz maximum for the 12 MHz device) divided by four, or an external signal input via Port 3. The Timer Mode register configures the external timer input as an external clock (1 MHz maximum), a trigger input that can be retriggerable or non-retriggerable, or as a gate input for the internal clock. The counter/timers can be porogrammably cascaded by connecting the T<sub>0</sub> output to the input of T<sub>1</sub>. Port 3 line P3<sub>6</sub> also serves as a timer output (T<sub>OUT</sub>) through which T<sub>0</sub>, T<sub>1</sub> or the internal clock can be output.

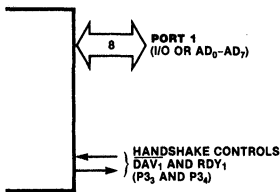
### I/O Ports

The Z8601 has 32 lines dedicated to input and output. These lines are grouped into four ports of eight lines each and are configurable as input, output or address/data. Under software control, the ports can be programmed to provide address outputs, timing, status signals, serial I/O, and parallel I/O with or without handshake. All ports have active pull-ups and pull-downs compatible with TTL loads.

**Port 1** can be programmed as a byte I/O port or as an address/data port for interfacing external memory. When used as an I/O port, Port 1 may be placed under handshake control. In this configuration, Port 3 lines P<sub>33</sub> and P<sub>34</sub> are used as the handshake controls RDY<sub>1</sub> and DAV<sub>1</sub> (Ready and Data Available).

Memory locations greater than 2048 are referenced through Port 1. To interface external memory, Port 1 must be programmed for the multiplexed Address/Data mode. If more than 256 external locations are required, Port 0 must output the additional lines.

Port 1 can be placed in the high-impedance state along with Port 0,  $\overline{AS}$ ,  $\overline{DS}$  and R/W, allowing the Z8601 to share common resources in multiprocessor and DMA applications. Data transfers can be controlled by assigning P<sub>33</sub> as a Bus Acknowledge input and P<sub>34</sub> as a Bus Request output.

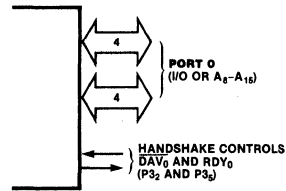


**Figure 9a. Port 1**

**Port 0** can be programmed as a nibble I/O port, or as an address port for interfacing external memory. When used as an I/O port,

Port 0 may be placed under handshake control. In this configuration, Port 3 lines P<sub>32</sub> and P<sub>35</sub> are used as the handshake controls DAV<sub>0</sub> and RDY<sub>0</sub>. Handshake signal assignment is dictated by the I/O direction of the upper nibble P<sub>04</sub>-P<sub>07</sub>.

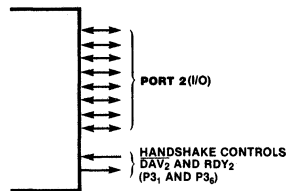
For external memory references, Port 0 can provide address bits A<sub>8</sub>-A<sub>11</sub> (lower nibble) or A<sub>8</sub>-A<sub>15</sub> (lower and upper nibble) depending on the required address space. If the address range requires 12 bits or less, the upper nibble of Port 0 can be programmed independently as I/O while the lower nibble is used for addressing. When Port 0 nibbles are defined as address bits, they can be set to the high-impedance state along with Port 1 and the control signals  $\overline{AS}$ ,  $\overline{DS}$  and R/W.



**Figure 9b. Port 0**

**Port 2** bits can be programmed independently as input or output. The port is always available for I/O operations. In addition, Port 2 can be configured to provide open-drain outputs.

Like Ports 0 and 1, Port 2 may also be placed under handshake control. In this configuration, Port 3 lines P<sub>31</sub> and P<sub>36</sub> are



**Figure 9c. Port 2**



## I/O Ports (Continued)

used as the handshake controls lines  $\overline{DAV}_2$  and  $\overline{RDY}_2$ . The handshake signal assignment for Port 3 lines P3<sub>1</sub> and P3<sub>6</sub> is dictated by the direction (input or output) assigned to bit 7 of Port 2.

**Port 3** lines can be configured as I/O or control lines. In either case, the direction of the eight lines is fixed as four input (P3<sub>0</sub>-P3<sub>3</sub>) and four output (P3<sub>4</sub>-P3<sub>7</sub>). For serial I/O, lines P3<sub>0</sub> and P3<sub>7</sub> are programmed as serial in and serial out respectively.

Port 3 can also provide the following control functions: handshake for Ports 0, 1 and 2 ( $\overline{DAV}$  and  $\overline{RDY}$ ); four external

interrupt request signals (IRQ<sub>0</sub>-IRQ<sub>3</sub>); timer input and output signals (T<sub>IN</sub> and T<sub>OUT</sub>) and Data Memory Select (DM).

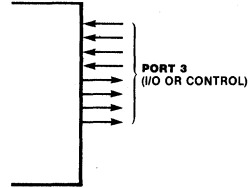


Figure 9d. Port 3

## Interrupts

The Z8601 allows six different interrupts from eight sources: the four Port 3 lines P3<sub>0</sub>-P3<sub>3</sub>, Serial In, Serial Out, and the two counter/timers. These interrupts are both maskable and prioritized. The Interrupt Mask register globally or individually enables or disables the six interrupt requests. When more than one interrupt is pending, priorities are resolved by a programmable priority encoder that is controlled by the Interrupt Priority register.

All Z8601 interrupts are vectored. When an interrupt request is granted, and interrupt machine cycle is entered. This disables all

subsequent interrupts, saves the Program Counter and status flags, and branches to the program memory vector location reserved for that interrupt. This memory location and the next byte contain the 16-bit address of the interrupt service routine for that particular interrupt request.

Polled interrupt systems are also supported. To accommodate a polled structure, any or all of the interrupt inputs can be masked and the Interrupt Request register polled to determine which of the interrupt requests needs service.

## Clock

The on-chip oscillator has a high-gain, parallel-resonant amplifier for connection to a crystal or to any suitable external clock source (XTAL1 = Input, XTAL2 = Output).

The crystal source is connected across XTAL1 and XTAL2, using the recommended

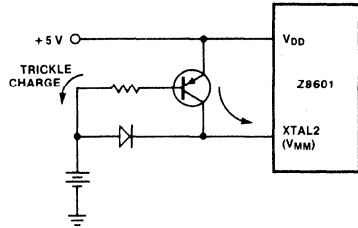
capacitors ( $C_1 \leq 15$  pF) from each pin to ground. The specifications for the crystal are as follows:

- AT cut, parallel resonant
- Fundamental types, 8/12 MHz maximum
- Series resistance,  $R_s \leq 100 \Omega$ .

### Power Down Standby Option

The low-power standby mode allows power to be removed without losing the contents of the 124 general-purpose registers. This mode is available to the user as a bonding option whereby pin 2 (normally XTAL2) is replaced by the  $V_{MM}$  (standby) power supply input. This necessitates the use of an external clock generator (input = XTAL1) rather than a crystal source.

The removal of power, whether intended or due to power failure, must be preceded by a software routine that stores the appropriate status into the register file. Figure 10 shows the recommended circuit for a battery back-up supply system.



**Figure 10. Recommended Driver Circuit for Power Down Operation**

### Instruction Set Notation

**Addressing Modes.** The following notation is used to describe the addressing modes and instruction operations as shown in the instruction summary.

<b>IRR</b>	Indirect register pair or indirect working-register pair address
<b>Irr</b>	Indirect working-register pair only
<b>X</b>	Indexed address
<b>DA</b>	Direct address
<b>RA</b>	Relative address
<b>IM</b>	Immediate
<b>R</b>	Register or working-register address
<b>r</b>	Working-register address only
<b>IR</b>	Indirect-register or indirect working-register address
<b>Ir</b>	Indirect working-register address only
<b>RR</b>	Register pair or working register pair address

**Symbols.** The following symbols are used in describing the instruction set.

<b>dst</b>	Destination location or contents
<b>src</b>	Source location or contents
<b>cc</b>	Condition code (see list)
<b>@</b>	Indirect address prefix
<b>SP</b>	Stack pointer (control registers 254-255)
<b>PC</b>	Program counter
<b>FLAGS</b>	Flag register (control register 252)
<b>RP</b>	Register pointer (control register 253)
<b>IMR</b>	Interrupt mask register (control register 251)

Assignment of a value is indicated by the symbol " $\leftarrow$ ". For example,

$$dst \leftarrow dst + src$$

indicates that the source data is added to the destination data and the result is stored in the destination location. The notation "addr(n)" is used to refer to bit "n" of a given location. For example,

$$dst(7)$$

refers to bit 7 of the destination operand.

**Flags.** Control Register R252 contains the following six flags:

<b>C</b>	Carry flag	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td style="text-align: center;">b7</td> <td style="text-align: center;">b0</td> </tr> <tr> <td style="text-align: center;">C</td> <td style="text-align: center;">Z</td> <td style="text-align: center;">S</td> <td style="text-align: center;">V</td> <td style="text-align: center;">D</td> <td style="text-align: center;">H</td> <td style="text-align: center;">F2</td> <td style="text-align: center;">F1</td> </tr> </table>	b7	b0	C	Z	S	V	D	H	F2	F1
b7	b0											
C	Z		S	V	D	H	F2	F1				
<b>Z</b>	Zero flag											
<b>S</b>	Sign flag											
<b>V</b>	Overflow flag											
<b>D</b>	Decimal-adjust flag											
<b>H</b>	Half-carry flag											

F1 } user flags  
F2 }

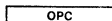
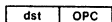
Affected flags are indicated by:

<b>0</b>	Cleared to zero
<b>1</b>	Set to one
<b>*</b>	Set or cleared according to operation
<b>-</b>	Unaffected
<b>X</b>	Undefined

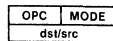


**Condition Codes**

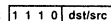
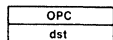
<b>Value</b>	<b>Mnemonic</b>	<b>Meaning</b>	<b>Flags Set</b>
1000		Always true	...
0111	C	Carry	C = 1
1111	NC	No carry	C = 0
0110	Z	Zero	Z = 1
1110	NZ	Not zero	Z = 0
1101	PL	Plus	S = 0
0101	MI	Minus	S = 1
0100	OV	Overflow	V = 1
1100	NOV	No overflow	V = 0
0110	EQ	Equal	Z = 1
1110	NE	Not equal	Z = 0
1001	GE	Greater than or equal	(S XOR V) = 0
0001	LT	Less than	(S XOR V) = 1
1010	GT	Greater than	[Z OR (S XOR V)] = 0
0010	LE	Less than or equal	[Z OR (S XOR V)] = 1
1111	UGE	Unsigned greater than or equal	C = 0
0111	ULT	Unsigned less than	C = 1
1011	UGT	Unsigned greater than	(C = 0 AND Z = 0) = 1
0011	ULE	Unsigned less than or equal	(C OR Z) = 1
0000		Never true	...

**Instruction Formats**

 CCF, DI, EI, IRET, NOP,  
RCF, RET, SCF


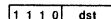
INC r

**One-Byte Instructions**


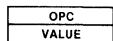
OR


 CLR, CPL, DA, DEC,  
DECW, INC, INCW, POP,  
PUSH, RL, RLC, RR,  
RRC, SRA, SWAP


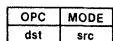
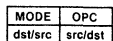
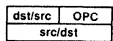
OR



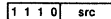
JP, CALL (Indirect)



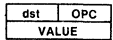
SRP


 ADC, ADD, AND, CP,  
OR, SBC, SUB,  
TCM, TM, XOR

 LD, LDE, LDEI,  
LDC, LDCI


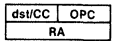
OR



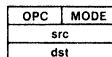
LD



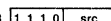
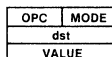
LD



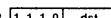
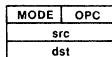
DJNZ, JR



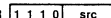
OR


 ADC, ADD, AND, CP,  
LD, OR, SBC, SUB,  
TCM, TM, XOR


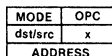
OR


 ADC, ADD, AND, CP,  
LD, OR, SBC, SUB,  
TCM, TM, XOR


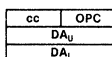
OR



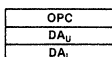
LD



LD



JP



CALL

**Two-Byte Instructions**
**Three-Byte Instructions**






Figure 12. Instruction Formats



Z8601/L

Instruction Summary

Instruction and Operation	Addr Mode		Opcode Byte (Hex)	Flags Affected					
	dst	src		C	Z	S	V	D	H
<b>ADC</b> dst,src dst - dst + src + C	(Note 1)		1□	*	*	*	*	0	*
<b>ADD</b> dst,src dst - dst + src	(Note 1)		0□	*	*	*	*	0	*
<b>AND</b> dst,src dst - dst AND src	(Note 1)		5□	-	*	*	0	-	-
<b>CALL</b> dst SP - SP - 2 @SP - PC; PC - dst	DA IRR		D6 D4	-	-	-	-	-	-
<b>CCF</b> C - NOT C			EF	*	-	-	-	-	-
<b>CLR</b> dst dst - 0	R IR		B0 B1	-	-	-	-	-	-
<b>COM</b> dst dst - NOT dst	R IR		60 61	-	*	*	0	-	-
<b>CP</b> dst,src dst - src	(Note 1)		A□	*	*	*	*	-	-
<b>DA</b> dst dst - DA dst	R IR		40 41	*	*	*	X	-	-
<b>DEC</b> dst dst - dst - 1	R IR		00 01	-	*	*	*	-	-
<b>DECW</b> dst dst - dst - 1	RR IR		80 81	-	*	*	*	-	-
<b>DI</b> IMR (7) - 0			8F	-	-	-	-	-	-
<b>DJNZ</b> r,dst r - r - 1 if r ≠ 0 PC - PC + dst Range: +127, -128	RA		rA r=0-F	-	-	-	-	-	-
<b>EI</b> IMR (7) - 1			9F	-	-	-	-	-	-
<b>INC</b> dst dst - dst + 1	r R IR		rE r=0-F 20 21	-	*	*	*	-	-
<b>INCW</b> dst dst - dst + 1	RR IR		A0 A1	-	*	*	*	-	-
<b>IRET</b> FLAGS - @SP; SP - SP + 1 PC - @SP; SP - SP + 2; IMR (7) - 1			BF	*	*	*	*	*	*
<b>JP</b> cc,dst if cc is true PC - dst	DA IRR		cD c=0-F 30	-	-	-	-	-	-
<b>JR</b> cc,dst if cc is true, PC - PC + dst Range: +127, -128	RA		cB c=0-F	-	-	-	-	-	-

Instruction and Operation	Addr Mode		Opcode Byte (Hex)	Flags Affected					
	dst	src		C	Z	S	V	D	H
<b>LD</b> dst,src dst - src	r r R	Im R r	rC r8 r9 r=0-F	-	-	-	-	-	-
	r X r R R R IR IR	X r Ir r R R Im Im R	C7 D7 E3 F3 E4 E5 E6 E7 F5						
<b>LDC</b> dst,src dst - src	r Irr	Irr r	C2 D2	-	-	-	-	-	-
<b>LDCI</b> dst,src dst - src r - r + 1; rr - rr + 1	Ir Irr	Irr Ir	C3 D3	-	-	-	-	-	-
<b>LDE</b> dst,src dst - src	r Irr	Irr r	82 92	-	-	-	-	-	-
<b>LDEI</b> dst,src dst - src r - r + 1; rr - rr + 1	Ir Irr	Irr Ir	83 93	-	-	-	-	-	-
<b>NOP</b>			FF	-	-	-	-	-	-
<b>OR</b> dst,src dst - dst OR src	(Note 1)		4□	-	*	*	0	-	-
<b>POP</b> dst dst - @SP SP - SP + 1	R IR		50 51	-	-	-	-	-	-
<b>PUSH</b> src SP - SP - 1; @SP - src	R IR		70 71	-	-	-	-	-	-
<b>RCF</b> C - 0			CF	0	-	-	-	-	-
<b>RET</b> PC - @SP; SP - SP + 2			AF	-	-	-	-	-	-
<b>RL</b> dst	 R IR		90 91	*	*	*	*	-	-
<b>RLC</b> dst	 R IR		10 11	*	*	*	*	-	-
<b>RR</b> dst	 R IR		E0 E1	*	*	*	*	-	-
<b>RRC</b> dst	 R IR		C0 C1	*	*	*	*	-	-
<b>SBC</b> dst,src dst - dst - src - C	(Note 1)		3□	*	*	*	*	1	*
<b>SCF</b> C - 1			DF	1	-	-	-	-	-
<b>SRA</b> dst	 R IR		D0 D1	*	*	*	0	-	-



**Instruction Summary (Continued)**

Instruction and Operation	Addr Mode		Opcode Byte (Hex)	Flags Affected							
	dst	src		C	Z	S	V	D	H		
<b>SRP</b> src RP ← src		Im	31	-	-	-	-	-	-		
<b>SUB</b> dst,src dst ← dst - src		(Note 1)	2□	*	*	*	*	1	*		
<b>SWAP</b> dst	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>□</td></tr> <tr><td>□</td></tr> </table>	□	□	R IR	F0 F1	X	*	*	X	-	-
□											
□											

Instruction and Operation	Addr Mode		Opcode Byte (Hex)	Flags Affected					
	dst	src		C	Z	S	V	D	H
<b>TCM</b> dst,src (NOT dst) AND src		(Note 1)	6□	-	*	*	0	-	-
<b>TM</b> dst,src dst AND src		(Note 1)	7□	-	*	*	0	-	-
<b>XOR</b> 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 is found in the instruction set table above. The second nibble is expressed symbolically by a □ in this table, and its value is found in the following table to the left of the applicable addressing mode pair.

For example, to determine the opcode of an ADC instruction use the addressing modes r (destination) and Ir (source). The result is 13.

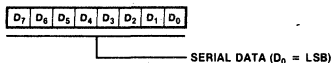
Addr Mode		Lower Opcode Nibble
dst	src	
r	r	2
r	Ir	3
R	R	4
R	IR	5
R	IM	6
IR	IM	7



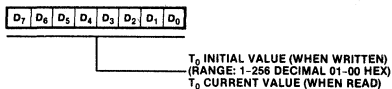
# Z8601/L

## Registers

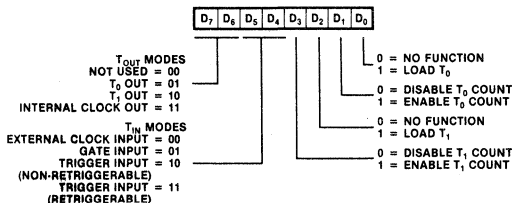
**R240 SIO**  
Serial I/O Register  
(F0H; Read/Write)



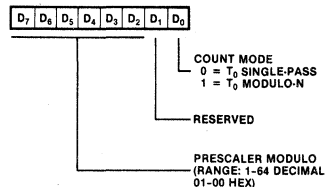
**R244 T0**  
Counter/Timer 0 Register  
(F4H; Read/Write)



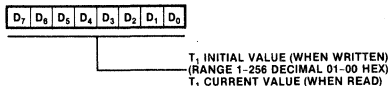
**R241 TMR**  
Timer Mode Register  
(F1H; Read/Write)



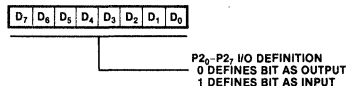
**R245 PRE0**  
Prescaler 0 Register  
(F5H; Write Only)



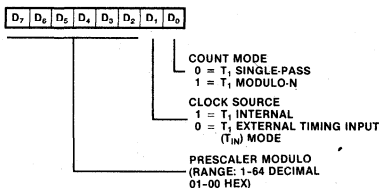
**R242 T1**  
Counter Timer 1 Register  
(F2H; Read/Write)



**R246 P2M**  
Port 2 Mode Register  
(F6H; Write Only)



**R243 PRE1**  
Prescaler 1 Register  
(F3H; Write Only)



**R247 P3M**  
Port 3 Mode Register  
(F7H; Write Only)

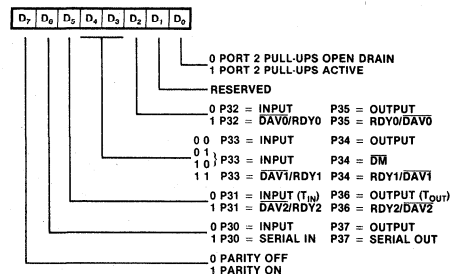
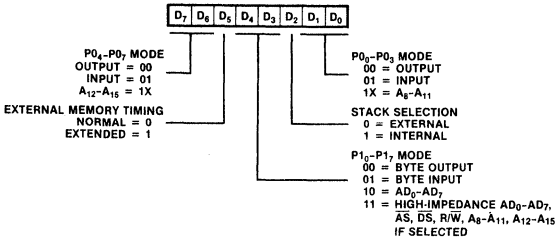


Figure 12. Control Registers

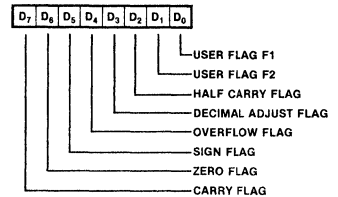


Registers (Continued)

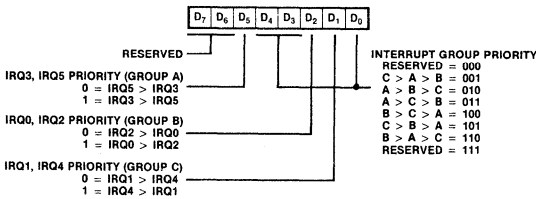
**R248 P01M**  
Port 0 and 1 Mode Register  
(F8H; Write Only)



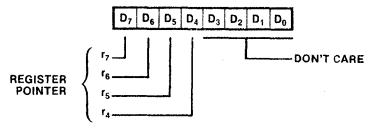
**R252 FLAGS**  
Flag Register  
(FCH; Read/Write)



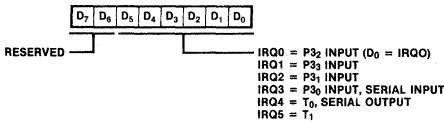
**R249 IPR**  
Interrupt Priority Register  
(F9H; Write Only)



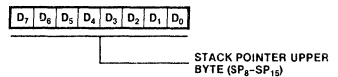
**R253 RP**  
Register Pointer  
(FDH; Read/Write)



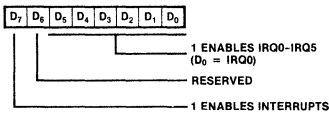
**R250 IRQ**  
Interrupt Request Register  
(FAH; Read/Write)



**R254 SPH**  
Stack Pointer  
(FEH; Read/Write)



**R251 IMR**  
Interrupt Mask Register  
(FBH; Read/Write)



**R255 SPL**  
Stack Pointer  
(FFH; Read/Write)

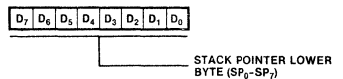


Figure 12. Control Registers (Continued)

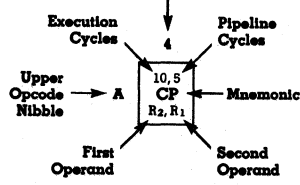


Z8601/L

Opcode Map

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	6,5 DEC R <sub>1</sub>	6,5 DEC IR <sub>1</sub>	6,5 ADD r <sub>1</sub> , r <sub>2</sub>	6,5 ADD r <sub>1</sub> , Ir <sub>2</sub>	10,5 ADD R <sub>2</sub> , R <sub>1</sub>	10,5 ADD IR <sub>2</sub> , R <sub>1</sub>	10,5 ADD R <sub>1</sub> , IM	10,5 ADD IR <sub>1</sub> , IM	6,5 LD r <sub>1</sub> , R <sub>2</sub>	6,5 LD r <sub>2</sub> , R <sub>1</sub>	12/10,5 DJNZ r <sub>1</sub> , RA	12/10,0 JR cc, RA	6,5 LD r <sub>1</sub> , IM	12/10,0 JP cc, DA	6,5 INC r <sub>1</sub>	
1	6,5 RLC R <sub>1</sub>	6,5 RLC IR <sub>1</sub>	6,5 ADC r <sub>1</sub> , r <sub>2</sub>	6,5 ADC r <sub>1</sub> , Ir <sub>2</sub>	10,5 ADC R <sub>2</sub> , R <sub>1</sub>	10,5 ADC IR <sub>2</sub> , R <sub>1</sub>	10,5 ADC R <sub>1</sub> , IM	10,5 ADC IR <sub>1</sub> , IM								
2	6,5 INC R <sub>1</sub>	6,5 INC IR <sub>1</sub>	6,5 SUB r <sub>1</sub> , r <sub>2</sub>	6,5 SUB r <sub>1</sub> , Ir <sub>2</sub>	10,5 SUB R <sub>2</sub> , R <sub>1</sub>	10,5 SUB IR <sub>2</sub> , R <sub>1</sub>	10,5 SUB R <sub>1</sub> , IM	10,5 SUB IR <sub>1</sub> , IM								
3	8,0 JP IRR <sub>1</sub>	6,1 SRP IM	6,5 SBC r <sub>1</sub> , r <sub>2</sub>	6,5 SBC r <sub>1</sub> , Ir <sub>2</sub>	10,5 SBC R <sub>2</sub> , R <sub>1</sub>	10,5 SBC IR <sub>2</sub> , R <sub>1</sub>	10,5 SBC R <sub>1</sub> , IM	10,5 SBC IR <sub>1</sub> , IM								
4	8,5 DA R <sub>1</sub>	8,5 DA IR <sub>1</sub>	6,5 OR r <sub>1</sub> , r <sub>2</sub>	6,5 OR r <sub>1</sub> , Ir <sub>2</sub>	10,5 OR R <sub>2</sub> , R <sub>1</sub>	10,5 OR IR <sub>2</sub> , R <sub>1</sub>	10,5 OR R <sub>1</sub> , IM	10,5 OR IR <sub>1</sub> , IM								
5	10,5 POP R <sub>1</sub>	10,5 POP IR <sub>1</sub>	6,5 AND r <sub>1</sub> , r <sub>2</sub>	6,5 AND r <sub>1</sub> , Ir <sub>2</sub>	10,5 AND R <sub>2</sub> , R <sub>1</sub>	10,5 AND IR <sub>2</sub> , R <sub>1</sub>	10,5 AND R <sub>1</sub> , IM	10,5 AND IR <sub>1</sub> , IM								
6	6,5 COM R <sub>1</sub>	6,5 COM IR <sub>1</sub>	6,5 TCM r <sub>1</sub> , r <sub>2</sub>	6,5 TCM r <sub>1</sub> , Ir <sub>2</sub>	10,5 TCM R <sub>2</sub> , R <sub>1</sub>	10,5 TCM IR <sub>2</sub> , R <sub>1</sub>	10,5 TCM R <sub>1</sub> , IM	10,5 TCM IR <sub>1</sub> , IM								
7	10/12, 1 PUSH R <sub>2</sub>	12/14, 1 PUSH IR <sub>2</sub>	6,5 TM r <sub>1</sub> , r <sub>2</sub>	6,5 TM r <sub>1</sub> , Ir <sub>2</sub>	10,5 TM R <sub>2</sub> , R <sub>1</sub>	10,5 TM IR <sub>2</sub> , R <sub>1</sub>	10,5 TM R <sub>1</sub> , IM	10,5 TM IR <sub>1</sub> , IM								
8	10,5 DECW RR <sub>1</sub>	10,5 DECW IR <sub>1</sub>	12,0 LDE r <sub>1</sub> , Ir <sub>2</sub>	18,0 LDEI Ir <sub>1</sub> , Ir <sub>2</sub>												6,1 DI
9	6,5 RL R <sub>1</sub>	6,5 RL IR <sub>1</sub>	12,0 LDE r <sub>2</sub> , Ir <sub>1</sub>	18,0 LDEI Ir <sub>2</sub> , Ir <sub>1</sub>												6,1 EI
A	10,5 INCW RR <sub>1</sub>	10,5 INCW IR <sub>1</sub>	6,5 CP r <sub>1</sub> , r <sub>2</sub>	6,5 CP r <sub>1</sub> , Ir <sub>2</sub>	10,5 CP R <sub>2</sub> , R <sub>1</sub>	10,5 CP IR <sub>2</sub> , R <sub>1</sub>	10,5 CP R <sub>1</sub> , IM	10,5 CP IR <sub>1</sub> , IM								14,0 RET
B	6,5 CLR R <sub>1</sub>	6,5 CLR IR <sub>1</sub>	6,5 XOR r <sub>1</sub> , r <sub>2</sub>	6,5 XOR r <sub>1</sub> , Ir <sub>2</sub>	10,5 XOR R <sub>2</sub> , R <sub>1</sub>	10,5 XOR IR <sub>2</sub> , R <sub>1</sub>	10,5 XOR R <sub>1</sub> , IM	10,5 XOR IR <sub>1</sub> , IM								16,0 IRET
C	6,5 RRC R <sub>1</sub>	6,5 RRC IR <sub>1</sub>	12,0 LDC r <sub>1</sub> , Ir <sub>2</sub>	18,0 LDCI Ir <sub>1</sub> , Ir <sub>2</sub>				10,5 LD r <sub>1</sub> , r <sub>2</sub> , R <sub>2</sub>								6,5 RCF
D	6,5 SRA R <sub>1</sub>	6,5 SRA IR <sub>1</sub>	12,0 LDC r <sub>2</sub> , Ir <sub>1</sub>	18,0 LDCI Ir <sub>2</sub> , Ir <sub>1</sub>	20,0 CALL* IR <sub>1</sub>		20,0 CALL DA	10,5 LD r <sub>2</sub> , r <sub>2</sub> , R <sub>1</sub>								6,5 SCF
E	6,5 RR R <sub>1</sub>	6,5 RR IR <sub>1</sub>		6,5 LD r <sub>1</sub> , Ir <sub>2</sub>	10,5 LD R <sub>2</sub> , R <sub>1</sub>	10,5 LD IR <sub>2</sub> , R <sub>1</sub>	10,5 LD R <sub>1</sub> , IM	10,5 LD IR <sub>1</sub> , IM								6,5 CCF
F	8,5 SWAP R <sub>1</sub>	8,5 SWAP IR <sub>1</sub>		6,5 LD Ir <sub>1</sub> , r <sub>2</sub>		10,5 LD R <sub>2</sub> , IR <sub>1</sub>										6,0 NOP

Bytes per Instruction



**Legend:**  
 R = 8-Bit Address  
 r = 4-Bit Address  
 R<sub>1</sub> or r<sub>1</sub> = Dest Address  
 R<sub>2</sub> or r<sub>2</sub> = Src Address

**Sequence:**  
 Opcode, First Operand, Second Operand

**Note:** The blank areas are not defined.

### Absolute Maximum Ratings

Voltages on all pins with respect to GND ..... -0.3 V to +7.0 V  
 Operating Ambient Temperature ..... 0°C to +70°C  
 Storage Temperature ..... -65°C to +150°C

Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

### Standard Test Conditions

The characteristics below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND. Positive current flows into the reference pin. Standard conditions are as follows:

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

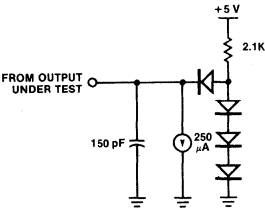


Figure 13. Test Load 1

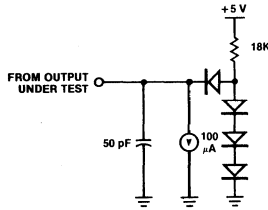


Figure 14. Test Load 2

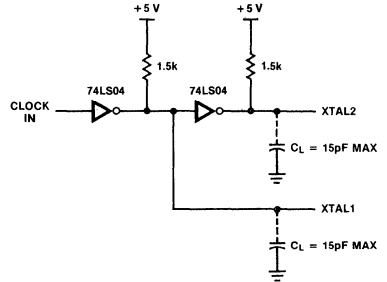


Figure 15. External Clock Interface Circuit  
 (Both the clock and complement are required)



**Z8601/L****DC Characteristics**

Symbol	Parameter	Min	Max	Unit	Condition
V <sub>CH</sub>	Clock Input High Voltage	3.8	V <sub>CC</sub>	V	Driven by External Clock Generator
V <sub>CL</sub>	Clock Input Low Voltage	-0.3	0.8	V	Driven by External Clock Generator
V <sub>IH</sub>	Input High Voltage	2.0	V <sub>CC</sub>	V	
V <sub>IL</sub>	Input Low Voltage	-0.3	0.8	V	
V <sub>RH</sub>	Reset Input High Voltage	3.8	V <sub>CC</sub>	V	
V <sub>RL</sub>	Reset Input Low Voltage	-0.3	0.8	V	
V <sub>OH</sub>	Output High Voltage	2.4		V	I <sub>OH</sub> = -250 $\mu$ A
V <sub>OL</sub>	Output Low Voltage		0.4	V	I <sub>OL</sub> = +2.0 mA
I <sub>IL</sub>	Input Leakage	-10	10	$\mu$ A	0 V $\leq$ V <sub>IN</sub> $\leq$ +5.25 V
I <sub>OL</sub>	Output Leakage	-10	10	$\mu$ A	0 V $\leq$ V <sub>IN</sub> $\leq$ +5.25 V
I <sub>IR</sub>	Reset Input Current		-50	$\mu$ A	V <sub>CC</sub> = +5.25 V, V <sub>RL</sub> = 0 V
I <sub>CC</sub>	V <sub>CC</sub> Supply Current		120	mA	
			80*		
I <sub>MM</sub>	V <sub>MM</sub> Supply Current		10	mA	Power Down Mode
V <sub>MM</sub>	Backup Supply Voltage	3	V <sub>CC</sub>	V	Power Down

\* This value is for Z8601 only.



External I/O or Memory Read and Write Timing

No	Symbol	Paramter	Z8601/L		Z8601A		Notes*†
			Min	Max	Min	Max	
1	TdA(AS)	Address Valid to $\overline{AS}$ $\uparrow$ Delay	50		35		1,2,3
2	TdAS(A)	$\overline{AS}$ $\uparrow$ to Address Float Dealy	70		45		1,2,3
3	TdAS(DR)	$\overline{AS}$ $\uparrow$ to Read Data Required Valid		360		220	1,2,3,4
4	TwAS	$\overline{AS}$ Low Width	80		55		1,2,3
5	TdAz(DS)	Address Float to $\overline{DS}$ $\downarrow$	0		0		1
6	TwDSR	$\overline{DS}$ (Read) Low Width	250		185		1,2,3,4
7	TwDSW	$\overline{DS}$ (Write) Low Width	160		110		1,2,3,4
8	TdDSR(DR)	$\overline{DS}$ $\downarrow$ to Read Data Required Valid		200		130	1,2,3,4
9	ThDR(DS)	Read Data to $\overline{DS}$ $\uparrow$ Hold Time	0		0		1
10	TdDS(A)	$\overline{DS}$ $\uparrow$ to Address Active Delay	70		45		1,2,3
11	TdDS(AS)	$\overline{DS}$ $\uparrow$ to $\overline{AS}$ $\downarrow$ Delay	70		55		1,2,3
12	TdR/W(AS)	R/W Valid to $\overline{AS}$ $\uparrow$ Delay	50		30		1,2,3
13	TdDS(R/W)	$\overline{DS}$ $\uparrow$ to R/W Not Valid	60		35		1,2,3
14	TdDW(DSW)	Write Data Valid to $\overline{DS}$ (Write) $\downarrow$ Delay	50		35		1,2,3
15	TdDS(DW)	$\overline{DS}$ $\uparrow$ to Write Data Not Valid Delay	70		45		1,2,3
16	TdA(DR)	Address Valid to Read Data Required Valid		410		255	1,2,3,4
17	TdAS(DS)	$\overline{AS}$ $\uparrow$ to $\overline{DS}$ $\downarrow$ Delay	80		55		1,2,3

- NOTES:
1. Test Load 1
  2. Timing numbers given are for minimum T<sub>pC</sub>.
  3. Also see clock cycle time dependent characteristics table.
  4. When using extended memory timing add 2 T<sub>pC</sub>.
  5. All timing reference use 2.0 V for a logic «1» and 0.8 V for a logic «0».
  - \* All units in nanoseconds (ns).
  - † Timings are preliminary and subject to change.

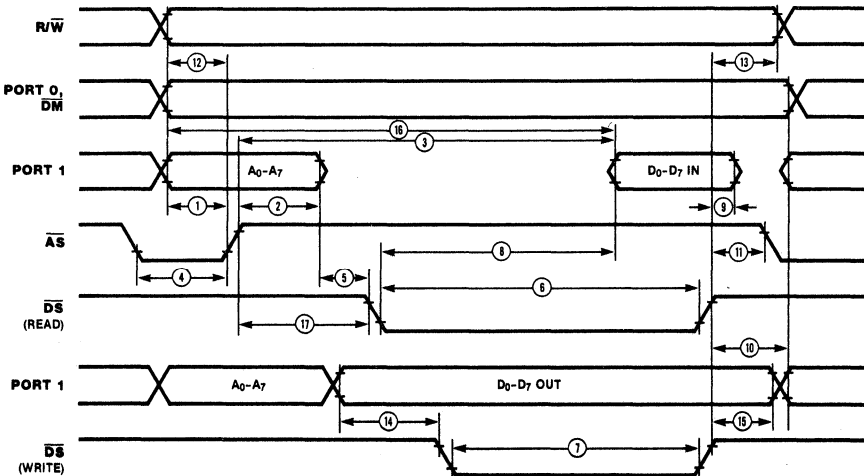


Figure 16. External I/O or Memory Read/Write



Additional Timing Table

No	Symbol	Parameter	Z8601/L		Z8601A		Notes**†
			Min	Max	Min	Max	
1	TpC	Input Clock Period	125	1000	83	1000	1
2	TrC, TfC	Clock Input Rise And Fall Times		25		15	1
3	TwC	Input Clock Width	37		26		1
4	TwTinL	Timer Input Low Width	100		70		2
5	TwTinH	Timer Input High Width	3TpC		3TpC		2
6	TpTin	Timer Input Period	8TpC		8TpC		2
7	TrTin, TfTin	Timer Input Rise And Fall Times		100		100	2
8a	TwiL	Interrupt Request Input Low Time	100		70		2,3
8b	TwiH	Interrupt Request Input Low Time	3TpC		3TpC		2,4
9	TwiH	Interrupt Request Input High Time	3TpC		3TpC		2,3

NOTES:

- 1. Clock timing references uses 3.8 V for a logic "1" and 0.8 V for a logic "0".
- 2. Timing reference uses 2.0 V for a logic "1" and 0.8 V for a logic "0".

- 3. Interrupt request via Port 3 (P31-P33).
- 4. Interrupt request via Port 3 (P30).

\* Units in nanoseconds (ns).

† Timings are preliminary and subject to change.

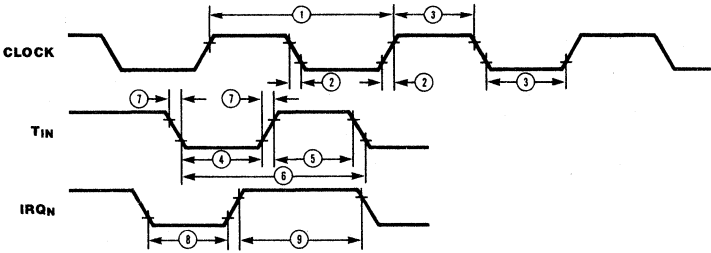


Figure 17. Additional Timing



**Handshake Timing**

No	Symbol	Paramter	Z8601/L		Z8601A		Notes*†
			Min	Max	Min	Max	
1	TsDI(DAV)	Data In Setup Time	0		0		
2	ThDI(DAV)	Data In Hold Time	230		160		
3	TwDAV	Data Available Width	175		120		
4	TdDAVif(RDY)	$\overline{DAV}$ ↓ Input to RDY ↓ Delay		175		120	1,2
5	TdDAVoif(RDY)	$\overline{DAV}$ ↓ Output to RDY ↓ Delay	0		0		1,3
6	TdDAVif(RDY)	$\overline{DAV}$ ↑ Input to RDY ↑ Delay		175		120	1,2
7	TdDAVoif(RDY)	$\overline{DAV}$ ↑ Output to RDY ↑ Delay	0		0		1,3
8	TdDO(DAV)	Data Out to $\overline{DAV}$ ↓ Delay	50		30		1
9	TdRDY(DAV)	Rdy ↓ Input to $\overline{DAV}$ ↑ Delay	0	200	0	140	1

NOTES:

1. Test Load 1
2. Input handshake
3. Output handshake
4. All timing references use 2.0 V for a logic "1" and 0.8 V for a logic "0".
- \* Units in nanoseconds (ns).
- † Timings are preliminary and subject to change.

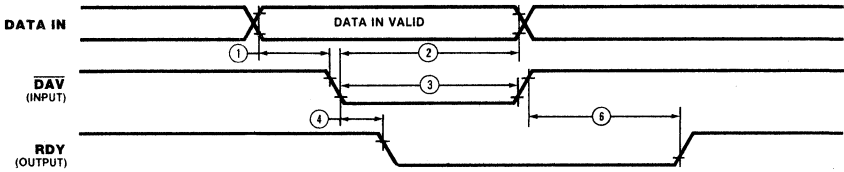


Figure 18. Input Handshake

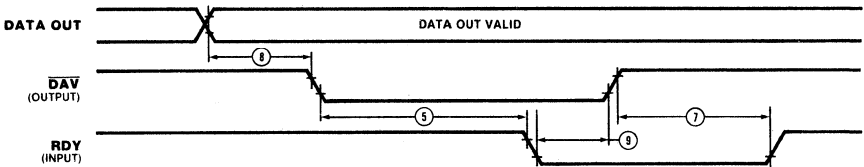


Figure 19. Output Handshake



# Z8601/L

## Clock-Cycle-Time-Dependent Characteristics

Number	Symbol	Z8601/L Equation	Z8601A Equation
1	TdA(AS)	TpC-75	TpC-50
2	TdAS(A)	TpC-55	TpC-40
3	TdAS(DR)	4TpC-140*	4TpC-110*
4	TwAS	TpC-45	TpC-30
6	TwDSR	3TpC-125*	3TpC-65*
7	TwDSW	2TpC-90*	2TpC-55*
8	TdDSR(DR)	3TpC-175*	3TpC-120*
10	Td(DS)A	TpC-55	TpC-40
11	TdDS(AS)	TpC-55	TpC-30
12	TdR/W(AS)	TpC-75	TpC-55
13	TdDS(R/W)	TpC-65	TpC-50
14	TdDW(DSW)	TpC-75	TpC-50
15	TdDS(DW)	TpC-55	TpC-40
16	TdA(DR)	5TpC-215*	5TpC-160*
17	TdAS(DS)	TpC-45	TpC-30

\* Add 2TpC when using extended memory timing

## Ordering Information

Type	Package	Temp.	Clock	Description
Z8601 B1	Plastic	0/+70°C	8 MHz	2K ROM Microcomputer
Z8601 B6	Plastic	-40/+85°C		
Z8601 D1	Ceramic	0/+70°C		
Z8601 D6	Ceramic	-40/+85°C		
Z8601 C1	Plastic Chip Carrier	0/+70°C		
Z8601 C6	Plastic Chip Carrier	-40/+85°C		
Z8601 K1	Ceramic Chip Carrier	0/+70°C		
Z8601 K6	Ceramic Chip Carrier	-40/+85°C		
Z8601A B1	Plastic	0/+70°C	12 MHz	2K ROM Microcomputer
Z8601A B6	Plastic	-40/+85°C		
Z8601A D1	Ceramic	0/+70°C		
Z8601A D6	Ceramic	-40/+85°C		
Z8601A C1	Plastic Chip Carrier	0/+70°C		
Z8601A C6	Plastic Chip Carrier	-40/+85°C		
Z8601A K1	Ceramic Chip Carrier	0/+70°C		
Z8601A K6	Ceramic Chip Carrier	-40/+85°C		
Z8601L B1	Plastic	0/+70°C	8 MHz	2K ROM Microcomputer Low Power version
Z8601L B6	Plastic	-40/+85°C		
Z8601L D1	Ceramic	0/+70°C		
Z8601L D6	Ceramic	-40/+85°C		
Z8601L C1	Plastic Chip Carrier	0/+70°C		
Z8601L C6	Plastic Chip Carrier	-40/+85°C		
Z8601L K1	Ceramic Chip Carrier	0/+70°C		
Z8601L K6	Ceramic Chip Carrier	-40/+85°C		



# Z8611/L

## Z8 4K ROM Microcomputer

- Complete microcomputer, 4K bytes of ROM, 128 bytes of RAM, 32 I/O lines, and up to 60K bytes addressable external space each for program and data memory.
- 144-byte register file, including 124 general-purpose registers, four I/O port registers, and 16 status and control registers.
- Minimum instruction execution time 1  $\mu$ s at 12 MHz.
- Vectored, priority interrupts for I/O, counter/timers, and UART.
- Full-duplex UART and two programmable 8-bit counter/timers, each with a 6-bit programmable prescaler.
- Register Pointer so that short, fast instructions can access any of nine working register groups in 1  $\mu$ s.
- On-chip oscillator which accepts crystal or external clock drive.
- Low-power standby option that retains contents of general-purpose registers.
- Single +5 V power supply—all pins TTL-compatible.
- Low Power version (Z8611L):
  - Available 8 MHz
  - Current consumption 80 mA
- Available in 8 and 12 MHz versions.

### General Description

The Z8611 microcomputer introduces a new level of sophistication to single-chip architecture. Compared to earlier single-chip microcomputers, the Z8611 offers faster execution; more efficient use of memory; more sophisticated interrupt, input/output and bit-manipulation capabilities; and easier system expansion.

Under program control, the Z8611 can be

tailored to the needs of its user. It can be configured as a stand-alone microcomputer with 4K bytes of internal ROM, a traditional microprocessor that manages up to 120K bytes of external memory, or a parallel-processing element in a system with other processors and peripheral controllers linked by the Z-BUS. In all configurations, a large number of pins remain available for I/O.

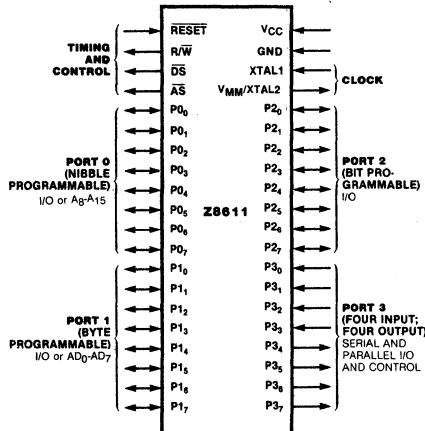


Figure 1. Logic Functions



# Z8611/L

## General Description (Continued)

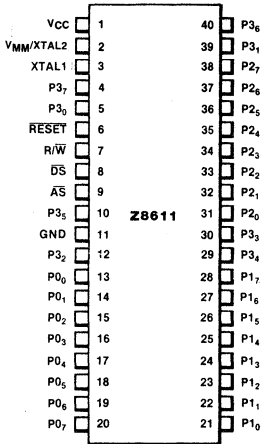


Figure 2. Pin Configuration

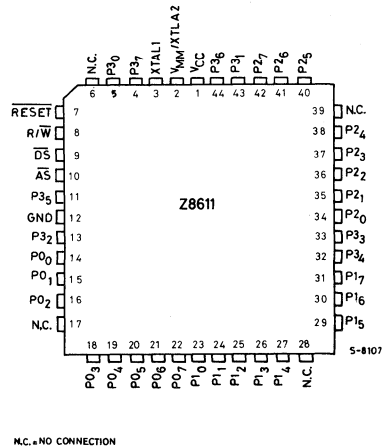


Figure 2a. Chip Carrier Pin Configuration

## Architecture

Z8611 architecture is characterized by a flexible I/O scheme, an efficient register and address space structure and a number of ancillary features that are helpful in many applications.

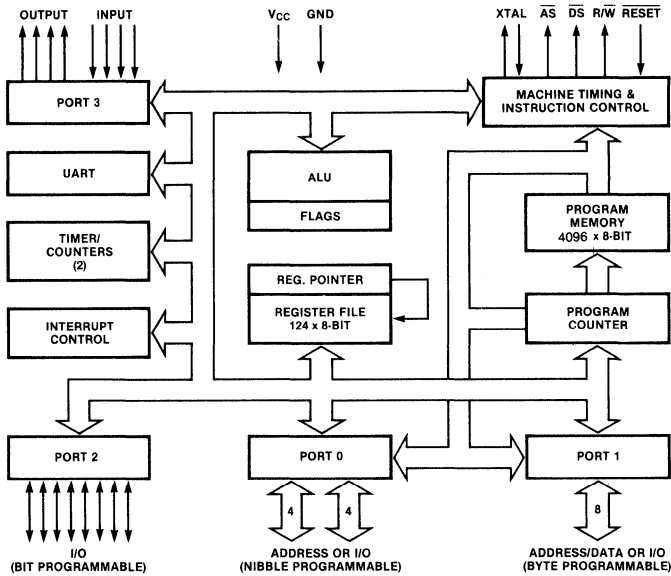
Microcomputer applications demand powerful I/O capabilities. The Z8611 fulfills this with 32 pins dedicated to input and output. These lines are grouped into four ports of eight lines each and are configurable under software control to provide timing, status signals, serial or parallel I/O with or without handshake, and an address/data bus for interfacing external memory.

Because the multiplexed address/data bus is merged with the I/O-oriented ports, the Z8611 can assume many different memory and I/O configurations. These configurations

range from a self-contained microcomputer to a microprocessor that can address 120K bytes of external memory (Figure 3).

Three basic address spaces are available to support this wide range of configurations: program memory (internal and external), data memory (external) and the register file (internal). The 144-byte random-access register file is composed of 124 general-purpose registers, four I/O port registers, and 16 control and status registers.

To unburden the program from coping with real-time problems such as serial data communication and counting/timing, an asynchronous receiver/transmitter (UART) and two counter/timers with a large number of user-selectable modes are offered on-chip. Hardware support for the UART is minimized because one of the on-chip timers supplies the bit rate.

**Architecture** (Continued)

**Figure 3. Functional Block Diagram**
**Pin Description**

**AS.** *Address Strobe* (output, active Low). Address Strobe is pulsed once at the beginning of each machine cycle. Addresses output via Port 1 for all external program or data memory transfers are valid at the trailing edge of AS. Under program control, AS can be placed in the high-impedance state along with Ports 0 and 1, Data Strobe and Read/Write.

**DS.** *Data Strobe* (output, active Low). Data Strobe is activated once for each external memory transfer.

**P0<sub>0</sub>-P0<sub>7</sub>.** *I/O Port Lines* (input/output, TTL compatible). 8 lines Nibble Programmable that can be configured under program control for I/O or external memory interface.

**P1<sub>0</sub>-P1<sub>7</sub>.** *I/O Port Lines* (input/output, TTL compatible). 8 lines Byte Programmable that can be configured under program control for I/O or multiplexed address (A<sub>0</sub>-A<sub>7</sub>) and data (D<sub>0</sub>-D<sub>7</sub>) lines used to interface with program/data memory.

**P2<sub>0</sub>-P2<sub>7</sub>.** *I/O Port Lines* (input/output, TTL compatible). 8 lines Bit Programmable. In addition they can be configured to provide open-drain output.

**P3<sub>0</sub>-P3<sub>4</sub>.** *Input Port Lines* (TTL compatible). They can also be configured as control lines.

**P3<sub>5</sub>-P3<sub>7</sub>.** *Output Port Lines* (TTL compatible). They can also be configured as control lines.





---

**Pin Description** (Continued)

**RESET.** *Reset* (input, active Low).  $\overline{\text{RESET}}$  initializes the Z8611. When  $\overline{\text{RESET}}$  is deactivated, program execution begins from internal program location 000C<sub>H</sub>.

**R/ $\overline{\text{W}}$ .** *Read/Write* (output). R/ $\overline{\text{W}}$  is Low when the Z8611 is writing to external program or data memory.

**XTAL1, XTAL2.** *Crystal 1, Crystal 2* (time-base input and output). These pins connect a parallel-resonant crystal (8 or 12 MHz maximum) or an external single-phase clock (8 or 12 MHz maximum) to the on-chip clock oscillator and buffer.

---

**Address Spaces**

**Program Memory.** The 16-bit program counter addresses 64K bytes of program memory space. Program memory can be located in two areas: one internal and the other external (Figure 4). The first 4096 bytes consist of on-chip mask-programmed ROM. At addresses 4096 and greater, the Z8611 executes external program memory fetches.

The first 12 bytes of program memory are reserved for the interrupt vectors. These locations contain six 16-bit vectors that correspond to the six available interrupts.

**Data Memory.** The Z8611 can address 60K bytes of external data memory beginning at location 4096 (Figure 5). External data memory may be included with or separated from the external program memory space. DM, an optional I/O function that can be programmed to appear on pin P3<sub>4</sub>, is used to distinguish between data and program memory space.

**Register File.** The 144-byte register file includes four I/O port registers (R0-R3), 124

general-purpose registers (R4-R127) and 16 control and status registers (R240-R255). These registers are assigned the address locations shown in Figure 6.

Z8611 instructions can access registers directly or indirectly with an 8-bit address field. The Z8611 also allows short 4-bit register addressing using the Register Pointer (one of the control registers). In the 4-bit mode, the register file is divided into nine working-register groups, each occupying 16 contiguous locations (Figure 7). The Register Pointer addresses the starting location of the active working-register group.

**Stacks.** Either the internal register file or the external data memory can be used for the stack. A 16-bit Stack Pointer (R254 and R255) is used for the external stack, which can reside anywhere in data memory between locations 4096 and 65535. An 8-bit Stack Pointer (R255) is used for the internal stack that resides within the 124 general-purpose registers (R4-R127).

### Address Spaces (Continued)

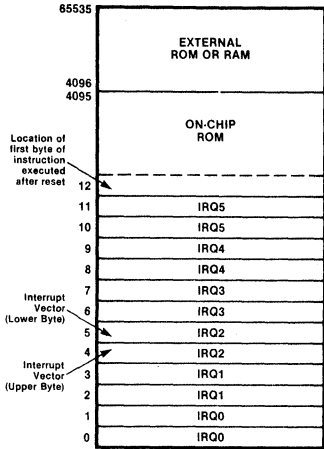


Figure 4. Program Memory Map

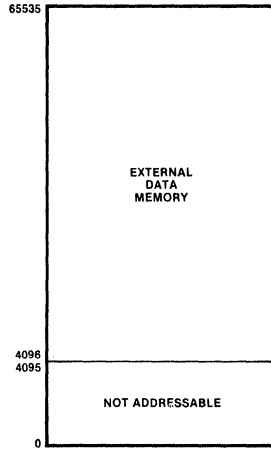


Figure 5. Data Memory Map

LOCATION	IDENTIFIERS
285	STACK POINTER (BITS 7-0) SPL
284	STACK POINTER (BITS 16-8) SPH
283	REGISTER POINTER RP
282	PROGRAM CONTROL FLAGS FLAGS
281	INTERRUPT MASK REGISTER IMR
280	INTERRUPT REQUEST REGISTER IRQ
248	INTERRUPT PRIORITY REGISTER IPR
248	PORTS 0-1 MODE P01M
247	PORT 3 MODE P3M
248	PORT 2 MODE P2M
245	T0 PRESCALER PRE0
244	TIMER/COUNTER 0 T0
243	T1 PRESCALER PRE1
242	TIMER/COUNTER 1 T1
241	TIMER MODE TMR
240	SERIAL I/O SIO
NOT IMPLEMENTED	
127	GENERAL-PURPOSE REGISTERS
4	PORT 3 P3
3	PORT 2 P2
2	PORT 1 P1
1	PORT 0 P0
0	PORT 0 P0

Figure 6. The Register File

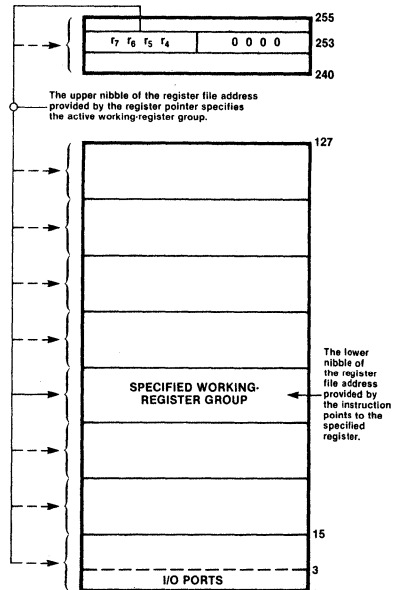


Figure 7. The Register Pointer



### Serial Input/Output

Port 3 lines P3<sub>0</sub> and P3<sub>7</sub> can be programmed as serial I/O lines for full-duplex serial asynchronous receiver/transmitter operation. The bit rate is controlled by Counter/Timer 0, with a maximum rate of 62.5K bits/second for 8.

The Z8611 automatically adds a start bit and two stop bits to transmitted data (Figure 8). Odd parity is also available as an option. Eight data bits are always transmitted,

regardless of parity selection. If parity is enabled, the eighth bit is the odd parity bit. An interrupt request (IRQ<sub>4</sub>) is generated on all transmitted characters.

Received data must have a start bit, eight data bits and at least one stop bit. If parity is on, bit 7 of the received data is replaced by a parity error flag. Received characters generate the IRQ<sub>3</sub> interrupt request.

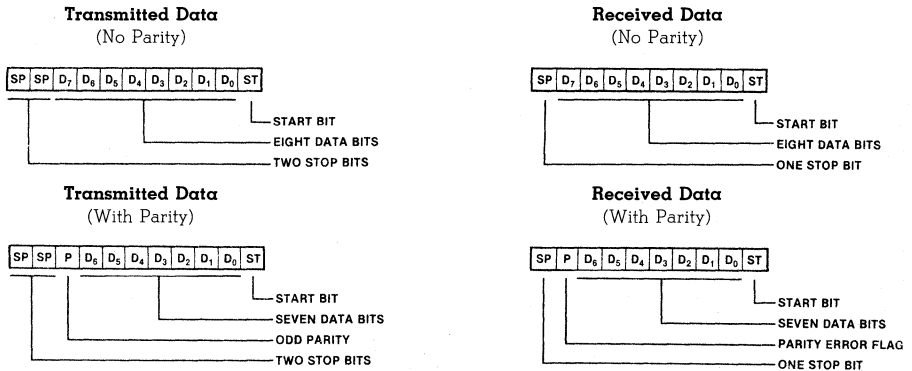


Figure 8. Serial Data Formats

### Counter/Timers

The Z8611 contains two 8-bit programmable counter/timers (T<sub>0</sub> and T<sub>1</sub>), each driven by its own 6-bit programmable prescaler. The T<sub>1</sub> prescaler can be driven by internal or external clock sources; however, the T<sub>0</sub> prescaler is driven by the internal clock only.

The 6-bit prescalers can divide the input frequency of the clock source by any number from 1 to 64. Each prescaler drives its counter, which decrements the value (1 to 256) that has been loaded into the counter. When the counter reaches the end of count, a timer interrupt request—IRQ<sub>4</sub> (T<sub>0</sub>) or IRQ<sub>5</sub> (T<sub>1</sub>)—is generated.

The counters can be started, stopped, restarted to continue, or restarted from the initial value. The counters can also be programmed to stop upon reaching zero (single-pass mode) or to automatically reload

the initial value and continue counting (modulo-n continuous mode). The counters, but not the prescalers, can be read any time without disturbing their value or count Mode.

The clock source for T<sub>1</sub> is user-definable and can be the internal microprocessor clock (4 MHz maximum for the 8 MHz device and a 6 MHz maximum for the 12 MHz device) divided by four, or an external signal input via Port 3. The Timer Mode register configures the external timer input as an external clock (1 MHz maximum), a trigger input that can be retriggerable or non-retriggerable, or as a gate input for the internal clock. The counter/timers can be programmably cascaded by connecting the T<sub>0</sub> output to the input of T<sub>1</sub>. Port 3 line P3<sub>6</sub> also serves as a timer output (T<sub>OUT</sub>) through which T<sub>0</sub>, T<sub>1</sub> or the internal clock can be output.

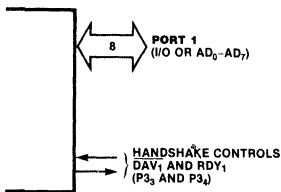
### I/O Ports

The Z8611 has 32 lines dedicated to input and output. These lines are grouped into four ports of eight lines each and are configurable as input, output or address/data. Under software control, the ports can be programmed to provide address outputs, timing, status signals, serial I/O, and parallel I/O with or without handshake. All ports have active pull-ups and pull-downs compatible with TTL loads.

**Port 1** can be programmed as a byte I/O port or as an address/data port for interfacing external memory. When used as an I/O port, Port 1 may be placed under handshake control. In this configuration, Port 3 lines P<sub>33</sub> and P<sub>34</sub> are used as the handshake controls RDY<sub>1</sub> and DAV<sub>1</sub> (Ready and Data Available).

Memory locations greater than 4096 are referenced through Port 1. To interface external memory, Port 1 must be programmed for the multiplexed Address/Data mode. If more than 256 external locations are required, Port 0 must output the additional lines.

Port 1 can be placed in the high-impedance state along with Port 0,  $\overline{AS}$ ,  $\overline{DS}$  and R/W, allowing the Z8611 to share common resources in multiprocessor and DMA applications. Data transfers can be controlled by assigning P<sub>33</sub> as a Bus Acknowledge input and P<sub>34</sub> as a Bus Request output.

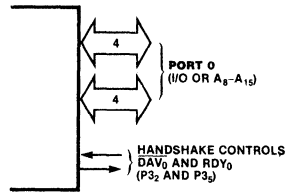


**Figure 9a. Port 1**

**Port 0** can be programmed as a nibble I/O port, or as an address port for interfacing external memory. When used as an I/O port, Port 0 may be placed under handshake control. In this configuration, Port 3 lines

P<sub>32</sub> and P<sub>35</sub> are used as the handshake controls  $\overline{DAV}_0$  and RDY<sub>0</sub>. Handshake signal assignment is dictated by the I/O direction of the upper nibble P<sub>04</sub>-P<sub>07</sub>.

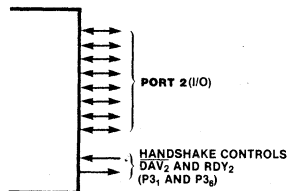
For external memory references, Port 0 can provide address bits A<sub>8</sub>-A<sub>11</sub> (lower nibble) or A<sub>8</sub>-A<sub>15</sub> (lower and upper nibble) depending on the required address space. If the address range requires 12 bits or less, the upper nibble of Port 0 can be programmed independently as I/O while the lower nibble is used for addressing. When Port 0 nibbles are defined as address bits, they can be set to the high-impedance state along with Port 1 and the control signals  $\overline{AS}$ ,  $\overline{DS}$  and R/W.



**Figure 9b. Port 0**

**Port 2** bits can be programmed independently as input or output. The port is always available for I/O operations. In addition, Port 2 can be configured to provide open-drain outputs.

Like Ports 0 and 1, Port 2 may also be placed under handshake control. In this configuration, Port 3 lines P<sub>31</sub> and P<sub>36</sub> are used as the handshake controls lines  $\overline{DAV}_2$  and RDY<sub>2</sub>. The handshake signal assignment for Port 3 lines P<sub>31</sub> and P<sub>36</sub> is dictated by the direction (input or output) assigned to bit 7 of Port 2.



**Figure 9c. Port 2**



### I/O Ports (Continued)

**Port 3** lines can be configured as I/O or control lines. In either case, the direction of the eight lines is fixed as four input (P3<sub>0</sub>-P3<sub>3</sub>) and four output (P3<sub>4</sub>-P3<sub>7</sub>). For serial I/O, lines P3<sub>0</sub> and P3<sub>7</sub> are programmed as serial in and serial out respectively.

Port 3 can also provide the following control functions: handshake for Ports 0, 1 and 2 (D<sub>AV</sub> and R<sub>DY</sub>); four external interrupt request signals (I<sub>RQ0</sub>-I<sub>RQ3</sub>); timer input and output signals (T<sub>IN</sub> and T<sub>OUT</sub>) and Data Memory Select (D<sub>M</sub>).

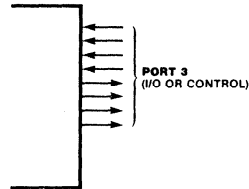


Figure 9d. Port 3

### Interrupts

The Z8611 allows six different interrupts from eight sources: the four Port 3 lines P3<sub>0</sub>-P3<sub>3</sub>, Serial In, Serial Out, and the two counter/timers. These interrupts are both maskable and prioritized. The Interrupt Mask register globally or individually enables or disables the six interrupt requests. When more than one interrupt is pending, priorities are resolved by a programmable priority encoder that is controlled by the Interrupt Priority register.

All Z8611 interrupts are vectored. When an interrupt request is granted, and interrupt machine cycle is entered. This disables all

subsequent interrupts, saves the Program Counter and status flags, and branches to the program memory vector location reserved for that interrupt. This memory location and the next byte contain the 16-bit address of the interrupt service routine for that particular interrupt request.

Polled interrupt systems are also supported. To accommodate a polled structure, any or all of the interrupt inputs can be masked and the Interrupt Request register polled to determine which of the interrupt requests needs service.

### Clock

The on-chip oscillator has a high-gain, parallel-resonant amplifier for connection to a crystal or to any suitable external clock source (XTAL1 = Input, XTAL2 = Output).

The crystal source is connected across XTAL1 and XTAL2, using the recommended

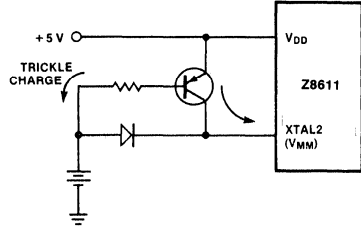
capacitors ( $C_1 \leq 15 \text{ pF}$ ) from each pin to ground. The specifications for the crystal are as follows:

- AT cut, parallel resonant
- Fundamental types, 8/12 MHz maximum
- Series resistance,  $R_s \leq 100 \Omega$ .

### Power Down Standby Option

The low-power standby mode allows power to be removed without losing the contents of the 124 general-purpose registers. This mode is available to the user as a bonding option whereby pin 2 (normally XTAL2) is replaced by the  $V_{MM}$  (standby) power supply input. This necessitates the use of an external clock generator (input = XTAL1) rather than a crystal source.

The removal of power, whether intended or due to power failure, must be preceded by a software routine that stores the appropriate status into the register file. Figure 10 shows the recommended circuit for a battery back-up supply system.



**Figure 10. Recommended Driver Circuit for Power Down Operation**

### Instruction Set Notation

**Addressing Modes.** The following notation is used to describe the addressing modes and instruction operations as shown in the instruction summary.

- IRR** Indirect register pair + indirect working-register pair address
- Irr** Indirect working-register pair only
- X** Indexed address
- DA** Direct address
- RA** Relative address
- IM** Immediate
- R** Register or working-register address
- r** Working-register address only
- IR** Indirect-register or indirect working-register address
- Irr** Indirect working-register address only
- RR** Register pair or working register pair address

**Symbols.** The following symbols are used in describing the instruction set.

- dst** Destination location or contents
- src** Source location or contents
- cc** Condition code (see list)
- @** Indirect address prefix
- SP** Stack pointer (control registers 254-255)
- PC** Program counter
- FLAGS** Flag register (control register 252)
- RP** Register pointer (control register 253)

**IMR** Interrupt mask register (control register 251)

Assignment of a value is indicated by the symbol " $\leftarrow$ ". For example,

$$dst \leftarrow dst + src$$

indicates that the source data is added to the destination data and the result is stored in the destination location. The notation "addr(n)" is used to refer to bit "n" of a given location. For example,

$$dst(7)$$

refers to bit 7 of the destination operand.

**Flags.** Control Register R252 contains the following six flags:

<b>C</b>	Carry flag	b7	C	Z	S	V	D	H	F2	F1	b0
<b>Z</b>	Zero flag										
<b>S</b>	Sign flag										
<b>V</b>	Overflow flag										
<b>D</b>	Decimal-adjust flag	<b>F1</b>									
<b>H</b>	Half-carry flag	<b>F2</b>									

} user flags

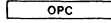
Affected flags are indicated by:

- 0** Cleared to zero
- 1** Set to one
- \*** Set or cleared according to operation
- Unaffected
- X** Undefined

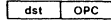
**Condition Codes**

<b>Value</b>	<b>Mnemonic</b>	<b>Meaning</b>	<b>Flags Set</b>
1000		Always true	...
0111	C	Carry	C=1
1111	NC	No carry	C=0
0110	Z	Zero	Z=1
1110	NZ	Not zero	Z=0
1101	PL	Plus	S=0
0101	MI	Minus	S=1
0100	OV	Overflow	V=1
1100	NOV	No overflow	V=0
0110	EQ	Equal	Z=1
1110	NE	Not equal	Z=0
1001	GE	Greater than or equal	(S XOR V)=0
0001	LT	Less than	(S XOR V)=1
1010	GT	Greater than	[Z OR (S XOR V)]=0
0010	LE	Less than or equal	[Z OR (S XOR V)]=1
1111	UGE	Unsigned greater than or equal	C=0
0111	ULT	Unsigned less than	C=1
1011	UGT	Unsigned greater than	(C=0 AND Z=0)=1
0011	ULE	Unsigned less than or equal	(C OR Z)=1
0000		Never true	...

### Instruction Formats

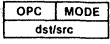


CCF, DI, EI, IRET, NOP,  
RCF, RET, SCF



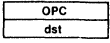
INC r

### One-Byte Instruction



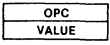
OR 1 1 1 0 dst/src

CLR, CPL, DA, DEC,  
DECW, INC, INCW, POP,  
PUSH, RL, RLC, RR,  
RRC, SRA, SWAP

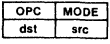


OR 1 1 1 0 dst

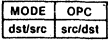
JP, CALL (Indirect)



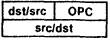
SRP



ADC, ADD, AND,  
CP, OR, SBC, SUB,  
TCM, TM, XOR

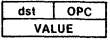


LD, LDE, LDEI,  
LDC, LDCI

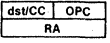


OR 1 1 1 0 src

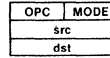
LD



LD

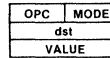


DJNZ, JR



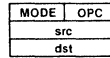
OR 1 1 1 0 src

ADC, ADD, AND, CP,  
LD, OR, SBC, SUB,  
TCM, TM, XOR



OR 1 1 1 0 dst

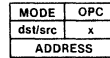
ADC, ADD, AND, CP,  
LD, OR, SBC, SUB,  
TCM, TM, XOR



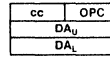
OR 1 1 1 0 src

LD

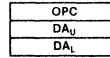
OR 1 1 1 0 dst



LD



JP



CALL

### Two-Byte Instruction






### Three-Byte Instruction

Figure 11. Instruction Formats




**Instruction Summary**

Instruction and Operation	Addr Mode		Opcode Byte (Hex)	Flags Affected					
	dst	src		C	Z	S	V	D	H
<b>ADC</b> dst,src dst - dst + src + C	(Note 1)		1□	*	*	*	*	0	*
<b>ADD</b> dst,src dst - dst + src	(Note 1)		0□	*	*	*	*	0	*
<b>AND</b> dst,src dst - dst AND src	(Note 1)		5□	-	*	*	0	-	-
<b>CALL</b> dst SP - SP - 2 @SP - PC; PC - dst	DA IRR		D6 D4	-	-	-	-	-	-
<b>CCF</b> C - NOT C			EF	*	-	-	-	-	-
<b>CLR</b> dst dst - 0	R IR		B0 B1	-	-	-	-	-	-
<b>COM</b> dst dst - NOT dst	R IR		60 61	-	*	*	0	-	-
<b>CP</b> dst,src dst - src	(Note 1)		A□	*	*	*	*	-	-
<b>DA</b> dst dst - DA dst	R IR		40 41	*	*	*	X	-	-
<b>DEC</b> dst dst - dst - 1	R IR		00 01	-	*	*	*	-	-
<b>DECW</b> dst dst - dst - 1	RR IR		80 81	-	*	*	*	-	-
<b>DI</b> IMR (7) - 0			8F	-	-	-	-	-	-
<b>DINZ</b> r,dst r - r - 1 if r ≠ 0 PC - PC + dst Range: +127, -128	RA		rA r=0-F	-	-	-	-	-	-
<b>EI</b> IMR (7) - 1			9F	-	-	-	-	-	-
<b>INC</b> dst dst - dst + 1	r R IR		rE r=0-F 20 21	-	*	*	*	-	-
<b>INCW</b> dst dst - dst + 1	RR IR		A0 A1	-	*	*	*	-	-
<b>IRET</b> FLAGS - @SP; SP - SP + 1 PC - @SP; SP - SP + 2; IMR (7) - 1			BF	*	*	*	*	*	*
<b>JP</b> cc,dst if cc is true PC - dst	DA IRR		cD c=0-F 30	-	-	-	-	-	-
<b>JR</b> cc,dst if cc is true, PC - PC + dst Range: +127, -128	RA		cB c=0-F	-	-	-	-	-	-

Instruction and Operation	Addr Mode		Opcode Byte (Hex)	Flags Affected					
	dst	src		C	Z	S	V	D	H
<b>LD</b> dst,src dst - src	r R R	Im R r	rC r8 r9 r=0-F	-	-	-	-	-	-
	r X r R R R R IR	X r Ir r R R Im Im R	C7 D7 E3 F3 E4 E5 E6 E7 F5						
<b>LDC</b> dst,src dst - src	r Irr	Irr r	C2 D2	-	-	-	-	-	-
<b>LDCI</b> dst,src dst - src r - r + 1; rr - rr + 1	Ir Irr	Irr Ir	C3 D3	-	-	-	-	-	-
<b>LDE</b> dst,src dst - src	r Irr	Irr r	82 92	-	-	-	-	-	-
<b>LDEI</b> dst,src dst - src r - r + 1; rr - rr + 1	Ir Irr	Irr Ir	83 93	-	-	-	-	-	-
<b>NOP</b>			FF	-	-	-	-	-	-
<b>OR</b> dst,src dst - dst OR src	(Note 1)		4□	-	*	*	0	-	-
<b>POP</b> dst dst - @SP SP - SP + 1	R IR		50 51	-	-	-	-	-	-
<b>PUSH</b> src SP - SP - 1; @SP - src	R IR		70 71	-	-	-	-	-	-
<b>RCF</b> C - 0			CF	0	-	-	-	-	-
<b>RET</b> PC - @SP; SP - SP + 2			AF	-	-	-	-	-	-
<b>RL</b> dst 	R IR		90 91	*	*	*	*	-	-
<b>RLC</b> dst 	R IR		10 11	*	*	*	*	-	-
<b>RR</b> dst 	R IR		E0 E1	*	*	*	*	-	-
<b>RRC</b> dst 	R IR		C0 C1	*	*	*	*	-	-
<b>SBC</b> dst,src dst - dst - src - C	(Note 1)		3□	*	*	*	*	1	*
<b>SCF</b> C - 1			DF	1	-	-	-	-	-
<b>SRA</b> dst 	R IR		D0 D1	*	*	*	0	-	-



Instruction Summary (Continued)

Instruction and Operation	Addr Mode		Opcode Byte (Hex)	Flags Affected						
	dst	src		C	Z	S	V	D	H	
<b>SRP</b> src RP - src		Im	31	-	-	-	-	-	-	-
<b>SUB</b> dst,src dst - dst - src		(Note 1)	2□	*	*	*	*	1	*	
<b>SWAP</b> dst		R IR	F0 F1	X	*	*	X	-	-	

Instruction and Operation	Addr Mode		Opcode Byte (Hex)	Flags Affected						
	dst	src		C	Z	S	V	D	H	
<b>TCM</b> dst,src (NOT dst) AND src		(Note 1)	6□	-	*	*	0	-	-	
<b>TM</b> dst, src dst AND src		(Note 1)	7□	-	*	*	0	-	-	
<b>XOR</b> 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 is found in the instruction set table above. The second nibble is expressed symbolically by a □ in this table, and its value is found in the following table to the left of the applicable addressing mode pair.

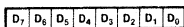
For example, to determine the opcode of an ADC instruction using the addressing modes r (destination) and Ir (source) is 13.

Addr Mode		Lower Opcode Nibble
dst	src	
r	r	2
r	Ir	3
R	R	4
R	IR	5
R	IM	6
IR	IM	7



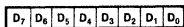
Registers

**R240 SIO**  
Serial I/O Register  
(F0H; Read/Write)



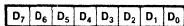
SERIAL DATA (D<sub>0</sub> = LSB)

**R244 T0**  
Counter/Timer 0 Register  
(F4H; Read/Write)



T<sub>0</sub> INITIAL VALUE (WHEN WRITTEN)  
(RANGE: 1-256 DECIMAL 01-00 HEX)  
T<sub>0</sub> CURRENT VALUE (WHEN READ)

**R241 TMR**  
Timer Mode Register  
(F1H; Read/Write)



**T<sub>OUT</sub> MODES**  
NOT USED = 00  
T<sub>0</sub> OUT = 01  
T<sub>1</sub> OUT = 10  
INTERNAL CLOCK OUT = 11

**T<sub>IN</sub> MODES**  
EXTERNAL CLOCK INPUT = 00  
GATE INPUT = 01  
TRIGGER INPUT (NON-RETRIGGERABLE) = 10  
TRIGGER INPUT (RETRIGGERABLE) = 11

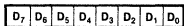
0 = NO FUNCTION  
1 = LOAD T<sub>0</sub>

0 = DISABLE T<sub>0</sub> COUNT  
1 = ENABLE T<sub>0</sub> COUNT

0 = NO FUNCTION  
1 = LOAD T<sub>1</sub>

0 = DISABLE T<sub>1</sub> COUNT  
1 = ENABLE T<sub>1</sub> COUNT

**R245 PRE0**  
Prescaler 0 Register  
(F5H; Write Only)

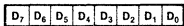


COUNT MODE  
0 = T<sub>0</sub> SINGLE-PASS  
1 = T<sub>0</sub> MODULO-N

RESERVED

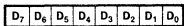
PRESCALER MODULO  
(RANGE: 1-64 DECIMAL 01-00 HEX)

**R242 T1**  
Counter Timer 1 Register  
(F2H; Read/Write)



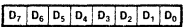
T<sub>1</sub> INITIAL VALUE (WHEN WRITTEN)  
(RANGE 1-256 DECIMAL 01-00 HEX)  
T<sub>1</sub> CURRENT VALUE (WHEN READ)

**R246 P2M**  
Port 2 Mode Register  
(F6H; Write Only)



P<sub>2</sub>-P<sub>2</sub>, I/O DEFINITION  
0 DEFINES BIT AS OUTPUT  
1 DEFINES BIT AS INPUT

**R243 PRE1**  
Prescaler 1 Register  
(F3H; Write Only)

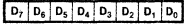


COUNT MODE  
0 = T<sub>1</sub> SINGLE-PASS  
1 = T<sub>1</sub> MODULO-N

CLOCK SOURCE  
1 = T<sub>1</sub> INTERNAL  
0 = T<sub>1</sub> EXTERNAL TIMING INPUT (T<sub>IN</sub>) MODE

PRESCALER MODULO  
(RANGE: 1-64 DECIMAL 01-00 HEX)

**R247 P3M**  
Port 3 Mode Register  
(F7H; Write Only)



0 PORT 2 PULL-UPS OPEN DRAIN  
1 PORT 2 PULL-UPS ACTIVE

RESERVED

0 P32 = INPUT P35 = OUTPUT  
1 P32 =  $\overline{\text{DAV0}}/\text{RDY0}$  P35 =  $\text{RDY0}/\overline{\text{DAV0}}$

0 P33 = INPUT P34 = OUTPUT  
1 P33 = INPUT P34 =  $\overline{\text{DM}}$

0 P31 = INPUT (T<sub>IN</sub>) P36 = OUTPUT (T<sub>OUT</sub>)  
1 P31 =  $\overline{\text{DAV1}}/\text{RDY1}$  P36 =  $\text{RDY1}/\overline{\text{DAV1}}$

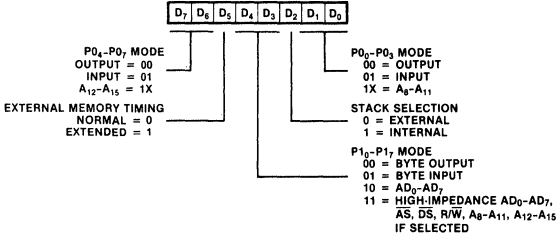
0 P30 = INPUT P37 = OUTPUT  
1 P30 = SERIAL IN P37 = SERIAL OUT

0 PARITY OFF  
1 PARITY ON

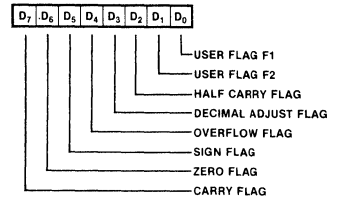
Figure 12. Control Registers

**Registers (Continued)**

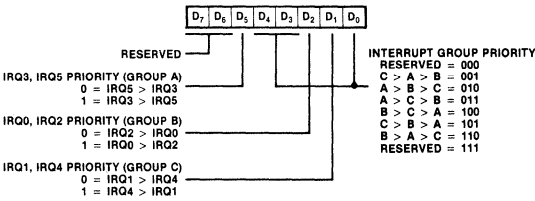
**R248 P01M**  
Port 0 and 1 Mode Register  
(F8<sub>H</sub>; Write Only)



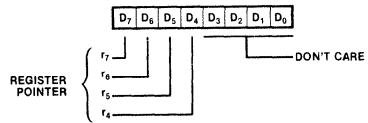
**R252 FLAGS**  
Flag Register  
(FC<sub>H</sub>; Read/Write)



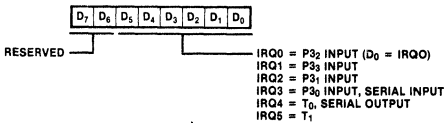
**R249 IPR**  
Interrupt Priority Register  
(F9<sub>H</sub>; Write Only)



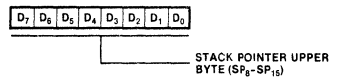
**R253 RP**  
Register Pointer  
(FD<sub>H</sub>; Read/Write)



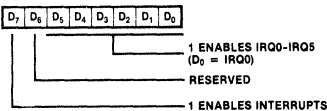
**R250 IRQ**  
Interrupt Request Register  
(FA<sub>H</sub>; Read/Write)



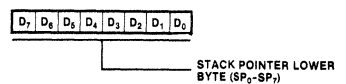
**R254 SPH**  
Stack Pointer  
(FE<sub>H</sub>; Read/Write)



**R251 IMR**  
Interrupt Mask Register  
(FB<sub>H</sub>; Read/Write)



**R255 SPL**  
Stack Pointer  
(FF<sub>H</sub>; Read/Write)§



**Figure 12a. Control Registers (Continued)**



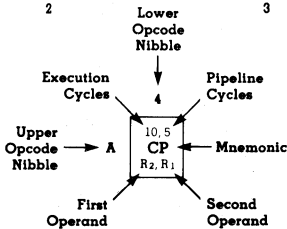
Z8611/L

Opcode Map

Lower Nibble (Hex)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
Upper Nibble (Hex)	0	6,5 DEC R <sub>1</sub>	6,5 DEC IR <sub>1</sub>	6,5 ADD r <sub>1</sub> , r <sub>2</sub>	6,5 ADD r <sub>1</sub> , IR <sub>2</sub>	10,5 ADD R <sub>2</sub> , R <sub>1</sub>	10,5 ADD IR <sub>2</sub> , R <sub>1</sub>	10,5 ADD R <sub>1</sub> , IM	10,5 ADD R <sub>1</sub> , IM	6,5 LD r <sub>1</sub> , R <sub>2</sub>	6,5 LD r <sub>2</sub> , R <sub>1</sub>	12/10,5 DJNZ r <sub>1</sub> , RA	12/10,0 JR cc, RA	6,5 LD r <sub>1</sub> , IM	12/10,0 JP cc, DA	6,5 INC r <sub>1</sub>	
	1	6,5 RLC R <sub>1</sub>	6,5 RLC IR <sub>1</sub>	6,5 ADC r <sub>1</sub> , r <sub>2</sub>	6,5 ADC r <sub>1</sub> , IR <sub>2</sub>	10,5 ADC R <sub>2</sub> , R <sub>1</sub>	10,5 ADC IR <sub>2</sub> , R <sub>1</sub>	10,5 ADC R <sub>1</sub> , IM	10,5 ADC R <sub>1</sub> , IM								
	2	6,5 INC R <sub>1</sub>	6,5 INC IR <sub>1</sub>	6,5 SUB r <sub>1</sub> , r <sub>2</sub>	6,5 SUB r <sub>1</sub> , IR <sub>2</sub>	10,5 SUB R <sub>2</sub> , R <sub>1</sub>	10,5 SUB R <sub>2</sub> , R <sub>1</sub>	10,5 SUB R <sub>1</sub> , IM	10,5 SUB R <sub>1</sub> , IM								
	3	8,0 JP IRR <sub>1</sub>	6,1 SRP IM	6,5 SBC r <sub>1</sub> , r <sub>2</sub>	6,5 SBC r <sub>1</sub> , IR <sub>2</sub>	10,5 SBC R <sub>2</sub> , R <sub>1</sub>	10,5 SBC IR <sub>2</sub> , R <sub>1</sub>	10,5 SBC R <sub>1</sub> , IM	10,5 SBC R <sub>1</sub> , IM								
	4	8,5 DA R <sub>1</sub>	8,5 DA IR <sub>1</sub>	6,5 OR r <sub>1</sub> , r <sub>2</sub>	6,5 OR r <sub>1</sub> , IR <sub>2</sub>	10,5 OR R <sub>2</sub> , R <sub>1</sub>	10,5 OR R <sub>2</sub> , R <sub>1</sub>	10,5 OR R <sub>1</sub> , IM	10,5 OR R <sub>1</sub> , IM								
	5	10,5 POP R <sub>1</sub>	10,5 POP IR <sub>1</sub>	6,5 AND r <sub>1</sub> , r <sub>2</sub>	6,5 AND r <sub>1</sub> , IR <sub>2</sub>	10,5 AND R <sub>2</sub> , R <sub>1</sub>	10,5 AND R <sub>2</sub> , R <sub>1</sub>	10,5 AND R <sub>1</sub> , IM	10,5 AND R <sub>1</sub> , IM								
	6	6,5 COM R <sub>1</sub>	6,5 COM IR <sub>1</sub>	6,5 TCM r <sub>1</sub> , r <sub>2</sub>	6,5 TCM r <sub>1</sub> , IR <sub>2</sub>	10,5 TCM R <sub>2</sub> , R <sub>1</sub>	10,5 TCM R <sub>2</sub> , R <sub>1</sub>	10,5 TCM R <sub>1</sub> , IM	10,5 TCM R <sub>1</sub> , IM								
	7	10/12,1 PUSH R <sub>2</sub>	12/14,1 PUSH IR <sub>2</sub>	6,5 TM r <sub>1</sub> , r <sub>2</sub>	6,5 TM r <sub>1</sub> , IR <sub>2</sub>	10,5 TM R <sub>2</sub> , R <sub>1</sub>	10,5 TM R <sub>2</sub> , R <sub>1</sub>	10,5 TM R <sub>1</sub> , IM	10,5 TM R <sub>1</sub> , IM								
	8	10,5 DECW RR <sub>1</sub>	10,5 DECW IR <sub>1</sub>	12,0 LDE r <sub>1</sub> , IRR <sub>2</sub>	18,0 LDEI IR <sub>1</sub> , IRR <sub>2</sub>												6,1 DI
	9	6,5 RL R <sub>1</sub>	6,5 RL IR <sub>1</sub>	12,0 LDE r <sub>2</sub> , IRR <sub>1</sub>	18,0 LDEI IR <sub>2</sub> , IRR <sub>1</sub>												6,1 EI
	A	10,5 INCW RR <sub>1</sub>	10,5 INCW IR <sub>1</sub>	6,5 CP r <sub>1</sub> , r <sub>2</sub>	6,5 CP r <sub>1</sub> , IR <sub>2</sub>	10,5 CP R <sub>2</sub> , R <sub>1</sub>	10,5 CP R <sub>2</sub> , R <sub>1</sub>	10,5 CP R <sub>1</sub> , IM	10,5 CP R <sub>1</sub> , IM								14,0 RET
	B	6,5 CLR R <sub>1</sub>	6,5 CLR IR <sub>1</sub>	6,5 XOR r <sub>1</sub> , r <sub>2</sub>	6,5 XOR r <sub>1</sub> , IR <sub>2</sub>	10,5 XOR R <sub>2</sub> , R <sub>1</sub>	10,5 XOR R <sub>2</sub> , R <sub>1</sub>	10,5 XOR R <sub>1</sub> , IM	10,5 XOR R <sub>1</sub> , IM								16,0 IRET
	C	6,5 RRC R <sub>1</sub>	6,5 RRC IR <sub>1</sub>	12,0 LDC r <sub>1</sub> , IRR <sub>2</sub>	18,0 LDCI IR <sub>1</sub> , IRR <sub>2</sub>				10,5 LD r <sub>1</sub> , x, R <sub>2</sub>								6,5 RCF
	D	6,5 SRA R <sub>1</sub>	6,5 SRA IR <sub>1</sub>	12,0 LDC r <sub>2</sub> , IRR <sub>1</sub>	18,0 LDCI IR <sub>2</sub> , IRR <sub>1</sub>	20,0 CALL* IRR <sub>1</sub>		20,0 CALL DA	10,5 LD r <sub>2</sub> , x, R <sub>1</sub>								6,5 SCF
	E	6,5 RR R <sub>1</sub>	6,5 RR IR <sub>1</sub>	6,5 LD r <sub>1</sub> , IR <sub>2</sub>	10,5 LD r <sub>1</sub> , IR <sub>2</sub>	10,5 LD R <sub>2</sub> , R <sub>1</sub>	10,5 LD IR <sub>2</sub> , R <sub>1</sub>	10,5 LD R <sub>1</sub> , IM	10,5 LD R <sub>1</sub> , IM								6,5 CCF
	F	6,7 SWAP R <sub>1</sub>	6,7 SWAP IR <sub>1</sub>		6,5 LD IR <sub>1</sub> , r <sub>2</sub>		10,5 LD R <sub>2</sub> , IR <sub>1</sub>										6,0 NOP

Bytes per Instruction



Legend:

R = 8-Bit Address  
r = 4-Bit Address  
R<sub>1</sub> or r<sub>1</sub> = Dest Address  
R<sub>2</sub> or r<sub>2</sub> = Src Address

Sequence:

Opcode, First Operand, Second Operand

Note: The blank areas are not defined.

\*2-byte instruction; fetch cycle appears as a 3-byte instruction



### Absolute Maximum Ratings

Voltages on all pins with respect to GND ..... -0.3 V to +7.0 V

Operating Ambient Temperature ..... 0°C to +70°C

Storage Temperature ..... -65°C to +150°C

Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

### Standard Test Conditions

The characteristics below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND. Positive current flows into the reference pin. Standard conditions are as follows:

- +4.75 V ≤ V<sub>CC</sub> ≤ +5.25 V
- GND = 0 V
- 0°C ≤ T<sub>A</sub> ≤ +70°C

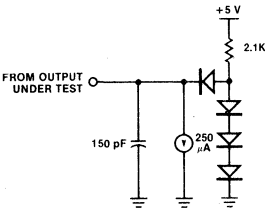


Figure 13. Test Load 1

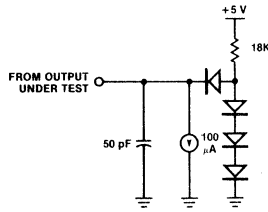


Figure 14. Test Load 2

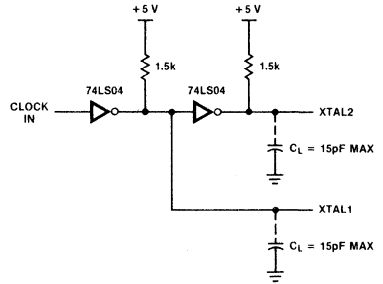


Figure 15. External Clock Interface Circuit  
(Both the clock and its complement are required)

**DC Characteristics**

Symbol	Parameter	Min	Max	Unit	Condition
V <sub>CH</sub>	Clock Input High Voltage	3.8	V <sub>CC</sub>	V	Driven by External Clock Generator
V <sub>CL</sub>	Clock Input Low Voltage	-0.3	0.8	V	Driven by External Clock Generator
V <sub>IH</sub>	Input High Voltage	2.0	V <sub>CC</sub>	V	
V <sub>IL</sub>	Input Low Voltage	-0.3	0.8	V	
V <sub>RH</sub>	Reset Input High Voltage	3.8	V <sub>CC</sub>	V	
V <sub>RL</sub>	Reset Input Low Voltage	-0.3	0.8	V	
V <sub>OH</sub>	Output High Voltage	2.4		V	I <sub>OH</sub> = -250 $\mu$ A
V <sub>OL</sub>	Output Low Voltage		0.4	V	I <sub>OL</sub> = +2.0 mA
I <sub>IL</sub>	Input Leakage	-10	10	$\mu$ A	0 V $\leq$ V <sub>IN</sub> $\leq$ +5.25 V
I <sub>OL</sub>	Output Leakage	-10	10	$\mu$ A	0 V $\leq$ V <sub>IN</sub> $\leq$ +5.25 V
I <sub>IR</sub>	Reset Input Current		-50	$\mu$ A	V <sub>CC</sub> = +5.25 V, V <sub>RL</sub> = 0 V
I <sub>CC</sub>	V <sub>CC</sub> Supply Current		120 80*	mA	
I <sub>MM</sub>	V <sub>MM</sub> Supply Current		10	mA	Power Down Mode
V <sub>MM</sub>	Backup Supply Voltage	3	V <sub>CC</sub>	V	Power Down

\* This value is for Z8611L only



External I/O or Memory Read and Write Timing

No	Symbol	Parameter	Z8611/L		Z8611A		Notes*†
			Min	Max	Min	Max	
1	TdA(AS)	Address Valid to $\overline{AS}$ $\uparrow$ Delay	50			360	1,2,3
2	TdAS(A)	$\overline{AS}$ $\uparrow$ to Address Float Delay	70		45		1,2,3
3	TdAS(DR)	$\overline{AS}$ $\uparrow$ to Read Data Required Valid		360		220	1,2,3,4
4	TwAS	$\overline{AS}$ Low Width	80		55		1,2,3
5	TdAz(DS)	Address Float to $\overline{DS}$ $\downarrow$	0		0		1
6	TwDSR	$\overline{DS}$ (Read) Low Width	250		185		1,2,3,4
7	TwDSW	$\overline{DS}$ (Write) Low Width	160		110		1,2,3,4
8	TdDSR(DR)	$\overline{DS}$ $\downarrow$ to Read Data Required Valid		200		130	1,2,3,4
9	ThDR(DS)	Read Data to $\overline{DS}$ $\downarrow$ Hold Time	0		0		1
10	TdDS(A)	$\overline{DS}$ $\uparrow$ to Address Active Delay	70		45		1,2,3
11	TdDS(AS)	$\overline{DS}$ $\uparrow$ to $\overline{AS}$ $\downarrow$ Delay	70		55		1,2,3
12	TdR/W(AS)	R/W Valid to $\overline{AS}$ $\uparrow$ Delay	50		30		1,2,3
13	TdDS(R/W)	$\overline{DS}$ $\uparrow$ to R/W Not Valid	60		35		1,2,3
14	TdDW(DSW)	Write Data Valid to $\overline{DS}$ (Write) $\downarrow$ Delay	50		35		1,2,3
15	TdDS(DW)	$\overline{DS}$ $\uparrow$ to Write Data Not Valid Delay	70		45		1,2,3
16	TdA(DR)	Address Valid to Read Data Required Valid		410		255	1,2,3,4
17	TdAS(DS)	$\overline{AS}$ $\uparrow$ to $\overline{DS}$ $\downarrow$ Delay	80		55		1,2,3

NOTES:

1. Test Load 1
2. Timing numbers given are for minimum T<sub>pC</sub>.
3. Also see clock cycle time dependent characteristics table.
4. When using extended memory timing add 2 T<sub>pC</sub>.
5. All timing reference use 2.0 V for a logic "1" and 0.8 V for a logic "0".
- \* All units in nanoseconds (ns).
- † Timings are preliminary and subject to change.

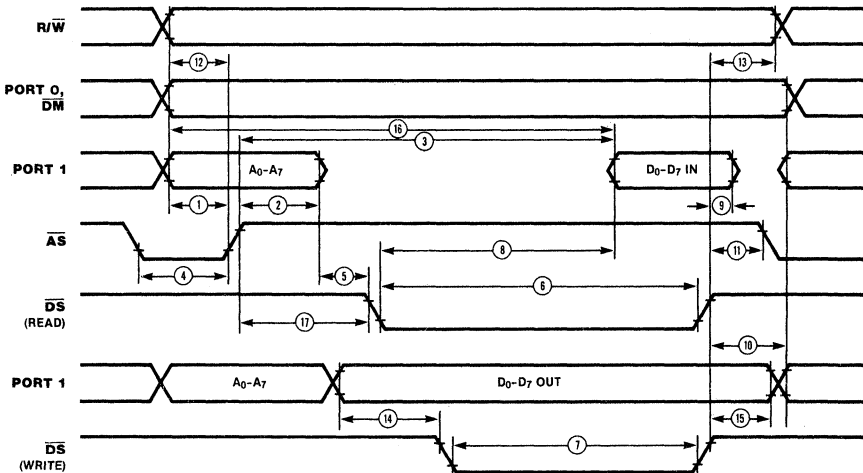


Figure 16. External I/O or Memory Read/Write



**Additional Timing Table**

No	Symbol	Parameter	Z8611/L		Z8611A		Notes*†
			Min	Max	Min	Max	
1	TpC	Input Clock Period	125	1000	83	1000	1
2	TrC, TfC	Clock Input Rise And Fall Times		25		15	1
3	TwC	Input Clock Width	37		26		1
4	TwTinL	Timer Input Low Width	100		70		2
5	TwTinH	Timer Input High Width	3TpC		3TpC		2
6	TpTin	Timer Input Period	8TpC		8TpC		2
7	TrTin, TfTin	Timer Input Rise And Fall Times		100		100	2
8a	TwIL	Interrupt Request Input Low Time	100		70		2,3
8b	TwIH	Interrupt Request Input Low Time	3TpC		3TpC		2,4
9	TwIH	Interrupt Request Input High Time	3TpC		3TpC		2,3

**NOTES:**

1. Clock timing references uses 3.8 V for a logic "1" and 0.8 V for a logic "0".

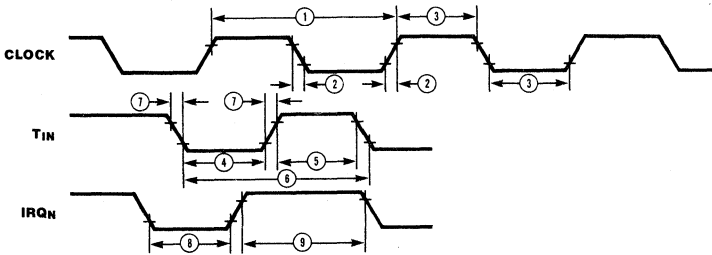
2. Timing reference uses 2.0 V for a logic "1" and 0.8 V for a logic "0".

3. Interrupt request via Port 3 (P3<sub>1</sub>-P3<sub>3</sub>).

4. Interrupt request via Port 3 (P3<sub>0</sub>).

\* Units in nanoseconds (ns).

† Timings are preliminary and subject to change.


**Figure 17. Additional Timing**

### Handshake Timing

No	Symbol	Parameter	Z8611/L		Z8611A		Notes*†
			Min	Max	Min	Max	
1	TsDI(DAV)	Data In Setup Time	0		0		
2	ThDI(DAV)	Data In Hold Time	230		160		
3	TwDAV	Data Available Width	175		120		
4	TdDAV <sub>I</sub> (RDY)	$\overline{DAV}$ ↓ Input to RDY ↓ Delay		175		120	1,2
5	TdDAV <sub>O</sub> (RDY)	$\overline{DAV}$ ↓ Output to RDY ↓ Delay	0		0		1,3
6	TdDAV <sub>I</sub> r(RDY)	$\overline{DAV}$ ↑ Input to RDY ↑ Delay		175		120	1,2
7	TdDAV <sub>O</sub> r(RDY)	$\overline{DAV}$ ↑ Output to RDY ↑ Delay	0		0		1,3
8	TdDO(DAV)	Data Out to $\overline{DAV}$ ↓ Delay	50		30		1
9	TdRDY(DAV)	Rdy ↓ Input to $\overline{DAV}$ ↑ Delay	0	200	0	140	1

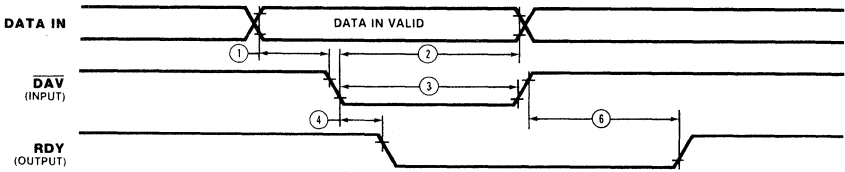
**NOTES:**

1. Test Load 1
2. Input handshake
3. Output handshake

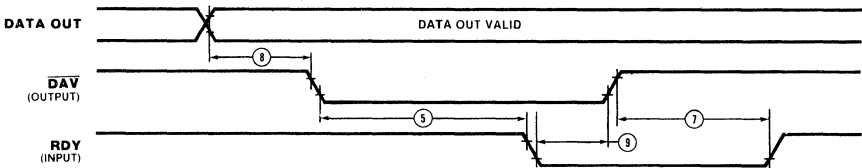
4. All timing references use 2.0 V for a logic "1" and 0.8 V for a logic "0".

\* Units in nanoseconds (ns).

† Timings are preliminary and subject to change.



**Figure 18a. Input Handshake**



**Figure 18b. Output Handshake**



# Z8611/L

## Clock-Cycle-Time-Dependent Characteristics

Number	Symbol	Z8611/L Equation	Z8611A Equation
1	TdA(AS)	TpC-75	TpC-50
2	TdAS(A)	TpC-55	TpC-40
3	TdAS(DR)	4TpC-140*	4TpC-110*
4	TwAS	TpC-45	TpC-30
6	TwDSR	3TpC-125*	3TpC-65*
7	TwDSW	2TpC-90*	2TpC-55*
8	TdDSR(DR)	3TpC-175*	3TpC-120*
10	Td(DS)A	TpC-55	TpC-40
11	TdDS(AS)	TpC-55	TpC-30
12	TdR/W(AS)	TpC-75	TpC-55
13	TdDS(R/W)	TpC-65	TpC-50
14	TdDW(DSW)	TpC-75	TpC-50
15	TdDS(DW)	TpC-55	TpC-40
16	TdA(DR)	5TpC-215*	5TpC-160*
17	TdAS(DS)	TpC-45	TpC-30

\* Add 2TpC when using extended memory timing.

## Ordering Information

Type	Package	Temp.	Clock	Description
Z8611 B1	Plastic	0/ +70°C	8 MHz	4K ROM Microcomputer
Z8611 B6	Plastic	-40/ +85°C		
Z8611 D1	Ceramic	0/ +70°C		
Z8611 D6	Ceramic	-40/ +85°C		
Z8611 C1	Plastic Chip Carrier	0/ +70°C		
Z8611 C6	Plastic Chip Carrier	-40/ +85°C		
Z8611 K1	Ceramic Chip Carrier	0/ +70°C		
Z8611 K6	Ceramic Chip Carrier	-40/ +85°C		
Z8611A B1	Plastic	0/ +70°C	12 MHz	4K ROM Microcomputer
Z8611A B6	Plastic	-40/ +85°C		
Z8611A D1	Ceramic	0/ +70°C		
Z8611A D6	Ceramic	-40/ +85°C		
Z8611A C1	Plastic Chip Carrier	0/ +70°C		
Z8611A C6	Plastic Chip Carrier	-40/ +85°C		
Z8611A K1	Ceramic Chip Carrier	0/ +70°C		
Z8611A K6	Ceramic Chip Carrier	-40/ +85°C		
Z8611L B1	Plastic	0/ +70°C	8 MHz	4K ROM Microcomputer Low Power version
Z8611L B6	Plastic	-40/ +85°C		
Z8611L D1	Ceramic	0/ +70°C		
Z8611L D6	Ceramic	-40/ +85°C		
Z8611L C1	Plastic Chip Carrier	0/ +70°C		
Z8611L C6	Plastic Chip Carrier	-40/ +85°C		
Z8611L K1	Ceramic Chip Carrier	0/ +70°C		
Z8611L K6	Ceramic Chip Carrier	-40/ +85°C		



# Z8621/L

## Z8 8K ROM Microcomputer

- Complete microcomputer, 8K bytes of ROM, 240 bytes of RAM, 32 I/O lines, and up to 56K bytes addressable external space each for program and data memory.
- 256-byte register file, including 236 general-purpose registers, four I/O port registers, and 16 status and control registers.
- Minimum instruction execution time 1  $\mu$ s at 12 MHz.
- Vectored, priority interrupts for I/O, counter/timers, and UART.
- Full-duplex UART and two programmable 8-bit counter/timers, each with a 6-bit programmable prescaler.
- Register Pointer so that short, fast instructions can access any of nine working register groups in 1  $\mu$ s.
- On-chip oscillator that accepts crystal or external clock drive.
- Low-power standby option that retains contents of general-purpose registers.
- Single +5 V power supply—all pins TTL-compatible.
- Low Power version (Z8621L):
  - Available 8 MHz
  - Current consumption 80 mA
- Available in 8 and 12 MHz versions.

### General Description

The Z8621 microcomputer introduces a new level of sophistication to single-chip architecture. Compared to earlier single-chip microcomputers, the Z8621 offers faster execution; more efficient use of memory; more sophisticated interrupt, input/output and bit-manipulation capabilities; and easier system expansion.

Under program control, the Z8621 can be

tailored to the needs of its user. It can be configured as a stand-alone microcomputer with 8K bytes of internal ROM, a traditional microprocessor that manages up to 112K bytes of external memory, or a parallel-processing element in a system with other processors and peripheral controllers linked by the Z-BUS. In all configurations, a large number of pins remain available for I/O.

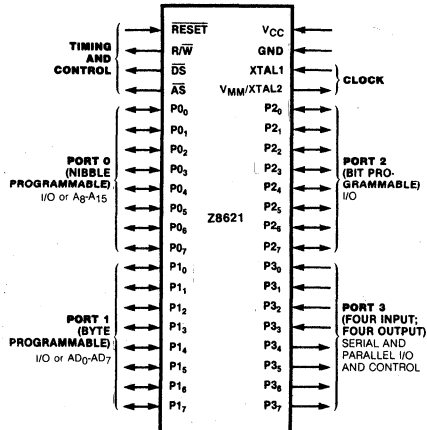


Figure 1. Logic Functions



# Z8621/L

## General Description (Continued)

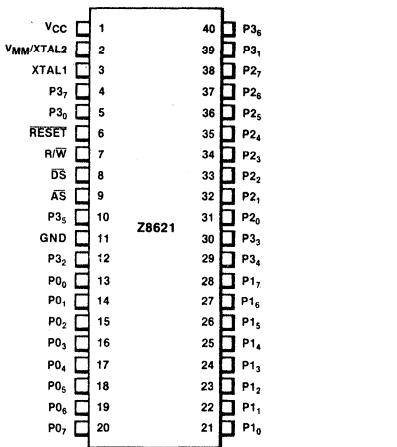


Figure 2. Pin Configuration

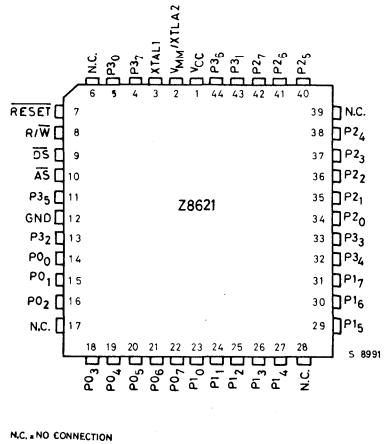


Figure 2a. Chip Carrier Pin Configuration

## Architecture

Z8621 architecture is characterized by a flexible I/O scheme, an efficient register and address space structure and a number of ancillary features that are helpful in many applications.

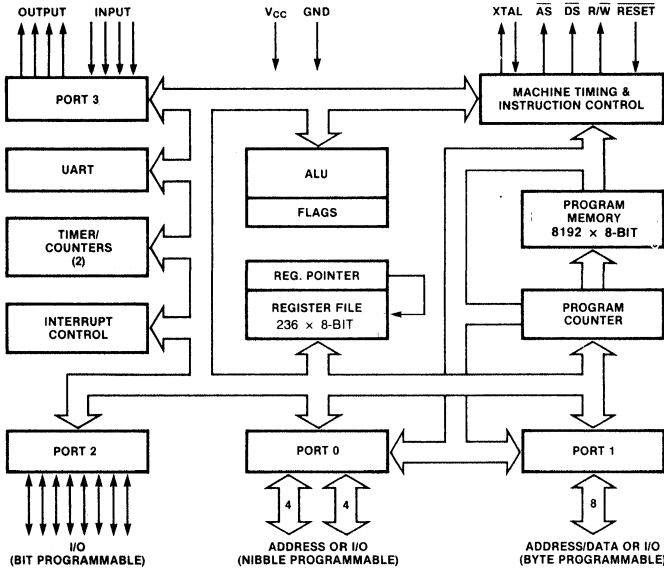
Microcomputer applications demand powerful I/O capabilities. The Z8621 fulfills this with 32 pins dedicated to input and output. These lines are grouped into four ports of eight lines each and are configurable under software control to provide timing, status signals, serial or parallel I/O with or without handshake, and an address/data bus for interfacing external memory.

Because the multiplexed address/data bus is merged with the I/O-oriented ports, the Z8621 can assume many different memory and I/O configurations. These configurations range from a self-contained microcomputer

to a microprocessor that can address 112K bytes of external memory (Figure 3).

Three basic address spaces are available to support this wide range of configurations: program memory (internal and external), data memory (external) and the register file (internal). The 256-byte random-access register file is composed of 236 general-purpose registers, four I/O port registers, and 16 control and status registers.

To unburden the program from coping with real-time problems such as serial data communication and counting/timing, an asynchronous receiver/transmitter (UART) and two counter/timers with a large number of user-selectable modes are offered on-chip. Hardware support for the UART is minimized because one of the on-chip timers supplies the bit rate.

**Architecture (Continued)**

**Figure 3. Block Diagram**
**Pin Description**

**AS.** *Address Strobe* (output, active Low). Address Strobe is pulsed once at the beginning of each machine cycle. Addresses output via Port 1 for all external program or data memory transfers are valid at the trailing edge of AS. Under program control, AS can be placed in the high-impedance state along with Ports 0 and 1, Data Strobe and Read/Write.

**DS.** *Data Strobe* (output, active Low). Data Strobe is activated once for each external memory transfer.

**P0<sub>0</sub>-P0<sub>7</sub>.** *I/O Port Lines* (input/output, TTL compatible). 8 lines Nibble Programmable that can be configured under program control for I/O or external memory interface.

**P1<sub>0</sub>-P1<sub>7</sub>.** *I/O Port Lines* (input/output, TTL compatible). 8 lines Byte Programmable that can be configured under program control for I/O or multiplexed address (A<sub>0</sub>-A<sub>7</sub>) and data (D<sub>0</sub>-D<sub>7</sub>) lines used to interface with program/data memory.

**P2<sub>0</sub>-P2<sub>7</sub>.** *I/O Port Lines* (input/output, TTL compatible). 8 lines Bit Programmable. In addition they can be configured to provide open-drain output.

**P3<sub>0</sub>-P3<sub>4</sub>.** *Input Port Lines* (TTL compatible). They can also be configured as control lines.

**P3<sub>5</sub>-P3<sub>7</sub>.** *Output Port Lines* (TTL compatible). They can also be configured as control lines.



---

**Pin Description** (Continued)

**RESET.** *Reset* (input, active Low).  $\overline{\text{RESET}}$  initializes the Z8621. When  $\overline{\text{RESET}}$  is deactivated, program execution begins from internal program location 000C<sub>H</sub>.

**R/ $\overline{\text{W}}$ .** *Read/Write* (output). R/ $\overline{\text{W}}$  is Low when the Z8621 is writing to external program or data memory.

**XTAL1, XTAL2.** *Crystal 1, Crystal 2* (time-base input and output). These pins connect a parallel-resonant crystal (8 or 12 MHz maximum) or an external single-phase clock (8 or 12 MHz maximum) to the on-chip clock oscillator and buffer.

---

**Address Spaces**

**Program Memory.** The 16-bit program counter addresses 64K bytes of program memory space. Program memory can be located in two areas: one internal and the other external (Figure 4). The first 8192 bytes consist of on-chip mask-programmed ROM. At addresses 8192 and greater, the Z8621 executes external program memory fetches.

The first 12 bytes of program memory are reserved for the interrupt vectors. These locations contain six 16-bit vectors that correspond to the six available interrupts.

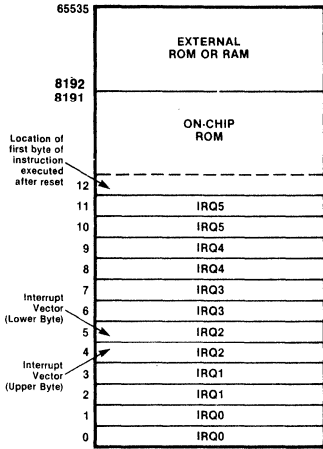
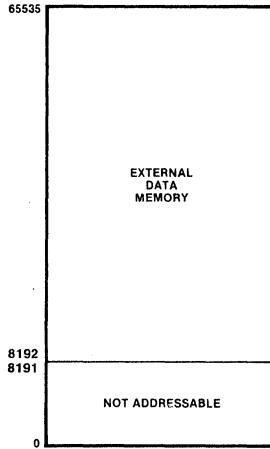
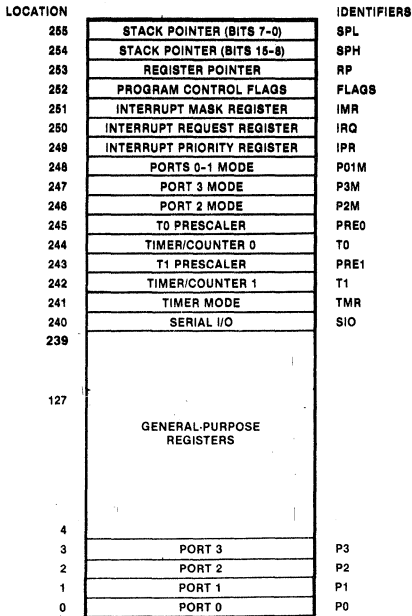
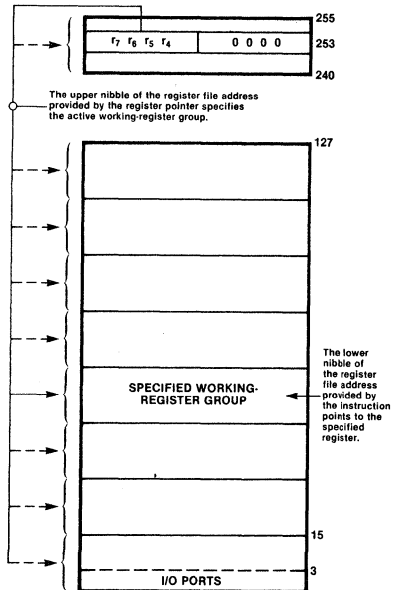
**Data Memory.** The Z8621 can address 56K bytes of external data memory beginning at locations 8192 (Figure 5). External data memory may be included with or separated from the external program memory space.  $\overline{\text{DM}}$ , an optional I/O function that can be programmed to appear on pin P3<sub>4</sub>, is used to distinguish between data and program memory space.

**Register File.** The 256-byte register file includes four I/O port registers (R0-R3), 236

general-purpose registers (R4-R238) and 16 control and status registers (R240-R255). These registers are assigned the address locations shown in Figure 6.

Z8621 instructions can access registers directly or indirectly with an 8-bit address field. The Z8621 also allows short 4-bit register addressing using the Register Pointer (one of the control registers). In the 4-bit mode, the register file is divided into nine working-register groups, each occupying 16 contiguous locations (Figure 7). The Register Pointer addresses the starting location of the active working-register group.

**Stacks.** Either the internal register file or the external data memory can be used for the stack. A 16-bit Stack Pointer (R254 and R255) is used for the external stack, which can reside anywhere in data memory between locations 8192 and 65535. An 8-bit Stack Pointer (R255) is used for the internal stack that resides within the 236 general-purpose registers (R4-R238).

**Address Spaces (Continued)**

**Figure 4. Program Memory Map**

**Figure 5. Data Memory Map**

**Figure 6. The Register File**

**Figure 7. The Register Pointer**





## Serial Input/Output

Port 3 lines P<sub>30</sub> and P<sub>37</sub> can be programmed as serial I/O lines for full-duplex serial asynchronous receiver/transmitter operation. The bit rate is controlled by Counter/Timer 0, with a maximum rate of 62.5K bits/second for 8.

The Z8621 automatically adds a start bit and two stop bits to transmitted data (Figure 8). Odd parity is also available as an option. Eight data bits are always transmitted,

regardless of parity selection. If parity is enabled, the eighth bit is the odd parity bit. An interrupt request (IRQ<sub>4</sub>) is generated on all transmitted characters.

Received data must have a start bit, eight data bits and at least one stop bit. If parity is on, bit 7 of the received data is replaced by a parity error flag. Received characters generate the IRQ<sub>3</sub> interrupt request.

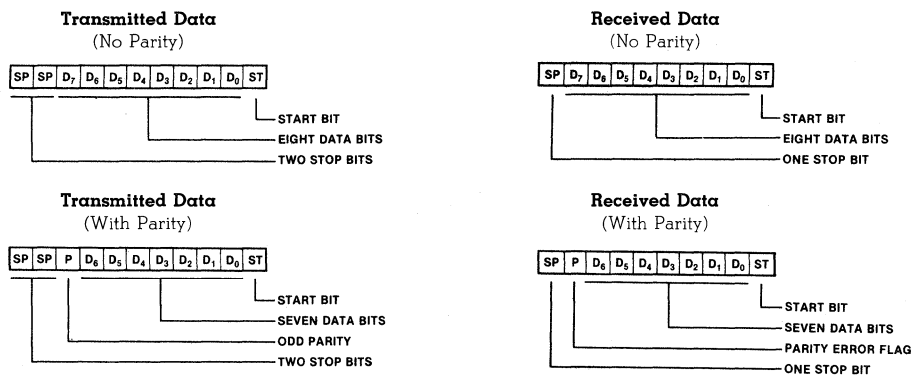


Figure 8. Serial Data Formats

## Counter/Timers

The Z8621 contains two 8-bit programmable counter/timers (T<sub>0</sub> and T<sub>1</sub>), each driven by its own 6-bit programmable prescaler. The T<sub>1</sub> prescaler can be driven by internal or external clock sources; however, the T<sub>0</sub> prescaler is driven by the internal clock only.

The 6-bit prescalers can divide the input frequency of the clock source by any number from 1 to 64. Each prescaler drives its counter, which decrements the value (1 to 256) that has been loaded into the counter. When the counter reaches the end of count, a timer interrupt request—IRQ<sub>4</sub> (T<sub>0</sub>) or IRQ<sub>5</sub> (T<sub>1</sub>)—is generated.

The counters can be started, stopped, restarted to continue, or restarted from the initial value. The counters can also be programmed to stop upon reaching zero (single-pass mode) or to automatically reload

the initial value and continue counting (modulo-n continuous mode). The counters, but not the prescalers, can be read any time without disturbing their value or count mode.

The clock source for T<sub>1</sub> is user-definable and can be the internal microprocessor clock (4 MHz maximum for the 8 MHz device and 6 MHz maximum for the 12 MHz device) divided by four, or an external signal input via Port 3. The Timer Mode register configures the external timer input as an external clock (1 MHz maximum), a trigger input that can be retriggerable or non-retriggerable, or as a gate input for the internal clock. The counter/timers can be porogrammably cascaded by connecting the T<sub>0</sub> output to the input of T<sub>1</sub>. Port 3 line P<sub>36</sub> also serves as a timer output (T<sub>OUT</sub>) through which T<sub>0</sub>, T<sub>1</sub> or the internal clock can be output.

### I/O Ports

The Z8621 has 32 lines dedicated to input and output. These lines are grouped into four ports of eight lines each and are configurable as input, output or address/data. Under software control, the ports can be programmed to provide address outputs, timing, status signals, serial I/O, and parallel I/O with or without handshake. All ports have active pull-ups and pull-downs compatible with TTL loads.

**Port 1** can be programmed as a byte I/O port or as an address/data port for interfacing external memory. When used as an I/O port, Port 1 may be placed under handshake control. In this configuration, Port 3 lines P<sub>33</sub> and P<sub>34</sub> are used as the handshake controls RDY<sub>1</sub> and  $\overline{\text{DAV}}_1$  (Ready and Data Available).

Memory locations greater than 8192 are referenced through Port 1. To interface external memory, Port 1 must be programmed for the multiplexed Address/Data mode. If more than 256 external locations are required, Port 0 must output the additional lines.

Port 1 can be placed in the high-impedance state along with Port 0,  $\overline{\text{AS}}$ ,  $\overline{\text{DS}}$  and R/W, allowing the Z8621 to share common resources in multiprocessor and DMA applications. Data transfers can be controlled by assigning P<sub>33</sub> as a Bus Acknowledge input and P<sub>34</sub> as a Bus Request output.

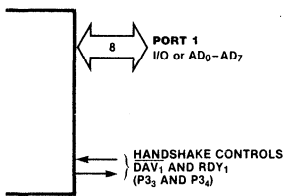


Figure 9a. Port 1

**Port 0** can be programmed as a nibble I/O port, or as an address port for interfacing external memory. When used as an I/O port,

Port 0 may be placed under handshake control. In this configuration, Port 3 lines P<sub>32</sub> and P<sub>35</sub> are used as the handshake controls  $\overline{\text{DAV}}_0$  and RDY<sub>0</sub>. Handshake signal assignment is dictated by the I/O direction of the upper nibble P<sub>04</sub>-P<sub>07</sub>.

For external memory references, Port 0 can provide address bits A<sub>8</sub>-A<sub>11</sub> (lower nibble) or A<sub>8</sub>-A<sub>15</sub> (lower and upper nibble) depending on the required address space. If the address range requires 12 bits or less, the upper nibble of Port 0 can be programmed independently as I/O while the lower nibble is used for addressing. When Port 0 nibbles are defined as address bits, they can be set to the high-impedance state along with Port 1 and the control signals  $\overline{\text{AS}}$ ,  $\overline{\text{DS}}$  and R/W.

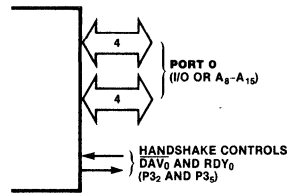


Figure 9b. Port 0

**Port 2** bits can be programmed independently as input or output. The port is always available for I/O operations. In addition, Port 2 can be configured to provide open-drain outputs.

Like Ports 0 and 1, Port 2 may also be placed under handshake control. In this configuration, Port 3 lines P<sub>31</sub> and P<sub>36</sub> are

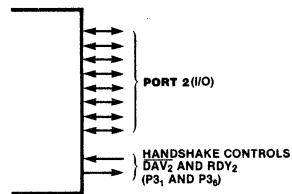


Figure 9c. Port 2



## I/O Ports (Continued)

used as the handshake controls lines  $\overline{DAV}_2$  and  $RDY_2$ . The handshake signal assignment for Port 3 lines  $P3_1$  and  $P3_6$  is dictated by the direction (input or output) assigned to bit 7 of Port 2.

**Port 3** lines can be configured as I/O or control lines. In either case, the direction of the eight lines is fixed as four input ( $P3_0$ - $P3_3$ ) and four output ( $P3_4$ - $P3_7$ ). For serial I/O, lines  $P3_0$  and  $P3_7$  are programmed as serial in and serial out respectively.

Port 3 can also provide the following control functions: handshake for Ports 0, 1 and 2 ( $\overline{DAV}$  and  $RDY$ ); four external

interrupt request signals ( $IRQ_0$ - $IRQ_3$ ); timer input and output signals ( $T_{IN}$  and  $T_{OUT}$ ) and Data Memory Select ( $\overline{DM}$ ).

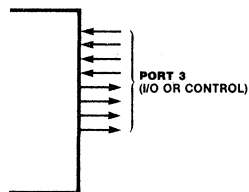


Figure 9d. Port 3

## Interrupts

The Z8621 allows six different interrupts from eight sources: the four Port 3 lines  $P3_0$ - $P3_3$ , Serial In, Serial Out, and the two counter/timers. These interrupts are both maskable and prioritized. The Interrupt Mask register globally or individually enables or disables the six interrupt requests. When more than one interrupt is pending, priorities are resolved by a programmable priority encoder that is controlled by the Interrupt Priority register.

All Z8621 interrupts are vectored. When an interrupt request is granted, and interrupt machine cycle is entered. This disables all

subsequent interrupts, saves the Program Counter and status flags, and branches to the program memory vector location reserved for that interrupt. This memory location and the next byte contain the 16-bit address of the interrupt service routine for that particular interrupt request.

Polled interrupt systems are also supported. To accommodate a polled structure, any or all of the interrupt inputs can be masked and the Interrupt Request register polled to determine which of the interrupt requests needs service.

## Clock

The on-chip oscillator has a high-gain, parallel-resonant amplifier for connection to a crystal or to any suitable external clock source ( $XTAL1$  = Input,  $XTAL2$  = Output).

The crystal source is connected across  $XTAL1$  and  $XTAL2$ , using the recommended

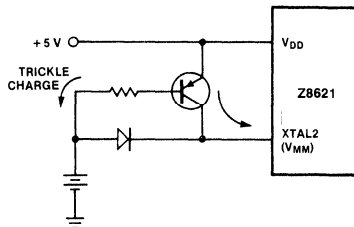
capacitors ( $C_1 \leq 15$  pF) from each pin to ground. The specifications for the crystal are as follows:

- AT cut, parallel resonant
- Fundamental types, 8/12 MHz maximum
- Series resistance,  $R_s \leq 100 \Omega$ .

### Power Down Standby Option

The low-power standby mode allows power to be removed without losing the contents of the 236 general-purpose registers. This mode is available to the user as a bonding option whereby pin 2 (normally XTAL2) is replaced by the  $V_{MM}$  (standby) power supply input. This necessitates the use of an external clock generator (input = XTAL1) rather than a crystal source.

The removal of power, whether intended or due to power failure, must be preceded by a software routine that stores the appropriate status into the register file. Figure 10 shows the recommended circuit for a battery back-up supply system.



**Figure 10. Recommended Driver Circuit for Power Down Operation**

### Instruction Set Notation

**Addressing Modes.** The following notation is used to describe the addressing modes and instruction operations as shown in the instruction summary.

- IRR** Indirect register pair or indirect working-register pair address
- Irr** Indirect working-register pair only
- X** Indexed address
- DA** Direct address
- RA** Relative address
- IM** Immediate
- R** Register or working-register address
- r** Working-register address only
- IR** Indirect-register or indirect working-register address
- Ir** Indirect working-register address only
- RR** Register pair or working register pair address

**Symbols.** The following symbols are used in describing the instruction set.

- dst** Destination location or contents
- src** Source location or contents
- cc** Condition code (see list)
- @** Indirect address prefix
- SP** Stack pointer (control registers 254-255)
- PC** Program counter
- FLAGS** Flag register (control register 252)
- RP** Register pointer (control register 253)
- IMR** Interrupt mask register (control register 251)

Assignment of a value is indicated by the symbol " $\leftarrow$ ". For example,

$$dst \leftarrow dst + src$$

indicates that the source data is added to the destination data and the result is stored in the destination location. The notation "addr(n)" is used to refer to bit "n" of a given location. For example,

$$dst(7)$$

refers to bit 7 of the destination operand.

**Flags.** Control Register R252 contains the following six flags:

<b>C</b>	Carry flag	b7							b0	
<b>Z</b>	Zero flag		C	Z	S	V	D	H	F2	F1
<b>S</b>	Sign flag									
<b>V</b>	Overflow flag									
<b>D</b>	Decimal-adjust flag									
<b>H</b>	Half-carry flag									
			<b>F1</b>		<b>F2</b>		} user flags			

Affected flags are indicated by:

- 0** Cleared to zero
- 1** Set to one
- \*** Set or cleared according to operation
- Unaffected
- X** Undefined

**Condition Codes**

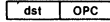
<b>Value</b>	<b>Mnemonic</b>	<b>Meaning</b>	<b>Flags Set</b>
1000		Always true	...
0111	C	Carry	C=1
1111	NC	No carry	C=0
0110	Z	Zero	Z=1
1110	NZ	Not zero	Z=0
1101	PL	Plus	S=0
0101	MI	Minus	S=1
0100	OV	Overflow	V=1
1100	NOV	No overflow	V=0
0110	EQ	Equal	Z=1
1110	NE	Not equal	Z=0
1001	GE	Greater than or equal	(S XOR V)=0
0001	LT	Less than	(S XOR V)=1
1010	GT	Greater than	[Z OR (S XOR V)]=0
0010	LE	Less than or equal	[Z OR (S XOR V)]=1
1111	UGE	Unsigned greater than or equal	C=0
0111	ULT	Unsigned less than	C=1
1011	UGT	Unsigned greater than	(C=0 AND Z=0)=1
0011	ULE	Unsigned less than or equal	(C OR Z)=1
0000		Never true	...



Instruction Formats

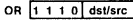
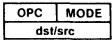


CCF, DI, EI, IRET, NOP,  
RCF, RET, SCF

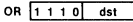
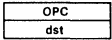


INC r

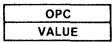
One-Byte Instructions



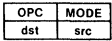
CLR, CPL, DA, DEC,  
DECW, INC, INCW, POP,  
PUSH, RL, RLC, RR,  
RRC, SRA, SWAP



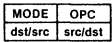
JP, CALL (Indirect)



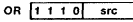
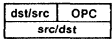
SRP



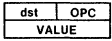
ADC, ADD, AND, CP,  
OR, SBC, SUB, TCM, TM, XOR



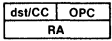
LD, LDE, LDEI,  
LDC, LDCI



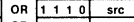
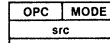
LD



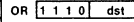
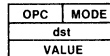
LD



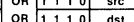
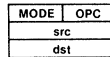
DJNZ, JR



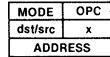
ADC, ADD, AND, CP,  
LD, OR, SBC, SUB,  
TCM, TM, XOR



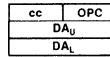
ADC, ADD, AND, CP,  
LD, OR, SBC, SUB,  
TCM, TM, XOR



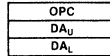
LD



LD



JP



CALL

Two-Byte Instructions

Three-Byte Instructions

Figure 11. Instruction Formats




Instruction Summary

Instruction and Operation	Addr Mode		Opcode Byte (Hex)	Flags Affected					
	dst	src		C	Z	S	V	D	H
<b>ADC</b> dst,src dst - dst + src + C	(Note 1)		1□	*	*	*	*	0	*
<b>ADD</b> dst,src dst - dst + src	(Note 1)		0□	*	*	*	*	0	*
<b>AND</b> dst,src dst - dst AND src	(Note 1)		5□	-	*	*	0	-	-
<b>CALL</b> dst SP - SP - 2 @SP - PC; PC - dst	DA IRR		D6 D4	-	-	-	-	-	-
<b>CCF</b> C - NOT C			EF	*	-	-	-	-	-
<b>CLR</b> dst dst - 0	R IR		B0 B1	-	-	-	-	-	-
<b>COM</b> dst dst - NOT dst	R IR		60 61	-	*	*	0	-	-
<b>CP</b> dst,src dst - src	(Note 1)		A□	*	*	*	*	-	-
<b>DA</b> dst dst - DA dst	R IR		40 41	*	*	*	X	-	-
<b>DEC</b> dst dst - dst - 1	R IR		00 01	-	*	*	*	-	-
<b>DECW</b> dst dst - dst - 1	RR IR		80 81	-	*	*	*	-	-
<b>DI</b> IMR (7) - 0			8F	-	-	-	-	-	-
<b>DJNZ</b> r,dst r - r - 1 if r ≠ 0 PC - PC + dst Range: +127, -128	RA		rA r=0-F	-	-	-	-	-	-
<b>EI</b> IMR (7) - 1			9F	-	-	-	-	-	-
<b>INC</b> dst dst - dst + 1	r R IR		rE r=0-F 20 21	-	*	*	*	*	-
<b>INCW</b> dst dst - dst + 1	RR IR		A0 A1	-	*	*	*	*	-
<b>IRET</b> FLAGS - @SP; SP - SP + 1 PC - @SP; SP - SP + 2; IMR (7) - 1			BF	*	*	*	*	*	*
<b>JP</b> cc,dst if cc is true PC - dst	DA IRR		cD c=0-F 30	-	-	-	-	-	-
<b>JR</b> cc,dst if cc is true, PC - PC + dst Range: +127, -128	RA		cB c=0-F	-	-	-	-	-	-

Instruction and Operation	Addr Mode		Opcode Byte (Hex)	Flags Affected					
	dst	src		C	Z	S	V	D	H
<b>LD</b> dst,src dst - src	r r R	Im R r	rC r8 r9 r=0-F	-	-	-	-	-	-
	r X r r R R R R IR IR	X r Ir r R R IR Im Im R	C7 D7 E3 F3 E4 E5 E6 E7 F5						
<b>LDC</b> dst,src dst - src	r Irr	Irr r	C2 D2	-	-	-	-	-	-
<b>LDCI</b> dst,src dst - src r - r + 1; rr - rr + 1	Ir Irr	Irr Ir	C3 D3	-	-	-	-	-	-
<b>LDE</b> dst,src dst - src	r Irr	Irr r	82 92	-	-	-	-	-	-
<b>LDEI</b> dst,src dst - src r - r + 1; rr - rr + 1	Ir Irr	Irr Ir	83 93	-	-	-	-	-	-
<b>NOP</b>			FF	-	-	-	-	-	-
<b>OR</b> dst,src dst - dst OR src	(Note 1)		4□	-	*	*	0	-	-
<b>POP</b> dst dst - @SP SP - SP + 1	R IR		50 51	-	-	-	-	-	-
<b>PUSH</b> src SP - SP - 1; @SP - src		R IR	70 71	-	-	-	-	-	-
<b>RCF</b> C - 0			CF	0	-	-	-	-	-
<b>RET</b> PC - @SP; SP - SP + 2			AF	-	-	-	-	-	-
<b>RL</b> dst		R IR	90 91	*	*	*	*	-	-
<b>RLC</b> dst		R IR	10 11	*	*	*	*	-	-
<b>RR</b> dst		R IR	E0 E1	*	*	*	*	-	-
<b>RRC</b> dst		R IR	C0 C1	*	*	*	*	-	-
<b>SBC</b> dst,src dst - dst - src - C	(Note 1)		3□	*	*	*	*	1	*
<b>SCF</b> C - 1			DF	1	-	-	-	-	-
<b>SRA</b> dst		R IR	D0 D1	*	*	*	0	-	-



Instruction Summary (Continued)

Instruction and Operation	Addr Mode		Opcode Byte (Hex)	Flags Affected					
	dst	src		C	Z	S	V	D	H
<b>SRP</b> src RP - src		Im	31	-	-	-	-	-	-
<b>SUB</b> dst,src dst - dst - src		(Note 1)	2□	*	*	*	*	1	*
<b>SWAP</b> dst		R IR	F0 F1	X	*	*	X	-	-

Instruction and Operation	Addr Mode		Opcode Byte (Hex)	Flags Affected					
	dst	src		C	Z	S	V	D	H
<b>JCM</b> dst,src (NOT dst) AND src		(Note 1)	6□	-	*	*	0	-	-
<b>TM</b> dst, src dst AND src		(Note 1)	7□	-	*	*	0	-	-
<b>XOR</b> 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 is found in the instruction set table above. The second nibble is expressed symbolically by a □ in this table, and its value is found in the following table to the left of the applicable addressing mode pair.

For example, to determine the opcode of an ADC instruction using the addressing modes r (destination) and Ir (source), is 13.

Addr Mode		Lower Opcode Nibble
dst	src	
r	r	2
r	Ir	3
R	R	4
R	IR	5
R	IM	6
IR	IM	7

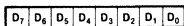




# Z8621/L

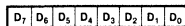
## Registers

**R240 SIO**  
Serial I/O Register  
(F0H; Read/Write)



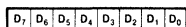
SERIAL DATA ( $D_0$  = LSB)

**R244 T0**  
Counter/Timer 0 Register  
(F4H; Read/Write)



$T_0$  INITIAL VALUE (WHEN WRITTEN)  
(RANGE: 1-256 DECIMAL 01-00 HEX)  
 $T_0$  CURRENT VALUE (WHEN READ)

**R241 TMR**  
Timer Mode Register  
(F1H; Read/Write)



$T_{OUT}$  MODES  
NOT USED = 00  
 $T_0$  OUT = 01  
 $T_1$  OUT = 10  
INTERNAL CLOCK OUT = 11

$T_{IN}$  MODES  
EXTERNAL CLOCK INPUT = 00  
GATE INPUT = 01  
TRIGGER INPUT = 10  
(NON-RETRIGGERABLE)  
TRIGGER INPUT = 11  
(RETRIGGERABLE)

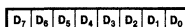
0 = NO FUNCTION  
1 = LOAD  $T_0$

0 = DISABLE  $T_0$  COUNT  
1 = ENABLE  $T_0$  COUNT

0 = NO FUNCTION  
1 = LOAD  $T_1$

0 = DISABLE  $T_1$  COUNT  
1 = ENABLE  $T_1$  COUNT

**R245 PRE0**  
Prescaler 0 Register  
(F5H; Write Only)

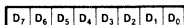


COUNT MODE  
0 =  $T_0$  SINGLE-PASS  
1 =  $T_0$  MODULO-N

RESERVED (MUST BE 0)

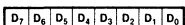
PRESCALER MODULO  
(RANGE: 1-64 DECIMAL  
01-00 HEX)

**R242 T1**  
Counter Timer 1 Register  
(F2H; Read/Write)



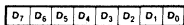
$T_1$  INITIAL VALUE (WHEN WRITTEN)  
(RANGE: 1-256 DECIMAL 01-00 HEX)  
 $T_1$  CURRENT VALUE (WHEN READ)

**R246 P2M**  
Port 2 Mode Register  
(F6H; Write Only)



$P2_0$ - $P2_7$  I/O DEFINITION  
0 DEFINES BIT AS OUTPUT  
1 DEFINES BIT AS INPUT

**R243 PRE1**  
Prescaler 1 Register  
(F3H; Write Only)

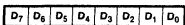


COUNT MODE  
0 =  $T_1$  SINGLE-PASS  
1 =  $T_1$  MODULO-N

CLOCK SOURCE  
1  $T_1$  INTERNAL  
0  $T_1$  EXTERNAL TIMING INPUT  
( $T_{IN}$ ) MODE

PRESCALER MODULO  
(RANGE: 1-64 DECIMAL  
01-00 HEX)

**R247 P3M**  
Port 3 Mode Register  
(F7H; Write Only)



0 PORT 2 PULL-UPS OPEN DRAIN  
1 PORT 2 PULL-UPS ACTIVE

RESERVED (MUST BE 0)

0  $P3_2$  = INPUT  $P3_5$  = OUTPUT  
1  $P3_2$  =  $DAV0/RDY0$   $P3_5$  =  $RDY0/DAV0$

0 0  $P3_3$  = INPUT  $P3_4$  = OUTPUT  
0 1  $P3_3$  = INPUT  $P3_4$  =  $DM$   
1 0  $P3_3$  =  $DAV1/RDY1$   $P3_4$  =  $RDY1/DAV1$   
1 1  $P3_3$  =  $DAV1/RDY1$   $P3_4$  =  $RDY1/DAV1$

0  $P3_1$  = INPUT ( $T_{IN}$ )  $P3_6$  = OUTPUT ( $T_{OUT}$ )  
1  $P3_1$  =  $DAV2/RDY2$   $P3_6$  =  $RDY2/DAV2$

0  $P3_0$  = INPUT  $P3_7$  = OUTPUT  
1  $P3_0$  = SERIAL IN  $P3_7$  = SERIAL OUT

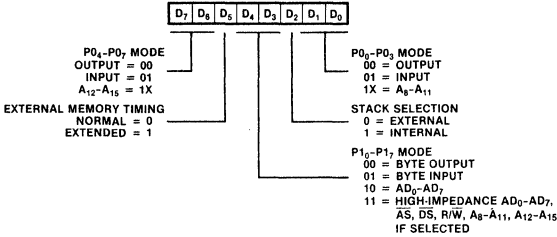
0 PARITY OFF  
1 PARITY ON

Figure 12. Control Registers

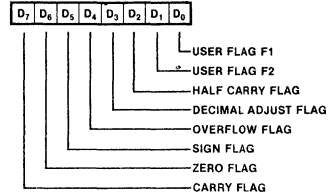


Registers (Continued)

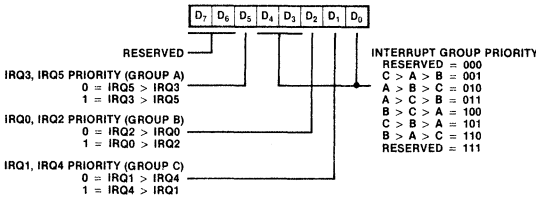
R248 P01M  
Port 0 and 1 Mode Register  
(F8H; Write Only)



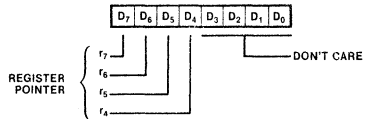
R252 FLAGS  
Flag Register  
(FC<sub>H</sub>; Read/Write)



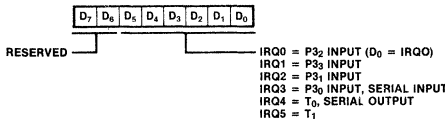
R249 IPR  
Interrupt Priority Register  
(F9H; Write Only)



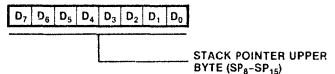
R253 RP  
Register Pointer  
(FDH; Read/Write)



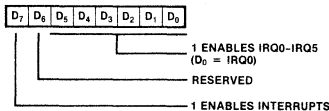
R250 IRQ  
Interrupt Request Register  
(FAH; Read/Write)



R254 SPH  
Stack Pointer  
(FEH; Read/Write)



R251 IMR  
Interrupt Mask Register  
(FBH; Read/Write)



R255 SPL  
Stack Pointer  
(FFH; Read/Write)

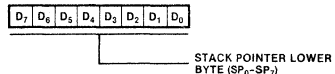


Figure 12. Control Registers (Continued)



Z8621/L

Opcode Map

Lower Nibble (Hex)

		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
Upper Nibble (Hex)	0	6,5 DEC R <sub>1</sub>	6,5 DEC IR <sub>1</sub>	6,5 ADD r <sub>1</sub> , r <sub>2</sub>	6,5 ADD r <sub>1</sub> , Ir <sub>2</sub>	10,5 ADD R <sub>2</sub> , R <sub>1</sub>	10,5 ADD IR <sub>2</sub> , R <sub>1</sub>	10,5 ADD R <sub>1</sub> , IM	10,5 ADD IR <sub>1</sub> , IM	6,5 LD r <sub>1</sub> , R <sub>2</sub>	6,5 LD r <sub>2</sub> , R <sub>1</sub>	12/10,5 DJNZ r <sub>1</sub> , RA	12/10,0 JR cc, RA	6,5 LD r <sub>1</sub> , IM	12/10,0 JP cc, DA	6,5 INC r <sub>1</sub>		
	1	6,5 RLC R <sub>1</sub>	6,5 RLC IR <sub>1</sub>	6,5 ADC r <sub>1</sub> , r <sub>2</sub>	6,5 ADC r <sub>1</sub> , Ir <sub>2</sub>	10,5 ADC R <sub>2</sub> , R <sub>1</sub>	10,5 ADC IR <sub>2</sub> , R <sub>1</sub>	10,5 ADC R <sub>1</sub> , IM	10,5 ADC IR <sub>1</sub> , IM									
	2	6,5 INC R <sub>1</sub>	6,5 INC IR <sub>1</sub>	6,5 SUB r <sub>1</sub> , r <sub>2</sub>	6,5 SUB r <sub>1</sub> , Ir <sub>2</sub>	10,5 SUB R <sub>2</sub> , R <sub>1</sub>	10,5 SUB IR <sub>2</sub> , R <sub>1</sub>	10,5 SUB R <sub>1</sub> , IM	10,5 SUB IR <sub>1</sub> , IM									
	3	8,0 JP IRR <sub>1</sub>	6,1 SRP IM	6,5 SBC r <sub>1</sub> , r <sub>2</sub>	6,5 SBC r <sub>1</sub> , Ir <sub>2</sub>	10,5 SBC R <sub>2</sub> , R <sub>1</sub>	10,5 SBC IR <sub>2</sub> , R <sub>1</sub>	10,5 SBC R <sub>1</sub> , IM	10,5 SBC IR <sub>1</sub> , IM									
	4	8,5 DA R <sub>1</sub>	8,5 DA IR <sub>1</sub>	6,5 OR r <sub>1</sub> , r <sub>2</sub>	6,5 OR r <sub>1</sub> , Ir <sub>2</sub>	10,5 OR R <sub>2</sub> , R <sub>1</sub>	10,5 OR IR <sub>2</sub> , R <sub>1</sub>	10,5 OR R <sub>1</sub> , IM	10,5 OR IR <sub>1</sub> , IM									
	5	10,5 POP R <sub>1</sub>	10,5 POP IR <sub>1</sub>	6,5 AND r <sub>1</sub> , r <sub>2</sub>	6,5 AND r <sub>1</sub> , Ir <sub>2</sub>	10,5 AND R <sub>2</sub> , R <sub>1</sub>	10,5 AND IR <sub>2</sub> , R <sub>1</sub>	10,5 AND R <sub>1</sub> , IM	10,5 AND IR <sub>1</sub> , IM									
	6	6,5 COM R <sub>1</sub>	6,5 COM IR <sub>1</sub>	6,5 TCM r <sub>1</sub> , r <sub>2</sub>	6,5 TCM r <sub>1</sub> , Ir <sub>2</sub>	10,5 TCM R <sub>2</sub> , R <sub>1</sub>	10,5 TCM IR <sub>2</sub> , R <sub>1</sub>	10,5 TCM R <sub>1</sub> , IM	10,5 TCM IR <sub>1</sub> , IM									
	7	10/12,1 PUSH R <sub>2</sub>	12/14,1 PUSH IR <sub>2</sub>	6,5 TM r <sub>1</sub> , r <sub>2</sub>	6,5 TM r <sub>1</sub> , Ir <sub>2</sub>	10,5 TM R <sub>2</sub> , R <sub>1</sub>	10,5 TM IR <sub>2</sub> , R <sub>1</sub>	10,5 TM R <sub>1</sub> , IM	10,5 TM IR <sub>1</sub> , IM									
	8	10,5 DECW RR <sub>1</sub>	10,5 DECW IR <sub>1</sub>	12,0 LDE r <sub>1</sub> , Ir <sub>2</sub>	18,0 LDEI Ir <sub>1</sub> , Ir <sub>2</sub>													6,1 DI
	9	6,5 RL R <sub>1</sub>	6,5 RL IR <sub>1</sub>	12,0 LDE r <sub>2</sub> , Ir <sub>1</sub>	18,0 LDEI Ir <sub>2</sub> , Ir <sub>1</sub>													6,1 EI
	A	10,5 INCW RR <sub>1</sub>	10,5 INCW IR <sub>1</sub>	6,5 CP r <sub>1</sub> , r <sub>2</sub>	6,5 CP r <sub>1</sub> , Ir <sub>2</sub>	10,5 CP R <sub>2</sub> , R <sub>1</sub>	10,5 CP IR <sub>2</sub> , R <sub>1</sub>	10,5 CP R <sub>1</sub> , IM	10,5 CP IR <sub>1</sub> , IM									14,0 RET
	B	6,5 CLR R <sub>1</sub>	6,5 CLR IR <sub>1</sub>	6,5 XOR r <sub>1</sub> , r <sub>2</sub>	6,5 XOR r <sub>1</sub> , Ir <sub>2</sub>	10,5 XOR R <sub>2</sub> , R <sub>1</sub>	10,5 XOR IR <sub>2</sub> , R <sub>1</sub>	10,5 XOR R <sub>1</sub> , IM	10,5 XOR IR <sub>1</sub> , IM									16,0 IRET
	C	6,5 RRC R <sub>1</sub>	6,5 RRC IR <sub>1</sub>	12,0 LDC r <sub>1</sub> , Ir <sub>2</sub>	18,0 LDCI Ir <sub>1</sub> , Ir <sub>2</sub>				10,5 LD r <sub>1</sub> , x, R <sub>2</sub>									6,5 RCF
	D	6,5 SRA R <sub>1</sub>	6,5 SRA IR <sub>1</sub>	12,0 LDC r <sub>2</sub> , Ir <sub>1</sub>	18,0 LDCI Ir <sub>2</sub> , Ir <sub>1</sub>	20,0 CALL* IRR <sub>1</sub>		20,0 CALL DA	10,5 LD r <sub>2</sub> , x, R <sub>1</sub>									6,5 SCF
	E	6,5 RR R <sub>1</sub>	6,5 RR IR <sub>1</sub>		6,5 LD r <sub>1</sub> , Ir <sub>2</sub>	10,5 LD R <sub>2</sub> , R <sub>1</sub>	10,5 LD IR <sub>2</sub> , R <sub>1</sub>	10,5 LD R <sub>1</sub> , IM	10,5 LD IR <sub>1</sub> , IM									6,5 CCF
	F	6,7 SWAP R <sub>1</sub>	6,7 SWAP IR <sub>1</sub>		6,5 LD Ir <sub>1</sub> , r <sub>2</sub>		10,5 LD R <sub>2</sub> , IR <sub>1</sub>											6,0 NOP

Bytes per Instruction

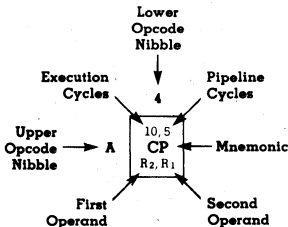
2

3

2

3

1



Legend:

- R = 8-Bit Address
- r = 4-Bit Address
- R<sub>1</sub> or r<sub>1</sub> = Dest Address
- R<sub>2</sub> or r<sub>2</sub> = Src Address

Sequence:

Opcode, First Operand, Second Operand

Note: The blank areas are not defined.

\*2-byte instruction: fetch cycle appears as a 3-byte instruction

### Absolute Maximum Ratings

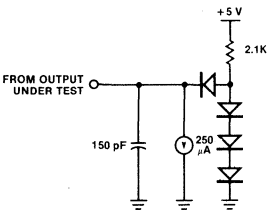
Voltages on all pins with respect to GND ..... -0.3 V to +7.0 V  
 Operating Ambient Temperature ..... 0°C to +70°C  
 Storage Temperature ..... -65°C to +150°C

Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

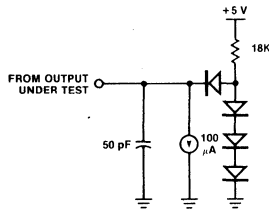
### Standard Test Conditions

The characteristics below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND. Positive current flows into the reference pin. Standard conditions are as follows:

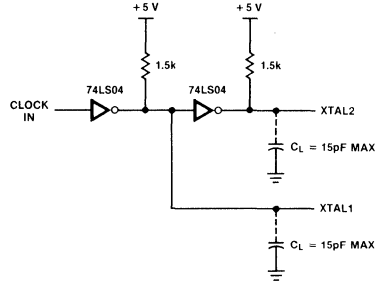
- +4.75 V ≤ V<sub>CC</sub> ≤ +5.25 V
- GND = 0 V
- 0°C ≤ T<sub>A</sub> ≤ +70°C



**Figure 13. Test Load 1**



**Figure 14. Test Load 2**



**Figure 15. External Clock Interface Circuit**  
 (Both the clock and complement are required)



## Z8621/L

### DC Characteristics

Symbol	Parameter	Min	Max	Unit	Condition
V <sub>CH</sub>	Clock Input High Voltage	3.8	V <sub>CC</sub>	V	Driven by External Clock Generator
V <sub>CL</sub>	Clock Input Low Voltage	-0.3	0.8	V	Driven by External Clock Generator
V <sub>IH</sub>	Input High Voltage	2.0	V <sub>CC</sub>	V	
V <sub>IL</sub>	Input Low Voltage	-0.3	0.8	V	
V <sub>RH</sub>	Reset Input High Voltage	3.8	V <sub>CC</sub>	V	
V <sub>RL</sub>	Reset Input Low Voltage	-0.3	0.8	V	
V <sub>OH</sub>	Output High Voltage	2.4		V	I <sub>OH</sub> = -250 $\mu$ A
V <sub>OL</sub>	Output Low Voltage		0.4	V	I <sub>OL</sub> = +2.0 mA
I <sub>IL</sub>	Input Leakage	-10	10	$\mu$ A	0 V $\leq$ V <sub>IN</sub> $\leq$ +5.25 V
I <sub>OL</sub>	Output Leakage	-10	10	$\mu$ A	0 V $\leq$ V <sub>IN</sub> $\leq$ +5.25 V
I <sub>IR</sub>	Reset Input Current		-50	$\mu$ A	V <sub>CC</sub> = +5.25 V, V <sub>RL</sub> = 0 V
I <sub>CC</sub>	V <sub>CC</sub> Supply Current		120	mA	
			80*		
I <sub>MM</sub>	V <sub>MM</sub> Supply Current		10	mA	Power Down Mode
V <sub>MM</sub>	Backup Supply Voltage	3	V <sub>CC</sub>	V	Power Down

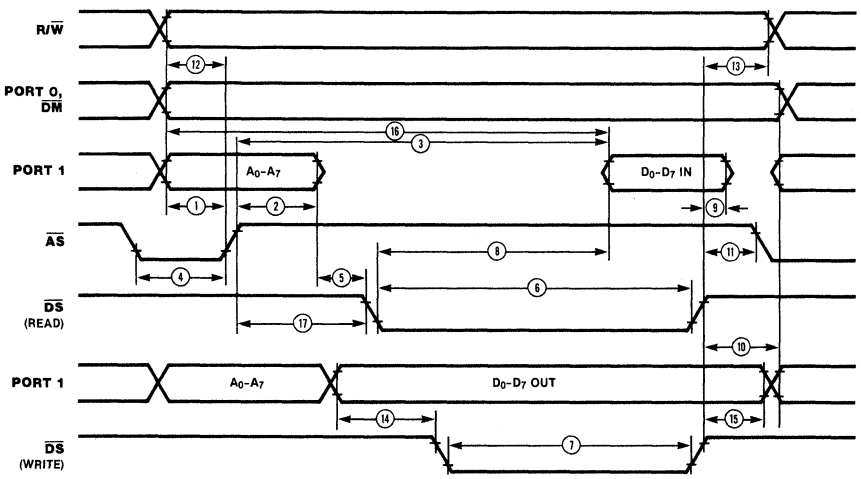
\* This value is for Z8621L only

**External I/O or Memory Read and Write Timing**

No	Symbol	Parameter	Z8621/L		Z8621A		Notes*†
			Min	Max	Min	Max	
1	TdA(AS)	Address Valid to $\overline{AS}$ $\uparrow$ Delay	50			360	1,2,3
2	TdAS(A)	$\overline{AS}$ $\uparrow$ to Address Float Delay	70		45		1,2,3
3	TdAS(DR)	$\overline{AS}$ $\uparrow$ to Read Data Required Valid		360		220	1,2,3,4
4	TwAS	$\overline{AS}$ Low Width	80		55		1,2,3
5	TdAz(DS)	Address Float to $\overline{DS}$ $\downarrow$	0		0		1
6	TwDSR	$\overline{DS}$ (Read) Low Width	250		185		1,2,3,4
7	TwDSW	$\overline{DS}$ (Write) Low Width	160		110		1,2,3,4
8	TdDSR(DR)	$\overline{DS}$ $\downarrow$ to Read Data Required Valid		200		130	1,2,3,4
9	ThDR(DS)	Read Data to $\overline{DS}$ $\downarrow$ Hold Time	0		0		1
10	TdDS(A)	$\overline{DS}$ $\uparrow$ to Address Active Delay	70		45		1,2,3
11	TdDS(AS)	$\overline{DS}$ $\uparrow$ to $\overline{AS}$ $\downarrow$ Delay	70		55		1,2,3
12	TdR/W(AS)	R/W Valid to $\overline{AS}$ $\uparrow$ Delay	50		30		1,2,3
13	TdDS(R/W)	$\overline{DS}$ $\uparrow$ to R/W Not Valid	60		35		1,2,3
14	TdDW(DSW)	Write Data Valid to $\overline{DS}$ (Write) $\downarrow$ Delay	50		35		1,2,3
15	TdDS(DW)	$\overline{DS}$ $\uparrow$ to Write Data Not Valid Delay	70		45		1,2,3
16	TdA(DR)	Address Valid to Read Data Required Valid		410		255	1,2,3,4
17	TdAS(DS)	$\overline{AS}$ $\uparrow$ to $\overline{DS}$ $\downarrow$ Delay	80		55		1,2,3

**NOTES:**

1. Test Load 1
  2. Timing numbers given are for minimum TpC.
  3. Also see clock cycle time dependent characteristics table.
  4. When using extended memory timing add 2 TpC.
  5. All timing reference use 2.0 V for a logic "1" and 0.8 V for a logic "0".
- \* All units in nanoseconds (ns).  
 † Timings are preliminary and subject to change.


**Figure 16. External I/O or Memory Read/Write**



**Additional Timing Table**

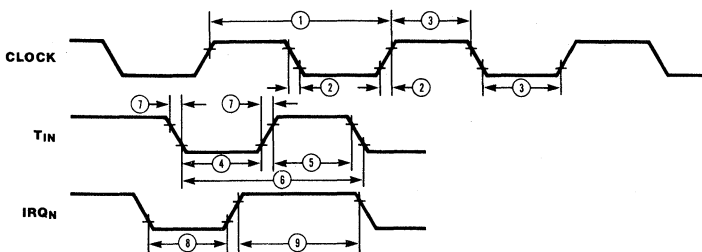
No	Symbol	Parameter	Z8621/L		Z8621A		Notes*†
			Min	Max	Min	Max	
1	TpC	Input Clock Period	125	1000	83	1000	1
2	TrC, TfC	Clock Input Rise And Fall Times		25		15	1
3	TwC	Input Clock Width	37		26		1
4	TwTinL	Timer Input Low Width	100		70		2
5	TwTinH	Timer Input High Width	3TpC		3TpC		2
6	TpTin	Timer Input Period	8TpC		8TpC		2
7	TrTin, TfTin	Timer Input Rise And Fall Times		100		100	2
8a	TwIL	Interrupt Request Input Low Time	100		70		2,3
8b	TwIH	Interrupt Request Input Low Time	3TpC		3TpC		2,4
9	TwIH	Interrupt Request Input High Time	3TpC		3TpC		2,3

**NOTES:**

1. Clock timing references uses 3.8 V for a logic "1" and 0.8 V for a logic "0".
2. Timing reference uses 2.0 V for a logic "1" and 0.8 V for a logic "0".

3. Interrupt request via Port 3 (P3<sub>1</sub>-P3<sub>3</sub>).
4. Interrupt request via Port 3 (P3<sub>0</sub>).

\* Units in nanoseconds (ns).  
 † Timings are preliminary and subject to change.



**Figure 17. Additional Timing**



### Handshake Timing

No	Symbol	Parameter	Z8621/L		Z8621A		Notes*†
			Min	Max	Min	Max	
1	TsDI(DAV)	Data In Setup Time	0		0		
2	ThDI(DAV)	Data In Hold Time	230		160		
3	TwDAV	Data Available Width	175		120		
4	TdDAVIh(RDY)	$\overline{DAV}$ ↓ Input to RDY ↓ Delay		175		120	1,2
5	TdDAVOh(RDY)	$\overline{DAV}$ ↓ Output to RDY ↓ Delay	0		0		1,3
6	TdDAVIr(RDY)	$\overline{DAV}$ ↑ Input to RDY ↑ Delay		175		120	1,2
7	TdDAVOr(RDY)	$\overline{DAV}$ ↑ Output to RDY ↑ Delay	0		0		1,3
8	TdDO(DAV)	Data Out to $\overline{DAV}$ ↓ Delay	50		30		1
9	TdRDY(DAV)	Rdy ↓ Input to $\overline{DAV}$ ↑ Delay	0	200	0	140	1

NOTES:

- 1. Test Load 1
- 2. Input handshake
- 3. Output handshake
- 4. All timing references use 2.0 V for a logic "1" and 0.8 V for a logic "0".
- \* Units in nanoseconds (ns).
- † Timings are preliminary and subject to change.

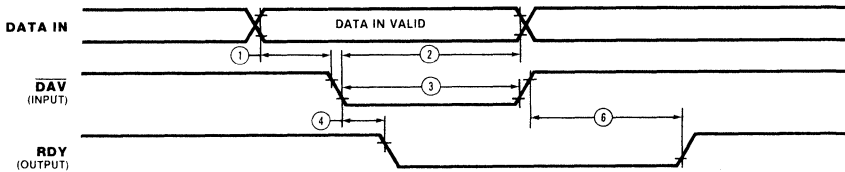


Figure 18a. Input Handshake

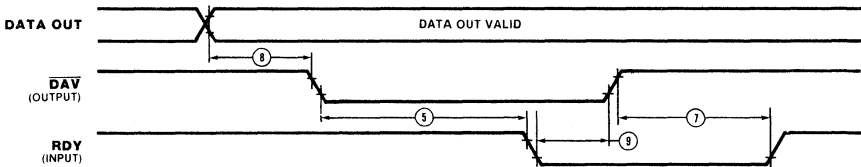


Figure 18b. Output Handshake





# Z8621/L

## Clock-Cycle-Time-Dependent Characteristics

Number	Symbol	Z8621/L Equation	Z8621A Equation
1	TdA(AS)	TpC-75	TpC-50
2	TdAS(A)	TpC-55	TpC-40
3	TdAS(DR)	4TpC-140*	4TpC-110*
4	TwAS	TpC-45	TpC-30
6	TwDSR	3TpC-125*	3TpC-65*
7	TwDSW	2TpC-90*	2TpC-55*
8	TdDSR(DR)	3TpC-175*	3TpC-120*
10	Td(DS)A	TpC-55	TpC-40
11	TdDS(AS)	TpC-55	TpC-30
12	TdR/W(AS)	TpC-75	TpC-55
13	TdDS(R/W)	TpC-65	TpC-50
14	TdDW(DSW)	TpC-75	TpC-50
15	TdDS(DW)	TpC-55	TpC-40
16	TdA(DR)	5TpC-215*	5TpC-160*
17	TdAS(DS)	TpC-45	TpC-30

\* Add 2TpC when using extended memory timing

## Ordering Information

Type	Package	Temp.	Clock	Description
Z8621 B1	Plastic	0/ +70°C	8 MHz	8K ROM Microcomputer
Z8621 B6	Plastic	-40/ +85°C		
Z8621 D1	Ceramic	0/ +70°C		
Z8621 D6	Ceramic	-40/ +85°C		
Z8621 C1	Plastic Chip Carrier	0/ +70°C		
Z8621 C6	Plastic Chip Carrier	-40/ +85°C		
Z8621 K1	Ceramic Chip Carrier	0/ +70°C		
Z8621 K6	Ceramic Chip Carrier	-40/ +85°C		
Z8621A B1	Plastic	0/ +70°C	12 MHz	8K ROM Microcomputer Low Power version
Z8621A B6	Plastic	-40/ +85°C		
Z8621A D1	Ceramic	0/ +70°C		
Z8621A D6	Ceramic	-40/ +85°C		
Z8621A C1	Plastic Chip Carrier	0/ +70°C		
Z8621A C6	Plastic Chip Carrier	-40/ +85°C		
Z8621A K1	Ceramic Chip Carrier	0/ +70°C		
Z8621A K6	Ceramic Chip Carrier	-40/ +85°C		
Z8621L B1	Plastic	0/ +70°C	8 MHz	8K ROM Microcomputer Low Power version
Z8621L B6	Plastic	-40/ +85°C		
Z8621L D1	Ceramic	0/ +70°C		
Z8621L D6	Ceramic	-40/ +85°C		
Z8621L C1	Plastic Chip Carrier	0/ +70°C		
Z8621L C6	Plastic Chip Carrier	-40/ +85°C		
Z8621L K1	Ceramic Chip Carrier	0/ +70°C		
Z8621L K6	Ceramic Chip Carrier	-40/ +85°C		



# Z8671

## Z8 BASIC/Debug Interpreter

- The Z8671 MCU is a complete microcomputer preprogrammed with a BASIC/Debug interpreter. Interaction between the interpreter and its user is provided through an on-board UART.
- BASIC/Debug can directly address the Z8671's internal registers and all external memory. It provides quick examination and modification of any external memory location or I/O port.
- The BASIC/Debug interpreter can call machine language subroutines to increase execution speed.
- The Z8671's auto start-up capability allows a program to be executed on power-up or Reset without operator intervention.
- Single + 5 V power supply – all I/O pins TTL-compatible.
- Available in 8 MHz version.

### General Description

The Z8671 Single-Chip Microcomputer (MCU) is one of a line of preprogrammed chips – in this case with a BASIC/Debug interpreter in ROM-offered by SGS. As a member of the Z8 Family of microcomputers, it offers the same abundance of resources as the other Z8 microcomputers.

Because the BASIC/Debug interpreter is already part of the chip circuit, programming is made much easier. The Z8671 MCU thus offers a combination of software and hardware that is ideal for many industrial applications. The Z8671 MCU allows fast hardware tests and bit-by-bit examination and modification of memory location, I/O ports, or registers. It also allows bit manipulation and logical operations. A self-contained line editor supports interactive debugging, further speeding up program development.

The BASIC/Debug interpreter, a subset of Dartmouth BASIC, operates with three kinds of memory: on-chip registers and external ROM or RAM. The BASIC/Debug interpreter is located in the 2K bytes of on-chip ROM.

Additional features of the Z8671 MCU include the ability to call machine language subroutines to increase execution speed and the ability to have a program execute on power-up or Reset, without operator intervention.

Maximum memory addressing capabilities include 62K bytes of external program

memory and 62K bytes of data memory with program storage beginning at location 800 hex. This provides up to 124K bytes of useable memory space. Very few 8-bit microcomputers can directly access this amount of memory.

Each Z8671 Microcomputer has 32 I/O lines, a 144-byte register file, an on-board UART, and two counter/timers.

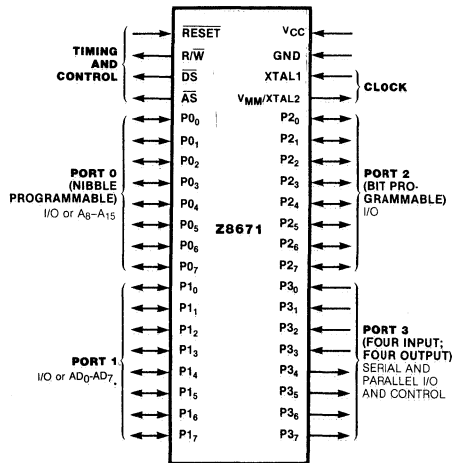


Figure 1. Logic Function



Z8671

General Description (Continued)

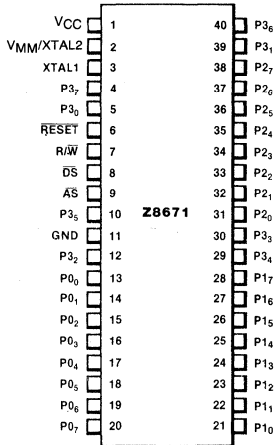


Figure 2a. Pin Configuration

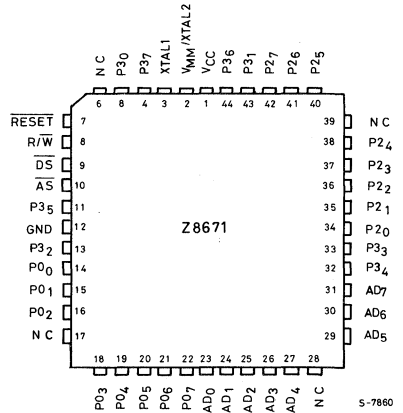


Figure 2b. Chip Carrier Pin Configuration

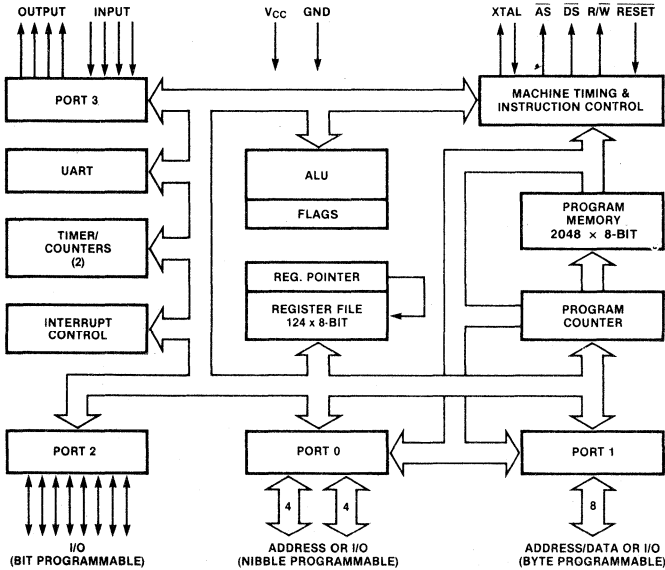
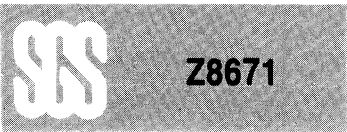


Figure 3. Functional Block Diagram



## Architecture

Z8671 architecture is characterized by a flexible I/O scheme, and efficient register and address space structure, and a number of ancillary features that are helpful in many applications.

Microcomputer applications demand powerful I/O capabilities. The Z8671 fulfills this with 32 pins dedicated to input and output. These lines are grouped into four ports of eight lines each and are configurable under software control to provide timing, status signals, serial or parallel I/O with or without handshake, and an address/data bus for interfacing external memory.

Because the multiplexed address/data bus is merged with the I/O-oriented ports, the Z8671 can assume many different memory and I/O configurations. These configurations range from a self-contained microcomputer

to a microprocessor that can address 124K bytes of external memory.

Three basic address spaces are available to support this wide range of configurations: program memory (internal and external), data memory (external) and the register file (internal). The 144-byte random-access register file is composed of 124 general-purpose registers, four I/O port registers, and 16 control and status registers.

To unburden the program from coping with real-time problems such as serial data communication and counting/timing, an asynchronous receiver/transmitter (UART) and two counter/timers with a large number of user-selectable modes are offered on-chip. Hardware support for the UART is minimized because one of the on-chip timers supplies the bit rate.

## Pin Description

**$\overline{AS}$**  *Address Strobe* (output, active Low).

Address Strobe is pulsed once at the beginning of each machine cycle. Address output via Port 1 for all external program or data memory transfers are valid at the trailing edge of  $\overline{AS}$ . Under program control,  $\overline{AS}$  can be placed in the high-impedance state along with Ports 0 and 1, Data Strobe and Read/Write.

**$\overline{DS}$**  *Data Strobe* (output, active Low). Data Strobe is activated once for each external memory transfer.

**P0<sub>0</sub>-P0<sub>7</sub>** *I/O Port Lines* (input/output, TTL compatible). 8 lines Nibble Programmable that can be configured under program control for I/O or external memory interface.

**P1<sub>0</sub>-P1<sub>7</sub>** *I/O Port Lines* (input/output, TTL compatible). 8 lines Byte Programmable that can be configured under program control for I/O or multiplexed address (A<sub>0</sub>-A<sub>7</sub>) and data (D<sub>0</sub>-D<sub>7</sub>) lines used to interface with program/data memory.

**P2<sub>0</sub>-P2<sub>7</sub>** *I/O Port Lines* (input/output, TTL compatible). 8 lines Bit Programmable. In addition they can be configured to provide open-drain outputs.

**P3<sub>0</sub>-P3<sub>4</sub>** *Input Port Lines* (TTL compatible). They can also be configured as control lines.

**P3<sub>5</sub>-P3<sub>7</sub>** *Output Port Lines* (TTL compatible). They can also be configured as control lines.

**$\overline{RESET}$**  *Reset* (input, active Low).  $\overline{RESET}$  initializes the Z8671. When  $\overline{RESET}$  is deactivated, program execution begins from internal program location 000C<sub>H</sub>.

**R/ $\overline{W}$**  *Read/Write* (output). R/ $\overline{W}$  is Low when the Z8671 is writing to external program or data memory.

**XTAL1, XTAL2** *Crystal 1, Crystal 2* (time-base input and output). These pins connect a parallel-resonant crystal (8 or 12 MHz maximum) or an external single-phase clock (8 or 12 MHz maximum) to the on-chip clock oscillator and buffer.



Z8671

### Address Spaces

**Program Memory.** The Z8671's 16-bit program counter can address 64K bytes of program memory space. Program memory consists of 2K bytes of internal ROM and up to 62K bytes of external ROM, EPROM, or RAM. The first 12 bytes of program memory are reserved for interrupt vectors (Figure 4). These locations contain six 16-bit vectors that correspond to the six available interrupts. The BASIC/Debug interpreter is located in the 2K bytes of internal ROM. The interpreter begins at address 12 and extends to 2047.

**Data Memory.** The Z8671 can address up to 62K bytes of external data memory beginning at location 2048 (Figure 5). External data memory may be included with, or separated from, the external program memory space. DM, an optional I/O function that can be programmed to appear on pin P3<sub>4</sub>, is used to distinguish data and program memory space.

**Register File.** The 144-byte register file may be accessed by BASIC programs as memory locations 0-127 and 240-255. The register

file includes four I/O port registers (R0-R3), 124 general-purpose registers (R4-R127), and 16 control and status registers (Figure 6).

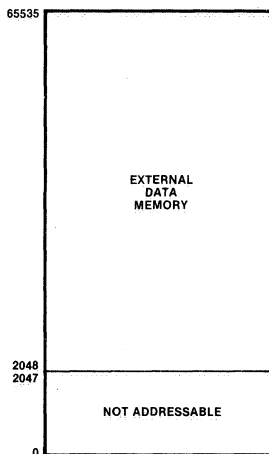


Figure 5. Data Memory Map

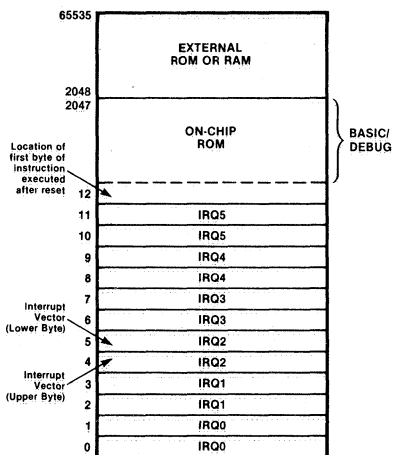


Figure 4. Programm Memory Map

LOCATION	IDENTIFIERS
255	STACK POINTER (BITS 7-0) SPL
254	STACK POINTER (BITS 15-8) SPH
253	REGISTER POINTER RP
252	PROGRAM CONTROL FLAGS FLAGS
251	INTERRUPT MASK REGISTER IMR
250	INTERRUPT REQUEST REGISTER IRQ
249	INTERRUPT PRIORITY REGISTER IPR
248	PORTS 0-1 MODE P01M
247	PORT 3 MODE P3M
246	PORT 2 MODE P2M
245	T0 PRESCALER PRE0
244	TIMER/COUNTER 0 TO
243	T1 PRESCALER PRE1
242	TIMER/COUNTER 1 T1
241	TIMER MODE TMR
240	SERIAL I/O SIO
	NOT IMPLEMENTED

Figure 6. Control and Status Registers



Address Spaces (Continued)

The BASIC/Debug Interpreter uses many of the general-purpose registers as pointers, scratch workspace, and internal variables. Consequently, these registers cannot be used by a machine language subroutine or other user programs. On power-up/Reset, BASIC/Debug searches for external RAM memory and checks for an auto start-up program. In a non-destructive method, memory is tested at relative location xxFD(hex). When BASIC/Debug discovers RAM in the system, it initializes the pointer registers to mark the boundaries between

areas of memory that are assigned specific uses. The top page of RAM is allocated for the line buffer, variable storage, and the GOSUB stack. Figure 7a illustrates the contents of the general-purpose registers in the Z8671 system with external RAM. When BASIC/Debug tests memory and finds no RAM, it uses an internal stack and shares register space with the input line buffer and variables. Figure 7b illustrates the contents of the general-purpose registers in the Z8671 system without external RAM.

127	EXPRESSION EVALUATION STACK
64	
63	FREF
34	
33	COUNTER
32	
31	USED INTERNALLY
30	SCRATCH
29	POINTER TO CONSTANT BLOCK
28	USED INTERNALLY
27	USED INTERNALLY
24	LINE NUMBER
23	ARGUMENT FOR SUBROUTINE
22	ARGUMENT/ROUTINE FOR SUBROUTINE CALL
21	SCRATCH
20	POINTER TO INPUT LINE BUFFER
19	POINTER TO END OF LINE BUFFER
18	POINTER TO STACK BOTTOM
17	ADDRESS OF USER PROGRAM
16	POINTER TO GOSUB STACK
15	POINTER TO END OF PROGRAM
14	POINTER TO END OF PROGRAM
13	POINTER TO END OF PROGRAM
12	POINTER TO END OF PROGRAM
11	POINTER TO END OF PROGRAM
10	POINTER TO END OF PROGRAM
9	POINTER TO END OF PROGRAM
8	POINTER TO END OF PROGRAM
7	POINTER TO END OF PROGRAM
6	POINTER TO END OF PROGRAM
5	POINTER TO END OF PROGRAM
4	POINTER TO END OF PROGRAM
3	POINTER TO END OF PROGRAM
0	I/O PORTS

Figure 7a. General-Purpose Registers with External RAM

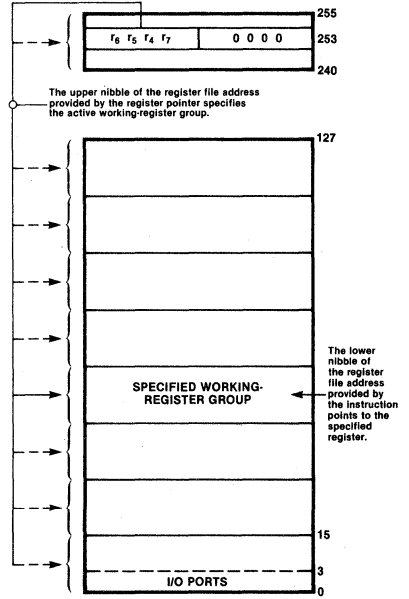
127	SHARED BY EXPRESSION STACK AND LINE BUFFER
104	
103	GOSUB STACK
86	
85	SHARED BY GOSUB AND VARIABLES
64	
63	VARIABLES
34	
33	FREE, AVAILABLE FOR USR ROUTINES
32	COUNTER
31	USED INTERNALLY
30	SCRATCH
29	POINTER TO CONSTANT BLOCK
28	USED INTERNALLY
27	USED INTERNALLY
24	LINE NUMBER
23	ARGUMENT FOR SUBROUTINE CALL
22	ARGUMENT/RESULT FOR SUBROUTINE CALL
21	SCRATCH
20	POINTER TO NEXT CHARACTER
19	POINTER TO LINE BUFFER
18	POINTER TO GOSUB
17	POINTER TO BASIC PROGRAM
16	POINTER TO GOSUB
15	POINTER TO GOSUB
14	POINTER TO GOSUB
13	POINTER TO GOSUB
12	POINTER TO GOSUB
11	POINTER TO GOSUB
10	POINTER TO GOSUB
9	POINTER TO GOSUB
8	POINTER TO GOSUB
7	POINTER TO GOSUB
6	POINTER TO GOSUB
5	FREE
4	FREE
3	FREE
0	I/O PORTS

Figure 7b. General-Purpose Registers without External RAM

**Address Spaces** (Continued)

**Stacks.** Either the internal register file or the external data memory can be used for the stack. A 16-bit Stack Pointer (R254 and R255) is used for the external stack, which can reside anywhere in data memory between location 2048 and 65535. An 8-bit Stack Pointer (R255) is used for the internal stack that resides within the 124 general-purpose registers (R4-R127).

**Register Addressing.** Z8671 instructions can access registers directly or indirectly with an 8-bit address field. The Z8671 also allows short 4-bit register addressing using the Register Pointer (one of the control registers). In the 4-bit mode, the register file is divided into nine working-register groups, each group consisting of 16 contiguous registers (Figure 8). The Register Pointer addresses the starting location of the active working-register group.


**Figure 8. The Register Pointer**
**Program Execution**

**Automatic Start-up.** The Z8671 has an automatic start-up capability which allows a program stored in ROM to be executed without operator intervention. Automatic execution occurs on power-on or Reset when the program is stored at address 1020 (hex).

**Execution Modes.** The Z8671's BASIC/Debug Interpreter operates in two execution modes: Run and Immediate. Programs are edited

and interactively debugged in the Immediate mode. Some BASIC/Debug commands are used almost exclusively in this mode. The Run mode is entered from the Immediate mode by entering the command RUN. If there is a program in RAM, it is executed. The system returns to the Immediate mode when program execution is complete or interrupted by an error.

## Interactive Debugging

Interactive debugging is accomplished with the self-contained line editor which operates in the Immediate mode. In addition to changing program lines, the editor can correct an immediate command before it is executed. It also allows the correction of typing and other errors as a program is entered.

BASIC/Debug allows interruptions and changes during a program run to correct

errors and add new instructions without disturbing the sequential execution of the program. A program run is interrupted with the use of the escape key. The run is restarted with a GOTO command (followed by the appropriate line number) after the desired changes are entered. The same procedure is used to enter corrections after BASIC/Debug returns an error.

## Commands

BASIC/Debug recognizes 15 command keywords. For detailed instructions of command usage, refer to the BASIC/Debug software manual.

**GO** The GO command unconditionally branches to a machine language subroutine. This statement is similar to the USR function except that no value is returned by the assembly language routine.

**GOSUB** GOSUB unconditionally branches to a subroutine at a line number specified by the user.

**GOTO** GOTO unconditionally changes the sequence of program execution (branches to a line number).

**IF/THEN** This command is used for conditional operations and branches.

**INPUT/IN** These commands request information from the user with the prompt "?", then read the input values (which must be separated by commas) from the keyboard, and store them in the indicated variables. INPUT discards any values remaining in the buffer from previous IN, INPUT, or RUN statements, and requests new data

from the operator. IN uses any values left in the buffer first, then requests new data.

**LET** LET assigns the value of an expression to a variable or memory location.

**LIST** This command is used in the interactive mode to generate a listing of program lines stored in memory on the terminal device.

**NEW** The NEW command resets pointer R10-11 to the beginning of user memory, thereby marking the space as empty and ready to store a new program.

**PRINT** PRINT lists its arguments, which may be text messages or numerical values, on the output terminal.

**REM** This command is used to insert explanatory messages into the program.

**RETURN** This command returns control to the line following a GOSUB statement.

**RUN** RUN initiates sequential execution of all instructions in the current program.

**STOP** STOP ends program execution and clears the GOSUB stack.





### Functions

BASIC/Debug supports two functions: AND and USR.

The AND function performs a logical AND. It can be used to mask, turn off, or isolate bits. This function is used in the following format:

AND (expression, expression)

The two expressions are evaluated, and their bit patterns are ANDed together. If only one value is included in the parentheses, it is ANDed with itself. A logical OR can also be performed by complementing the AND function. This is accomplished by subtracting each expression from -1. For example, the function below is equivalent to the OR of A and B.

$$-1 - \text{AND}(-1 - A, -1 - B)$$

The USR function calls a machine language subroutine and returns a value. This is useful for applications in which a subroutine can be performed more quickly and efficiently in machine language than in BASIC/Debug.

The address of the first instruction of the subroutine is the first argument of the USR function. The address can be followed by one or two values to be processed by the subroutine. In the following example, BASIC/Debug executes the subroutine located at address 2000 using values literal 256 and variable C.

USR(%2000,256,C)

The resulting value is stored in Registers 18-19.

### Serial Input/Output

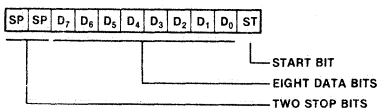
Port 3 lines P3<sub>0</sub> and P3<sub>7</sub> can be programmed as serial I/O lines for full-duplex serial asynchronous receiver/transmitter operation. The bit rate is controlled by Counter/Timer 0, with a maximum rate of 62.5K bit/second for 8MHz, and a maximum rate of 94.8K bit/second for 12MHz parts.

The Z8671 automatically adds a start bit and two stop bits to transmitted data (Figure 9). Odd parity is also available as an

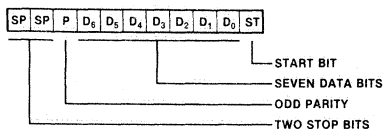
option. Eight data bits are always transmitted, regardless of parity selection. If parity is enabled, the eighth data bit is used as the odd parity bit. An interrupt request (IRQ<sub>4</sub>) is generated on all transmitted characters.

Received data must have a start bit, eight data bits, and at least one stop bit. If parity is on, bit 7 of the received data is replaced by a parity error flag. Received characters generate the IRQ<sub>3</sub> interrupt request.

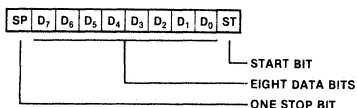
**Transmitted Data**  
(No Parity)



**Transmitted Data**  
(With Parity)



**Received Data**  
(No Parity)



**Received Data**  
(With Parity)

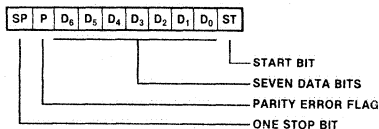


Figure 9. Serial Data Formats

## I/O Ports

The Z8671 has 32 lines dedicated to input and output. These lines are grouped into four ports of eight lines each and are configurable as input, output or address/data. Under software control, the ports can be programmed to provide address outputs, timing, status signals, serial I/O, and parallel I/O with or without handshake. All ports have active pull-ups and pull-downs compatible with TTL loads.

**Port 1** can be programmed as a byte I/O port or as an address/data port for interfacing external memory. When used as an I/O port, Port 1 may be placed under handshake control. In this configuration, Port 3 lines P<sub>33</sub> and P<sub>34</sub> are used as the handshake controls RDY<sub>1</sub> and DAV<sub>1</sub> (Ready and Data Available).

Memory locations greater than 2048 are referenced through Port 1. To interface external memory, Port 1 must be programmed for the multiplexed Address/Data mode. If more than 256 external locations are required, Port 0 must output the additional lines.

Port 1 can be placed in the high-impedance state along with Port 0,  $\overline{AS}$ ,  $\overline{DS}$  and R/W, allowing the Z8671 to share common resources in multiprocessor and DMA applications. Data transfers can be controlled by assigning P<sub>33</sub> as a Bus Acknowledge input and P<sub>34</sub> as a Bus Request output.

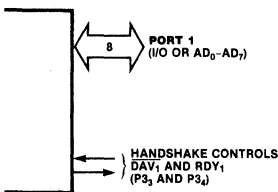


Figure 10a. Port 1

**Port 0** can be programmed as a nibble I/O port, or as an address port for interfacing external memory. When used as an I/O port, Port 0 may be placed under handshake control. In this configuration, Port 3 lines

P<sub>32</sub> and P<sub>35</sub> are used as the handshake controls  $\overline{DAV}_0$  and RDY<sub>0</sub>. Handshake signal assignment is dictated by the I/O direction of the upper nibble P<sub>04</sub>-P<sub>07</sub>.

For external memory references, Port 0 can provide address bits A<sub>8</sub>-A<sub>11</sub> (lower nibble) or A<sub>8</sub>-A<sub>15</sub> (lower and upper nibble) depending on the required address space. If the address range requires 12 bits or less, the upper nibble of Port 0 can be programmed independently as I/O while the lower nibble is used for addressing. When Port 0 nibbles are defined as address bits, they can be set to the high-impedance state along with Port 1 and the control signals  $\overline{AS}$ ,  $\overline{DS}$  and R/W.

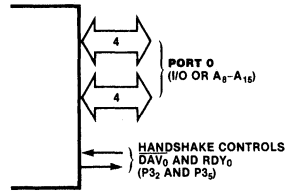


Figure 10b. Port 0

**Port 2** bits can be programmed independently as input or output. The port is always available for I/O operations. In addition, Port 2 can be configured to provide open-drain outputs.

Like Ports 0 and 1, Port 2 may also be placed under handshake control. In this configuration, Port 3 lines P<sub>31</sub> and P<sub>36</sub> are used as the handshake controls lines  $\overline{DAV}_2$  and RDY<sub>2</sub>. The handshake signal assignment for Port 3 lines P<sub>31</sub> and P<sub>36</sub> is dictated by the direction (input or output) assigned to bit 7 of Port 2.

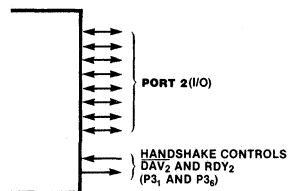
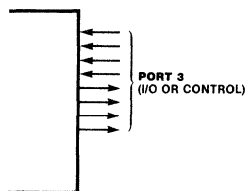


Figure 10c. Port 2

### I/O Ports (Continued)

**Port 3** lines can be configured as I/O or control lines. In either case, the direction of the eight lines is fixed as four input (P3<sub>0</sub>-P3<sub>3</sub>) and four output (P3<sub>4</sub>-P3<sub>7</sub>). For serial I/O, lines P3<sub>0</sub> and P3<sub>7</sub> are programmed as serial in and serial out respectively.

Port 3 can also provide the following control functions: handshake for Port 0, 1 and 2 ( $\overline{DAV}$  and RDY); four external interrupt request signals (IRQ<sub>0</sub>-IRQ<sub>3</sub>); timer input and output signals (T<sub>IN</sub> and T<sub>OUT</sub>) and Data Memory Select (DM).



**Figure 10d. Port 3**

### Counter/Timers

The Z8671 contains two 8-bit programmable counter/timers (T<sub>0</sub> and T<sub>1</sub>), each driven by its own 6-bit programmable prescaler. The T<sub>1</sub> prescalers can be driven by internal or external clock sources; however, the T<sub>0</sub> prescaler is driven by the internal clock only.

The 6-bit prescalers can divide the input frequency of the clock source by any number from 1 to 64. Each prescaler drives its counter, which decrements the value (1 to 256) that has been loaded into the counter. When the counter reaches the end of count, a timer interrupt request—IRQ<sub>4</sub> (T<sub>0</sub>) or IRQ<sub>5</sub> (T<sub>1</sub>)—is generated.

The counters can be started, stopped, restarted to continue, or restarted from the initial value. The counters can also be programmed to stop upon reaching zero (single-pass mode) or to automatically reload

the initial value and continue counting (modulo-n continuous mode). The counters, but not the prescalers, can be read any time without disturbing their value or count mode.

The clock source for T<sub>1</sub> is user-definable; it can be either the internal microprocessor clock (4 MHz maximum for the 8 MHz device and 6 MHz maximum for the 12 MHz device) divided by four, or an external signal input via Port 3. The Timer Mode register configures the external timer input as an external clock, a trigger input that can be retriggerable or non-retriggerable, or as a gate input for the internal clock. The counter/timers can be programmably cascaded by connecting the T<sub>0</sub> output to the input of T<sub>1</sub>. Port 3 line P3<sub>6</sub> also serves as a timer output (T<sub>OUT</sub>) through which T<sub>0</sub>, T<sub>1</sub> or the internal can be output.



## Interrupts

The Z8671 allows six different interrupts from eight sources: the four Port 3 lines P3<sub>0</sub>-P3<sub>3</sub>, Serial In, Serial Out, and the two counter/timers. These interrupts are both maskable and prioritized. The Interrupt Mask register globally or individually enables or disables the six interrupt requests. When more than one interrupt is pending, priorities are resolved by a programmable priority encoder that is controlled by the Interrupt Priority register.

All Z8671 interrupts are vectored; however, the internal UART operates in a polling fashion. To accommodate a polled structure, any or all of the interrupt inputs can be masked and the Interrupt Request register polled to determine which of the interrupt requests needs service.

The BASIC/Debug Interpreter does not process interrupts. Interrupts are vectored through locations in internal ROM which point to addresses 1000-1011 (hex). To process interrupts, jump instructions can be entered to the interrupt handling routines at the appropriate addresses as shown in Table 1.

Address (hex)	Contains Jump Instruction and Subroutine Address for:
1000-1002	IRQ <sub>0</sub>
1003-1005	IRQ <sub>1</sub>
1006-1008	IRQ <sub>2</sub>
1009-100B	IRQ <sub>3</sub>
100C-100E	IRQ <sub>4</sub>
100F-1011	IRQ <sub>5</sub>

**Table 1. Interrupt Jump instruction**

## Clock

The on-chip oscillator has a high-gain, parallel-resonant amplifier for connection to a crystal or to any suitable external clock source (XTAL1 = Input, XTAL2 = Output).

The crystal source is connected across XTAL1 and XTAL2, using the recommended

capacitance ( $C_L \leq 15$  pF maximum) from each pin to ground. The specifications for the crystal are as follows:

- AT cut, parallel resonant
- Fundamental type, 8 MHz maximum
- Series resistance,  $R_s \leq 100 \Omega$



### Instruction Set Notation

**Addressing Modes.** The following notation is used to describe the addressing modes and instruction operations as shown in the instruction summary.

- IRR** Indirect register pair or indirect working-register pair address
- Irr** Indirect working-register pair only
- X** Indexed address
- DA** Direct address
- RA** Relative address
- IM** Immediate
- R** Register or working-register address
- r** Working-register address only
- IR** Indirect-register or indirect working-register address
- Ir** Indirect working-register address only
- RR** Register pair or working register pair address

**Symbols.** The following symbols are used in describing the instruction set.

- dst** Destination location or contents
- src** Source location or contents
- cc** Condition code (see list)
- @** Indirect address prefix
- SP** Stack pointer (control registers 254-255)
- PC** Program counter
- FLAGS** Flag register (control register 252)
- RP** Register pointer (control register 253)

**IMR** Interrupt mask register (control register 251)

Assignment of a value is indicated by the symbol "←". For example,

$$dst \leftarrow dst + src$$

indicates that the source data is added to the destination data and the result is stored in the destination location. The notation "addr(n)" is used to refer to bit "n" of a given location. For example,

$$dst(7)$$

refers to bit 7 of the destination operand.

**Flags.** Control Register R252 contains the following six arithmetical flags plus two user selectable flags:

- C** Carry flag
  - Z** Zero flag
  - S** Sign flag
  - V** Overflow flag
  - D** Decimal-adjust flag
  - H** Half-carry flag
- |   |   |   |   |   |   |    |    |
|---|---|---|---|---|---|----|----|
| C | Z | S | V | D | H | F2 | F1 |
|---|---|---|---|---|---|----|----|

**F1** } user flags

**F2** }

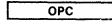
Affected flags are indicated by:

- 0** Cleared to zero
- 1** Set to one
- \*** Set or cleared according to operation
- Unaffected
- X** Undefined

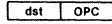
### Conditions Codes

Value	Mnemonic	Meaning	Flags Set
1000		Always true	---
0111	C	Carry	C = 1
1111	NC	No carry	C = 0
0110	Z	Zero	Z = 1
1110	NZ	Not zero	Z = 0
1101	PL	Plus	S = 0
0101	MI	Minus	S = 1
0100	OV	Overflow	V = 1
1100	NOV	No overflow	V = 0
0110	EQ	Equal	Z = 1
1110	NE	Not equal	Z = 0
1001	GE	Greater than or equal	(S XOR V) = 0
0001	LT	Less than	(S XOR V) = 1
1010	GT	Greater than	[Z OR (S XOR V)] = 0
0010	LE	Less than or equal	[Z OR (S XOR V)] = 1
1111	UGE	Unsigned greater than or equal	C = 0
0111	ULT	Unsigned less than	C = 1
1011	UGT	Unsigned greater than	(C = 0 AND Z = 0) = 1
0011	ULE	Unsigned less than or equal	(C OR Z) = 1
0000		Never true	---

## Instruction Formats

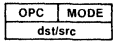


CCF, DI, EI, IRET, NOP,  
RCF, RET, SCF



INC r

### One-Byte Instructions

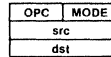


OR 

1	1	1	0
---	---	---	---

 dst/src

CLR, CPL, DA, DEC,  
DECW, INC, INCW, POP,  
PUSH, RL, RLC, RR,  
RRC, SRA, SWAP



OR 

1	1	1	0
---	---	---	---

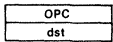
 src

OR 

1	1	1	0
---	---	---	---

 dst

ADC, ADD, AND, CP,  
LD, OR, SBC, SUB,  
TCM, TM, XOR

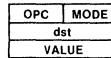


OR 

1	1	1	0
---	---	---	---

 dst

JP, CALL (Indirect)

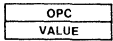


OR 

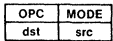
1	1	1	0
---	---	---	---

 dst

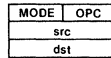
ADC, ADD, AND, CP,  
LD, OR, SBC, SUB,  
TCM, TM, XOR



SRP



ADC, ADD, AND,  
CP, OR, SBC, SUB,  
TCM, TM, XOR



OR 

1	1	1	0
---	---	---	---

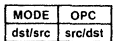
 src

OR 

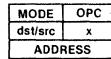
1	1	1	0
---	---	---	---

 dst

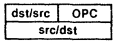
LD



LD, LDE, LDEI,  
LDC, LDCI



LD

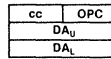


OR 

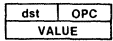
1	1	1	0
---	---	---	---

 src

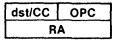
LD



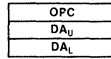
JP



LD



DJNZ, JR



CALL

Two-Byte instruction

Three-Byte instruction


Figure 11. Instruction Formats



Z8671

**Instruction Summary**

Instruction and Operation	Addr Mode		Opcode Byte (Hex)	Flags Affected					
	dst	src		C	Z	S	V	D	H
<b>ADC</b> dst,src dst - dst + src + C	(Note 1)		1□	*	*	*	*	0	*
<b>ADD</b> dst,src dst - dst + src	(Note 1)		0□	*	*	*	*	0	*
<b>AND</b> dst,src dst - dst AND src	(Note 1)		5□	-	*	*	0	-	-
<b>CALL</b> dst SP - SP - 2 @SP - PC; PC - dst	DA IRR		D6 D4	-	-	-	-	-	-
<b>CCF</b> C - NOT C			EF	*	-	-	-	-	-
<b>CLR</b> dst dst - 0	R IR		B0 B1	-	-	-	-	-	-
<b>COM</b> dst dst - NOT dst	R IR		60 61	-	*	*	0	-	-
<b>CP</b> dst,src dst - src	(Note 1)		A□	*	*	*	*	-	-
<b>DA</b> dst dst - DA dst	R IR		40 41	*	*	*	X	-	-
<b>DEC</b> dst dst - dst - 1	R IR		00 01	-	*	*	*	-	-
<b>DECW</b> dst dst - dst - 1	RR IR		80 81	-	*	*	*	-	-
<b>DI</b> IMR (7) - 0			8F	-	-	-	-	-	-
<b>DJNZ</b> r,dst r - r - 1 if r ≠ 0 PC - PC + dst Range: +127, -128	RA		rA r=0-F	-	-	-	-	-	-
<b>EI</b> IMR (7) - 1			9F	-	-	-	-	-	-
<b>INC</b> dst dst - dst + 1	r R IR		rE r=0-F 20 21	-	*	*	*	-	-
<b>INCW</b> dst dst - dst + 1	RR IR		A0 A1	-	*	*	*	-	-
<b>IRET</b> FLAGS - @SP; SP - SP + 1 PC - @SP; SP - SP + 2; IMR (7) - 1			BF	*	*	*	*	*	*

Instruction and Operation	Addr Mode		Opcode Byte (Hex)	Flags Affected					
	dst	src		C	Z	S	V	D	H
<b>JP</b> cc,dst if cc is true PC - dst	DA		cD c=0-F 30	-	-	-	-	-	-
<b>JR</b> cc,dst if cc is true, PC - PC + dst Range: +127, -128	RA		cB c=0-F	-	-	-	-	-	-
<b>LD</b> dst,src dst - src	r R	Im R r	rC r8 r9 r=0-F	-	-	-	-	-	-
	r X r Ir R R IR R IR IR	X r Ir r R R Im Im R	C7 D7 E3 F3 E4 E5 E6 E7 F5						
<b>LDC</b> dst,src dst - src	r Irr	Irr r	C2 D2	-	-	-	-	-	-
<b>LDCI</b> dst,src dst - src r - r + 1; rr - rr + 1	Ir Irr	Irr Ir	C3 D3	-	-	-	-	-	-
<b>LDE</b> dst,src dst - src	r Irr	Irr r	82 92	-	-	-	-	-	-
<b>LDEI</b> dst,src dst - src r - r + 1; rr - rr + 1	Ir Irr	Irr Ir	83 93	-	-	-	-	-	-
<b>NOP</b>			FF	-	-	-	-	-	-
<b>OR</b> dst,src dst - dst OR src	(Note 1)		4□	-	*	*	0	-	-
<b>POP</b> dst dst - @SP SP - SP + 1	R IR		50 51	-	-	-	-	-	-
<b>PUSH</b> src SP - SP - 1; @SP - src	R IR		70 71	-	-	-	-	-	-
<b>RCF</b> C - 0			CF	0	-	-	-	-	-
<b>RET</b> PC - @SP; SP - SP + 2			AF	-	-	-	-	-	-
<b>RL</b> dst		R IR	90 91	*	*	*	*	-	-



**Instruction Summary (Continued)**

Instruction and Operation	Addr Mode		Opcode Byte (Hex)	Flags Affected						
	dst	src		C	Z	S	V	D	H	
<b>RLC</b> dst	R		10	*	*	*	*	*	-	-
	IR		11	*	*	*	*	*	-	-
<b>RR</b> dst	R		E0	*	*	*	*	*	-	-
	IR		E1	*	*	*	*	*	-	-
<b>RRC</b> dst	R		C0	*	*	*	*	*	-	-
	IR		C1	*	*	*	*	*	-	-
<b>SBC</b> dst,src dst - dst - src - C	(Note 1)		3□	*	*	*	*	1	*	
<b>SCF</b> C - 1			DF	1	-	-	-	-	-	-
<b>SRA</b> dst	R		D0	*	*	*	0	-	-	
	IR		D1	*	*	*	0	-	-	

Instruction and Operation	Addr Mode		Opcode Byte (Hex)	Flags Affected						
	dst	src		C	Z	S	V	D	H	
<b>SRP</b> src RP - src		Im	31	-	-	-	-	-	-	-
<b>SUB</b> dst,src dst - dst - src	(Note 1)		2□	*	*	*	*	1	*	
<b>SWAP</b> dst	R	IR	F0 F1	X	*	*	X	-	-	
<b>TCM</b> dst,src (NOT dst) AND src	(Note 1)		6□	-	*	*	0	-	-	
<b>TM</b> dst, src dst AND src	(Note 1)		7□	-	*	*	0	-	-	
<b>XOR</b> dst,src 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 is found in the instruction set table above. The second nibble is expressed symbolically by a □ in this table, and its value is found in the following table to the left of the applicable addressing mode pair.

For example, the opcode of an ADC instruction using the addressing modes r (destination) and Ir (source) is 13.

Addr Mode		Lower Opcode Nibble
dst	src	
r	r	2
r	Ir	3
R	R	4
R	IR	5
R	IM	6
IR	IM	7

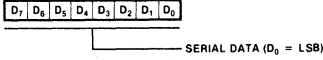




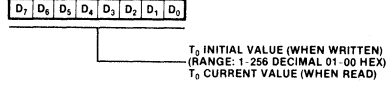
# Z8671

## Registers

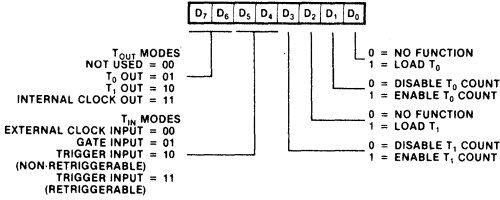
### R240 SIO Serial I/O Register (F0H; Read/Write)



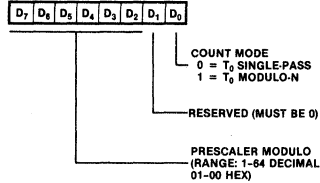
### R244 T0 Counter/Timer 0 Register (F4H; Read/Write)



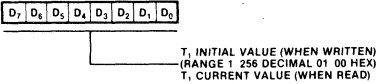
### R241 TMR Timer Mode Register (F1H; Read/Write)



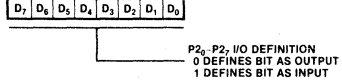
### R245 PRE0 Prescaler 0 Register (F5H; Write Only)



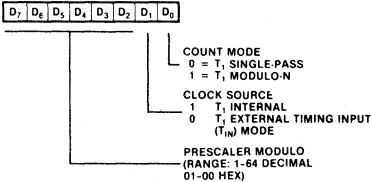
### R242 T1 Counter Timer 1 Register (F2H; Read/Write)



### R246 P2M Port 2 Mode Register (F6H; Write Only)



### R243 PRE1 Prescaler 1 Register (F3H; Write Only)



### R247 P3M Port 3 Mode Register (F7H; Write Only)

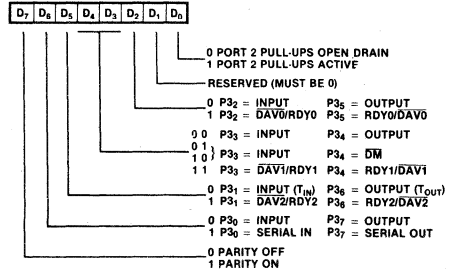
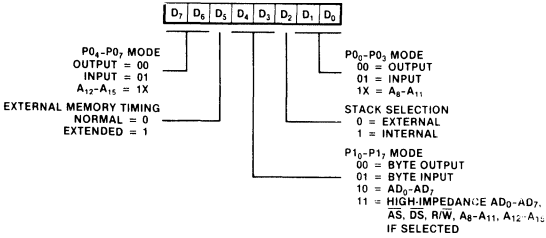
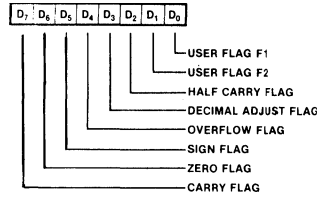
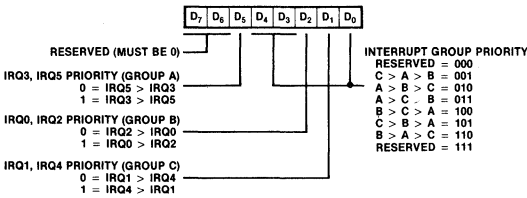
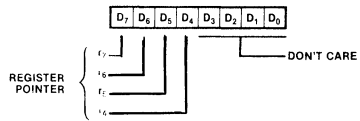
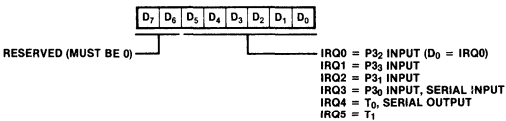
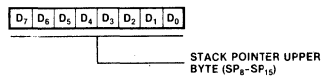
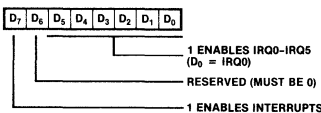
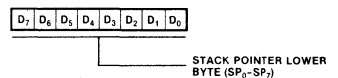


Figure 12. Control Registers

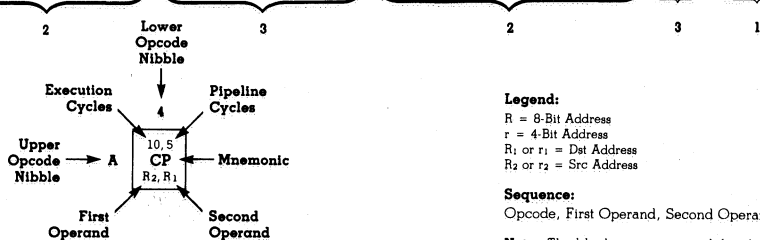
**Registers (Continued)**
**R248 P01M**  
**Port 0 and 1 Mode Register**  
 (F8<sub>H</sub>; Write Only)

**R252 FLAGS**  
**Flag Register**  
 (FC<sub>H</sub>; Read/Write)

**R249 IPR**  
**Interrupt Priority Register**  
 (F9<sub>H</sub>; Write Only)

**R253 RP**  
**Register Pointer**  
 (FD<sub>H</sub>; Read/Write)

**R250 IRQ**  
**Interrupt Request Register**  
 (FA<sub>H</sub>; Read/Write)

**R254 SHP**  
**Stack Pointer**  
 (FE<sub>H</sub>; Read/Write)

**R251 IMR**  
**Interrupt Mask Register**  
 (FB<sub>H</sub>; Read/Write)

**R255 SPL**  
**Stack Pointer**  
 (FF<sub>H</sub>; Read/Write)

**Figure 12. Control Registers (Continued)**

# Opcode Map

Lower Nibble (Hex)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
Upper Nibble (Hex)	0	6,5 DEC R <sub>1</sub>	6,5 DEC IR <sub>1</sub>	6,5 ADD r <sub>1</sub> , r <sub>2</sub>	6,5 ADD r <sub>1</sub> , IR <sub>2</sub>	10,5 ADD RR <sub>2</sub> , R <sub>1</sub>	10,5 ADD IR <sub>2</sub> , R <sub>1</sub>	10,5 ADD R <sub>1</sub> , IM	10,5 ADD IR <sub>1</sub> , IM	6,5 LD r <sub>1</sub> , R <sub>2</sub>	6,5 LD r <sub>2</sub> , R <sub>1</sub>	12/10,5 DJNZ r <sub>1</sub> , RA	12/10,0 JR cc, RA	6,5 LD r <sub>1</sub> , IM	12/10,0 JP cc, DA	6,5 INC r <sub>1</sub>	
	1	6,5 RLC R <sub>1</sub>	6,5 RLC IR <sub>1</sub>	6,5 ADC r <sub>1</sub> , r <sub>2</sub>	6,5 ADC r <sub>1</sub> , IR <sub>2</sub>	10,5 ADC R <sub>2</sub> , R <sub>1</sub>	10,5 ADC IR <sub>2</sub> , R <sub>1</sub>	10,5 ADC R <sub>1</sub> , IM	10,5 ADC IR <sub>1</sub> , IM								
	2	6,5 INC R <sub>1</sub>	6,5 INC IR <sub>1</sub>	6,5 SUB r <sub>1</sub> , r <sub>2</sub>	6,5 SUB r <sub>1</sub> , IR <sub>2</sub>	10,5 SUB R <sub>2</sub> , R <sub>1</sub>	10,5 SUB IR <sub>2</sub> , R <sub>1</sub>	10,5 SUB R <sub>1</sub> , IM	10,5 SUB IR <sub>1</sub> , IM								
	3	8,0 JP IRR <sub>1</sub>	6,1 SRP IM	6,5 SBC r <sub>1</sub> , r <sub>2</sub>	6,5 SBC r <sub>1</sub> , IR <sub>2</sub>	10,5 SBC R <sub>2</sub> , R <sub>1</sub>	10,5 SBC IR <sub>2</sub> , R <sub>1</sub>	10,5 SBC R <sub>1</sub> , IM	10,5 SBC IR <sub>1</sub> , IM								
	4	8,5 DA R <sub>1</sub>	8,5 DA IR <sub>1</sub>	6,5 OR r <sub>1</sub> , r <sub>2</sub>	6,5 OR r <sub>1</sub> , IR <sub>2</sub>	10,5 OR R <sub>2</sub> , R <sub>1</sub>	10,5 OR IR <sub>2</sub> , R <sub>1</sub>	10,5 OR R <sub>1</sub> , IM	10,5 OR IR <sub>1</sub> , IM								
	5	10,5 POP R <sub>1</sub>	10,5 POP IR <sub>1</sub>	6,5 AND r <sub>1</sub> , r <sub>2</sub>	6,5 AND r <sub>1</sub> , IR <sub>2</sub>	10,5 AND R <sub>2</sub> , R <sub>1</sub>	10,5 AND IR <sub>2</sub> , R <sub>1</sub>	10,5 AND R <sub>1</sub> , IM	10,5 AND IR <sub>1</sub> , IM								
	6	6,5 COM R <sub>1</sub>	6,5 COM IR <sub>1</sub>	6,5 TCM r <sub>1</sub> , r <sub>2</sub>	6,5 TCM r <sub>1</sub> , IR <sub>2</sub>	10,5 TCM R <sub>2</sub> , R <sub>1</sub>	10,5 TCM IR <sub>2</sub> , R <sub>1</sub>	10,5 TCM R <sub>1</sub> , IM	10,5 TCM IR <sub>1</sub> , IM								
	7	10/12,1 PUSH R <sub>2</sub>	12/14,1 PUSH IR <sub>2</sub>	6,5 TM r <sub>1</sub> , r <sub>2</sub>	6,5 TM r <sub>1</sub> , IR <sub>2</sub>	10,5 TM R <sub>2</sub> , R <sub>1</sub>	10,5 TM IR <sub>2</sub> , R <sub>1</sub>	10,5 TM R <sub>1</sub> , IM	10,5 TM IR <sub>1</sub> , IM								
	8	10,5 DECW RR <sub>1</sub>	10,5 DECW IR <sub>1</sub>	12,0 LDE r <sub>1</sub> , IRR <sub>2</sub>	18,0 LDEI IR <sub>1</sub> , IRR <sub>2</sub>												6,1 DI
	9	6,5 RL R <sub>1</sub>	6,5 RL IR <sub>1</sub>	12,0 LDE r <sub>2</sub> , IRR <sub>1</sub>	18,0 LDEI IR <sub>2</sub> , IRR <sub>1</sub>												6,1 EI
	A	10,5 INCW RR <sub>1</sub>	10,5 INCW IR <sub>1</sub>	6,5 CP r <sub>1</sub> , r <sub>2</sub>	6,5 CP r <sub>1</sub> , IR <sub>2</sub>	10,5 CP R <sub>2</sub> , R <sub>1</sub>	10,5 CP IR <sub>2</sub> , R <sub>1</sub>	10,5 CP R <sub>1</sub> , IM	10,5 CP IR <sub>1</sub> , IM								14,0 RET
	B	6,5 CLR R <sub>1</sub>	6,5 CLR IR <sub>1</sub>	6,5 XOR r <sub>1</sub> , r <sub>2</sub>	6,5 XOR r <sub>1</sub> , IR <sub>2</sub>	10,5 XOR R <sub>2</sub> , R <sub>1</sub>	10,5 XOR IR <sub>2</sub> , R <sub>1</sub>	10,5 XOR R <sub>1</sub> , IM	10,5 XOR IR <sub>1</sub> , IM								16,0 IRET
	C	6,5 RRC R <sub>1</sub>	6,5 RRC IR <sub>1</sub>	12,0 LDC r <sub>1</sub> , IRR <sub>2</sub>	18,0 LDCI IR <sub>1</sub> , IRR <sub>2</sub>									10,5 LD r <sub>1</sub> , x, R <sub>2</sub>			6,5 RCF
	D	6,5 SRA R <sub>1</sub>	6,5 SRA IR <sub>1</sub>	12,0 LDC r <sub>2</sub> , IRR <sub>1</sub>	18,0 LDCI IR <sub>2</sub> , IRR <sub>1</sub>	20,0 CALL* IRR <sub>1</sub>		20,0 CALL DA	10,5 LD r <sub>2</sub> , x, R <sub>1</sub>								6,5 SCF
	E	6,5 RR R <sub>1</sub>	6,5 RR IR <sub>1</sub>		6,5 LD r <sub>1</sub> , IR <sub>2</sub>	10,5 LD R <sub>2</sub> , R <sub>1</sub>	10,5 LD IR <sub>2</sub> , R <sub>1</sub>	10,5 LD R <sub>1</sub> , IM	10,5 LD IR <sub>1</sub> , IM								6,5 CCF
	F	8,5 SWAP R <sub>1</sub>	8,5 SWAP IR <sub>1</sub>		6,5 LD IR <sub>1</sub> , r <sub>2</sub>		10,5 LD R <sub>2</sub> , IR <sub>1</sub>										6,0 NOP

Bytes per Instruction



**Legend:**

R = 8-Bit Address  
 r = 4-Bit Address  
 R<sub>1</sub> or r<sub>1</sub> = Dst Address  
 R<sub>2</sub> or r<sub>2</sub> = Src Address

**Sequence:**

Opcode, First Operand, Second Operand

**Note:** The blank areas are not defined.

\*2-byte instruction; fetch cycle appears as a 3-byte instruction

### Absolute Maximum Ratings

Voltage on all pins with respect to GND . . .  $-0.3\text{ V to }+7.0\text{ V}$   
 Operating Ambient Temperature . . . . .  $0^{\circ}\text{C to }+70^{\circ}\text{C}$   
 Storage Temperature . . .  $-65^{\circ}\text{C to }+150^{\circ}\text{C}$

Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

### Standard Test Conditions

The characteristics below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND. Positive current flows into the

reference pin. Standard conditions are as follows:

- $+4.75\text{ V} \leq V_{CC} \leq +5.25\text{ V}$
- $\text{GND} = 0\text{ V}$
- $0^{\circ}\text{C} \leq +70^{\circ}\text{C}$

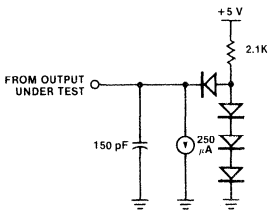


Figure 13. Test Load 1

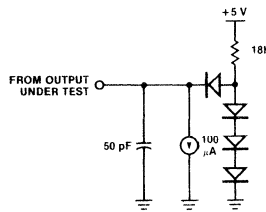


Figure 14. Test Load 2

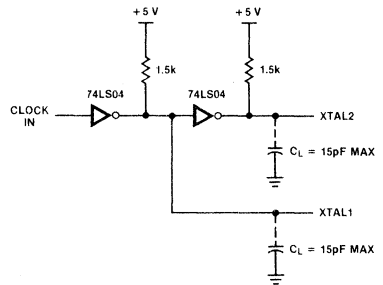


Figure 15. TTL External Clock Interface Circuit  
 (Both the clock and its complement are required)

### 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	2.0	$V_{CC}$	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	$I_{OH} = -250\ \mu\text{A}$
$V_{OL}$	Output Low Voltage		0.4	V	$I_{OL} = +2.0\ \text{mA}$
$I_{IL}$	Input Leakage	-10	10	$\mu\text{A}$	$0\text{ V} \leq V_{IN} \leq +5.25\text{ V}$
$I_{OL}$	Output Leakage	-10	10	$\mu\text{A}$	$0\text{ V} \leq V_{IN} \leq +5.25\text{ V}$
$I_{IR}$	Reset Input Current		-50	$\mu\text{A}$	$V_{CC} = +5.25\text{ V}, V_{RL} = 0\text{ V}$
$I_{CC}$	$V_{CC}$ Supply Current		120	mA	
$I_{MM}$	$V_{MM}$ Supply Current		10	mA	Power Down Mode
$V_{MM}$	Backup Supply Voltage	3	$V_{CC}$	V	Power Down



Z8671

External I/O or Memory Read/Write

No.	Symbol	Parameter	Z8671		Notes*†
			Min	Max	
1	TdA(AS)	Address Valid to $\overline{AS}$ $\uparrow$ Delay	50		1,2,3
2	TdAS(A)	$\overline{AS}$ $\uparrow$ to Address Float Delay	70		1,2,3
3	TdA(DS)	$\overline{AS}$ $\uparrow$ to Read Data Required Valid		360	1,2,3,4
4	TwAS	$\overline{AS}$ Low Width	80		1,2,3
5	TdAz(DS)	Address Float to $\overline{DS}$ $\downarrow$	0		1
6	TwDSR	$\overline{DS}$ (Read) Low Width	250		1,2,3,4
7	TwDSW	$\overline{DS}$ (Write) low Width	160		1,2,3,4
8	TdDSR(DR)	$\overline{DS}$ $\downarrow$ to Read Data Required Valid		200	1,2,3,4
9	ThDR(DS)	Read Data to $\overline{DS}$ $\uparrow$ Hold Time	0		1
10	TdDS(A)	$\overline{DS}$ $\uparrow$ to Address Active Delay	70		1,2,3
11	TdDS(AS)	$\overline{DS}$ $\uparrow$ to $\overline{AS}$ $\downarrow$ Delay	70		1,2,3
12	TdR/W(AS)	R/W Valid to $\overline{AS}$ $\uparrow$ Delay	50		1,2,3
13	TdDS(R/W)	$\overline{DS}$ $\uparrow$ to R/W Not Valid	60		1,2,3
14	TdDW(DSW)	Write Data Valid to $\overline{DS}$ (Write) $\downarrow$ Delay	50		1,2,3
15	TdDS(DW)	$\overline{DS}$ $\uparrow$ to Write data Not Valid Delay	70		1,2,3
16	TdA(DR)	Address valid to Read Data Required Valid		410	1,2,3,4
17	TdAS(DS)	$\overline{AS}$ $\uparrow$ to $\overline{DS}$ $\downarrow$ Delay	80		1,2,3

NOTES:

1. Test Load 1
2. Timing numbers given are for minimum TpC
3. Also see clock cycle time dependent characteristics table.
4. When using extended memory timing add 2 TpC.

5. All timing references use 2.0 V for a logic "1" and 0.8 for a logic "0".
- \* All units in nanoseconds (ns).
- † All timings are preliminary and subject to change.

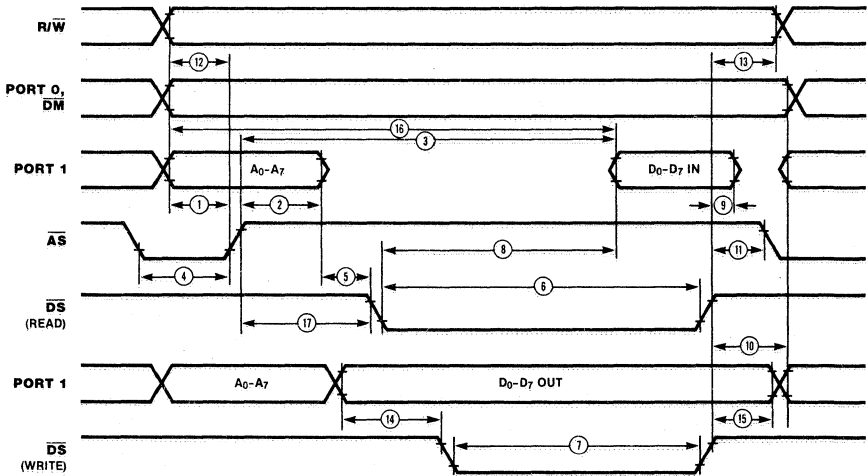


Figure 15. External I/O or Memory Read/Write



**Additional Timing**

No.	Symbol	Parameter	Z8671		Notes †
			Min	Max	
1	TpC	Input Clock Period	125	1000	1
2	TrC,TfC	Clock Input Rise And Fall Times		25	1
3	TwC	Input Clock Width	37		1
4	TwTinL	Timer Input low Width	100		2
5	TwTinH	Timer Input High Width	3TpC		2
6	TpTin	Timer Input Period	8TpC		2
7	TrTin,TfTin	Timer input Rise And Fall Times		100	2
8a	TwIL	Interrupt Request Input Low Time	100		2,3
8b	TwIH	Interrupt Request Input Low Time	3TpC		2,4
9	TwIH	Interrupt Request Input High Time	3TpC		2,3

NOTES:

1. Clock timing references uses 3.8 V for a logic "1" and 0.8 V for a logic "0"
2. Timing reference uses 2.0 V for a logic "1" and 0.8 V for a logic "0"
3. Interrupt request via Port 3 (P3<sub>1</sub>-P3<sub>3</sub>).
4. Interrupt request via Port 3 (P3<sub>0</sub>).
- \* Units in nanoseconds (ns).
- † All timings are preliminary and subject to change.

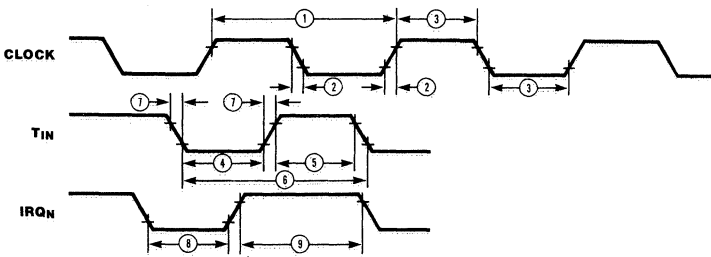


Figure 16. Additional Timing

**Memory Port Timing**

No.	Symbol	Parameter	Z8671		Notes †
			Min	Max	
1	TdA(DI)	Address Valid to Data Input Delay		460	1,2
2	ThDI(A)	Data In Hold Time	0		1

NOTES:

1. Test Load 2
2. This is a Clock-Cycle-Dependent parameter. For clock frequencies other than the maximum, use the following formula:  
 $Z8671 = 5 \text{ TpC} - 165$ ;  $Z8671-12 = 5 \text{ TpC} - 95$
- \* Units are nanoseconds unless otherwise specified; timing are preliminary and subject to change.

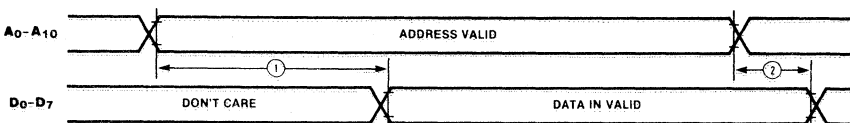


Figure 17. Memory Port Timing



Z8671

Handshake Timing

No.	Symbol	Parameter	Z8671		Notes †
			Min	Max	
1	TsDI(DAV)	Data In Setup Time	0		
2	ThDI(DAV)	Data In Hold Time	230		
3	TwDAV	Data Available Width	175		
4	TdDAVIf(RDY)	$\overline{DAV}$ ↓ Input to RDY ↓ Delay		175	1,2
5	TdDAVOf(RDY)	$\overline{DAV}$ ↓ Output to RDY ↓ Delay	0		1,3
6	TdDAVIr(RDY)	$\overline{DAV}$ ↑ Input to RDY ↑ Delay		175	1,2
7	TdDAVOr(RDY)	$\overline{DAV}$ ↑ Output to RDY ↑ Delay	0		1,3
8	TdDO(DAV)	Data Out to $\overline{DAV}$ ↓ Delay	50		1
9	TdRDY(DAV)	Rdy ↓ input to $\overline{DAV}$ ↑ Delay	0	200	1

NOTES:

1. Test load 1
2. Input handshake
3. Output handshake
4. All timing references use 2.0 V for a logic "1" and 0.8 V for a logic "0".

\* Units in nanoseconds (ns).

† All timings are preliminary and subject to change.

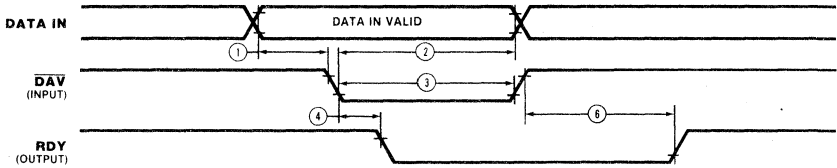


Figure 17a. Input Handshake

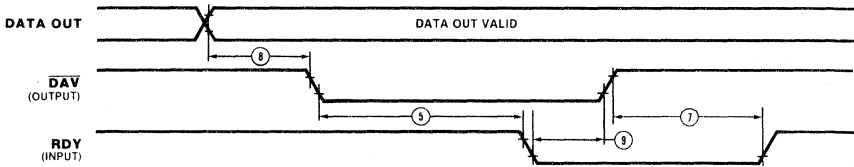
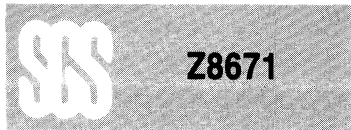


Figure 17b. Output Handshake



### Clock-Cycle-Time-Dependent Characteristics

Number	Symbols	Z8671	
		Equation	
1	TdA(AS)	TpC-75	
2	TdAS(A)	TpC-55	
3	TdAS(DR)	4TpC-140*	
4	TwAS	TpC-45	
6	TwDSR	3TpC-125*	
7	TwDSW	2TpC-90*	
8	TdDSR(DR)	3TpC-175*	
10	Td(DS)A	TpC-55	
11	TdDS(AS)	TpC-55	
12	TdR/W(AS)	TpC-75	
13	TdDS(R/W)	TpC-65	
14	TdDW(DSW)	TpC-75	
15	TdDS(DW)	TpC-55	
16	TdA(DR)	5TpC-215*	
17	TdAS(DS)	TpC-45	

\* Add 2TpC when using extended memory timing.

### Ordering Information

Type	Package	Temp.	Clock	Description
Z8671 B1	Plastic	0/+70°C	8 MHz	Z8 Mask programmed with BASIC/Debug Interpreter
Z8671 B6	Plastic	-40/+85°C		
Z8671 D1	Ceramic	0/+70°C		
Z8671 D6	Ceramic	-40/+85°C		
Z8671 C1	Plastic Chip Carrier	0/+70°C		
Z8671 C6	Plastic Chip Carrier	-40/+85°C		
Z8671 K1	Ceramic Chip Carrier	0/+70°C		
Z8671 K6	Ceramic Chip Carrier	-40/+85°C		







**Z8681/L**  
**Z8682/L**  
**Z8684/L**

## Z8 ROMless Microcomputer

- Complete microcomputer, 24 I/O lines, and up to 64K bytes of addressable external space each for program and data memory.
- 143-byte register file, including 124 general-purpose registers, three I/O port registers, and 16 status and control registers.
- Vectored, priority interrupts for I/O, counter/timers, and UART.
- On-chip oscillator that accepts crystal or external clock drive.
- Full-duplex UART and two programmable 8-bit counter/timers, each with a 6-bit programmable prescaler.
- Register Pointer so that short, fast instructions can access any one of the nine working-register groups.
- Low-power standby option that retains contents of general-purpose registers.
- Single +5 V power supply-all I/O pins TTL compatible.
- Available in 8 and 12 MHz versions.
- Low Power version:
  - available 8 MHz
  - current consumption 80 mA.

### General Description

The Z8681, Z8682 and Z8684 are ROMless versions of the Z8 single-chip microcomputer. The Z8682/4 are usually more cost effective. These products differ only slightly and can be used interchangeably with proper system design to provide maximum flexibility in meeting price and delivery needs. The Z8681/2/4 offers all the outstanding features of the Z8 family architecture except an on-chip program ROM. Use of external memory rather than a preprogrammed ROM enables this Z8 microcomputer to be used in low volume applications or where code flexibility is required.

The Z8681/2/4 can provide up to 16 output address lines, thus permitting an address space of up to 64K bytes of data or program memory. Eight address outputs ( $AD_0-AD_7$ ) are provided by a multiplexed, 8-bit, Address/Data bus. The remaining 8 bits can be provided by the software configuration of Port 0 to output address bits  $A_8-A_{15}$ .

Available address space can be doubled (up to 128K bytes for the Z8681, 124K bytes for the Z8682 and 120K bytes for the Z8684) by programming bit 4 of Port 3 ( $P_{34}$ ) to act as a data memory select output (DM).

The two states of DM together with the 16 address outputs can define separate data and memory address spaces of up to 64K/62K/60K bytes each.

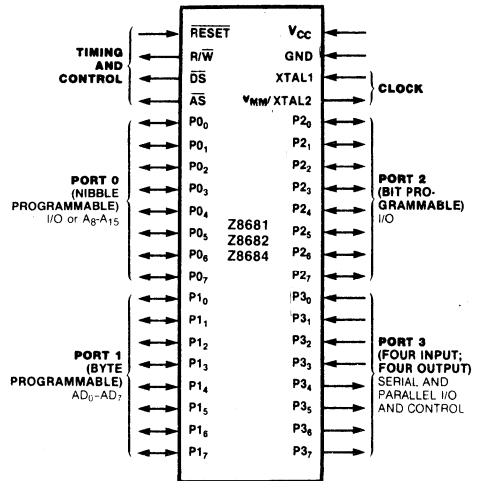


Figure 1. Logic Function

## General Description (Continued)

There are 143 bytes of RAM located on-chip and organized as a register file of 124 general-purpose registers, 16 control and status registers, and three I/O port registers. This register file can be divided into nine

groups of 16 working registers each. Configuring the register file in this manner allows the use of short format instructions; in addition, any of the individual registers may be accessed directly.

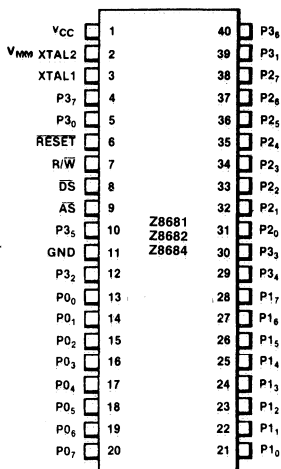


Figure 2a. DIP Pin Configuration

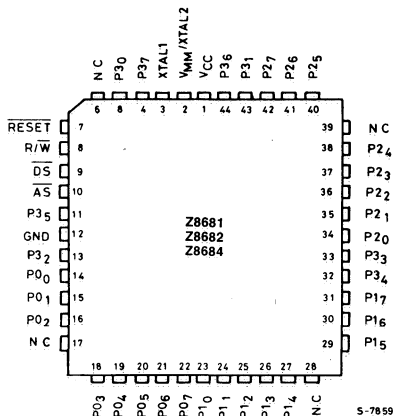


Figure 2b. Chip Carrier Pin Configuration

## Architecture

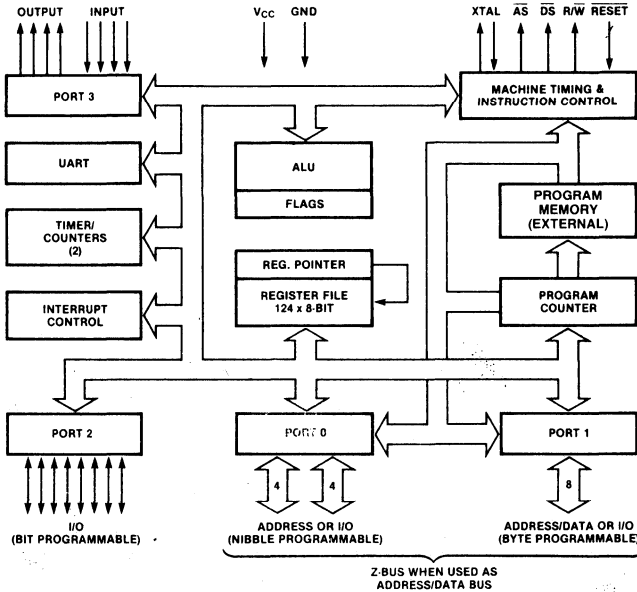
Z8681/2/4 architecture is characterized by a flexible I/O scheme, an efficient register and address space structure and a number of ancillary features that are helpful in many applications.

Microcomputer applications demand powerful I/O capabilities. The Z8681/2/4 fulfills this with 24 pins available for input and output. These lines are grouped into three ports of eight lines each and are configurable under software control to provide timing, status signals, serial or parallel I/O with or without handshake, and an Address bus for interfacing external memory.

Three basic address space are available: program memory, data memory and the register file (internal). The 143-byte random-access register file is composed of 124 general-purpose registers, three I/O port registers, and 16 control and status registers.

To unburden the program from coping with real-time problems such as serial data communication and counting/timing, an asynchronous receiver/transmitter (UART) and two counter/timers with a large number of user-selectable modes are offered on-chip. Hardware support for the UART is minimized because one of the on-chip timers supplies the bit rate.

**Architecture** (Continued)



**Figure 3. Functional Block Diagram**

**Pin Description**

**AS** *Address Strobe* (output, active Low). Address Strobe is pulsed once at the beginning of each machine cycle. Addresses output via Port 1 for all external program or data memory transfers are valid at the trailing edge of AS.

**DS** *Data Strobe* (output, active Low). Data Strobe is activated once for each external memory transfer.

**P0<sub>0</sub>-P0<sub>7</sub>** *I/O Port Lines* (input/output, TTL compatible). 8 lines Nibble Programmable that can be configured under program control for I/O or external memory interface.

**P1<sub>0</sub>-P1<sub>7</sub>** *Address/Data Port* (bidirectional). Multiplexed address (A<sub>0</sub>-A<sub>7</sub>) and data (D<sub>0</sub>-D<sub>7</sub>) lines used to interface with program and data memory.

**P2<sub>0</sub>-P2<sub>7</sub>** *I/O Port Lines* (input/output, TTL compatible). 8 lines Bit Programmable.

In addition they can be configured to provide open drain outputs.

**P3<sub>0</sub>-P3<sub>4</sub>** *Input Port Lines* (TTL compatible). They can also be configured as control lines.

**P3<sub>5</sub>-P3<sub>7</sub>** *Output Port Lines* (TTL compatible). They can also be configured as control lines.

**RESET\*** *Reset* (input, active Low). RESET initializes the Z8681/2/4. When RESET is deactivated, program execution begins from program location 000C<sub>H</sub> for the Z8681, 0812<sub>H</sub> for the Z8682 and 1012<sub>H</sub> for the Z8684.

**R/W** *Read/Write* (output). R/W is Low when the Z8681/2/4 is writing to external program or data memory.

**XTAL1, XTAL2** *Crystal 1, Crystal 2* (time-base input and output). These pins connect a parallel-resonant crystal to the on-chip clock oscillator and buffer.



Z8681/L  
Z8682/L  
Z8684/L

## Summary of Z8681, Z8682 and Z8684 Differences

Feature	Z8681	Z8682	Z8684
Address of first instruction executed after Reset	12	2066	4114
Addressable memory space	0-64K	2K-64K	4K-64K
Address of interrupt vectors	0-11	2048-2065	4096-4113
Reset input high voltage	TTL levels*	7.35-8.0 V	7.35-8.0V
Port 0 configuration after Reset	Input, float after reset. Can be programmed as Address bits.	Output, configured as Address bits $A_8-A_{15}$	Output, configured as Address bits $A_8-A_{15}$
External memory timing start-up configurations	Extended Timing	Normal Timing	Normal Timing
Interrupt vectors	2 byte vectors point directly to service routines	2 byte vectors in internal ROM point to 3 byte Jump instructions, which point to service routines.	2 byte vectors in internal ROM point to 3 byte Jump instructions, which point to service routines.
Interrupt response time	26 $\mu$ sec	36 $\mu$ sec	36 $\mu$ sec

\* 8.0 V  $V_{IN}$  max.

## Address Spaces

**Program Memory.\*** The Z8681/2/4 addresses 64K/62K/60K bytes of external program memory space (Figure 4).

For the Z8681, the first 12 bytes of program memory are reserved for the interrupt vectors. These locations contain six 16-bit vectors that correspond to the six available interrupts. Program execution begins at location 000C<sub>H</sub> after a reset.

The Z8682 has six 24-bit interrupt vectors beginning at address 0800<sub>H</sub>. The vectors consist of Jump Absolute instructions. After a reset, program execution begins at location 0812<sub>H</sub> for the Z8682.

The Z8684 has six 24-bit interrupt vectors beginning at address 1000<sub>H</sub>. The vectors consist of Jump Absolute instructions. After a reset, program execution begins at location 1012<sub>H</sub> for the Z8684.

**Data Memory.\*** The Z8681/2/4 can address 64K/62K/60K bytes of external data memory. External data memory may be included with

or separated from the external program memory space.  $\overline{DM}$ , an optional I/O function that can be programmed to appear on pin P3<sub>4</sub>, is used to distinguish between data and program memory space.

**Register File.** The 143-byte register file includes three I/O port registers (R0, R2, R3), 124 general-purpose registers (R4-R127) and 16 control and status registers (R240-R255). These registers are assigned the address locations shown in Figure 5.

Z8681/2/4 instructions can access registers directly or indirectly with an 8-bit address field. This also allows short 4-bit register addressing using the Register Pointer (one of the control registers). In the 4-bit mode, the register file is divided into nine working-register groups, each occupying 16 contiguous locations (Figure 5). The Register Pointer addresses the starting location of the active working-register group (Figure 6).

\*This feature differs in the Z8681, Z8682 and Z8684



Z8681/L  
Z8682/L  
Z8684/L

### Address Spaces (Continued)

**Stacks.** Either the internal register file or the external data memory can be used for the stack. A 16-bit Stack Pointer (R254 and R255) is used for the external stack, which

can reside anywhere in data memory. An 8-bit Stack Pointer (R255) is used for the internal stack that resides within the 124 general-purpose registers (R4-R127).

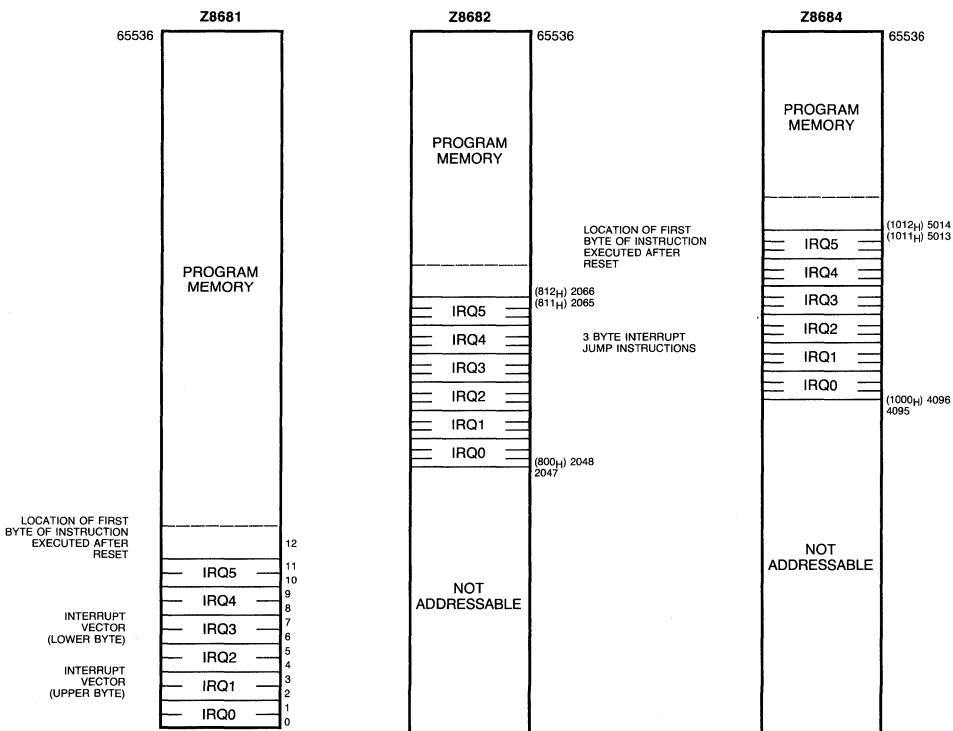
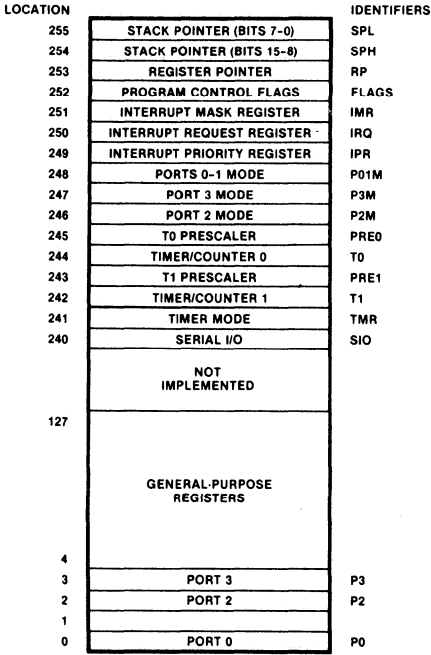


Figure 4. Z8681/2/4 Program Memory Map

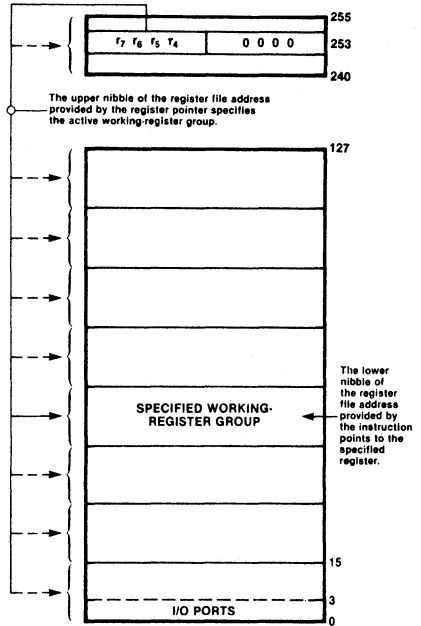


**Z8681/L**  
**Z8682/L**  
**Z8684/L**

**Address Spaces (Continued)**



**Figure 5. The Register File**



**Figure 6. The Register Pointer**

**Serial Input/Output**

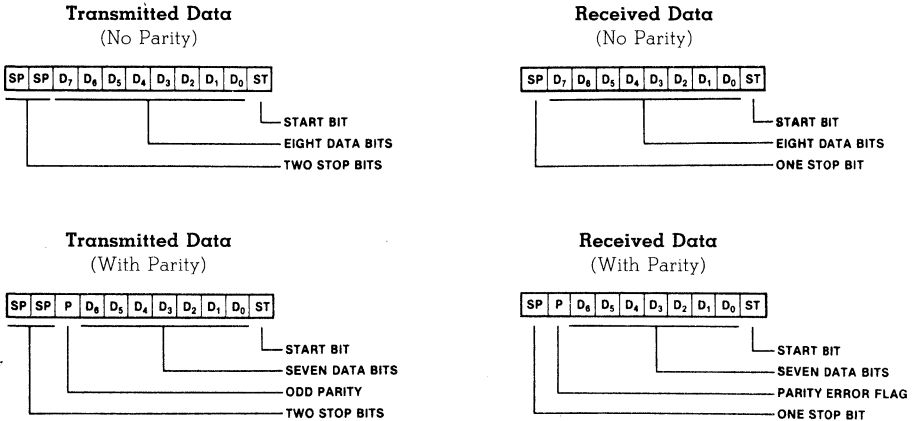
Port 3 lines P3<sub>0</sub> and P3<sub>7</sub> can be programmed as serial I/O lines for full-duplex serial asynchronous receiver/transmitter operation. The bit rate is controlled by Counter/Timer 0, with a maximum rate of 62.5K bits/second at 8 MHz and 93.75K bits/second at 12 MHz.

The Z8681/2/4 automatically adds a start bit and two stop bits to transmitted data (Figure 7). Odd parity is also available as an option.

Eight data bits are always transmitted, regardless of parity selection. If parity is enabled, the eighth data bit is used as the odd parity bit. An interrupt request (IRQ<sub>4</sub>) is generated on all transmitted characters.

Received data must have a start bit, eight data bits and at least one stop bit. If parity is on, bit 7 of the received data is replaced by a parity error flag. Received characters generate the IRQ<sub>3</sub> interrupt request.

**Serial Input/Output (Continued)**



**Figure 7. Serial Data Formats**

**Counter/Timers**

The Z8681/2/4 contains two 8-bit programmable counter/timers ( $T_0$  and  $T_1$ ), each driven by its own 6-bit programmable prescaler. The  $T_1$  prescaler can be driven by internal or external clock sources; however, the  $T_0$  prescaler is driven by the internal clock only.

The 6-bit prescalers can divide the input frequency of the clock source by any number from 1 to 64. Each prescaler drives its counter, which decrements the value (1 to 256) that has been loaded into the counter. When the counter reaches the end of count, a timer interrupt request –  $IRQ_4$  ( $T_0$ ) or  $IRQ_5$  ( $T_1$ ) – is generated.

The counters can be started, stopped, restarted to continue, or restarted from the initial value. The counters can also be programmed to stop upon reaching zero

(single-pass mode) or to automatically reload the initial value and continue counting (modulo-n continuous mode). The counters, but not the prescalers, can be read any time without disturbing their value or count mode.

The clock source for  $T_1$  is user-definable; it can be either the internal microprocessor clock divided by four, or an external signal input via Port 3. The Timer Mode register configures the external timer input as an external clock, a trigger input that can be retriggerable or non-retriggerable, or as a gate input for the internal clock. The counter/timers can be programmably cascaded by connecting the  $T_0$  output to the input of  $T_1$ . Port 3 line P3<sub>6</sub> also serves as a timer output ( $T_{OUT}$ ) through which  $T_0$ ,  $T_1$  or the internal clock can be output.





Z8681/L  
Z8682/L  
Z8684/L

## I/O Ports

The Z8681/2/4 has 24 lines available for input and output. These lines are grouped into three ports of eight lines each and are configurable as input, output or address. Under software control, the ports can be programmed to provide address outputs, timing, status signals, serial I/O, and parallel I/O with or without handshake. All ports have active pull-ups and pull-downs compatible with TTL loads.

**Port 1** is a dedicated Z-BUS compatible memory interface. The operations of Port 1 are supported by the Address Strobe ( $\overline{AS}$ ) and Data Strobe ( $\overline{DS}$ ) lines, and by the Read/Write ( $R/\overline{W}$ ) and Data Memory ( $\overline{DM}$ ) control lines. The low-order program and data memory addresses ( $A_0-A_7$ ) are output through Port 1 (Figure 8) and are multiplexed with data in/out ( $D_0-D_7$ ). Instruction fetch and data memory read/write operations are done through this port.

Port 1 cannot be used as a register nor can a handshake mode be used with this port.

The Z8681, the Z8682 and the Z8684 wake up with the 8 bits of Port 1 configured as address outputs for external memory. If more than eight address line are required with the Z8681, additional lines can be obtained by programming Port 0 bits as address bits. The least-significant four bits of Port 0 can be configured to supply address bits  $A_8-A_{11}$  for 4K byte addressing or both nibbles of Port 0 can be configured to supply address bits  $A_8-A_{15}$  for 64K byte addressing.

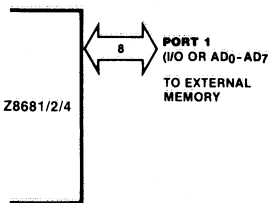


Figure 8. Port 1

**Port 0\*** can be programmed as a nibble I/O port, or as an address port for interfacing external memory (Figure 9). When used as an I/O port, Port 0 may be placed under handshake control. In this configuration, Port 3 lines  $P3_2$  and  $P3_5$  are used as the handshake controls  $DAV_0$  and  $RDY_0$ . Handshake signal assignment is dictated by the I/O direction of the upper nibble  $P0_4-P0_7$ .

For external memory references, Port 0 can provide address bits  $A_8-A_{11}$  (lower nibble) or  $A_8-A_{15}$  (lower and upper nibbles) depending on the required address space. If the address range requires 12 bits or less, the upper nibble of Port 0 can be programmed independently as I/O while the lower nibble is used for addressing.

In the Z8681\*, Port 0 lines float after reset; their logic state is unknown until the execution of an initialization routine that configures Port 0.

Such an initialization routine must reside within the first 256 bytes of executable code and must be physically mapped into memory by forcing the Port 0 address lines to a known state. See Figure 10. The proper Port initialization sequence is:

1. Write initial address ( $A_8-A_{15}$ ) of initialization routine to Port 0 address lines.
2. Configure Port 0 Mode Register to output  $A_8-A_{15}$  (or  $A_8-A_{11}$ ).

To permit the use of slow memory, an automatic wait mode of two oscillator clock cycles is configured for the bus timing of the Z8681 after each reset. The initialization

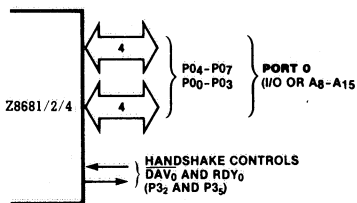


Figure 9. Port 0

\* This feature differs in the Z8681, Z8682 and Z8684

I/O Ports (Continued)

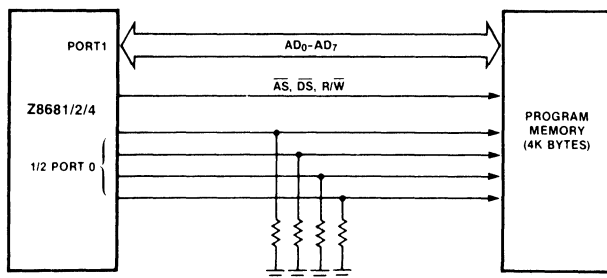


Figure 10. Port 0 Address Lines Tied to Logic 0

routine could include reconfiguration to eliminate this extended timing mode.

The following example illustrates the manner in which an initialization routine can be mapped in a Z8681 system with 4K of memory.

*Example.* In Figure 10, the initialization routine is mapped to the first 256 bytes of program memory. Pull-down resistors maintain the address lines at a logic 0 level when these lines are floating. The leakage current caused by fanout must be taken into consideration when selecting the value of the pull-down resistors. The resistor value must be large enough to allow the Port 0 output driver to pull the line to a logic one.

Generally, pull-down resistors are incompatible with TTL loads. If Port 0 drives into TTL input loads ( $I_{LOW} = 1.6 \text{ ma}$ ) the external resistors should be tied to  $V_{CC}$  and the initialization routine put in address space  $FF00_H - FFFF_H$ .

In the Z8682/4\*, Port 0 lines are configured as address lines  $A_8 - A_{15}$  after a Reset. If one or both nibbles are needed for I/O operation, they must be configured by writing to the Port 0 Mode register. The Z8682/4 is in the fast memory timing mode

after Reset, so the initialization routine must be in fast memory.

Port 2 bits can be programmed independently as input or output (Figure 11). This port is always available for I/O operations. In addition, Port 2 can be configured to provide open-drain outputs.

Like Port 0, Port 2 may also be placed under handshake control. In this configuration, Port 3 lines  $P_{31}$  and  $P_{36}$  are used as the handshake controls lines  $DAV_2$  and  $RDY_2$ . The handshake signal assignment for Port 3 lines  $P_{31}$  and  $P_{36}$  is dictated by the direction (input or output) assigned to bit 7 of Port 2.

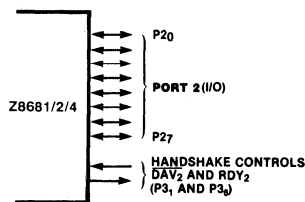


Figure 11. Port 2

\* This feature differs in the Z8681, Z8682 and Z8684



**Z8681/L**  
**Z8682/L**  
**Z8684/L**

## I/O Ports (Continued)

**Port 3** lines can be configured as I/O or control lines (Figure 12). In either case, the direction of the eight lines is fixed as four input (P3<sub>0</sub>-P3<sub>3</sub>) and four output (P3<sub>4</sub>-P3<sub>7</sub>). For serial I/O, lines P3<sub>0</sub> and P3<sub>7</sub> are programmed as serial in and serial out, respectively.

Port 3 can also provide the following control functions: handshake for Ports 0 and 2 (DAV and RDY); four external interrupt request signals (IRQ<sub>0</sub>-IRQ<sub>3</sub>); timer input and output signals (T<sub>IN</sub> and T<sub>OUT</sub>) and Data Memory Select (DM).

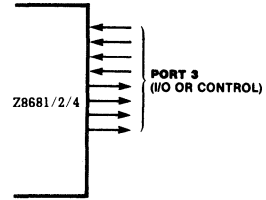


Figure 12. Port 3

## Interrupts\*

The Z8681/2/4 allows six different interrupts from eight sources: the four Port 3 lines P3<sub>0</sub>-P3<sub>3</sub>, Serial In, Serial Out, and the two counter/timers. These interrupts are both maskable and prioritized. The interrupt Mask register globally or individually enables or disables the six interrupt requests. When more than one interrupt is pending, priorities are resolved by a programmable priority encoder that is controlled by the Interrupt Priority register.

All Z8681, Z8682 and Z8684 interrupts are vectored through locations in program memory. When an interrupt request is granted, an interrupt machine cycle is entered. This disables all subsequent interrupts, saves the Program Counter and status flags, and access the program memory vector location reserved for that interrupt. In the Z8681, this memory location and the next byte contain the 16-bit address of the interrupt service routine for that particular interrupt request. The Z8681 takes 26 system clock cycles to enter an interrupt subroutine.

The Z8682/4 have a small internal ROM that contains six 2-byte interrupt vectors pointing to addresses 2048-2065/4096-4114,

where 3-byte jump absolute instructions are located (See Figure 4). These jump instructions each contain a 1-byte opcode and a 2-byte starting address for the interrupt service routine. The Z8682/4 take 36 system clock cycles to enter an interrupt subroutine.

Z8682 Address (Hex)	Z8684 Address (Hex)	Contains Jump Instruction and Subroutine Address For
800-802	1000-1002	IRQ <sub>0</sub>
803-805	1003-1005	IRQ <sub>1</sub>
806-808	1006-1008	IRQ <sub>2</sub>
809-80B	1009-1000	IRQ <sub>3</sub>
80C-80E	100C-100E	IRQ <sub>4</sub>
80F-811	100F-1011	IRQ <sub>5</sub>

Table 1. Z8682/4 Interrupt Processing

Polled interrupt systems are also supported. To accommodate a polled structure, any or all of the interrupt inputs can be masked and the Interrupt Request register polled to determine which of the interrupt requests needs service.

\* This feature differs in the Z8681 and Z8682

### Clock

The on-chip oscillator has a high-gain, parallel-resonant amplifier for connection to a crystal or to any suitable external clock source (XTAL1 = Input, XTAL2 = Output).

The crystal source is connected across XTAL1 and XTAL2, using the recommended capacitance ( $C_L \leq 15$  pF maximum) from

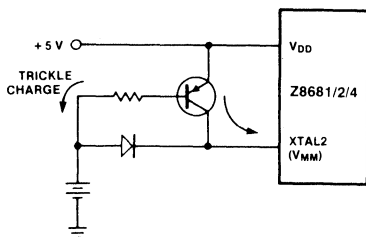
each pin to ground. The specifications for the crystal are as follows:

- AT cut, parallel-resonant
- Series resistance,  $R_s \leq 100 \Omega$
- For Z8681/2/4, Z868XL 8 MHz maximum
- For Z8681A/2A/4A, 12 MHz maximum

### Power Down Standby Option

The low-power standby mode allows power to be removed without losing the contents of the 124 general-purpose registers. This mode is available only to the user as a bonding option whereby pin 2 (normally XTAL2) is replaced by the  $V_{MM}$  (standby) power supply input. This necessitates the use of an external clock generator (input = XTAL1) rather than a crystal source.

The removal of power, whether intended or due to power failure, must be preceded by a software routine that stores the appropriate status into the register file. Figure 13 shows the recommended circuit for a battery back-up supply system.



**Figure 13. Recommended Driver Circuit for Power-Down Operation**

### Z8681/2/4 Interchangeability

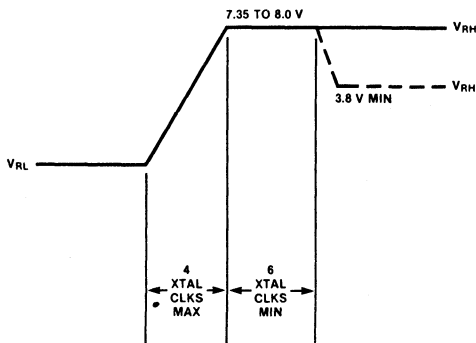
Although the Z8681, Z8682 and Z8684 have minor differences, a system can be designed for compatibility with both ROMless versions. To achieve interchangeability, the design must take into account the special requirements of each device in the external interface, initialization, and memory mapping.

**External Interface.** The Z8682/4 requires a 7.5 V positive logic level on the  $\overline{\text{RESET}}$  pin for at least 6 clock periods immediately following reset, as shown in Figure 14. The Z8681 requires a 3.8 V or higher positive logic level, but is compatible with the Z8682/4  $\overline{\text{RESET}}$  waveform. Figure 15 shows a simple circuit for generating the 7.5 V level.

**Initialization.** The Z8681 wakes up after reset with Port 0 configured as an input, which means Port 0 lines are floating in a high-impedance state. Because of this pullup or pulldown, resistors must be attached to Port 0 lines to force them to a valid logic level

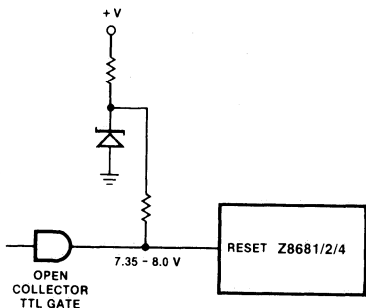
until Port 0 is configured as an address port.

Port 0 initialization is discussed in the section on ports. An example of an initialization routine for Z8681/2/4 compatibility is shown in Table 2. Only the Z8681 need execute this program.



**Figure 14. Z8682/4  $\overline{\text{RESET}}$  Pin Input Waveform**

**Z8681/2/4 Interchangeability** (Continued)

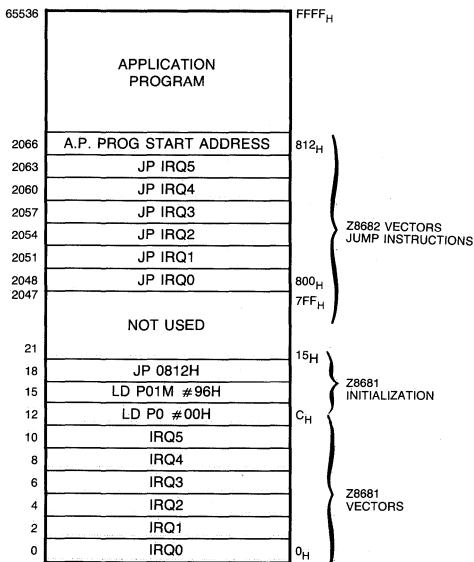


**Figure 15. RESET Circuit**

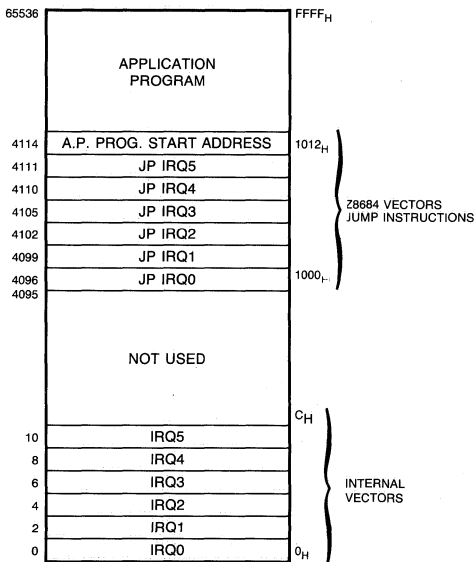
Address	Opcodes	Instruction	Comments
000C	E6 00 00	LD PO #00H	Set A <sub>8</sub> -A <sub>15</sub> to 0
000F	E6 F8 96	LD P01M#96H	Configure Port 0 as A <sub>8</sub> -A <sub>15</sub> . Eliminate extended memory timing.
0012	8D 08 12	JP START ADDRESS	Execute application program.

**Table 2. Initialization Routine**

**Memory Mapping.** The Z8681, Z8682 and Z8684 lower memory boundaries are located at 0, 2048 and 4076 respectively. A single program ROM can be used with either product if the logical program memory map shown in Figure 16 is followed. The Z8681 vectors and initialization routine must be starting at address 0 and the Z8682/4 3-byte vectors (jump instructions) must be at address 2048/4096 and higher. Addresses in the range 21-2047/21-4095 are not used. Figure 17 shows practical schemes for implementing this memory map using 4K and 2K ROMs.

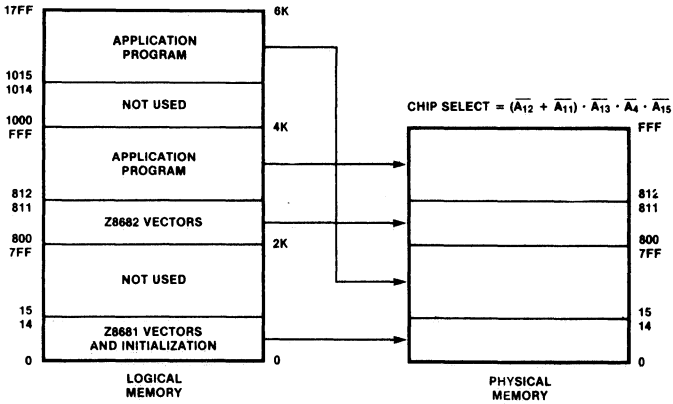


**Figure 16a. Z8681/2 Logical Program Memory Mapping**

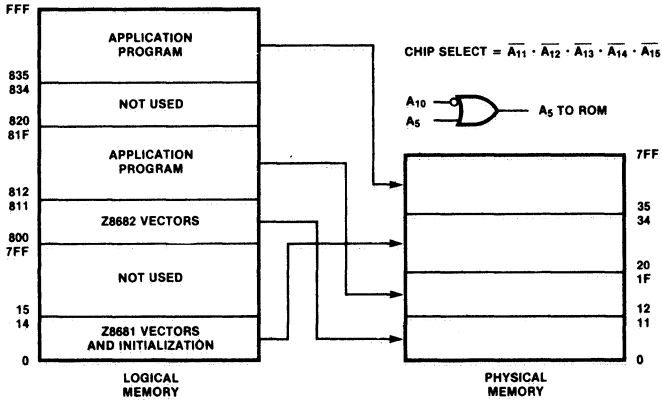


**Figure 16b. Z8684 Logical Program Memory Mapping**

**Z8681/Z8682/Z8684 Interchangeability (Continued)**



**a. Logical to Physical Memory Mapping for 4K ROM**



**b. Logical to Physical Memory Mapping for 2K ROM**

**Figure 17. Practical Schemes for Implementing Z8681 and Z8682 Compatible Memory Map**



**Z8681/L**  
**Z8682/L**  
**Z8684/L**

## Instruction Set Notation

**Addressing Modes.** The following notation is used to describe the addressing modes and instruction operations as shown in the instruction summary.

- IRR** Indirect register pair or indirect working-register pair address
- Irr** Indirect working-register pair only
- X** Indexed address
- DA** Direct address
- RA** Relative address
- IM** Immediate
- R** Register or working-register address
- r** Working-register address only
- IR** Indirect-register or indirect working-register address
- Ir** Indirect working-register address only
- RR** Register pair or working register pair address

**Symbols.** The following symbols are used in describing the instruction set.

- dst** Destination location or contents
- src** Source location or contents
- cc** Condition code (see list)
- @** Indirect address prefix
- SP** Stack pointer (control registers 254-255)
- PC** Program counter
- FLAGS** Flag register (control register 252)
- RP** Register pointer (control register 253)

**IMR** Interrupt mask register (control register 251)

Assignment of a value is indicated by the symbol " $\leftarrow$ ". For example,  
 $dst \leftarrow dst + src$   
 indicates that the source data is added to the destination data and the result is stored in the destination location. The notation "addr(n)" is used to refer to bit "n" of a given location. For example,  
 $dst(7)$   
 refers to bit 7 of the destination operand.

**Flags.** Control Register R252 contains the following six arithmetic flags plus two user selectable flags:

<b>C</b>	Carry flag	$b_7$ ..	$b_0$								
<b>Z</b>	Zero flag	<table style="border-collapse: collapse; margin: auto;"> <tr> <td style="border: 1px solid black; padding: 2px;">C</td> <td style="border: 1px solid black; padding: 2px;">Z</td> <td style="border: 1px solid black; padding: 2px;">S</td> <td style="border: 1px solid black; padding: 2px;">V</td> <td style="border: 1px solid black; padding: 2px;">D</td> <td style="border: 1px solid black; padding: 2px;">H</td> <td style="border: 1px solid black; padding: 2px;">F2</td> <td style="border: 1px solid black; padding: 2px;">F1</td> </tr> </table>		C	Z	S	V	D	H	F2	F1
C	Z			S	V	D	H	F2	F1		
<b>S</b>	Sign flag										
<b>V</b>	Overflow flag										
<b>D</b>	Decimal-adjust flag										
<b>H</b>	Half-carry flag										
		F1	F2								

Affected flags are indicated by:

- 0** Cleared to zero
- 1** Set to one
- \*** Set or cleared according to operation
- Unaffected
- X** Undefined

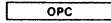
## Condition Codes

Value	Mnemonic	Meaning	Flags Set
1000		Always true	---
0111	C	Carry	C = 1
1111	NC	No carry	C = 0
0110	Z	Zero	Z = 1
1110	NZ	Not zero	Z = 0
1101	PL	Plus	S = 0
0101	MI	Minus	S = 1
0100	OV	Overflow	V = 1
1100	NOV	No overflow	V = 0
0110	EQ	Equal	Z = 1
1110	NE	Not equal	Z = 0
1001	GE	Greater than or equal	(S XOR V) = 0
0001	LT	Less than	(S XOR V) = 1
1010	GT	Greater than	[Z OR (S XOR V)] = 0
0010	LE	Less than or equal	[Z OR (S XOR V)] = 1
1111	UGE	Unsigned greater than or equal	C = 0
0111	ULT	Unsigned less than	C = 1
1011	UGT	Unsigned greater than	(C = 0 AND Z = 0) = 1
0011	ULE	Unsigned less than or equal	(C OR Z) = 1
0000		Never true	---

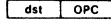


Z8681/L  
Z8682/L  
Z8684/L

## Instruction Formats

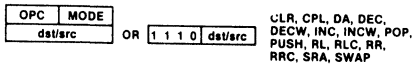


CCF, DI, EI, IRET, NOP,  
RCF, RET, SCF

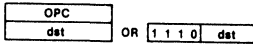


INC r

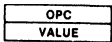
### One-Byte Instruction



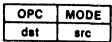
CLR, CPL, DA, DEC,  
DECW, INC, INCW, POP,  
PUSH, RL, RLC, RR,  
RRC, SRA, SWAP



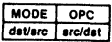
JP, CALL (Indirect)



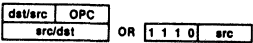
SRP



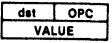
ADC, ADD, AND, CP,  
OR, SBC, SUB,  
TCM, TM, XOR



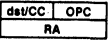
LD, LDE, LDEI,  
LDC, LDCI



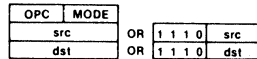
LD



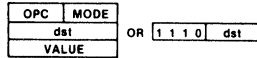
LD



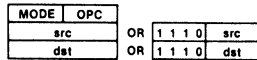
DJNZ, JR



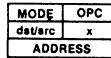
ADC, ADD, AND, CP,  
LD, OR, SBC, SUB,  
TCM, TM, XOR



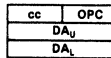
ADC, ADD, AND, CP,  
LD, OR, SBC, SUB,  
TCM, TM, XOR



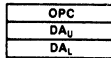
LD



LD



JP



CALL

### Two-Byte Instruction

### Three-Byte Instruction

Figure 18. Instruction Formats





Z8681/L  
Z8682/L  
Z8684/L

### Instruction Summary


Instruction and Operation	Addr Mode		Opcode Byte (Hex)	Flags Affected						
	dst	src		C	Z	S	V	D	H	
<b>ADC</b> dst,src dst ← dst + src + C	(Note 1)		1□	*	*	*	*	0	*	
<b>ADD</b> dst,src dst ← dst + src	(Note 1)		0□	*	*	*	*	0	*	
<b>AND</b> dst,src dst ← dst AND src	(Note 1)		5□	-	*	*	0	-	-	
<b>CALL</b> dst SP ← SP - 2 @SP ← PC; PC ← dst	DA IRR		D6 D4	-	-	-	-	-	-	
<b>CCF</b> C ← NOT C			EF	*	-	-	-	-	-	
<b>CLR</b> dst dst ← 0	R IR		B0 B1	-	-	-	-	-	-	
<b>COM</b> dst dst ← NOT dst	R IR		60 61	-	*	*	0	-	-	
<b>CP</b> dst,src dst - src	(Note 1)		A□	*	*	*	*	-	-	
<b>DA</b> dst dst ← DA dst	R IR		40 41	*	*	*	X	-	-	
<b>DEC</b> dst dst ← dst - 1	R IR		00 01	-	*	*	*	-	-	
<b>DECW</b> dst dst ← dst - 1	RR IR		80 81	-	*	*	*	-	-	
<b>DI</b> IMR (7) ← 0			8F	-	-	-	-	-	-	
<b>DJNZ</b> r,dst r ← r - 1 if r ≠ 0 PC ← PC + dst Range: +127, -128	RA		rA r=0-F	-	-	-	-	-	-	
<b>EI</b> IMR (7) ← 1			9F	-	-	-	-	-	-	
<b>INC</b> dst dst ← dst + 1	r R IR		rE r=0-F 20 21	-	*	*	*	-	-	
<b>INCW</b> dst dst ← dst + 1	RR IR		A0 A1	-	*	*	*	-	-	
<b>IRET</b> FLAGS ← @SP; SP ← SP + 1 PC ← @SP; SP ← SP + 2; IMR (7) ← 1			BF	*	*	*	*	*	*	
<b>JP</b> cc,dst if cc is true PC ← dst	DA IRR		cD c=0-F 30	-	-	-	-	-	-	
<b>JR</b> cc,dst if cc is true, PC ← PC + dst Range: +127, -128	RA		cB c=0-F	-	-	-	-	-	-	

Instruction and Operation	Addr Mode		Opcode Byte (Hex)	Flags Affected						
	dst	src		C	Z	S	V	D	H	
<b>LD</b> dst,src dst ← src	r R	Im R	rC r8 r9	-	-	-	-	-	-	
			r=0-F							
	r X r Ir R R R R IR	X r Ir r R R R R IR	C7 D7 E3 F3 E4 E5 E6 E7 F5							
<b>LDC</b> dst,src dst ← src	r Irr	Imm r	C2 D2	-	-	-	-	-	-	
<b>LDCI</b> dst,src dst ← src r ← r + 1; rr ← rr + 1	Ir Irr	Irr Ir	C3 D3	-	-	-	-	-	-	
<b>LDE</b> dst,src dst ← src	r Irr	Imm r	82 92	-	-	-	-	-	-	
<b>LDEI</b> dst,src dst ← src r ← r + 1; rr ← rr + 1	Ir Irr	Irr Ir	83 93	-	-	-	-	-	-	
<b>NOP</b>			FF	-	-	-	-	-	-	
<b>OR</b> dst,src dst ← dst OR src	(Note 1)		4□	-	*	*	0	-	-	
<b>POP</b> dst dst ← @SP SP ← SP + 1	R IR		50 51	-	-	-	-	-	-	
<b>PUSH</b> src SP ← SP - 1; @SP ← src		R IR	70 71	-	-	-	-	-	-	
<b>RCF</b> C ← 0			CF	0	-	-	-	-	-	
<b>RET</b> PC ← @SP; SP ← SP + 2			AF	-	-	-	-	-	-	
<b>RL</b> dst		R IR	90 91	*	*	*	*	-	-	
<b>RLC</b> dst		R IR	10 11	*	*	*	*	-	-	
<b>RR</b> dst		R IR	E0 E1	*	*	*	*	-	-	
<b>RRC</b> dst		R IR	C0 C1	*	*	*	*	-	-	
<b>SBC</b> dst,src dst ← dst - src - C	(Note 1)		3□	*	*	*	*	1	*	
<b>SCF</b> C ← 1			DF	1	-	-	-	-	-	
<b>SRA</b> dst		R IR	D0 D1	*	*	*	0	-	-	



Z8681/L  
Z8682/L  
Z8684/L

Instruction Summary (Continued)

Instruction and Operation	Addr Mode		Opcode Byte (Hex)	Flags Affected						
	dst	src		C	Z	S	V	D	H	
<b>SRP</b> src RP ← src		Im	31	-	-	-	-	-	-	-
<b>SUB</b> dst,src dst ← dst - src		(Note 1)	2□	*	*	*	*	1	*	*
<b>SWAP</b> dst		R IR	F0 F1	X	*	*	X	-	-	-

Instruction and Operation	Addr Mode		Opcode Byte (Hex)	Flags Affected						
	dst	src		C	Z	S	V	D	H	
<b>TCM</b> dst,src (NOT dst) AND src		(Note 1)	6□	-	*	*	0	-	-	-
<b>TM</b> dst, src dst AND src		(Note 1)	7□	-	*	*	0	-	-	-
<b>XOR</b> 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 is found in the instruction set table above. The second nibble is expressed symbolically by a □ in this table, and its value is found in the following table to the left of the applicable addressing mode pair.

For example, to determine the opcode of an ADC instruction using the addressing modes r (destination) and Ir (source) is 13.

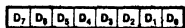
Addr Mode		Lower Opcode Nibble
dst	src	
r	r	2
r	Ir	3
R	R	4
R	IR	5
R	IM	6
IR	IM	7



**Z8681/L**  
**Z8682/L**  
**Z8684/L**

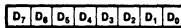
## Registers

**R240 SIO**  
 Serial I/O Register  
 (F0<sub>H</sub>; Read/Write)



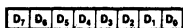
SERIAL DATA (D<sub>6</sub> = LSB)

**R244 T0**  
 Counter/Timer 0 Register  
 (F4<sub>H</sub>; Read/Write)



T<sub>0</sub> INITIAL VALUE (WHEN WRITTEN)  
 (RANGE: 1-256 DECIMAL 01-00 HEX)  
 T<sub>0</sub> CURRENT VALUE (WHEN READ)

**R241 TMR**  
 Timer Mode Register  
 (F1<sub>H</sub>; Read/Write)



**T<sub>OUT</sub> MODES**  
 NOT USED = 00  
 T<sub>0</sub> OUT = 01  
 T<sub>1</sub> OUT = 10  
 INTERNAL CLOCK OUT = 11

**T<sub>IN</sub> MODES**  
 EXTERNAL CLOCK INPUT = 00  
 GATE INPUT = 01  
 TRIGGER INPUT = 10  
 (NON-RETRIGGERABLE)  
 TRIGGER INPUT = 11  
 (RETRIGGERABLE)

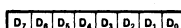
0 = NO FUNCTION  
 1 = LOAD T<sub>0</sub>

0 = DISABLE T<sub>0</sub> COUNT  
 1 = ENABLE T<sub>0</sub> COUNT

0 = NO FUNCTION  
 1 = LOAD T<sub>1</sub>

0 = DISABLE T<sub>1</sub> COUNT  
 1 = ENABLE T<sub>1</sub> COUNT

**R245 PRE0**  
 Prescaler 0 Register  
 (F5<sub>H</sub>; Write Only)

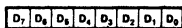


COUNT MODE  
 0 = T<sub>0</sub> SINGLE-PASS  
 1 = T<sub>0</sub> MODULO-N

RESERVED (MUST BE 0)

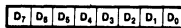
PRESCALER MODULO  
 (RANGE: 1-64 DECIMAL  
 01-00 HEX)

**R242 T1**  
 Counter Timer 1 Register  
 (F2<sub>H</sub>; Read/Write)



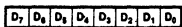
T<sub>1</sub> INITIAL VALUE (WHEN WRITTEN)  
 (RANGE 1-256 DECIMAL 01-00 HEX)  
 T<sub>1</sub> CURRENT VALUE (WHEN READ)

**R246 P2M**  
 Port 2 Mode Register  
 (F6<sub>H</sub>; Write Only)



P<sub>2</sub>...P<sub>20</sub> I/O DEFINITION  
 0 DEFINES BIT AS OUTPUT  
 1 DEFINES BIT AS INPUT

**R243 PRE1**  
 Prescaler 1 Register  
 (F3<sub>H</sub>; Write Only)

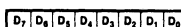


COUNT MODE  
 1 = T<sub>1</sub> MODULO-N  
 0 = T<sub>1</sub> SINGLE-PASS

CLOCK SOURCE  
 1 = T<sub>1</sub> INTERNAL  
 0 = T<sub>1</sub> EXTERNAL  
 TIMING INPUT  
 (T<sub>IN</sub>) MODE

PRESCALER MODULO  
 (RANGE: 1-64 DECIMAL  
 01-00 HEX)

**R247 P3M**  
 Port 3 Mode Register  
 (F7<sub>H</sub>; Write Only)



0 PORT 2 PULL-UPS OPEN DRAIN  
 1 PORT 2 PULL-UPS ACTIVE

RESERVED (MUST BE 0)

0 P<sub>32</sub> = INPUT P<sub>35</sub> = OUTPUT  
 1 P<sub>32</sub> = DAV0/RDY0 P<sub>35</sub> = RDY0/DAV0

0 0 P<sub>33</sub> = INPUT P<sub>34</sub> = OUTPUT  
 0 1 P<sub>33</sub> = INPUT P<sub>34</sub> = DM  
 1 0 P<sub>33</sub> = INPUT P<sub>34</sub> = RDY1/DAV1  
 1 1 P<sub>33</sub> = DAV1/RDY1 P<sub>34</sub> = RDY2/DAV2

0 P<sub>31</sub> = INPUT (T<sub>IN</sub>) P<sub>36</sub> = OUTPUT (T<sub>OUT</sub>)  
 1 P<sub>31</sub> = DAV2/RDY2 P<sub>36</sub> = RDY2/DAV2

0 P<sub>30</sub> = INPUT P<sub>37</sub> = OUTPUT  
 1 P<sub>30</sub> = SERIAL IN P<sub>37</sub> = SERIAL OUT

0 PARITY OFF  
 1 PARITY ON

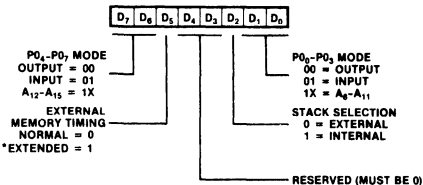
Figure 19. Control Registers



Z8681/L  
Z8682/L  
Z8684/L

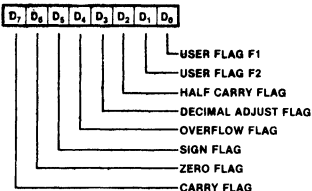
Registers (Continued)

**R248 P01M**  
Port 0 Register  
(F8<sub>H</sub>; Write Only)

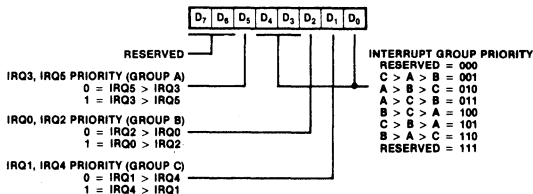


\*ALWAYS EXTENDED TIMING AFTER RESET

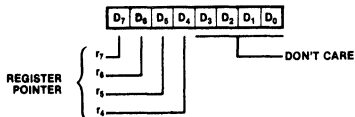
**R252 FLAGS**  
Flag Register  
(FC<sub>H</sub>; Read/Write)



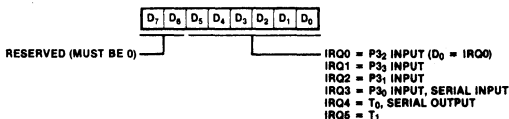
**R249 IPR**  
Interrupt Priority Register  
(F9<sub>H</sub>; Write Only)



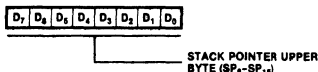
**R253 RP**  
Register Pointer  
(FD<sub>H</sub>; Read/Write)



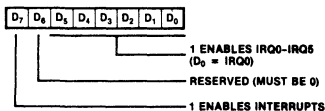
**R250 IRQ**  
Interrupt Request Register  
(FA<sub>H</sub>; Read/Write)



**R254 SPH**  
Stack Pointer  
(FE<sub>H</sub>; Read/Write)



**R251 IMR**  
Interrupt Mask Register  
(FB<sub>H</sub>; Read/Write)



**R255 SPL**  
Stack Pointer  
(FF<sub>H</sub>; Read/Write)

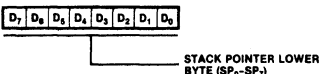


Figure 19. Control Registers (Continued)



Z8681/L  
Z8682/L  
Z8684/L

Opcode Map

Lower Nibble (Hex)

		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
Upper Nibble (Hex)	0	6,5 DEC R <sub>1</sub>	6,5 DEC IR <sub>1</sub>	6,5 ADD r <sub>1</sub> , r <sub>2</sub>	6,5 ADD r <sub>1</sub> , Ir <sub>2</sub>	10,5 ADD R <sub>2</sub> , R <sub>1</sub>	10,5 ADD IR <sub>2</sub> , R <sub>1</sub>	10,5 ADD R <sub>1</sub> , IM	10,5 ADD IR <sub>1</sub> , IM	6,5 LD r <sub>1</sub> , R <sub>2</sub>	6,5 LD r <sub>2</sub> , R <sub>1</sub>	12/10,5 DJNZ r <sub>1</sub> , RA	12/10,0 JR cc, RA	6,5 LD r <sub>1</sub> , IM	12/10,0 JP cc, DA	6,5 INC r <sub>1</sub>		
	1	6,5 RLC R <sub>1</sub>	6,5 RLC IR <sub>1</sub>	6,5 ADC r <sub>1</sub> , r <sub>2</sub>	6,5 ADC r <sub>1</sub> , Ir <sub>2</sub>	10,5 ADC R <sub>2</sub> , R <sub>1</sub>	10,5 ADC IR <sub>2</sub> , R <sub>1</sub>	10,5 ADC R <sub>1</sub> , IM	10,5 ADC IR <sub>1</sub> , IM									
	2	6,5 INC R <sub>1</sub>	6,5 INC IR <sub>1</sub>	6,5 SUB r <sub>1</sub> , r <sub>2</sub>	6,5 SUB r <sub>1</sub> , Ir <sub>2</sub>	10,5 SUB R <sub>2</sub> , R <sub>1</sub>	10,5 SUB IR <sub>2</sub> , R <sub>1</sub>	10,5 SUB R <sub>1</sub> , IM	10,5 SUB IR <sub>1</sub> , IM									
	3	8,0 JP IR <sub>1</sub>	6,1 SRP IM	6,5 SBC r <sub>1</sub> , r <sub>2</sub>	6,5 SBC r <sub>1</sub> , Ir <sub>2</sub>	10,5 SBC R <sub>2</sub> , R <sub>1</sub>	10,5 SBC IR <sub>2</sub> , R <sub>1</sub>	10,5 SBC R <sub>1</sub> , IM	10,5 SBC IR <sub>1</sub> , IM									
	4	8,5 DA R <sub>1</sub>	8,5 DA IR <sub>1</sub>	6,5 OR r <sub>1</sub> , r <sub>2</sub>	6,5 OR r <sub>1</sub> , Ir <sub>2</sub>	10,5 OR R <sub>2</sub> , R <sub>1</sub>	10,5 OR IR <sub>2</sub> , R <sub>1</sub>	10,5 OR R <sub>1</sub> , IM	10,5 OR IR <sub>1</sub> , IM									
	5	10,5 POP R <sub>1</sub>	10,5 POP IR <sub>1</sub>	6,5 AND r <sub>1</sub> , r <sub>2</sub>	6,5 AND r <sub>1</sub> , Ir <sub>2</sub>	10,5 AND R <sub>2</sub> , R <sub>1</sub>	10,5 AND IR <sub>2</sub> , R <sub>1</sub>	10,5 AND R <sub>1</sub> , IM	10,5 AND IR <sub>1</sub> , IM									
	6	6,5 COM R <sub>1</sub>	6,5 COM IR <sub>1</sub>	6,5 TCM r <sub>1</sub> , r <sub>2</sub>	6,5 TCM r <sub>1</sub> , Ir <sub>2</sub>	10,5 TCM R <sub>2</sub> , R <sub>1</sub>	10,5 TCM IR <sub>2</sub> , R <sub>1</sub>	10,5 TCM R <sub>1</sub> , IM	10,5 TCM IR <sub>1</sub> , IM									
	7	10/12,1 PUSH R <sub>2</sub>	12/14,1 PUSH IR <sub>2</sub>	6,5 TM r <sub>1</sub> , r <sub>2</sub>	6,5 TM r <sub>1</sub> , Ir <sub>2</sub>	10,5 TM R <sub>2</sub> , R <sub>1</sub>	10,5 TM IR <sub>2</sub> , R <sub>1</sub>	10,5 TM R <sub>1</sub> , IM	10,5 TM IR <sub>1</sub> , IM									
	8	10,5 DECW RR <sub>1</sub>	10,5 DECW IR <sub>1</sub>	12,0 LDE r <sub>1</sub> , Ir <sub>2</sub>	18,0 LDEI Ir <sub>1</sub> , Ir <sub>2</sub>													6,1 DI
	9	6,5 RL R <sub>1</sub>	6,5 RL IR <sub>1</sub>	12,0 LDE r <sub>2</sub> , Ir <sub>1</sub>	18,0 LDEI Ir <sub>2</sub> , Ir <sub>1</sub>													6,1 EI
	A	10,5 INCW RR <sub>1</sub>	10,5 INCW IR <sub>1</sub>	6,5 CP r <sub>1</sub> , r <sub>2</sub>	6,5 CP r <sub>1</sub> , Ir <sub>2</sub>	10,5 CP R <sub>2</sub> , R <sub>1</sub>	10,5 CP IR <sub>2</sub> , R <sub>1</sub>	10,5 CP R <sub>1</sub> , IM	10,5 CP IR <sub>1</sub> , IM									14,0 RET
	B	6,5 CLR R <sub>1</sub>	6,5 CLR IR <sub>1</sub>	6,5 XOR r <sub>1</sub> , r <sub>2</sub>	6,5 XOR r <sub>1</sub> , Ir <sub>2</sub>	10,5 XOR R <sub>2</sub> , R <sub>1</sub>	10,5 XOR IR <sub>2</sub> , R <sub>1</sub>	10,5 XOR R <sub>1</sub> , IM	10,5 XOR IR <sub>1</sub> , IM									16,0 IRET
	C	6,5 RRC R <sub>1</sub>	6,5 RRC IR <sub>1</sub>	12,0 LDC r <sub>1</sub> , Ir <sub>2</sub>	18,0 LDCI Ir <sub>1</sub> , Ir <sub>2</sub>													10,5 LD r <sub>1</sub> , x, R <sub>2</sub>
	D	6,5 SRA R <sub>1</sub>	6,5 SRA IR <sub>1</sub>	12,0 LDC r <sub>2</sub> , Ir <sub>1</sub>	18,0 LDCI Ir <sub>2</sub> , Ir <sub>1</sub>	20,0 CALL* IR <sub>1</sub>		20,0 CALL DA	10,5 LD r <sub>2</sub> , x, R <sub>1</sub>									6,5 SCF
	E	6,5 RR R <sub>1</sub>	6,5 RR IR <sub>1</sub>		6,5 LD r <sub>1</sub> , Ir <sub>2</sub>	10,5 LD R <sub>2</sub> , R <sub>1</sub>	10,5 LD IR <sub>2</sub> , R <sub>1</sub>	10,5 LD R <sub>1</sub> , IM	10,5 LD IR <sub>1</sub> , IM									6,5 CCF
	F	8,5 SWAP R <sub>1</sub>	8,5 SWAP IR <sub>1</sub>		6,5 LD Ir <sub>1</sub> , r <sub>2</sub>		10,5 LD R <sub>2</sub> , IR <sub>1</sub>											6,0 NOP

Bytes per Instruction

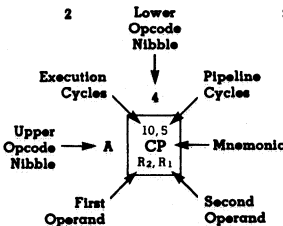
2

3

2

3

1



Legend:

- R = 8-Bit Address
- r = 4-Bit Address
- R<sub>1</sub> or r<sub>1</sub> = Dst Address
- R<sub>2</sub> or r<sub>2</sub> = Src Address

Sequence:

Opcode, First Operand, Second Operand

Note: The blank areas are not defined.

\*2-byte instruction; fetch cycle appears as a 3-byte instruction

### Absolute Maximum Ratings

Voltage on all pins\*  
 with respect to GND . . . -0.3 V to +7.0 V  
 Operating Ambient  
 Temperature . . . . . 0°C to +70°C  
 Storage Temperature . . . -65°C to +150°C

Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

### Test Conditions

The characteristics below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND. Positive current flows into the reference pin. Standard conditions are as follows:

- $+4.75\text{ V} \leq V_{CC} \leq +5.25\text{ V}$
- $\text{GND} = 0\text{ V}$
- $0^\circ\text{C} \leq T_A \leq +70^\circ\text{C}$  for S (Standard temperature)
- $-40^\circ\text{C} \leq T_A \leq +85^\circ\text{C}$  for E (Extended temperature)

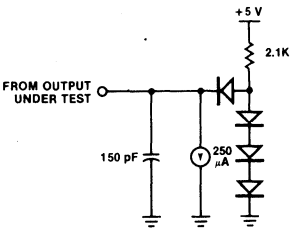


Figure 20. Test Load 1

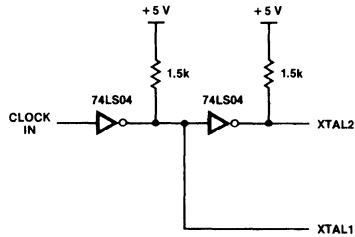


Figure 21. External Clock Interface Circuit

### 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	2.0	$V_{CC}$	V	
$V_{IL}$	Input Low Voltage	-0.3	0.8	V	
$V_{RH}$	Reset Input High Voltage	3.8	$V_{CC}$	V	See Note
$V_{RL}$	Reset Input Low Voltage	-0.3	0.8	V	
$V_{OH}$	Output High Voltage	2.4		V	$I_{OH} = -250\ \mu\text{A}$
$V_{OL}$	Output Low Voltage		0.4	V	$I_{OL} = +2.0\ \text{mA}$
$I_{IL}$	Input Leakage	-10	10	$\mu\text{A}$	$0\text{ V} \leq V_{IN} \leq +5.25\text{ V}$
$I_{OL}$	Output Leakage	-10	10	$\mu\text{A}$	$0\text{ V} \leq V_{IN} \leq +5.25\text{ V}$
$I_{IR}$	Reset Input Current		-50	$\mu\text{A}$	$V_{CC} = +5.25\text{ V}, V_{RL} = 0\text{ V}$
$I_{CC}$	$V_{CC}$ Supply Current		120	mA	
			80*		
$I_{MM}$	$V_{MM}$ Supply Current		10	mA	Power Down Mode
$V_{MM}$	Backup Supply Voltage	3	$V_{CC}$	V	Power Down

NOTE:  
 The Reset line (pin 6) is used to place the Z8682 in external memory mode. This is accomplished as shown in Figure 14

\* This value is for Z8681L/2L/4L only.



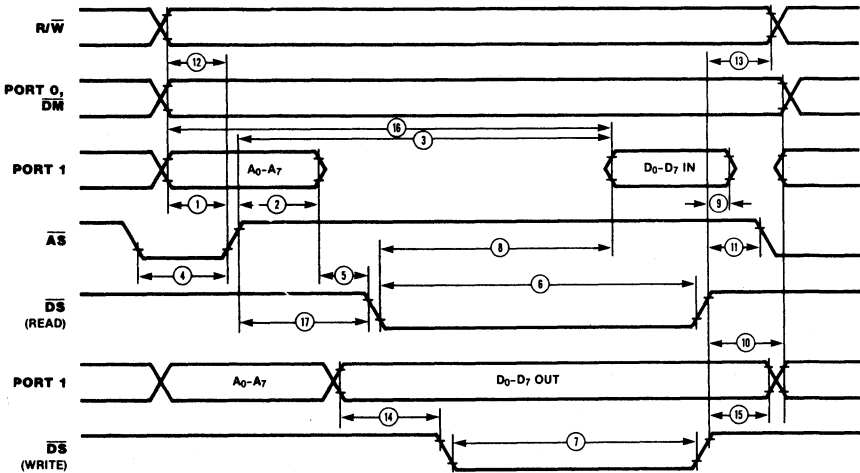
**Z8681/L**  
**Z8682/L**  
**Z8684/L**

**External I/O or Memory Read and Write Timing**

No.	Symbol	Parameter	Z8681/2/4		Z8681A/2A/4A		Notes <sup>†</sup>
			Min	Max	Min	Max	
1	TdA(AS)	Address Valid to $\overline{AS}$ $\uparrow$ Delay	50		35		1,2,3
2	TdAS(A)	$\overline{AS}$ $\uparrow$ to Address Float Delay	70		45		1,2,3
3	TdAS(DR)	$\overline{AS}$ $\uparrow$ to Read Data Required Valid		360		220	1,2,3,4
4	TwAS	$\overline{AS}$ Low Width	80		55		1,2,3
5	TdAz(DS)	Address Float to $\overline{DS}$ $\downarrow$	0		0		1
6	TwDSR	$\overline{DS}$ (Read) Low Width	250		185		1,2,3,4
7	TwDSW	$\overline{DS}$ (Write) Low Width	160		110		1,2,3,4
8	TdDSR(DR)	$\overline{DS}$ $\downarrow$ to Read Data Required Valid		200		130	1,2,3,4
9	ThDR(DS)	Read Data to $\overline{DS}$ $\uparrow$ Hold Time	0		0		1
10	TdDS(A)	$\overline{DS}$ $\uparrow$ to Address Active Delay	70		45		1,2,3
11	TdDS(AS)	$\overline{DS}$ $\uparrow$ to $\overline{AS}$ Delay	70		55		1,2,3
12	TdR/W(AS)	R/ $\overline{W}$ Valid to $\overline{AS}$ $\uparrow$ Delay	50		30		1,2,3
13	TdDS(R/W)	$\overline{DS}$ $\uparrow$ to R/ $\overline{W}$ Not Valid	60		35		1,2,3
14	TdDW(DSW)	Write Data Valid to $\overline{DS}$ (Write) $\downarrow$ Delay	50		35		1,2,3
15	TdDS(DW)	$\overline{DS}$ $\uparrow$ to Write Data Not Valid Delay	70		45		1,2,3
16	TdA(DR)	Address valid to Read Data Required Valid		410		255	1,2,3,4
17	TdAS(DS)	$\overline{AS}$ $\uparrow$ to $\overline{DS}$ $\downarrow$ Delay	80		55		1,2,3

NOTES:

1. Test Load 1
2. Timing numbers given are for minimum T<sub>pC</sub>
3. Also see clock cycle dependent characteristics table.
4. When using extended memory timing add 2 T<sub>pC</sub>.
5. All timing reference use 2.0 V for logic "1" and 0.8V for a logic "0".
- \* All units in nanoseconds (ns).
- † Timings are preliminary and subject to change.



**Figure 22. External I/O or Memory Read/Write Timing**



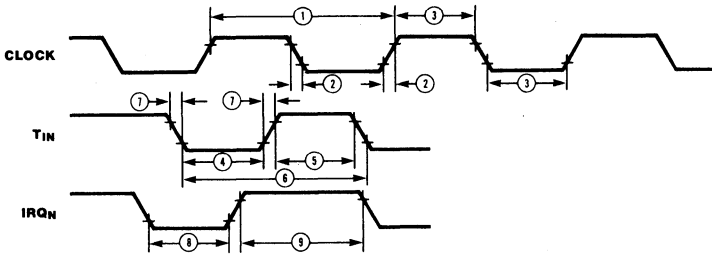
**Additional Timing Table**

No.	Symbol	Parameter	Z8681/2/4		Z8681A/2A/4A		Notes †
			Min	Max	Min	Max	
1	TpC	Input Clock Period	125	1000	83	1000	1
2	TrC, TfC	Clock Input Rise And Fall Times		25		15	1
3	TwC	Input Clock Width	37		26		1
4	TwTinL	Timer Input Low Width	100		70		2
5	TwTinH	Timer Input High Width	3TpC		3TpC		2
6	TpTin	Timer Input Period	$\frac{TpC}{8}$		$\frac{TpC}{8}$		2
7	TrTin, TfTin	Timer Input Rise And Fall Times		100		100	2
8	TwIL	Interrupt Request Input Low Time	100		70		2,3
9	TwIH	Interrupt Request Input High Time	3TpC		3TpC		2,3

NOTES:

1. Clock timing references uses 3.8 V for a logic "1" and 0.8 V for a logic "0"
2. Timing reference uses 2.0 V for a logic "1" and 0.8 V for a logic "0"

3. Interrupt request via Port 3.
- \* Units in nanoseconds (ns).
- † Timings are preliminary and subject to change.



**Figure 23. Additional Timing**



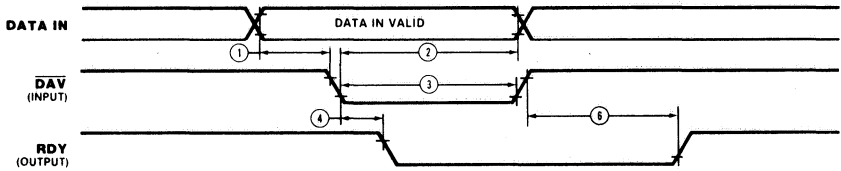
### Handshake Timing

No.	Symbol	Parameter	Z8681/2/4 Z8681A/2A/4A		Z8681L/2L/4L		Notes <sup>*,†</sup>
			Min	Max	Min	Max	
			1	TsDI(DAV)	Data In Setup Time	0	
2	ThDI(DAV)	Data In Hold Time	230		160		
3	TwDAV	Data Available Width	175		120		
4	TdDAV <sub>i</sub> (RDY)	$\overline{DAV}$ ↓ Input to RDY ↓ Delay		175	120	1,2	
5	TdDAV <sub>o</sub> (RDY)	$\overline{DAV}$ ↓ Output to RDY ↓ Delay	0		0	1,3	
6	TdDAV <sub>i</sub> r(RDY)	$\overline{DAV}$ ↑ Input to RDY ↑ Delay		175	120	1,2	
7	TdDAV <sub>o</sub> r(RDY)	$\overline{DAV}$ ↑ Output to RDY ↑ Delay	0		0	1,3	
8	TdDO(DAV)	Data Out to $\overline{DAV}$ ↓ Delay	50		30	1	
9	TdRDY(DAV)	Rdy ↓ Input to $\overline{DAV}$ ↑ Delay	0	200	0	140	1

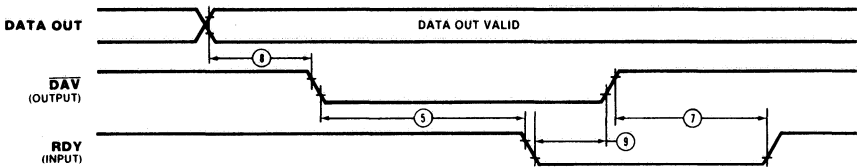
**NOTES:**

1. Test load 1
2. Input handshake
3. Output handshake
4. All timing references use 2.0 V for a logic "1" and 0.8 V for a logic "0".

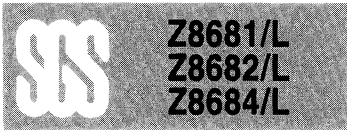
\* Units in nanoseconds (ns).  
† Timings are preliminary and subject to change.



**Figure 24a. Input Handshake Timing**



**Figure 24b. Output Handshake Timing**



### Clock-Cycle-Time-Dependent Characteristics

Number	Symbols	Z8681/2/4	Z8681A/2A/4A
		Z8681L/2L/4L Equation	Equation
1	TdA(AS)	TpC-75	TpC-50
2	TdAS(A)	TpC-55	TpC-40
3	TdAS(DR)	4TpC-140*	4TpC-110*
4	TwAS	TpC-45	TpC-30
6	TwDSR	3TpC-125*	3TpC-65*
7	TwDSW	2TpC-90*	2TpC-55*
8	TdDSR(DR)	3TpC-175*	3TpC-120*
10	Td(DS)A	TpC-55	TpC-40
11	TdDS(AS)	TpC-55	TpC-30
12	TdR/W(AS)	TpC-75	TpC-55
13	TdDS(R/W)	TpC-65	TpC-50
14	TdDW(DSW)	TpC-75	TpC-50
15	TdDS(DW)	TpC-55	TpC-40
16	TdA(DR)	5TpC-215*	5TpC-160*
17	TdAS(DS)	TpC-45	TpC-30

\* Add 2TpC when using extended memory timing.

### Ordering Information

Type	Package	Temp.	Clock	Description
Z8681/2/4	B1 Plastic	0/+70°C	8 MHz	ROMless Microcomputer
Z8681/2/4	B6 Plastic	-40/+85°C		
Z8681/2/4	D1 Ceramic	0/+70°C		
Z8681/2/4	D6 Ceramic	-40/+85°C		
Z8681/2/4	C1 Plastic Chip Carrier	0/+70°C		
Z8681/2/4	C6 Plastic Chip Carrier	-40/+85°C		
Z8681/2/4	K1 Ceramic Chip Carrier	0/+70°C		
Z8681/2/4	K6 Ceramic Chip Carrier	-40/+85°C		
Z8681/2/4A	B1 Plastic	0/+70°C	12 MHz	
Z8681/2/4A	B6 Plastic	-40/+85°C		
Z8681/2/4A	D1 Ceramic	0/+70°C		
Z8681/2/4A	D6 Ceramic	-40/+85°C		
Z8681/2/4A	C1 Plastic Chip Carrier	0/+70°C		
Z8681/2/4A	C6 Plastic Chip Carrier	-40/+85°C		
Z8681/2/4A	K1 Ceramic Chip Carrier	0/+70°C		
Z8681/2/4A	K6 Ceramic Chip Carrier	-40/+85°C		
Z8681L/2L/4L	B1 Plastic	0/+70°C	8 MHz	ROMless Microcomputer Low Power Version
Z8681L/2L/4L	B6 Plastic	-40/+85°C		
Z8681L/2L/4L	D1 Ceramic	0/+70°C		
Z8681L/2L/4L	D6 Ceramic	-40/+85°C		
Z8681L/2L/4L	C1 Plastic Chip Carrier	0/+70°C		
Z8681L/2L/4L	C6 Plastic Chip Carrier	-40/+85°C		
Z8681L/2L/4L	K1 Ceramic Chip Carrier	0/+70°C		
Z8681L/2L/4L	K6 Ceramic Chip Carrier	-40/+85°C		





# Z86E11

Preliminary Data

## Z8 4K EPROM Microcomputer

- Complete microcomputer, 4K bytes of EPROM, 128 bytes of RAM, 32 I/O lines, and up to 60K bytes addressable external space each for program and data memory. Fully compatible with standard ROM version.
- 144-byte register file, including 124 general-registers, four I/O port registers, and 16 status and control registers.
- Minimum instruction execution time 1  $\mu$ s, at 12 MHz.
- Vectored priority interrupts for I/O, counter/timers, and UART.
- Full-duplex UART and two programmable 8-bit counter/timers, each with a 6-bit programmable prescaler.
- Register Pointer so that short, fast instructions can access any of nine working-register groups in 1.5  $\mu$ s (8MHz).
- On-chip oscillator which accepts crystal or external clock drive.
- Low-power standby option which retains contents of general-purpose registers.
- Single +5 V power supply - all pins TTL compatible.
- Two Eprom programming modes:
  - Eprom-like, using a standard Eprom programmer,
  - Autoprogram, self-programming during normal program execution.
  - An on-chip ROM provides a Program/Verify facility to allow a simple and time-efficient self-program operation.
- Integrated programmable protection avoids EPROM content read-out.
- Available in 8 and 12 MHz versions.

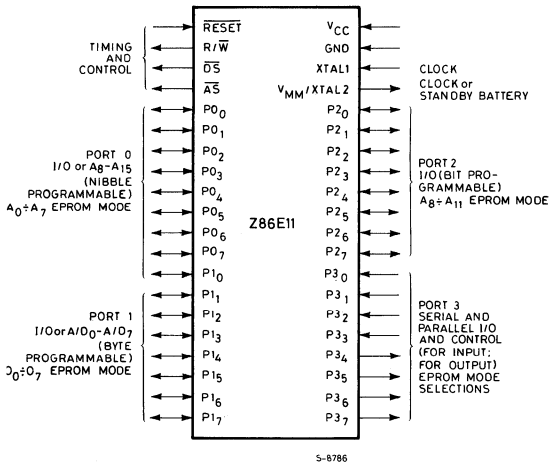


Figure 1. Logic Functions

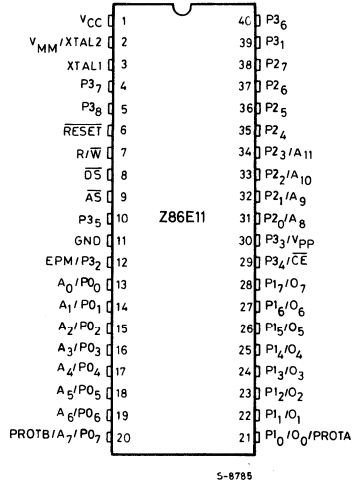


Figure 2. Pin Configuration



## General Description

The Z86E11 microcomputer is an EPROM member of the Z8 family; completely developed by SGS, it maintains the sophisticated architecture and full compatibility with the currently available ROM-based units.

It can be configured as a stand-alone microcomputer with 4K bytes of internal EPROM, or as a traditional microcomputer that manages up to 120K bytes of external memory, or as a parallel-processing element in a system with other processors and peripheral controllers.

The 4K x 8 on-board EPROM can be programmed in two modes, Eprom-like and Autoprogram. In Eprom-like, the programming procedure is similar to that for

a M2732, with the only exception being for the programming voltage which must be 12.5 V related to the SGS NMOS-E3 used technology. Autoprogram permits byte-programming during normal microcomputer program execution.

An important facility is the programmable read-out protections which allow the user to inhibit external access to proprietary program code by programming 2 non-volatile transistors. These locks can be reset only by erasing the entire EPROM array.

For its characteristics, the Z86E11 can be considered as a low cost development tool for the Z8 microcomputer family.

## Architecture

Z86E11 architecture is characterized by a flexible I/O scheme, an efficient register and address space structure and a number of ancillary features that are helpful in many applications.

Microcomputer applications demand powerful I/O capabilities. The Z86E11 fulfills this with 32 pins dedicated to input/output. These lines are grouped into four ports of eight lines each and are configurable under software control to provide timing, status signals, serial or parallel I/O with or without handshake, address/data bus for interfacing external memory, and address, data and selections in EPROM mode.

Because the multiplexed address/data bus is merged with the I/O-oriented ports, the Z86E11 can assume many different memory and I/O configurations. These configurations range from a self-contained microcomputer to a microprocessor that can address 120K bytes of external memory (Figure 3).

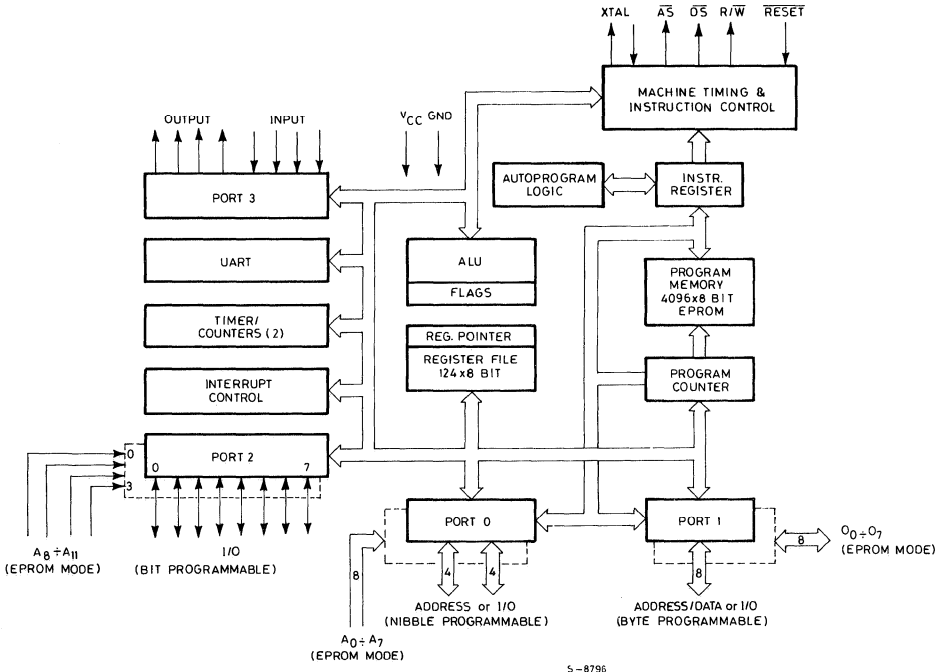
Three basic address spaces are available to support this wide range of configurations: program memory (internal and external), data memory (external) and the register file

(internal). The 144-byte random-access register file is composed of 124 general-purpose registers, four I/O port registers, and 16 control and status registers.

To unburden the program from coping with real-time problems such as serial data communication and counting/timing, an asynchronous receiver/transmitter (UART) and two counter/timers with a large number of user-selectable modes are offered on-chip. Hardware support for the UART is minimized because one of the on-chip timers supplies the bit rate.

An autoprogram logic permits Eprom byte-programming during the normal microcomputer program execution, using the STORE constant instruction.

This permits the microcomputer to look at the contents of the working register's register for an external RAM allocated to the program memory space, addressed using two of the register file registers. The renamed external RAM was expressly developed for the Autoprogram function, thus it is externally inaccessible.

**Architecture (Continued)**

**Figure 3. Block Diagram**
**Pin Description**

**P0-P07.** *I/O Port Lines* (input/outputs, TTL compatible). 8 lines Nibble Programmable that can be configured under program control for I/O or external memory interface and,  $A_0-A_7$  in EPROM mode. P07 can be configured as Read-out protection in memory expansion mode (PROTB), applying a high voltage level (10 V)

**P1-P17.** *I/O Port Lines* (input/outputs, TTL compatible). 8 lines Byte Programmable that can be configured under program control for I/O or multiplexed address ( $A_0-A_7$ ) and data ( $D_0-D_7$ ) lines used to interface with program/data memory, and  $0_0-0_7$  in EPROM mode. P10 can be configured as Read-out

protection in Testing-mode and EPROM-mode (PROTA), applying a high voltage level (10 V).

**P2-P27.** *I/O Port Lines* (input/outputs, TTL compatible). 8 lines Bit Programmable where the 4 less significant bits can be configured as  $A_8-A_{11}$  in EPROM mode.

**P3-P37.** *I/O Port Lines* (TTL compatible) 4 lines input (P30-P33), 4 lines output (P34-P37). They can also be configured as control lines. In EPROM mode: P32 becomes EPM (Eprom-like) when a high voltage level ( $\geq 7$  V) is applied, P33 becomes  $V_{pp}$  ( $12.5 \text{ V} \pm 300 \text{ mV}$ ), P34 becomes CE to perform program enable/verify.



## Pin Description (Continued)

**AS.** *Address Strobe* (output, active Low). Address Strobe is pulsed once at the beginning of each machine cycle. Addresses output via Port 1 for all external program or data memory transfers are valid at the trailing edge of AS. Under program control, AS can be placed in the high-impedance state along with ports 0 and 1, Data Strobe and Read/Write.

**DS.** *Data Strobe* (output, active Low). Data Strobe is activated once for each external memory transfer.

**RESET.** *Reset* (input, active Low). RESET

initializes the Z86E11. When RESET is deactivated, program execution begins from internal program location 000C<sub>H</sub>.

**R/W.** *Read/Write* (output). R/W is Low when the Z86E11 is writing to external program or data memory.

**XTAL1, XTAL2.** *Crystal 1, Crystal 2* (time-base input and output). These pins connect a series-resonant crystal (8 MHz maximum) or an external single-phase clock (8 MHz maximum) to the on-chip clock oscillator and buffer.

## Address Spaces

**Program Memory.** The 16-bit program counter addresses 64K bytes of program memory space. Program memory can be located in two areas: one internal and the other external (Figure 4). The first 4096 bytes consist of on-chip EPROM. At addresses 4096 and greater, the Z86E11 executes external program memory fetches.

The first 12 bytes of program memory are reserved for the interrupt vectors. These locations contain six 16-bit vectors that correspond to the six available interrupts.

**Data Memory.** The Z86E11 can address 60K bytes of external data memory beginning at locations 4096 (Figure 5). External data memory may be included with or separated from the external program memory space. DM, an optional I/O function that can be programmed to appear on pin P3<sub>4</sub>, is used to distinguish between data and program memory space.

**Register File.** The 144-byte register file includes four I/O port registers (R0-R3), 124 general-purpose registers (R4-R127) and 16

control and status registers (R240-R255). These registers are assigned the address locations shown in Figure 6.

Z86E11 instructions can access registers directly or indirectly with an 8-bit address field. The Z86E11 also allows short 4-bit register addressing using the Register Pointer (one of the control registers). In the 4-bit mode, the register file is divided into nine working-register groups, each occupying 16 contiguous locations (Figure 7). The Register Pointer addresses the starting location of the active working-register group.

**Stacks.** Either the internal register file or the external data memory can be used for the stack. A 16-bit Stack Pointer (R254 and R255) is used for the external stack, which can reside anywhere in data memory between locations 4096 and 65535. An 8-bit Stack Pointer (R255) is used for the internal stack that resides within the 124 general-purpose registers (R4-R127).

### Address Spaces (Continued)

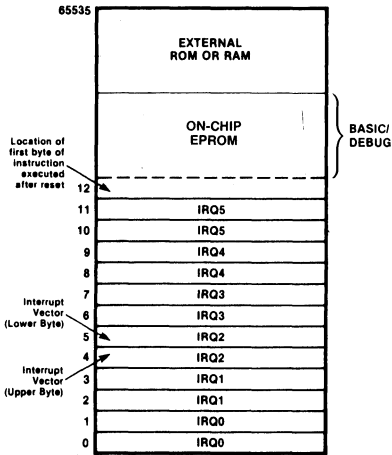


Figure 4. Program Memory Map

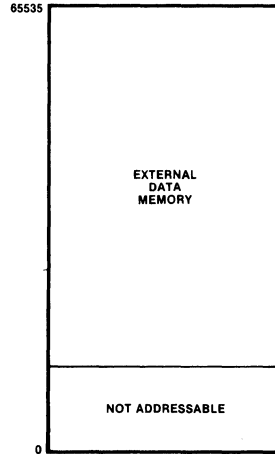


Figure 5. Data Memory Map

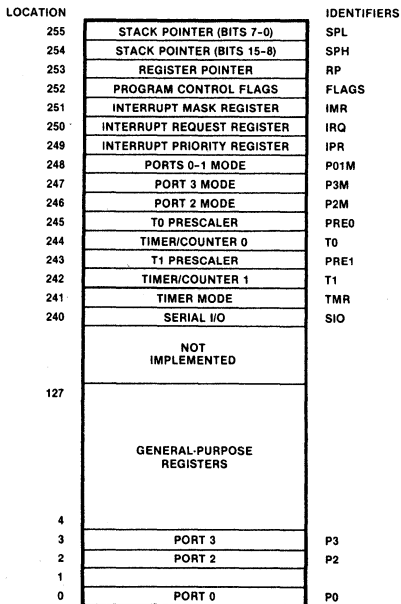


Figure 6. The Register File

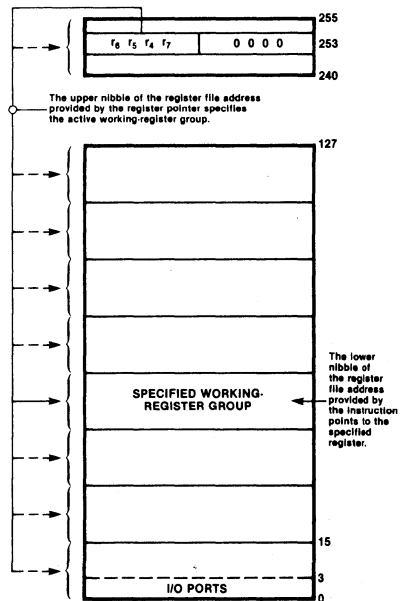


Figure 7. The Register Pointer



## Serial Input/Output

Port 3 lines P3<sub>0</sub> and P3<sub>7</sub> can be programmed as serial I/O lines for full-duplex serial asynchronous receiver/transmitter operation. The bit rate is controlled by Counter/Timer 0, with a maximum rate of 62.5K bits/second.

The Z86E11 automatically adds a start bit and two stop bits to transmitted data (Figure 8). Odd parity is also available as an option. Eight data bits are always transmitted,

regardless of parity selection. If parity is enabled, the eighth bit is the odd parity bit. An interrupt request (IRQ<sub>4</sub>) is generated on all transmitted characters.

Received data must have a start bit, eight data bits and at least one stop bit. If parity is on, bit 7 of the received data is replaced by a parity error flag. Received characters generate the IRQ<sub>3</sub> interrupt request.

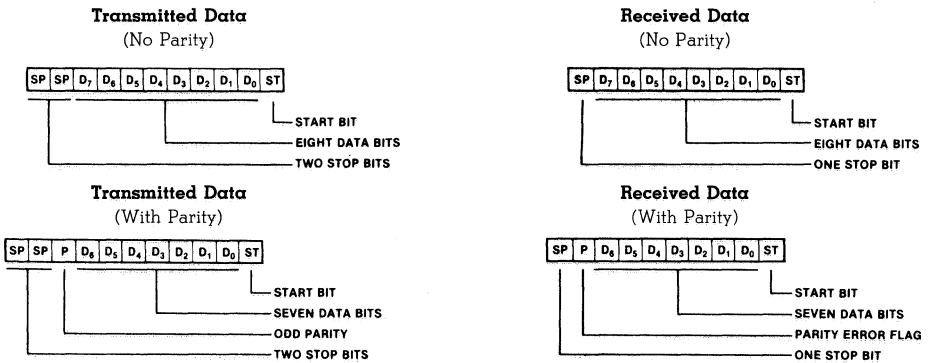


Figure 8. Serial Data Formats

## Counter/Timers

The Z86E11 contains two 8-bit programmable counter/timers (T<sub>0</sub> and T<sub>1</sub>), each driven by its own 6-bit programmable prescaler. The T<sub>1</sub> prescaler can be driven by internal or external clock sources; however, the T<sub>0</sub> prescaler is driven by the internal clock only.

The 6-bit prescaler can divide the input frequency of the clock source by any number from 1 to 64. Each prescaler drives its counter, which decrements the value (1 to 256) that has been loaded into the counter. When the counter reaches the end of count, a timer interrupt request—IRQ<sub>4</sub> (T<sub>0</sub>) or IRQ<sub>5</sub> (T<sub>1</sub>)—is generated.

The counters can be started, stopped, restarted to continue, or restarted from the initial value. The counters can also be programmed to stop upon reaching zero (single-pass mode) or to automatically reload

the initial value and continue counting (modulo-n continuous mode). The counters, but not the prescalers, can be read any time without disturbing their value or count mode.

The clock source for T<sub>1</sub> is user-definable and can be the internal microprocessor clock (4 MHz maximum) divided by four, or an external signal input via Port 3. The Timer Mode register configures the external timer input as an external clock (1 MHz maximum), a trigger input that can be retriggerable or non-retriggerable, or as a gate input for the internal clock. The counter/timers can be programmably cascaded by connecting the T<sub>0</sub> output to the input of T<sub>1</sub>. Port 3 line P3<sub>6</sub> also serves as a timer output (T<sub>OUT</sub>) through which T<sub>0</sub>, T<sub>1</sub> or the internal clock can be output.

## I/O Ports

The Z86E11 has 32 lines dedicated to input and output. These lines are grouped into four ports of eight lines each and are configurable as input, output or address/data. Under software control, the ports can be programmed to provide address outputs, timing, status signals, serial I/O, and parallel I/O with or without handshake. All ports have active pull-ups and pull-downs compatible with TTL loads. All the ports assume different configurations in EPROM mode.

**Port 1** can be programmed as a byte I/O port or as an address/data port for interfacing external memory. When used as an I/O port, Port 1 may be placed under handshake control. In this configuration, Port 3 lines P<sub>33</sub> and P<sub>34</sub> are used as the handshake controls RDY<sub>1</sub> and DAV<sub>1</sub> (Ready and Data Available).

Memory locations greater than 4096 are referenced through Port 1. To interface external memory, Port 1 must be programmed for the multiplexed Address/Data mode. If more than 256 external locations are required, Port 0 must output the additional lines.

Port 1 can be placed in the high-impedance state along with Port 0,  $\overline{AS}$ ,  $\overline{DS}$  and R/W, allowing the Z86E11 to share common resources in multiprocessor and DMA applications. Data transfers can be controlled by assigning P<sub>33</sub> as a Bus Acknowledge input and P<sub>34</sub> as a Bus Request output.

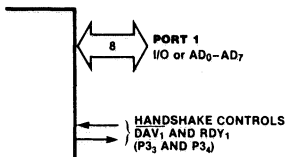


Figure 9a. Port 1

**Port 0** can be programmed as a nibble I/O port, or as an address port for interfacing external memory. When used as an I/O port, Port 0 may be placed under handshake control. In this configuration, Port 3 lines

lines P<sub>32</sub> and P<sub>35</sub> are used as the handshake controls DAV<sub>0</sub> and RDY<sub>0</sub>. Handshake signal assignment is dictated by the I/O direction of the upper nibble P<sub>04</sub>-P<sub>07</sub>.

For external memory references, Port 0 can provide address bits A<sub>8</sub>-A<sub>11</sub> (lower nibble) or A<sub>8</sub>-A<sub>15</sub> (lower and upper nibble) depending on the required address space. If the address range requires 12 bits or less, the upper nibble of Port 0 can be programmed independently as I/O while the lower nibble is used for addressing. When Port 0 nibbles are defined as address bits, they can be set to the high-impedance state along with Port 1 and the control signals  $\overline{AS}$ ,  $\overline{DS}$  and R/W.

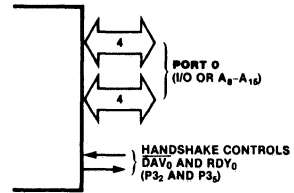


Figure 9b. Port 0

**Port 2** bits can be programmed independently as input or output. This port is always available for I/O operations. In addition, Port 2 can be configured to provide open-drain outputs.

Like Ports 0 and 1, Port 2 may also be placed under handshake control. In this configuration, Port 3 lines P<sub>31</sub> and P<sub>36</sub> are used as the handshake controls lines DAV<sub>2</sub> and RDY<sub>2</sub>. The handshake signal assignment for Port 3 lines P<sub>31</sub> and P<sub>36</sub> is dictated by the direction (input or output) assigned to bit 7 of Port 2.

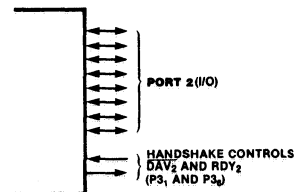


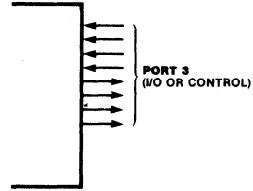
Figure 9c. Port 2



**I/O Ports (Continued)**

**Port 3** lines can be configured as I/O or control lines. In either case, the direction of the eight lines is fixed as four input (P3<sub>0</sub>-P3<sub>3</sub>) and four output (P3<sub>4</sub>-P3<sub>7</sub>). For serial I/O, lines P3<sub>0</sub> and P3<sub>7</sub> are programmed as serial in and serial out respectively.

Port 3 can also provide the following control functions: handshake for Ports 0, 1 and 2 (DAV and RDY); four external interrupt request signals (IRQ<sub>0</sub>-IRQ<sub>3</sub>); timer input and output signals (T<sub>IN</sub> and T<sub>OUT</sub>) and Data Memory Select (DM).



**Figure 9d. Port 3**

**Interrupts**

The Z86E11 allows six different interrupts from eight sources: the four Port 3 lines P3<sub>0</sub>-P3<sub>3</sub>, Serial In, Serial Out, and the two counter/timers. These interrupts are both maskable and prioritized. The Interrupt Mask register globally or individually enables or disables the six interrupt requests. When more than one interrupt is pending, priorities are resolved by a programmable priority encoder that is controlled by the Interrupt Priority register.

All Z86E11 interrupts are vectored. When an interrupt request is granted, and interrupt machine cycle is entered. This disables all

subsequent interrupts, saves the Program Counter and status flags, and branches to the program memory vector location reserved for that interrupt. This memory location and the next byte contain the 16-bit address of the interrupt service routine for that particular interrupt request.

Polled interrupt systems are also supported. To accommodate a polled structure, any or all of the interrupt inputs can be masked and the Interrupt Request register polled to determine which of the interrupt requests needs service.

**Clock**

The on-chip oscillator has a high-gain, parallel-resonant amplifier for connection to a crystal or to any suitable external clock source (XTAL1 = Input, XTAL2 = Output).

The crystal source is connected across XTAL1 and XTAL2, using the recommended capacitors (C<sub>1</sub> ≤ 15 pF) from each pin to

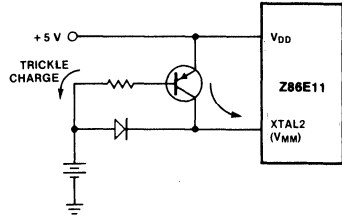
ground. The specifications for the crystal are as follows:

- AT cut, series resonant
- Fundamental types, 8/12 MHz maximum
- Series resistance, R<sub>s</sub> ≤ 100 Ω.

### Power Down Standby Option

The low-power standby mode allows power to be removed without losing the contents of the 124 general-purpose registers. This mode is available to the user as a bonding option whereby pin 2 (normally XTAL2) is replaced by the  $V_{MM}$  (standby) power supply input. This necessitates the use of an external clock generator (input = XTAL1) rather than a crystal source.

The removal of power, whether intended or due to power failure, must be preceded by a software routine that stores the appropriate status into the register file. Figure 10 shows the recommended circuit for a battery back-up supply system.



**Figure 10. Recommended Driver Circuit for Power Down Operation**

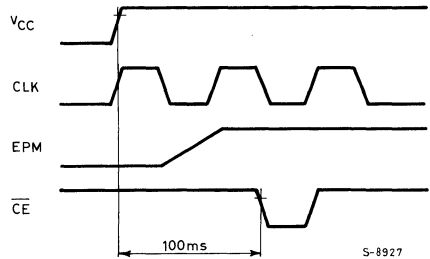
### EPROM Mode

Eprom-like programming. In this mode, the microcomputer memory is programmed, using a standard Eprom Programmer, with the same procedure as for our M2732 (32K EPROM). This has been made possible by the following Z86E11 configuration, where P1<sub>0</sub>-P1<sub>7</sub> are used as 8-bit I/O data (0<sub>0</sub>-0<sub>7</sub>), P0<sub>0</sub>-P0<sub>7</sub> and P2<sub>0</sub>-P2<sub>3</sub> are used as 12-bit Addresses (A<sub>0</sub>-A<sub>11</sub>); the microcomputer must be in Reset state, forcing the related pin to GND, and the clock must be active for the complete operation.

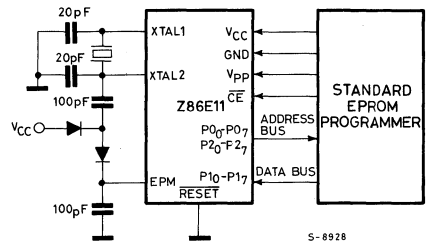
Three other pins are available for that purpose: the EPM pin on port P3<sub>2</sub>, which allows the microcomputer to recognize the Eprom-like condition when a high voltage ( $\geq 7V$ ) is applied; the V<sub>pp</sub> pin on port P3<sub>3</sub>, which is used to furnish programming voltage fixed on  $12.5V \pm 300mV$ ; and the CE pin on port P3<sub>4</sub>, which is used to perform program enable/verify.

For a correct microcomputer set-up the V<sub>CC</sub> must be applied at least 100 ms before the programming procedure starting (Figure 11).

A simple interface board, described in Figure 12, allows programming to be carried out through use of a standard Eprom-programmer.



**Figure 11. Set-up Waveforms**



**Figure 12. EPROM Programmer Interface Board**



**EPROM Mode** (Continued)

**Autoprogramming.** This mode permits programming one byte of the on-chip Eprom during normal microcomputer program execution. The instruction to be used is the Load Constant LCD @RR1,R2 (operating code D2).

This instruction allows the standard Z8 to load the contents of the working register to an external RAM memory allocated in the program memory space, addressed by a working register pair. The Z86E11 uses this instruction also to program the 4K bytes on-chip Eprom.

Addressing one of the on-chip Eprom bytes, using this instruction, the programming operation takes place when an high voltage level on the Vpp pin (12.5 V ± 300mV) is applied.

In this case, both the address and the data memory are internally stored for the necessary programming time, where the time is defined by the execution of 1024 NOP operations (1 NOP operation = 12 clock pulses). The programming time is contained between 1ms (12 MHz clock) and 12 ms (1 MHz clock).

As just mentioned, during this time, the CPU is internally forced to execute NOP instructions (operating code FF), while a RET instruction (operating code AF) is automatically executed at the end of programming.

For a correct program, restart is necessary to save the address of the Load Constant (LDC) next instruction in the Stack. This can be done by loading into the Stack the return address calling a Subroutine like follow, where, to permit a correct return to the main program, it is necessary to disable the interrupt before LDC execution.

```
.....  
DI  
CALL WRITE  
EI  
.....  
WRITE LDC @RR1,R2  
RET
```

**Programming Facility.** The most flexible way for on-chip Eprom programming is, as

we know, the use of a standard Eprom-programmer, selecting the Eprom-like facility, and using an appropriate interface board (Figure 12).

If, however, the planned operation is only a particular memory loading into the on-chip Eprom, it is possible to perform this operation in a much simpler way, using a board which allows the Z86E11 to read and load the renamed particular memory, using the autoloading procedure.

The software required for this operation is stored in the Z86E11 Test-memory (inaccessible). Figure 13 shows the autoloading program flow-chart.

When the microcomputer is forced in Test mode by applying a high voltage level (≥ 7 V) on the Reset pin, ports P0 and P1 are configured as Address/Data to access the external memories.

At this point, a test on port P2 is executed to decide if the on-chip Eprom autoloading is to be executed. This facility is accessed by forcing the values 40H or 41H on port P2 to execute, respectively, the Verify or Autoloading routine.

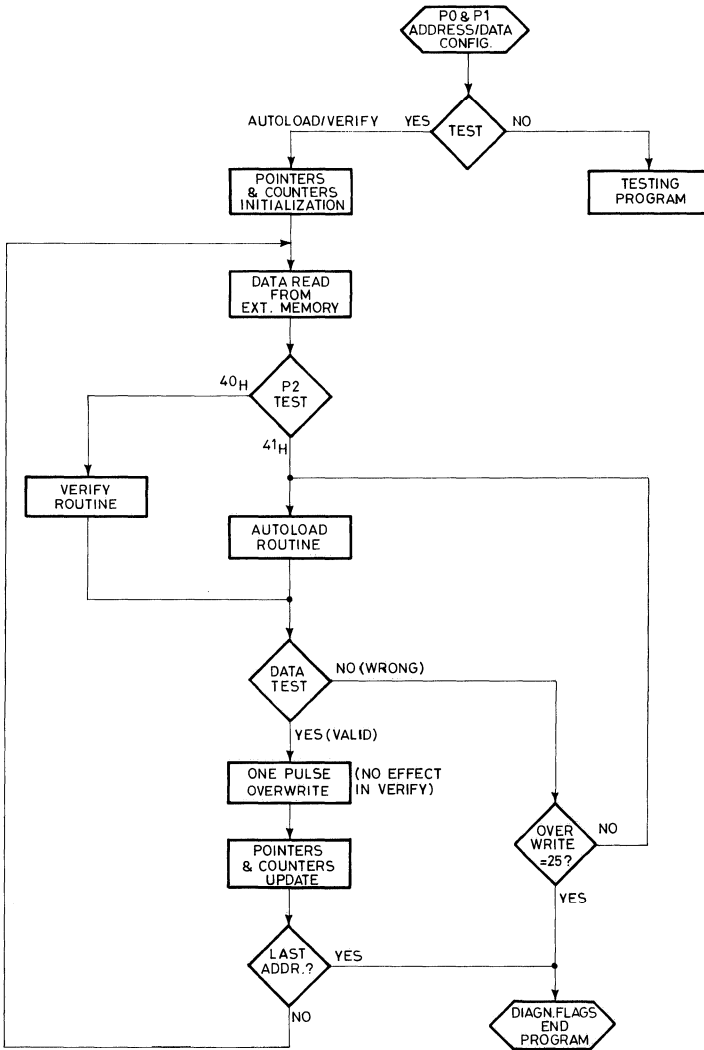
Consequently, the registers required for the operation are initialized, the data to be compared or stored is read, and the routine chosen is executed.

The Autoloading routine is an intelligent programming which executes a number of overwriting cycles equal to three times the number of programming cycles required to perform a correct byte programming (up to a maximum of 25). In this way, the on-chip Eprom programming time is optimized and equal to 25 sec. with an 8 MHz clock.

The verify routine is simply a byte-byte comparison between the external memory and the on-chip Eprom.

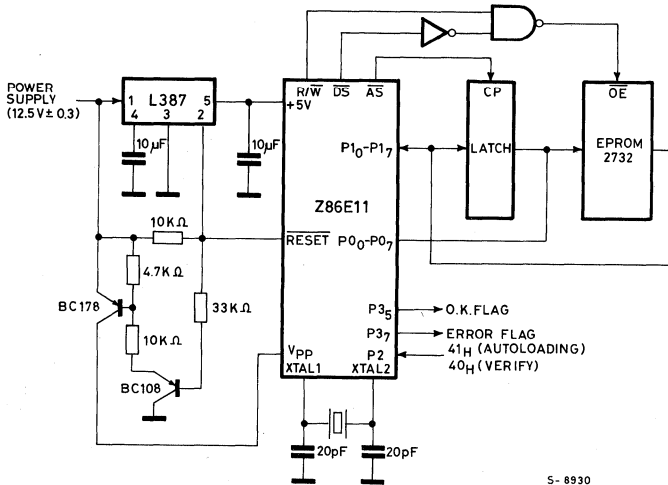
A possible failure, whether in Autoloading or Verify, produces a High logical level forced on P37. Similarly, when the operation is finished, the positive conclusion is underlined, bringing P35 High. An Autoloading/Verify Board diagram is shown in Figure 14, where the Vpp line control is necessary to not allow high voltage into the device when it has not yet been supplied.

**EPROM Mode (Continued)**



S-8929

**Figure 13. Autoloading Flow Chart**

**EPROM Mode (Continued)**


S- 8930

**Figure 14. Autoloading/Verify Board**

**Memory Read-Out Protection.** The protection, once activated, blocks reading memory content. Such reading can be carried out in two ways:

1. Entering Test Mode you can execute an external memory program which allows the on-chip Eprom reading through LOAD instructions execution.
2. Entering Eprom-like Mode, using the Verify facility.

Programming the first protection bit blocks reading in these two conditions (PROTA on port P1<sub>0</sub>).

Another protection bit (PROTB on port P0<sub>7</sub>) can be activated when the Z86E11 is in external memory configuration. This protection prevents software

manipulation of the external memory from who decides to read the on-chip memory content for a complete understanding of the user application board.

When the Z86E11 works in external memory facility the ports P0 and P1 are configured as Address/Data bus, so that the external memory instructions can be executed during the normal microcomputer operation. These instructions can be also an appropriate routine able to pull out the all on-chip memory content using LOAD instructions. When the protection is activated, each reading attempt of the internal memory content, using LOAD instructions, is vainificated because the data out will be always "FF".

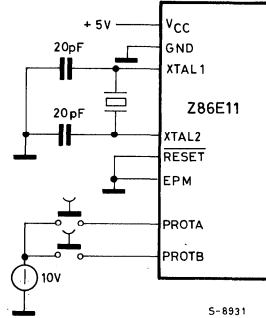
### EPROM Mode (Continued)

In consequence this protection activation inhibit the LOAD instructions execution from external to internal memory. To overcome this problem is necessary to call a "READ" routine written in on-chip memory space.

The protections are activated by programming 2 non-volatile transistors simply forcing 10 V for a time more than 100 ms on the desired pin, on condition that the microcomputer is in Reset state, the Clock signal is present and the EPM pin is not set.

If a complete protection is desired, both protections must be programmed.

A simple board diagram for read-out protection activation is shown in Figure 15.



**Figure 15. Read-out Protection Activation Diagram**

### Instruction Set Notation

**Addressing Modes.** The following notation is used to describe the addressing modes and instruction operations as shown in the instruction summary.

<b>IRR</b>	Indirect register pair or indirect working-register pair address
<b>Irr</b>	Indirect working-register pair only
<b>X</b>	Indexed address
<b>DA</b>	Direct address
<b>RA</b>	Relative address
<b>IM</b>	Immediate
<b>R</b>	Register or working-register address
<b>r</b>	Working-register address only
<b>IR</b>	Indirect-register or indirect working-register address
<b>Ir</b>	Indirect working-register address only
<b>RR</b>	Register pair or working register pair address

**Symbols.** The following symbols are used in describing the instruction set.

<b>dst</b>	Destination location or contents
<b>src</b>	Source location or contents
<b>cc</b>	Condition code (see list)
<b>@</b>	Indirect address prefix
<b>SP</b>	Stack pointer (control registers 254-255)
<b>PC</b>	Program counter
<b>FLAGS</b>	Flag register (control register 252)
<b>RP</b>	Register pointer (control register 253)

**IMR** Interrupt mask register (control register 251)

Assignment of a value is indicated by the symbol " $\leftarrow$ ". For example,

$$dst \leftarrow dst + src$$

indicates that the source data is added to the destination data and the result is stored in the destination location. The notation "addr(n)" is used to refer to bit "n" of a given location. For example,

$$dst(7)$$

refers to bit 7 of the destination operand.

**Flags.** Control Register R252 contains the following six flags:

<b>C</b>	Carry flag	$\begin{array}{ c c c c c c } \hline b_7 & & & & & b_0 \\ \hline \boxed{C} & \boxed{Z} & \boxed{S} & \boxed{V} & \boxed{D} & \boxed{H} & \boxed{F_2} & \boxed{F_1} \\ \hline \end{array}$	$\left. \begin{array}{l} F_1 \\ F_2 \end{array} \right\} \text{user flags}$
<b>Z</b>	Zero flag		
<b>S</b>	Sign flag		
<b>V</b>	Overflow flag		
<b>D</b>	Decimal-adjust flag		
<b>H</b>	Half-carry flag		

Affected flags are indicated by:

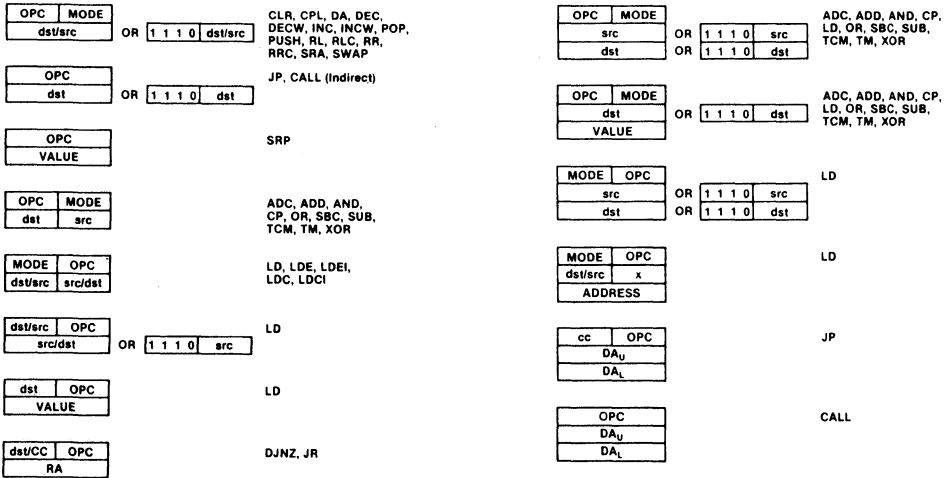
<b>0</b>	Cleared to zero
<b>1</b>	Set to one
<b>*</b>	Set or cleared according to operation
<b>-</b>	Unaffected
<b>X</b>	Undefined



**Condition Codes**

<b>Value</b>	<b>Mnemonic</b>	<b>Meaning</b>	<b>Flags Set</b>
1000		Always true	...
0111	C	Carry	C = 1
1111	NC	No carry	C = 0
0110	Z	Zero	Z = 1
1110	NZ	Not zero	Z = 0
1101	PL	Plus	S = 0
0101	MI	Minus	S = 1
0100	OV	Overflow	V = 1
1100	NOV	No overflow	V = 0
0110	EQ	Equal	Z = 1
1110	NE	Not equal	Z = 0
1001	GE	Greater than or equal	(S XOR V) = 0
0001	LT	Less than	(S XOR V) = 1
1010	GT	Greater than	[Z OR (S XOR V)] = 0
0010	LE	Less than or equal	[Z OR (S XOR V)] = 1
1111	UGE	Unsigned greater than or equal	C = 0
0111	ULT	Unsigned less than	C = 1
1011	UGT	Unsigned greater than	(C = 0 AND Z = 0) = 1
0011	ULE	Unsigned less than or equal	(C OR Z) = 1
0000		Never true	...

**Instruction Formats**

**One-Byte Instruction**

**Two-Byte instruction**
**Three-Byte instruction**
**Figure 16. Instruction Formats**



**Instruction Summary**

Instruction and Operation	Addr Mode		Opcode Byte (Hex)	Flags Affected					
	dst	src		C	Z	S	V	D	H
<b>ADC</b> dst,src dst ← dst + src + C	(Note 1)		1□	*	*	*	*	0	*
<b>ADD</b> dst,src dst ← dst + src	(Note 1)		0□	*	*	*	*	0	*
<b>AND</b> dst,src dst ← dst AND src	(Note 1)		5□	-	*	*	0	-	-
<b>CALL</b> dst SP ← SP - 2 @SP ← PC; PC ← dst	DA IRR		D6 D4	-	-	-	-	-	-
<b>CCF</b> C ← NOT C			EF	*	-	-	-	-	-
<b>CLR</b> dst dst ← 0	R IR		B0 B1	-	-	-	-	-	-
<b>COM</b> dst dst ← NOT dst	R IR		60 61	-	*	*	0	-	-
<b>CP</b> dst,src dst - src	(Note 1)		A□	*	*	*	*	-	-
<b>DA</b> dst dst ← DA dst	R IR		40 41	*	*	*	X	-	-
<b>DEC</b> dst dst ← dst - 1	R IR		00 01	-	*	*	*	-	-
<b>DECW</b> dst dst ← dst - 1	RR IR		80 81	-	*	*	*	-	-
<b>DI</b> IMR (7) ← 0			8F	-	-	-	-	-	-
<b>DJNZ</b> r,dst r ← r - 1 if r ≠ 0 PC ← PC + dst Range: +127, -128	RA		rA r=0-F	-	-	-	-	-	-
<b>EI</b> IMR (7) ← 1			9F	-	-	-	-	-	-
<b>INC</b> dst dst ← dst + 1	r R IR		rE r=0-F 20 21	-	*	*	*	-	-
<b>INCW</b> dst dst ← dst + 1	RR IR		A0 A1	-	*	*	*	-	-
<b>IRET</b> FLAGS ← @SP; SP ← SP + 1 PC ← @SP; SP ← SP + 2; IMR (7) ← 1			BF	*	*	*	*	*	*
<b>JP</b> cc,dst if cc is true PC ← dst	DA IRR		cD c=0-F 30	-	-	-	-	-	-
<b>JR</b> cc,dst if cc is true, PC ← PC + dst Range: +127, -128	RA		cB c=0-F	-	-	-	-	-	-

Instruction and Operation	Addr Mode		Opcode Byte (Hex)	Flags Affected					
	dst	src		C	Z	S	V	D	H
<b>LD</b> dst,src dst ← src	r R R	Im R r	rC r8 r9 r=0-F	-	-	-	-	-	-
	r X R Ir Ir R R R IR IR IR	X r Ir r R R Im Im R	C7 D7 E3 F3 E4 E5 E6 E7 F5						
<b>LDC</b> dst,src dst ← src	r Irr	Irr r	C2 D2	-	-	-	-	-	-
<b>LDCI</b> dst,src dst ← src r ← r + 1; rr ← rr + 1	Ir Irr	Irr Ir	C3 D3	-	-	-	-	-	-
<b>LDE</b> dst,src dst ← src	r Irr	Irr r	82 92	-	-	-	-	-	-
<b>LDEI</b> dst,src dst ← src r ← r + 1; rr ← rr + 1	Ir Irr	Irr Ir	83 93	-	-	-	-	-	-
<b>NOP</b>			FF	-	-	-	-	-	-
<b>OR</b> dst,src dst ← dst OR src	(Note 1)		4□	-	*	*	0	-	-
<b>POP</b> dst dst ← @SP SP ← SP + 1	R IR		50 51	-	-	-	-	-	-
<b>PUSH</b> src SP ← SP - 1; @SP ← src	R IR		70 71	-	-	-	-	-	-
<b>RCF</b> C ← 0			CF	0	-	-	-	-	-
<b>RET</b> PC ← @SP; SP ← SP + 2			AF	-	-	-	-	-	-
<b>RL</b> dst		R IR	90 91	*	*	*	*	-	-
<b>RLC</b> dst		R IR	10 11	*	*	*	*	-	-
<b>RR</b> dst		R IR	E0 E1	*	*	*	*	-	-
<b>RRC</b> dst		R IR	C0 C1	*	*	*	*	-	-
<b>SBC</b> dst,src dst ← dst - src - C	(Note 1)		3□	*	*	*	*	1	*
<b>SCF</b> C ← 1			DF	1	-	-	-	-	-
<b>SRA</b> dst		R IR	D0 D1	*	*	*	0	-	-



Instruction Summary (Continued)

Instruction and Operation	Addr Mode		Opcode Byte (Hex)	Flags Affected						
	dst	src		C	Z	S	V	D	H	
<b>SRP</b> src RP -- src		Im	31	-	-	-	-	-	-	-
<b>SUB</b> dst,src dst -- dst - src		(Note 1)	2□	*	*	*	*	1	*	
<b>SWAP</b> dst		R IR	F0 F1	X	*	*	X	-	-	

Note 1

These instructions have an identical set of addressing modes, which are encoded for brevity. The first opcode nibble is found in the instruction set table above. The second nibble is expressed symbolically by a □ in this table, and its value is found in the following table to the left of the applicable addressing mode pair.

For example, the opcode of an ADC instruction using the addressing modes r (destination) and Ir (source) is 13.

Instruction and Operation	Addr Mode		Opcode Byte (Hex)	Flags Affected						
	dst	src		C	Z	S	V	D	H	
<b>TCM</b> dst,src (NOT dst) AND src		(Note 1)	6□	-	*	*	0	-	-	
<b>TM</b> dst, src dst AND src		(Note 1)	7□	-	*	*	0	-	-	
<b>XOR</b> dst,src dst -- dst XOR src		(Note 1)	B□	-	*	*	0	-	-	

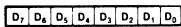
Addr Mode		Lower Opcode Nibble
dst	src	
r	r	2
r	Ir	3
R	R	4
R	IR	5
R	IM	6
IR	IM	7



# Z86E11

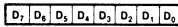
## Registers

**R240 SIO**  
**Serial I/O Register**  
 (F0H; Read/Write)



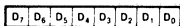
SERIAL DATA (D<sub>0</sub> = LSB)

**R244 T0**  
**Counter/Timer 0 Register**  
 (F4H; Read/Write)



T<sub>0</sub> INITIAL VALUE (WHEN WRITTEN)  
 (RANGE: 1 256 DECIMAL 01 00 HEX)  
 T<sub>0</sub> CURRENT VALUE (WHEN READ)

**R241 TMR**  
**Timer Mode Register**  
 (F1H; Read/Write)

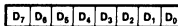


T<sub>OUT</sub> MODES  
 NOT USED = 00  
 T<sub>0</sub> OUT = 01  
 T<sub>1</sub> OUT = 10  
 INTERNAL CLOCK OUT = 11

T<sub>IN</sub> MODES  
 EXTERNAL CLOCK INPUT = 00  
 GATE INPUT = 01  
 TRIGGER INPUT (NON-RETRIGGERABLE) = 10  
 TRIGGER INPUT = 11 (RETRIGGERABLE)

0 = NO FUNCTION  
 1 = LOAD T<sub>0</sub>  
 0 = DISABLE T<sub>0</sub> COUNT  
 1 = ENABLE T<sub>0</sub> COUNT  
 0 = NO FUNCTION  
 1 = LOAD T<sub>1</sub>  
 0 = DISABLE T<sub>1</sub> COUNT  
 1 = ENABLE T<sub>1</sub> COUNT

**R245 PRE0**  
**Prescaler 0 Register**  
 (F5H; Write Only)

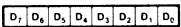


COUNT MODE  
 0 = T<sub>0</sub> SINGLE-PASS  
 1 = T<sub>0</sub> MODULO-N

RESERVED (MUST BE 0)

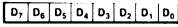
PRESCALER MODULO  
 (RANGE: 1-64 DECIMAL  
 01-00 HEX)

**R242 T1**  
**Counter Timer 1 Register**  
 (F2H; Read/Write)



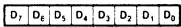
T<sub>1</sub> INITIAL VALUE (WHEN WRITTEN)  
 (RANGE 1-256 DECIMAL 01 00 HEX)  
 T<sub>1</sub> CURRENT VALUE (WHEN READ)

**R246 P2M**  
**Port 2 Mode Register**  
 (F6H; Write Only)



P<sub>20</sub>-P<sub>27</sub> I/O DEFINITION  
 0 DEFINES BIT AS OUTPUT  
 1 DEFINES BIT AS INPUT

**R243 PRE1**  
**Prescaler 1 Register**  
 (F3H; Write Only)

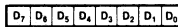


COUNT MODE  
 0 = T<sub>1</sub> SINGLE-PASS  
 1 = T<sub>1</sub> MODULO-N

CLOCK SOURCE  
 1 T<sub>1</sub> INTERNAL  
 0 T<sub>1</sub> EXTERNAL TIMING INPUT (T<sub>IN</sub>) MODE

PRESCALER MODULO  
 (RANGE: 1-64 DECIMAL  
 01-00 HEX)

**R247 P3M**  
**Port 3 Mode Register**  
 (F7H; Write Only)



0 PORT 2 PULL-UPS OPEN DRAIN  
 1 PORT 2 PULL-UPS ACTIVE

RESERVED (MUST BE 0)

0 P<sub>32</sub> = INPUT P<sub>35</sub> = OUTPUT  
 1 P<sub>32</sub> =  $\overline{\text{DAV0}}/\text{RDY0}$  P<sub>35</sub> =  $\text{RDY0}/\overline{\text{DAV0}}$

0 P<sub>33</sub> = INPUT P<sub>34</sub> = OUTPUT  
 1 P<sub>33</sub> =  $\overline{\text{DAV1}}/\text{RDY1}$  P<sub>34</sub> =  $\text{RDY1}/\overline{\text{DAV1}}$

0 P<sub>31</sub> = INPUT (T<sub>IN</sub>) P<sub>36</sub> = OUTPUT (T<sub>OUT</sub>)  
 1 P<sub>31</sub> =  $\overline{\text{DAV2}}/\text{RDY2}$  P<sub>36</sub> =  $\text{RDY2}/\overline{\text{DAV2}}$

0 P<sub>30</sub> = INPUT P<sub>37</sub> = OUTPUT  
 1 P<sub>30</sub> = SERIAL IN P<sub>37</sub> = SERIAL OUT

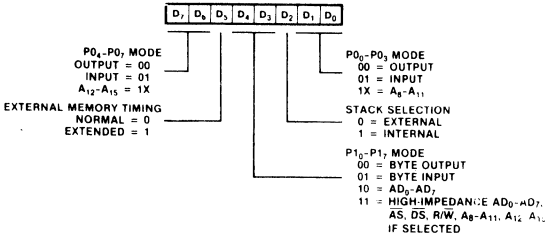
0 PARITY OFF  
 1 PARITY ON

Figure 17. Control Registers

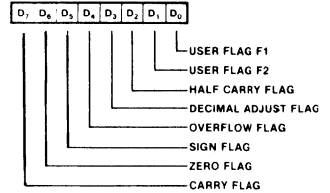


Registers (Continued)

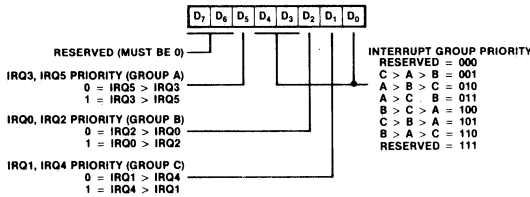
**R248 P01M**  
Port 0 and 1 Mode Register  
(F8<sub>H</sub>; Write Only)



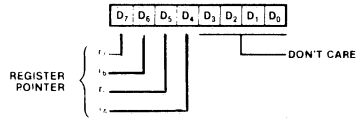
**R252 FLAGS**  
Flag Register  
(FC<sub>H</sub>; Read/Write)



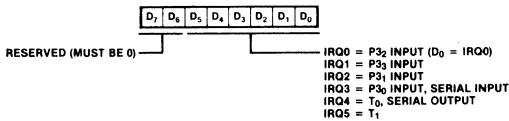
**R249 IPR**  
Interrupt Priority Register  
(F9<sub>H</sub>; Write Only)



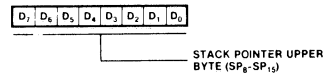
**R253 RP**  
Register Pointer  
(FD<sub>H</sub>; Read/Write)



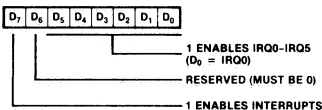
**R250 IRQ**  
Interrupt Request Register  
(FA<sub>H</sub>; Read/Write)



**R254 SHP**  
Stack Pointer  
(FE<sub>H</sub>; Read/Write)



**R251 IMR**  
Interrupt Mask Register  
(FB<sub>H</sub>; Read/Write)



**R255 SPL**  
Stack Pointer  
(FF<sub>H</sub>; Read/Write)

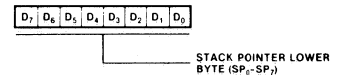


Figure 17. Control Registers (Continued)



# Z86E11

## Opcode Map

Lower Nibble (Hex)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
Upper Nibble (Hex)	0	6,5 DEC R <sub>1</sub>	6,5 DEC IR <sub>1</sub>	6,5 ADD r <sub>1</sub> , r <sub>2</sub>	6,5 ADD r <sub>1</sub> , IR <sub>2</sub>	10,5 ADD R <sub>2</sub> , R <sub>1</sub>	10,5 ADD IR <sub>2</sub> , R <sub>1</sub>	10,5 ADD R <sub>1</sub> , IM	10,5 ADD IR <sub>1</sub> , IM	6,5 LD r <sub>1</sub> , R <sub>2</sub>	6,5 LD r <sub>2</sub> , R <sub>1</sub>	12/10,5 DJNZ r <sub>1</sub> , RA	12/10,0 JR cc, RA	6,5 LD r <sub>1</sub> , IM	12/10,0 JP cc, DA	6,5 INC r <sub>1</sub>	
	1	6,5 RLC R <sub>1</sub>	6,5 RLC IR <sub>1</sub>	6,5 ADC r <sub>1</sub> , r <sub>2</sub>	6,5 ADC r <sub>1</sub> , IR <sub>2</sub>	10,5 ADC R <sub>2</sub> , R <sub>1</sub>	10,5 ADC IR <sub>2</sub> , R <sub>1</sub>	10,5 ADC R <sub>1</sub> , IM	10,5 ADC IR <sub>1</sub> , IM								
	2	6,5 INC R <sub>1</sub>	6,5 INC IR <sub>1</sub>	6,5 SUB r <sub>1</sub> , r <sub>2</sub>	6,5 SUB r <sub>1</sub> , IR <sub>2</sub>	10,5 SUB R <sub>2</sub> , R <sub>1</sub>	10,5 SUB IR <sub>2</sub> , R <sub>1</sub>	10,5 SUB R <sub>1</sub> , IM	10,5 SUB IR <sub>1</sub> , IM								
	3	8,0 JP IRR <sub>1</sub>	6,1 SRP IM	6,5 SBC r <sub>1</sub> , r <sub>2</sub>	6,5 SBC r <sub>1</sub> , IR <sub>2</sub>	10,5 SBC R <sub>2</sub> , R <sub>1</sub>	10,5 SBC IR <sub>2</sub> , R <sub>1</sub>	10,5 SBC R <sub>1</sub> , IM	10,5 SBC IR <sub>1</sub> , IM								
	4	8,5 DA R <sub>1</sub>	8,5 DA IR <sub>1</sub>	6,5 OR r <sub>1</sub> , r <sub>2</sub>	6,5 OR r <sub>1</sub> , IR <sub>2</sub>	10,5 OR R <sub>2</sub> , R <sub>1</sub>	10,5 OR IR <sub>2</sub> , R <sub>1</sub>	10,5 OR R <sub>1</sub> , IM	10,5 OR IR <sub>1</sub> , IM								
	5	10,5 POP R <sub>1</sub>	10,5 POP IR <sub>1</sub>	6,5 AND r <sub>1</sub> , r <sub>2</sub>	6,5 AND r <sub>1</sub> , IR <sub>2</sub>	10,5 AND R <sub>2</sub> , R <sub>1</sub>	10,5 AND IR <sub>2</sub> , R <sub>1</sub>	10,5 AND R <sub>1</sub> , IM	10,5 AND IR <sub>1</sub> , IM								
	6	6,5 COM R <sub>1</sub>	6,5 COM IR <sub>1</sub>	6,5 TCM r <sub>1</sub> , r <sub>2</sub>	6,5 TCM r <sub>1</sub> , IR <sub>2</sub>	10,5 TCM R <sub>2</sub> , R <sub>1</sub>	10,5 TCM IR <sub>2</sub> , R <sub>1</sub>	10,5 TCM R <sub>1</sub> , IM	10,5 TCM IR <sub>1</sub> , IM								
	7	10/12,1 PUSH R <sub>2</sub>	12/14,1 PUSH IR <sub>2</sub>	6,5 TM r <sub>1</sub> , r <sub>2</sub>	6,5 TM r <sub>1</sub> , IR <sub>2</sub>	10,5 TM R <sub>2</sub> , R <sub>1</sub>	10,5 TM IR <sub>2</sub> , R <sub>1</sub>	10,5 TM R <sub>1</sub> , IM	10,5 TM IR <sub>1</sub> , IM								
	8	10,5 DECW RR <sub>1</sub>	10,5 DECW IR <sub>1</sub>	12,0 LDE r <sub>1</sub> , IRR <sub>2</sub>	18,0 LDEI IR <sub>1</sub> , IRR <sub>2</sub>												6,1 DI
	9	6,5 RL R <sub>1</sub>	6,5 RL IR <sub>1</sub>	12,0 LDE r <sub>2</sub> , IRR <sub>1</sub>	18,0 LDEI IR <sub>2</sub> , IRR <sub>1</sub>												6,1 EI
	A	10,5 INCW RR <sub>1</sub>	10,5 INCW IR <sub>1</sub>	6,5 CP r <sub>1</sub> , r <sub>2</sub>	6,5 CP r <sub>1</sub> , IR <sub>2</sub>	10,5 CP R <sub>2</sub> , R <sub>1</sub>	10,5 CP IR <sub>2</sub> , R <sub>1</sub>	10,5 CP R <sub>1</sub> , IM	10,5 CP IR <sub>1</sub> , IM								14,0 RET
	B	6,5 CLR R <sub>1</sub>	6,5 CLR IR <sub>1</sub>	6,5 XOR r <sub>1</sub> , r <sub>2</sub>	6,5 XOR r <sub>1</sub> , IR <sub>2</sub>	10,5 XOR R <sub>2</sub> , R <sub>1</sub>	10,5 XOR IR <sub>2</sub> , R <sub>1</sub>	10,5 XOR R <sub>1</sub> , IM	10,5 XOR IR <sub>1</sub> , IM								16,0 IRET
	C	6,5 RRC R <sub>1</sub>	6,5 RRC IR <sub>1</sub>	12,0 LDC r <sub>1</sub> , IRR <sub>2</sub>	18,0 LDCI IR <sub>1</sub> , IRR <sub>2</sub>				10,5 LD r <sub>1</sub> , x, R <sub>2</sub>								6,5 RCF
	D	6,5 SRA R <sub>1</sub>	6,5 SRA IR <sub>1</sub>	12,0 LDC r <sub>2</sub> , IRR <sub>1</sub>	18,0 LDCI IR <sub>2</sub> , IRR <sub>1</sub>	20,0 CALL* IRR <sub>1</sub>		20,0 CALL DA	10,5 LD r <sub>2</sub> , x, R <sub>1</sub>								6,5 SCF
	E	6,5 RR R <sub>1</sub>	6,5 RR IR <sub>1</sub>		6,5 LD r <sub>1</sub> , IR <sub>2</sub>	10,5 LD R <sub>2</sub> , R <sub>1</sub>	10,5 LD IR <sub>2</sub> , R <sub>1</sub>	10,5 LD R <sub>1</sub> , IM	10,5 LD IR <sub>1</sub> , IM								6,5 CCF
	F	8,5 SWAP R <sub>1</sub>	8,5 SWAP IR <sub>1</sub>		6,5 LD r <sub>1</sub> , r <sub>2</sub>		10,5 LD R <sub>2</sub> , IR <sub>1</sub>										6,0 NOP

Bytes per Instruction

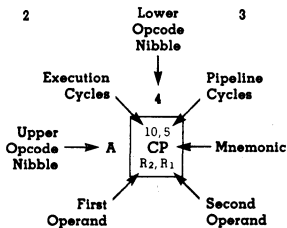
2

3

2

3

1



### Legend:

R = 8-Bit Address  
r = 4-Bit Address  
R<sub>1</sub> or r<sub>1</sub> = Dat Address  
R<sub>2</sub> or r<sub>2</sub> = Src Address

### Sequence:

Opcode, First Operand, Second Operand

Note: The blank areas are not defined.

\*2-byte instruction; fetch cycle appears as a 3-byte instruction

### Absolute Maximum Ratings

Voltages on all pins with respect to GND ..... -0.3 V to +7.0 V  
 Operating Ambient Temperature ..... 0°C to +70°C  
 Storage Temperature ..... -65°C to +150°C

Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

### Test Conditions

The characteristics below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND. Positive current flows into the reference pin. Standard conditions are as follows:

- +4.75 V ≤ V<sub>CC</sub> ≤ +5.25 V
- GND = 0 V
- 0°C ≤ T<sub>A</sub> ≤ +70°C

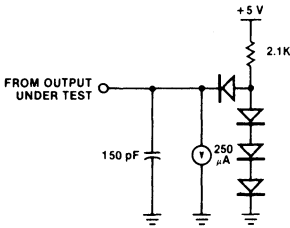


Figure 18. Test Load 1

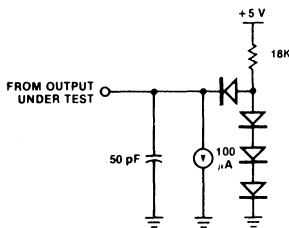


Figure 19. Test Load 2

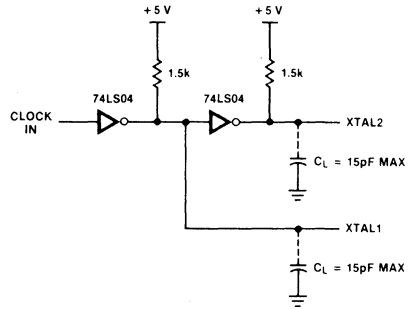


Figure 20. External Clock Interface Circuit





## Z86E11

### DC Characteristics

Symbol	Parameter	Min	Max	Unit	Condition
V <sub>CH</sub>	Clock Input High Voltage	3.8	V <sub>CC</sub>	V	Driven by External Clock Generator
V <sub>CL</sub>	Clock Input Low Voltage	-0.3	0.8	V	Driven by External Clock Generator
V <sub>IH</sub>	Input High Voltage	2.0	V <sub>CC</sub>	V	
V <sub>IL</sub>	Input Low Voltage	-0.3	0.8	V	
V <sub>RH</sub>	Reset Input High Voltage	3.8	V <sub>CC</sub>	V	
V <sub>RL</sub>	Reset Input Low Voltage	-0.3	0.8	V	
V <sub>OH</sub>	Output High Voltage	2.4		V	I <sub>OH</sub> = -250 $\mu$ A
V <sub>OL</sub>	Output Low Voltage		0.4	V	I <sub>OL</sub> = +2.0 mA
I <sub>IL</sub>	Input Leakage	-10	10	$\mu$ A	0 V $\leq$ V <sub>IN</sub> $\leq$ +5.25 V
I <sub>OL</sub>	Output Leakage	-10	10	$\mu$ A	0 V $\leq$ V <sub>IN</sub> $\leq$ +5.25 V
I <sub>IR</sub>	Reset Input Current		-50	$\mu$ A	V <sub>CC</sub> = +5.25 V, V <sub>RL</sub> = 0 V
I <sub>CC</sub>	V <sub>CC</sub> Supply Current		120	mA	
I <sub>MM</sub>	V <sub>MM</sub> Supply Current		10	mA	Power Down Mode
V <sub>MM</sub>	Backup Supply Voltage	3	V <sub>CC</sub>	V	Power Down

**External I/O or Memory Read and Write Timing**

No	Symbol	Parameter	Z86E11		Z86E11A		Notes*†
			Min	Max	Min	Max	
1	TdA(AS)	Address Valid to $\overline{AS}$ $\uparrow$ Delay	50		35		1,2,3
2	TdAS(A)	$\overline{AS}$ $\uparrow$ to Address Float Delay	70		45		1,2,3
3	TdAS(DR)	$\overline{AS}$ $\uparrow$ to Read Data Required Valid		360		220	1,2,3,4
4	TwAS	$\overline{AS}$ Low Width	80		55		1,2,3
5	TdAz(DS)	Address Float to $\overline{DS}$ $\downarrow$	0		0		1
6	TwDSR	$\overline{DS}$ (Read) Low Width	250		185		1,2,3,4
7	TwDSW	$\overline{DS}$ (Write) Low Width	160		110		1,2,3,4
8	TdDSR(DR)	$\overline{DS}$ $\downarrow$ to Read Data Required Valid		200		130	1,2,3,4
9	ThDR(DS)	Read Data to $\overline{DS}$ $\uparrow$ Hold Time	0		0		1
10	TdDS(A)	$\overline{DS}$ $\uparrow$ to Address Active Delay	70		45		1,2,3
11	TdDS(AS)	$\overline{DS}$ $\uparrow$ to $\overline{AS}$ $\downarrow$ Delay	70		55		1,2,3
12	TdR/W(AS)	R/W Valid to $\overline{AS}$ $\uparrow$ Delay	50		30		1,2,3
13	TdDS(R/W)	$\overline{DS}$ $\uparrow$ to R/W Not Valid	60		35		1,2,3
14	TdDW(DSW)	Write Data Valid to $\overline{DS}$ (Write) $\downarrow$ Delay	50		35		1,2,3
15	TdDS(DW)	$\overline{DS}$ $\uparrow$ to Write Data Not Valid Delay	70		45		1,2,3
16	TdA(DR)	Address Valid to Read Data Required Valid		410		255	1,2,3,4
17	TdAS(DS)	$\overline{AS}$ $\uparrow$ to $\overline{DS}$ $\downarrow$ Delay	80		55		1,2,3

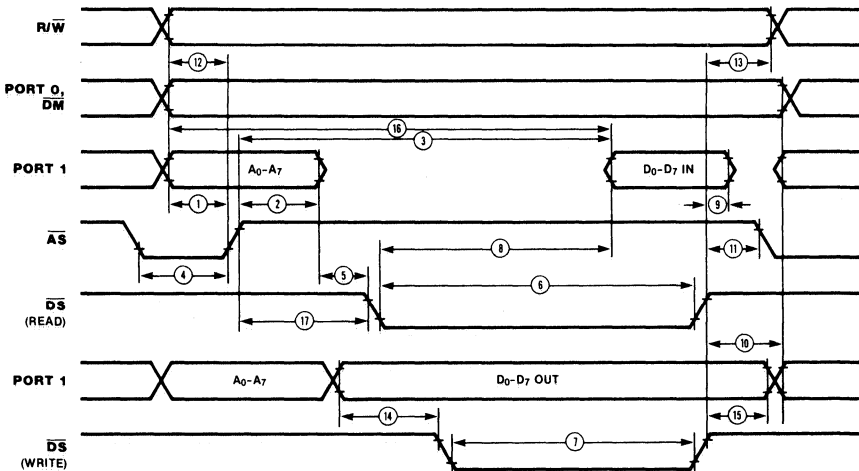
**NOTES:**

1. Test Load 1
2. Timing numbers given are for minimum T<sub>pC</sub>.
3. Also see clock cycle time dependent characteristics table.
4. When using extended memory timing add 2 T<sub>pC</sub>.

5. All timing references use 2.0 V for a logic "1" and 0.8 V for a logic "0".

\* All units in nanoseconds (ns).

† Timings are preliminary and subject to change.


**Figure 21. External I/O or Memory Read/Write**



Additional Timing Table

No	Symbol	Parameter	Z86E11		Z86E11A		Notes*†
			Min	Max	Min	Max	
1	TpC	Input Clock Period	125	1000	83	1000	1
2	TrC, TfC	Clock Input Rise And Fall Times		25		15	1
3	TwC	Input Clock Width	37		26		1
4	TwTinL	Timer Input Low Width	100		70		2
5	TwTinH	Timer Input High Width	3TpC		3TpC		2
6	TpTin	Timer Input Period	8TpC		8TpC		2
7	TrTin, TfTin	Timer Input Rise And Fall Times		100		100	2
8a	TwiL	Interrupt Request Input Low Time	100		70		2,3
8b	TwiH	Interrupt Request Input Low Time	3TpC		3TpC		2,3
9	TwiH	Interrupt Request Input High Time	3TpC		3TpC		2,3

NOTES:

1. Clock timing references uses 3.8 V for a logic "1" and 0.8 V for a logic "0".  
 2. Timing reference uses 2.0 V for a logic "1" and 0.8 V for a logic "0".

3. Interrupt request via Port 3 (P3<sub>1</sub>-P3<sub>3</sub>).  
 4. Interrupt request via Port 3 (P3<sub>0</sub>).  
 \* Units in nanoseconds (ns).

† Timings are preliminary and subject to change.

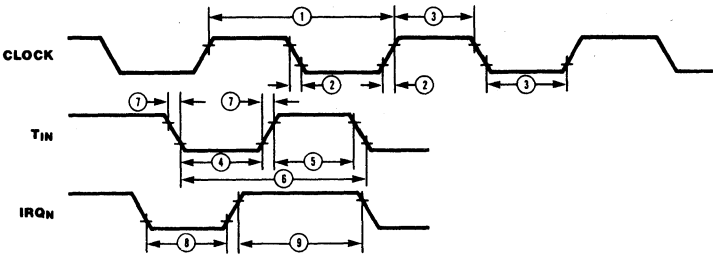


Figure 22. Additional Timing



### Handshake Timing

No	Symbol	Parameter	Z86E11		Z86E11A		Notes*†
			Min	Max	Min	Max	
1	TsDI(DAV)	Data In Setup Time	0		0		
2	ThDI(DAV)	Data In Hold Time	230		160		
3	TwDAV	Data Available Width	175		120		
4	TdDAVH(RDY)	$\overline{DAV}$ ↓ Input to RDY ↓ Delay		175		120	1,2
5	TdDAVO(RDY)	$\overline{DAV}$ ↓ Output to RDY ↓ Delay	0		0		1,3
6	TdDAVr(RDY)	$\overline{DAV}$ ↑ Input to RDY ↑ Delay		175		120	1,2
7	TdDAVo(RDY)	$\overline{DAV}$ ↑ Output to RDY ↑ Delay	0		0		1,3
8	TdDO(DAV)	Data Out to $\overline{DAV}$ ↓ Delay	50		30		1
9	TdRDY(DAV)	Rdy ↓ Input to DAV ↑ Delay	0	200	0	140	1

NOTES:

- 1. Test Load 1
- 2. Input handshake
- 3. Output handshake

4. All timing references use 2.0 V for a logic "1" and 0.8 V for a logic "0".

\* Units in nanoseconds (ns).

† Timings are preliminary and subject to change.

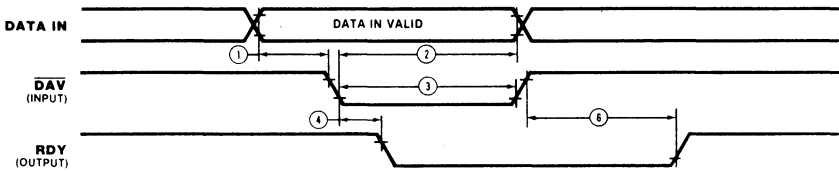


Figure 23a. Input Handshake

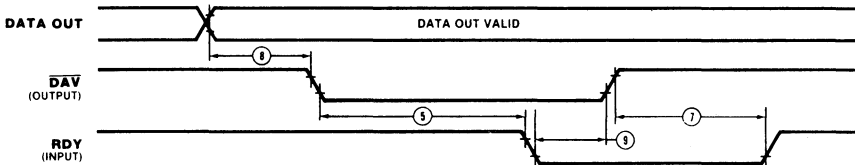


Figure 23b. Output Handshake



# Z86E11

## Clock-Cycle-Time-Dependent Characteristics

Number	Symbol	Z86E11 Equation	Z86E11A Equation
1	TdA(AS)	TpC-75	TpC-50
2	TdAS(A)	TpC-55	TpC-40
3	TdAS(DR)	4TpC-140*	4TpC-110*
4	TwAS	TpC-45	TpC-30
6	TwDSR	3TpC-125*	3TpC-65*
7	TwDSW	2TpC-90*	2TpC-55*
8	TdDSR(DR)	3TpC-175*	3TpC-120*
10	Td(DS)A	TpC-55	TpC-40
11	TdDS(AS)	TpC-55	TpC-30
12	TdR/W(AS)	TpC-75	TpC-55
13	TdDS(R/W)	TpC-65	TpC-50
14	TdDW(DSW)	TpC-75	TpC-50
15	TdDS(DW)	TpC-55	TpC-40
16	TdA(DR)	5TpC-215*	5TpC-160*
17	TdAS(DS)	TpC-45	TpC-30

\* Add 2TpC when using extended memory timing.

## Ordering Information

Type	Package	Temp.	Clock	Description
Z86E11 D1	Ceramic	0/+70°C	8 MHz	4K EPROM Microcomputer
Z86E11 D6	Ceramic	-40/+85°C		
Z86E11A D1	Ceramic	0/+70°C	12 MHz	
Z86E11A D6	Ceramic	-40/+85°C		

## Z8 8K EPROM Microcomputer

- Complete microcomputer, 4K bytes of EPROM, 240 bytes of RAM, 32 I/O lines, and up to 56K bytes addressable external space each for program and data memory. Fully compatible with standard ROM version.
- 256-byte register file, including 236 general-registers, four I/O port registers, and 16 status and control registers.
- Minimum instruction execution time 1  $\mu$ s at 12 MHz.
- Vectored priority interrupts for I/O, counter/timers, and UART.
- Full-duplex UART and two programmable 8-bit counter/timers, each with a 6-bit programmable prescaler.
- Register Pointer so that short, fast instructions can access any of nine working-register groups in 1.5  $\mu$ s (8MHz).
- On-chip oscillator which accepts crystal or external clock drive.
- Low-power standby option which retains contents of general-purpose registers.
- Single +5 V power supply - all pins TTL compatible.
- Two Eprom programming modes:
  - Eprom-like, using a standard Eprom programmer,
  - Autoprogram, self-programming during normal program execution.
  - An on-chip ROM provides a Program/Verify facility to allow a simple and time-efficient self-program operation.
- Integrated programmable protection avoids EPROM content read-out.
- Available in 8 and 12 MHz versions.

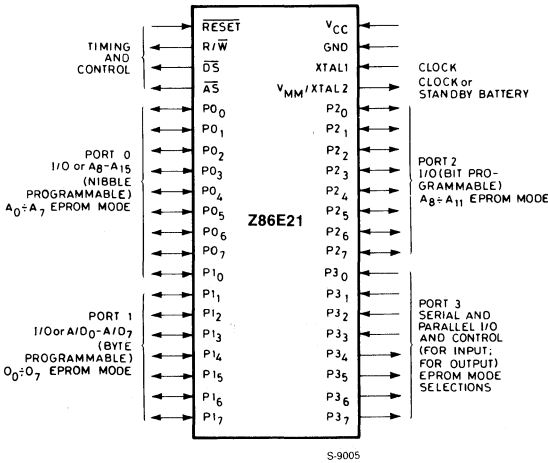


Figure 1. Logic Functions

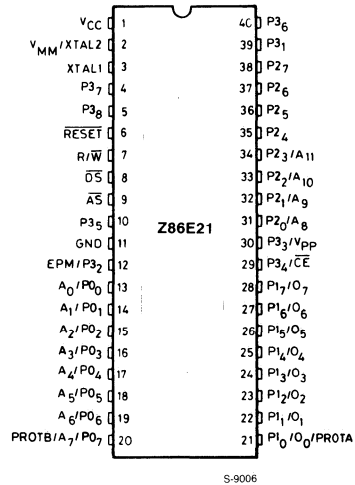


Figure 2. Pin Configuration



## General Description

The Z86E21 microcomputer is an EPROM member of the Z8 family; completely developed by SGS, it maintains the sophisticated architecture and full compatibility with the currently available ROM-based units.

It can be configured as a stand-alone microcomputer with 8K bytes of internal EPROM, or as a traditional microcomputer that manages up to 112K bytes of external memory, or as a parallel-processing element in a system with other processors and peripheral controllers.

The 8K × 8 on-board EPROM can be programmed in two modes, Eprom-like and Autoprogram. In Eprom-like, the programming procedure is similar to that for

a M2764, with the only exception being for the programming voltage which must be 12.5 V related to the SGS NMOS-E3 used technology. Autoprogram permits byte-programming during normal microcomputer program execution.

An important facility is the programmable read-out protections which allow the user to inhibit external access to proprietary program code by programming 2 non-volatile transistors. These locks can be reset only by erasing the entire EPROM array.

For its characteristics, the Z86E21 can be considered as a low cost development tool for the Z8 microcomputer family.

## Architecture

Z86E21 architecture is characterized by a flexible I/O scheme, an efficient register and address space structure and a number of ancillary features that are helpful in many applications.

Microcomputer applications demand powerful I/O capabilities. The Z86E21 fulfills this with 32 pins dedicated to input/output. These lines are grouped into four ports of eight lines each and are configurable under software control to provide timing, status signals, serial or parallel I/O with or without handshake, address/data bus for interfacing external memory, and address, data and selections in EPROM mode.

Because the multiplexed address/data bus is merged with the I/O-oriented ports, the Z86E21 can assume many different memory and I/O configurations. These configurations range from a self-contained microcomputer to a microprocessor that can address 112K bytes of external memory (Figure 3).

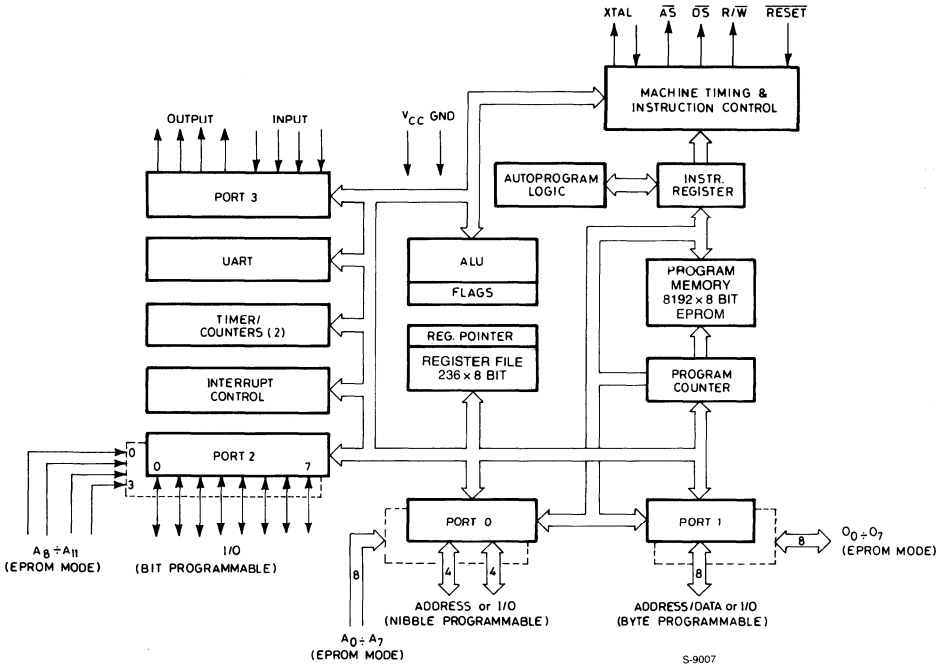
Three basic address spaces are available to support this wide range of configurations: program memory (internal and external), data memory (external) and the register file

(internal). The 256-byte random-access register file is composed of 236 general-purpose registers, four I/O port registers, and 16 control and status registers.

To unburden the program from coping with real-time problems such as serial data communication and counting/timing, an asynchronous receiver/transmitter (UART) and two counter/timers with a large number of user-selectable modes are offered on-chip. Hardware support for the UART is minimized because one of the on-chip timers supplies the bit rate.

An autoprogram logic permits Eprom byte-programming during the normal microcomputer program execution, using the STORE constant instruction.

This permits the microcomputer to look at the contents of the working register's register for an external RAM allocated to the program memory space, addressed using two of the register file registers. The renamed external RAM was expressly developed for the Autoprogram function, thus it is externally inaccessible.

**Architecture (Continued)**

**Figure 3. Block Diagram**
**Pin Description**

**P0<sub>0</sub>-P0<sub>7</sub>.** *I/O Port Lines* (input/outputs, TTL compatible). 8 lines Nibble Programmable that can be configured under program control for I/O or external memory interface and, A<sub>0</sub>-A<sub>7</sub> in EPROM mode. P0<sub>7</sub> can be configured as Read-out protection in memory expansion mode (PROTB), applying a high voltage level (10 V)

**P1<sub>0</sub>-P1<sub>7</sub>.** *I/O Port Lines* (input/outputs, TTL compatible). 8 lines Byte Programmable that can be configured under program control for I/O or multiplexed address (A<sub>0</sub>-A<sub>7</sub>) and data (D<sub>0</sub>-D<sub>7</sub>) lines used to interface with program/data memory, and O<sub>0</sub>-O<sub>7</sub> in EPROM mode. P1<sub>0</sub> can be configured as Read-out

protection in Testing-mode and EPROM-mode (PROTA), applying a high voltage level (10 V).

**P2<sub>0</sub>-P2<sub>7</sub>.** *I/O Port Lines* (input/outputs, TTL compatible). 8 lines Bit Programmable where the 4 less significant bits can be configured as A<sub>8</sub>-A<sub>11</sub> in EPROM mode.

**P3<sub>0</sub>-P3<sub>7</sub>.** *I/O Port Lines* (TTL compatible) 4 lines input (P3<sub>0</sub>-P3<sub>3</sub>), 4 lines output (P3<sub>4</sub>-P3<sub>7</sub>). They can also be configured as control lines. In EPROM mode: P3<sub>2</sub> becomes EPM (Eprom-like) when a high voltage level ( $\geq 7$  V) is applied, P3<sub>3</sub> becomes V<sub>pp</sub> (12.5 V  $\pm$  300 mV), P3<sub>4</sub> becomes CE to perform program enable/verify.





## Pin Description (Continued)

**$\overline{AS}$ .** *Address Strobe* (output, active Low). Address Strobe is pulsed once at the beginning of each machine cycle. Addresses output via Port 1 for all external program or data memory transfers are valid at the trailing edge of  $\overline{AS}$ . Under program control,  $\overline{AS}$  can be placed in the high-impedance state along with ports 0 and 1, Data Strobe and Read/Write.

**$\overline{DS}$ .** *Data Strobe* (output, active Low). Data Strobe is activated once for each external memory transfer.

**$\overline{RESET}$ .** *Reset* (input, active Low).  $\overline{RESET}$

initializes the Z86E21. When  $\overline{RESET}$  is deactivated, program execution begins from internal program location  $000C_H$ .

**$R/\overline{W}$ .** *Read/Write* (output).  $R/\overline{W}$  is Low when the Z86E21 is writing to external program or data memory.

**$XTAL1$ ,  $XTAL2$ .** *Crystal 1, Crystal 2* (time-base input and output). These pins connect a series-resonant crystal (8 MHz maximum) or an external single-phase clock (8 MHz maximum) to the on-chip clock oscillator and buffer.

## Address Spaces

**Program Memory.** The 16-bit program counter addresses 64K bytes of program memory space. Program memory can be located in two areas: one internal and the other external (Figure 4). The first 8192 bytes consist of on-chip EPROM. At addresses 8192 and greater, the Z86E21 executes external program memory fetches.

The first 12 bytes of program memory are reserved for the interrupt vectors. These locations contain six 16-bit vectors that correspond to the six available interrupts.

**Data Memory.** The Z86E21 can address 56K bytes of external data memory beginning at locations 8192 (Figure 5). External data memory may be included with or separated from the external program memory space.  $\overline{DM}$ , an optional I/O function that can be programmed to appear on pin  $P3_4$ , is used to distinguish between data and program memory space.

**Register File.** The 256-byte register file includes four I/O port registers (R0-R3), 236 general-purpose registers (R4-R127) and 16

control and status registers (R240-R255). These registers are assigned the address locations shown in Figure 6.

Z86E21 instructions can access registers directly or indirectly with an 8-bit address field. The Z86E21 also allows short 4-bit register addressing using the Register Pointer (one of the control registers). In the 4-bit mode, the register file is divided into nine working-register groups, each occupying 16 contiguous locations (Figure 7). The Register Pointer addresses the starting location of the active working-register group.

**Stacks.** Either the internal register file or the external data memory can be used for the stack. A 16-bit Stack Pointer (R254 and R255) is used for the external stack, which can reside anywhere in data memory between locations 8192 and 65535. An 8-bit Stack Pointer (R255) is used for the internal stack that resides within the 236 general-purpose registers (R4-R127).

### Address Spaces (Continued)

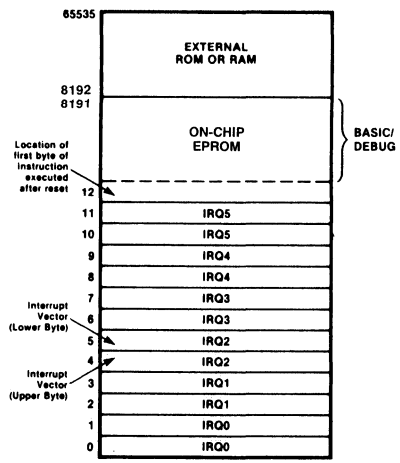


Figure 4. Program Memory Map

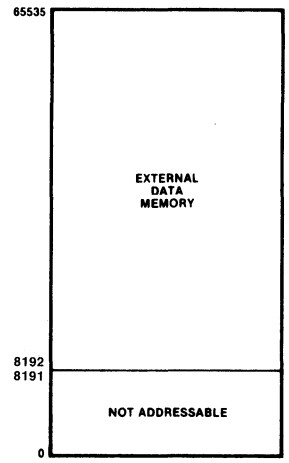


Figure 5. Data Memory Map

LOCATION	IDENTIFIERS
255	STACK POINTER (BITS 7-0) SPL
254	STACK POINTER (BITS 15-8) SPH
253	REGISTER POINTER RP
252	PROGRAM CONTROL FLAGS FLAGS
251	INTERRUPT MASK REGISTER IMR
250	INTERRUPT REQUEST REGISTER IRQ
249	INTERRUPT PRIORITY REGISTER IPR
248	PORTS 0-1 MODE P01M
247	PORT 3 MODE P3M
246	PORT 2 MODE P2M
245	T0 PRESCALER PRE0
244	TIMER/COUNTER 0 T0
243	T1 PRESCALER PRE1
242	TIMER/COUNTER 1 T1
241	TIMER MODE TMR
240	SERIAL I/O SIO
239	
GENERAL-PURPOSE REGISTERS	
4	
3	PORT 3 P3
2	PORT 2 P2
1	
0	PORT 0 P0

Figure 6. The Register File

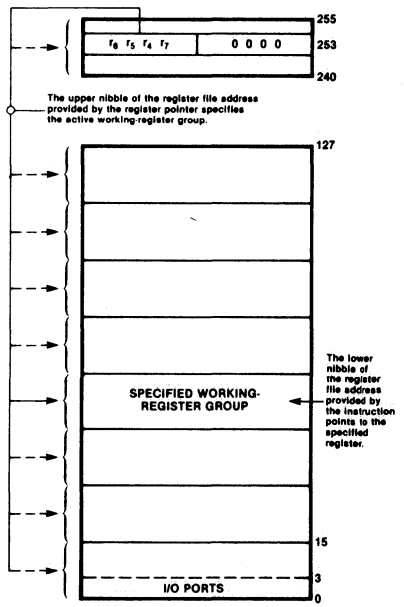


Figure 7. The Register Pointer



### Serial Input/Output

Port 3 lines P3<sub>0</sub> and P3<sub>7</sub> can be programmed as serial I/O lines for full-duplex serial asynchronous receiver/transmitter operation. The bit rate is controlled by Counter/Timer 0, with a maximum rate of 62.5K bits/second.

The Z86E21 automatically adds a start bit and two stop bits to transmitted data (Figure 8). Odd parity is also available as an option. Eight data bits are always transmitted,

regardless of parity selection. If parity is enabled, the eighth bit is the odd parity bit. An interrupt request (IRQ<sub>4</sub>) is generated on all transmitted characters.

Received data must have a start bit, eight data bits and at least one stop bit. If parity is on, bit 7 of the received data is replaced by a parity error flag. Received characters generate the IRQ<sub>3</sub> interrupt request.

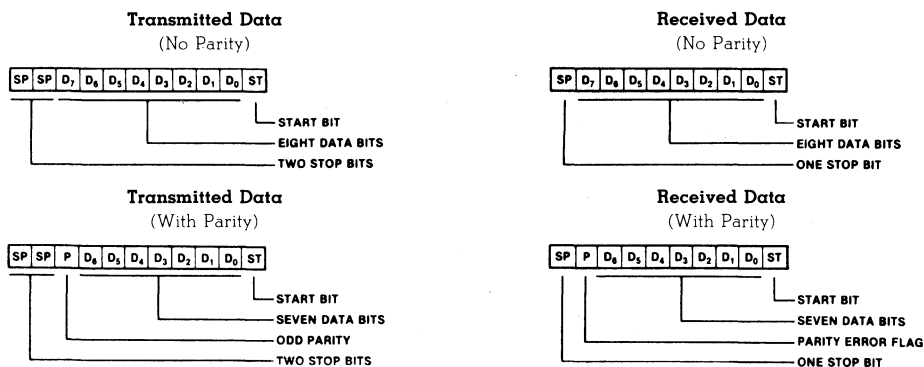


Figure 8. Serial Data Formats

### Counter/Timers

The Z86E21 contains two 8-bit programmable counter/timers (T<sub>0</sub> and T<sub>1</sub>), each driven by its own 6-bit programmable prescaler. The T<sub>1</sub> prescaler can be driven by internal or external clock sources; however, the T<sub>0</sub> prescaler is driven by the internal clock only.

The 6-bit prescaler can divide the input frequency of the clock source by any number from 1 to 64. Each prescaler drives its counter, which decrements the value (1 to 256) that has been loaded into the counter. When the counter reaches the end of count, a timer interrupt request—IRQ<sub>4</sub> (T<sub>0</sub>) or IRQ<sub>5</sub> (T<sub>1</sub>)—is generated.

The counters can be started, stopped, restarted to continue, or restarted from the initial value. The counters can also be programmed to stop upon reaching zero (single-pass mode) or to automatically reload

the initial value and continue counting (modulo-n continuous mode). The counters, but not the prescalers, can be read any time without disturbing their value or count mode.

The clock source for T<sub>1</sub> is user-definable and can be the internal microprocessor clock (4 MHz maximum) divided by four, or an external signal input via Port 3. The Timer Mode register configures the external timer input as an external clock (1 MHz maximum), a trigger input that can be retriggerable or non-retriggerable, or as a gate input for the internal clock. The counter/timers can be programmably cascaded by connecting the T<sub>0</sub> output to the input of T<sub>1</sub>. Port 3 line P3<sub>6</sub> also serves as a timer output (T<sub>OUT</sub>) through which T<sub>0</sub>, T<sub>1</sub> or the internal clock can be output.

## I/O Ports

The Z86E21 has 32 lines dedicated to input and output. These lines are grouped into four ports of eight lines each and are configurable as input, output or address/data. Under software control, the ports can be programmed to provide address outputs, timing, status signals, serial I/O, and parallel I/O with or without handshake. All ports have active pull-ups and pull-downs compatible with TTL loads.

All the ports assume different configurations in EPROM mode.

**Port 1** can be programmed as a byte I/O port or as an address/data port for interfacing external memory. When used as an I/O port, Port 1 may be placed under handshake control. In this configuration, Port 3 lines P3<sub>3</sub> and P3<sub>4</sub> are used as the handshake controls RDY<sub>1</sub> and DAV<sub>1</sub> (Ready and Data Available).

Memory locations greater than 4096 are referenced through Port 1. To interface external memory, Port 1 must be programmed for the multiplexed Address/Data mode. If more than 256 external locations are required, Port 0 must output the additional lines.

Port 1 can be placed in the high-impedance state along with Port 0,  $\overline{AS}$ ,  $\overline{DS}$  and R/W, allowing the Z86E21 to share common resources in multiprocessor and DMA applications. Data transfers can be controlled by assigning P3<sub>3</sub> as a Bus Acknowledge input and P3<sub>4</sub> as a Bus Request output.

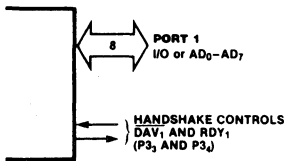


Figure 9a. Port 1

**Port 0** can be programmed as a nibble I/O port, or as an address port for interfacing external memory. When used as an I/O port, Port 0 may be placed under handshake control. In this configuration, Port 3 lines

lines P3<sub>2</sub> and P3<sub>5</sub> are used as the handshake controls DAV<sub>0</sub> and RDY<sub>0</sub>. Handshake signal assignment is dictated by the I/O direction of the upper nibble P0<sub>4</sub>-P0<sub>7</sub>.

For external memory references, Port 0 can provide address bits A<sub>8</sub>-A<sub>11</sub> (lower nibble) or A<sub>8</sub>-A<sub>15</sub> (lower and upper nibble) depending on the required address space. If the address range requires 12 bits or less, the upper nibble of Port 0 can be programmed independently as I/O while the lower nibble is used for addressing. When Port 0 nibbles are defined as address bits, they can be set to the high-impedance state along with Port 1 and the control signals  $\overline{AS}$ ,  $\overline{DS}$  and R/W.

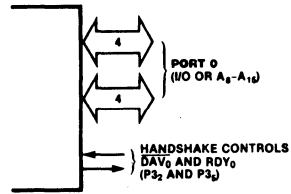


Figure 9b. Port 0

**Port 2** bits can be programmed independently as input or output. This port is always available for I/O operations. In addition, Port 2 can be configured to provide open-drain outputs.

Like Ports 0 and 1, Port 2 may also be placed under handshake control. In this configuration, Port 3 lines P3<sub>1</sub> and P3<sub>6</sub> are used as the handshake controls lines DAV<sub>2</sub> and RDY<sub>2</sub>. The handshake signal assignment for Port 3 lines P3<sub>1</sub> and P3<sub>6</sub> is dictated by the direction (input or output) assigned to bit 7 of Port 2.

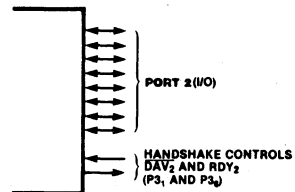


Figure 9c. Port 2



I/O Ports (Continued)

Port 3 lines can be configured as I/O or control lines. In either case, the direction of the eight lines is fixed as four input (P3<sub>0</sub>-P3<sub>3</sub>) and four output (P3<sub>4</sub>-P3<sub>7</sub>). For serial I/O, lines P3<sub>0</sub> and P3<sub>7</sub> are programmed as serial in and serial out respectively.

Port 3 can also provide the following control functions: handshake for Ports 0, 1 and 2 ( $\overline{DAV}$  and RDY); four external interrupt request signals (IRQ<sub>0</sub>-IRQ<sub>3</sub>); timer input and output signals (T<sub>IN</sub> and T<sub>OUT</sub>) and Data Memory Select ( $\overline{DM}$ ).

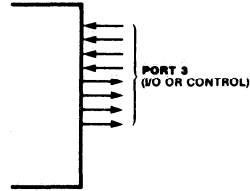


Figure 9d. Port 3

Interrupts

The Z86E21 allows six different interrupts from eight sources: the four Port 3 lines P3<sub>0</sub>-P3<sub>3</sub>, Serial In, Serial Out, and the two counter/timers. These interrupts are both maskable and prioritized. The Interrupt Mask register globally or individually enables or disables the six interrupt requests. When more than one interrupt is pending, priorities are resolved by a programmable priority encoder that is controlled by the Interrupt Priority register.

All Z86E21 interrupts are vectored. When an interrupt request is granted, and interrupt machine cycle is entered. This disables all

subsequent interrupts, saves the Program Counter and status flags, and branches to the program memory vector location reserved for that interrupt. This memory location and the next byte contain the 16-bit address of the interrupt service routine for that particular interrupt request.

Polled interrupt systems are also supported. To accommodate a polled structure, any or all of the interrupt inputs can be masked and the Interrupt Request register polled to determine which of the interrupt requests needs service.

Clock

The on-chip oscillator has a high-gain, parallel-resonant amplifier for connection to a crystal or to any suitable external clock source (XTAL1 = Input, XTAL2 = Output).

The crystal source is connected across XTAL1 and XTAL2, using the recommended capacitors ( $C_1 \leq 15$  pF) from each pin to

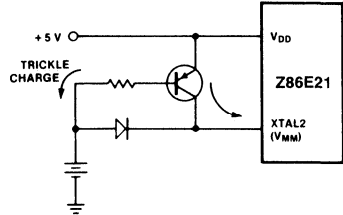
ground. The specifications for the crystal are as follows:

- AT cut, series resonant
- Fundamental types, 8/12 MHz maximum.
- Series resistance,  $R_s \leq 100 \Omega$ .

### Power Down Standby Option

The low-power standby mode allows power to be removed without losing the contents of the 236 general-purpose registers. This mode is available to the user as a bonding option whereby pin 2 (normally XTAL2) is replaced by the  $V_{MM}$  (standby) power supply input. This necessitates the use of an external clock generator (input = XTAL1) rather than a crystal source.

The removal of power, whether intended or due to power failure, must be preceded by a software routine that stores the appropriate status into the register file. Figure 10 shows the recommended circuit for a battery back-up supply system.



**Figure 10. Recommended Driver Circuit for Power Down Operation**

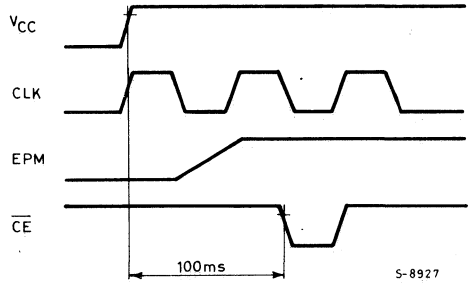
### EPROM Mode

Eprom-like programming. In this mode, the microcomputer memory is programmed, using a standard Eprom Programmer, with the same procedure as for our M2764 (64K EPROM). This has been made possible by the following Z86E21 configuration, where P1<sub>0</sub>-P1<sub>7</sub> are used as 8-bit I/O data (0<sub>0</sub>-0<sub>7</sub>), P0<sub>0</sub>-P0<sub>7</sub> and P2<sub>0</sub>-P2<sub>3</sub> are used as 12-bit Addresses (A<sub>0</sub>-A<sub>11</sub>); the microcomputer must be in Reset state, forcing the related pin to GND, and the clock must be active for the complete operation.

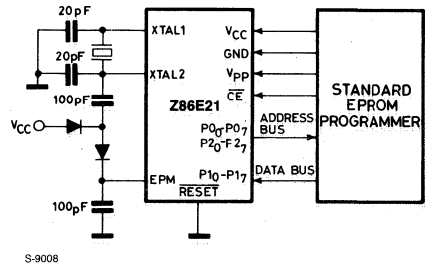
Three other pins are available for that purpose: the EPM pin on port P3<sub>2</sub>, which allows the microcomputer to recognize the Eprom-like condition when a high voltage ( $\geq 7$  V) is applied; the V<sub>pp</sub> pin on port P3<sub>3</sub>, which is used to furnish programming voltage fixed on 12.5 V  $\pm$  300mV; and the CE pin on port P3<sub>4</sub>, which is used to perform program enable/verify.

For a correct microcomputer set-up the V<sub>CC</sub> must be applied at least 100 ms before the programming procedure starting (Figure 11).

A simple interface board, described in Figure 12, allows programming to be carried out through use of a standard Eprom-programmer.



**Figure 11. Set-up Waveforms**



**Figure 12. EPROM Programmer Interface Board**



## EPROM Mode (Continued)

**Autoprogramming.** This mode permits programming one byte of the on-chip Eprom during normal microcomputer program execution. The instruction to be used is the Load Constant LCD @RR1,R2 (operating code D2).

This instruction allows the standard Z8 to load the contents of the working register to an external RAM memory allocated in the program memory space, addressed by a working register pair. The Z86E21 uses this instruction also to program the 4K bytes on-chip Eprom.

Addressing one of the on-chip Eprom bytes, using this instruction, the programming operation takes place when an high voltage level on the Vpp pin (12.5 V ± 300mV) is applied.

In this case, both the address and the data memory are internally stored for the necessary programming time, where the time is defined by the execution of 1024 NOP operations (1 NOP operation = 12 clock pulses). The programming time is contained between 1ms (12 MHz clock) and 12 ms (1 MHz clock).

As just mentioned, during this time, the CPU is internally forced to execute NOP instructions (operating code FF), while a RET instruction (operating code AF) is automatically executed at the end of programming.

For a correct program, restart is necessary to save the address of the Load Constant (LDC) next instruction in the Stack. This can be done by loading into the Stack the return address calling a Subroutine like follow, where, to permit a correct return to the main program, it is necessary to disable the interrupt before LDC execution.

```
.....  
DI  
CALL WRITE  
EI  
.....  
WRITE LDC @RR1,R2  
RET
```

**Programming Facility.** The most flexible way for on-chip Eprom programming is, as

we know, the use of a standard Eprom-programmer, selecting the Eprom-like facility, and using an appropriate interface board (Figure 12).

If, however, the planned operation is only a particular memory loading into the on-chip Eprom, it is possible to perform this operation in a much simpler way, using a board which allows the Z86E21 to read and load the renamed particular memory, using the autoloading procedure.

The software required for this operation is stored in the Z86E21 Test-memory (inaccessible). Figure 13 shows the autoloading program flow-chart.

When the microcomputer is forced in Test mode by applying a high voltage level ( $\geq 7$  V) on the Reset pin, ports P0 and P1 are configured as Address/Data to access the external memories.

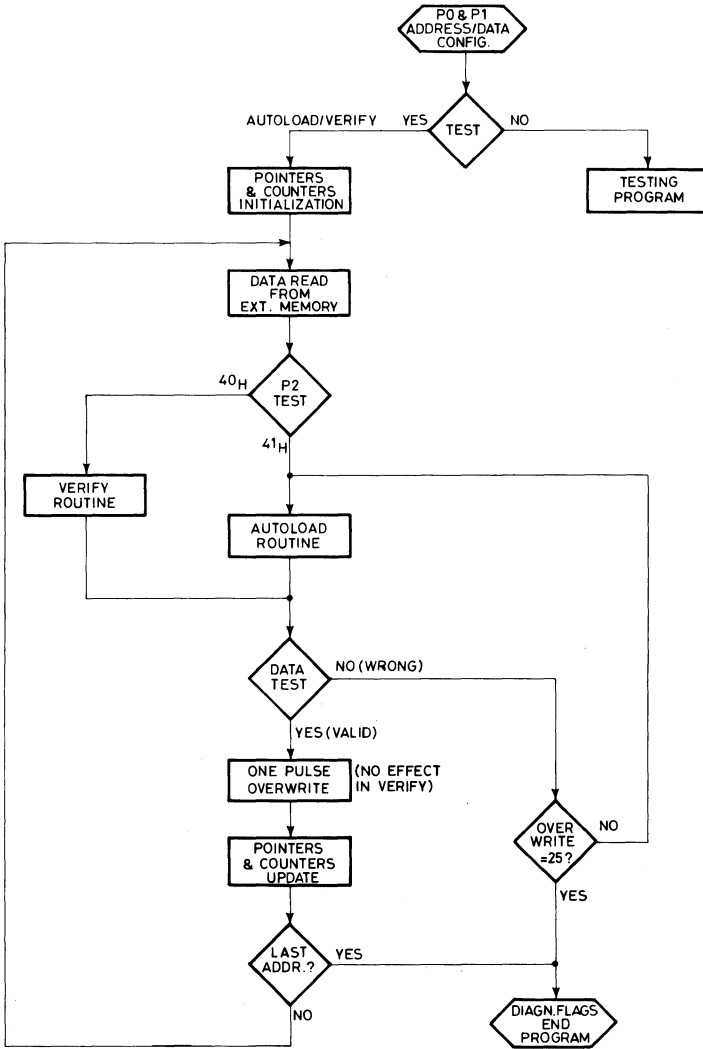
At this point, a test on port P2 is executed to decide if the on-chip Eprom autoloading is to be executed. This facility is accessed by forcing the values 40<sub>H</sub> or 41<sub>H</sub> on port P2 to execute, respectively, the Verify or Autoloading routine.

Consequently, the registers required for the operation are initialized, the data to be compared or stored is read, and the routine chosen is executed.

The Autoloading routine is an intelligent programming which executes a number of overwriting cycles equal to three times the number of programming cycles required to perform a correct byte programming (up to a maximum of 25). In this way, the on-chip Eprom programming time is optimized and equal to 25 sec. with an 8 MHz clock.

The verify routine is simply a byte-byte comparison between the external memory and the on-chip Eprom.

A possible failure, whether in Autoloading or Verify, produces a High logical level forced on P37. Similarly, when the operation is finished, the positive conclusion is underlined, bringing P35 High. An Autoloading/Verify Board diagram is shown in Figure 14, where the Vpp line control is necessary to not allow high voltage into the device when it has not yet been supplied.

**EPROM Mode (Continued)**


S-8929

**Figure 13. Autoloading Flow Chart**



## EPROM Mode (Continued)

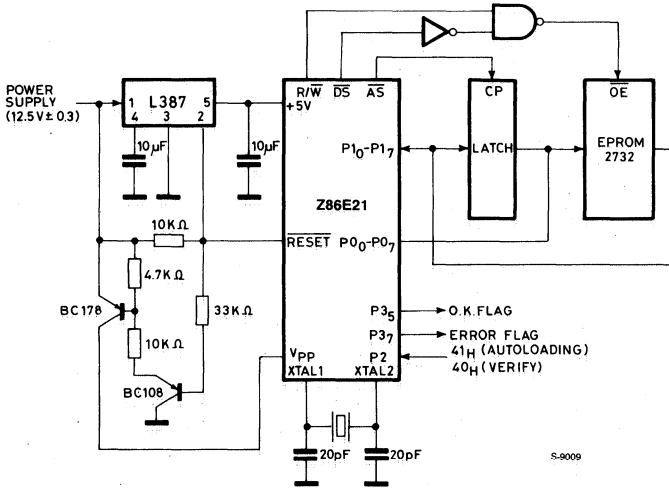


Figure 14. Autoloading/Verify Board

**Memory Read-Out Protection.** The protection, once activated, blocks reading memory content. Such reading can be carried out in two ways:

1. Entering Test Mode you can execute an external memory program which allows the on-chip Eprom reading through LOAD instructions execution.
2. Entering Eprom-like Mode, using the Verify facility.

Programming the first protection bit blocks reading in these two conditions (PROTA on port P1<sub>0</sub>).

Another protection bit (PROTB on port P0<sub>7</sub>) can be activated when the Z86E21 is in external memory configuration.

This protection prevents software

manipulation of the external memory from who decides to read the on-chip memory content for a complete understanding of the user application board.

When the Z86E21 works in external memory facility the ports P0 and P1 are configured as Address/Data bus, so that the external memory instructions can be executed during the normal microcomputer operation. These instructions can be also an appropriate routine able to pull out the all on-chip memory content using LOAD instructions. When the protection is activated, each reading attempt of the internal memory content, using LOAD instructions, is vainificated because the data out will be always "FF".

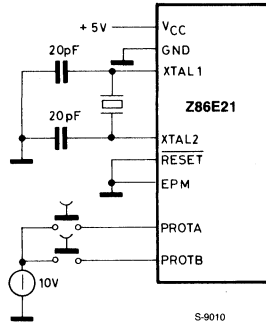
### EPROM Mode (Continued)

In consequence this protection activation inhibit the LOAD instructions execution from external to internal memory. To overcome this problem is necessary to call a "READ" routine written in on-chip memory space.

The protections are activated by programming 2 non-volatile transistors simply forcing 10 V for a time more than 100 ms on the desired pin, on condition that the microcomputer is in Reset state, the Clock signal is present and the EPM pin is not set.

If a complete protection is desired, both protections must be programmed.

A simple board diagram for read-out protection activation is shown in Figure 15.



**Figure 15. Read-out Protection Activation Diagram**

### Instruction Set Notation

**Addressing Modes.** The following notation is used to describe the addressing modes and instruction operations as shown in the instruction summary.

<b>IRR</b>	Indirect register pair or indirect working-register pair address
<b>Irr</b>	Indirect working-register pair only
<b>X</b>	Indexed address
<b>DA</b>	Direct address
<b>RA</b>	Relative address
<b>IM</b>	Immediate
<b>R</b>	Register or working-register address
<b>r</b>	Working-register address only
<b>IR</b>	Indirect-register or indirect working-register address
<b>Ir</b>	Indirect working-register address only
<b>RR</b>	Register pair or working register pair address

**Symbols.** The following symbols are used in describing the instruction set.

<b>dst</b>	Destination location or contents
<b>src</b>	Source location or contents
<b>cc</b>	Condition code (see list)
<b>@</b>	Indirect address prefix
<b>SP</b>	Stack pointer (control registers 254-255)
<b>PC</b>	Program counter
<b>FLAGS</b>	Flag register (control register 252)
<b>RP</b>	Register pointer (control register 253)

**IMR** Interrupt mask register (control register 251)

Assignment of a value is indicated by the symbol «←». For example,

$$dst \leftarrow dst + src$$

indicates that the source data is added to the destination data and the result is stored in the destination location. The notation "addr(n)" is used to refer to bit "n" of a given location. For example,

$$dst(7)$$

refers to bit 7 of the destination operand.

**Flags.** Control Register R252 contains the following six flags:

<b>C</b>	Carry flag	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>b7</td> <td colspan="6"></td> <td>b0</td> </tr> <tr> <td>C</td> <td>Z</td> <td>S</td> <td>V</td> <td>D</td> <td>I</td> <td>F2</td> <td>F1</td> </tr> </table>	b7							b0	C	Z	S	V	D	I	F2	F1
b7							b0											
C	Z		S	V	D	I	F2	F1										
<b>Z</b>	Zero flag																	
<b>S</b>	Sign flag																	
<b>V</b>	Overflow flag																	
<b>D</b>	Decimal-adjust flag																	
<b>H</b>	Half-carry flag																	

Affected flags are indicated by:

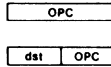
<b>0</b>	Cleared to zero
<b>1</b>	Set to one
<b>*</b>	Set or cleared according to operation
<b>—</b>	Unaffected
<b>X</b>	Undefined

**Condition Codes**

<b>Value</b>	<b>Mnemonic</b>	<b>Meaning</b>	<b>Flags Set</b>
1000		Always true	...
0111	C	Carry	C = 1
1111	NC	No carry	C = 0
0110	Z	Zero	Z = 1
1110	NZ	Not zero	Z = 0
1101	PL	Plus	S = 0
0101	MI	Minus	S = 1
0100	OV	Overflow	V = 1
1100	NOV	No overflow	V = 0
0110	EQ	Equal	Z = 1
1110	NE	Not equal	Z = 0
1001	GE	Greater than or equal	(S XOR V) = 0
0001	LT	Less than	(S XOR V) = 1
1010	GT	Greater than	[Z OR (S XOR V)] = 0
0010	LE	Less than or equal	[Z OR (S XOR V)] = 1
1111	UGE	Unsigned greater than or equal	C = 0
0111	ULT	Unsigned less than	C = 1
1011	UGT	Unsigned greater than	(C = 0 AND Z = 0) = 1
0011	ULE	Unsigned less than or equal	(C OR Z) = 1
0000		Never true	...



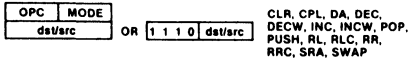
Instruction Formats



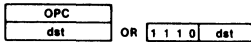
CCF, DI, EI, IRET, NOP, RCF, RET, SCF

INC r

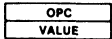
One-Byte Instruction



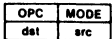
CLR, CPL, DA, DEC, DECW, INC, INCW, POP, PUSH, RL, RLC, RR, RRC, SRA, SWAP



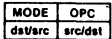
JP, CALL (Indirect)



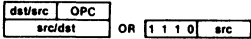
SRP



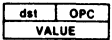
ADC, ADD, AND, CP, OR, SBC, SUB, TCM, TM, XOR



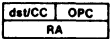
LD, LDE, LDEI, LDC, LDCI



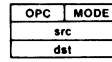
LD



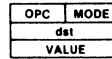
LD



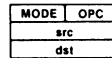
DJNZ, JR



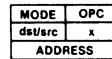
ADC, ADD, AND, CP, LD, OR, SBC, SUB, TCM, TM, XOR



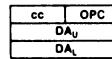
ADC, ADD, AND, CP, LD, OR, SBC, SUB, TCM, TM, XOR



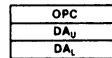
LD



LD



JP



CALL

Two-Byte instruction

Three-Byte instruction

Figure 16. Instruction Formats



# Z86E21


## Instruction Summary

Instruction and Operation	Addr Mode		Opcode Byte (Hex)	Flags Affected						
	dst	src		C	Z	S	V	D	H	
<b>ADC</b> dst,src dst - dst + src + C	(Note 1)		1□	*	*	*	*	0	*	
<b>ADD</b> dst,src dst - dst + src	(Note 1)		0□	*	*	*	*	0	*	
<b>AND</b> dst,src dst - dst AND src	(Note 1)		5□	-	*	*	0	-	-	
<b>CALL</b> dst SP - SP - 2 @SP - PC; PC - dst	DA IRR		D6 D4	-	-	-	-	-	-	
<b>CCF</b> C - NOT C			EF	*	-	-	-	-	-	
<b>CLR</b> dst dst - 0	R IR		B0 B1	-	-	-	-	-	-	
<b>COM</b> dst dst - NOT dst	R IR		60 61	-	*	*	0	-	-	
<b>CP</b> dst,src dst - src	(Note 1)		A□	*	*	*	*	-	-	
<b>DA</b> dst dst - DA dst	R IR		40 41	-	*	*	X	-	-	
<b>DEC</b> dst dst - dst - 1	R IR		00 01	-	*	*	*	-	-	
<b>DECW</b> dst dst - dst - 1	RR IR		80 81	-	*	*	*	-	-	
<b>DI</b> IMR (7) - 0			8F	-	-	-	-	-	-	
<b>DJNZ</b> r,dst r - r - 1 if r ≠ 0 PC - PC + dst Range: +127, -128	RA		rA r=0-F	-	-	-	-	-	-	
<b>EI</b> IMR (7) - 1			9F	-	-	-	-	-	-	
<b>INC</b> dst dst - dst + 1	r R IR		rE r=0-F 20 21	-	*	*	*	-	-	
<b>INCW</b> dst dst - dst + 1	RR IR		A0 A1	-	*	*	*	-	-	
<b>IRET</b> FLAGS - @SP; SP - SP + 1 PC - @SP; SP - SP + 2; IMR (7) - 1			BF	*	*	*	*	*	*	
<b>JP</b> cc,dst if cc is true PC - dst	DA IRR		cD c=0-F 30	-	-	-	-	-	-	
<b>JR</b> cc,dst if cc is true, PC - PC + dst Range: +127, -128	RA		cB c=0-F	-	-	-	-	-	-	

Instruction and Operation	Addr Mode		Opcode Byte (Hex)	Flags Affected						
	dst	src		C	Z	S	V	D	H	
<b>LD</b> dst,src dst - src	r r R	Im R r	rC r8 +9 r=0-F	-	-	-	-	-	-	
	r X r r Ir r R R R R IR	X r Ir r R R Im Im R	C7 D7 E3 F3 E4 E5 E6 E7 F5							
<b>LDC</b> dst,src dst - src	r	Irr r	C2 D2	-	-	-	-	-	-	
<b>LDCI</b> dst,src dst - src r - r + 1; rr - rr + 1	Ir Irr	Irr Ir	C3 D3	-	-	-	-	-	-	
<b>LDE</b> dst,src dst - src	r Irr	Irr r	82 92	-	-	-	-	-	-	
<b>LDEI</b> dst,src dst - src r - r + 1; rr - rr + 1	Ir Irr	Irr Ir	83 93	-	-	-	-	-	-	
<b>NOP</b>			FF	-	-	-	-	-	-	
<b>OR</b> dst,src dst - dst OR src	(Note 1)		4□	-	*	*	0	-	-	
<b>POP</b> dst dst - @SP SP - SP + 1	R IR		50 51	-	-	-	-	-	-	
<b>PUSH</b> src SP - SP - 1; @SP - src	R IR		70 71	-	-	-	-	-	-	
<b>RCF</b> C - 0			CF	0	-	-	-	-	-	
<b>RET</b> PC - @SP; SP - SP + 2			AF	-	-	-	-	-	-	
<b>RL</b> dst 	R IR		90 91	*	*	*	*	-	-	
<b>RLC</b> dst 	R IR		10 11	*	*	*	*	-	-	
<b>RR</b> dst 	R IR		E0 E1	*	*	*	*	-	-	
<b>RRC</b> dst 	R IR		C0 C1	*	*	*	*	-	-	
<b>SBC</b> dst,src dst - dst - src - C	(Note 1)		3□	*	*	*	*	1	*	
<b>SCF</b> C - 1			DF	1	-	-	-	-	-	
<b>SRA</b> dst 	R IR		D0 D1	*	*	*	0	-	-	



Instruction Summary (Continued)

Instruction and Operation	Addr Mode		Opcode Byte (Hex)	Flags Affected						
	dst	src		C	Z	S	V	D	H	
<b>SRP</b> src RP - src		Im	31	-	-	-	-	-	-	-
<b>SUB</b> dst,src dst - dst - src		(Note 1)	2□	*	*	*	*	1	*	*
<b>SWAP</b> dst		R IR	F0 F1	X	*	*	X	-	-	-

Instruction and Operation	Addr Mode		Opcode Byte (Hex)	Flags Affected						
	dst	src		C	Z	S	V	D	H	
<b>TCM</b> dst,src (NOT dst) AND src		(Note 1)	6□	-	*	*	0	-	-	-
<b>TM</b> dst, src dst AND src		(Note 1)	7□	-	*	*	0	-	-	-
<b>XOR</b> 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 is found in the instruction set table above. The second nibble is expressed symbolically by a □ in this table, and its value is found in the following table to the left of the applicable addressing mode pair.

For example, the opcode of an ADC instruction using the addressing modes r (destination) and Ir (source) is 13.

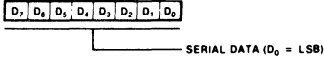
Addr Mode		Lower Opcode Nibble
dst	src	
r	r	2
r	Ir	3
R	R	4
R	IR	5
R	IM	6
IR	IM	7



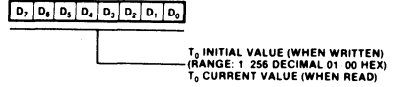
# Z86E21

## Registers

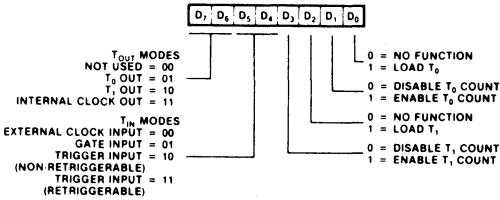
### R240 SIO Serial I/O Register (F0H; Read/Write)



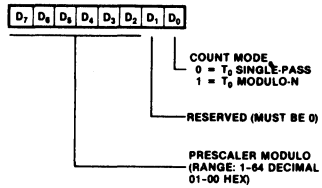
### R244 T0 Counter/Timer 0 Register (F4H; Read/Write)



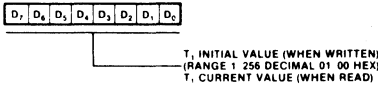
### R241 TMR Timer Mode Register (F1H; Read/Write)



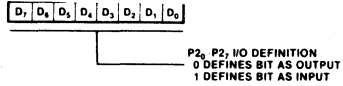
### R245 PRE0 Prescaler 0 Register (F5H; Write Only)



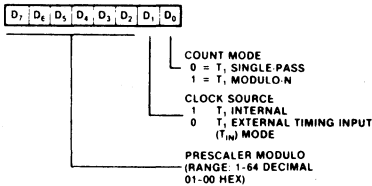
### R242 T1 Counter Timer 1 Register (F2H; Read/Write)



### R246 P2M Port 2 Mode Register (F6H; Write Only)



### R243 PRE1 Prescaler 1 Register (F3H; Write Only)



### R247 P3M Port 3 Mode Register (F7H; Write Only)

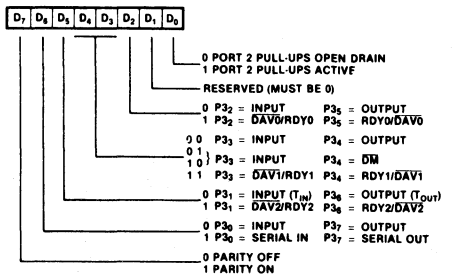


Figure 17. Control Registers

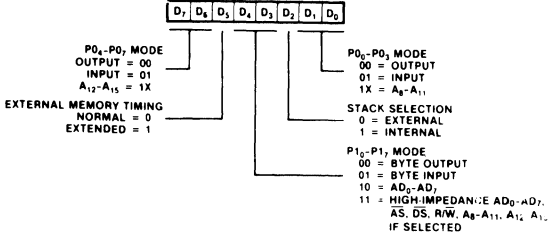
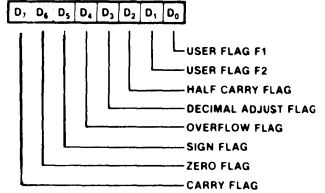
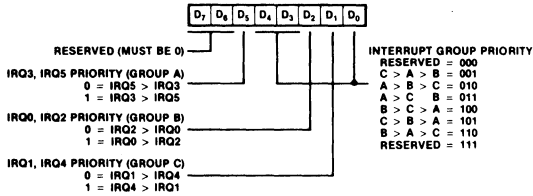
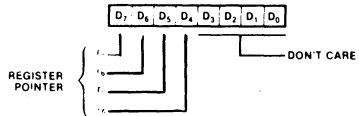
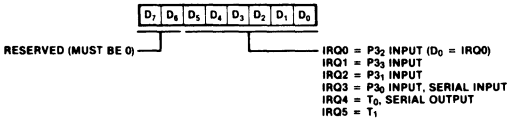
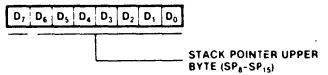
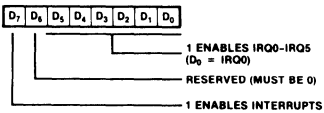
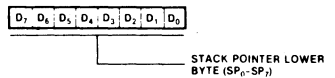
**Registers (Continued)**
**R248 P01M**  
**Port 0 and 1 Mode Register**  
 (F8<sub>H</sub>; Write Only)

**R252 FLAGS**  
**Flag Register**  
 (FC<sub>H</sub>; Read/Write)

**R249 IPR**  
**Interrupt Priority Register**  
 (F9<sub>H</sub>; Write Only)

**R253 RP**  
**Register Pointer**  
 (FD<sub>H</sub>; Read/Write)

**R250 IRQ**  
**Interrupt Request Register**  
 (FA<sub>H</sub>; Read/Write)

**R254 SHP**  
**Stack Pointer**  
 (FE<sub>H</sub>; Read/Write)

**R251 IMR**  
**Interrupt Mask Register**  
 (FB<sub>H</sub>; Read/Write)

**R255 SPL**  
**Stack Pointer**  
 (FF<sub>H</sub>; Read/Write)


Figure 17. Control Registers (Continued)





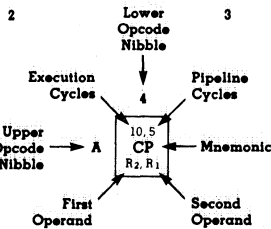
# Z86E21

## Opcode Map

Lower Nibble (Hex)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
Upper Nibble (Hex)	0	6,5 DEC R <sub>1</sub>	6,5 DEC IR <sub>1</sub>	6,5 ADD r <sub>1</sub> , r <sub>2</sub>	6,5 ADD r <sub>1</sub> , IR <sub>2</sub>	10,5 ADD R <sub>2</sub> , R <sub>1</sub>	10,5 ADD R <sub>1</sub> , IM	10,5 ADD R <sub>1</sub> , IM	10,5 ADD IR <sub>1</sub> , IM	6,5 LD r <sub>1</sub> , R <sub>2</sub>	6,5 LD R <sub>2</sub> , R <sub>1</sub>	12/10,5 DJNZ r <sub>1</sub> , RA	12/10,0 JR cc, RA	6,5 LD r <sub>1</sub> , IM	12/10,0 JP cc, DA	6,5 INC r <sub>1</sub>	
	1	6,5 RLC R <sub>1</sub>	6,5 RLC IR <sub>1</sub>	6,5 ADC r <sub>1</sub> , r <sub>2</sub>	6,5 ADC r <sub>1</sub> , IR <sub>2</sub>	10,5 ADC R <sub>2</sub> , R <sub>1</sub>	10,5 ADC R <sub>2</sub> , R <sub>1</sub>	10,5 ADC R <sub>1</sub> , IM	10,5 ADC IR <sub>1</sub> , IM								
	2	6,5 INC R <sub>1</sub>	6,5 INC IR <sub>1</sub>	6,5 SUB r <sub>1</sub> , r <sub>2</sub>	6,5 SUB r <sub>1</sub> , IR <sub>2</sub>	10,5 SUB R <sub>2</sub> , R <sub>1</sub>	10,5 SUB R <sub>2</sub> , R <sub>1</sub>	10,5 SUB R <sub>1</sub> , IM	10,5 SUB IR <sub>1</sub> , IM								
	3	8,0 JP IRR <sub>1</sub>	6,1 SRP IM	6,5 SBC r <sub>1</sub> , r <sub>2</sub>	6,5 SBC r <sub>1</sub> , IR <sub>2</sub>	10,5 SBC R <sub>2</sub> , R <sub>1</sub>	10,5 SBC R <sub>2</sub> , R <sub>1</sub>	10,5 SBC R <sub>1</sub> , IM	10,5 SBC IR <sub>1</sub> , IM								
	4	8,5 DA R <sub>1</sub>	8,5 DA IR <sub>1</sub>	6,5 OR r <sub>1</sub> , r <sub>2</sub>	6,5 OR r <sub>1</sub> , IR <sub>2</sub>	10,5 OR R <sub>2</sub> , R <sub>1</sub>	10,5 OR R <sub>2</sub> , R <sub>1</sub>	10,5 OR R <sub>1</sub> , IM	10,5 OR IR <sub>1</sub> , IM								
	5	10,5 POP R <sub>1</sub>	10,5 POP IR <sub>1</sub>	6,5 AND r <sub>1</sub> , r <sub>2</sub>	6,5 AND r <sub>1</sub> , IR <sub>2</sub>	10,5 AND R <sub>2</sub> , R <sub>1</sub>	10,5 AND R <sub>2</sub> , R <sub>1</sub>	10,5 AND R <sub>1</sub> , IM	10,5 AND IR <sub>1</sub> , IM								
	6	6,5 COM R <sub>1</sub>	6,5 COM IR <sub>1</sub>	6,5 TCM r <sub>1</sub> , r <sub>2</sub>	6,5 TCM r <sub>1</sub> , IR <sub>2</sub>	10,5 TCM R <sub>2</sub> , R <sub>1</sub>	10,5 TCM R <sub>2</sub> , R <sub>1</sub>	10,5 TCM R <sub>1</sub> , IM	10,5 TCM IR <sub>1</sub> , IM								
	7	10/12,1 PUSH R <sub>2</sub>	12/14,1 PUSH IR <sub>2</sub>	6,5 TM r <sub>1</sub> , r <sub>2</sub>	6,5 TM r <sub>1</sub> , IR <sub>2</sub>	10,5 TM R <sub>2</sub> , R <sub>1</sub>	10,5 TM R <sub>2</sub> , R <sub>1</sub>	10,5 TM R <sub>1</sub> , IM	10,5 TM IR <sub>1</sub> , IM								
	8	10,5 DECW RR <sub>1</sub>	10,5 DECW IR <sub>1</sub>	12,0 LDE r <sub>1</sub> , IR <sub>2</sub>	18,0 LDEI IR <sub>1</sub> , IR <sub>2</sub>												6,1 DI
	9	6,5 RL R <sub>1</sub>	6,5 RL IR <sub>1</sub>	12,0 LDE r <sub>2</sub> , IRR <sub>1</sub>	18,0 LDEI IR <sub>2</sub> , IRR <sub>1</sub>												6,1 EI
	A	10,5 INCW RR <sub>1</sub>	10,5 INCW IR <sub>1</sub>	6,5 CP r <sub>1</sub> , r <sub>2</sub>	6,5 CP r <sub>1</sub> , IR <sub>2</sub>	10,5 CP R <sub>2</sub> , R <sub>1</sub>	10,5 CP R <sub>2</sub> , R <sub>1</sub>	10,5 CP R <sub>1</sub> , IM	10,5 CP IR <sub>1</sub> , IM								14,0 RET
	B	6,5 CLR R <sub>1</sub>	6,5 CLR IR <sub>1</sub>	6,5 XOR r <sub>1</sub> , r <sub>2</sub>	6,5 XOR r <sub>1</sub> , IR <sub>2</sub>	10,5 XOR R <sub>2</sub> , R <sub>1</sub>	10,5 XOR R <sub>2</sub> , R <sub>1</sub>	10,5 XOR R <sub>1</sub> , IM	10,5 XOR IR <sub>1</sub> , IM								16,0 IRET
	C	6,5 RRC R <sub>1</sub>	6,5 RRC IR <sub>1</sub>	12,0 LDC r <sub>1</sub> , IR <sub>2</sub>	18,0 LDCI IR <sub>1</sub> , IR <sub>2</sub>												10,5 LD r <sub>1</sub> , x, R <sub>2</sub>
	D	6,5 SRA R <sub>1</sub>	6,5 SRA IR <sub>1</sub>	12,0 LDC r <sub>2</sub> , IRR <sub>1</sub>	18,0 LDCI IR <sub>2</sub> , IRR <sub>1</sub>	20,0 CALL* IRR <sub>1</sub>		20,0 CALL DA	10,5 LD r <sub>2</sub> , x, R <sub>1</sub>								6,5 SCF
	E	6,5 RR R <sub>1</sub>	6,5 RR IR <sub>1</sub>		6,5 LD r <sub>1</sub> , IR <sub>2</sub>	10,5 LD R <sub>2</sub> , R <sub>1</sub>	10,5 LD R <sub>2</sub> , R <sub>1</sub>	10,5 LD R <sub>1</sub> , IM	10,5 LD IR <sub>1</sub> , IM								6,5 CCF
	F	8,5 SWAP R <sub>1</sub>	8,5 SWAP IR <sub>1</sub>		6,5 LD IR <sub>1</sub> , r <sub>2</sub>			10,5 LD R <sub>2</sub> , IR <sub>1</sub>									6,0 NOP

Bytes per Instruction



### Legend:

R = 8-Bit Address  
r = 4-Bit Address  
R<sub>1</sub> or r<sub>1</sub> = Dest Address  
R<sub>2</sub> or r<sub>2</sub> = Src Address

### Sequence:

Opcode, First Operand, Second Operand

**Note:** The blank areas are not defined.

\*2-byte instruction; fetch cycle appears as a 3-byte instruction



### Absolute Maximum Ratings

Voltages on all pins with respect to GND ..... -0.3 V to +7.0 V  
Operating Ambient Temperature ..... 0°C to +70°C  
Storage Temperature ..... -65°C to +150°C

Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

### Test Conditions

The characteristics below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND. Positive current flows into the reference pin. Standard conditions are as follows:

- +4.75 V ≤ V<sub>CC</sub> ≤ +5.25 V
- GND = 0 V
- 0°C ≤ T<sub>A</sub> ≤ +70°C

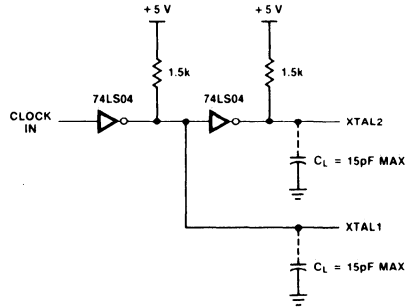
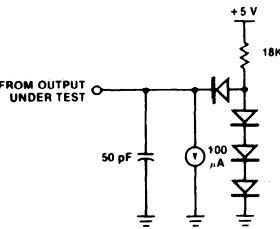
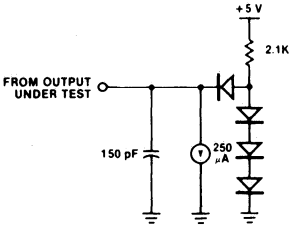


Figure 18. Test Load 1

Figure 19. Test Load 2

Figure 20. External Clock Interface Circuit

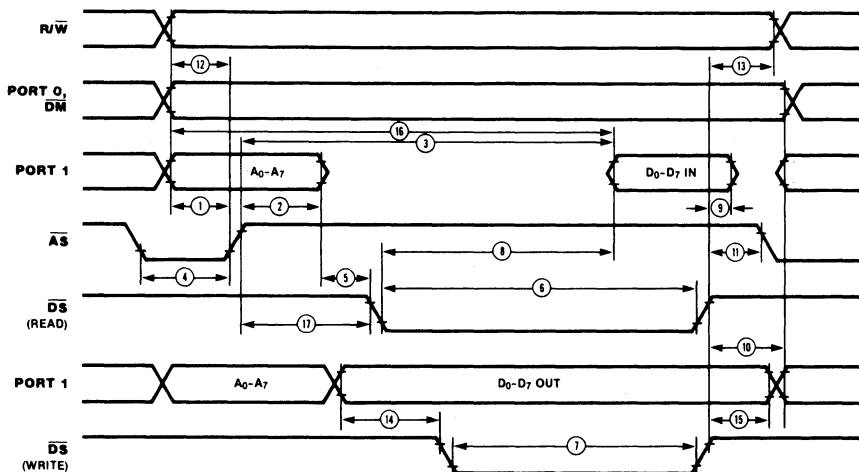
**Z86E21****DC Characteristics**

Symbol	Parameter	Min	Max	Unit	Condition
V <sub>CH</sub>	Clock Input High Voltage	3.8	V <sub>CC</sub>	V	Driven by External Clock Generator
V <sub>CL</sub>	Clock Input Low Voltage	-0.3	0.8	V	Driven by External Clock Generator
V <sub>IH</sub>	Input High Voltage	2.0	V <sub>CC</sub>	V	
V <sub>IL</sub>	Input Low Voltage	-0.3	0.8	V	
V <sub>RH</sub>	Reset Input High Voltage	3.8	V <sub>CC</sub>	V	
V <sub>RL</sub>	Reset Input Low Voltage	-0.3	0.8	V	
V <sub>OH</sub>	Output High Voltage	2.4		V	I <sub>OH</sub> = -250 $\mu$ A
V <sub>OL</sub>	Output Low Voltage		0.4	V	I <sub>OL</sub> = +2.0 mA
I <sub>IL</sub>	Input Leakage	-10	10	$\mu$ A	0 V $\leq$ V <sub>IN</sub> $\leq$ +5.25 V
I <sub>OL</sub>	Output Leakage	-10	10	$\mu$ A	0 V $\leq$ V <sub>IN</sub> $\leq$ +5.25 V
I <sub>IR</sub>	Reset Input Current		-50	$\mu$ A	V <sub>CC</sub> = +5.25 V, V <sub>RL</sub> = 0 V
I <sub>CC</sub>	V <sub>CC</sub> Supply Current		120	mA	
I <sub>MM</sub>	V <sub>MM</sub> Supply Current		10	mA	Power Down Mode
V <sub>MM</sub>	Backup Supply Voltage	3	V <sub>CC</sub>	V	Power Down

**External I/O or Memory Read and Write Timing**

No	Symbol	Parameter	Z86E21		Z86E21A		Notes*†
			Min	Max	Min	Max	
1	TdA(AS)	Address Valid to $\overline{AS}$ $\uparrow$ Delay	50		35		1,2,3
2	TdAS(A)	$\overline{AS}$ $\uparrow$ to Address Float Delay	70		45		1,2,3
3	TdAS(DR)	$\overline{AS}$ $\uparrow$ to Read Data Required Valid		360		220	1,2,3,4
4	TwAS	$\overline{AS}$ Low Width	80		55		1,2,3
5	TdAz(DS)	Address Float to $\overline{DS}$ $\downarrow$	0		0		1
6	TwDSR	$\overline{DS}$ (Read) Low Width	250		185		1,2,3,4
7	TwDSW	$\overline{DS}$ (Write) Low Width	160		110		1,2,3,4
8	TdDSR(DR)	$\overline{DS}$ $\downarrow$ to Read Data Required Valid		200		130	1,2,3,4
9	ThDR(DS)	Read Data to $\overline{DS}$ $\uparrow$ Hold Time	0		0		1
10	TdDS(A)	$\overline{DS}$ $\uparrow$ to Address Active Delay	70		45		1,2,3
11	TdDS(AS)	$\overline{DS}$ $\uparrow$ to $\overline{AS}$ $\downarrow$ Delay	70		55		1,2,3
12	TdR/W(AS)	R/W Valid to $\overline{AS}$ $\uparrow$ Delay	50		30		1,2,3
13	TdDS(R/W)	$\overline{DS}$ $\uparrow$ to R/W Not Valid	60		35		1,2,3
14	TdDW(DSW)	Write Data Valid to $\overline{DS}$ (Write) $\downarrow$ Delay	50		35		1,2,3
15	TdDS(DW)	$\overline{DS}$ $\uparrow$ to Write Data Not Valid Delay	70		45		1,2,3
16	TdA(DR)	Address Valid to Read Data Required Valid		410		255	1,2,3,4
17	TdAS(DS)	$\overline{AS}$ $\uparrow$ to $\overline{DS}$ $\downarrow$ Delay	80		55		1,2,3

- NOTES:
1. Test Load 1
  2. Timing numbers given are for minimum T<sub>PC</sub>.
  3. Also see clock cycle time dependent characteristics table.
  4. When using extended memory timing add 2 T<sub>PC</sub>.
  5. All timing references use 2.0 V for a logic "1" and 0.8 V for a logic "0".
  - \* All units in nanoseconds (ns).
  - † Timings are preliminary and subject to change.


**Figure 21. External I/O or Memory Read/Write**



# Z86E21

## Additional Timing Table

No	Symbol	Paramter	Z86E21		Z86E21A		Notes*†
			Min	Max	Min	Max	
1	TpC	Input Clock Period	125	1000	83	1000	1
2	TrC, TfC	Clock Input Rise And Fall Times		25		15	1
3	TwC	Input Clock Width	37		26		1
4	TwTinL	Timer Input Low Width	100		70		2
5	TwTinH	Timer Input High Width	3TpC		3TpC		2
6	TpTin	Timer Input Period	8TpC		8TpC		2
7	TrTin, TfTin	Timer Input Rise And Fall Times		100		100	2
8a	TwIL	Interrupt Request Input Low Time	100		70		2,3
8b	TwIH	Interrupt Request Input Low Time	3TpC		3TpC		2,3
9	TwIH	Interrupt Request Input High Time	3TpC		3TpC		2,3

**NOTES:**

1. Clock timing references uses 3.8 V for a logic "1" and 0.8 V for a logic "0".

2. Timing reference uses 2.0 V for a logic "1" and 0.8 V for a logic "0".

3. Interrupt request via Port 3 (P3<sub>1</sub>-P3<sub>3</sub>).

4. Interrupt request via Port 3 (P3<sub>0</sub>).

\* Units in nanoseconds (ns).

† Timings are preliminary and subject to change.

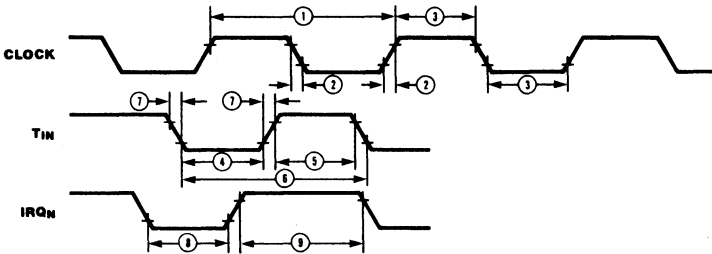


Figure 22. Additional Timing

### Handshake Timing

No	Symbol	Paramter	Z86E21		Z86E21A		Notes*†
			Min	Max	Min	Max	
1	TsDI(DAV)	Data In Setup Time	0		0		
2	ThDI(DAV)	Data In Hold Time	230		160		
3	TwDAV	Data Available Width	175		120		
4	TdDAVih(RDY)	$\overline{DAV}$ ↓ Input to RDY ↓ Delay		175		120	1,2
5	TdDAVOf(RDY)	$\overline{DAV}$ ↓ Output to RDY ↓ Delay	0		0		1,3
6	TdDAVlr(RDY)	$\overline{DAV}$ ↑ Input to RDY ↑ Delay		175		120	1,2
7	TdDAVOr(RDY)	$\overline{DAV}$ ↑ Output to RDY ↑ Delay	0		0		1,3
8	TdDO(DAV)	Data Out to $\overline{DAV}$ ↓ Delay	50		30		1
9	TdRDY(DAV)	Rdy ↓ Input to DAV ↑ Delay	0	200	0	140	1

NOTES:

1. Test Load 1
2. Input handshake
3. Output handshake

4. All timing referenes use 2.0 V for a logic "1" and 0.8 V for a logic "0".

\* Units in nanoseconds (ns).

† Timings are preliminary and subject to change.

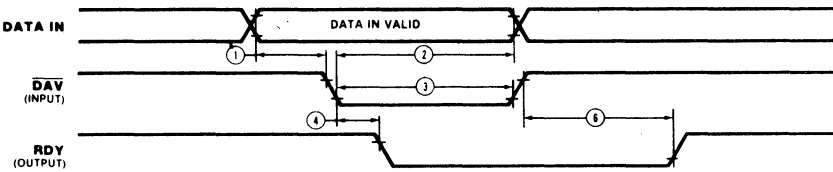


Figure 23a. Input Handshake

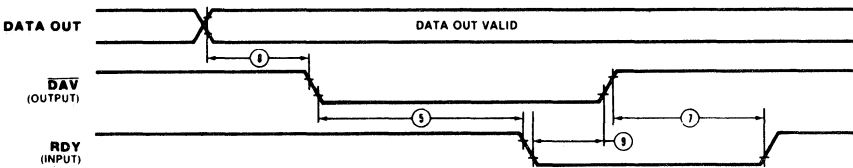


Figure 23b. Output Handshake



# Z86E21

## Clock-Cycle-Time-Dependent Characteristics

Number	Symbol	Z86E21 Equation	Z86E21A Equation
1	TdA(AS)	TpC-75	TpC-50
2	TdAS(A)	TpC-55	TpC-40
3	TdAS(DR)	4TpC-140*	4TpC-110*
4	TwAS	TpC-45	TpC-30
6	TwDSR	3TpC-125*	3TpC-65*
7	TwDSW	2TpC-90*	2TpC-55*
8	TdDSR(DR)	3TpC-175*	3TpC-120*
10	Td(DS)A	TpC-55	TpC-40
11	TdDS(AS)	TpC-55	TpC-30
12	TdR/W(AS)	TpC-75	TpC-55
13	TdDS(R/W)	TpC-65	TpC-50
14	TdDW(DSW)	TpC-75	TpC-50
15	TdDS(DW)	TpC-55	TpC-40
16	TdA(DR)	5TpC-215*	5TpC-160*
17	TdAS(DS)	TpC-45	TpC-30

\* Add 2TpC when using extended memory timing.

## Ordering Information

Type	Package	Temp.	Clock	Description
Z86E21 D1	Ceramic	0/ +70°C	8 MHz	8K EPROM Microcomputer
Z86E21 D6	Ceramic	-40/ +85°C		
Z86E21A D1	Ceramic	0/ +70°C	12 MHz	
Z86E21A D6	Ceramic	-40/ +85°C		

---

## **Packages**

---

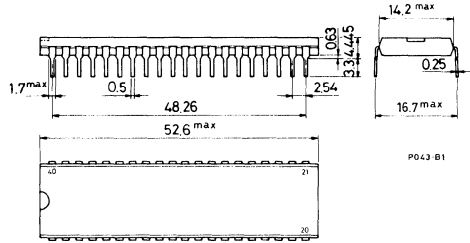
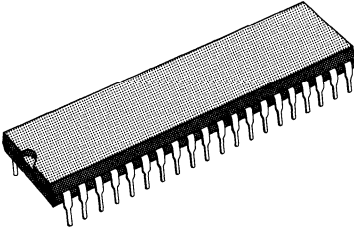
---



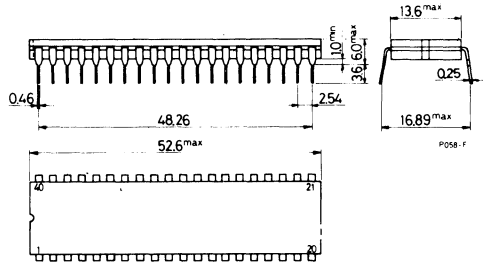
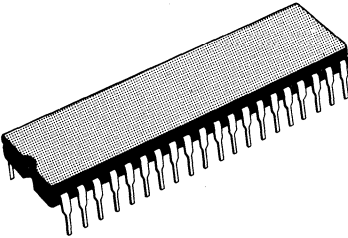


# Packages

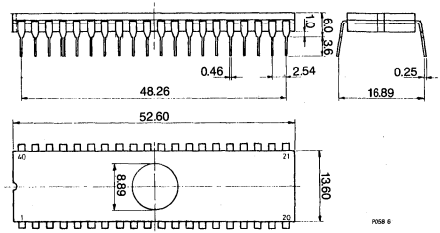
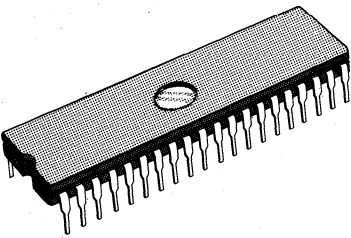
## 40 - Lead Plastic DIP



## 40 - Lead Ceramic DIP (Frit-Seal)

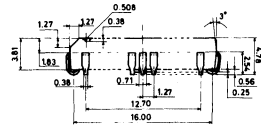
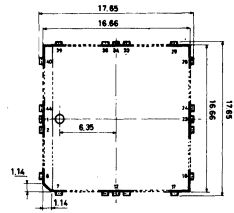
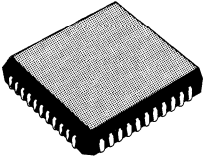


## 40 - Lead Ceramic DIP (Glass lens)



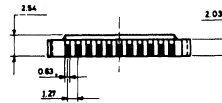
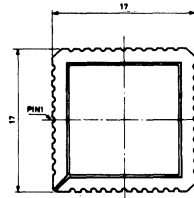
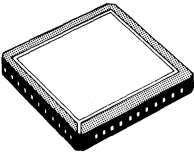
# Packages

## 44 - Leaded Plastic Chip Carrier



REF 6-1

## 44 - Leadless Ceramic Chip Carrier



REF 6

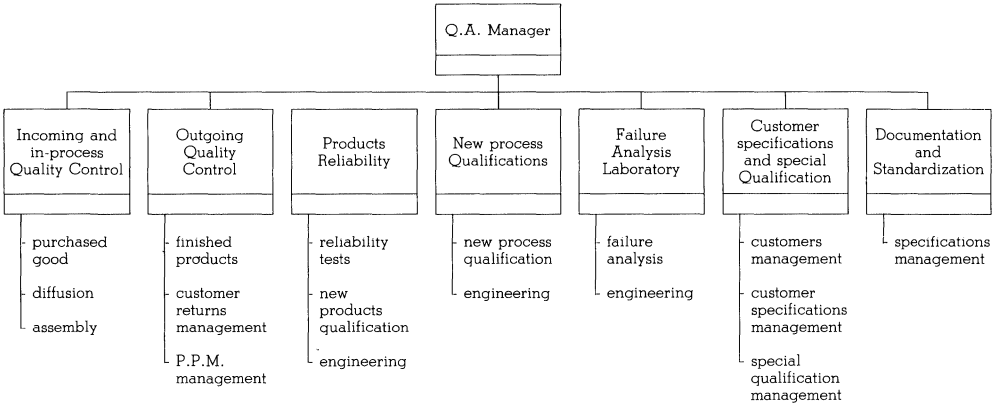
---

## **Reliability Informations**

---



## Quality Assurance Organization Chart



## Handling

SGS has chosen a no-compromise strategy in MOS ESD protection. From wafer level to the shipping of finished units, we fully guarantee each work station and processing of the parts. This is achieved through total adoption of shielding and grounding media. Final shipment is in antistatic sticks vacuum sealed, in a conductive shielding bag.

The supplier's best commitment is useless if the end user does not provide the same level of protection and care in application. Here are the basic static control protection rules:

A - Handle all components in a static-safe work area.

B - Transport all components in static shielding containers.

To comply with the rules the following procedures must be set up.

1 - Static control wrist strap (from a qualified source) should be worn and connected properly.

2 - Each work surface must be protected with a conductive mat, properly grounded.

3 - Extensive use of conductive floor mats.

4 - Static control shoe straps should be worn insulative footwear, such as those with crepe or thick rubber soles is worn.

5 - Ionized air blowers are a necessary part of the protective system, to neutralize static

charges on conductive items.

6 - Use only the grounded tip variety of soldering iron.

7 - Single components, tubes, printed circuit cards should always be contained in static shielding bags; keep our parts in the original bags up to the very last possible point in your operation line.

8 - If bigger containers (tote box) are used for in-plant transport of devices on PC boards they must be electrically conductive, like the carbon loaded types.

9 - All tools, persons, testing machines, which could contact device leads must be conductive and grounded.

10 - Avoid the usage of high dielectric materials (like polystyrene) for subassembly construction, storage, transportation.

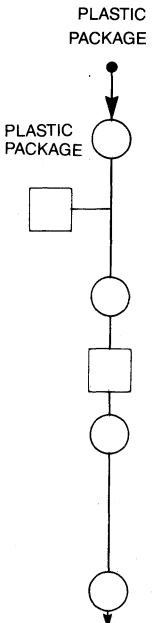
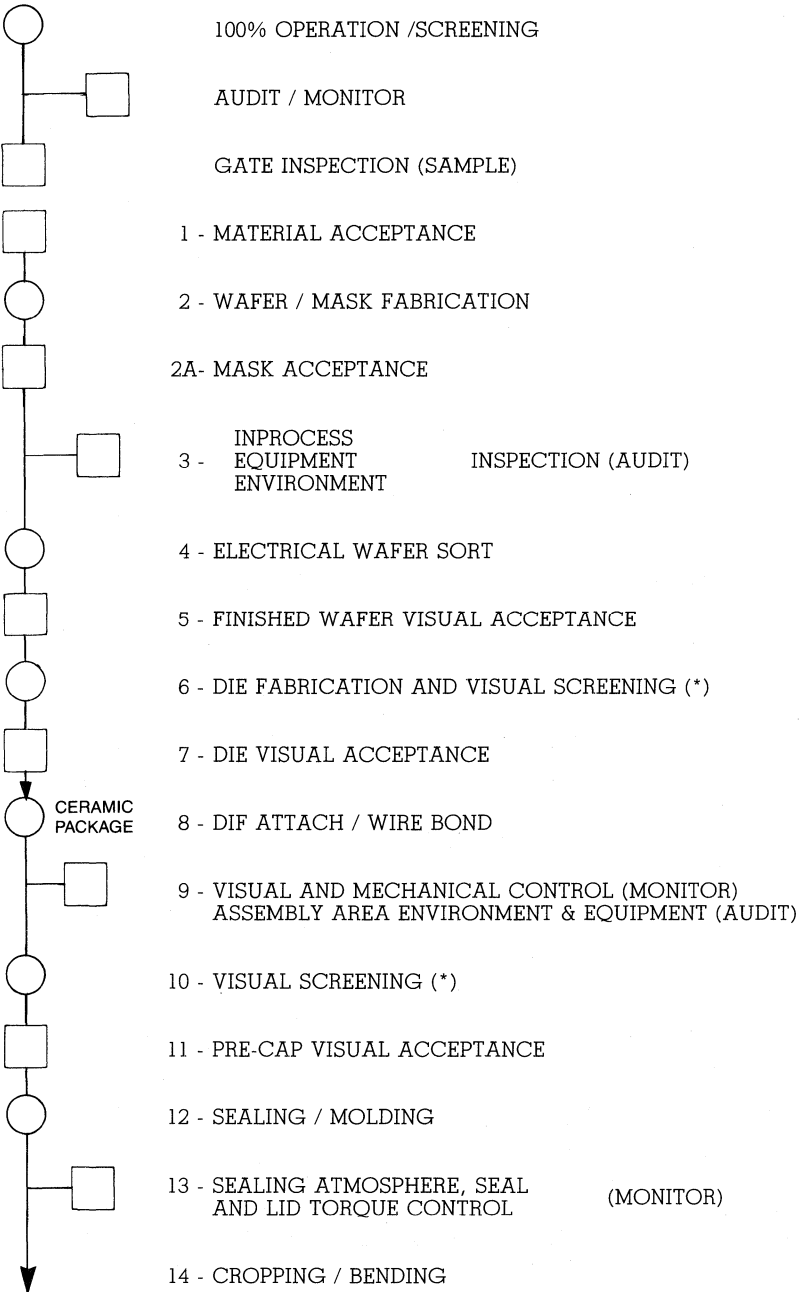
11 - Follow a proper power supply sequence in testing and application. Supply voltage should be applied before and removed after input signals; insertion removal from sockets should be done with no power applied.

12 - Filtration, noise suppression, slow voltage surges should be guaranteed on the supply lines.

13 - Any open (floating) input pin is a potential hazard to your circuit: ground or short them to  $V_{DD}$  whenever possible.

# MOS /CMOS STD Process Flow-Chart

KEY:



# MOS /CMOS STD Process Flow-Chart (Continued)

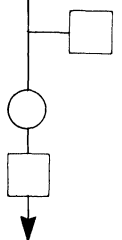
CERAMIC PACKAGE



- 15 - VISUAL CONTROL
- 16 - TINNING ACCEPTANCE
- 17 - TINNING PLATING ACCEPTANCE

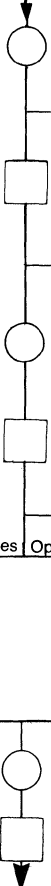
(\*) Omitted when intrinsic quality meets the specified quality level.

PLASTIC PACKAGE



# MOS/CMOS STD & Optional Process Flow-Chart

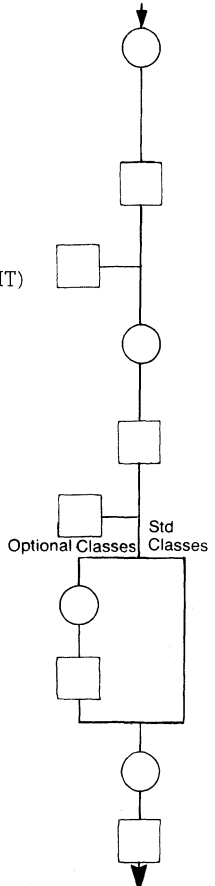
CERAMIC PACKAGE



- 18 - LEAD TRIMMING
- 19 - INTERNAL WATER-VAPOR CONTENT (MONITOR)
- 20 - RAW-LINE ACCEPTANCE
- 21 - RELIABILITY TESTS STD CLASS (REAL TIME & GROUP B C D TESTS) (MONITOR-AUDIT)
- 22 - ELECTRICAL TESTING AND MARKING
- 23 - VISUAL AND ELECTRICAL FINAL ACCEPTANCE (SEE PAGE 191)
- 23A- MECHANICAL (MONITOR)
- 24 - SCREENING OPTIONS (TO BE AGREED WITH CUSTOMER)
- 25 - VISUAL, ELECTRICAL AND RELIABILITY TESTS FINAL ACCEPTANCE (OPTIONAL CLASSES)
- 26 - PACKING
- 27 - PACKING AND DOCUMENTATION ACCEPTANCE

Std Classes | Optional Classes

PLASTIC PACKAGE



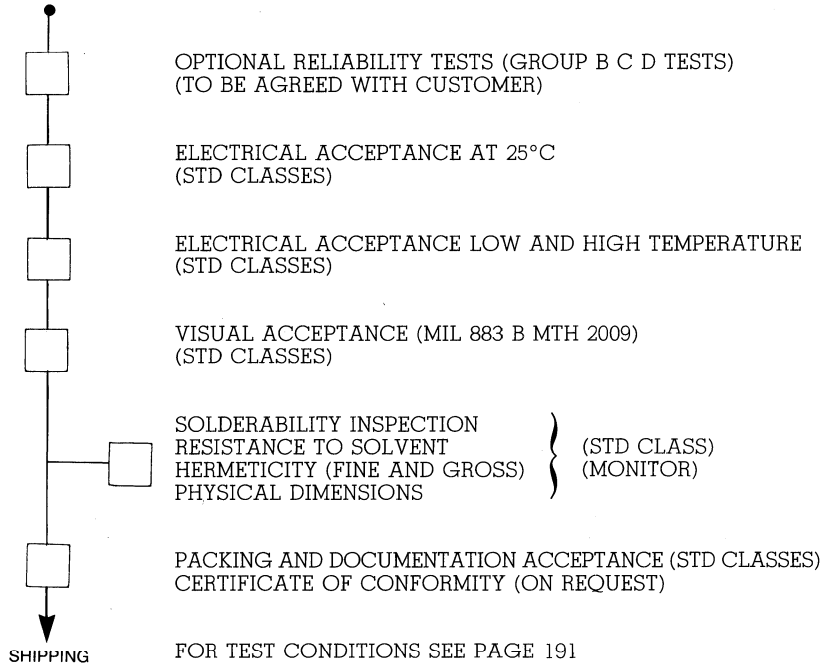
Optional Classes | Std Classes

SHIPPING

SHIPPING



# MOS /CMOS STD Visual and Electrical Final Acceptance



## Production Quality Tests Description and Screenings

Process Steps	Tests	Descriptions
1	MATERIAL ACCEPTANCE	WAFER - MASKS - WIRES - FRAMES - PHOTORESIST - CHEMICALS - PREFORMS - RESIN - BONDING TOOLS - PLASTIC TUBES - GLAZED CERAMIC PARTS MULTILAYER CERAMIC PACK - GOLD PLATED CAPS
2 A	MASK ACCEPTANCE	VISUAL DIMENSIONS
3	WAFER FABRICATION INSPECTION  INPROCESS    EQUIPMENT   ENVIRONMENT	PHOSPHORUS CONTENT IN P. VAPOX  GLASSIVATION INTEGRITY (MIL-STD 883C MTH 2021)  S.E.M. INSPECTION (MIL-STD 883C MTH 2018)  VISUAL AND DIMENSIONAL INSPECTION (MIL STD 883C MTH 2010 COND. B)  CONTAMINATION  D.I. WATER RESISTIVITY  BACTERIOLOGICAL ANALYSIS OF THE D.I. WATER  DUST COUNT HUMIDITY TEMPERATURE
5	FINISHED WAFERS VISUAL ACCEPTANCE	MIL-STD 883C MTH 2010 COND. B
7	DIE VISUAL ACCEPTANCE	MIL-STD 883C MTH 2010 COND. B
9	DIE ATTACH CONTROL  BONDING CONTROL  ASSEMBLY AREA ENVIRONMENT CONTROL  EQUIPMENT	MIL-STD 883C MTH 2010 COND. B (INTERNAL VISUAL) AND MTH 2019 (DIE SHEAR STRENGHT)  MIL-STD 883C MTH 2010 COND. B (INTERNAL VISUAL) AND MTH 2011 COND. D (BOND STRENGTH)  DUST COUNT HUMIDITY TEMPERATURE  D.I. WATER RESISTIVITY
11	PRECAP ACCEPTANCE	MIL-STD 883C MTH 2010 COND B (INTERNAL VISUAL)
12	SEALING  MOLDING AND STABILIZATION BAKE	VACUUM PREBAKE: 2 HRS AT 220°C HIGH TEMP. FINAL SEAL: 8 MINUTES ABOUT AT 450°C  STABILIZATION BAKE: 8 HRS AT 175°C
13	SEALING ATMOSPHERE CONTROL  SEAL CONTROL	MOISTURE CONTENT: < 200 PPM  FINE LEAK MIL-STD 883C MTH 1014 COND. A1 HELIUM LEAK DETECTOR AFTER PRESSURIZATION IN HE FOR 2 HRS AT 4 ATM LIMIT: $5 \times 10^{-8}$ CC/S FOR ICV* < 0.4 CC $2 \times 10^{-7}$ CC/S FOR ICV > 0.4 CC * (ICV = INTERNAL CAVITY VOLUME)

## Production Quality Tests Description and Screenings (Continued)

Process Steps	Tests	Descriptions
	LID TORQUE CONTROL	GROSS LEAK MIL-STD 883C MTH 1014 COND. C (FLUOROCARBON GROSS LEAK) 5 TORR VACUUM FOR 1 HR EXCEPT FOR ICV > 0.1 CC FOLLOWED BY PRESSURIZATION IN MINERAL OIL AT: 4 ATM FOR 2 HRS FOR ICV < 0.1 CC OR 4 ATM FOR 10 H FOR ICV > 0.1 CC AND SUBSEQUENT IMMERSION IN MINERAL OIL AT $T_{amb} = 125^{\circ}\text{C}$  CERAMIC PACKAGES ONLY MIL-STD 883C MTH 2024
15	CROPPING / BENDING CONTROL	MIL STD 883C MTH 2009
17	TIMING ACCEPTANCE	SOLDERABILITY MIL STD 883C MTH 2003 $T_{amb} = 245 + 5^{\circ}\text{C}$ FOR 5 + 0.5 SEC. WITH PRECONDITIONING FOR 1 HR ABOVE BOILING DIST. WATER (I.E.C. MTH AVAIL. ON REQUEST)
19	INTERNAL WATER VAPOR CONTENT CONTROL	DEW POINT MTH MIL-STD 883C MTH 1018 PROCEDURE 3-5000 PPM MAX (DEW POINT TEMPER. LESS THAN $-15^{\circ}\text{C}$ )
20	RAW LINE ACCEPTANCE	EXTERNAL VISUAL MIL-STD 883C MTH 2009  LID TORQUE TEST: AS PER STEP 13  CONSTANT ACCELERATION MIL-STD 883C MTH 2001 COND. E (30,0006) Y1 ORIENTATION ONLY*  SEAL CONTROL: AS PER STEP 13
21	RELIABILITY TEST (REAL TIME AND GROUP B C D TESTS)	MONITOR ON STD CLASSES, GATE ON REQUEST (TEST AND SAMPLE SIZE TO BE AGREED WITH CUSTOMER) AUDIT ON ALL FACTORIES
23	VISUAL AND ELECTRICAL FINAL ACCEPTANCE STD CLASS	VISUAL AND MECHANICAL INSPECTION  CUMULATIVE ELECTRICAL AND INOPERATIVE MECHANICAL FAILURES
23 A	MECHANICAL	SOLDERABILITY INSEPCION RESISTANCE TO SOLVENT HERMETICITY (FINE AND GROSS) PHYSICAL DIMENSIONS
27	PACKING AND DOCUMENTATION ACC.ANCE	VISUAL

\* 20,000 G FOR PACKAGES WITH CAVITY PERIMETER OF 5 CM OF MORE AND/OR WITH A MASS OF 5 GRAMS OR MORE

## Reliability Group B Tests Description

Performed every week or every 3 months on raw line material and/or finished products

Test	MIL-STD-883C	
	Method	Condition
<b>SUBGROUP 1</b> (1) PHYSICAL DIMENSIONS	2016	MAJOR DIMENSIONS ACCORDING TO DATA SHEET
<b>SUBGROUP 2</b> (1) RESISTANCE TO SOLVENT	2015	1 MINUTE IMMERSION IN SOLVENT SOLUTION FOLLOWED BY 10 STROKES WITH A SOFT BRUSH (THE PROCEDURE SHALL BE REPEATED 3 TIMES) SOLVENT SOLUTION 2.1a ONLY FOR MOULD. PACK.
<b>SUBGROUP 3</b> (1) SOLDERABILITY	2003	SOLDERING TEMPERATURE $245 \pm 5^{\circ}\text{C}$ for $5 \pm 0.5$ SEC. WITH PRECONDITIONING FOR 1 HR ABOVE BOILING DISTILLED WATER AND 5 TO 10 SEC. IN ROSIN BASE FLUX
<b>SUBGROUP 4</b> STEADY STATE AND OPERATING LIFE TEST  END-POINT ELECTRICAL PARAMETERS	1005	1000 HRS AT $T_{amb} = 125^{\circ}\text{C}$ ; ACCORDING TO DETAIL SPECIFICATION  AS SPECIFIED IN THE APPLICABLE DEVICE SPEC. MEASUREMENTS AT 0, 168, 500, AND 1000 HRS
<b>SUBGROUP 5</b> (HERMETIC PACKAGES ONLY) TEMPERATURE CYCLING  CONSTANT ACCELERATION  SEAL (1) - FINE - GROSS  END-POINT ELECTRICAL PARAMETERS	1010  2001  1014	TEST CONDITION C (10 CYCLES $T_{amb} = -65^{\circ}\text{C}$ TO $+150^{\circ}\text{C}$ ); 10 MINUTES AT EXTREME TEMPERATURES; 5 MINUTES TRANSFER TIME  TEST CONDITION E (30000 G) Y1 ORIENTATION ONLY (2)  TEST CONDITION A1 (SEE STEP 13 PAGE 191) TEST CONDITION C  AS SPECIFIED IN THE APPLICABLE DEVICE SPEC.
<b>SUBGROUP 6</b> (1) (MOULDED PACKAGES ONLY)  PRESSURE POT  END-POINT ELECTRICAL PARAMETERS		$T_{amb} = 121^{\circ}\text{C}$ , 2 ATM, 96 HRS  AS SPECIFIED IN THE APPLICABLE DEVICE SPEC.

1) Performed weekly on finished products.

2) 20000 g for package with cavity perimeter of 5 cm or more and/or with a mass of 5 grams or more

## Reliability Group C Tests Description

Performed every 6 months on raw line material

Test	MIL-STD-883C	
	Method	Condition
<b>SUBGROUP 1</b> LEAD INTEGRITY	2004	TEST CONDITION B2 (LEAD FATIGUE) - WIRE LEADS: A FORCE OF $0.229 \pm 0.014$ KG FOR THREE $90 \pm 5^\circ$ ARCS ON EACH LEAD BENDING CYCLE: 2 TO 5 SEC. - DUAL-IN-LINE MOULDED PACKAGE: THREE LEADS SHALL BE BENT, 3 TIMES, SIMULTANEOUSLY FOR AT LEAST $15^\circ$ PERMANENT BEND, RETURNING THEN TO THE ORIGINAL POSITION.
SEAL (HERMETIC PACKAGES ONLY) - FINE - GROSS	1014	TEST CONDITION A1 TEST CONDITION C (SEE STEP 13 PAGE 191)
<b>SUBGROUP 2</b> THERMAL SHOCK (2) (HERMETIC PACKAGES ONLY)	1011	TEST CONDITION B; 15 CYCLES ( $T_{amb} = -55^\circ\text{C}$ TO $+125^\circ\text{C}$ ) 5 MIN. AT EXTREME TEMPERATURES TRANSFER TIME $\leq 10$ SEC.
TEMPERATURE CYCLING (2)	1010	TEST CONDITION C; 100 CYCLES ( $T_{amb} = -65^\circ\text{C}$ TO $+150^\circ\text{C}$ ) 10 MIN AT EXTREME TEMPERATURES; TRANSFER TIME 5 MINUTES
MOISTURE RESISTANCE (HERMETIC PACKAGES ONLY)	1004	10 CYCLES OF 24 HRS; $T_{amb} = 25^\circ\text{C}$ TO $65^\circ\text{C}$ RH = 80% TO 100% ONE 3 HRS CYCLE AT $T_{amb} = -10^\circ\text{C}$
SEAL (HERMETIC PACKAGES ONLY) - FINE - GROSS	1014	TEST CONDITION A1 TEST CONDITION C (SEE STEP 13 PAGE 191)
VISUAL EXAMINATION		PER VISUAL CRITERIA OF METHOD 1004 AND 1010
END-POINT ELECTRICAL PARAMETERS		AS SPECIFIED IN THE APPLICABLE DEVICE SPEC.
<b>SUBGROUP 3</b> (HERMETIC PACKAGE ONLY)		
MECHANICAL SHOCK	2002	TEST CONDITION B; 1500 G - 0.5 MSEC. - 5 BLOWS IN EACH OF THE 6 ORIENTATIONS - NOT OPERATING
VIBRATION, VARIABLE FREQUENCY	2007	TEST CONDITION A; 20 G - 3 ORIENTATIONS $F=20$ TO 2000 CPS; FOUR 4 MINUTES CYCLES, 48 MINUTES TOTAL - NOT OPERATING
CONSTANT ACCELERATION (1) SEAL - FINE - GROSS	2001 1014	TEST CONDITION E (30000 G), Y1 ORIENTATION ONLY TEST CONDITION A1 TEST CONDITION C (SEE STEP 13 PAGE 191)
VISUAL EXAMINATION		PER VISUAL CRITERIA OF METHOD 1011 OR 1010
END-POINT ELECTRICAL PARAMETERS		AS SPECIFIED IN THE APPLICABLE DEVICE SPEC.

1) 20000 grams for packages with cavity perimeter of 5 cm or more and/or mass of 5 grams or more.

2) Performed weekly.

## Reliability Group C Tests Description (Continued)

Performed every 6 months on raw line material

Test	MIL-STD-883C	
	Method	Condition
<b>SUBGROUP 4</b> SALT ATMOSPHERE  SEAL (HERMETIC PACKAGE ONLY) - FINE - GROSS  VISUAL EXAMINATION	1009	TEST CONDITION A; 10 TO 50 G OF NaCl PER SQUARE METER PER DAY FOR 24 HRS $T_{amb} = -35^{\circ}\text{C}$  TEST CONDITION A1 TEST CONDITION C (SEE STEP 13 PAGE 191)  PER VISUAL CRITERIA OF METHOD 1009
<b>SUBGROUP 5</b> (MOULDED PACKAGES ONLY)  HUMIDITY TEST (4)  END-POINT ELECTRICAL PARAMETERS	CECC 9000	85°C/85% RH WITH BIAS, $t = 1000$ HRS ACCORDING TO DETAIL SPECIFICATION  AS SPECIFIED IN THE APPLICABLE DEVICE SPEC. MEASUREMENTS AT 0, 168, 500 AND 1000 HRS
<b>SUBGROUP 6</b> (HERMETIC PACKAGE ONLY)  INTERNAL WATER-VAPOR CONTENT	1018	DEW POINT METHOD-PROCEDURE 3 (5000 PPM MAX)
<b>SUBGROUP 7</b>  LID TORQUE (3) (HERMETIC PACKAGES ONLY)	2024	(SEE STEP 13 PAG. 191)
<b>SUBGROUP 8</b>  ELECTROSTATIC DISCHARGE SENSITIVITY  END-POINT ELECTRICAL PARAMETERS	3015	R = 1.5K $\Omega$ C = 100pF V = ACCORDING TO DETAIL SPECIFICATION  AS SPECIFIED IN THE APPLICABLE DEVICE SPECIFICATION

3) Lid torque test shall apply only to packages which use a glass-frit-seal to lead frame, lead or package body (i.e. wherever frit seal establishes hermeticity or package integrity).

4) Performed monthly.



---

# **Development Products**

---

---





## General Purpose Foundation Module for MCU Emulation

- Combined with dedicated personality packages it allows complete hardware and software development and debugging for MCUs
- Emulation memory: 8K bytes of static RAM
- Breakpoint memory: 8K bits of static RAM allowing up to 8196 breakpoints on addresses and/or data
- Real time trace memory: 1K x 25 bits of static RAM allowing up to 1024 events to be recorded during program execution
- Personality module interface: a parallel interface with an external personality POD is provided
- Standard double - euro card format fully compatible with SGS UX8-22 Development System and with the Z8 Emulation and Development package for IBM compatible Personal Computers

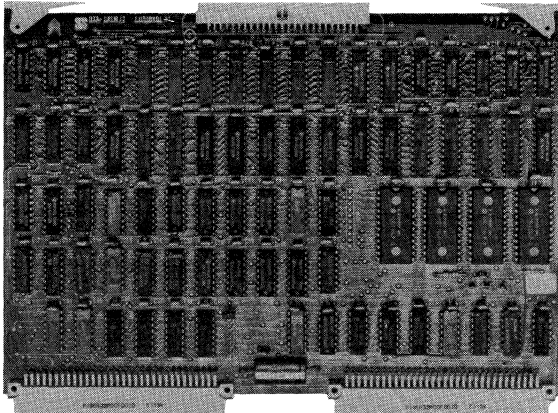
### General Description

The UX8-GPFM/2 is a general purpose foundation module for MCUs emulation. It is designed to be plugged into a UX8-22 Development System and to interface with an externally connected POD.

UX8-GPFM/2 gives the user the basic and general hardware tools in order to carry out complete debugging and emulation for MCUs.

In order to set up a complete development and debugging tool for a specific single-chip microcomputer it's necessary to integrate the UX8-GPFM/2 with a specific personality

package. The personality package is basically composed of a software package for cross-assembling/linking/ debugging and of an external POD for the hardware interface with the target microcomputer (see UX8-EDPZ8 data sheet for Z8 personality package general features). The basic features of the UX8-GPFM/2 include: emulation memory, breakpoint memory, real time trace memory, control logic, interface with GAMMA-BUS and a parallel interface for an external POD.

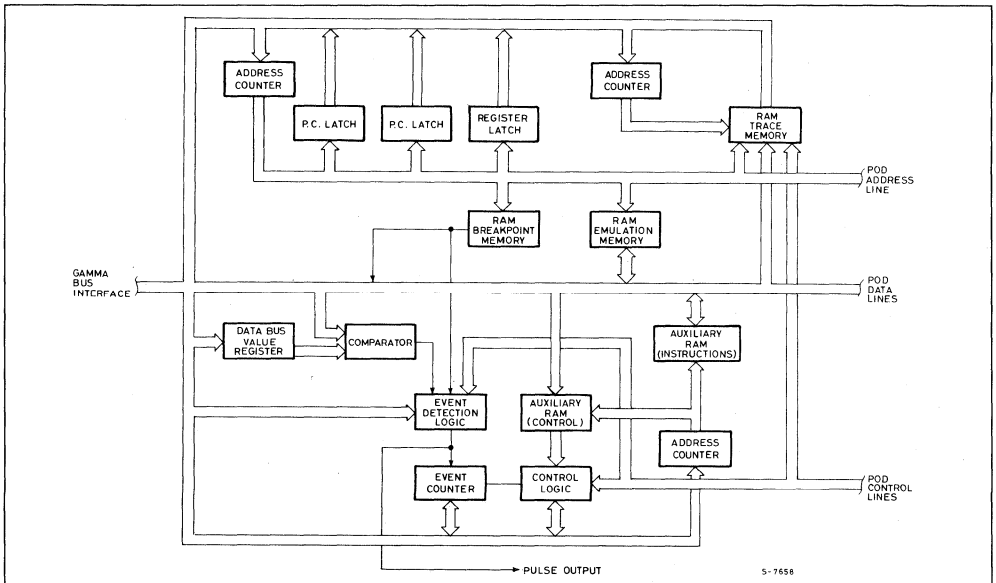




## Hardware Features

- **EMULATION MEMORY:** the emulation memory is implemented with 8K bytes of static RAM. It is a dual port access memory connected from one side to the GAMMA-BUS and to the other to the interface circuitry with the external POD.
- **BREAKPOINT MEMORY:** the breakpoint memory is composed of 8K bits of static RAM. The memory is addressed in parallel with the emulation memory and allows some addresses to be defined as "significant" for breakpoint purposes. Up to 8192 breakpoints can be contemporarily active and breakpoint detection on a specific location can be specified in read/write/fetch/op code/generic or preset value mode.
- **BREAKPOINT COUNTER:** an 8 bit event counter is also provided to allow easy implementation of single/multi-step program execution.
- **REAL TIME TRACE MEMORY:** 1K x 25 bits of static RAM memory is used to record, during program execution by the emulated microprocessor, up to 1024 events, including address, data read/written, type of cycle (fetch/read/write) and interrupts.
- **GAMMA-BUS INTERFACE:** the interface with UX8-22 resources takes place by means of 32 contiguous I/O addresses, therefore the emulator doesn't share or request memory resources at UX8-22 level.
- **POD INTERFACE:** 13 address lines, 8 data lines, 7 control lines and power supply lines are provided for interfacing with an external POD. Physical connection is made by means of a 50 wire flat cable.

Figure 1 - Block Diagram



---

**Ordering Information**

<b>Type</b>	<b>Description</b>
UX8-GPFM/2	General Purpose Foundation Module for Single Chip Emulation

DOCUMENTATION: The complete technical manual with product description, schematics drawings and component layout is normally supplied with each product. Technical manuals are also available separately for purchase.



## Z8 Emulation and Development Package

- Designed to run on SGS UX8-22 Development System and to interface with UX8-GPFM/2 board (General Purpose Foundation Module) for single chip microprocessor emulation
- Composed of:
  - Z8 personality POD
  - Software development and debugging package
- Z8 personality POD: hardware interface between GPFM/2 board (installed inside UX8-22 Development System) and Z8 based target
- Software development and debugging package running on UX8-22 Development System under CP/M\* 2.2, for:
  - cross assembling
  - cross linking
  - debugging

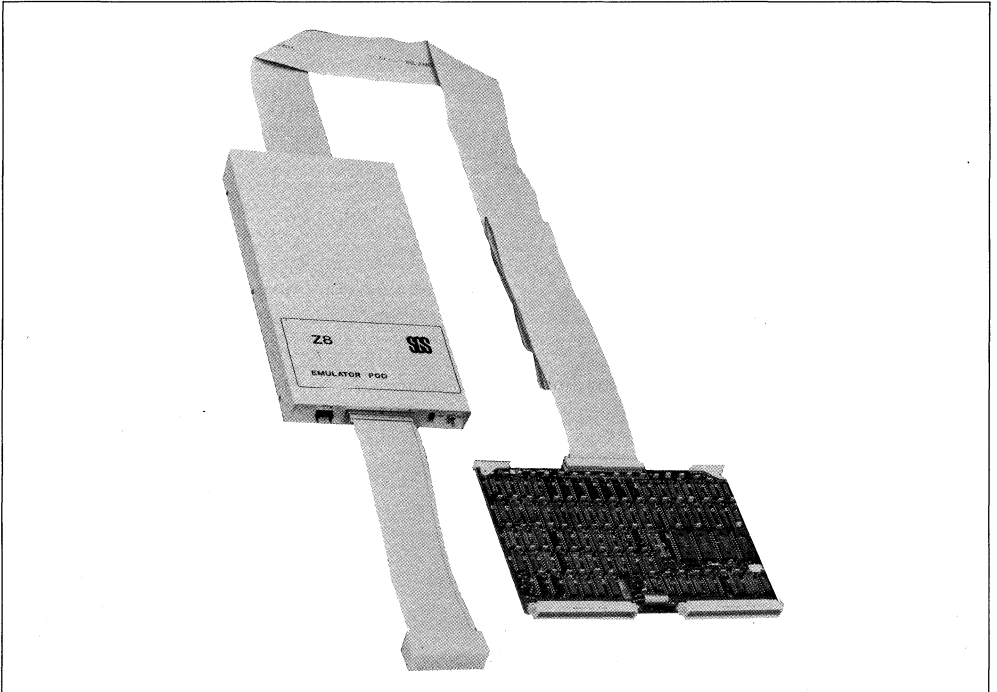
### General Description

UX8-EDPZ8/2 is an advanced hardware and software emulation and development package for the Z8 single chip microcomputer.

The package has been especially designed for the SGS UX8-22 Development System

and is composed of 2 separate items:

- Z8 personality POD
- Software development and debugging package





---

## General Description (Continued)

The EDPZ8/2 package requires as a prerequisite at UX8-22 level the UX8-GPFM/2 Module (General Purpose Foundation Module for MCU Emulation).

The Z8 personality POD is the hardware interface between UX8-GPFM/2 and a Z8 based user target.

The software development and debugging package is formed by a cross assembler, linker loader and debugger for Z8

microcomputer; it runs on the UX8-22 under CP/M 2.2 operating system.

Using the EDPZ8/2 package the designer can develop software programs and carry out complete hardware and software debugging of Z8 based systems using the UX8-22 Development System. A simple but effective command set is provided in order to access, read and modify registers, memory and ports. Breakpoint and real time trace capabilities are also provided.

---

## Main Features

**Z8 Personality POD.** Z8 Personality POD contains the Z8 development microcomputer and the interface circuits both with the GPFM/2 (the general purpose emulation board) and with the Z8 based target system. Connections take place at both sides by means of flat cables.

The POD employs the development

microcomputer Z8612, adequate for the emulation of Z8 2K ROM (Z8601) and Z8 4K ROM (Z8611).

The POD is able both to generate the clock for the development microcomputer or to use the signal of an external clock (switch selectable option).

---

## Software Development and Debugging Package

The software development and debugging package is basically composed of MAKZ8, a program development package and Z8DBG, a debugging package.

**MAKZ8 - Assembler.** MAKZ8 is a powerful disk-based editor/assembler system for Z8 single chip microprocessor program development.

It includes also all the features necessary for the creation/modification of assembly languages programs for Z8 and file handling, such as:

Executive for handling all input/output operation.  
Text Editor for creation and modification of

source program files.

Assembler: a relocating macroassembler with cross reference generator and linking loader.

**Z8DBG - Debugger.** Z8DBG is a software tool allowing complete debugging of previously assembled programs.

Its main function are:

- Control of the interface and data transfer among Z8 personality POD, the general purpose emulator GPFM/2 and system resources.
- handling of debugging session and execution of debugging commands required by the user.

**Software Development and Debugging Package** (Continued)

**Z8DBG Command List**

BAs e x l o l d	Change/select base of numbers
Break [(options)]	Display/set breakpoint
CB [(addr) l (from) (to)]	Clear breakpoints
Cmp [(ad1) (ad2) [(count)]]	Compare memory
DL [(addr) l (from) (to)]	Display memory in listing format
DM [(addr) l (from) (to)]	Display/change program memory
DR [(n) l (reg)]	Display/change registers
FM (from) (to) (pat)	Fill memory with pattern
Fr (from) (to) (pat)	Fill registers with pattern
Go [(from)]	Start user program execution
Help	Gives this help
HWTEST	Execute Diagnostic test
Load (file) [ / ]	Load memory from a file
Move (from) (to) [(count)]	Move memory block
Next [(steps)]	Single/multi step mode
Quit	Abandon the program
REset	Reset CPU
SAve (file) [(from) [(to)]]	Save memory into a file
SB (addr) l (from) (to)	Set address breakpoints
SEarch (from) (pat)	Search pattern in memory
SET [(options)]	Set/Display system options
SR [(n) l (reg) [(value)]]	Set register
Trace [(count)]	Display traced execution
Use (file)	Execute command file
VE	Enter Execute screen mode
VM [(addr)]	Enter Memory screen mode
VR	Enter Register screen mode
WR	Display working register set

**Ordering Information**

Type	Description
UX8-EDPZ8/2	Z8 Emulation and Development Package

DOCUMENTATION: The complete technical manual with product description, schematics drawings and component layout is normally supplied with each product. Technical manuals are also available separately for purchase.







## Z8 Emulation and Development Package for IBM Compatible Personal Computers

- Designed to run on Personal Computers IBM-PC parallel bus compatible.
- Composed of:
  - GPFM/2 General Purpose Foundation Module for single chip emulator
  - Z8 personality POD
  - Bus interface board between GAMMA-BUS and IBM-BUS
  - Power Supplier (110/220V - 50/60Hz)
- Software Development and Debugging package running under MS/DOS operating system for:
  - cross assembling
  - cross linking
  - debugging
- Hardware features are:
  - able to emulate Z8 2K and/or 4K ROM versions
  - 4 Kbytes of emulation memory
  - 4 Kbytes of break point memory
  - 1 Kbyte of time trace memory
  - IBM parallel bus interface
  - 8 MHz internal clock or up to 12 MHz switch selectable external clock
  - pulse output (TTL compatible) enabled every time the PC reaches a Break Point address

### General Description

The EMU-Z8PC is an advanced hardware and software emulation package for the Z8 microcomputer family.

The package has been especially designed by SGS to run and interface to IBM compatible personal computers and is composed of four main parts:

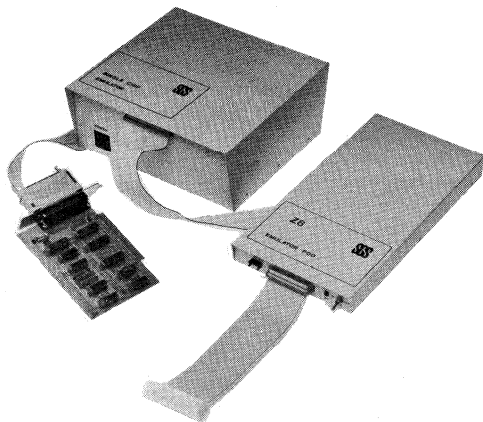
- the single chip emulator box which contains

- the back panel board and the main power supplier;

- the Z8 personality POD;

- the interface board;

- the Z8 Assembler, Linker and Debugger software, delivered on a 5 1/4" single side, single density, Floppy Disk.





## Hardware Features

**GPFM/2.** The General Purpose Foundation Module is a double Eurocard format board, essentially made up of a random-access memory (RAM) with a twin access port: from one side (port), it provides the program memory required by the development microcomputer, while, on the other, it opens access to the common memory, both in read and write, to the IBM PC host computer. For more detailed explanations, reference should be made to the previous chapter (see page 199).

**Z8 Personality POD.** This is a module which contains the Z8 development microcomputer and the interface circuits both with the GPFM/2, contained in the emulation box, and with the Z8-based target system, the socket to be inserted into the user board in place of the emulated microcomputer. Connections take place on both sides by means of flat cables. The POD employs the development microcomputer Z8612 adequate for the emulation of Z8 2K ROM (Z8601) and Z8 4K ROM (Z8611). The POD is able to generate the clock for the development microcomputer or to use the signal of an external clock (switch selectable option).

**Power Supply.** A main power supplier for 220V/50Hz and 110V/60Hz is built inside the single chip emulator box to supply the GPFM/2 and the personality POD.

**Interface Board.** This allows electrical interface between the GPFM/2 board and the IBM compatible personal computer bus, converting the GAMMA-BUS running on GPFM/2 to an IBM parallel bus. This board is inserted in the internal personal computer slot with an addressing range dedicated by

the IBM personal computer to prototype card communication.

- Minimum System Configuration Required
- 64K bytes system memory
  - 1 disk driver
  - alphanumeric and monochromatic video

**Software Development and Debugging Package.** The software development and debugging package is basically composed of XMAC-Z8, a program development package and Z8DBG, a debugging package.

**XMAC-Z8 - Assembler.** XMAC-Z8 is a powerful disk-based editor/assembler system for Z8 single chip microcomputer program development. It includes also all the features necessary for the creation/modification of assembly language programs for Z8 and file handling, such as:

- Executive for handling all input/output operation;
- Text Editor for creation and modification of source program files;
- Assembler a relocating macroassembler with cross reference generator and linking loader.

**Z8DBG - Debugger.** Z8DBG is a software too, running under the MS/DOS operating system, that allows complete debugging of previously assembled programs. Its main functions are:

- control of the interface and data transfer among Z8 personality POD, the general purpose emulator GPFM/2 and system resources.
- handling of debugging session and execution of debugging commands required by the user.

**Z8DBG Command List**

BAsE x l o l d	Change/select base of numbers
Break [(options)]	Display/set breakpoint
CB [<addr> l [<from> <to>]	Clear breakpoints
Cmp [<ad1> <ad2> [(count)]	Compare memory
DL [<addr> l [<from> <to>]	Display memory in listing format
DM [<addr> l [<from> <to>]	Display/change program memory
DR [<n> l <reg>]	Display/change registers
FM <from> <to> <pat>	Fill memory with pattern
Fr <from> <to> <pat>	Fill registers with pattern
Go [<from>]	Start user program execution
Help	Gives this help
HWTEST	Execute Diagnostic test
Load <file> [ / ]	Load memory from a file
Move <from> <to> [(count)]	Move memory block
Next [(steps)]	Single/multi step mode
Quit	Abandon the program
REset	Reset CPU
SAve <file> [(from)][<to> ]	Save memory into a file
SB <addr> l [<from> <to>]	Set address breakpoints
SEarch <from> <pat>	Search pattern in memory
SET [(options)]	Set/Display system options
SR [(n) l <reg> [<value> ]	Set register
Trace [(count)]	Display traced execution
Use <file>	Execute command file
VE	Enter Execute screen mode
VM [<addr>]	Enter Memory screen mode
VR	Enter Register screen mode
WR	Display working register set

**Ordering Information**

Type	Description
EMU-Z8PC	Z8 Emulation and Development Package for IBM Compatible Personal Computers

DOCUMENTATION: The complete technical manual (User, Hardware and Installation) with product description, schematics drawings and component layout is normally supplied with each product. Technical manuals are also available separately for purchase.



---

**Technical and  
Application Notes**

---



# Double Layer P-Vapox and Si<sub>3</sub>N<sub>4</sub> Glass Passivation

A. Panchieri Q.A. MOS DIVISION

**Newly developed passivation process for NMOS/HS-CMOS devices gives improved protection to die encapsulated in plastic packages.**

## Process Description

The process consists of a two layer film of P-Vapox (phosphorus doped silicon oxide) and Si<sub>3</sub>N<sub>4</sub> (silicon nitride), obtained by two different masking and etching steps to avoid defects caused by lack of dielectric integrity.

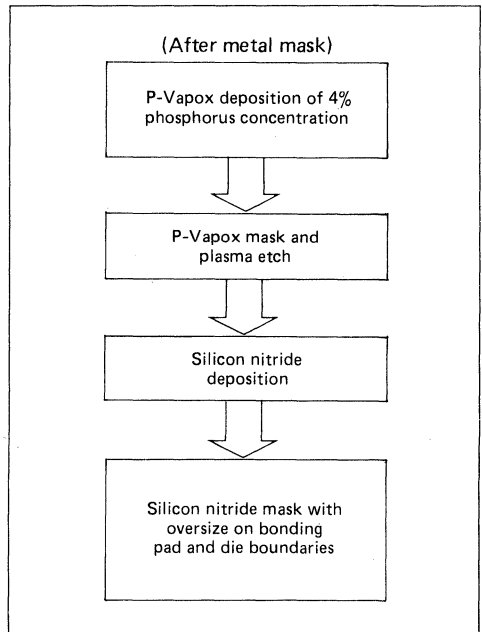
The process gives good metal step coverage together with PECVD (Plasma Enhanced Chemical Vapox Deposition) to avoid cracking near metal edge and possible hillocks defects.

The double layer enables us, by means of an appropriate oversize either at the boundaries of the die side or at the bonding pad side, to ensure full sealing of the underlying P-Vapox layer.

This prevents the layer from being exposed to moisture coming from the package. Thus the probability of metal corrosion on the bonding pad due to phosphoric acid is drastically reduced.

As a result the die is provided with a very good humidity immunity.

## Process Flow





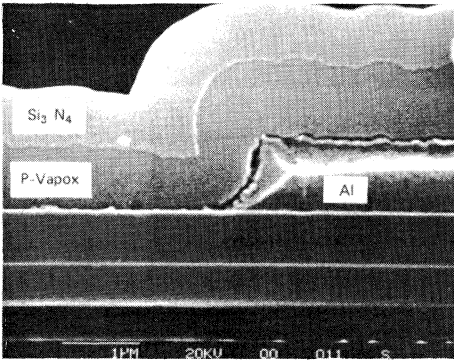


Fig. 1 - Typical Microsection of Device with Nitride Passivation.

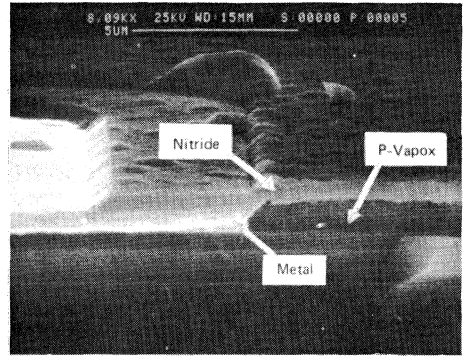


Fig. 3 - Section Along the Pad

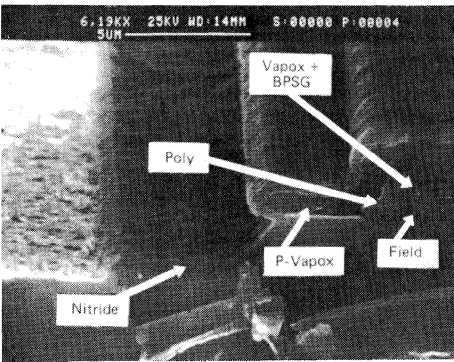


Fig. 2 - Section Along the Scribing Line.

### Reliability Results

The reliability performance in moist ambient was evaluated using both 85°C/85% RH/BIAS and 121°C Pressure Pot test:

Different products in different plastic packages were tested.

To give an idea of reliability performances obtained on products with the new passivation process, we have set out the process qualification test results in the following table:

	CMOS Logics			$\mu$ Processors		Memories	
	Hours	Sample	Fail	Sample	Fail	Sample	Fail
Static/Dynamic	1000	1355	0	385	0	270	0
Life Test	2000	915	0	385	0	270	0
T <sub>A</sub> = 125°C	3000	600	0	—	—	—	—
V <sub>CC</sub> = std.	4000	600	0	—	—	—	—
Temp Humidity	1000	740	0	240	0	360	0
BIAS (85°C/85% RH)	2000	510	0	240	0	360	0
V <sub>CC</sub> = std.	3000	260	1 funct.	—	—	—	—
	4000	259	0	—	—	—	—
Pressure Pot	96	965	0	540	0	250	0
T <sub>A</sub> = 121°C-2 atm	144	775	1 funct.	540	0	250	0
	192	320	0	300	0	120	0
	288	320	0	300	1 funct.	120	0

# PMZ8: Z8681 in Single Board Computer Application

The PMZ8 Single Board Computer is a very small system, based on the Z8681 MCU, which is one of the ROMless version of the Z8 single chip microcomputer family.

The Z8681 offers all the outstanding features of the Z8 family architecture except the on-chip program ROM. It provides up to 16 output address lines, thus permitting an address space of up to 64K bytes of data on program memory. Available address space (up to 128K bytes) can be doubled by programming bit 4 of Port 3 (P34) to act as a data memory select output (DM). The two states of DM, together with the 16 address outputs, can define separate data and memory address spaces of up to 64K

bytes each.

The available address space is mapped into three devices, with up to 2K bytes each: one EPROM and one RAM/EPROM for the program space and one data space RAM. The EPROM contains the monitor program (the first 2K bytes), which allows the user to change the content of memory and registers, load or save the memory from and to a Host System and to run programs. The second 2K bytes of address space in the data space are further decoded to four strobe lines (WREX0/1 and RDEX0/1), which permits the addition of two 8-bit input and two 8-bit output ports externally.

---

## Operation

**Start Up.** After power-on or reset, the ports, the timer and the stack pointer are initialized. No interrupt is enabled. After that, the memory device in the IC 5 socket is tested. If this device is an EPROM, the program therein is started from location 00CH; otherwise, the monitor software is entered. This auto start feature permits starting a user program without any keyboard entry.

In the user program, the location 000CH must not contain 0FFH (NOP).

**Terminal.** The PMZ8 can be operate with any ASCII terminal, using the following set-up:

```
9600 BAUD
8 DATA BITS
1 or 2 STOP BITS
NO PARITY
XON-XOFF PROTOCOL
```

No hardware handshake (CTS/RTS, etc.) is provided.

**Monitor.** The monitor software provides the following commands:

```
? ..... Help (display this)
c[* ,/] [addr]..... Change memory content
d[* ,/] [addr]..... Dump memory content
g[addr]..... Go and execute program
l..... Load from host
r[reg] ..... Dump or change register
s< * ,/> addr, leng . Save to host
t..... Test (terminal)
```

Upper and lower case letters are treaded as the same. All numbers are in hexadecimal.

"" and "/" indicate the selected memory space, where "" is the program space and "/" is the data space.

---

## Operation (Continued)

**c: Change Memory Content.** The type of memory (\* or /) is the last one used if not specified. The address of the location to be changed must be entered. The type of memory, the current address and the current content will be displayed, and the user may or may not enter a new value for that location. The input will be accepted after the following keys:

- RETURN The new value (if any) will be stored, and the next location will be displayed.
- ^ The new value (if any) will be stored, and the previous location will be displayed.
- The new value (if any) will be stored, and the current location will be displayed again.
- q The new value (if any) will be stored, and the c-Command will be terminated.

**d: Dump Memory Content.** The type of memory (\* or /) is the last one used if not specified. If the address is not specified, then it is the address of the location which would be displayed next if the previous command was also a d-Command. The memory content is displayed in HEX, and in ASCII, if possible.

**g: Go and Execute.** If the address is not specified, 00CH is used.

**l: Load from Host.** The type of memory, the address and the length will be sent by the host.

**r: Dump or Change Register Contents.** If no register number is specified, then a complete dump of all registers takes place. Otherwise, the register number, the abbreviation (if any) and the current content is displayed. The software doesn't take account of write only or read only registers. The input of new values is the same as described above for the c-Command.

**s: Save Memory Content to Host.** The type of memory, the address and the length must be specified. The content of the specified memory block is sent in pure binary, and the software running on the host is responsible for converting the data to HEX. No blocks longer than 2048 bytes may be transferred immediately.

**Monitor Entry Points.** There is a jump table at 800H, which allows the use of some of the monitor routines, as you can see opposite.

The user program may be simply terminated by return (AFH). In this case, the monitor software is re-entered.

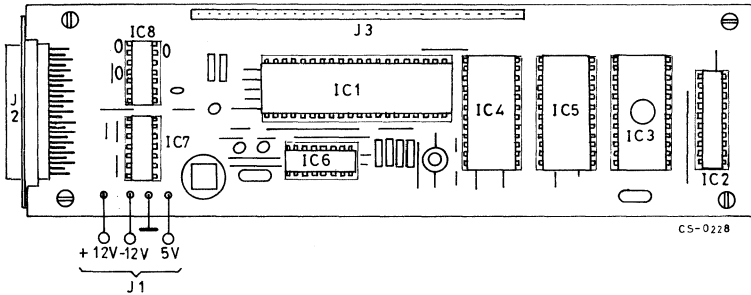
If the configuration of the microcomputer is changed by the user's software, it must be restored before returning to the monitor.

The monitor uses only the registers from 4 to 0FH and some locations below 80H for the stack.

**Operation** (Continued)

Address	Routine	Description
800	JP MON	Restart or start monitor
803	JP PUTC	Sends character in R7 to the terminal
806	JP GETC	Waits for character from terminal, returns it in R7
809	JP UNGETC	Ungets one character until next getc
80C	JP GETHEX	Reads HEX number from terminal until first non-HEX character is entered, returns it in RR4
80F	JP TOUP	Converts character in R7 to upper case, and stores it in R6
812	JP PRTHEx	Prints number in R6 in HEX on the terminal
815	JP PRTCLRF	Prints CR and LF on the terminal
818	JP PRTSTR	Prints string on the terminal: start address of string in RR4; bit 7 high in last character
81B	JP SPC1	Prints one space on the terminal
81E	JP SPC2	Prints two spaces on the terminal
821	JP SPC4	Prints four spaces on their terminal
F0C		This is reset entry point.

**Jump Table**



**Figure 1 - P.C. Board and Components Layout**

# Operation (Continued)

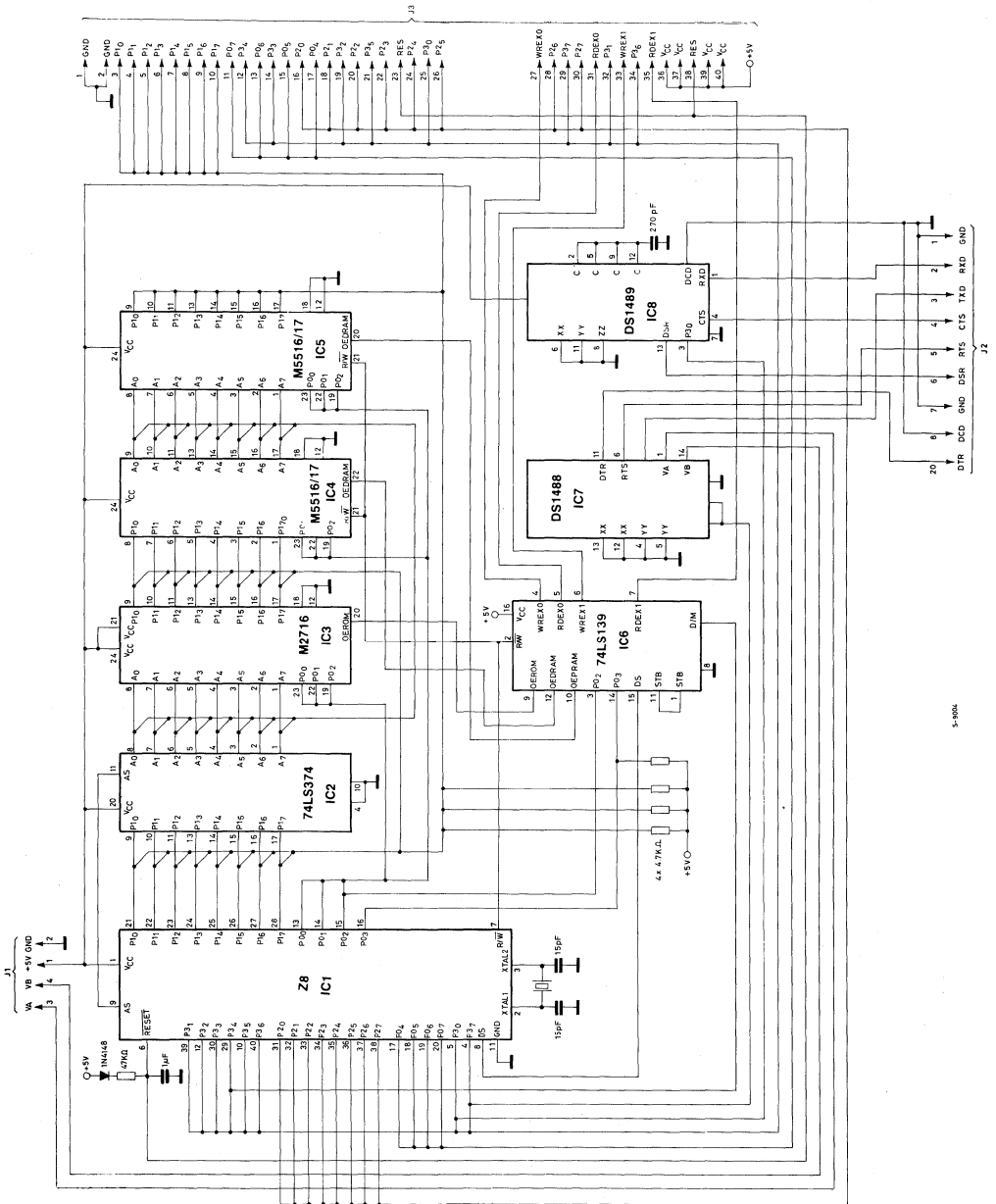


Figure 2 - Functional Description

# Single Board Computer Using Z8671

## Introduction

The Z8671 single chip microcomputer is actually a Z8601 with 2K bytes of Basic/Debug interpreter masked in the chip's resident ROM space. Like the Z8601, the Z8671 has 2 counter timers with prescalers, synchronous serial communication interface with programmable baud rates. Addressing capability includes 62K bytes of external program memory, 144 bytes of on-chip register and 62K bytes of external data memory.

The Z8671 single board computer is an evaluation board which via a RS-232 link to a terminal allows the user to access the

features of the Z8601 using BASIC commands.

The evaluation board with 2K bytes of RAM on board allows immediate or program mode execution of BASIC commands.

Applications of this board ranges from microcontrollers to a tiny BASIC microcomputer. This Technical Bulletin introduces the hardware and software aspects of the Z8671 single board computer to the user. To use this board to its' full potential, the *Z8 Technical Manual (O.C. DAZ8TM/2)* and the *Z8671 Basic/Debug Software Manual (DAZ8671SOF/1)* should be read.

---

## Installation

The Z8671 board needs +5V and ground to run all the components on the board except the 1488 EIA line driver. The 1488 needs +12V and -12V in addition to the +5V and ground.

Power supply connection is via molex connectors of +5V, GND, +12V and -12V.

Terminal connection is via molex connector too, using only 3 pins, serial in,

serial out and GND.

Any terminal with a standard RS232 interface can be used. Baud rate is set in EPROM location FFFDH.

With power connected to the board and the terminal connected to it, the reset button resets the Z8671 and the prompt character appears (":"). The board is then ready for a Basic command when the ":" appears.

---

## Features

- 2K bytes Basic/Debug interpreter in the internal ROM
- 2K bytes of user RAM
- 2K bytes of user-programmable Eprom
- Full-duplex serial operation with programmable baud rates
- RS232 interface
- 8 bit counter/timer with associated 6 bit prescalers
- 124 general-purpose registers internal to the Z8671
- 14 I/O lines available to the user
- 3 lines for external interrupts
- 3 sources of internal interrupts
- Vectored interrupt structure with programmable priority levels. Each can be individually enabled or disabled
- External memory expansion up to 124 bytes
- Allows calling of assembly language subroutines for time critical applications
- Allows writing and debugging of programs in Basic, simplifying the programming of on-critical subroutines
- Basic/Debug can directly address the Z8671's internal registers and all external memory. It provides quick examination and modification of any external memory location or I/O port
- On board static RAM can be replaced by Eprom for auto program execution on power up or Reset.

## Architecture

The Z8671 Single Board Computer uses the following IC packages

IC1	Z8671	(Z8601 preprogrammed with BASIC/DEBUG)
IC2	M5516	(2K bytes of static RAM)
IC3	T74LS373	(8-bit latch for address)
IC4	M2716/M2732	(2K bytes of EPROM)
IC5	MC1488	(RS232 line driver)
IC6	MC1489	(RS232 line receiver)
IC7	T74LS30	(8 inputs nand gate)
IC8	T74LS30	(8 inputs nand gate)

**Memory Addressing.** The addressing of the on board 2716 EPROM is fully decoded at F800H to FFFFH and the 2K RAM at 1000H to 17FFH. The M2716 Eprom can be replaced with a 4K Eprom with the correct jumper pads soldered. To use as a microcontroller, the 2K RAM can be replaced by a 2K Eprom after the ram contents is transferred to the 2K Eprom.

Decimal	Hex	Contents
0-2047	(0-7FF)	Internal ROM (BASIC/DEBUG)
4069-6143	(1000-17FF)	RAM (5116)
63488-65535	(F800-FFFF)	EPROM (2716)

Table 1. Memory Addressing

## Interfacing the Z8671 with RS232 Port

The Z8671 uses its serial communication port to communicate with the RS232 port line driver and receiver are used to supply the proper RS232 signals.

The serial interface does not use the control signals Clear to Send, Data Set Ready, etc. It uses only serial In, Serial Out and Ground, so it is a very simple interface.

The Z8671 uses one timer and its associated prescaler for baud rate control. On reset, the Z8671 reads location FFFD and uses the byte stored there to select the baud rate.

On reset the Z8671 reads FFFDH, which is in EPROM and decodes the baud rate from the contents of that location.

Table 2 shows what the location FFFDH

should read to select a particular baud rate to be used by the Z8671.

Content of location FFFDH			Baud rate
		LSB	
1	1	1	300
1	1	0	110
1	0	1	1200
1	0	0	2400
0	1	1	4800
0	1	0	9600
0	0	1	19200
0	0	0	150

Table 2 Baud Rate Code

## I/O Ports

Port 1 of the Z8671 is used as a multiplexed Address/Data bus, a octal latch (IC3) is used to latch on the address signals.

The latch is enabled by the  $\overline{AS}$  output of the Z8671.

Port 0 outputs the higher 8 bits of the address from A<sub>8</sub>-A<sub>15</sub>.

Two 8-inputs nand gates (IC7 & IC8) fully decodes the addressing of the M5116 RAM and M2716 Eprom. On Port 3, line 0 and 7

are used for serial input/output communication. These lines are driven by the line drivers/receivers (IC5 & IC6).

Port 3 lines 1 to 6 are free for user configuration.

All 8 lines of Port 2 are also available to the user. These lines can be programmed for open-collector or normal TTL outputs as well as inputs.

---

## Software

When the Z8671 is reset, a prompt (":") appears, Basic commands can then be in direct or program mode.

The following sequence is a simple I/O example.

```
:10 input a
:20 "a=";a
:run
?5
a=5
:list
10 input a
20 "a=";a
:
```

When a number is entered as the first character of a line, the Basic monitor stores the line as part of a program. In this example, "10 input a" is entered. Basic stores this instruction in memory and prints another ":" prompt. The run command causes execution of the stored program. In this example, Basic asked for input by printing "?". A number (5) is typed at the terminal. Basic accepts the number, stores it in the variable "a", and executes the next instruction. The next instruction (20 "a=";a) is an implied print statement: writing an actual "print" command is not necessary here. The command "list" caused Basic to display the program stored in memory on the terminal.

**Reading Directly From Memory.** Basic lets the user directly read any byte or word in memory using the Print command and "@" for byte references or "↑" for word references:

```
:print @8
10
:printhex (@8)
A
:printhex (↑8)
AF6
:
```

The first statement prints the decimal value of register 8.

The next statement prints the hexadecimal value of Register 8 and the last statement prints the hexadecimal value of Register 8 (0FH) and Register 9 (F6H).

**Writing Directly to Memory.** Basic lets the user write directly to any register or RAM location in memory using the Let command and either "@" or "↑".

```
:@%a=%ff
:↑4096=255
:print @10
255
:printhex(↑%1000)
FF
:
```

The let command is implied to save memory space but can be included. The first statement loads the hex-decimal FF into register 10 decimal (AH). The next instruction loads the decimal value 253 into register 4096 decimal (1000H). The print commands write to the terminal the values that were put in with the first two instructions.



---

## Software (Continued)

**Commands.** Basic/Debug recognises 15 command keywords. For detail instructions of command usage refer to the Basic/Debug Software Reference manual.

**GO** The GO command unconditionally branches to a machine language subroutine. This statement is similar to the USR function except that no value is returned by the assembly language routine.

**GOSUB** GOSUB unconditionally branches to a subroutine at a line number specified by the user.

**GOTO** GOTO unconditionally changes the sequence of program execution (branches to a line number).

**IF/THEN** This command is used for conditional operations and branches.

**INPUT/IN** These commands request information from the user with the prompt "?" then read the input values (which must be separated by commas) from the keyboard, and store them in the indicated variables INPUT discards any values remaining in the buffer from previous IN, INPUT or RUN statements, and requests new data from the operator. IN uses any values left in the buffer first, then requests new data.

**LET** LET assigns the value of an expression to a variable or memory on the terminal device.

**NEW** The NEW command resets pointer R10-11 to the beginning of user memory, thereby marking the space as empty and ready to store a new program.

**PRINT** PRINT lists its arguments, which may be text messages or numerical values, on the output terminal.

**REM** This command is used to insert explanatory messages into the program.

**RETURN** This command returns control to the line following a GOSUB statement.

**STOP** STOP ends program execution and clears the GOSUB stack.

**RUN** RUN initiates sequential execution of all instructions in the current program.

**Functions.** Basic/Debug supports two functions: AND and USR.

The AND function performs a logical AND. It can be used to mask, turn off, or isolate bits. This function is used in the following format:

AND (expression, expression)

The two expressions are evaluated, and their bit patterns are ANDed together. If only one value is included in the parentheses, it is ANDed with itself. A logical OR can also be performed by complementing the AND function. This is accomplished by subtracting each expression from -1. For example the function below is equivalent to the OR of A and B.

$-1 - \text{AND}(-1 - A, -1 - B)$

**Machine Language Functions.** An application often requires a subroutine which can be performed more quickly and efficiently in machine language than in Basic/Debug.

Basic/Debug can call a machine language subroutine which returns a value for further computation by the USR function. To call a subroutine which return no value, the GO@ command is available.

After the machine language routine is assembled and placed in unoccupied memory, it can be called as follows:

USR (% 2000)

## Software (Continued)

The address and arguments are expressions separated by commas. Argument 256 and c variable are passed to the subroutine via registers R20-21 and R18-19 and the machine language subroutine must leave the return value in R18-19.

Call	R18-19 Contains	R20-21 Contains
USR (% 700,A,B)	B	A
USR (% 700,A,)	A	A

**Table 3 USR Argument and Registers**

The machine language subroutine must end with a RET instruction, and leave the value to be returned in R18-19.

Basic/Debug assigns addresses 0 through 255 to the register file. The 144 registers

include four I/O port registers (R0-R3), 124 general purpose registers and sixteen control and status registers. These registers are common to all Z8 Family CPU chips and are described in the Z8 Family Technical Manual. However, Basic/Debug uses many of the general purpose registers as pointers, scratch workspace and internal variables. So, in the Z8671 single board computer, these registers cannot be used by a machine language subroutine or other user programs.

The 2K of internal ROM on the Z8671 chip contains the Basic/Debug interpreter. It begins at address 00 and extends up to 2047, but because Basic/Debug assigns the addresses 0-255 to the register file, the lower 256 bytes of internal ROM may be accessed only by machine language instructions.

127

127	EXPRESSION EVALUATION STACK
84	
83	FREE
34	
33	COUNTER
32	
31	USED INTERNALLY
30	
29	SCRATCH
28	
27	POINTER TO CONSTANT BLOCK
24	
23	USED INTERNALLY
22	
21	LINE NUMBER
20	
19	ARGUMENT FOR SUBROUTINE
18	
17	ARGUMENT/ROUTINE FOR SUBROUTINE CALL
16	
15	SCRATCH
14	
13	POINTER TO INPUT LINE BUFFER
12	
11	POINTER TO END OF LINE BUFFER
10	
9	POINTER TO STACK BOTTOM
8	
7	ADDRESS OF USER PROGRAM
6	
5	POINTER TO GOSUB STACK
4	
3	POINTER TO END OF PROGRAM
2	
1	I/O PORTS
0	

**Table 4 General Purpose Registers With External RAM**

127

127	SHARED BY EXPRESSION STACK AND LINE BUFFER
104	
103	GOSUB STACK
86	
85	SHARED BY GOSUB AND VARIABLES
84	
83	VARIABLE
34	
33	FREE AVAILABLE FOR USER ROUTINES
32	
31	COUNTER
30	
29	USED INTERNALLY
28	
27	SCRATCH
24	
23	POINTER TO CONSTANT BLOCK
22	
21	USED INTERNALLY
20	
19	LINE NUMBER
18	
17	ARGUMENT FOR SUBROUTINE CALL
16	
15	ARGUMENT/RESULT FOR SUBROUTINE CALL
14	
13	SCRATCH
12	
11	POINTER TO NEXT CHARACTER
10	
9	POINTER TO LINE BUFFER
8	
7	POINTER TO GOSUB
6	
5	POINTER TO BASIC PROGRAM
4	
3	POINTER TO GOSUB
2	
1	FREE
0	
0	I/O PORTS

**Table 5 General Purpose Registers Without External RAM**

## Initialization And Automatic Start-Up.

On Power-On/Reset, Basic/Debug checks for external RAM memory and also for an auto start-up program.

Basic/Debug non-destructively tests memory from low to high addresses. Only one byte of every 256 is tested at relative location XXFD(hex). The first byte of RAM found determines the lower boundary of user memory. Basic/Debug assumes that if XXFD is RAM, XX00 is also RAM and sets pointer R8-9 to XX00 (hex). Basic/Debug continues to test up through memory until it finds a byte that does not contain in RAM.

Basic/Debug assumes it has RAM up to and including YYFF(hex) where YYFD is the last location tested that contained RAM. The top of user memory pointer, R4-5, is set to YY20.

Automatic start-up allows a program stored in ROM to be executed on reset without operator intervention. Automatic execution occurs on power-up/Reset when the program

- is stored in ROM
- begins at 1020(hex)
- begins with a line number between 1 and 254 inclusive.

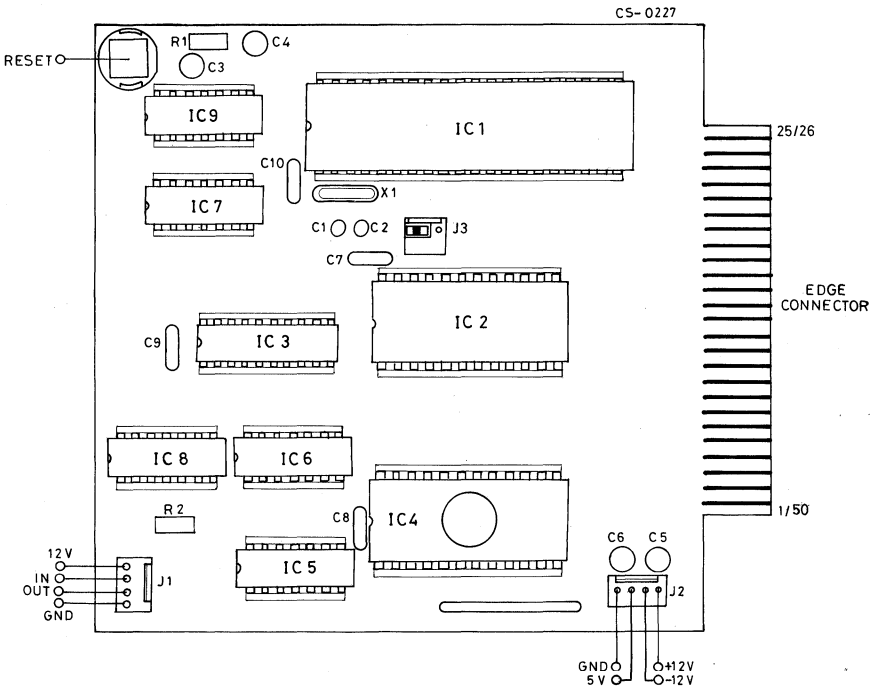
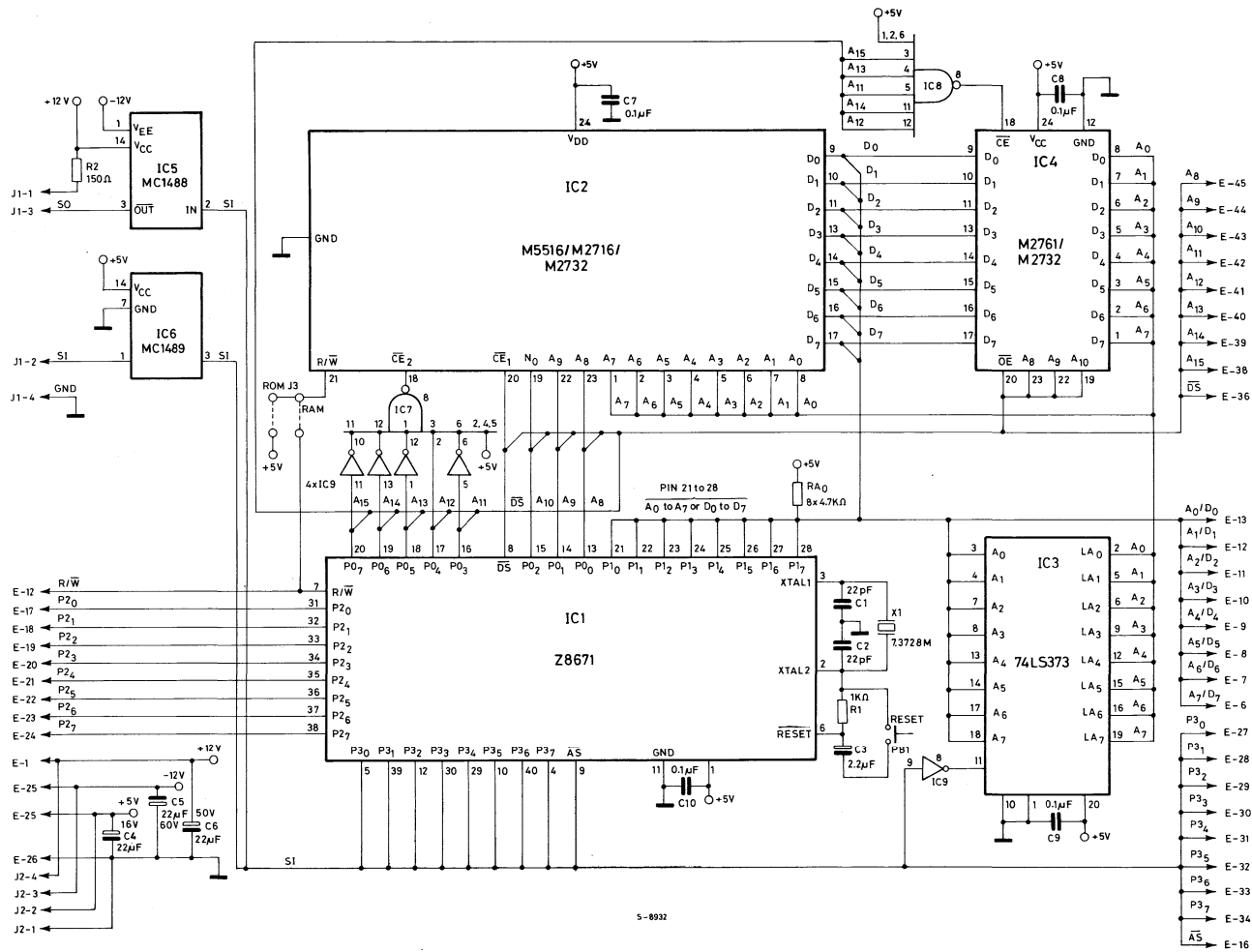


Figure 1 - P.C. Board and Components Layout

Figure 2 - Functional Diagram



5-8932



# Z8 in Electronic Private Automated Branch Exchange (EPABX 2 Ext./8 Int. Lines)

## EPABX Typical Features

**Day/Night Service.** During night service a telephone set has some priority features (E.G., operator). It can receive external calls and switch on others.

**Put in Hold.** During a conversation one party can be placed in "hold condition" to allow other connections and then be reconnected to the original conversation.

**Transfer.** A party can be transferred to another third party.

**Conference Call.** 3 people can speak simultaneously.

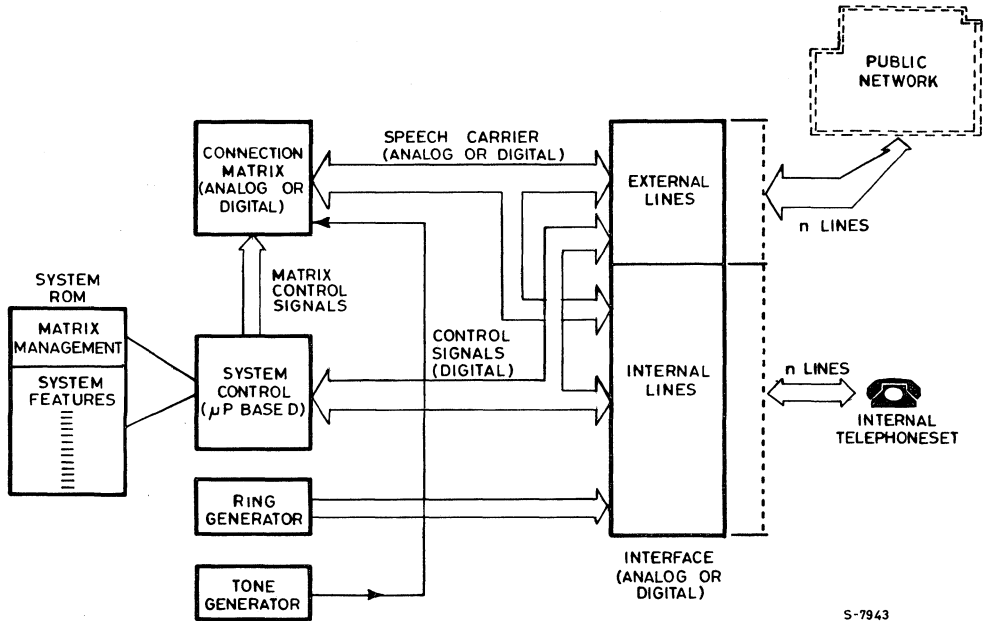
**Alternative Conversation.** Led by one party, connection can be alternated between two other parties.

**Telephone Set.** Uses standard two wire 48V/30mA telephone set.

## General Description

The EPABX is a small electronic private branch exchange with the ability to set up connections between internal extensions and external two wire trunk lines. The EPABX is suitable for connection to a public telephone network.

The switching matrix and the related control logic are semiconductor devices driven by a Z8 single chip microprocessor. The system concept is based on the space-division principle.



S-7943

EPABX: General Structure

## Operating Modes

Two basic modes of operation are possible:

**Day Service** (see Figure 1 and 2). The incoming calls from either external trunk line ring all internal extensions. Any extension can pick up the incoming call by going off-hook.

Access to outgoing line is gained by dialling 0.

At point C (in Figure 1) an alternate conversation (I or t) controlled by "I" is possible using the command "\*5". Or a call conference using the command \*4 or a recall using the command \*6. For flow chart B the call cannot be picked up if "I" has a person on hold.

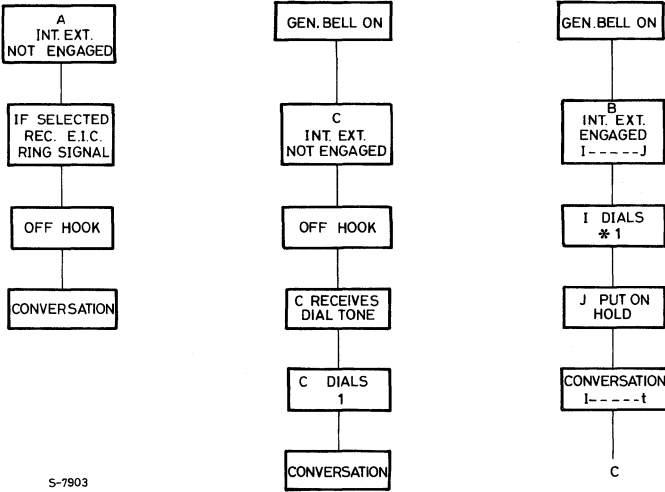


Figure 1. Pick Up an Incoming Call

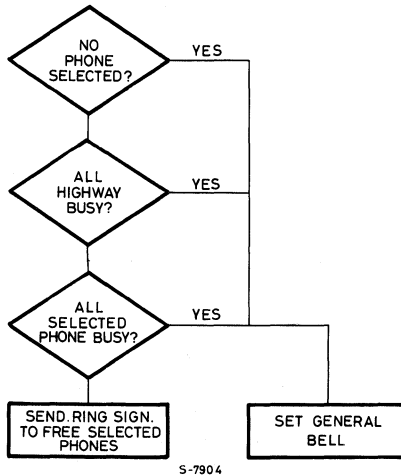
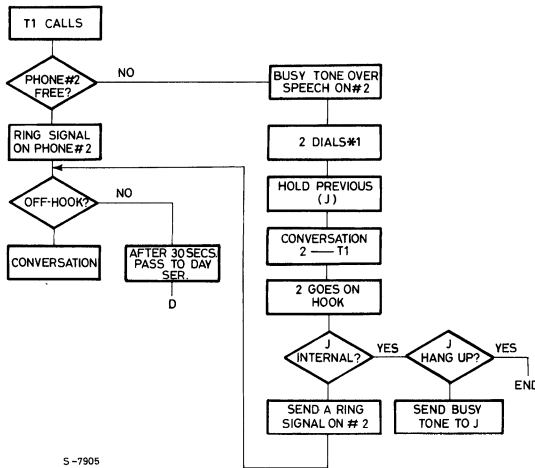


Figure 2. Generation of an External Ring Signal in Day Service

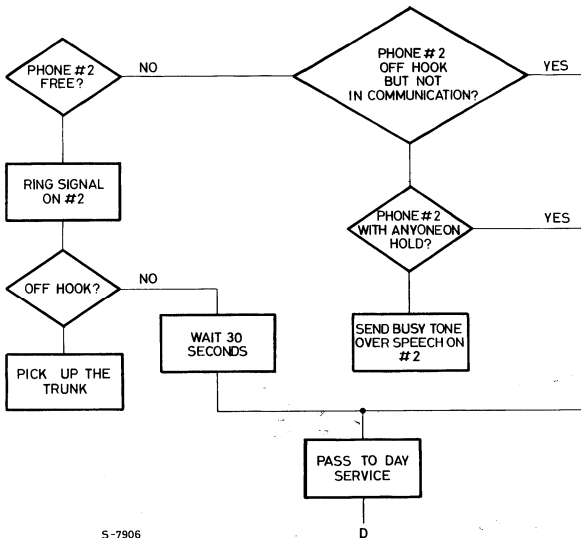
**Operating Modes (Continued)**

**Night Service** (see figure 3 and 4). The incoming calls are routed to one extension (the operator's telephone set). If the operator does not answer within 30 seconds, all the telephone are activated and any internal

extension may pick up the call. If the operator's extension is busy, with either an external or internal conversation, the incoming call will cause a busy tone to be activated into the operator's connection.



**Figure 3. Night Service**



**Figure 4. Generation of an External Ring Signal in Night Service**



## Network Configuration

The network permits the connection of up to 8 internal extensions to the 2 trunk lines. Up to 4 simultaneous conversations can take place (2 internal and 2 external maximum). (see Figure 5 and 6).

**Power Supplies and Power Failure.** The EPABX is powered from an AC line. A power failure will automatically route the two external trunks to two internal extensions which will operate normally without AC power.

**Physical Description.** The EPABX should be housed in a wall-mounted enclosure. The internal lines (telephone sets) and the external trunk lines are connected to a terminal strip.

**Type of Internal Telephone Set and Line Resistance.** The EPABX uses a standard 48V/30mA telephone set with either pulse or DTMF dialing. The maximum line resistance for internal extensions is 1000 ohms.

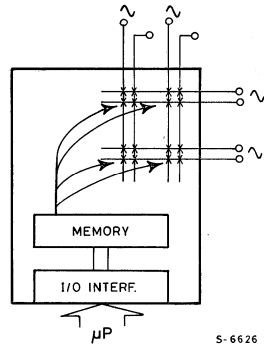


Figure 5. Single Cell Matrix Structure by M079 2x2 Cross Point, for Balanced Connection

The M079 is used in severe conditions (low attenuation, fully separation of channel, due to double switches).

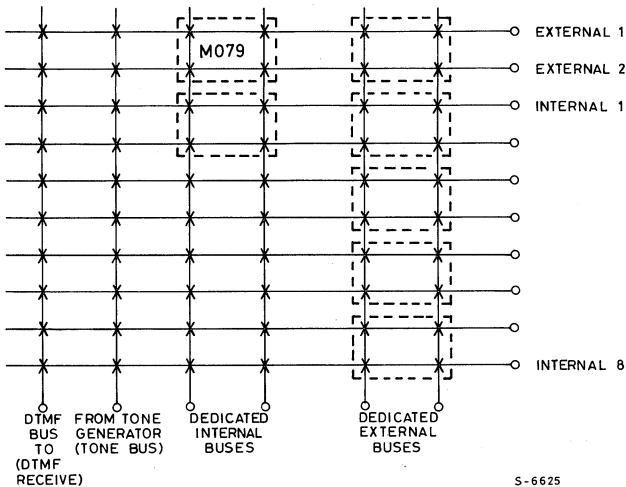


Figure 6. EPABX (Matrix)

## Network Configuration (Continued)

**Internal Circuit Blocks.** The main EPABX system is divided into the following blocks:  
 The power supply unit providing -48V supply for the telephone sets, 70V/20Hz ringing voltage with 1 sec/4 sec modulation, tone generator for

- Busy tone,
- Dial tone,
- Ringing tone,

giving a 425Hz sine wave modulated as shown in flow chart 1, with an output level of -10dBm,

A central unit containing the microprocessor and switching matrix circuits (Figure 7), and line interface circuitry for 8 local extensions and two trunk lines. (Figure 8 and 9). Both Must Perform:

- Galvanic insulation
- DC/AC DC coupling
- AC signal bias (to be treated by electronic matrix)

- Line status supplied to microprocessor protection

"A" Type must provide:

- Dialling actuator (only for pulse dialling)

"B" Type must provide:

- Ring injection (70V, 20 Hz)
- DC feeding (-48V)

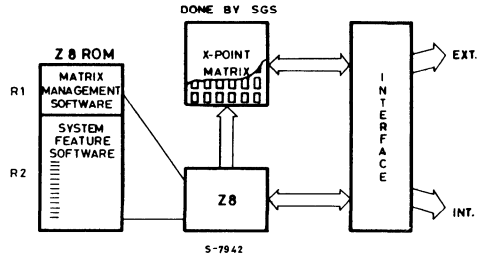


Figure 7. Switching Matrix Circuit

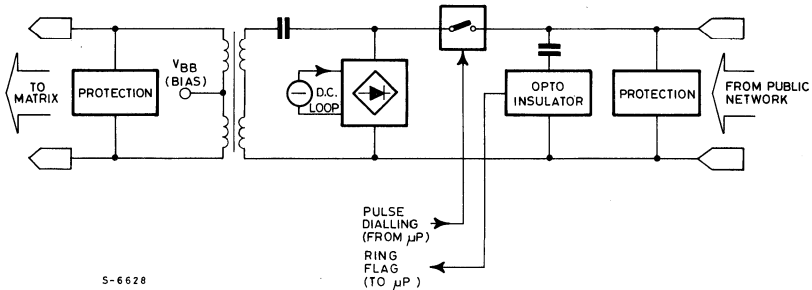


Figure 8. "A" Type: External Interface

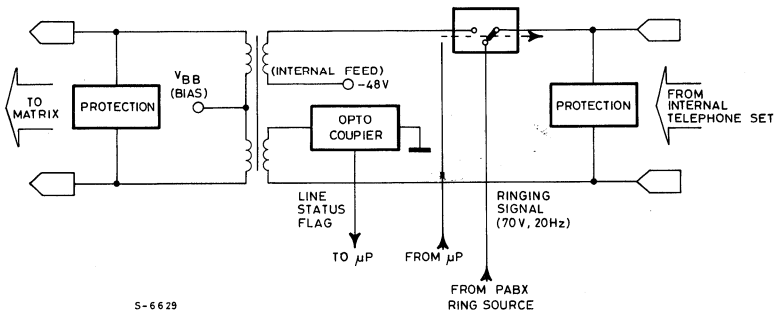


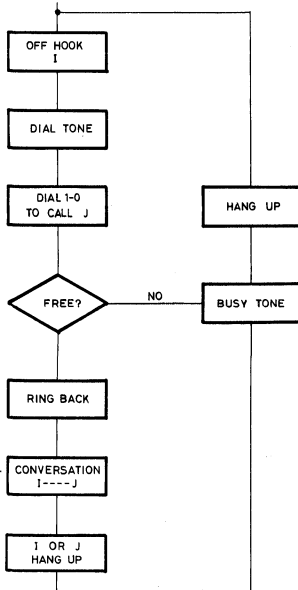
Figure 9. "B" Type: Internal Interface

## Network Configuration (Continued)

### Allocation of Numbers for Extensions and Functions.

Internal extensions	2 to 9
Prefix to a call	0 for trunk access 1 for trunk call answer
Suffix during a call	4 for conference call 5 alternate conversation 6 recall/hold

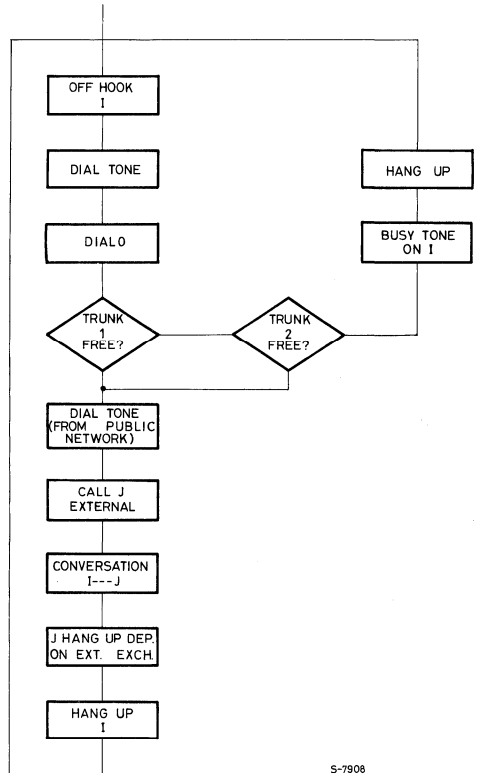
**Internal Calls.** (see Figure 10). When an extension is taken "off hook" a dial tone is received. An extension number from 2 to 9 may be dialed. If the called party is idle, a ringing tone is received, if the called party is busy, a busy tone is received. When the conversation has ended, and one of the extensions is placed "on-hook", the other will receive a busy tone.



S-7907

Figure 10. Internal Calls

**Outgoing Calls.** (see Figure 11). An outgoing call is initiated when an extension is taken "off-hook" and a "0" is dialed. If one of the two trunk lines is free, it will become available and will be received a dial tone from the public exchange. There is no limit on, or control over the number subsequently dialed.



S-7906

Figure 11. Outgoing Calls

## Network Configuration (Continued)

**Hold/Recall.** (see Figure 12). After a call is placed on hold by following the appropriate procedure, another internal or external line may be dialed. The call 'on hold' may be recalled by dialing 6 and the previous conversation resumed.

**Transfer.** (see Figure 12). After placing a conversation on hold and establishing a conversation with another person, a transfer may be executed by going "on-hook". However, a transfer to interconnect two external lines is not allowed.

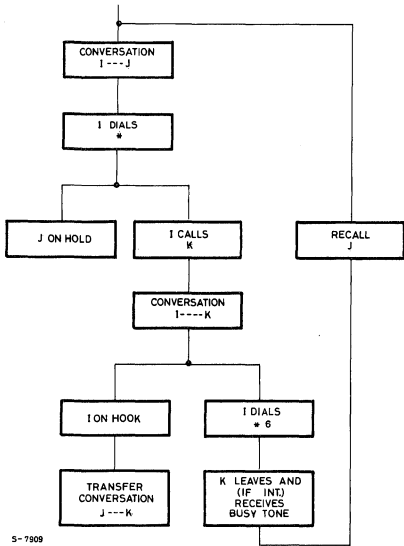


Figure 12. Hold/Recall and Transfer

### Notes:

- J can be external or internal
- K can be external or internal, transfer will not take place if J and K one both external.
- I is the master of the call and J and K are "slaves". If the generator goes on-hook the second person called becomes the master (only if the second person called is not a trunk, otherwise the first person called becomes the master or viceversa). If a slave goes on-hook, the generator remains the master.

**Alternate Conversation.** (see Figure 13). When a call has been placed on hold and another conversation initiated, a user may alternate between the two lines by dialing 5.

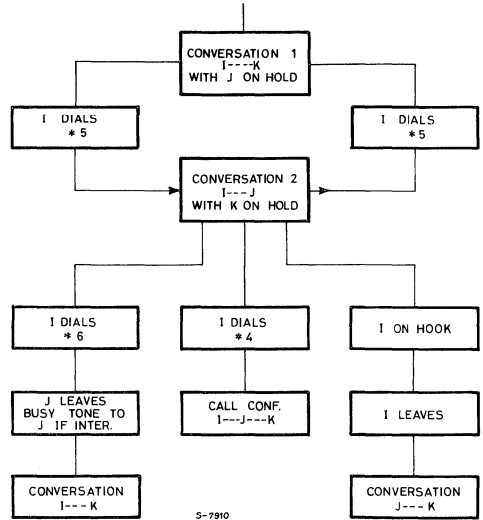


Figure 13. Alternate Conversation

### Notes:

- I is an internal extension and is the master of the call.
- J & K are internal or external but if J & K are external call conference is allowed.
- Alternate calls are controlled (\*5) by the person who called the 3rd party with \*6.
- If a person on-hold hangs up, he is automatically disconnected, and consequently not found with the \*5 command; the generator receives a busy tone and the original conversation can, however, be resumed with the \*6 command. If the generator goes on-hook the second person called becomes the master (only if the 2nd person called is not a trunk, otherwise the first person called becomes the master or viceversa).
- If a slave goes on-hook the generator remains the master.

## Network Configuration (Continued)

**Conferece Call.** (see Figure 14). A conference call is possible up to a maximum of either one external and 2 internal conversations, 3 internal conversations, or 2 external and 1 internal (where allowed).

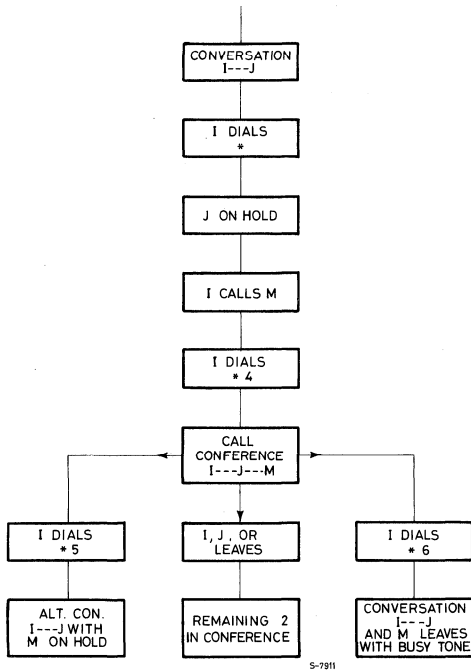


Figure 14. Conference Call

### Notes:

- I, J and M may be:
  - (a) three internal lines
  - (b) two internal and one external or viceversa.
- Two external lines can be put in conference.
- I is an internal extension and is the master.
- If I leaves, the second person called becomes the master (only if the 2nd person called is not a trunk, otherwise the first person called becomes the master or viceversa.
- If J or M (slaves) leaves the generator (I) remains the master.

### Flow Chart Comments

In these flow charts the following symbols are used:

- |                |   |
|----------------|---|
| I, J, K, M     | user identifications<br>(internal or external)  |
| T1, T2         | trunk lines 1 and 2   |
| dial tone      | self explanatory  |
| busy tone      | self explanatory  |
| ring back tone | self explanatory  |
| *              | momentary depression of<br>the rest (or pushing the<br>momentary connection<br>button on phones equip<br>with this feature) |
| 1 - 0          | dialling 1 to 0   |
| I...J          | connection and<br>conversation of I with J  |
| E.I.C.         | external incoming call  |

## Tone Definition Timing

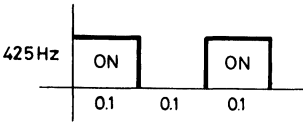


Figure 15. Busy Tone Over Speech

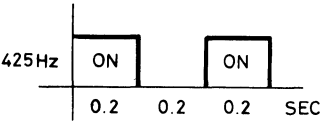


Figure 16. Definition of Tones, Busy Tone.

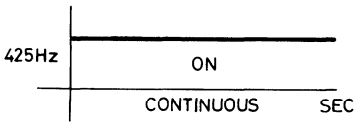


Figure 17. Dial Tone

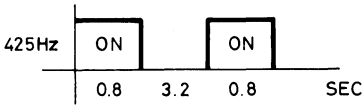


Figure 18. Ring Back Tone

S-7912

## Ring Signals Definition Timing

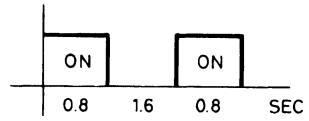


Figure 19. General Bell Ring Signal

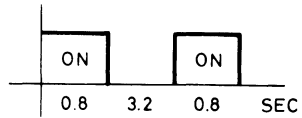


Figure 20. Internal Incoming Call, Ring Signal

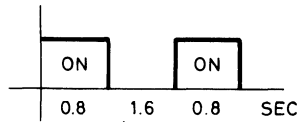


Figure 21. External Incoming Call, Ring Signal

S-7913

## General Timing Description

Parameter	Characteristic
Dial pulse internal - Break space - Mark space	More than 50 ms. Less than 75 ms More than 30 ms
Dial pulse on trunk	As generated on internal extension
Inter digit	800 ms
Flash	More than 200 ms - Less than 800 ms
Flash on trunk line	80 ms
On-hook	More than 800 ms
Antibounce on DTMF key	37.5 ms
Delay after freeing the trunk line before engaging the line again	200 ms
Dial tone	Continuous 425 Hz
Busy tone	200 ms ON/200 ms OFF/425Hz
Busy tone over speech	100 ms ON/3.2 sec OFF/425Hz
Ring back tone	800 ms ON/3,2 sec OFF/425Hz
Internal call ring signal	800 ms ON/3.2 sec OFF
External call ring signal	800 ms ON/1.6 sec OFF
Gen bell signal	800 ms ON/1.6 sec OFF
Antibounce on ext-incoming call signal receiving	37.5 ms sec

# Using Z8 MCU in Keyboard Controller

## Introduction

The Z8 MCU can be used advantageously in a professional keyboard controller, with serial scanning.

The principal points on which the application is based are, in essence:

- Use of a new generation high-performance microcomputer (Z8), which is well adapted to real time applications; and
- implementation of a serial scanning technique for the matrix, facilitated by Z8's remarkable execution velocity, with the possibility of defining a single dedicated interface circuit which includes a limited number of pins (28), and is thus less costly.

---

## Description

It should be particularly noted that, if the architecture and the software available in the Z8 MCU are used in a timely way, it is possible to achieve scanning times at a level of 2.7 msec for 128 keys with the 8 MHz Z8. Using faster versions (12 MHz), it is possible to achieve shorter scanning times.

The matrix interface circuit interacts with Z8 by means of 2 lines:

- CK coming from Z8; and
- IN going toward Z8.

The CK signal is generated by Z8 and enables the column counter and the line multiplexer with its rising edge.

The circuit interfaces with the matrix via the 16 enabling column outputs in open-drain, and the 8 line inputs.

For these inputs, the specific thresholds ( $V_{ILmax} = 1.5 V$ ) guarantee a correct interface with the matrix.

During the micro's initialization phase, this takes place when the interface matrix is reset, transmitting a timely impulse to the IN pin, which thus becomes bidirectional.

This arrangement permits leaving out a

less reliable function, power-on reset, which is built into the device, without adding it to a later pin, which would result in the package being carried to a higher pin number (40), thus at a higher cost.

The complexity of the interface function has been valued at the equivalent of about 200 gates.

The other functions available in the keyboard, namely: mouse control; and serial conversation control synchronized with the P.C. can be easily worked out using 3 of the 4 external IRQ pins.

These vectorized interrupt requests can be masked and prioritized through use of software. An eventual function, namely controlling asynchronous serial conversation, can be freely obtained, since it already exists in the Z8. In this case, one of the 2 internal timers is used as a baud rate generator.

Driving eventual LEDs (caps lock, number lock) can be achieved directly by programming the respective pins of port 2 in open-drain. Using LED limiting resistance in series, the typical level of current of 10 mA can be programmed.

---

## Note Concerning the Block Diagram

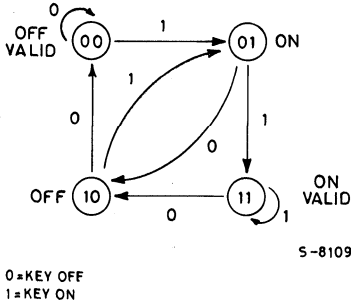
It is possible to write a single program which can automatically recognize if the device being used is a Z8 ROMless or a Z8 masked. This permits an immediate branching from the version with external

EPROM to that using Z8 2K/4K ROM. The test is applied to bits D3 and D4 of register 248 (MODE REGISTER P0 and P1) immediately after reset.



**Note Concerning the Printout**

Status is coded on 2 bits. The status table is made up of 32 bytes with 4 states per byte. The status coding used is the following:



This type of coding permits:

- Branching from the previous state S1 S2 to the new state S2 IN by retaining a bit from the previous state (S2) and introducing the real state of the key (IN). Status updating is, therefore, easily achieved through rotation and shift operations.
- A simple test for a valid key: the valid key (on or off) is identified by having the 2 status bits identical (00, 11). This permits testing key validity and jumping to the corresponding service routine, using the single instruction JR NOV, OUT. The previous instruction on rotation really sets an overflow bit if the two status bits are different.

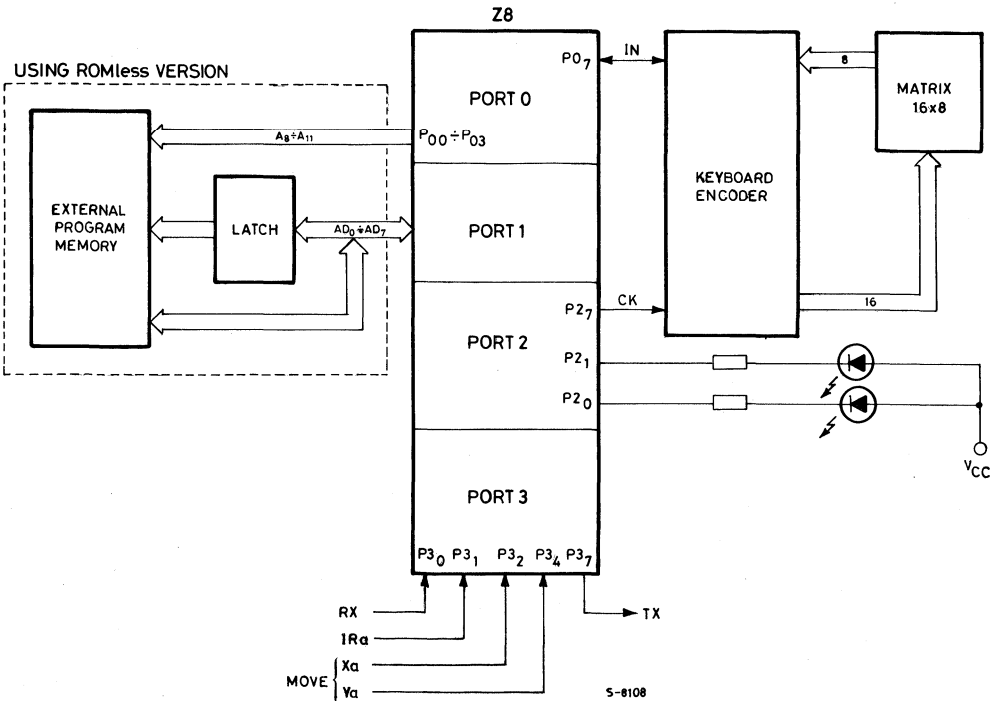


Figure 1. Block Diagram

---

## Program Routine

```
A>TYPE KEYX.ASM
RBASE: EQU 0E0H          ;BASE FOR WORKING REGISTERS
TPTR:  EQU RBASE+13     ;R13=TABLE POINTER (TPTR)
NSCAN: EQU RBASE+14     ;R14=COUNTER OF N. OF SCANNING INTO THE TABLE (4)
NBYTE: EQU RBASE+15     ;R15=COUNTER OF N. OF BYTES INTO THE TABLE (32)
      ORG 0CH
      LD P2M,#0         ;P20-P27 OUTPUTS,CLK OUTPUTS ON P27
      LD P3M,#01        ;P2 WITH PULL-UPS ACTIVATED
      LD P01M,#45H      ;P10-P17 OUTPUTS,P00-P07 INPUTS,INTERN.STACK
      ;KEY INPUT ON P00
      LD TPTR,#10H      ;POINTER TO 1ST BYTE TO PRESET
      LD NBYTE,#20H     ;NUMBER OF BYTES TO CLEAR
      LD R7,#0FFH      ;MASK TO PRESET STATUS TABLE
OPEN:  LD @TPTR,R7
      INC TPTR
      DJNZ NBYTE,OPEN   ;PRESET ALL KEY STATUS TO OPEN CONDITION
      LD R4,#80H        ;MASK TO RESET CLK
      LD R5,#7FH        ;MASK TO SET CLK
      OR R2,R4          ;RESET CLK
      LD NBYTE,#01
      LD NSCAN,#01
      JR IN
START0: LD TPTR,#0FH    ;TPTR=STARTING @ OF STATUS TABLE MINUS ONE
      LD NBYTE,#20H    ;NBYTE=NUMBER OF BYTES INTO THE TABLE
START1: LD NSCAN,#04    ;NSCAN=N. OF SCANNING TO DO FOR EACH BYTE
      INC TPTR
START2: RL @TPTR
      RL @TPTR
      RRC R0            ;SHIFT INPUT TO CARRY FLAG
      OR R2,R4          ;RESET CLK
      RRC @TPTR
      RRC @TPTR        ;GENERATE NEW STATUS ON BITS 6 AND 7
      JR NOV,OUT       ;TEST NEW STATUS AND GO OUT IF IT'S 00 OR 11
IN:    AND R2,R5        ;SET CLK
      RL @TPTR
      RL @TPTR        ;PREPARE THE TABLE FOR THE NEXT KEY VALUE
      DJNZ NSCAN,START2 ;CHECK IF ALL THIS BYTE IS SCANNED
      DJNZ NBYTE,START1 ;CHECK IF THE TABLE IS COMPLETED
      JR START0        ;AT THE FIRST KEY JP TO START0
OUT:   JR IN
```



# Z8 MCU in Dynamic Keyboard

## The Measurements of Pressure Velocity on the Keyboard of a Musical Instrument

### Introduction

Among the characteristics requested from new generations of musical instruments, in particular the piano effect, is the control of amplitude in connection with the velocity of keyboard pressure, in other words, the so-called dynamic keyboard.

Software using the Z8 and supported by the M112 has been studied by SGS in order to permit the creation of a dynamic keyboard with unlimited polyphonicity, up to a maximum of 104 usable keys on one or two keyboards.

The Z8/D, which may be considered as a standard component for this function,

together with the M112, makes it possible for builders of musical instruments to create a dynamic keyboard without having to develop the software or to use complex and costly circuits.

The number of M112s used together with the Z8/D depends on the polyphonicity desired, given the fact that each M112 provides a polyphonicity of 8 notes.

The circuitry necessary for each channel's configuration is significantly reduced thanks to the M112's characteristics of being able to carry out internally the SUSTAIN and RELEASE phase.

---

### Main Characteristics

- The number of usable keys is programmable from 40 to 104.
- Contact may be made by a double bar or by a double rubber contact.
- Total polyphonicity, limited only by the number of M112s and by the velocity of the master/micro.
- Asynchronous conversation with master micro.
- 21 tables which specify the amplitude of the sound identified in relation to flight time (key speed).
- 2 output forms for each key, arranged for choice among the 21 available.
- Each table consists of 15 words of 6 bits each.
- Transition time between the two bars: min. 3 ms and max. 39 ms with  $CK=8$  MHz (from 0 to 3 ms, equals maximum amplitude, while, beyond, 39 ms equals the minimum level).
- Possibility to change the transition time by varying the CK.
- Exclusion of rebounds on either bar.
- Sending reset to master/micro during initialization and in case of error.
- Percussive and/or sustained effects.
- Each key is defined by 14 bits in the following way: 7 bits codify the key number from 0 to 103; 1, the condition of the key; 6 codify the amplitude.

Note: the M112s should have  $3V \pm 10\%$  on pin 12 (Vt); HOLD command = 0 and the code for attack time should never be  $a1 = a2 = a3 = 0$ . Thus the RELEASE phase can be controlled by the M112.

---

### Operation

The keyboard, or keyboards, the extension of which is programmable, make up a matrix of  $8 \times N$  ( $N_{max} = 13$ , or in other words, 104

keys max.) and every N line is made up of 2 bars, one for release and the other for pressure (see Figure 8) - the double contact

## Operation (Continued)

rubber can also be used (see Figure 9).

Keys not being used are in contact with the release bar and when they are played, they touch the pressure bar.

Transition time between one bar and the other, which we will call "flight" time, does not include rebounds insofar as the count starts from the last rebound on the release bar and ends with the first contact with the pressure bar.

The calculation of flight time takes place by increasing, every time the keyboard shifts, a 4-bit counter with which a pair of tables chosen by the user from among the 21 available can be addressed.

Acquisition of the two tables selected and of the number of keys is only possible at mains on.

The time for a shift cycle represents the basic measuring unit of "flight" time and thus is fixed at a constant value of 1.5 ms through use of the Z8/D's internal timer with  $CK = 8$  Mhz.

The first 5 steps are increased every 1.5 ms, while the successive 10 are every 3 ms.

A key is considered pressed at its first contact with the pressure bar, and released at its first contact with the release bar; eventual rebounds, even unlimited ones on the same bar, are ignored.

The Z8/D microprocessor reads the keyboard in a sequential mode and measures flight time for each key that is pressed; that

of release is not taken into consideration. This time can thus only assume 15 discrete values.

Each of these 15 values addresses a pair of amplitudes or levels which should have the note which corresponds to that key.

The micro Z8/D will present on 6 pins (3 associated with port P0 and 3 associated with P3) the code for the first amplitude. After 2  $\mu s$ , the 3 pins P0 (234) which serve, through the use of 3 input AND to strobe and thus memorize this first amplitude, are set to 1.

The second amplitude is made available on the same 6 pins, 20  $\mu s$  after the first, and is maintained until the release of the key or until the successive output (see Figure 3).

The acceptance of keys by the Z8/D is conditioned by the velocity of the acquisition of the master/micro with respect to the quantity of variations of keys in the unit of time.

In order to not limit polyphonicity because of excessive slowness of the master/micro, a FIFO type memory area, has been prepared inside the Z8/D, capable of holding up to 10 variations ready for the master/micro to use.

The Z8/D micro accepts any key pressed indistinctly, taking account of the fact that those coming after filling the FIFO are lost. The decision of which keys to send to the M112 and thus to play, is left to the master/micro.

## Communicating with the Master/Micro

At switch on, the Z8/D puts, on port P1, the code FF which is received by the master/micro and interpreted as a reset.

This code is also sent when, for any reason, the FIFO finds itself in an anomalous condition - for example when the read counter is more advanced than the write pointer. In effect this could leave the "keys pressed" and never cancel them again.

It is therefore necessary that the master micro cancels all pressed keys it has in memory when it receives FF.

All data supplied by the Z8/D is comprised of 14 bits which have the following format:

— 7 bits (P1<sub>0</sub> - P1<sub>6</sub>) give the key number with P1<sub>0</sub> being the LSB.

— 1 bit (P1<sub>7</sub>) gives the key condition (1 = pressed)

— 6 bits (P0[5:6:7] and P3[5:6:7]) give the two amplitudes with P0<sub>5</sub> being the LSB.

Transmission is asynchronous with two signals from the Z8/D being used: DAV (P3<sub>4</sub>) which, when a "1", signals to the master/micro that data is ready to be transmitted and RDY (P3<sub>3</sub>) which must go to "1" to allow the Z8/D to make the data available on its ports.

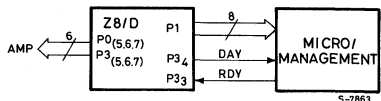


Figure 1

## Communicating with the Master/Micro (Continued)

The timing shown in Figure 2 is relative to an 8MHz clock.

DAV can go to "1" only if RDY is zero and vice versa.

a = 972  $\mu$ s

b = 16.5  $\mu$ s If, however, RDY is not yet at zero, DAV will wait until it goes to zero and, then, after 2.5 us, will go to "1".

c, e, g, m: These times depend on the master/micro. However they should not be less than zero.

d = 7  $\mu$ s In this case, RDY cannot last less than 7  $\mu$ s.

f can vary from 150  $\mu$ s to 1.5 ms.

h = 53  $\mu$ s When this time has passed, the data is complete, but the partial times are:

- \* 13  $\mu$ s OUTPUT P1 (TP and Key No.)
- \* 33.5  $\mu$ s OUTPUT first amplitude with related strobe
- \* 55  $\mu$ s OUTPUT second amplitude

i = 4  $\mu$ s

n can vary from 7  $\mu$ s to 1.5 ms.

Figure 3 shows the detailed timing beginning from the switching of RDY, in other words, from the request for new data and on the supposition that such data is already available:

The data on P1 and those concerning the second amplitude remain stable until the next switching of RDY.

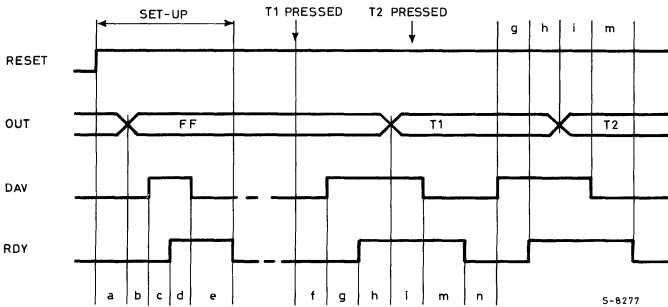


Figure 2

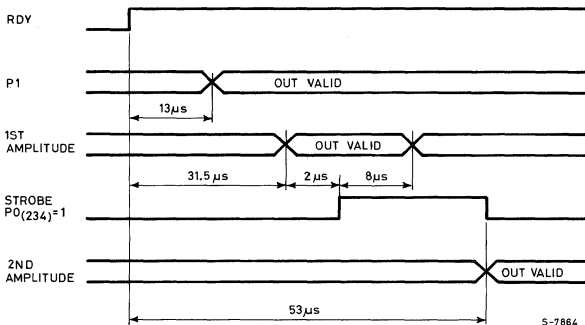


Figure 3. RDY Timing

## Typical Application

Possible applications of Z8/D are many; in this note, two of these are shown, related to percussive effects.

The technique used in this application is based on the fact that the attack phases are not contemporary, in other words, it is not possible to have two or more keys with their attack time overlapping.

This first application allows the saving of a latch or sample and hold for each channel, however the cross-over level between two note decay levels of will not be defined and thus this point must be defined by a timer which, in this case, is made up of rd and Cd.

The following waveforms demonstrate the movement in the signals concerning diagram, in Figure 10.

$$T_a = \frac{(R_o(4051) + R_o(D/A)) * C}{Kohm * C} = \tau$$

$$T_d = RD * C \text{ (RD can be controlled with the assistance of a duty-cycle variable oscillator)}$$

$$T_s = 3Mohm * (3Mohm \text{ is the internal resistance to M112 connected to the mass})$$

$$T_r = R_p * C \text{ (R}_p \text{ is the internal resistance to M112 programmable among r1, r2 and r3)}$$

$T_p$  represents the time interval during which the key is pressed.

St.D and  $t_a$  are generated by the master/micro:

- St.D is the strobe for data transmission for the M112. The negative transition renders the M112 data operational, and initiates the attack phase of the external development circuit.
- $t_a$ , which should begin before the defined St.D, with its negative transition, the attack phase. This time is quite well controlled by the master/micro because, given the characteristics of this application, the next key can begin the attack phase only at the end of the present  $t_a$ .
- $t_d$  is the decay phase time, which should not be confused with the time constant  $T_d$ ; its value is determined by the product  $rd * Cd$  and by the threshold of 4503.

The second application (see Figure 11), requires a temporary memorization of the SUSTAIN level for each channel, however, it permits better envelope control.

In fact, the DECAY phase now tends toward a digitally controlled level and thus the cross-over point between the two slopes is perfectly calculable even to the variation of maximum size in relation to minimum key pressure.

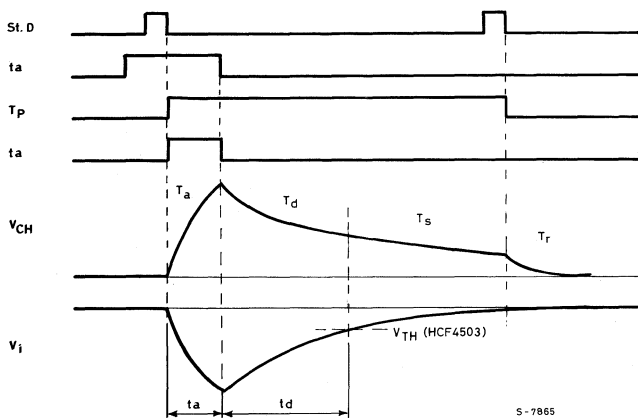


Figure 4. Signal Diagram Relative to Figure 10

## Typical Application (Continued)

Both of the applications just described can be carried out either with the double bar contacts or those with conductive rubber.

If one examines various tables among the 21 available, it is sufficient to choose table 21 and any other one to obtain from the 4 least important bits (P05, P06, P07 and P35) the "flight" time code, from 1 to 15, with which one can address any table inside the

master micro, as is shown in Figure 5, or else an external memory.

Figure 5 shows a possible circuit to read the tables located in the micro/management minimizing the number of pins used in the conversation.

Table 1 and Figure 6 show the numerical and graphical progression of the 21 tables contained in Z8/D.

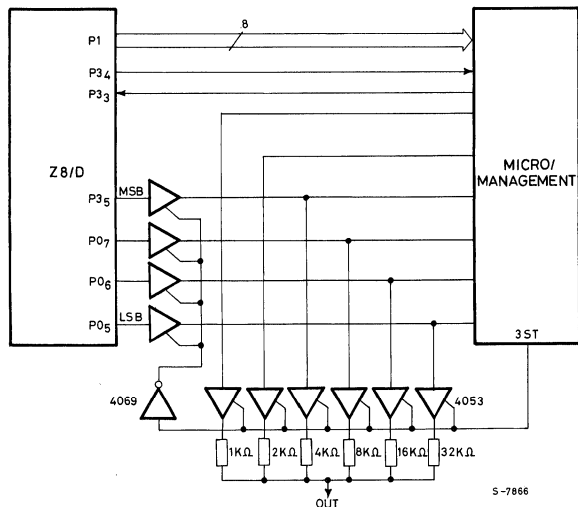


Figure 5. Circuit to Read Tables Resident in the Master/Micro

Flying time (ms)																					
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
3	63	63	63	63	63	63	63	63	63	54	48	45	44	40	31	29	28	28	27	26	1
4,5	63	63	62	62	61	61	59	56	52	51	44	42	40	37	33	31	30	29	28	27	2
6	63	62	61	60	58	56	52	48	45	49	42	36	37	35	35	32	32	30	29	27	3
7,5	62	59	58	58	54	51	47	43	41	47	39	30	35	33	36	32	33	30	30	28	4
9	60	54	50	56	47	47	41	39	39	44	37	27	33	32	34	30	34	29	31	28	5
12	53	46	42	51	43	45	35	36	37	40	33	25	30	29	31	28	33	27	32	29	6
15	47	43	40	49	42	43	34	35	37	35	31	25	28	27	29	27	30	36	32	30	7
18	44	39	38	47	41	42	33	34	35	32	29	24	27	26	29	26	28	25	32	30	8
21	43	37	36	45	40	42	32	34	34	30	27	24	26	25	28	26	27	25	31	30	9
24	42	34	33	40	37	40	31	33	34	28	26	24	25	24	27	25	26	25	30	29	10
27	40	32	30	35	34	37	29	31	31	27	26	24	25	24	27	25	25	24	28	28	11
30	38	30	29	32	31	35	28	29	29	26	25	23	25	23	26	24	25	24	26	26	12
33	36	29	28	30	29	34	27	27	27	25	24	23	24	23	25	24	24	24	26	25	13
36	35	28	28	27	34	25	26	26	24	24	23	24	23	24	24	24	24	23	25	24	14
39	34	28	27	22	26	34	24	24	24	24	24	22	24	22	24	23	24	23	24	24	15

Tab. 1. Table of Amplitude Codes Against Flight Times



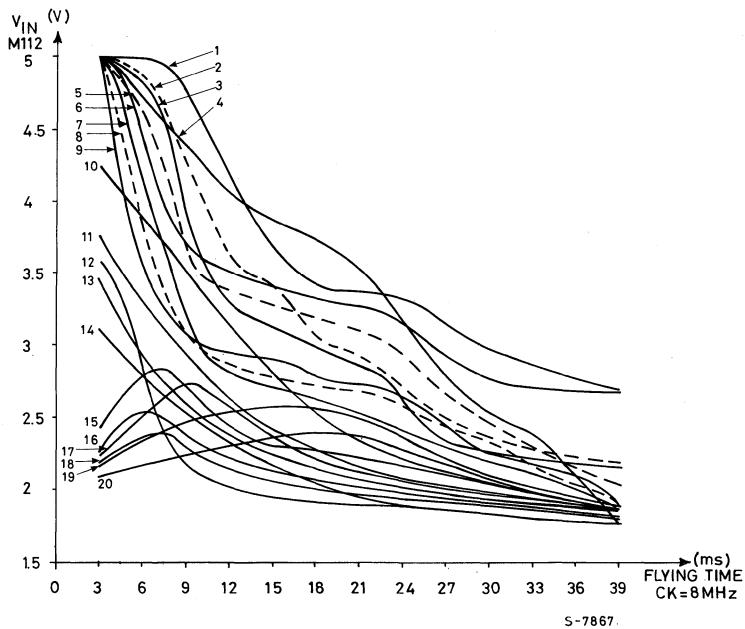


Figure 6. Layout of Tables Resident in the Z8/D

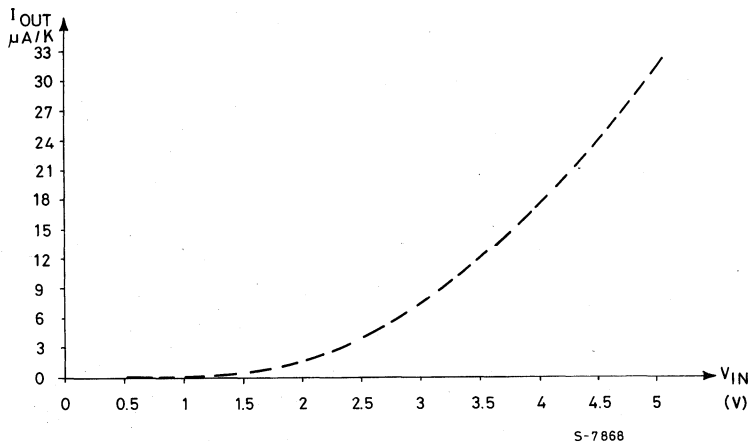


Figure 7

- The release bars are even-numbered.
- Unusual keys are closed with a diode near the release bar.
- The matrix is read at regular intervals of 1,5 ms with  $CK = 8$  MHz.

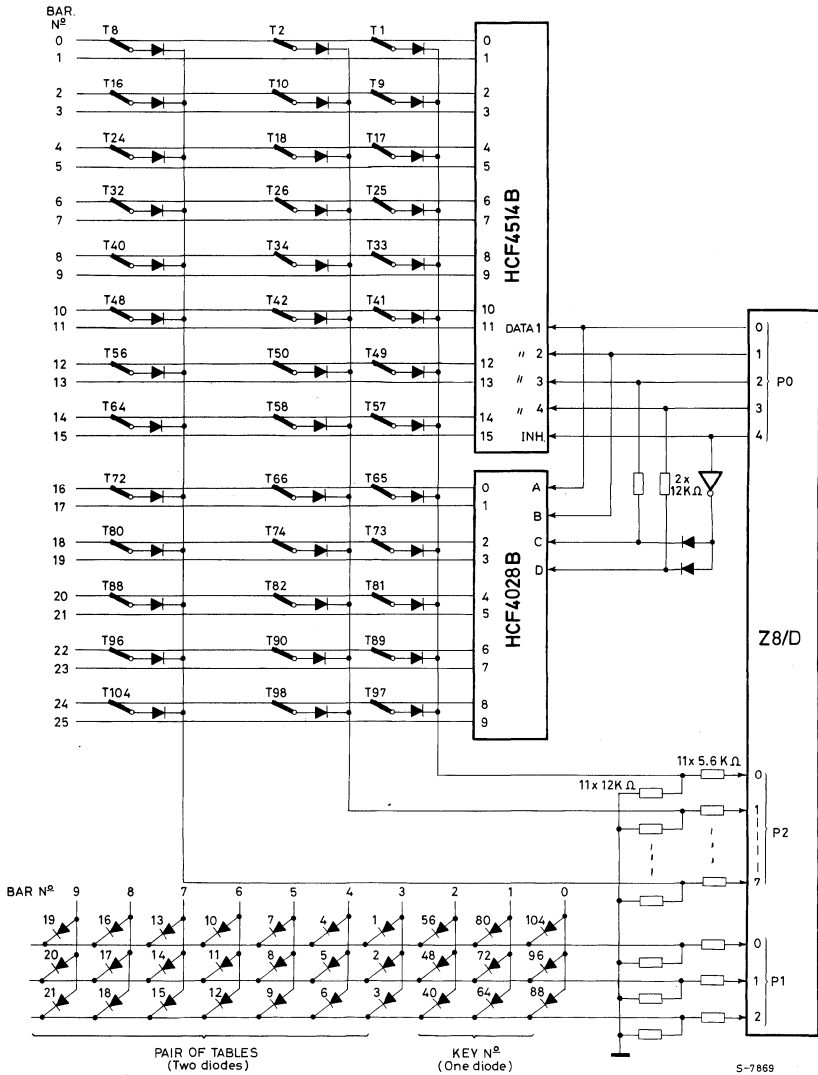
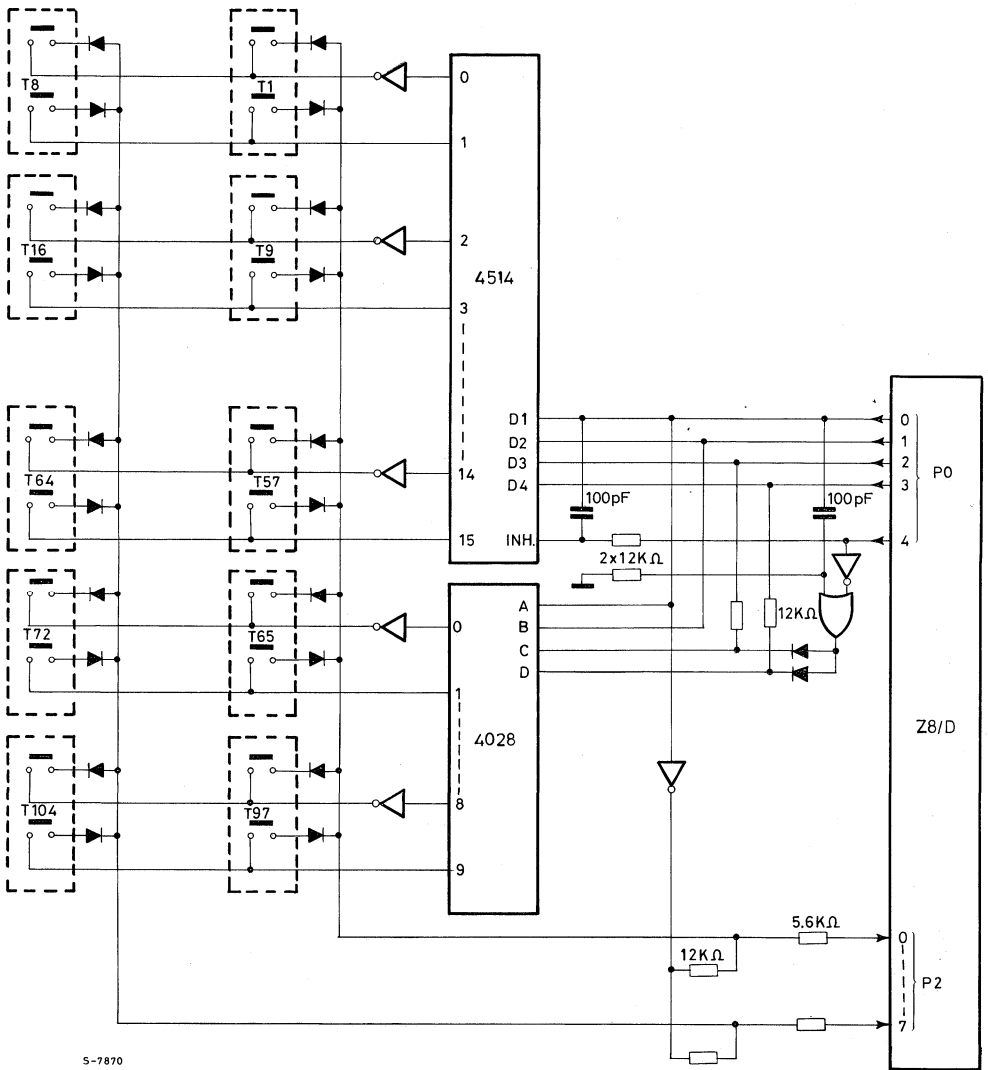
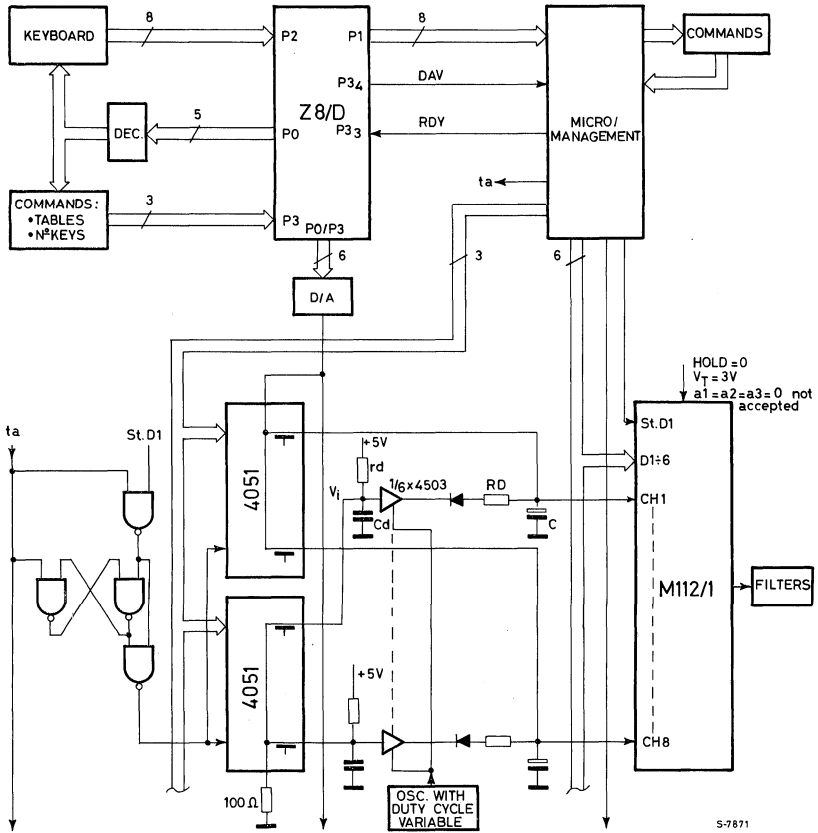


Figure 8. Diagram of Double Bar Keyboard and Commands



5-7870

Figure 9. Diagram of Double Rubber Contact Keyboard



Converter D/A

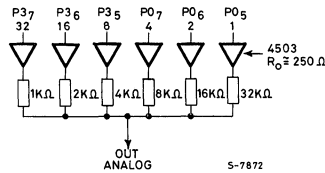


Figure 10. Diagram for Percussive Effects

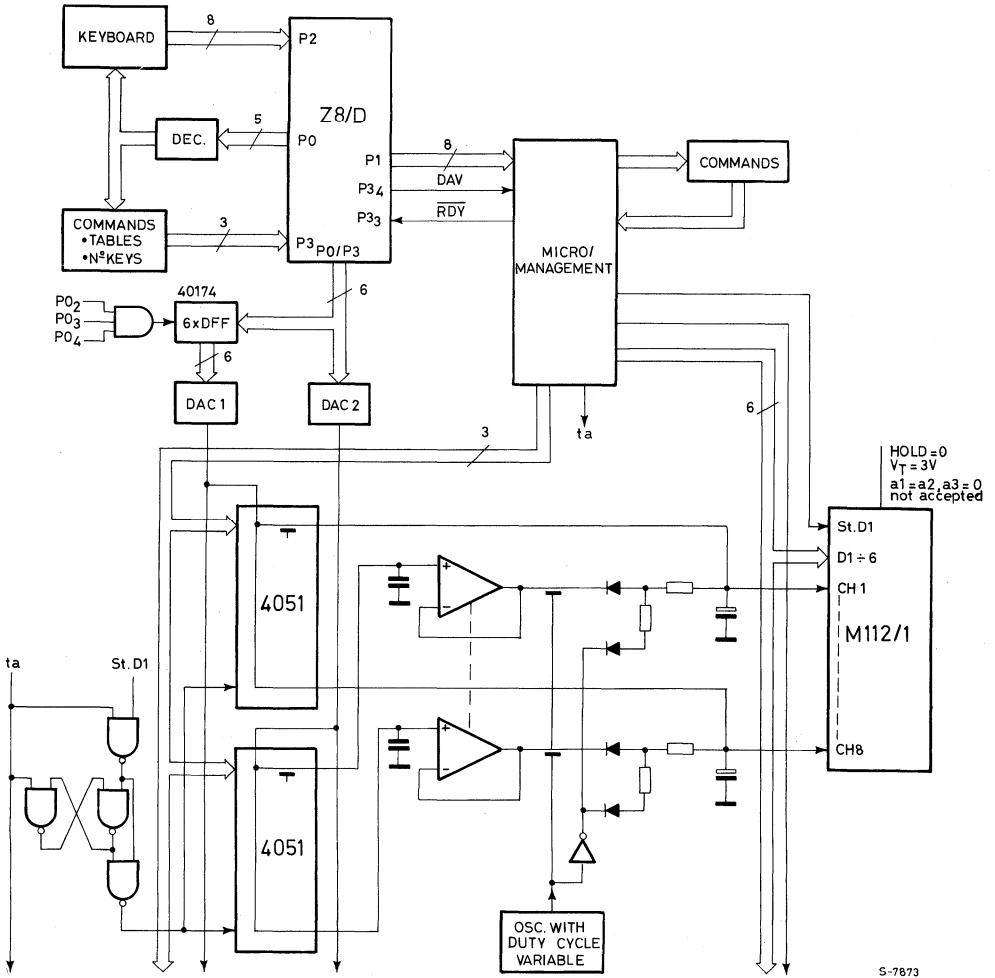


Figure 11. Diagram for Percussive Effects

# Comparison of Z8611, 8051 and MC6801 Microcomputers

## Introduction

The microcomputer industry has recently developed single-chip microcomputers that incorporate on one chip functions previously performed by peripherals. These microcomputer units (MCUs) are aimed at markets requiring a dedicated computer.

This report describes and compares the most powerful MCUs in today's market: the Z8611, the 8051, and the MC6801. Table 1 lists facts that should be considered when comparing these MCUs.

FEATURES	Z8611*	8051	MC6801
<b>On-Chip ROM</b>	4K × 8*	4K × 8	2K × 8
<b>General-Purpose Registers</b>	124	128	128
<b>Special-Function Registers</b>			
Status/Control	16	16	17
I/O ports	4	4	4
<b>I/O</b>			
Parallel lines	32	32	29
Ports	Four 8-bit	Four 8-bit	Three 8-bit, one 5-bit
Handshake	Hardware on three ports	None	Hardware on one port
<b>Interrupts</b>			
Source	8	5	7
External source	4	2	2
Vector	6	5	7
Priority	48 Programmable orders	2 Programmable orders	Nonprogrammable
Maskable	6	5	6
<b>External Memory</b>	120K bytes	124K bytes	64K bytes
<b>Stack</b>			
Stack pointer	16-bit	8-bit	16-bit
Internal stack	Yes, uses 8-bits	Yes	Yes
External stack	Yes	No	Yes
<b>Counter/Timers</b>			
Counters	Two 8-bit	Two 16-bit or two 8-bit	One 16-bit
Prescalers	Two 6-bit	No prescale with 16-bits; 5-bit prescale with 8-bits	None

\* Z8601 = 2K ROM version

Table 1. MCU Comparison

**Introduction** (Continued)

<b>FEATURES</b>	<b>Z8611*</b>	<b>8051</b>	<b>MC6801</b>
<b>Addressing Modes</b>			
Register	Yes	Yes	No
Indirect Register	Yes	Yes	No
Indexed	Yes	Yes	Yes
Direct	Yes	Yes	Yes
Relative	Yes	Yes	Yes
Immediate	Yes	Yes	Yes
Implied	Yes	Yes	Yes
<b>Index Registers</b>	124, Any general-purpose register	1, Uses the accumulator for 8-bit offset	1, Uses 16-bit index register
<b>Serial Communication Interface</b>			
Full duplex UART	Yes	Yes	Yes
Interrupts for transmit and receive	One for each	One for both	One for both
Registers Double buffer	Rcccivcr	Receiver	Transmitter/Receiver
Serial Data Rate	62.5K b/s @8 MHz	187.5K b/s @12 MHz	62.5K b/s @4 MHz
<b>Speed</b>			
Instruction execution average	2.2 $\mu$ sec	1.5 $\mu$ sec	3.9 $\mu$ sec
Longest instruction	4.25 $\mu$ sec	4 $\mu$ sec	10 $\mu$ sec
<b>Clock Frequency</b>	8 MHz *	12 MHz	4 MHz
<b>Power Down Mode</b>	Saves first 124 registers	Save first 128 registers	Saves first 64 registers
<b>Context Switching</b>	Saves PC and flags	Saves PC; programmer must save all register	Saves PC, PSW, accumulators, and Index register
<b>Development</b>	40-Pin ROMless (Z8681/82/84)	40-Pin (8751)	40-Pin (68701)
<b>Eprom</b>	4K bytes 8K bytes	4K bytes	2K bytes

\* Z8601 = 2K ROM version

**Table 1. MCU Comparison** (Continued)

---

## Architectural Overview

The three chips have somewhat similar architectures. There are, however, fundamental differences in design criteria.

The 8051 and the MC6801 were designed to maintain compatibility with older products, whereas the Z8611 design is free from such restrictions and incorporates many new ideas. Because of this, the accumulator architectures of the MC6801 and the 8051 are not as flexible as that of the Z8611, which allows any register to be used as an accumulator.

**Memory Spaces.** The Z8611 CPU manipulates data in four memory spaces:

- 60K bytes of external data memory
- 60K bytes of external program memory
- 4K bytes of internal program memory (ROM)
- 144-byte register file

The 8051 CPU manipulates data in four memory spaces;

- 64K bytes of external data memory
- 60K bytes of external program memory
- 4K bytes of internal program memory
- 148-byte register file

The MC6801 manipulates data in three memory spaces:

- 62K bytes of external memory
- 2K bytes of internal program memory
- 149-byte register file

**On-Chip ROM.** All three chips have internal ROM for program memory. The Z8611 and the 8051 have 4K bytes of internal ROM, and the MC6801 has 2K bytes. In some cases, external memory may be required with the MC6801 that is not necessary with the Z8611 or the 8051.

**On-Chip RAM.** All three chips use internal RAM as registers. These registers are divided into two categories: general-purpose registers and special function registers (SFRs).

The 124 general-purpose registers in the Z8611 are divided into eight groups of 16

registers each. In the first group, the lowest four registers are the I/O port registers. The other registers are general purpose and can be accessed with an 8-bit address or a short 4-bit address. Using the 4-bit address saves bytes and execution time. Four-bit short addresses are discussed later. All general-purpose registers can be used as accumulators, address pointers, or Index registers.

The 128 general-purpose registers in the 8051 are grouped into two sets. The lower 32 bytes are allocated as four 8-register banks, and the upper registers are used for the stack or for general purpose. The registers cannot be used for indexing or as address pointers.

The MC6801 also has a 128-byte, general-purpose register bank, which can be used as a stack or as address pointers, but not as index registers.

The main contrast is that any of the Z8611 general-purpose registers can be used for indexing; the MC6801 and the 8051 cannot use registers this way. The Z8611 can use any register as an accumulator; the MC6801 and the 8051 have fixed accumulators. The use of registers as memory pointers is very valuable, and only the Z8611 can use its registers in this way.

The number of general-purpose registers on each chip is comparable. However, because of its flexible design, the Z8611 clearly has a more powerful register architecture.

The Z8611 has 20 special function registers used for status, control, and I/O. These registers include:

- Two registers for 16-bit Stack Pointer (SPH, SPL)
- One register used as Register Pointer for working registers (RP)
- One register for the status flags (FLAGS)
- One register for interrupt priority (IPR)
- One register for interrupt mask (IMR)
- One register for interrupt request (IRQ)
- Three mode registers for the four ports (P1M, P2M, P3M)
- Serial communications port used like a register (SIO)
- Two counter/timer registers (T0, T1)



---

## Architectural Overview (Continued)

- One Timer Mode Register (TMR)
- Two prescaler registers (PRE0, PRE1)
- Four I/O ports accessed as registers (PORT0, PORT1, PORT2, PORT3)

The 8051 also has 20 special function registers used for status, control, and I/O. They include:

- One register for the Stack Pointer (SP)
- Two accumulators (A, B)
- One register for the Program Status Word (PSW)
- Two registers for pointing to data memory (DPH, DPL)
- Four registers that serve as two 16-bit counter/timers (TH0, TH1, TL0, TL1)
- One mode register for the counter/timers (TMOD)
- One control register for the counter/timers (TCON)
- One register for interrupt enable (IEC)
- One register for interrupt priority (IPC)
- One register for serial communications buffer (SBUF)
- One register for serial communications control (SCON)
- Four registers used as the four I/O ports (P0, P1, P2, P3)

The MC6801 has 21 special function registers used for status, control, and I/O. These include:

- One register for RAM/ROM control
- One serial receive register
- One serial transmit register
- One register for serial control and status
- One register for serial rate and mode
- One register for status and control of port 3
- One register for status and control of the timer
- Two registers for the 16-bit timer
- Two registers for 16-bit input capture used with timer

- Two registers for 16-bit output compare used with timer
- Four data direction registers associated with the four I/O ports
- Four I/O ports

The special function registers in the three chips seem comparable in number and function. However, upon closer examination, the SFRs of the MC6801 prove less efficient than those of the Z8611. The MC6801 has five registers associated with the I/O ports, whereas the Z8611 uses only three registers for the same functions. The MC6801 uses four registers to perform the serial communication function, whereas the Z8611 uses only one register and part of another.

The 8051 uses two registers for the accumulators; the Z8611 is not limited by this restriction. The 8051 also uses two registers for the serial communication interface, whereas the Z8611 accomplishes the same job with one register. Another two registers in the 8051 are used for data pointers; these are not necessary in the Z8611 since any register can be used as an address pointer.

The Z8611 uses registers more efficiently than either the MC6801 or the 8051. The registers saved by this optimal design are used to perform the functions needed for enhanced interrupt handling and for register pointing with short addresses. The Z8611 also supplies the extra register required for the external stack. These features are not available on the 8051 or the MC6801.

**External Memory.** All three chips can access external memory. The Z8611 and the 8051 can generate signals used for selecting either program or data memory. The Data Memory strobe (the signal used for selecting data or program memory) gives the Z8611 access to 120K bytes of external memory (60K bytes in each program and data memory). The 8051 can use 124K bytes of external memory (64K bytes of external data memory and 60K bytes of external program memory). The MC6801 can access only 62K bytes of external memory and does not distinguish between program and data memory. Thus, the Z8611 and the 8051 are clearly able to access more external memory than the MC6801.

---

## Architectural Overview (Continued)

**On-Chip Peripheral Function.** In addition to the CPU and memory spaces, all chips provide an interrupt system and extensive I/O facilities including I/O pins, parallel I/O ports, a bidirectional address/data bus, and a serial port.

**Interrupts.** The Z8611 acknowledges interrupts from eight sources, four are external from pins IRQ<sub>0</sub>-IRQ<sub>3</sub>, and four are internal from serial-in, serial-out, and the two counter/timers. All interrupts are maskable, and a wide variety of priorities are realized with the Interrupt Mask Register and the Interrupt Priority Registers (see Table 1). All Z8611 interrupts are vectored, with six vectors located in the on-chip ROM. The vectors are fixed locations, two bytes long, that contain the memory address of the service routine.

The 8051 acknowledges interrupts from five sources: two external sources (from INTO and INT1) and three internal sources (one from each of the internal counters and one from the serial I/O port). All interrupts can be disabled individually or globally. Each of the five sources can be assigned one of two priorities: high or low. All 8051 interrupts are vectored. There are five fixed locations in memory, each eight bytes long, allocated to servicing the interrupt.

The MC6801 has one external interrupt, one nonmaskable interrupt, an internal interrupt request, and a software interrupt. The internal interrupts are caused by the serial I/O port, timer overflow, timer output compare, and timer input capture. The priority of each interrupt is preset and cannot be changed. The external interrupt can be masked in the Condition Code register. The MC6801 vectors the interrupts to seven fixed addresses in ROM where the 16-bit address of the service routine is located.

When an interrupt occurs in the 8051, only the Program Counter is saved; the user must save the flags, accumulator, and any registers that the interrupt service routine might affect. The MC6801 saves the Program Counter, accumulators, Index register, and the PSW; the user must save all registers that the interrupt service routine might affect. The Z8611 saves the Program Counter and the Flags register. To save the 16 working

registers, only the Registers Pointer register need be pushed onto the stack and another set of working registers is used for the service routine.

With regard to interrupts, the Z8611 is clearly superior. The Z8611 requires only one command to save all the working registers, which greatly increases the efficiency of context switching.

**I/O Facilities.** The Z8611 has 32 lines dedicated to I/O functions. These lines are grouped into four ports with eight lines per port. The ports can be configured individually under software control to provide input, output, multiplexed address/data lines, timing, and status. Input and output can be serial or parallel, with or without handshake. One port can be configured for serial transmission and four ports can be configured for parallel transmission. With parallel transmission, ports 0, 1, and 2 can transmit data with the handshake provided by port 3.

The 8051 also has 32 I/O lines grouped together into four ports of eight lines each. The ports can be configured under program control for parallel or serial I/O. The ports can also be configured for multiplexed address/data lines, timing, and status. Handshake is provided by user software.

The MC6801 has 29 lines for I/O (three 8-bit ports and one 5-bit port). One port has two lines for handshake. The ports provide all the signals needed to control input and output either serially or in parallel, with or without multiplexed address/data lines. They can be used to interface with external memory.

The main differences in I/O facilities are the number of 8-bit ports and the hardware handshake. The Z8611 and the 8051 have four 8-bit ports, whereas the MC6801 has three 8-bit ports and an additional 5-bit port. The Z8611 has hardware handshake on three ports, the MC6801 has hardware handshake on only one port, and the 8051 has no hardware handshake.

**Counter/Timers.** The Z8611 has two 8-bit counters and two 6-bit programmable prescalars. One prescaler can be driven internally or externally; the other prescaler is driven internally only. Both timers can

---

## Architectural Overview (Continued)

interrupt the CPU when counting is completed. The counters can operate in one of two modes: they can count down until interrupted, or they can count down, reload the initial value, and start counting down again (continuously). The counters for the Z8611 can be used for measuring time intervals and pulse widths, counting events, or generating periodic interrupts.

The 8051 has two 16-bit counter/timers for measuring time intervals and pulse widths, generating pulse widths, counting events, and generating periodic interrupts. The counter/timers have several modes of operation. They can be used as 8-bit counters or timers with two 5-bit programmable prescalers. They can also be used as 16-bit counter/timers. Finally, they can be set as 8-bit modulo-n counters with the reload value held in the high byte of the 16-bit register. An interrupt is generated when the counter/timer has completed counting.

The MC6801 has one 16-bit counter which can be used for pulse-width measurement and generation. The counter/timer actually consists of three 16-bit registers and an 8-bit control/status register. The timer has an input capture register, an output compare register, and a free-running counter. All three 16-bit registers can generate interrupts.

**Serial Communications Interface.** The Z8611 has a programmable serial communication interface. The chip contains a UART for full-duplex, asynchronous, serial receiver/transmitter operation. The bit rate is controlled by counter/timer 0 and has a maximum bit rate of 93,500 b/s. An interrupt is generated when an assembled character is transferred to the receive buffer. The transmitted character generates a separate interrupt. The receive register is double-buffered. A hardware parity generator and

detector are integrated.

The 8501 handles serial I/O using one of its parallel ports. The 8051 bit rate is controlled by counter/timer 1 and has a maximum bit rate of 187,500 b/s. The 8051 generates one interrupt for both transmission and receipt. The receive register is double-buffered.

The MC6801 contains a full-duplex, asynchronous, serial communication interface. The bit rate is controlled by a rate register and by the MCU's clock or an external clock. The maximum bit rate is 62,500 b/s. Both the transmit and the receive registers are double-buffered. The MC6801 generates only one interrupt for both transmit and receive operations. No hardware parity generation or detection is available, although it does have automatic detection of framing errors and overrun conditions.

The 8051 and the MC6801 generate only one interrupt for both transmit and receive, whereas the Z8611 has a separate interrupt for each. The ability to generate separate interrupts greatly enhances the use of serial communications, since separate service routines are often required for transmitting and receiving.

Other differences between the Z8611, MC6801, and the 8051 occur in the hardware parity detector, the double-buffering of registers, framing error detectors and overrun conditions. The 8051 has a faster data rate than either the Z8611 or the MC6801. The MC6801 has the advantage of a hardware framing error detector and automatic detection of overrun conditions. The MC6801 also has both its transmit and receive registers double-buffered. The Z8611 has hardware parity detector, and for detection of framing errors and overrun conditions, a simple, low-overhead software check is possible using only two instructions.

---

## Instruction Architecture

The architecture of the Z8611 is designed specifically for microcomputer applications. This fact is manifest in the instruction composition. The arduous task of programming the MC6801 and the 8051 starkly contrasts with programming the Z8611.

**Addressing Modes.** The Z8611 and the 8051 both have six addressing modes: Register, Indirect Register, Indexed, Direct, Relative, and Immediate. The MC6801 has five addressing modes: Accumulator, Indexed, Direct, Relative, and Immediate. A quick comparison of these addressing modes

---

## Instruction Architecture (Continued)

reveals the versatility of the Z8611 and the 8051. The addressing modes of the MC6801 have several restrictions, as shown in Table 1. While the 8051 has all the addressing modes of the Z8611, its use of them is restricted. The Z8611 allows many more combinations of addressing modes per instruction, because any of its registers can be used as an accumulator. For example, the instructions to clear, complement, rotate, and swap nibbles are all accumulator oriented in the 8051 and operate on the accumulator only. These same commands in the Z8611 can use any register and access it either directly, with register addressing, or with indirect register addressing.

**Indexed Addressing.** All three chips differ in their handling of indexing. The Z8611 can use any register for indexing. The 8051 can use only the accumulator as an Index register in conjunction with the data pointer or the Program Counter. The MC6801 has one 16-bit Index register. The address located in the second byte of an instruction is added to the lower byte of the Index register. The carry is added to the upper byte for the complete address. The MC6801 requires the index value to be an immediate value.

The MC6801 has only one 16-bit Index register and an immediate 8-bit value from the second byte of the instruction. Hence, the Indexed mode of the MC6801 is much more restrictive than that of the Z8611. The 8051 must use the accumulator as its only Index register, loading the accumulator with the register, address each time a reference is made. Then, using indexing, the data is moved into the accumulator, eradicating the previous index. This forces a stream of data through the accumulator and requires a reload of the index before access can be made again. The Z8611 is clearly superior to both the MC6801 and the 8051 in the flexibility of its indexed addressing mode.

**Short and Long Addressing.** Short addressing helps to optimize memory space and execution speed. In sample applications of short register addressing, an eight percent decrease in the number of bytes used was recorded.

All three chips have short addressing modes, but the Z8611 has short addressing for both external memory and register memory. The 8051 has short addressing for the lowest 32 registers only.

The Z8611 has two different modes for register addressing. The full-byte address can be used to provide the address, or a 4-bit address can be used with the Register Pointer. To use the working registers, the Register Pointer is set for a particular bank of 16 registers, and then one of the 16 registers is addressed with four bits. Another feature for addressing external memory is the use of a 12-bit address in place of a full 16-bit address. To use the 12-bit address, one port supplies the eight multiplexed address/data lines and another port supplies four bits for the address. The remaining four bits of the second port can be used for I/O. This feature allows access to a maximum of 10K bytes of memory.

The 8051 uses short addresses by organizing its lowest 32 registers into four banks. The bank select is located in a 2-bit field in the PSW, with three bits addressing the register in the bank.

The MC6801 used extended addressing for addressing external memory. With a special, nonmultiplexed expansion mode, 256 bytes of external memory can be accessed without the need for an external address latch. The MC6801 consumes one 8-bit port for the address and another port for the data.

**Stacks.** The Z8611 and the MC6801 provide for external stacks, which require a 16-bit Stack Pointer. Internal stacks use an 8-bit Stack Pointer. The 8051 uses only a limited internal stack requiring an 8-bit Stack Pointer. Using an external stack saves the internal RAM register for general-purpose use.

**Summary.** The stack structure of the Z8611 and the MC6801 is better than that of the 8051. In most applications, the 8051 is more flexible and easier to program than the MC6801. The Z8611 is easier to use than either the 8051 or the MC6801 because of its register flexibility and its numerous combinations of addressing modes. The 8051 features a unique 4 $\mu$ s multiply and divide

---

## Instruction Architecture (Continued)

command. The MC6801 has a multiply, but it takes  $10\mu\text{s}$  to perform it.

In summary, the Z8611 has the most flexible addressing modes, the most

advanced indexing capabilities, and superior space-and time-saving abilities with respect to short addressing.

---

## Development Support

All vendors provide development support for their products. This section discusses the different support features, including development chips, software, and modules.

**Chips.** SGS offers an entire family of microcomputer chips for product development and final product. The Z8611 is a single-chip microcomputer with 4K bytes of mask-programmed ROM. For development, two other chips are offered. The Z86E11 4K EPROM and the Z86E21 8K EPROM versions.

Intel offers a similar line of development chips with its 8051 family. The 8031 has no internal ROM and the 8751 has 4K of internal EPROM.

Motorola offers the MC6801, MC6803, MC6803NR, and MC68701. These are all similar except the MC68701 has 2K bytes of EPROM and the MC6801 has 2K bytes of ROM. The MC6803 has no internal ROM and the MC6803NR has neither ROM nor RAM on board.

**Software.** Development software includes assemblers, and conversion programs. All manufacturers offer some or all of these features.

Since the MC6801 is compatible with the 6800, there is no need for a new assembler. The Z8611 and the 8051 both offer assemblers for their products. The Z8611 MACZ8 assembler generates relocatable and absolute object code. MACZ8 also supports high-level control and data statements, such as IF... THEN... ELSE. Intel offers an absolute macroassembler, ASM51, with their product. They also offer a program for converting 8048 code to 8051 code.

**Modules.** The Z8611 development module has two 64-pin development versions of the 40-pin, ROM-masked Z8611. Intel offers the EM-51 emulation board, which contains a modified 8051 and PROM or EPROM in place of memory. Motorola has the MEX6801EVM evaluation board for program development.

---

## Additional Features

Additional features include Power Down mode, selftesting, and family-compatibility.

**Power Down Mode.** All three microcomputers offer a Power Down mode. The Z8611 and the 8051 save all of their registers with an auxiliary power supply. The MC6801 uses an auxiliary power supply to save only the first 64 bytes of its register file.

The Z8611 uses one of the crystal input pins for the external power supply to power the registers in Power Down mode. Since the XTAL2 input must be used, an external clock generator is necessary and is input via

XTAL1. The 8051 and the MC6801 both have an input reserved for this function. The MC6801 uses the  $V_{CC}$  standby pin, and the 8051 uses the  $V_{pd}$  pin.

**Family Compatibility.** Another strength of the Z8611 is its expansion bus, which is completely with the Z8000 Family Z-BUS. This means that all Z-BUS peripherals can be used directly with the Z8611.

The MC6801 is fully compatible with all MC6800 family products. The 8051 is software compatible with the older 8048 series and all others in that family.

## Benchmark

The following benchmark tests were used in this report to compare the Z8611, 8051, and MC6801:

- Generate CRC check for 16-bit word.
- Search for a character in a block of memory.
- Execute a computed GOTO - jump to one of eight locations depending on which of the eight bits in set.
- Shift a 16-word five places to the right.
- Move a 64-byte block of data from external memory to the register file.
- Toggle a single bit on a port.
- Measure the subroutine overhead time.

These programs were selected because of their importance in microcomputer applications. Algorithms that reflect a unique function or feature were excluded for the sake of comparison. Although programs can be optimized for a particular chip and for a particular attribute (code density or speed) these programs were not.

The figures cited in this text are taken directly from the vendor's documentation. Therefore, the cycles given below for the MC6801 and the 8051 are in machine cycles and the Z8611 figures are given in clock cycles. The Z8611 clock cycles should be divided by six to give the instruction time in microseconds. The 8051 and MC6801 machine cycle is  $1\mu s$ , and the Z8611 clock cycle is  $.166\mu s$  at 12MHz.

Because of the lack of availability of the MC6801 and the 8051, the benchmark programs listed here have not yet been run. When these products are readily available, the programs will be run and later editions of this document will reflect any changes in the findings.

## Program Listing

		CRC Generation	
<b>8051</b>		<b>Machine Cycles</b>	<b>Bytes</b>
	MOV INDEX, #8	1	2
LOOP:	MOV A,DATA	1	2
	XRL A,HCHECK	1	2
	RLC A	1	1
	MOV A,LCHECK	1	2
	XRL A,LPOLY	1	2
	RLC A	1	1
	MOV LCHECK,A	1	2
	MOV A,HCHECK	1	2
	XRL A,HPOLY	1	2
	RCL A	1	1
	MOV HCHECK,A	1	2
	CLR C	1	1
	MOV A,DATA	1	2
	RCL A	1	1
	MOV DATA,A	1	2
	DJNZ INDEX,LOOP	2	3
	RET	2	1
N = 3 + 17X8 = 139 cycles @12 MHz = 139 $\mu s$ Instructions = 18 Bytes = 31			
<b>MC6801</b>		<b>Machine Cycles</b>	<b>Bytes</b>
	LDAA $\mu 808$	2	2
LOOP:	STAA COUNT	3	2
	LDAA HCHECK	3	2
	EORA DATA	3	2
	ROLA	2	1
	LDAD POLY	4	2
	EORA HCHECK	3	2
	EORB LCHECK	3	2
	ROLB	2	1
	ROLA	2	1
	STAD LCHECK	4	2
	ASL DATA	6	3
	DEC COUNT	6	3
	BNE LOOP	4	2
	RTS	5	1
N = 45X8 + 7 = 367 cycles @4 MHz = 367 $\mu s$ Instructions = 15 Bytes = 28			
<b>Z8611</b>		<b>Clock Cycles</b>	<b>Bytes</b>
	LD INDEX, #8	6	2
LOOP:	LD R6,DATA	6	2
	XOR R6,HCHECK	6	2
	RLC R6	6	2
	XOR LCHECK,LPOLY	6	2
	RLC LCHECK	6	2
	XOR HCHECK,HPOLY	6	2
	RLC HCHECK	6	2
	RCF	6	1
	RLC DATA	6	2
	DJNZ INDEX,LOOP	12or10	2
	RET	14	1
N = 20 + 66X7 + 64 = 546 cycles @12 MHz = 91 $\mu s$ Instructions = 12 Bytes = 22			

**Benchmarks (Continued)**

Character Search Through Block of 40 Bytes			
<b>8051</b>		<b>Machine</b>	
		<b>Cycles</b>	<b>Bytes</b>
	MOV INDEX, #41	1	2
	MOV DPTR, #TABLE	2	3
LOOP1:	DJNZ INDEX, LOOP2	2	2
	SMP OUT	2	2
LOOP2:	MOV A, INDEX	1	2
	MOVC A, @A+DPTR	2	1
	CJNE A, CHARAC, LOOP1	2	3
OUT:			
	N = 3 + 39X7 + 4 = 280 cycles		
	@12 MHz = 280µs		
	Instructions = 7		
	Bytes = 15		
<b>MC6801</b>		<b>Machine</b>	
		<b>Cycles</b>	<b>Bytes</b>
	LDAB # \$40	2	2
	LDAA #CHARAC	2	2
	LDX #TABLE	3	3
LOOP:	CMPA \$0,X	4	2
	BEQ OUT	4	2
	INX	3	1
	DECB	2	1
	BNE LOOP	4	2
OUT:	—		
	—		
	—		
	N = 7 + 40X7 = 687 cycles		
	@4 MHz = 687µs		
	Instructions = 8		
	Bytes = 15		
<b>Z8611</b>		<b>Clock</b>	
		<b>Cycles</b>	<b>Bytes</b>
	LD INDEX, #40	6	2
LOOP:	LD DATA, TABLE (INDEX)	10	3
	CP DATA, CHARAC	6	2
	JR Z, OUT	12 or 10	2
	DJNZ INDEX, LOOP	12 or 10	2
OUT:	—		
	—		
	N = 6 + 38X40 = 1524 cycles		
	@12 MHz = 254µs		
	Instructions = 5		
	Bytes = 11		

Shift 16-Bit Word to Right 5-Bits			
<b>8051</b>		<b>Machine</b>	
		<b>Cycles</b>	<b>Bytes</b>
	MOV INDEX #5	1	2
LOOP:	CLR C	1	1
	MOV A, WORD + 1	1	2
	RRC A	1	1
	MOV WORD + 1, A	1	2
	MOV A, WORK	1	2
	RRC A	1	1
	MOV WORD, A	1	2
	DJNZ INDEX, LOOP	2	2
	N = 1 + 9X5 = 46 Cycles		
	@12 MHz = 46µs		
	Instructions = 9		
	Bytes = 15		
<b>MC6801</b>		<b>Machine</b>	
		<b>Cycles</b>	<b>Bytes</b>
	LDX #5	6	3
	LDAD WORK	4	2
LOOP:	LSRD	3	1
	DEX	3	1
	BNE LOOP	4	2
	STAD WORD	4	2
	N = 10X5 + 11 = 61 Cycles		
	@4 MHz = 61µs		
	Instructions = 6		
	Bytes = 11		
<b>Z8611</b>		<b>Clock</b>	
		<b>Cycles</b>	<b>Bytes</b>
	LD INDEX, #5	6	2
LOOP:	CCF	6	1
	RRC WORD + 1	6	2
	RRC WORD	6	2
	DJNZ INDEX, LOOP	12 or 10	2
	N = 6 + 4X30 + 28 = 154 Cycles		
	@12 MHz = 26µs		
	Instructions = 5		
	Bytes = 9		

## Benchmarks (Continued)

Computed GOTO			
<b>8051</b>		<b>Machine</b>	
		<b>Cycles</b>	<b>Bytes</b>
	MOV INDEX,@40	1	2
LOOP:	MOV A,DATA	1	2
	RLC A	1	1
	JC OUT	2	2
	MOV A,INDEX	1	1
	ADD A,#3	1	2
	MOV INDEX,A	1	1
	SMP LOOP	2	2
OUT:	MOV DPTR,#TABLE	2	3
	MOV A,INDEX	1	1
	JMP @A + DPTR	2	1
TABLE:	LCALL ADDR1		3
	—		
	LCALL ADDR2	2	
	N = 1 + 9X7 + 11 = 75 Cycles		
	@12 MHz = 75µs		
	Instructions = 12		
	Bytes = 21		
<b>MC6801</b>		<b>Machine</b>	
		<b>Cycles</b>	<b>Bytes</b>
	LDAB #2	2	2
	LDX TABLE	3	3
LOOP:	RORA	2	1
	BCS OUT	4	2
	ABX	3	1
	JMP LOOP	3	2
OUT:	LDX O,X	5	3
	JMP O,X	4	3
	N = 8X12 + 14 = 110 Cycles		
	@4 MHz = 110µs		
	Instructions = 8		
	Bytes = 17		
<b>Z8611</b>		<b>Clock</b>	
		<b>Cycles</b>	<b>Bytes</b>
	CLR INDEX	6	2
LOOP:	INC INDEX	6	1
	RLC DATA	6	2
	JR NC,LOOP	12 or 10	2
	LD ADDR, TABLE 1,(INDEX)	10	3
	LD ADDR+1, TABLE 2,(INDEX)	10	3
	JP @ADDR	12	2
	N = 6 + 24X7 + 54 = 228 Cycles		
	@12 MHz = 38µs		
	Instructions = 7		
	Bytes = 15		

Move 64-Byte Block			
<b>8051</b>		<b>Machine</b>	
		<b>Cycles</b>	<b>Bytes</b>
	MOV INDEX,#COUNT	1	2
LOOP:	MOV DPTR,#ADDR1	2	3
	MOVX A,@DPTR	2	1
	INC #ADDR1	1	1
	MOV @ADDR2,A	1	1
	INC ADDR2	1	1
	DJNZ INDEX,LOOP	2	1
	N = 1 + 9X64 = 577 Cycles		
	@12 MHz = 577µs		
	Instructions = 7		
	Bytes = 10		
<b>MC6801</b>		<b>Machine</b>	
		<b>Cycles</b>	<b>Bytes</b>
	LDAB #COUNT	2	2
LOOP:	LDX ADDR1	4	3
	LDAA O,X	4	2
	INX	3	1
	STAA ADDR1	4	2
	LDX ADDR2	4	3
	STAA O,X	4	2
	INX	3	1
	STX ADDR2	4	2
	DECB	2	1
	BNE LOOP	4	2
	N = 64X36 + 2 = 2306 Cycles		
	@4 MHz = 2306µs		
	Instructions = 11		
	Bytes = 21		
<b>Z8611</b>		<b>Clock</b>	
		<b>Cycles</b>	<b>Bytes</b>
	LD INDEX,#COUNT	6	2
LOOP:	LDEI @ADDR2,@ADDR1	18	2
	DJNZ INDEX,LOOP	12 or 10	2
	N = 6 + 63X30 + 28 = 1924 Cycles		
	@12 MHz = 321µs		
	Instructions = 3		
	Bytes = 6		



**Benchmarks (Continued)**

<b>Toggle a Port Bit</b>			
<b>8051</b>		<b>Machine</b>	
		<b>Cycles</b>	<b>Bytes</b>
XRL PO, #YY		2	3
N = 2 Cycles			
@12 MHz = 2μs			
Instructions = 1			
Bytes = 3			
<b>MC6801</b>		<b>Machine</b>	
		<b>Cycles</b>	<b>Bytes</b>
LDAA PORT0		3	2
EORA #YY		2	2
STAA PORT0		3	2
N = 8 Cycles			
@4 MHz = 8μs			
Instructions = 3			
Bytes = 6			
<b>Z8611</b>		<b>Clock</b>	
		<b>Cycles</b>	<b>Bytes</b>
XOR PORT0#YY		10	2
N = 10 Cycles			
@12 MHz = 1.7μs			
Instructions = 1			
Byte = 2			

<b>Subroutine Call/Return Overhead</b>			
<b>8051</b>		<b>Machine</b>	
		<b>Cycles</b>	<b>Bytes</b>
LCALL SUBR		2	3
---			
---			
---			
SUBR: ---			
---			
---			
RET		2	1
N = 4 Cycles			
@12 MHz = 4μs			
Instructions = 2			
Bytes = 4			
<b>MC6801</b>		<b>Machine</b>	
		<b>Cycles</b>	<b>Bytes</b>
JSR SUBR		9	2
---			
---			
---			
SUBR: ---			
---			
---			
RTS		5	1
N = 14 Cycles			
@4 MHz = 14μs			
Instructions = 2			
Bytes = 3			
<b>Z8611</b>		<b>Clock</b>	
		<b>Cycles</b>	<b>Bytes</b>
CALL @SUBR		20	2
---			
---			
---			
SUBR: ---			
---			
---			
RET		14	1
N = 34 Cycles			
@12 MHz = 5.7μs			
Instructions = 2			
Bytes = 3			

## Benchmarks (Continued),

**Results** Table 2 summarizes the results of this comparison. The relative performance column lists the speeds of the MC6801 and 8051 divided by the Z8611 speeds (12 MHz). The overall performance averages the separate relative performances. The higher the number, the faster the Z8611 as compared to the MC6801 and the 8051.

The relative performance figures show that

the Z8611 runs 50 percent faster than the 8051 and 250 percent faster than the MC6801. Although speed is not necessarily the most important criterion for selecting a particular product, the Z8611 proves to be an undeniably superior product when speed is added to the advantages of programming ease, code density, and flexibility.

Benchmark Test	MC6801 (4 MHz) cycles time		8051 (12 MHz) cycles time		Z8 (8 MHz) cycles time		Z8 (12 MHz) cycles time		Relative Performance	
									MC6801	8051
CRC Generation	367	367	139	139	546	137	546	91	4.03	1.53
Character Search	687	687	280	280	1524	382	1524	254	2.70	1.10
Computed GOTO	110	110	75	75	228	57	228	38	2.89	1.97
Shift Right 5 Bits	61	61	46	46	154	38	154	26	2.35	1.78
Move 64-Byte block	2306	2306	577	577	1924	481	1924	321	7.18	1.80
Subroutine Overhead	14	14	4	4	34	8.5	34	5.7	2.46	0.70
Toggle a Port Bit	8	8	2	2	10	2.5	10	1.7	4.71	1.18
					Overall Performance				3.76	1.44

Note: All times are given in microseconds

**Table 2. Benchmark Program Result**

	Bytes			Instructions			Time (microseconds)		
	MC6801	8051	Z8611	MC6801	8051	Z8611	MC6801	8051	Z8611
CRC Generation	28	31	22	15	18	12	367	139	91
Character Search	15	15	11	8	7	5	687	280	254
Shift Right 5 Bits	11	15	9	6	9	5	61	46	26
Computed GOTO	17	21	15	8	12	7	110	75	38
Move Block	21	10	6	11	7	3	2306	577	321
Toggle Port Bit	6	3	2	3	1	1	8	2	1.7
Subroutine Call	3	4	3	2	2	2	14	4	5.7

**Table 3. Byte/Instruction/Time Comparison**

---

## Benchmarks (Continued)

**Summary** The hardware of the three chips compared is very similar. The Z8611, however, has several advantages, the most important of which is its interrupt structure. It is more advanced than the interrupt structures of both the 8051 and the MC6801. Other advantages of the Z8611 over either the MC6801 or the 8051 include I/O facilities with parity detection and hardware handshake and a larger amount of internal ROM (the MC6801 has only 2K bytes).

Substantial differences are apparent with regard to software architecture. The addressing modes of the Z8611 are more

flexible than those of either the MC6801 or the 8051. The Z8611 can use byte-saving addressing with working registers, and it has short external addresses for saving I/O lines. It can also provide for an external stack. The register architecture (as opposed to the accumulator architecture) of the Z8611 saves execution time and enhances programming speed by reducing the byte count.

The Z8611 microcomputer stands out as the most powerful chip of the three, and concurrently, it is the easiest to program and configure.

# A Programmer's Guide to the Z8 Microcomputer

## Introduction

The Z8 is the first microcomputer to offer both a highly integrated microcomputer on a single chip and a fully expandable microprocessor for I/O and memory-intensive applications. The Z8 Z8601, the first one, has two timer/counters, a UART, 2K bytes internal ROM, and a 144-byte internal register file including 124 bytes of RAM, 32 bits of I/O, and 16 control and status registers. In addition, the Z8 can address up to 124K bytes of external program and data memory, which can provide full, memory-mapped I/O capability.

This application note describes the

important features of the Z8, with software examples that illustrate its power and ease of use. It is divided into sections by topic; the reader need not read each section sequentially, but may skip around to the sections of current interest.

It is assumed that the reader is familiar with the Z8 and its assembly language, as described in the following documents:

- *Z8 Technical Manual (O.C.: DAZ8TM/2)*
- *Z8 Programming Manual (O.C.: DAZ8PM/2)*

---

## Accessing Register Memory

The Z8 register space consists of four I/O ports, 16 control and status registers, and 124 general-purpose registers. The general-purpose registers are RAM areas typically used for accumulators, pointers, and stack area. This section describes these registers and how they are used. Bit manipulation and stack operations affecting the register space are discussed in Sections 4 and 5, respectively.

**Registers and Register Pairs.** The Z8 supports 8-bit registers and 16-bit register pairs. A register pair consists of an even-numbered register concatenated with the next higher numbered register (00H and 01H, 02H and 03H, ... 7EH and 7FH, F0H and F1H, ... FEH and FFH). A register pair must be addressed by reference to the even-numbered register. For example,

F1H and F2H is not a valid register pair;  
F0H and F1H is a valid register pair,  
addressed by reference to F0H.

Register pairs may be incremented (INCW) and decremented (DECW) and are useful as pointers for accessing program and external data memory. Section 3 discusses the use of register pairs for this purpose.

Any instruction which can reference or modify an 8-bit register can do so to any of the 144 registers in the Z8, regardless of the inherent nature of that register. Thus, I/O ports, control, status, and general-purpose registers may all be accessed and manipulated without the need for special-purpose instructions. Similarly, instructions which reference or modify a 16-bit register pair can do so to any of the valid 72 register pairs. The only exceptions to this rule are:

- The DINZ (decrement and jump if non-zero) instruction may successfully operate on the general-purpose RAM registers (04H-7FH) only.
- Six control registers are write-only registers and therefore, may be modified only by such instructions as LOAD, POP, and CLEAR. Instructions such as OR and AND require that the current contents of the operand be readable and therefore will not function properly on the write-only registers. These registers are the following: *the timer/counter prescaler registers PRE0 and PRE1, the port mode registers P01M, P2M, and P3M, the interrupt priority register IPR.*

## Accessing Register Memory (Continued)

**Register Pointer.** Within the register addressing modes provided by the Z8, a register may be specified by its full 8-bit address (0-7FH, F0H-FFH) or by a short 4-bit address. In the latter case, the register is viewed as one of 16 working registers within a working register group. Such a group must be aligned on a 16-byte boundary and is addressed by Register Pointer RP (FDH). As an example, assume the Register Pointer contains 70H, thus pointing to the working register group from 70H to 7FH. The LD instruction may be used to initialize register 76H to an immediate value in one of two ways:

LD 76H, #1    18-bit register address is given by instruction (3 byte instruction)!

or

LD R6, #1    14-bit working register address is given by instruction; 4-bit working register group address is given by Register Pointer (2 byte instruction)!

The address calculation for the latter case is illustrated in Figure 1. Notice that 4-bit working-register addressing offers code compactness and fast execution compared to its 8-bit counterpart.

To modify the contents of the Register Pointer, the Z8 provides the instruction

SRP #value

Execution of this instruction will load the upper four bits of the Register Pointer; the lower four bits are always set to zero. Although a load instruction such as

LD RP, #value

could be used to perform the same function, SRP provides execution speed (six vs. ten cycles) and code space (two vs. three bytes) advantages over the LD instruction. The instruction

SRP #70H

is used to set the Register Pointer for the above example.

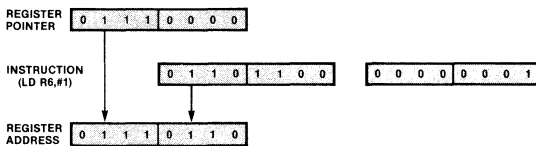


Figure 1. Address Calculation Using the Register Pointer

**Context Switching.** A typical function performed during an interrupt service routine is context switching. Context switching refers to the saving and subsequent restoring of the program counter, status, and registers of the interrupted task. During an interrupt machine cycle, the Z8 automatically saves the Program Counter and status flags on the stack. It is the responsibility of the interrupt service routine to preserve the register space. The recommended means to this end is to allocate a specific portion of the register file for use by the service routine. The service routine thus preserves the register space of the interrupted task by avoiding modification of registers not

allocated as its own. The most efficient scheme with which to implement this function in the Z8 is to allocate a working register group (or portion thereof) to the interrupt service routine. In this way, the preservation of the interrupted task's registers is solely a matter of saving the Register Pointer on entry to the service routine, setting the Register Pointer to its own working register group, and restoring the Register Pointer prior to exiting the service routine. For example, assume such a register allocation scheme has been implemented in which the interrupt service routine for IRQ0 may access only working register Group 4 (registers 40H-4FH). The

## Accessing Register Memory (Continued)

service routine for IRQ0 should be headed by the code sequence:

```
PUSH RP    !preserve Register Pointer of
            interrupted task!
SRP #40H  !address working register
            group 4!
```

Before exiting, the service routine should execute the instruction

```
POP RP
```

to restore the Register Pointer to its entry value.

It should be noted that the technique described above need not be restricted to interrupt service routines. Such a technique might prove efficient for use by a subroutine requiring intermediate registers to produce its outputs. In this way, the calling task can assume that its environment is intact upon return from the subroutine.

**Addressing Mode.** The Z8 provides three addressing modes for accessing the register space: Direct Register, Indirect Register, and Indexed.

*Direct Register Addressing.* This addressing mode is used when the target register address is known at assembly time. Both long (8-bit) register addressing and short (4-bit) working register addressing are supported in this mode. Most instructions supporting this mode provide access to single 8-bit registers. For example:

```
LD FEH, #HI STACK
            !load register FEH (SPH)
            with the upper 8-bits of the
            label STACK!
AND 0, MASK_REG
            !AND register 0 with register
            named MASK_REG!
OR 1, R5  !OR register 1 with working
            register 5!
```

Increment word (INCW) and decrement word (DECW) are the only two Z8 instructions which access 16-bit operands. These instructions are illustrated below for the direct register addressing mode.

```
INCW RRO !increment working register
            pair R0, R1:
            R1 ← R1 + 1
            R0 ← R0 + carry!
```

```
DECW 7EH !decrement working register
            pair 7EH, 7FH:
            7FH ← 7FH - 1
            7EH ← 7EH - carry!
```

Note that the instruction

```
INCW RR5
```

will be flagged as an error by the assembler (RR5 not even-numbered).

*Indirect Register Addressing.* In this addressing mode, the operand is pointed to by the register whose 8-bit register address or 4-bit working register address is given by the instruction. This mode is used when the target register address is not known at assembly time and must be calculated during program execution. For example, assume registers 60H - 7FH contain a buffer for output to the serial line via repetitive calls to procedure SERIAL\_OUT.

SERIAL\_OUT expects working register 0 to hold the output character. The following instructions illustrate the use of the indirect addressing mode to accomplish this task:

```
LD R1, #20H
            !working register 1 is the
            byte counter: output 20H
            bytes!
LD R2, #60H
            !working register 2 is the
            buffer pointer register!
```

out\_again:

```
LD R0, @R2
            !load into working register 0
            the byte pointed to by
            working register 2!
INC R2  !increment pointer!
CALL SERIAL_OUT
            !output the byte!
DJNZ R1, out_again
            !loop till done!
```

Indirect addressing may also be used for accessing a 16-bit register pair via the INCW and DECW instructions. For example.

```
INCW @R0 !increment the register pair
            whose address is contained
            in working register 0!
DECW @7FH
            !decrement the register pair
            whose address is contained
            in register 7FH!
```

---

## Accessing Register Memory (Continued)

The contents of registers R0 and 7FH should be even numbers for proper access; when referencing a register pair, the least significant address bit is forced to the appropriate value by the Z8. However, the register used to point to the register pair need not be an even-numbered register.

Since the indirect addressing mode permits calculation of a target address prior to the desired register access, this mode may be used to simulate other, more complex addressing modes. For example, the instruction

```
SUB 4,BASE(R5)
```

requires the indexed addressing mode which is not directly supported by the Z8 SUBtract instruction. This instruction can be simulated as follows:

```
LD R6,#BASE           !working register 6 has the
                       !base address!
ADD R6,R5             !calculate the target address!
SUB 4,@R6             !now use indirect addressing to
                       !perform the actual
                       !subtract!
```

Any available register or working register may be used in place of R6 in the above example.

*Indexed Addressing.* The indexed addressing mode is supported by the load instruction (LD) for the transference of bytes between a working register and another register. The effective address of the latter register is given by the instruction which is offset by the contents of a designated working (index) register. This addressing mode provides efficient memory usage when addressing consecutive bytes in a block of register memory, such as a table or a buffer. The working register used as the index in the effective address calculation can serve the additional role of counter for control of a loop's duration.

For example, assume an ASCII character buffer exists in register memory starting at address BUF for LENGTH bytes. In order to determine the logical length of the character

string, the buffer should be scanned backward until the first nonoccurrence of a blank character. The following code sequence may be used to accomplish this task:

```
LD R0,#LENGTH        !length of buffer!
                       !starting at buffer end, look
                       !for 1st non-blank!
loop:
LD R1,BUF-1(R0)
CP R1,#''
JR ne,found           !found non-blank!
DINZ R0,loop         !look at next!
all_blanks:         !length = 0!
found:
```

```
5 instructions
12 bytes
1.5  $\mu$ s overhead
10.5  $\mu$ s (average) per character tested
```

At labels "all\_blanks" and "found," R0 contains the length of the character string. These labels may refer to the same location, but they are shown separately for an application where special processing is required for a string of zero length. To perform this task without indexed addressing would require a code sequence such as:

```
LD R1,#BUF+LENGTH-1
LD R0,#LENGTH        !starting at buffer end, look
                       !for 1st non-blank!
loop:
CP @R1,#''
JR ne,found         !found non-blank!
DEC R1             !dec pointer!
DINZ R0,loop      !are we done?!
all_blanks:       !length = 0!
found:
6 instructions
13 bytes
3  $\mu$ s overhead
9.5  $\mu$ s (average) per character tested
```

---

---

## Accessing Register Memory (Continued)

The latter method requires one more byte of program memory than the former, but is faster by four execution cycles (1  $\mu$ s) per character tested.

As an alternate example, assume a buffer exists as described above, but it is desired to scan this buffer forward for the first occurrence of an ASCII carriage return. The following illustrates the code to do this:

```
LD R0, #-LENGTH
      !starting at buffer start, look
      for 1st carriage return
      (= 0DH)!
```

```
next:
LD r1, BUF + LENGTH(R0)
CP R1, #0DH
JR eq, cr !found it!
INC R0 !update counter/index!
JR nz, next
      !try again!
```

```
cr:
ADD R0, #LENGTH
      !R0 has length to CR!
```

```
7 instructions
16 bytes
1.5  $\mu$ s overhead
12  $\mu$ s (average) per character tested
```

---

## Accessing Program and External Data Memory

In a single instruction, the Z8 can transfer a byte between register memory and either program or external data memory. Load Constant (LDC) and Load Constant and Increment (LDCI) reference program memory; Load External (LDE) and Load External and Increment (LDEI) reference external data memory. These instructions require that a working register pair contain the address of the byte in either program or external data memory to be accessed by the instruction (indirect working register pair addressing mode). The register byte operand is specified by using the direct working register addressing mode in LDC and LDE or the indirect working register addressing mode in LDCI and LDEI. In addition to performing the designated byte transfer, LDCI and LDEI automatically increment both the indirect registers specified by the instruction. These instructions are therefore efficient for performing block moves between register and either program or external data memory. Since the indirect addressing mode is used to specify the operand address within program or external data memory, more complex addressing modes may be simulated as discussed earlier in Section 2.4.2. For example, the instruction

```
LDC R3, BASE(R2)
```

requires the indexed addressing mode, where BASE is the base address of a table in program memory and R2 contains the offset from table start to the desired table entry.

The following code sequence simulates this instruction with the use of two additional registers (R0 and R1 in this example).

```
LD R0, #.H BASE
LD R1, #.L BASE
      !RRO has table start address!
ADD R1, R2
ADC R0, #0
      !RRO has table entry address!
LDC R3, @R0
      !R3 has the table entry!
```

## Configuring the Z8 for I/O Applications vs.

**Memory Intensive Applications.** The Z8 offers a high degree of flexibility in memory and I/O intensive applications. Thirty-two port bits are provided of which 16, 12, eight, or zero may be configured as address bits to external memory. This allows for addressing of 62K, 4K or 256 bytes of external memory, which can be expanded to 124K, 8K, or 512 bytes if the Data Memory Select output ( $\overline{DM}$ ) is used to distinguish between program and data memory accesses. The following instructions illustrate the code sequence required to configure the Z8 with 12 external addressing lines and to enable the Data Memory Select output.

```
LD P01M, #(2)00010010H
      !bit 3-4: enable AD0-AD7;
      bit 0-1: enable A8-A11!
LD P3M, #(2)00001000H
      !bit 3-4: enable  $\overline{DM}$ !
```



## Accessing Program and External Data Memory (Continued)

The two bytes following the mode selection of ports 0 and 1-should not reference external memory due to pipelining of instructions within the Z8. Note that the load instruction to P3M satisfies this requirement (providing that it resides within the internal 2K bytes of memory).

**LDC and LDE.** To illustrate the use of the Load Constant (LDC) and Load External (LDE) instructions, assume there exists a hardware configuration with external memory and Data Memory Select enabled. The following module illustrates a program for

tokenizing an ASCII input buffer. The program assumes there is a list of delimiters (space, comma, tab, etc.) in program memory at address DELIM for COUNT bytes (accessed via LDC) and that an ASCII input buffer exists in external data memory (accessed via LDE). The program scans the input buffer from the current location and returns the start address of the next token (i.e. the address of the first nondelimiter found) and the length of that token (number of characters from token start to next delimiter).

```

MACZ8      2.0
LOC      OBJ CODE STMT SOURCE STATEMENT

          1 SCAN      MODULE
          2 CONSTANT
          3 COUNT    :=      6
          4 GLOBAL
          5           $SECTION PROGRAM
P 0000 20 3B 2C     6 DELIM  ARRAY  [COUNT BYTE]  :=
P 0003 2E 0A 0D
          7           [' ', ',', ' ', ' ', ' ', 0AH, 0DH]
          8
P 0006          9 scan      PROCEDURE
          10 !*****
          11 Purpose=      To find the next token within an
          12                ASCII buffer.
          13
          14 Input=       RR0 = address of current location
          15                within input buffer in external
          16                memory.
          17
          18 Output=      RR4 = address of start of next token
          19                RR0 = address of new token's ending
          20                delimiter
          21                R2 = length of token
          22                R3 = ending delimiter
          23                R6, R7, R8, R9 destroyed
          24
          25 ***** !
          26 ENTRY
P 0006 B0 E2       27                clr      R2          !init. length counter!
          28                D0
P 0008 82 30       29                LDE     R3,@RR0    !get byte from input buffer!
P 000A A0 E0       30                incw   RR0          !increment pointer!
P 000C D6 002E'    31                call   check      !look for non-delimiter!
P 000F FD 0015'   32                IF   C THEN
P 0012 8D 0018'   33                EXIT          !found token start!
          34                FI
P 0015 8D 0008'   35                OD
          36
P 0018 48 E0       37                ld      R4,R0
P 001A 58 E1       38                ld      R5,R1    !RR4 = token starting addr!
          39                D0
P 001C 2E         40                inc     R2          !inc. length counter!
P 001D 82 30      41                LDE     R3,@RR0    !get next input byte!

```

## Accessing Program and External Data Memory (Continued)

```

P 001F D6 002E' 42          call  check      !look for delimiter!
P 0022 7D 0028' 43          IF  NC  THEN
P 0025 8D 002D' 44          EXIT              !found token end!
                                     45          FI
P 0028 A0 E0      46          incw   RR0      !point to next byte!
P 002A 8D 001C' 47          OD
                                     48
P 002D AF        49          ret
P 002E          50 END      scan
                                     51
P 002E          52 check   PROCEDURE
53 !.....
54 Purpose =          compare current character with
55                  delimiter table until table
56                  end or match found
57
58 input =           DELIM  = start address of table
59                  COUNT  = length of that table
60                  R3     = byte to be scrutinized
61
62 output =         Carry flag = 1 => input byte
63                  is not a delimiter (no match found)
64
65                  Carry flag = 0 => input byte
66                  is a delimiter (match found)
67                  R6, R7, R8, R9 destroyed
68
69 !.....!
70 ENTRY
P 002E 6C 00*    71          ld          R6, #HI DELIM  !
P 0030 7C 00*    72          ld          R7, #LO DELIM  !RR6 points to
73                  delimiter list!
P 0032 8C 06     74          ld          R8, #COUNT !R8 = length of list!
75 here:
P 0034 C2 96     76          LDC         R9, @RR6      !get table entry!
P 0036 A0 E6     77          incw         RR6              !point to next entry!
P 0038 A2 93     78          cp          R9, R3      !R3 = delimiter?!
P 003A 6B 03     79          jr          eq, byte  !yes, carry = 0!
P 003C 8A F6     80          djnz        R8, here  !next entry!
P 003E DF        81          scf          !table done. R3
82                  not a delimiter!
83 bye:
P 003F AF        84          ret
P 0040          85 END      check
86 END          86 END      SCAN

```

0 ERRORS  
ASSEMBLY COMPLETE

27 instructions

58 bytes

Execution time is a function of the number of leading delimiters  
before token start (x) and the number of characters in the  
token (y):  $123 \mu\text{s overhead} + 59 \times \mu\text{s } 2 \text{ } 102y \mu\text{s}$   
(average) per token

---

## Accessing Program and External Data Memory (Continued)

**LDCI.** A common function performed in Z8 applications is the initialization of the register space. The most obvious approach to this function is the coding of a sequence of "load register with immediate value" instructions (each occupying three program bytes for a register or two program bytes for a working register). This approach is also the most efficient technique for initializing less than eight consecutive registers or 14 consecutive working registers. For a larger register block, the LDCI instruction provides an economical means of initializing consecutive registers from an initialization table in program memory. The following code excerpt illustrates this technique of initializing control registers F2H through FFH from a 14-byte array (INIT\_\_tab) in program memory:

```
SRP #00H          !RP not F0H!
LD R6, #.H INIT__tab
LD R7, #.L INIT__tab
LD R8, #F2H       !1st reg to be initialized!
LD R9, #14        !length of register block!
loop:
LDCI @R8, @RR6   !load a register from the init table!
DJNZ R9, loop    !continue till done!
```

7 instructions  
14 bytes  
7.5  $\mu$ s overhead  
7.5  $\mu$ s per register initialized

**LDEI.** The LDEI instruction is useful for moving blocks of data between external and register memory since auto-increment is performed on both indirect registers designated by the instruction. The following code excerpt illustrates a register buffer being saved at address 40H through 60H into external memory at address SAVE:

```
LD R10, #.H SAVE !external memory!
LD R11, #.L SAVE !address!
LD R8, #40H      !starting register!
LD R9, #21H      !number of registers to save in external data memory!
loop:
LDEI @RR10, @R8 !init a register!
DJNZ R9, loop    !until done!
6 instructions
12 bytes
6  $\mu$ s overhead
7.5  $\mu$ s per register saved
```

---

## Bit Manipulations

Support of the test and modification of an individual bit or group of bits is required by most software applications suited to the Z8 microcomputer. Initializing and modifying the Z8 control registers, polling interrupt requests, manipulating port bits for control of or communication with attached devices, and manipulation of software flags for internal control purposes are all examples of the heavy use of bit manipulation functions. These examples illustrate the need for such functions in all areas of the Z8 register space. These functions are supported in the Z8 primarily by six instructions:

- Test under Mask (TM)
- Test Complement under Mask (TCM)

- AND
- OR
- XOR
- Complement (COM)

These instructions may access any Z8 register, regardless of its inherent type (control, I/O, or general purpose), with the exception of the six write-only control registers (PRE0, PRE1, P01M, P2M, P3M, IPR) mentioned earlier in Section 2.1. Table 1 summarizes the function performed on the destination byte by each of the above instructions. All of these instructions, with the exception of COM, require a mask

## Bit Manipulations (Continued)

operand. The "selected" bits referenced in Table 1 are those bits in the destination operand for which the corresponding mask bit is a logic 1.

Opcode	Use
TM	To test selected bits for logic 0
TCM	To test selected bits for logic 1
AND	To reset all but selected bits to logic 0
OR	To set selected bits to logic 1
XOR	To complement selected bits
COM	To complement all bits

**Table 1. Bit Manipulation Instruction Usage**

The instructions AND, OR, XOR, and COM have functions common to today's micro-processors and therefore are not described in depth here. However, examples of the use of these instructions are laced throughout the remainder of this document, thus giving an integrated view of their uses in common functions. Since they are unique to the Z8, the functions of Test under Mask and Test Complement under Mask, are discussed in more detail next.

**Test under Mask (TM).** The Test under Mask instruction is used to test selected bits for logic 0. The logical operation performed is

destination AND source

Neither source nor destination operand is modified; the FLAGS control register is the only register affected by this instruction. The zero flag (Z) is set if all selected bits are logic 0; it is reset otherwise. Thus, if the selected destination bits are either all logic 1 or a combination of 1s and 0s, the zero flag would be cleared by this instruction. The sign flag (S) is either set or reset to reflect the result of the AND operation; the overflow flag (V) is always reset. All other flags are unaffected. Table 2 illustrates the flag settings which result from the TM instruction on a variety of source and destination

operand combinations. Note that a given TM instruction will never result in both the Z and S flags being set.

**Test Complement under Mask.** The Test Complement under Mask instruction is used to test selected bits for logic 1. The logical operation performed is

(NOT destination) AND source.

As in Test under Mask, the FLAGS control register is the only register affected by this operation. The zero flag (Z) is set if all selected destination bits are 1; it is reset otherwise. The sign flag (S) is set or reset to reflect the result of the AND operation; the overflow flag (V) is always reset. Table 3 illustrates the flag settings which result from the TCM instruction on a variety of source and destination operand combinations. As with the TM instruction, a given TCM instruction will never result in both the Z and S flags being set.

Destination	Source	Flags		
		Z	S	V
(binary)	(binary)			
10001100	01110000	1	0	0
01111100	01110000	0	0	0
10001100	11110000	0	1	0
11111100	11110000	0	1	0
00011000	10100001	1	0	0
01000000	10100001	1	0	0

**Table 2. Effects of the TM Instruction**

Destination	Source	Flags		
		Z	S	V
(binary)	(binary)			
10001100	01110000	0	0	0
01111100	01110000	1	0	0
10001100	11110000	0	0	0
11111100	11110000	1	0	0
00011000	10100001	0	1	0
01000000	10100001	0	1	0

**Table 3. Effects of the TCM Instruction**

## Stack Operations

The Z8 stack resides within an area of data memory (internal or external). The current address in the stack is contained in the stack pointer, which decrements as bytes are pushed onto the stack, and increments as bytes are popped from it. The stack pointer occupies two control register bytes (FEH and FFH) in the Z8 register space and may be manipulated like any other register. The stack is useful for subroutine calls, interrupt service routines, and parameter passing and saving. Figure 2 illustrates the downward growth of a stack as bytes are pushed onto it.

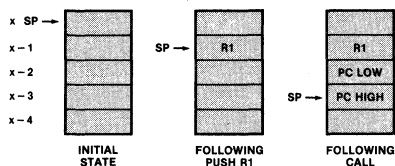


Figure 2. Growth of a Stack

**Internal vs. External Stack.** The location of the stack in data memory may be selected to be either internal register memory or external data memory. Bit 2 of control register P01M (F8H) controls this selection. Register pair SPH (FEH), SPL (FFH) serves as the stack pointer for an external stack. Register SPL is the stack pointer for an internal stack. In the latter configuration, SPH is available for use as a data register. The following illustrates a code sequence that initializes external stack operations:

```
LD P01M, #(2)00000000H
    !bit 2: select external stack!
LD SPH, #.H STACK
LD SPL, #.L STACK
```

**CALL.** A subroutine call causes the current Program Counter (the address of the byte following the CALL instruction) to be pushed onto the stack. The Program Counter is loaded with the address specified by the

CALL instruction. This address may be a direct address or an indirect register pair reference.

For example,

```
LABEL 1: CALL 4F98H
    !direct addressing: PC is
    loaded with the hex value
    4F98;
    address LABEL 1 + 3 is
    pushed onto the stack!
```

```
LABEL 2: CALL @RR4
    !indirect addressing: PC is
    loaded with the contents of
    working register pair R4,
    R5; address LABEL 2 + 2 is
    pushed onto the stack!
```

```
LABEL 3: CALL @7EH
    !indirect addressing: PC is
    loaded with the contents of
    register pair 7EH, 7FH;
    address LABEL 3 + 2 is
    pushed onto the stack!
```

**RET.** The return (RET) instruction causes the top two bytes to be popped from the stack and loaded into the Program Counter. Typically, this is the last instruction of a subroutine and thus restores the PC to the address following the CALL to that subroutine.

**Interrupt Machine Cycle.** During an interrupt machine cycle, the PC followed by the status flags is pushed onto the stack. (A more detailed discussion of interrupt processing is provided in Section 6).

**IRET.** The interrupt return (IRET) instruction causes the top byte to be popped from the stack and loaded into the status flag register. FLAGS (FCH); the next two bytes are then popped and loaded into the Program Counter. In this way, status is restored and program execution continues where it had left off when the interrupt was recognized.

---

## Stack Operations (Continued)

**PUSH and POP.** The PUSH and POP instructions allow the transfer of bytes between the stack and register memory, thus providing program access to the stack for saving and restoring needed values and passing parameters to subroutines.

Execution of a PUSH instruction causes the stack pointer to be decremented by 1; the operand byte is then loaded into the location pointed to by the decremented stack pointer. Execution of a POP instruction causes the byte addressed by the stack pointer to be loaded into the operand byte; the stack pointer is then incremented by 1. In both cases, the operand byte is designated by either a direct register address or an indirect

register reference. For example:

PUSH R1	!direct address: push working register 1 onto the stack!
POP 5	!direct address: pop the top stack byte into register 5!
PUSH @R4	!indirect address: pop the top stack byte into the byte pointed to by working register 4!
PUSH @17	!indirect address: push onto the stack the byte pointed to by register 17!

---

## Interrupts

The Z8 recognizes six different interrupts from four internal and four external sources, including internal timer/counters, serial I/O, and four Port 3 lines. Interrupts may be individually or globally enabled/disabled via Interrupt Mask Register IMR (FBH) and may be prioritized for simultaneous interrupt resolution via Interrupt Priority Register IPR (F9H). When enabled, interrupt request processing automatically vectors to the designated service routine. When disabled, and interrupt request may be polled to determine when processing is needed.

**Interrupt Initialization.** Before the Z8 can recognize interrupts following RESET, some initialization tasks must be performed. The initialization routine should configure the Z8 interrupt requests to be enabled/disabled, as required by the target application and assigned a priority (via IPR) for simultaneous enabled-interrupt resolution. An interrupt request is enabled if the corresponding bit in the IMR is set (= 1) and interrupts are globally enabled (bit 7 of IMR = 1). An interrupt request is disabled if the corresponding bit in the IMR is reset (= 0) or interrupts are globally disabled (bit 7 of IMR = 0).

A RESET of the Z8 causes the contents of the Interrupt Request Register IRQ (FAH) to be held to zero until the execution of an EI

instruction. Interrupts that occur while the Z8 is in this initial state will not be recognized, since the corresponding IRQ bit cannot be set. The EI instruction is specially decoded by the Z8 to enable the IRQ; simply setting bit 7 of IMR is therefore *not* sufficient to enable interrupt processing following RESET. However, subsequent to this initial EI instruction, interrupts may be globally enabled either by the instruction

EI                   !enable interrupts!

or by a register manipulation instruction such as

OR IMR, #80H

To globally disable interrupts, execute the instruction

DI                   !disable interrupts!

This will cause bit 7 of IMR to be reset.

Interrupts *must* be globally disabled prior to any modification of the IMR. IPR or enabled bits of the IRQ (those corresponding to enabled interrupt requests), unless it can be *guaranteed* that an enabled interrupt will not occur during the processing of such instructions. Since interrupts represent the occurrence of events asynchronous to program execution, it is highly unlikely that such a guarantee can be made reliably.

---

## Interrupts (Continued)

**Vectored Interrupt Processing:** Enabled interrupt requests are processed in an automatic vectored mode in which the interrupt service routine address is retrieved from within the first 12 bytes of program memory. When an enabled interrupt request is recognized by the Z8, the Program Counter is pushed onto the stack (low order 8 bits first, then high-order 8 bits) followed by the FLAGS register ( $\#FCH$ ). The corresponding interrupt request bit is reset in IRQ, interrupts are globally disabled (bit 7 of IMR is reset), and an indirect jump is taken on the word in location  $2x, 2x + 1$  ( $x =$  interrupt request number,  $0 \leq x \leq 5$ ). For example, if the bytes at addresses 0004H and 0005H contain 05H and 78H respectively, the interrupt machine cycle for IRQ2 will cause program execution to continue at address 0578H.

When interrupts are sampled, more than one interrupt may be pending. The Interrupt Priority Register (IPR) controls the selection of the pending interrupt with highest priority. While this interrupt is being serviced, a higher-priority interrupt may

occur. Such interrupts may be allowed service within the current interrupt service routine (nested) or may be held until the current service routine is complete (non-nested).

To allow nested interrupt processing, interrupts must be selectively enabled upon entry to an interrupt service routine. Typically, only higher-priority interrupts would be allowed to nest within the current interrupt service. To do this, an interrupt routine must "know" which interrupts have a higher priority than the current interrupt request. Selection of such nesting priorities is usually a reflection of the priorities established in the Interrupt Priority Register (IPR). Given this data, the first instructions executed in the service routine should be to save the current Interrupt Mask Register, mask off all interrupts of lower and equal priority, and globally enable interrupts (EI). For example, assume that service of interrupt requests 4 and 5 are nested within the service of interrupt request 3. The following illustrates the code required to enable IRQ4 and IRQ5:

```
CONSTANT      INT_MASK_3           :=      (2) 00110000H
GLOBAL
IRQ3_service  PROCEDURE           ENTRY
!service routine for IRQ3!
    PUSH IMR                        !save Interrupt Mask Register!
                                !interrupts were globally disabled during the interrupt
                                !machine cycle - no DI is needed prior to modification of IMR!
    AND IMR, #INT_MASK_3           !disable all but IRQ4 & 5!
    EI
    !...!                          !service interrupt!
                                !interrupts are globally enabled now - must disable them prior
                                !to modification of IMR!
    DI
    POP IMR                          !restore entry IMR!
    IRET
END IRQ3_service
```

## Interrupts (Continued)

Note that IRQ4 and IRQ5 are enabled by the above sequence only if their respective IMR bits=1 on entry to IRQ3\_service.

The service routine for an interrupt whose processing is to be completed without interruption should not allow interrupts to be nested within it. Therefore, it need not modify the IMR, since interrupts are disabled automatically during the interrupt machine cycle.

The service routine for an enabled interrupt is typically concluded with an IRET instruction, which restores the FLAGS register and Program Counter from the top of the stack and globally enables interrupts. To return from an interrupt service routine without re-enabling interrupts, the following code sequence could be used:

```
POP FLAGS
      !FLAGS←@SP!
RET      !PC←@SP!
```

This accomplishes all the functions of IRET, except that IMR is not affected.

**Polled Interrupt Processing.** Disabled interrupt requests may be processed in a polled mode, in which the corresponding bits of the Interrupt Request Register (IRQ) are examined by the software. When an interrupt request bit is found to be a logic 1, the interrupt should be processed by the appropriate service routine. During such processing, the interrupt request bit in the IRQ must be cleared by the software in order for subsequent interrupts on that line to be distinguished from the current one. If more than one interrupt request is to be processed in a polled mode, polling should occur in the order of established priorities. For example, assume that IRQ0, IRQ1, and IRQ4 are to be polled and that established priorities are, from high to low, IRQ4, IRQ0, IRQ1. An instruction sequence like the following should be used to poll and service the interrupts:

```
!...!
!poll interrupt inputs here!
      TCM      IRQ, # (2)00010000H      !IRQ4 need service?!
      JR       NZ, TEST0                !no!
      CALL    IRQ4_service              !yes!
TEST0:  TCM      IRQ, # (2)00000001H      !IRQ0 need service?!
      JR       NZ, TEST1                !no!
      CALL    IRQ0_service              !yes!
TEST1:  TCM      IRQ, # (2)00000010H      !IRQ1 need service?!
      JR       NZ, DONE                 !no!
      CALL    IRQ1_service              !yes!
DONE:   !...!

IRQ4_service  PROCEDURE      ENTRY
      !...!
      AND     IRQ, # (2)11101111H      !clear IRQ4!
      !...!
      RET
END IRQ4_service

IRQ0_service  PROCEDURE      ENTRY
      !...!
      AND     IRQ, # (2)11111110H      !clear IRQ0!
      !...!
      RET
END IRQ0_service

IRQ1_service  PROCEDURE      ENTRY
      !...!
      AND     IRQ, # (2)11111101H      !clear IRQ1!
      !...!
      RET
END IRQ1_service
!...!
```



## Timer/Counter Functions

The Z8 provides two 8-bit timer/counters,  $T_0$  and  $T_1$ , which are adaptable to a variety of application needs and thus allow the software (and external hardware) to be relieved of the bulk of such tasks. Included in the set of such uses are:

- Interval delay timer
- Maintenance of a timer-of-day clock
- Watch-dog timer
- External event counting
- Variable pulse train output
- Duration measurement of external event
- Automatical delay following external event detection

Each timer/counter is driven by its own 6-bit prescaler, which is in turn driven by the internal Z8 clock divided by four. For  $T_1$ , the internal clock may be gated or triggered by an external event or may be replaced by an external clock input. Each timer/counter may operate in either single-pass or continuous mode where, at end-of-count, either counting stops of the counter reloads and continues counting. The counter and prescaler registers may be altered individually while the timer/counter is running; the software controls whether the new values are loaded immediately or when end-of-count (EOC) is reached.

Although the timer/counter prescaler registers (PRE0 and PRE1) are write-only, there is a technique by which the timer/counters may simulate a readable prescaler. This capability is a requirement for high resolution measurement of an event's duration. The basic approach requires that one timer/counter be initialized with the desired counter and prescaler values. The second timer/counter is initialized with a counter equal to the prescaler of the first timer/counter and a prescaler of 1. The second timer/counter must be programmed for continuous mode. With both timer/counters driven by the internal clock and started and stopped simultaneously, they will run synchronous to one another; thus, the value read from the second counter will always be equivalent to the prescaler of the first.

**Time/Count Interval Calculation** To determine the time interval ( $i$ ) until EOC, the equation

$$i = t \times p \times v$$

characterizes the relation between the prescaler ( $p$ ), counter ( $v$ ), and clock input period ( $t$ );  $t$  is given by

$$1/(XTAL/8)$$

where XTAL is the Z8 input clock frequency;  $p$  is in the range 1–64;  $v$  is in the range 1–256. When programming the prescaler and counter registers, the maximum load value is truncated to six and eight bits, respectively, and is therefore programmed as zero. For an input clock frequency of 8 MHz, the prescaler and counter register values may be programmed to time and interval in the range

$$1 \mu s \times 1 \times 1 \leq i \leq 1 \mu s \times 64 \times 256$$
$$1 \mu s \leq i \leq 16.384 \text{ ms}$$

To determine the count ( $c$ ) until EOC for  $T_1$  with external clock input, the equation

$$c = p \times v$$

characterizes the relation between the  $T_1$  prescaler ( $p$ ) and the  $T_1$  counter ( $v$ ). The divide-by-8 on the input frequency is bypassed in this mode. The count range is

$$1 \times 1 \leq c \leq 64 \times 256$$
$$1 \leq c \leq 16,384$$

**T<sub>OUT</sub> Modes.** Port 3, bit 6 (P3<sub>6</sub>) may be configured as an output (T<sub>OUT</sub>) which is dynamically controlled by one of the following:

- $T_0$
- $T_1$
- Internal clock

When driven by  $T_0$  or  $T_1$ , T<sub>OUT</sub> is reset to a logic 1 when the corresponding load bit is set in timer control register TMR (F1H) and toggles on EOC from the corresponding counter. When T<sub>OUT</sub> is driven by the internal clock, that clock is directly output on P3<sub>6</sub>.

While programmed as T<sub>OUT</sub>, P3<sub>6</sub> is disabled from being modified by a write to port register 03H; however, its current output may be examined by the Z8 software by a read to port register 03H.

## Timer/Counter Functions (Continued)

**T<sub>1N</sub> Modes.** Port 3, bit 1 (P3<sub>1</sub>) may be configured as an input (T<sub>1N</sub>) which is used in conjunction with T<sub>1</sub> in one of four modes:

- External clock input
- Gate input for internal clock
- Nonretriggerable input for internal clock
- Retriggerable input for internal clock

For the latter two modes, it should be noted that the existence of a synchronizing circuit within the Z8 causes a delay of two to three internal clock periods following an external trigger before clocking of the counter actually begins.

*Each High-to-Low transition on T<sub>1N</sub> will generate interrupt request IRQ2, regardless of the selected T<sub>1N</sub> mode or the enabled/disabled state of T<sub>1</sub>. IRQ2 must therefore be masked or enabled according to the needs of the application.*

The "external clock input" T<sub>1N</sub> mode supports the counting of external events, where an event is seen as a High-to-Low transition on T<sub>1N</sub>. Interrupt request IRQ5 is generated on the n<sup>th</sup> occurrence (single-pass mode) or on every n<sup>th</sup> occurrence (continuous mode) of that event.

The "gate input for internal clock" T<sub>1N</sub> mode provides for duration measurement of an external event. In this mode, the T<sub>1</sub> prescaler is driven by the Z8 internal clock, gated by a High level on T<sub>1N</sub>. In other words, T<sub>1</sub> will count while T<sub>1N</sub> is High and stop counting while T<sub>1N</sub> is Low. Interrupt request IRQ2 is generated on the High-to-Low transition on T<sub>1N</sub>. Interrupt request IRQ5 is generated on T<sub>1</sub> EOC. This mode may be used when the width of a High-going pulse needs to be measured. In this mode, IRQ2 is typically the interrupt request of most importance, since it signals the end of the pulse being measured. If IRQ5 is generated prior to IRQ2 in this mode, the pulse width on T<sub>1N</sub> is too large for T<sub>1</sub> to measure in a single pass.

The "nonretriggerable input" T<sub>1N</sub> mode provides for automatic delay timing following an external event. In this mode, T<sub>1</sub> is loaded and clocked by the Z8 internal clock following the first High-to-Low transition on T<sub>1N</sub> after T<sub>1</sub> is enabled. T<sub>1N</sub> transitions that occur after this point do not affect T<sub>1</sub>. In single-pass mode, the enable bit is reset on

EOC; further T<sub>1N</sub> transitions will not cause T<sub>1</sub> to load and begin counting until the software sets the enable bit again. In continuous mode, EOC does not modify the enable bit, but the counter is reloaded and counting continues immediately; IRQ5 is generated every EOC until software resets the enable bit. This T<sub>1N</sub> mode may be used, for example, to time the line feed delay following end of line detection on a printer or to delay data sampling for some length of time following a sample strobe.

The "retriggerable input" T<sub>1N</sub> mode will load and clock T<sub>1</sub> with the Z8 internal clock on every occurrence of a High-to-Low transition on T<sub>1N</sub>. T<sub>1</sub> will time-out and generate interrupt request IRQ5 when the programmed time interval (determined by T<sub>1</sub> prescaler and load register values) has elapsed since the last High-to-Low transition on T<sub>1N</sub>. In single-pass mode, the enable bit is reset on EOC; further T<sub>1N</sub> transitions will not cause T<sub>1</sub> to load and begin counting until the software sets the enable bit again. In continuous mode, EOC does not modify the enable bit, but the counter is reloaded and counting continues immediately; IRQ5 is generated at every EOC until the software resets the enable bit. This T<sub>1N</sub> mode may provide such functions as watch-dog timer (e.g., interrupt if conveyor belt stopped or clock pulse missed), or keyboard time-out (e.g., interrupt if no input in x ms).

**Examples.** Several possible uses of the timer/counters are given in the following four examples.

*Time of Day Clock.* The following module illustrates the use of T<sub>1</sub> for maintenance of a time of day clock, which is kept in binary format in terms of hours, minutes, seconds, and hundredths of a second. It is desired that the clock be updated once every hundredth of a second; therefore, T<sub>1</sub> is programmed in continuous mode to interrupt 100 times a second. Although T<sub>1</sub> is used for this example, T<sub>0</sub> is equally suited for the task.

The procedure for initializing the timer (TOD\_\_INIT), the interrupt service routine (TOD) which updates the clock, and the interrupt vector for T<sub>1</sub> end-of-count (IRQ\_\_5) are illustrated below. XTAL = 7.3728 MHz is assumed.

## Timer/Counter Functions (Continued)

```

MACZ8      2.0
LOC      OBJ CODE STMT SOURCE STATEMENT
          1 TIMER1    MODULE
          2 CONSTANT
          3 HOUR      :=      R12
          4 MINUTE    :=      R13
          5 SECOND    :=      R14
          6 HUND      :=      R15
          7           $SECTION PROGRAM
          8 GLOBAL
          9 !IRQ5 interrupt vector!
P 0000 000F' 10          $ABS      10
          11 IRQ_5    ARRAY [1 WORD] := [TOD]
          12
          13          $REL
P 000C      14 TOD__INIT    PROCEDURE
          15 ENTRY
P 0000 E6 F3 93 16          LD      PRE1, # (2) 10010011H
          17                                     !bit 2-7: prescaler = 36,
          18                                     bit 1: internal clock;
          19                                     bit 0: continuous mode!
P 0003 E6 F2 00 20          LD      T1, #0      !(256) time-out =
          21                                     1/100 second!
P 0006 46 F1 0C 22          OR      TMR, #0CH  !load, enable T1!
P 0009 8F      23          DI
P 000A 46 FB 20 24          OR      IMR, #20H  !enable T1 interrupt!
P 000D 9F      25          EI
P 000E AF      26          RET
P 000F      27 END      TOD__INIT
          28
P 000F      29 TOD      PROCEDURE
          30 ENTRY
P 000F 70 FD 31          PUSH    RP
          32 !Working register file 10H to 1FH contains
          33 the time of day clock!
P 0011 31 10 34          SRP      #10H
P 0013 FE 35          INC      HUND      !1 more .01 sec!
P 0014 A6 EF 64 36          CP      HUND, #100  !full second yet?!
P 0017 EB 13 37          JR      NE, TOD__EXIT !jump if no!
P 0019 B0 EF 38          CLR      HUND
P 001B EE 39          INC      SECOND    !1 more second!
P 001C A6 EE 3C 40          CP      SECOND, #60  !full minute yet?!
P 001F EB 0B 41          JR      NE, TOD__EXIT !jump if no!
P 0021 B0 EE 42          CLR      SECOND
P 0023 DE 43          INC      MINUTE     !1 more minute!
P 0024 A6 ED 3C 44          CP      MINUTE, #60  !full hour yet?!
P 0027 EB 03 45          JR      NE, TOD__EXIT !jump if no!
P 0029 B0 ED 46          CLR      MINUTE
P 002B CE 47          INC      HOUR
          48 TOD__EXIT:
P 002C 50 FD 49          POP      RP      !restore entry RP!
P 002E BF 50          IRET
P 002F 51 END      TOD
          52 END      TIMER1

```

0 ERRORS  
ASSEMBLY COMPLETE

TOD\_\_INIT:  
7 instructions  
15 bytes  
16  $\mu$ s

TOD:  
17 instruction  
32 bytes  
19.5  $\mu$ s (average) including interrupt response time

## Timer/Counter (Continued)

*Variable Frequency, Variable Pulse Width Output.* The following module illustrates one possible use of T<sub>OUT</sub>. Assume it is necessary to generate a pulse train with a 10% duty cycle, where the output is repetitively high for 1.6 ms and then low for 14.4 ms. To do this, T<sub>OUT</sub> is controlled by end-of-count from T<sub>1</sub>, although T<sub>0</sub> could alternately be chosen. This example makes use of the Z8 feature that allows a timer's counter register to be modified without disturbing the count in progress. In continuous mode, the new value is loaded when T<sub>1</sub> reaches EOC. T<sub>1</sub> is first loaded and enabled with values to generate the short interval. The counter register is then immediately modified with the value to generate the long interval; this value is loaded into the counter automatically on T<sub>1</sub> EOC. The prescaler selected value must be the same for both long and short intervals. Note that the initial loading of the T<sub>1</sub> counter

register is followed by setting the T<sub>1</sub> load bit of timer control register TMR (F1H); this action causes T<sub>OUT</sub> to be reset to a logic 1 output. Each subsequent modification of the T<sub>1</sub> counter register does not affect the current T<sub>OUT</sub> level, since the T<sub>1</sub> load bit is NOT altered by the software. The new value is loaded on EOC, and T<sub>OUT</sub> will toggle at that time. The T<sub>1</sub> interrupt service routine should simply modify the T<sub>1</sub>, counter register with the new value, alternating between the long and short interval values.

In the example which follows, bit 0 of register 04H is used as a software flag to indicate which value was loaded last. This module illustrates the procedure for T<sub>1</sub>/T<sub>OUT</sub> initialization (PULSE\_INIT), the T<sub>1</sub> interrupt service routine (PULSE), and the interrupt vector for T<sub>1</sub> EOC (IRQ\_5). XTAL = 8 MHz is assumed.

```

MACZ8      2.0
LOC      OBJ CODE STMT SOURCE STATEMENT
          1 TIMER2  MODULE
          2          $SECTION PROGRAM
          3 GLOBAL
          4 !IRQ5 interrupt vector!
          5          $ABS 10
P 0000 0017' 6 IRQ_5  ARRAY          [1 WORD]      :=      [PULSE]
          7
          8          $REL
P 000C          9 PULSE_INIT  PROCEDURE
          10 ENTRY
P 0000 E6 F3 03 11          LD          PRE1, #(2)00000011H
          12
          13
          14
P 0003 E6 F7 00 15          LD          P3M, #00
P 0006 E6 F2 19 16          LD          T1, #25
P 0009 8F          17          DI
P 000A 46 FB 20 18          OR          IMR, #(2)00100000H
P 000D E6 F1 8C 19          LD          TMR, #(2)10001100H
          20
          21
          22
          23
          24 !Set long interval counter, to be loaded on T1 EOC!
P 0010 E6 F2 E1 25          LD          T1, #225
          26 !Clear alternating flag for PULSE!
P 0013 B0 04          27          CLR          04H
          28
          29
          30
          31 END          PULSE_INIT
          32
          33
P 0017          34 PULSE  PROCEDURE
          35 ENTRY

```

!bit 2-7: prescaler = 64;  
!bit 1: internal clock;  
!bit 0: continuous mode!  
!bit 5: let P36 be Tout!  
!for short interval!  
!enable T1 interrupt!  
!bit 6-7: Tout controlled  
by T1;  
!bit 3: enable T1;  
!bit 2: load T1!  
! = 0:25 next;  
! = 1:225 next!

## Timer/Counter Functions (Continued)

P 0017	E6	F2	E1	36	LD	T1, #225	!new load value!
P 001A	B6	04	01	37	XOR	04H, #1	!which value next?!
P 001D	6B	03		38	JR	Z,PULSE_EXIT	!should be 225!
P 001F	E6	F2	19	39	LD	T1, #25	!should be 25!
				40	PULSE_EXIT:		
P 0022	BF			41	IRET		
P 0023				42	END	PULSE	
				43	END	TIMER2	

0 ERRORS  
ASSEMBLY COMPLETE

### PULSE\_INIT:

10 instructions  
23 bytes  
23  $\mu$ s

### PULSE:

5 instructions  
12 bytes  
25  $\mu$ s (average) including interrupt response time

**Cascaded Timer/Counters.** For some applications it may be necessary to measure a greater time interval than a single timer/counter can measure (16.384 ms). In this case,  $T_{IN}$  and  $T_{OUT}$  may be used to

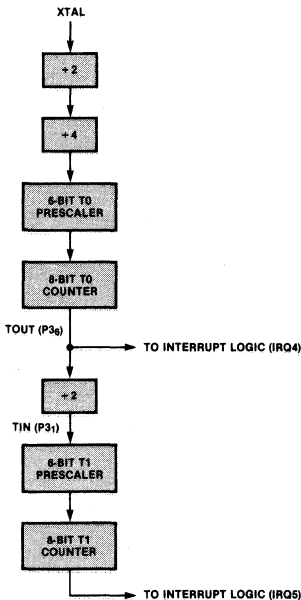


Figure 3. Cascaded Timer/Counters

cascade  $T_0$  and  $T_1$  to function as a single unit.  $T_{OUT}$ , programmed to toggle on  $T_0$  end-of-count, should be wired back to  $T_{IN}$ , which is selected as the external clock input for  $T_1$ . With  $T_0$  programmed for continuous mode,  $T_{OUT}$  (and therefore  $T_{IN}$ ) goes through a High-to-Low transition (causing  $T_1$  to count) on every other  $T_0$  EOC. Interrupt request  $IRQ_5$  is generated when the programmed time interval has elapsed. Interrupt requests  $IRQ_2$  (generated on every  $T_{IN}$  High-to-Low transition) and  $IRQ_4$  (generated on  $T_0$  EOC) are of no importance in this application and are therefore disabled.

To determine the time interval ( $i$ ) until EOC, the equation

$$i = t \times p_0 \times v_0 \times (2 \times p_1 \times v_1 - 1)$$

characterizes the relation between the  $T_0$  prescaler ( $p_0$ ) and counter ( $v_0$ ), the  $T_1$  prescaler ( $p_1$ ) and counter ( $v_1$ ), and the clock input period ( $t$ );  $t$  is defined in Section 7.1. Assuming  $XTAL = 8$  MHz, the measurable time interval range is

$$\begin{aligned}
 1 \mu\text{s} \times 1 \times 1 \times (2 \times 1 - 1) &\leq i \leq \\
 1 \mu\text{s} \times 64 \times 256 \times (2 \times 64 \times 256 - 1) & \\
 1 \mu\text{s} \leq i &\leq 536.854528 \text{ s}
 \end{aligned}$$

Figure 3 illustrates the interconnection between  $T_0$  and  $T_1$ . The following module illustrates the procedure required to initialize the timers for a 1.998 second delay interval:

## Timer/Counter Functions (Continued)

```

MACZ8      2.0
LOC      OBJ CODE STMT SOURCE STATEMENT
          1 TIMER3  MODULE
          2 GLOBAL
P 0000     3 TIMER__16  PROCEDURE
          4 ENTRY
P 0000  E6  F3  28          5          LD  PRE1, #(2)00101000H
          6                                !bit 2-7: prescaler = 10;
          7                                bit 1: external clock;
          8                                bit 0: single-pass mode!
P 0003  E6  F7  00          9          LD  P3M, #00          !bit 5: let P36 be Tout!
P 0006  E6  F2  64         10         LD  T1, #100          !T1 counter register!
P 0009  E6  F5  29         11         LD  PRE0, #(2)00101001H
          12                                !bit 2-7: prescaler = 10;
          13                                bit 0: continuous mode!
P 000C  E6  F4  64         14         LD  T0, #100          !T0 counter register!
P 000F  8F          15         DI
P 0010  56  FB  2B         16         AND  IMR, #(2)00101011H !disable IRQ2 (Tin);
          17                                and IRQ4 (T0) !
P 0013  46  FB  20         18         OR  IMR, #(2)00100000H !enable IRQ5 (T1)!
P 0016  9F          19         EI
P 0017  E6  F1  4F         20         LD  TMR, #(2)01001111H
          21                                !bit 6-7: Tout controlled
          22                                by T0;
          23                                bit 4-5: Tin mode is ext.
          24                                clock input;
          25                                bit 3: enable T1;
          26                                bit 2: load T1;
          27                                bit 1: enable T0;
          28                                bit 0: load T0!
P 001A  AF          29         RET
P 001B          30 END    TIMER__16
          31 END    TIMERS

```

0 ERRORS  
ASSEMBLY COMPLETE

11 instructions  
27 bytes  
26.5  $\mu$ s

*Clock Monitor.*  $T_1$  and  $T_{IN}$  may be used to monitor a clock line (in a diskette drive, for example) and generate an interrupt request when a clock pulse is missed. To accomplish this, the clock line to be monitored is wired to P3, ( $T_{IN}$ ).  $T_{IN}$  should be programmed as a retriggerable input to  $T_1$ , such that each falling edge on  $T_{IN}$  will cause  $T_1$  to reload and continue counting. If  $T_1$  is programmed to time-out after an interval of one-and-a-half times the clock period being monitored,  $T_1$  will time-out and generate interrupt request IRQ5 only if a clock pulse is missed.

The following module illustrates the procedure for initializing  $T_1$  and  $T_{IN}$  (MONITOR\_\_INIT) to monitor a clock with a period of 2  $\mu$ s. XTAL=8 MHz is assumed. Note that this example selects single-pass rather than continuous mode for  $T_1$ . This is to prevent a continuous stream of IRQ5 interrupt requests in the event that the monitored clock fails completely. Rather, the interrupt service routine (CLK\_\_ERR) is left with the choice of whether or not to re-enable the monitoring. Also shown is the  $T_1$  interrupt vector (IRQ\_\_5).

## Timer/Counter Functions (Continued)

MAC28 2.0

```

LOC  OBJ CODE STMT SOURCE STATEMENT
      1  TIMER4  MODULE
      2  $SECTION PROGRAM
      3  GLOBAL
      4  !IRQ5 interrupt vector!
      5  $ABS 10
P 0000 0015' 6  IRQ_5  ARRAY [1 WORD]  :=  [CLK_ERR]
      7
      8  $REL
P 000C      9  MONITOR_INIT  PROCEDURE
      10  ENTRY
P 0000  E6  F3  04 11  LD  PRE1,#(2)0000100H
      12  !bit 2-7: prescaler = 1;
      13  !bit 1: external clock;
      14  !bit 0: single-pass model!
P 0003  E6  F7  00 15  LD  P3M,#00  !bit 5: let P36 be Tout!
P 0006  E6  F2  03 16  LD  T1,#3  !T1 load register,
      17  = 1.5 * 2 usec!
P 0009  8F      18  DI
P 000A  56  FB  3B 19  AND  IMR,#(2)00111011H !disable IRQ2 (Tin)!
P 000D  46  FB  20 20  OR  IMR,#(2)00100000H !enable IRQ5 (T1)!
P 0010  9F      21  EI
      22
P 0011  E6  F1  38 23  LD  TMR,#(2)00111000H
      24  !bit 4-5: Tin mode is
      25  retrig. input;
      26  bit 3: enable T1!
P 0014  AF      27  RET
P 0015      28  END  MONITOR_INIT
      29
      30
P 0015      31  CLK_ERR PROCEDURE
      32  ENTRY
      33  !...!  !handle the missed clock!
      34
      35  !if clock monitoring should continue...!
P 0015  46  F1  08 36  OR  TMR,#(2)00001000H
      37  !bit 3: enable T1!
P 0018  BF      38  IRET
P 0019      39  END  CLK_ERR
      40  END  TIMER4

```

0 ERRORS

ASSEMBLY COMPLETE

MONITOR\_INIT:  
 9 instructions  
 21 bytes  
 21.5  $\mu$ s

CLK\_ERR:  
 2 + instructions  
 4 + bytes  
 18.5 +  $\mu$ s including interrupt response time

## I/O Functions

The Z8 provides 32 I/O lines mapped into registers 0-3 of the internal register file. Each nibble of port 0 is individually programmable as input, output, or address/data lines ( $A_{15}$ - $A_{12}$ ,  $A_{11}$ - $A_8$ ). Port 1 is programmable as a single entity to provide input, output, or address/data lines ( $AD_7$ - $AD_0$ ). The operating modes for the bits of Ports 0 and 1 are selected by control register P01M (F8H). Selection of I/O lines as address/data lines supports access to external program and data memory; this is discussed in Section 3. Each bit of Port 2 is individually programmable as an input or an output bit. Port 2 bits programmed as outputs may also be programmed (via bit 0 of P3M) to all have active pull-ups or all be open-drain (active pull-ups inhibited). In Port 3, four bits ( $P_{30}$ - $P_{33}$ ) are fixed as inputs, and four bits ( $P_{34}$ - $P_{37}$ ) are fixed as outputs, but their functions are programmable. Special functions provided by Port 3 bits are listed in Table 4. Use of the Data Memory select output is discussed in Section 3; uses of  $T_{IN}$  and  $T_{OUT}$  are discussed in Section 7.

### Asynchronous Receiver/Transmitter

**Operation.** Full-duplex, serial asynchronous receiver/transmitter operation is provided by the Z8 via  $P_{37}$  (output) and  $P_{30}$  (input) in conjunction with control register SIO (F0H), which is actually two registers: receiver buffer and transmitter buffer. Counter/Timer  $T_0$  provides the clock for control of the bit rate.

The Z8 always receives and transmits eight bits between start and stop bits. However, if parity is enabled, the eighth bit ( $D_7$ ) is replaced by the odd-parity bit when transmitted and a parity-error flag (= 1 if error) when received. Table 5 illustrates the state of the parity bit/parity error flag during serial I/O with parity enabled.

Although the Z8 directly supports either odd parity or no parity for serial I/O operation, even parity may also be provided with additional software support. To receive

Functions	Bit	Signal
Handshake	$P_{31}$	$\overline{DAV2}/RDY2$
	$P_{32}$	$\overline{DAV0}/RDY0$
	$P_{33}$	$\overline{DAV1}/RDY1$
	$P_{34}$	$RDY1/\overline{DAV1}$
	$P_{35}$	$RDY0/\overline{DAV0}$
	$P_{36}$	$RDY2/\overline{DAV2}$
Interrupt Request	$P_{30}$	IRQ3
	$P_{31}$	IRQ2
	$P_{32}$	IRQ0
	$P_{33}$	IRQ1
Counter/Timer	$P_{31}$	$T_{IN}$
	$P_{30}$	$T_{OUT}$
Data Memory Select Status Out	$P_{34}$	$\overline{DM}$
Serial I/O	$P_{30}$	Serial In
	$P_{37}$	Serial Out

Table 4. Port 3 Special Functions

and transmit with even parity, the Z8 should be configured for serial I/O with odd parity disabled. The Z8 software must calculate parity and modify the eighth bit prior to the load of a character into SIO and then modify a parity error flag following the load of a character from SIO. All other processing required for serial I/O (e.g. buffer management, error handling, etc.) is the same as that for odd parity operations.

To configure the Z8 for Serial I/O, it is necessary to:

- Enable  $P_{30}$  and  $P_{37}$  for serial I/O and select parity,
- Set up  $T_0$  for the desired bit rate,
- Configure IRQ3 and IRQ4 for polled or automatic interrupt mode,
- Load and enable  $T_0$ .

Character Loaded Into SIO	Transmitted To Serial Line	Received From Serial Line	Character Transferred to SIO	Note*
11000011	01000011	01000011	01000011	no error
11000011	01000011	01000111	11000111	error
01111000	11111000	11111000	01111000	no error
01111000	11111000	01111000	11111000	error

Table 5. Serial I/O With Odd Parity

\* Left-most bit is  $D_7$



## I/O Functions (Continued)

To enable P3<sub>0</sub> and P3<sub>7</sub> for serial I/O, bit 6 of P3M (R247) is set. To enable odd parity, bit 7 of P3M is set; to disable it, the bit is reset. For example, the instruction

```
LD P3M, #40H
```

will enable serial I/O, but disable parity. The instruction

```
LD P3M, #C0H
```

will enable serial I/O, and enable odd parity.

In the following discussions, bit rate refers to all transmitted bits, including start, stop and parity (if enabled). The serial bit rate is given by the equation:

$$\text{bit rate} = \frac{\text{input clock frequency}}{(2 \times 4 \times T_0 \text{ prescaler} \times T_0 \text{ counter} \times 16)}$$

The final divide-by-16 is incurred for serial communications, since in this mode T<sub>0</sub> runs at 16 times the bit rate in order to synchronize the data stream. To configure the Z8 for a specific bit rate, appropriate values must first be selected for T<sub>0</sub> prescaler and T<sub>0</sub> counter by the above equation; these values are then programmed into registers T<sub>0</sub> (F4H) and PRE0 (F5H) respectively. Note that PRE0 also controls the continuous vs. single-pass mode for T<sub>0</sub>; continuous mode should be selected for serial I/O. For example, given an input clock frequency of 7.3728 MHz and a selected bit rate of 9600 bits per second, the equation is satisfied by T<sub>0</sub> counter = 2 and prescaler = 3. The following code sequence will configure the T<sub>0</sub> counter and T<sub>0</sub> prescaler registers:

```
LD T0, #2 !T0 counter = 2!  
LD PRE0, #(2)00001101H  
!bit 2-7: prescaler = 3; bit 0:  
continuous mode!
```

Interrupt request 3 (IRQ3) is generated whenever a character is transferred into the receive buffer; interrupt request 4 (IRQ4) is generated whenever a character is transferred out of the transmit buffer. Before accepting such interrupt requests, the Interrupt Mask, Request, and Priority Registers (IMR, IRQ, and IPR) must be programmed to configure the mode of interrupt response. The section on Interrupt Processing provides a discussion of interrupt configurations.

To load and enable T<sub>0</sub>, set bits 0 and 1 of the timer mode register (TMR) via an instruction such as

```
OR TMR, #03H
```

This will cause the T<sub>0</sub> prescaler and counter registers (PRE0 and T<sub>0</sub>) to be transferred to the T<sub>0</sub> prescaler and counter. In addition, T<sub>0</sub> is enabled to count, and serial I/O operations will commence.

Characters to be output to the serial line should be written to serial I/O register SIO (F0H). IRQ4 will be generated when all bits have been transferred out.

Characters input from the serial line may be read from SIO. IRQ3 will be generated when a full character has been transferred into SIO.

The following module illustrates the receipt of a character and its immediate echo back to the serial line. It is assumed that the Z8 has been configured for serial I/O as described above, with IRQ3 (receive) enabled to interrupt, and IRQ4 (transmit) configured to be polled. The received character is stored in a circular buffer in register memory from address 42H to 5FH. Register 41H contains the address of the next available buffer position and should have been initialized by some earlier routine to #42H.

## I/O Functions (Continued)

```

MACZ8      2.0
LOC      OBJ CODE SIMT SOURCE STATEMENT
          1 SERIAL_IO      MODULE
          2 CONSTANT
          3 next_addr      :=      41H
          4 start         :=      42H
          5 length        :=      1EH
          6 $SECTION PROGRAM
          7 GLOBAL
          8 !IRQ3 vector!
          9      $ABS      6
P 0006  0000' 10 IRQ_3      ARRAY [1 WORD] := [GET_CHARACTER]
          11
          12      $REL      0
P 0000      13 GET_CHARACTER PROCEDURE      ENTRY
          14
          15 !Serial I/O receive interrupt service!
          16 !Echo received character and wait for
          17 echo completion!
P 0000  E4  F0  F0  18      ld      SIO,SIO      !echo!
          19
          20 !save it in circular buffer!
P 0003  F5  F0  41  21      ld      @next_addr,SIO      !save in buffer!
P 0006  20  41  22      inc     next_addr      !point to next position!
P 0008  A6  41  60  23      cp      next_addr, #start + length
          24      !wrap-around yet?!
P 000B  EB  03      25      jr      ne,echo_wait      !no.!
P 000D  E6  41  42  26      ld      next_addr, #start      !yes. point to start!
          27 !now, wait for echo complete!
          28 echo_wait:
P 0010  66  FA  10  29      tcm     IRQ, #10H      !transmitted yet?!
P 0013  EB  FB      30      jr      nz,echo_wait      !not yet!
          31
P 0015  56  FA  E-  32      and     IRQ, #EFH      !clear IRQ4!
          F
P 0018  BF      33      IRET      !return from interrupt!
P 0019      34 END      GET_CHARACTER
          35 END      SERIAL_IO

```

0 ERRORS  
ASSEMBLY COMPLETE

*10 instructions  
25 bytes  
35.5  $\mu$ s + 5.5  $\mu$ s for each additional pass through the echo\_wait loop,  
including interrupt response time*

**Automatic Bit Rate Detection.** In a typical system, where serial communication is required (e.g. system with a terminal), the desired bit rate is either user-selectable via a switch bank or nonvariable and "hard-coded" in the software. As an alternate method of bit-rate detection, it is possible to automatically determine the bit rate of serial data received by measuring the length of a start bit. The advantage of this method is that it places no requirements on the hardware design for this function and provides a convenient (automatic) operator interface.

In the technique described here, the serial channel of the Z8 is initialized to expect a bit rate of 19,200 bits per second. The number of bits (n) received through Port pin P30 for each bit transmitted is expressed by

$$n = 19,200/b$$

where b = transmission bit rate. For example, if the transmission bit rate were 1200 bits per second, each incoming bit would appear to the receiving serial line as 19,200/1200 or 16 bits.

**I/O Functions (Continued)**

The following example is capable of distinguishing between the bit rates shown in Table 6 and assumes an input clock frequency of 7.3728 MHz, a T<sub>0</sub> prescaler of 3, and serial I/O enabled with parity disabled. This example requires that a character with its low order bit = 1 (such as a carriage return) be sent to the serial channel. The start bit of this character can be measured by counting the number of zero bits collected before the low order 1 bit. The number of zero bits actually collected into data bits by the serial channel is less than n (as given in the above equation), due to the detection of start and stop bits. Figure 4 illustrates the collection (at 19,200 bits per second) of a zero bit transmitted to the Z8 at 1,200 bits per second. Notice that only 13 of the 16 zero bits received are collected as data bits.

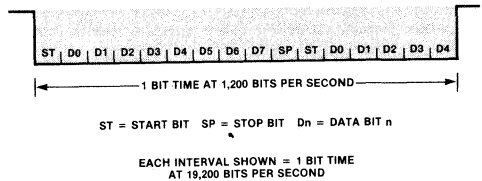
Once the number of zero bits in the start bit has been collected and counted, it remains to translate this count into the appropriate T<sub>0</sub> counter value and program that value into T<sub>0</sub> (F4H). The patterns shown in the two binary columns of Table 6 are utilized in the algorithm for this translation.

As a final step, if incoming data is to

commence immediately, it is advisable to wait until the remainder of the current "elongated" character has been received, thus "flushing" the serial line. This can be accomplished either via a software loop, or by programming T<sub>1</sub> to generate an interrupt request after the appropriate amount of time has elapsed. Since a character is composed of eight bits plus a minimum of one stop bit following the start bit, the length of time to delay may be expressed as

$$(9 \times n) / b$$

where n and b are as defined above. The following module illustrates a sample program for automatic bit rate detection.



**Figure 4. Collection of a Start Bit Transmitted at 1,200 BPS and Receive at 19,200 BPS**

Bit Rate	Number of Bits Received Per Bit Transmitted	Number of 0 Bits Collected as Data Bits		T <sub>0</sub> Counter	
		dec	binary	dec	binary
19200	1	0	00000000	1	00000001
9600	2	1	00000001	2	00000010
4800	4	3	00000011	4	00000100
2400	8	7	00000111	8	00001000
1200	16	13	00001101	16	00010000
600	32	25	00011001	32	00100000
300	64	49	00110001	64	01000000
150	128	97	01100001	128	10000000

**Table 6. Inputs to the Automatic Bit Rate Detection Algorithm**

```

MACZ8      2.0
LOC      OBJ CODE STMT SOURCE STATEMENT
                                1 bit_rate          MODULE
                                2 EXTERNAL
                                3 DELAY PROCEDURE
                                4 GLOBAL
P 0000     5 main          PROCEDURE
                                6          ENTRY
    
```

## I/O Functions (Continued)

```

P 0000 8F          7      di          !disable interrupts!
P 0001 56 FB 77   8      and          IMR, #77H !IRQ3 polled mode!
P 0004 56 FA F7   9      and          IRQ, #F7H !clear IRQ3!
P 0007 E6 F7 40  10     ld          P3M, #40H !enable serial I/O!
P 000A E6 F4 01  11     ld          T0, #1
P 000D E6 F5 0    12     ld          PRE0, #(3 SHL 2)+1 !bit rate=19,200;
D          13                    continuous count mode!
P 0010 B0 E0     14     clr          R0          !init. zero byte counter!
P 0012 E6 F1 03  15     ld          TMR, #3     !load and enable T0!
16
17 !collect input bytes by counting the number of null
18 characters received. Stop when non-zero byte received!
19 collect:
P 0015 76 FA 08  20     TM          IRQ, #08H !character received?!
P 0018 6B FB     21     jr          z,collect !not yet!
P 001A 18 F0     22     ld          R1,SIO !get the character!
P 001C 56 FA F7  23     and          IRQ, #F7 !clear interrupt request!
P 001F 1E        24     inc          R1          !compare to 0 ...!
P 0020 1A 05     25     djnz         R1,bitloop !...(in 3 bytes of code)!
P 0022 06 E0 08  26     add          R0, #8     !update count of 0 bits!
P 0025 8B EE     27     jr          collect
28 bitloop:          !add in zero bits from low!
29                    end of 1st non-zero byte!
P 0027 E0 E1     30     RR          R1
P 0029 7B 03     31     jr          c, count_done
P 002B 0E        32     inc          R0
P 002C 8B F9     33     jr          bitloop
34
35 !R0 has number of zero bits collected!
36 !translate R0 to the appropriate T0 counter value!
37 count_done:          !R0 has count of zero bits!
P 002E 1C 07     38     ld          R1, #7
P 0030 2C 80     39     ld          R2, #80H !R2 will have T0 counter
P 0032 90 E0     40     RL          R0          value!
41
P 0034 90 E0     42 loop:   RL          R0
P 0036 7B 04     43     jr          c,done
P 0038 E0 E2     44     RR          R2
P 003A 1A F8     45     djnz         r1,loop
46
P 003C 29 F4     47 done:   ld          T0,R2 !load value for detected
48                    bit rate!
49 !Delay long enough to clear serial line of bit stream!
P 003E D6 0000*  50     call         DELAY
51 !clear receive interrupt request!
P 0041 56 FA F7  52     and          IRQ, #F7H
53
P 0044          54 END    main
          55 END    bit_rate

```

0 ERRORS  
ASSEMBLY COMPLETE

30 instructions  
68 bytes

Execution time is variable based on transmission bit rate.

## I/O Functions (Continued)

**Port Handshake.** Each of Ports 0, 1 and 2 may be programmed to function under input or output handshake control. Table 7 defines the port bits used for the handshaking and the mode bit settings required to select handshaking. To input data under handshake control, the Z8 should read the input port when the  $\overline{DAV}$  input goes Low (signifying that data is available from the attached device). To output data under handshake control, the Z8 should write the output port when the RDY input goes Low (signifying that the previously output data has been accepted by the attached device). Interrupt requests IRQ0, IRQ1, and IRQ2 are generated by the falling edge of the handshake signal input to the Z8 for Port 0, Port 1, and Port 2 respectively. Port handshake operations may therefore be processed under interrupt control.

Consider a system that requires communication of eight parallel bits of data under hand-shake control from the Z8 to a peripheral device and that Port 2 is selected as the output port. The following assembly code illustrates the proper sequence for initializing Port 2 for output handshake.

```

CLR  P2M    !Port 2 mode register: all
                Port 2 bits are outputs!
OR   03H, #40H
                !set  $\overline{DAV}2$ : data not available!
LD   P3M, #20H
                !Port 3 mode register: enable
                Port 2 handshake!
LD   02H, DATA
                !output first data byte;  $\overline{DAV}2$ 
                will be cleared by the Z8 to
                indicate data available to
                the peripheral device!
    
```

Note that following the initialization of the output sequence, the software outputs the first data byte without regard to the state of the RDY2 input; the Z8 will automatically hold  $\overline{DAV}2$  High until the RDY2 input is High. The peripheral device should force the Z8 RDY2 input line Low after it has latched the data in response to a Low on  $\overline{DAV}2$ . The Low on RDY2 will cause the Z8 automatically force  $\overline{DAV}2$  High until the next byte is output. Subsequent bytes should be output in response to interrupt request IRQ2 (caused by the High-to-Low transition on RDY2) in either a polled or an enabled interrupt mode.

	Port 0	Port 1	Port 2
Input handshake lines	$\left\{ \begin{array}{l} P3_2 = \overline{DAV} \\ P3_5 = RDY \end{array} \right.$	$\left\{ \begin{array}{l} P3_3 = \overline{DAV} \\ P3_4 = RDY \end{array} \right.$	$\left\{ \begin{array}{l} P3_1 = \overline{DAV} \\ P3_6 = RDY \end{array} \right.$
Output handshake lines	$\left\{ \begin{array}{l} P3_2 = RDY \\ P3_5 = \overline{DAV} \end{array} \right.$	$\left\{ \begin{array}{l} P3_3 = RDY \\ P3_4 = \overline{DAV} \end{array} \right.$	$\left\{ \begin{array}{l} P3_1 = RDY \\ P3_6 = \overline{DAV} \end{array} \right.$
To select input handshake:	$\left\{ \begin{array}{l} \text{set bit 6 \& reset bit 7 of} \\ \text{P01M (program high} \\ \text{nibble as input)} \end{array} \right.$	$\left\{ \begin{array}{l} \text{set bit 3 \& reset bit 4 of} \\ \text{P01M (program byte as} \\ \text{input)} \end{array} \right.$	$\left\{ \begin{array}{l} \text{set bit 7 of P2M} \\ \text{(program high bit as input)} \end{array} \right.$
To select output handshake:	$\left\{ \begin{array}{l} \text{reset bits 6,7 of P01M} \\ \text{(program high nibble as} \\ \text{output)} \end{array} \right.$	$\left\{ \begin{array}{l} \text{reset bits 3, 4 of P01M} \\ \text{(program byte as output)} \end{array} \right.$	$\left\{ \begin{array}{l} \text{reset bit 7 of P2M} \\ \text{(program high bit as output)} \end{array} \right.$
To enable handshake:	$\left\{ \begin{array}{l} \text{set bit 5 of Port 3 (P3}_5\text{);} \\ \text{set bit 2 of P3M} \end{array} \right.$	$\left\{ \begin{array}{l} \text{set bit 4 of Port 3 (P3}_4\text{);} \\ \text{set bits 3, 4 of P3M} \end{array} \right.$	$\left\{ \begin{array}{l} \text{set bit 6 of Port 3 (P3}_6\text{);} \\ \text{set bit 5 of P3M} \end{array} \right.$

**Table 7. Port Handshake Selection**

## Arithmetic Routines

This section gives examples of the arithmetic and rotate instructions for use in multiplication, division, conversion, and BCD arithmetic algorithms.

**Binary to Hex ASCII.** The following module illustrates the use of the ADD and SWAP arithmetic instructions in the conversion of a 16-bit binary number to its hexadecimal ASCII representation. The 16-bit number is

viewed as a string of four nibbles and is processed one nibble at a time from left to right, beginning with the high-order nibble of the lower memory address. 30H is added to each nibble if it is the range 0 to 9; otherwise 37H is added. In this way, 0H is converted to 30H, 1H to 31H, ...AH to 41H, ...FH to 46H. Figure 5 illustrates the conversion of RR0 (contents = F2BEH) to its hex ASCII equivalent; the destination buffer is pointed to by RR4.

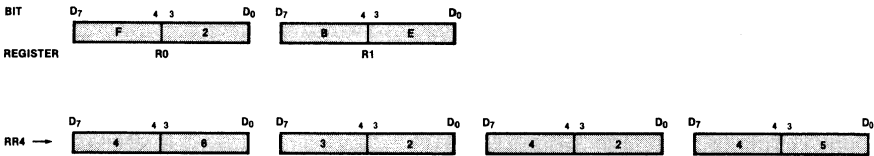


Figure 5. Conversion of (RR0) to Hex ASCII

MACZ8 2.99 INTERNAL RELEASE

LOC OBJ CODE STMT SOURCE STATEMENT

```

1 ARITH    MODULE
2 GLOBAL
P 0000 3 BINASC  PROCEDURE
4 !
5 Purpose =      To convert a 16-bit binary
6                number to Hex ASCII
7
8 Input =        RR0 = 16-bit binary number.
9                RR4 = pointer to destination
10               buffer in external memory.
11
12 Output =      Resulting ASCII string (4 bytes)
13                in destination buffer.
14                RR4 incremented by 4.
15                R0, R2, R6 destroyed.
16 *****
17 ENTRY
18
P 0000 6C 04     19      ld    R6,#04H    !nibble count!
P 0002 F0 E0     20  again: SWAP R0    !look at next nibble!
P 0004 28 E0     21      ld    R2,R0
P 0006 56 E2 0F  22      and   R2,#0FH    !isolate 4 bits!

```

## Arithmetic Routines (Continued)

```

23 lconvert to ASCII : R2 + #30H if R0 in range 0 to 9
24 else R2 + #37H (in range 0A to 0F)
25!
P 0009 06 E2 30 ADD R2,#30H
P 000C A6 E2 3A cp R2,#3AH
P 000F 7B 03 jr ult,skip
P 0011 06 E2 07 ADD R2,#07H
P 0014 92 24 30 skip: lde @RR4,R2 !save ASCII in buffer!
P 0016 A0 E4 31 incw RR4 !point to next
32 !buffer position!
P 0018 A6 E6 03 33 cp R6,#03H !time for second byte?!
P 001B EB 02 34 jr ne,same__byte !no.!
P 001D 08 E1 35 ld R0,R1 !2nd byte!
36 same__byte:
P 001F 6A E1 37 djnz R6,again
P 0021 AF 38 ret
P 0022 39 END BINASC
40 END ARITH

```

0 ERRORS  
ASSEMBLY COMPLETE

15 instructions  
34 bytes  
120.5  $\mu$ s (average)

**BCD Addition.** The following module illustrates the use of the add with carry (ADC) and decimal adjust (DA) instructions for the addition of two unsigned BCD strings of equal length. Within a BCD string, each nibble represents a decimal digit (0-9). Two such digits are packed per byte with the

most significant digit in bits 7-4. Bytes within a BDC string are arranged in memory with the most significant digits stored in the lowest memory location. Figure 6 illustrates the representation of 5970 in a 6-digit BCD string, starting in register 33H.

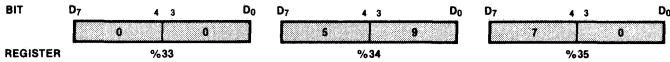


Figure 6. Unsigned BCD Representation

## Arithmetic Routine (Continued)

```

MACZ8      2.0
LOC      OBJ CODE STMT SOURCE STATEMENT

          1 ARITH      MODULE
          2 CONSTANT
          3 BCD_SRC := R1
          4 BCD_DST := R0
          5 BCD_LEN := R2
          6 GLOBAL
P 0000    7 BCDADD PROCEDURE
          8 |*****!
          9 Purpose =      To add two packed BCD strings of
         10                equal length.
         11                dst <-- dst + src
         12
         13 Input =       R0 = pointer to dst BCD string.
         14                R1 = pointer to src BCD string.
         15                R2 = byte count in BCD string
         16                (digit count = (R2)*2 ).
         17
         18 Output =      BCD string pointed to by R0 is
         19                the sum.
         20                Carry FLAG = 1 if overflow.
         21                R0, R1 as on entry.
         22                R2 = 0
         23                *****!
         24 ENTRY
         25
P 0000    26          add   BCD_SRC,BCD_LEN      !start at least...!
P 0002    27          add   BCD_DST,BCD_LEN      !significant digits!
P 0004    28          rcf                                !carry = 0!
         29 add_again:
P 0005    30          dec   BCD_SRC                !point to next two
         31                src digits!
P 0007    32          dec   BCD_DST                !point to next two
         33                dst digits!
P 0009    34          ld    R3,@BCD_SRC            !get src digits!
P 000B    35          ADC   R3,@BCD_DST            !add dst digits!
P 000D    36          DA    R3                    !decimal adjust!
P 000F    37          ld    @BCD_DST,R3           !move to dst!
P 0011    38          djnz  BCD_LEN,add_again      !loop for next
         39                digits!
P 0013    40          ret                                !all done!
         41
P 0014    42 END      BCDADD
         43 END      ARITH

```

0 ERRORS  
ASSEMBLY COMPLETE

11 instructions

20 bytes

Execution time is a function of the number of bytes (n) in input BCD string:

$20 \mu s + 12.5 (m - 1) \mu s$



## Arithmetic Routines (Continued)

**Multiply.** The following module illustrates an efficient algorithm for the multiplication of two unsigned 8-bit values, resulting in a 16-bit product. The algorithm repetitively shifts the multiplicand right (using RRC), with the low-order bit being shifted out (into the carry flag). If a one is shifted out, the multiplier is added to the high-order byte of

the partial product. As the high-order bits of the multiplicand are vacated by the shift, the resulting partial-product bits are rotated in. Thus, the multiplicand and the low byte of the product occupy the same byte, which saves register space, code, and execution time.

```
MACZ8 2.99 INTERNAL RELEASE
LOC   OBJ CODE STMT SOURCE STATEMENT

      1 ARITH  MODULE
      2 CONSTANT
      3 MULTIPLIER :=      R1
      4 PRODUCT_LO:=      R3
      5 PRODUCT_HI:=      R2
      6 COUNT     :=      R0
      7 GLOBAL
P 000 8 MULT   PROCEDURE
      9 !*****
     10 Purpose =      To perform an 8-bit by 8-bit unsigned
     11                binay multiplication.
     12
     13 Input  =      R1 = multiplier
     14                R3 = multiplicand
     15
     16 Output =      RR2 = product
     17                R0 destroyed
     18 ***** !
     19 ENTRY
P 0000 OC 09      20          ld   COUNT,#9      !8 BITS + 1!
P 0002 B0 E2      21          clr  PRODUCT_HI  !INIT HIGH RESULT BYTE!
P 0004 CF         22          RCF          !CARRY = 0!
P 0005 C0 E2      23 LOOP:  RRC  PRODUCT_HI
P 0007 C0 E3      24          RRC  PRODUCT_LO
P 0009 FB 02      25          jr   NC,NEXT
P 000B 02 21      26          ADD  PRODUCT_HI, MULTIPLIER
P 000D 0A F6      27 NEXT:  djnz  COUNT,LOOP
P 000F AF         28          ret
P 0010           29 END    MULT
           30 END    ARITH
```

0 ERRORS  
ASSEMBLY COMPLETE

9 instructions  
16 bytes  
92.5  $\mu$ s (average)

**Divide.** The following module illustrates an efficient algorithm for the division of a 16-bit unsigned value by an 8-bit unsigned value, resulting in an 8-bit unsigned quotient. The algorithm repetitively shifts the dividend left (via RLC). If the high-order bit shifted out is a one or if the resulting high-order dividend byte is greater than or equal to the divisor,

the divisor is subtracted from the high byte of the dividend. As the low-order bits of the dividend are vacated by the shift left, the resulting partial-quotient bits are rotated in. Thus, the quotient and the low byte of the dividend occupy the same byte, which saves register space, code, and execution time.

## Arithmetic Routine (Continued)

```

MAC28      2.0
LOC      OBJ CODE STMT SOURCE STATEMENT
          1 ARITH      MODULE
          2 CONSTANT
          3 COUNT      :=      R0
          4 DIVISOR    :=      R1
          5 DIVIDEND_HI :=      R2
          6 DIVIDEND_LO :=      R3
          7 GLOBAL
P 0000    8 DIVIDE     PROCEDURE
          9 !*****
          10 Purpose =   To perform an 16-bit by 8-bit unsigned
          11                binary division.
          12
          13 Input =     R1 = 8-bit divisor
          14                RR2 = 16-bit dividend
          15
          16 Output =    R3 = 8-bit quotient
          17                R2 = 8-bit remainder
          18                Carry flag = 1 if overflow
          19                = 0 no overflow
          20 !*****!
          21 ENTRY
P 0000    0C 08      22          ld      COUNT,#8 !LOOP COUNTER!
          23
          24 !CHECK IF RESULT WILL FIT IN 8 BITS!
P 0002    A2 12      25          cp      DIVISOR,DIVIDEND_HI
P 0004    BB 02      26          jr      UGT,LOOP !CARRY = 0 (FOR RLC)!
          27 !WON'T FIT. OVERFLOW!
P 0006    DF         28          SCF          !CARRY = 1!
P 0007    AF         29          ret
          30
          31 LOOP:      !RESULT WILL FIT. GO AHEAD WITH DIVISION
          32          RLC  DIVIDEND_LO,DIVIDEND * 2!
          33          RLC  DIVIDEND_HI
          34          jr   c,subt
          35          cp   DIVISOR,DIVIDEND_HI
          36          jr   UGT,next !CARRY = 0!
          37 subt:    SUB  DIVIDEND_HI,DIVISOR
          38          SCF          !TO BE SHIFTED INTO RESULT!
          39 next:    djnz COUNT,LOOP !no flags affected!
          40
          41 !ALL      DONE!
P 0017    10 E3      42          RLC  DIVIDEND_LO
          43                !CARRY = 0: no overflow!
          44          ret
P 0019    AF         45          END DIVIDE
P 001A                    46          END ARITH

```

0 ERRORS  
ASSEMBLY COMPLETE

15 instructions  
26 bytes  
124.5  $\mu$ s (average)

---

## Conclusion

This Application Note has focused on ways in which the Z8 microcomputer can easily yet effectively solve various application problems. In particular, the many sample routines illustrated in this document should

aid the reader in using the Z8 to greater advantage. The major features of the Z8 have been described so that the user can continue to expand and explore the Z8's repertoire of uses.

Copyright 1980, 1981, 1983 by Zilog Inc. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without the prior written permission of Zilog. The information contained herein is published by SGS and subject to change without notice. SGS assumes no responsibility for the use of circuitry embodied in the product. No other circuit patent licences are implied.

**SGS GROUP OF COMPANIES**

**Italy - Brazil - France - Malta - Malaysia - Singapore - Sweden - Switzerland - United Kingdom - U.S.A. - West Germany**

© 1986 SGS, All Rights Reserved - Printed in Italy

\* Z8 is a registered trademark of Zilog Inc.

# SGS OFFICES

## INTERNATIONAL HEADQUARTERS

SGS Microelettronica SpA  
Via C. Olivetti 2, - 20041 Agrate Brianza-Italy  
Tel.: 039 - 65551  
Telex: 330131 - 330141 - SGSAGR

## BENELUX

SGS Microelettronica SpA  
Sales Office:  
**Bruxelles 1040**  
Bld. Reyerslaan, 207 209  
Tel.: 02 - 7366060  
Telex: 24149

## BRAZIL

SGS Semicondutores LTDA  
Sales Office:  
**05413 Sao Paulo**  
Av. Henrique Schaumann 286 - CJ33  
Tel.: 011 - 853-5062  
Telex: 37988 UMBR BR

## DENMARK

SGS Semiconductor A.B.  
Sales Office:  
**2730 Herlev**  
Herlev Torv, 4  
Tel.: 02 - 948533  
Telex: 35411

## FRANCE

Société Générale de Semiconducteurs  
**92120 Montrouge**  
21-23 Rue de la Vanne  
Tel.: 01 - 47460800  
Telex: 250938F

## HOLLAND

SGS Microelettronica SpA  
**5812 CM Eindhoven**  
Kruisstraat, 130  
Tel.: 040 - 433566  
Telex: 51186 SGSEI NL

## HONG KONG

SGS Semiconductor Asia Limited  
**Hunghom, Kowloon**  
9th Floor, Block N,  
Kaiser Estate, Phase III,  
11 Hok Yuen St.,  
Tel.: 03-644251/6  
Telex: 33906 ESGIE HK

## ITALY

SGS Microelettronica SpA  
Direzione Italia e Sud Europa  
**20090 Assago (MI)**  
V.le Milanofiori - Strada 4 - Palazzo A/4/A  
Tel.: 02 - 8244131 (10 linee)  
Telex: 330131 - 330141 SGSAGR

## Sales Offices:

**40033 Casalecchio di Reno (BO)**  
Via R. Fucini 12  
Tel.: 051-591914  
Telex: 226363  
**00161 Roma**  
Via A. Torlonia, 15  
Tel.: 06-8444474

## KOREA

SGS Semiconductor Asia Ltd.  
Korea Liaison Office  
**Mapo, Seoul 121**  
Rm 1306 KMIC Bldg  
168-9 Yumlidong  
Tel.: 712-7071/2/3  
Telex: K 26493

## SINGAPORE

SGS Semiconductor (Pte) Ltd.  
**Singapore 2056**  
28 Ang Mo Kio  
Industrial Park 2  
Tel.: 482-1411  
Telex: RS 55201 ESGIES

## SPAIN

SGS Microelettronica SpA  
**28036 Madrid**  
Representative Office  
Calle Agustín De Foxá, 25  
Tel.: 01 - 7337043  
Telex: 41414

## SWEDEN

SGS Semiconductor A.B.  
**19500 Märsta**  
Bristagatan, 16  
Tel.: 0760 - 40120  
Telex: 054 - 10932

## SWITZERLAND

SGS Semiconductor S.A.  
Sales Offices:  
**1218 Grand-Saconnex (Geneve)**  
Chemin François-Lehmann, 18/A  
Tel.: 022 - 986462/3  
Telex: 28895

## TAIWAN-REPUBLIC OF CHINA

SGS Semiconductor Asia Ltd  
**Taipei Sec 4**  
6th floor, Pacific Commercial Bldg.  
285 Chung Hsiao E Road  
Tel.: 2-7728203  
Telex: 10310 ESGIETWN

## UNITED KINGDOM

SGS Semiconductor Limited  
**Aylesbury, Bucks**  
Planar House, Walton Street  
Tel.: 0296 - 5977  
Telex: 051 - 83245

## WEST GERMANY

SGS Halbleiter Bauelemente GmbH  
**8018 Grafing bei München**  
Haidling, 17  
Tel.: 08092-690  
Telex: 05 27378

## Sales Offices:

**3012 Langenhagen**  
Hans Boeckler Str., 2  
Tel.: 0511 - 789881  
Telex: 923195  
**8500 Nürnberg 40**  
Allersberger Str., 95  
Eingang Wilhelmstr. 1  
Tel.: 0911 - 464071  
Telex: 626243

## 8023 Pullach bei München

Seitnerstrasse, 42  
Tel.: 089 - 793 0662  
Telex: 5215784  
**7000 Stuttgart 31**  
Loewenmarkt, 5  
Tel.: 0711 - 881101  
Telex: 723625

## U.S.A.

SGS Semiconductor Corporation  
**Phoenix, AZ 85022**  
1000 East Bell Road  
Tel.: (602) 867-6100  
Telex: 249976 SGSPH UR

## Sales Offices:

**Bloomington, MN 55420**  
One Appletree Square  
Suite 201-K  
Tel.: (612) 854-0525  
**Ft. Lauderdale FL 33309**  
1001 NW 62nd Street  
Suite 314  
Tel.: (305) 4938881  
Telex: 291588

## Hauppauge, NY 11788

330 Motor Parkway  
Suite 100  
Tel.: (516) 435-1050  
Telex: 221275 SGSHA UR  
**Indianapolis, IN 46268**  
8777 Purdue Road  
Suite 113  
Tel.: (317) 872-4404  
Telex: 209144 SGSIN UR

## Irvine, CA 92714

18271 W. McDermott Drive  
Suite J.  
Tel.: (714) 863-1222  
Telex: 277793 SGSOR UR  
**Plano, TX 75074**  
850 East Central Parkway  
Suite 180  
Tel.: (214) 881-0848  
Telex: 203997 SGSDA UR  
**Poughkeepsie, NY 12601**  
201 South Avenue  
Suite 206  
Tel.: (914) 473-2255

## Santa Clara, CA 95051

2700 Augustine Drive  
Suite 209  
Tel.: (408) 727-3404  
Telex: 278833 SGSSA UR  
**Schaumburg, IL 60196**  
600 North Meacham Road  
Tel.: (312) 490-1890  
Telex: 210159 SGSCH UR  
**Southfield, MI 48076**  
21411 Civic Center Dr. 309  
Mark Plaza Bldg.  
Tel.: (313) 358-4250  
Telex: 810-224-4684 "MGA DET SOFD"  
**Waltham, MA 02154**  
240 Bear Hill Road  
Tel.: (617) 890-6688  
Telex: 200297 SGSWH UR



Technology  
and Service

ORDER CODE: DAZ8DB/3