CHAPTER I

INTRODUCTION

CONTENTS

INTRODUCING THE 2650 FAMILY

"5—VOLT SYSTEM REDUCES SYSTEM COSTS"
"2650 PUTS THE INTERFACE ON THE CHIP...NOT ON THE CIRCUIT BOARD"
"POWERFUL INSTRUCTION SET PROVIDES LOWER COST SYSTEMS"

The greatly increased sophistication and rising production costs of today's logic systems force the system designer to use every available resource in order to economically produce his system. In keeping with this cost reduction goal, Signetics has developed a powerful general purpose integrated microprocessor called the 2650. The first Signetics microprocessor, in conjunction with Signetics MOS and Bipolar memory and interface product lines, offers the system designer a viable and attractive alternative to the hard-wired approach to system design. For many applications, the system designer can use this general purpose microprocessor and standard memory and interface circuits to implement systems with lower cost than the hard-wired logic approach without sacrificing performance.

By using the 2650 and compatible products, the system designer can obtain two other major benefits of microcomputer systems. These benefits are greatly enhanced system flexibility and minimized design or modification cycles compared with the hard-wired logic approach.

The requirements of the majority of applications for integrated microprocessors (logic replacement and control functions) have defined a general set of processor parameters based on system and device economies, ease of use, and speed requirements.

These characteristics include:

- Single chip
- Fixed instruction set
- Eight bit parallel structure
- TTL compatibility

In addition to these characteristics, the design of the 2650 has been optimized around three generalized objectives:

- Lowest system cost
- Ease of use
- Capable of a wide range of applications

The optimum technology choice for implementing these features is the low threshold ion-implanted N-Channel silicon gate process. This process has matured in the past few years, providing a combination of high density, low threshold voltage, moderate speed and good manufacturing yields. Using this technology, a total of 576 bits of ROM, approximately 250 bits of register and about 900 logic gates are used to implement the processor function on the 2650 chip.

The instruction set consists of 75 instructions, of which about 40% consists of arithmetic instructions. This class contains the Boolean, arithmetic, and compare operations, each of which may be executed using any one of eight addressing modes. Another 30% of the instruction set consists of branch instructions which incorporate six addressing modes. The remaining 30% of the instruction set includes, amoung others, I/O instructions, instructions for performing operations on the two status registers, a decimal adjust instruction and the HALT instruction.

Utilizing multiple addressing modes greatly increases coding efficiency, allowing functions to be performed using fewer instructions than less powerful machines. The resulting reduction in routine execution time and memory capacity requirements directly translates into improved system performance and reduced memory cost. In this way the powerful instruction set and addressing modes of the 2650 allow a significant reduction in the memory required to perform a given function, resulting in sizeable system cost savings without sacrificing performance.

FEATURES OF THE 2650 FAMILY

2650 FAMILY APPROACH

- Low System Cost
  - — Low cost N-Channel products
  - — Intrinsic advantages of single 5V supply
  - — Uses standard low cost memories
  - — Low cost interfacing
- Ease of Use
  - — Easy interfacing
  - — Conventional instruction set
  - — Ease of programming
- Wide Range of Applications
  - — General purpose capability
  - — Powerful architecture
  - — Powerful instruction set
  - — Flexible
  - — Expanding family of devices

FEATURES OF THE MICROPROCESSOR

Basic 2650 Processor Characteristics

- Single chip 8-bit processor
- Signetics low threshold double ion-implanted silicon gate N-Channel technology
- Single +5V power supply
- Low power consumption: 525 mW maximum
- Single phase TTL-compatible clock
- Static operation: no minimum clock frequency
- Clock frequency: 1.25MHz maximum
- Cycle time: 2.4μs minimum
- Standard 40 pin DIP

2650 Interfaces

- TTL compatible inputs, outputs — no external resistors required
- Tri-state bus outputs for multiprocessor and direct memory access systems
- Asynchronous (handshaking) memory and I/O interface
- Accepts wide range of memory timing
- Interfaces directly with industry standard memories
- Powerful control interface
- Single-bit direct serial I/O path
- Parallel 8-bit I/O capability

2650 Processor Architecture

- 8-bit bidirectional tri-state data bus
- Separate tri-state address bus
- 32,768-byte addressing range
- Internal 8-bit parallel structure
- Seven 8-bit addressable general purpose registers
- Eight-level on-chip subroutine return address stack
- Program status word for flexibility and enhanced processing power
- Single-level hardware vectored interrupt capability
- Interrupt service routines may be located anywhere in addressable memory
- Separate adder for fast address calculation

2650 Instruction Set

- General purpose instruction set with substantial capabilities in arithmetic, character manipulation and control and I/O processing
- Fixed instruction set
- 75 instructions
- Up to eight addressing modes
- True indexing with optional auto increment/decrement
- One, two or three byte instructions
- One- and two-byte I/O instructions
- Selective test of individual bits
- Powerful instruction set and addressing modes minimize memory requirements

FEATURES OF COMPATIBLE PRODUCTS

2602, 2606, 1K RAMs

- Completely static operation
- N-Channel silicon gate technology
- 1024 X 1 organization (2602)
- 256 X 4 organization (2606)
- Single +5V power supply
- 200mW typical power dissipation
- Maximum access time:
  - 1μs : 2602
  - 750ns : 2606
  - 650ns : 2602-2
  - 500ns : 2602-1, 2606-1
- TTL-compatible
- Tri-state outputs
- Data I/O bus (2606 only)
- Standard 16 pin DIP

2608 8K ROM

- Completely static operation
- N-Channel silicon gate technology
- 1024 X 8 organization
- Single +5V power supply
- 400mW maximum power dissipation
- 650ns maximum access time
- TTL compatible
- Tri-state outputs
- Standard 24 pin DIP

8T26 Quad Transceiver

- Schottky TTL technology
- Four pairs of bus drivers/receivers
- Separate drive and receive enable lines
- Tri-state outputs
- Low current pnp inputs
- High fan out — driver sinks 40mA
- 20ns maximum propagation delay
- Standard 16 pin DIP

8T31 8-bit Bidirectional Port

- Schottky TTL technology
- Two independent bidirectional busses
- Eight bit latch register
- Independent read, write controls for each bus
- Bus A overrides if a write conflict occurs
- Register can be addressed as a memory location via Bus B Master Enable
- 30ns maximum propagation delay
- Low input current: 500μA
- High fan out — sinks 20mA
- Standard 24 pin DIP

8T95/6/7/8 Hex Buffers/Inverters

- Schottky TTL technology
- Six buffers or inverters per package
- Non-inverting (8T95, 8T97) or Inverting (8T96, 8T98)
- Buffered control lines
- Tri-state outputs
- Low current pnp inputs
- Standard 16 pin DIP

82S115/123/129 PROMs

- Schottky TTL technology
- Single +5V power supply
- 32 X 8 organization (82S123)
- 256 X 4 organization (82S129)
- 512 X 8 organization (82S115)
- Field programmable (Nichrome)
- On-chip storage latches (82S115 only)
- Low current pnp inputs
- Tri-state outputs
- 35ns typical access time
- Standard 24 pin DIP (82S115)
- Standard 16 pin DIP (82S123, 82S129)

(See Appendix for additional products and data sheets.)

PROCESSOR HARDWARE DESCRIPTION

ARCHITECTURE

GENERAL DESCRIPTION

A block diagram of the processor is shown in Figure 1. The first, second, and third bytes of instructions are read into the processor on the data bus and loaded into the Instruction Register, Holding Register, and Data Bus Register, respectively. The instructions are decoded through a combination of ROM and random logic.

The ALU performs arithmetic, Boolean, and combinatorial shifting functions. It operates on eight bits in parallel and utilizes carry-look-ahead logic. A second adder is used to increment the instruction address register and to calculate operand addresses for the indexed and relative addressing modes. This separate address adder allows complex addressing modes to be implemented with no increase in instruction execution time.

The General Purpose Register Stack and the Subroutine Return Address Stack are implemented with static RAM cells. The Register Stack consists of seven 8-bit registers. The Subroutine Stack can contain eight 15-bit addresses, thereby allowing eight levels of subroutine nesting. Placing the Subroutine Stack on the chip allows efficient ROM-only systems to be implemented in some applications. Separate 15-bit Instruction Address and Operand Address Registers and provided. The 2650 is an 8-bit binary processor with BCD capability. See Figure 2 for a diagram of the 2650 registers as seen by the programmer.

PROGRAM STATUS WORD

The Program Status Word (PSW) is a major feature of the 2650 with greatly increases its flexibility and processing power. The PSW is a special purpose register within the processor that contains status and control bits.

It is divided into two bytes called the Program Status Upper (PSU) and Program Status Lower (PSL). The PSW bits may be tested, loaded, stored, preset, or cleared using the instructions which affect the PSW. The bits are utilized as follows:

PSU0, 1,2	— SP	— **Pointer** for the Return Address **Stack.**
PSU5	— II	— Used to **Inhibit** recognition of additional **Interrupts.**
PSU6	— F	— **Flag** is a latch directly driving the flag output.
PSU7	— S	— **Sense** equals the state of the sense input.
PSL0	— C	— **Carry** stores any carry from the high-order bit of the ALU.
PSL1	— COM	— **Compare** determines if a logical or arithmetic comparison is to be made.
PSL2	— OVF	— **Overflow** is set if a two's complement overflow occurs.
PSL3	— WC	— **With Carry** determines is the carry is used in arithmetic and rotate instructions.
PSL4	— RS	— **Register Select** identifies which bank of 3 GP registers is being used.
PSL5	— IDC	— **Inter Digit Carry** stores the bit-3-to-bit-4 carry in arithmetic operations.
PSL6, 7	— CC	— **Condition Code** is affected by compare, test and arithmetic instructions.

INTERRUPT HANDLING CAPABILITY

The 2650 has a single level hardware vectored interrupt capability. When an interrupt occurs, the 2650 finishes the current instruction and sets the

Interrupt Inhibit bit in the PSW. The processor then executes a Branch to Subroutine Relative to location Zero (ZBSR) instruction and sends out Interrupt Acknowledge and Operation Request signals. On receipt of the INTACK signal the interrupting device inputs an 8-bit address, the interrupt vector, on the data bus. The relative and relative indirect addressing modes combined with this 8-bit address allow interrupt service routines to begin at any addressable memory location.

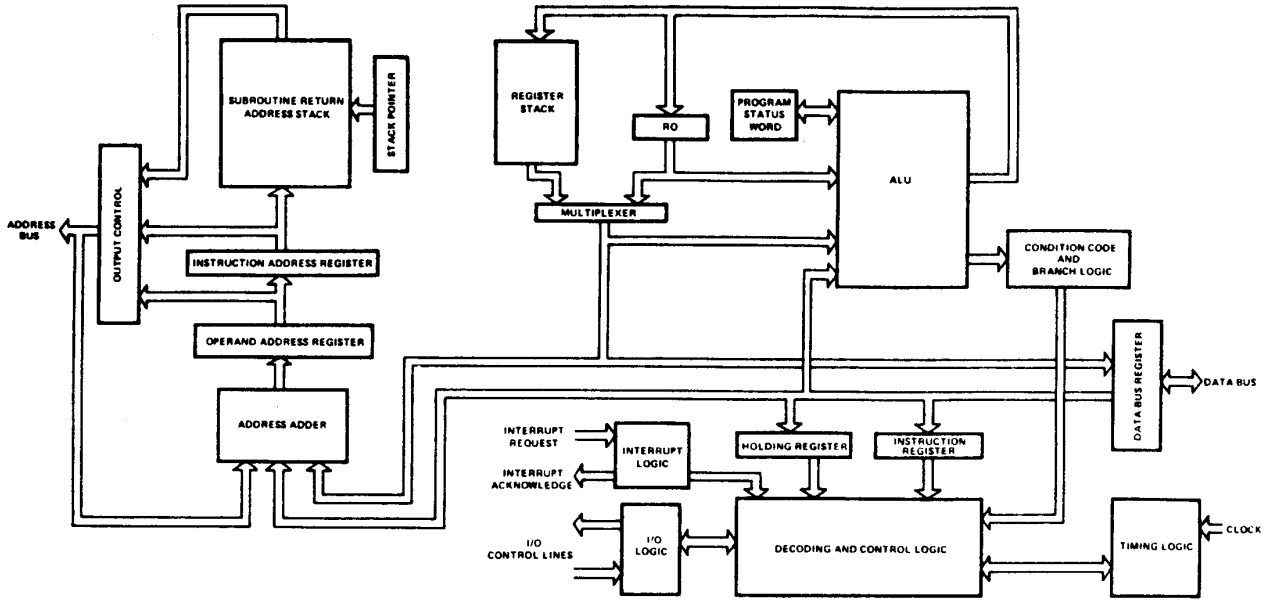
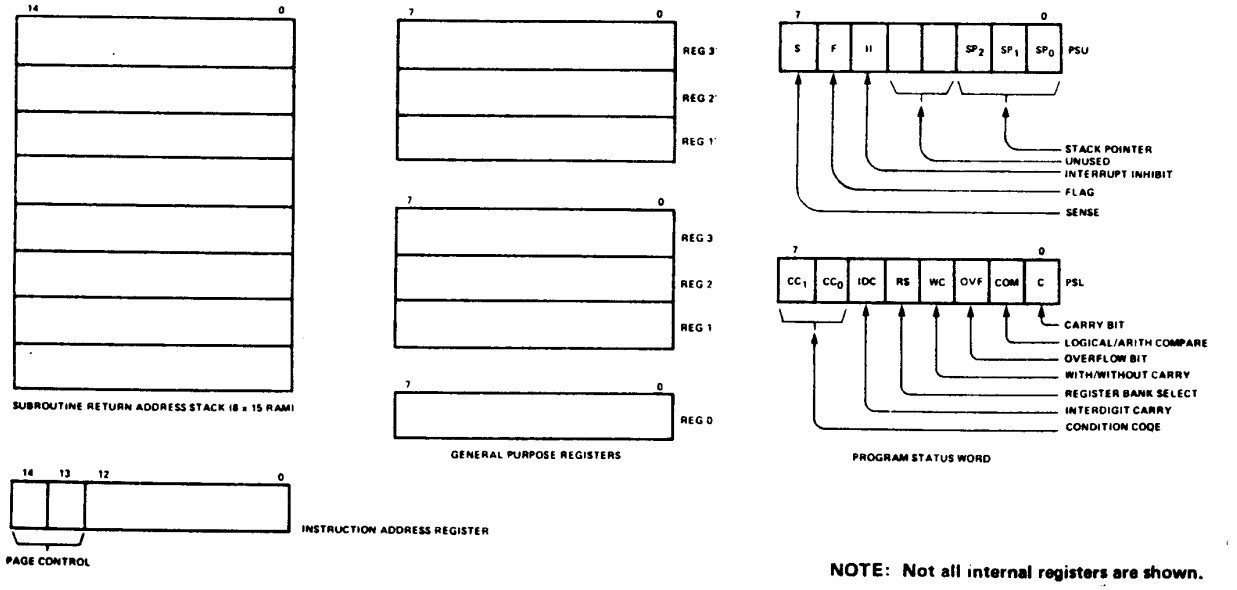


Figure 1. BLOCK DIAGRAM



NOTE: Not all internal registers are shown.

Figure 2. MAJOR 2650 REGISTERS

INTERFACING

INTRODUCTION TO INTERFACING WITH THE 2650

Five key concepts have been incorporated in the 2650 to make interfacing easy and inexpensive. The extent to which these concepts have been incorporated in the Signetics 2650 provides unique benefits of system density and low cost to the system designer.

1. SINGLE 5V POWER SUPPLY

Low threshold double ion-implanted Silicon Gate N-Channel MOS technology is used to allow operation from one +5V power supply with resultant cost savings and improved reliability. This reduces power consumption significantly compared with the multi-power supply approach.

2. INTERFACE CIRCUIT COMPATIBILITY

The 2650 inputs and outputs are specified to be compatible with widely available, standard, low cost logic families such as TTL, CMOS and Low-power STTL. This includes the single phase clock input which saves the cost of high level multiphase clock driver circuitry. Bus outputs are tri-state and capable of driving one 7400 TTL load or four 74LS loads. The 2650 is capable of driving several loads of pnp-buffered STTL inputs. Many MSI, Interface and Memory LSI circuits (for example, in Signetics 82S00 and 8T00 series) have these low current pnp inputs and are recommended for use in 2650 microcomputer systems. See Table 1 for DC characteristics of the 2650.

3. USE OF STANDARD MEMORIES

One of the major 2650 design achievements is to operate efficiently in a system using industry standard memories, for example 1024 X 1 and 256 X 4 N-channel RAMs and 1024 X 8 N-Channel ROMs. These standard memories are widely available and used in volume with corresponding low cost. Non-standard memories, particularly those produced by only one manufacturer will be less available, run in lower volume and often cost 2 to 3 times as much per bit as industry standard products. The 2650 operates successfully with memories of any access time, due to the completely asynchronous interface that is provided for this purpose. Memories which respond in less than 0.8 microseconds allow the processor to operate at maximum speed.

4. NO SPECIAL INTERFACE PRODUCTS

Similarly, another major achievement is to operate efficiently in a system using no special I/O products. This approach avoids the problems of a system requiring high cost specialized components with restricted availability.

TABLE 1. PRELIMINARY 2650 DC ELECTRICAL CHARACTERISTICS

			LIMITS		
SYMBOL	PARAMETER	TEST CONDITIONS	MIN	MAX	UNIT
I_{LI}	Input Load Current	V_{IN} = 0 to 5.25V		10	μA
I_{LOH}	Output Leakage Current	ADREN, DBUSEN = 2.2V, V_{OUT} = 4V		10	μA
I_{LOL}	Output Leakage Current	ADREN, DBUSEN = 2.2V, V_{OUT} = 0.45V		10	μA
I_{CC}	Power Supply Current	V_{CC} = 5.25V, T_A = 0°C		100	mA
V_{IL}	Input Low		-0.6	0.8	V
V_{IH}	Input High		2.2	V_{CC}	V
V_{OL}	Output Low	I_{OL} = 1.6 mA	0.0	0.45	V
V_{OH}	Output High	I_{OH} = -100 μA	2.4	V_{CC}-0.5	V
C_{IN}	Input Capacitance	V_{IN} = 0V		10	pF
C_{OUT}	Output Capacitance	V_{OUT} = 0V		10	pF

Conditions: T_A = 0°C to 70°C, V_{CC} = 5V ±5%

5. POWERFUL MEMORY AND I/O INTERFACE

The following features characterize the memory and I/O interfaces:

- Both memory and input/output may operate in a completely asynchronous fashion. Consequently, devices operating at any speed up to the maximum data transfer rate may be connected without buffering. External latching of data from these interfaces is not required.
- Data paths are driven with tri-state buffers, allowing multiprocessor and Direct Memory Access (DMA) configurations to be designed.
- Eight-bit data paths communicate data in parallel.
- One- and two-byte I/O instructions provide maximum flexibility and efficiency when interfacing with I/O devices.

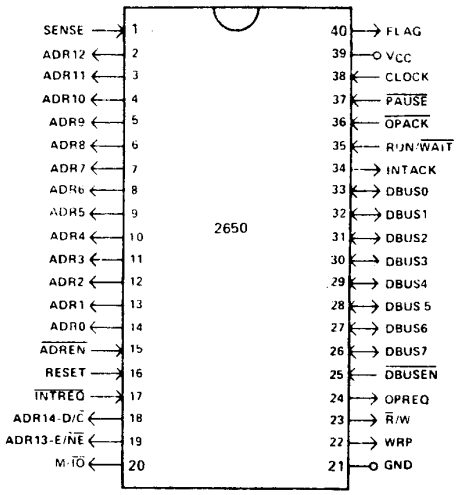


Figure 3. PIN CONFIGURATION

PIN CONFIGURATION AND INTERFACE SIGNAL DEFINITION

Refer to Figure 3 for the 2650 pin configuration. Signals are defined as follows:

ADR0-ADR12 — The low order 13 bits of address for memory access are on these pins. ADR0-ADR7 are also used in two-byte I/O instructions. These outputs are tri-state buffers controlled by ADREN.

ADR13-E/NE — This multiplexed output signal delivers the ADR13 address bit when M/IO is in the M phase or discriminates between Extended and Non-Extended I/O instructions when M/IO is in the I/O phase.

ADR14-D/C — Address 14 or Data/Control is a multiplexed output signal. This pin delivers the ADR14 address bit when M/IO is in the M phase or discriminates between Data and Control I/O instructions when M/IO is in the I/O phase.

ADREN — Address Bus Enable is an input providing the external control for the ADR0-ADR12 tri-state buffer drivers.

DBUS0-DBUS7 — This is the 8-bit, bidirectional tri-state bus over which most data is communicated into or out of the processor.

$\overline{\text{DBUSEN}}$ — Data Bus Enable is an input that controls the tri-state buffer drivers for DBUS0 to DBUS7.

OPREQ — Operation Request is an output signal that informs external devices that the information on other output pins is valid.

OPACK	— Operation Acknowledge is an input which is used by external devices to end an I/O or memory signaling sequence.
M/IO	— Memory/Input-Output. This output informs external devices whether Memory or Input/Output functions are being performed.
R/W	— This output signal describes an I/O or memory operation as Read or Write, and defines whether the bidirectional DBUS is transmitting or receiving.
WRP	— This Write Pulse is generated during write sequences and may be used to strobe memory or I/O devices.
SENSE	— Is an input, independent of the other I/O signals, that provides a direct input to the processor.
FLAG	— This pin provides a direct output signal that is completely independent of the other I/O signals.
INTREQ	— Interrupt Request. This input is used by external devices to force the processor into the Interrupt sequence.
INTACK	— Interrupt Acknowledge is the signal used by the processor to inform external devices that it has entered an interrupt sequence.
PAUSE	— Pause is used to temporarily stop the processor at the end of the current instruction. It may stop processing for an indefinite length of time and is available to use for DMA (Direct Memory Access).
RUN/WAIT	— Informs external circuits as to the Run/Wait status of the 2650 processor.
RESET	— Is an input used to cause the 2650 to begin processing from a known state.
CLOCK	— This is the only clock input to the processor. It accepts standard TTL levels.
VCC	— +5V power.
GND	— The logic and power supply ground for the processor.

2650 TIMING

The clock input to the 2650 provides the basic timing information that the processor uses for all its internal and external operations. The clock rate determines the instruction execution time, except to the extent that external memories and devices slow the processor down. The maximum clock rate of the standard 2650 is 1.25 Megacycles (one clock period is 800ns minimum). One unique feature of the 2650 is that the clock frequency may be slowed down to DC, allowing complete timing flexibility for interfacing. This feature permits single stepping the clock which can greatly simplify system checkout. It also provides an easy method to halt the processor. Each 2650 cycle is comprised of three clock periods. Direct instructions require either 2, 3, or 4 processor cycles for execution and, therefore, vary from 4.8 to 9.6μs in duration.

A timing diagram for a memory read cycle is shown in Figure 4. OPREQ (Operation Request) is the master control signal that coordinates all operations external to the processor. When true, OPREQ indicates that other output signals are valid. During a memory read cycle M/IO is in the M (Memory) state and R/W is in the R (Read) state. The address lines and the control lines become valid before OPREQ rises. The data to be read may be returned anytime after OPREQ becomes valid. An OPACK (Operation Acknowledge) should accompany the read data from the memory. The Data and OPACK signals should remain valid for 50 ns after OPREQ falls.

INPUT/OUTPUT INTERFACE

The 2650 microprocessor has a set of versatile I/O instructions and can perform I/O operations in a variety of ways. One- and two-byte I/O instructions are provided, as well as a special single-bit I/O facility. The I/O modes provided by the 2650 are designated as Data, Control, and Extended I/O.

Data or Control I/O instructions are one byte long. Any general purpose register can be used as the source or destination. A special control line indicates if either a Data or Control instruction is being executed. Extended I/O is a two-byte read or write instruction. Execution of an extended I/O instruction will cause an 8-bit address, taken from the second byte of the instruction, to be placed on the low order eight address lines. The data, which can originate or terminate with any general purpose register, is placed on the data bus. This type of I/O can be used to simultaneously select a device and send data to it.

Memory reference instructions that address data outside of physical memory may also be used for I/O operations. When an instruction is executed, the address may be decoded by the I/O device rather than memory.

MEMORY INTERFACE

The memory interface consists of the address bus, the 8-bit data bus and several signals that operate in an interlocked or handshaking mode.

The Write Pulse signal is designed to be used as a memory strobe signal for any memory type. It has been particularly optimized to be used as the Chip Enable or Read/Write signal for the Signetics 2602 and 2606 RAMs.

INTERFACING – A MINIMAL SYSTEM EXAMPLE

The 2650 has been designed for low cost, easy interfacing, which is dramatically illustrated by a minimal system configuration shown in Figure 5. This system has a Teletype interface, 1024 bytes of ROM, and 256 bytes of RAM, yet requires only seven (7) standard integrated circuit packages. The ROM can contain a bootstrap loader and I/O driver programs for the Teletype. Other programs could reside in ROM or be read into RAM via the Teletype. An alternative to the 2608 N-Channel MOS ROM is the 82S115 Bipolar PROM which offers a 512 X 8 organization. Only one +5-volt power supply is required for this system. The advantages of conceptual simplicity and minimum system costs of the 2650 approach will be obvious to the system designer, particularly when compared to alternative microprocessor products.

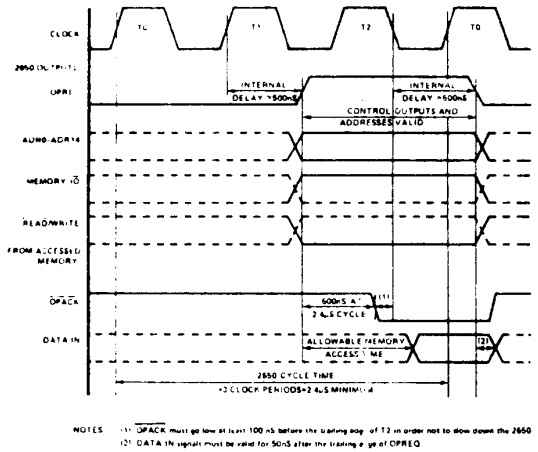


Figure 4. MEMORY READ CYCLE TIMING

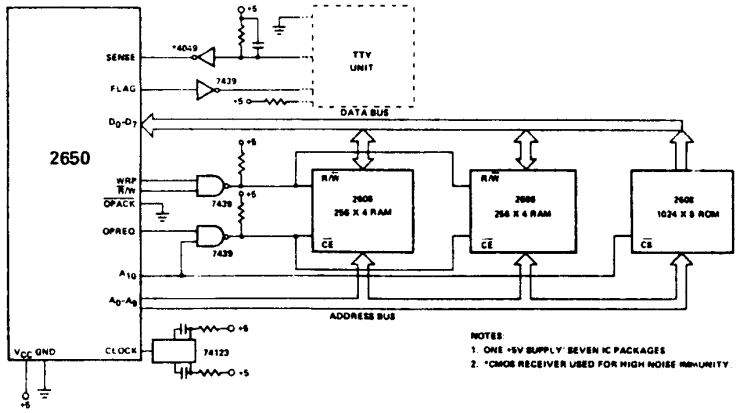


Figure 5. SEVEN PACKAGE MINIMAL SYSTEM

INSTRUCTION SET

It may be seen from examination of the 2650 instruction set that there are many powerful instructions which are all easily understood and are typical of larger computers. There are one-, two-, and three-byte instructions as a result of the multiplicity of addressing modes. See Table 2 for a complete listing and Figure 6 for instruction formats.

Automatic incrementing or decrementing of an index register is available in the arithmetic indexed instructions. All of the branch instructions except indexed branching can be conditional.

Register-to-register instructions are one byte; register-to-storage instructions are two or three bytes long. The two-byte register-to-memory instructions are either immediate or relative addressing types.

TABLE 2. INSTRUCTION SET

	MNEMONIC		OP CODE	FORMAT*	DESCRIPTION OF OPERATION	AFFECTS	CYCLES
LOAD/STORE	LOD	Z	000 000	1Z	Load Register Zero	CC (Note 1)	2
		I	000 001	2I	Load Immediate	CC (Note 1)	2
		R	000 010	2R	Load Relative	CC (Note 1)	3
		A	000 011	3A	Load Absolute	CC (Note 1)	4
	STR	Z	110 000	1Z	Store Register Zero (r ≠ 0)	CC (Note 1)	2
		R	110 010	2R	Store Relative	–	3
		A	110 011	3A	Store Absolute	–	4
ARITHMETIC	ADD	Z	100 000	1Z	Add to Register Zero w/wo Carry	C, CC (Note 1), IDC, OVF	2
		I	100 001	2I	Add Immediate w/wo Carry	C, CC (Note 1), IDC, OVF	2
		R	100 010	2R	Add Relative w/wo Carry	C, CC (Note 1), IDC, OVF	3
		A	100 011	3A	Add Absolute w/wo Carry	C, CC (Note 1), IDC, OVF	4
	SUB	Z	101 000	1Z	Subtract from Register Zero w/wo Borrow	C, CC (Note 1), IDC, OVF	2
		I	101 001	2I	Subtract Immediate w/wo Borrow	C, CC (Note 1), IDC, OVF	2
		R	101 010	2R	Subtract Relative w/wo Borrow	C, CC (Note 1), IDC, OVF	3
		A	101 011	3A	Subtract Absolute w/wo Borrow	C, CC (Note 1), IDC, OVF	4
	DAR		100 101	1Z	Decimal Adjust Register	CC (Note 2)	3
LOGICAL	AND	Z	010 000	1Z	AND to Register Zero (r ≠ 0)	CC (Note 1)	2
		I	010 001	2I	AND Immediate	CC (Note 1)	2
		R	010 010	2R	AND Relative	CC (Note 1)	3
		A	010 011	3A	AND Absolute	CC (Note 1)	4
	IOR	Z	011 000	1Z	Inclusive OR to Register Zero	CC (Note 1)	2
		I	011 001	2I	Inclusive OR Immediate	CC (Note 1)	2
		R	011 010	2R	Inclusive OR Relative	CC (Note 1)	3
		A	011 011	3A	Inclusive OR Absolute	CC (Note 1)	4
	EOR	Z	001 000	1Z	Exclusive OR to Register Zero	CC (Note 1)	2
		I	001 001	2I	Exclusive OR Immediate	CC (Note 1)	2
		R	001 010	2R	Exclusive OR Relative	CC (Note 1)	3
		A	001 011	3A	Exclusive OR Absolute	CC (Note 1)	4
COMPARE	COM	Z	111 000	1Z	Compare to Register Zero Arithmetic/Logical	CC (Note 3)	2
		I	111 001	2I	Compare Immediate Arithmetic/Logical	CC (Note 4)	2
		R	111 010	2R	Compare Relative Arithmetic/Logical	CC (Note 4)	3
		A	111 011	3A	Compare Absolute Arithmetic/Logical	CC (Note 4)	4
ROTATE	RRR		010 100	1Z	Rotate Register Right w/wo Carry	C, CC, IDC, OVF	2
	RRL		110 100	1Z	Rotate Register Left w/wo Carry	C, CC, IDC, OVF	2
BRANCH	BCT	R	000 110	2R	Branch On Condition True Relative	–	3
		A	000 111	3B	Branch On Condition True Absolute	–	3
	BCF	R	100 110	2R	Branch On Condition False Relative	–	3
		A	100 111	3B	Branch On Condition False Absolute	–	3
	BRN	R	010 110	2R	Branch On Register Non-Zero Relative	–	3
		A	010 111	3B	Branch On Register Non-Zero Absolute	–	3
	BIR	R	110 110	2R	Branch On Incrementing Register Relative	–	3
		A	110 111	3B	Branch On Incrementing Register Absolute	–	3
	BDR	R	111 110	2R	Branch On Decrementing Register Relative	–	3
		A	111 111	3B	Branch On Decrementing Register Absolute	–	3
	ZBRR		100 110 11	2ER	Zero Branch Relative, Unconditional	–	3
	BXA		100 111 11	3E8	Branch Indexed Absolute, Unconditional (Note 5)	–	3

TABLE 2. INSTRUCTION SET (CONTINUED)

	MNEMONIC	OP CODE	FORMAT*	DESCRIPTION OF OPERATION	AFFECTS	CYCLES
SUBROUTINE BRANCH/RETURN	BST R	001 110	2R	Branch To Subroutine On Condition True, Relative	SP	3
	BST A	001 111	3B	Branch To Subroutine On Condition True, Absolute	SP	3
	BSF R	101 110	2R	Branch To Subroutine On Condition False, Relative	SP	3
	BSF A	101 111	3B	Branch To Subroutine On Condition False, Absolute	SP	3
	BSN R	011 110	2R	Branch To Subroutine On Non-Zero Register, Relative	SP	3
	BSN A	011 111	3B	Branch To Subroutine On Non-Zero Register, Absolute	SP	3
	ZBSR	101 110 11	2ER	Zero Branch To Subroutine Relative, Unconditional	SP	3
	BSXA	101 111 11	3EB	Branch To Subroutine, Indexed, Absolute Unconditional (Note 5)	SP	3
	RET C	000 101	1Z	Return From Subroutine, Conditional	SP	3
	RET E	001 101	1Z	Return From Subroutine and Enable Interrupt, Conditional	SP, II	3
INPUT/OUTPUT	WRTD	111 100	1Z	Write Data	–	2
	REDD	011 100	1Z	Read Data	CC (Note 1)	2
	WRTC	101 100	1Z	Write Control	–	2
	REDC	001 100	1Z	Read Control	CC (Note 1)	2
	WRTE	110 101	2I	Write Extended	–	3
	REDE	010 101	2I	Read Extended	CC (Note 1)	3
MISC.	HALT	010 000 00	1E	Halt, Enter Wait State	–	2
	NOP	110 000 00	1E	No Operation	–	2
	TMI	111 101	2I	Test Under Mask Immediate	CC (Note 6)	3
PROGRAM STATUS	LPS U	100 100 10	1E	Load Program Status, Upper	F, II, SP	2
	LPS L	100 100 11	1E	Load Program Status, Lower	CC, IDC, RS, WC, OVF, COM, C	2
	SPS U	000 100 10	1E	Store Program Status, Upper	CC (Note 1)	2
	SPS L	000 100 11	1E	Store Program Status, Lower	CC (Note 1)	2
	CPS U	011 101 00	2EI	Clear Program Status, Upper, Masked	F, II, SP	3
	CPS L	011 101 01	2EI	Clear Program Status, Lower, Masked	CC, IDC, RS, WC, OVF, COM, C	3
	PPS U	011 101 10	2EI	Preset Program Status, Upper, Masked	F, II, SP	3
	PPS L	011 101 11	2EI	Preset Program Status, Lower, Masked	CC, IDC, RS, WC, OVF, COM, C	3
	TPS U	101 101 00	2EI	Test Program Status, Upper, Masked	CC (Note 6)	3
	TPS L	101 101 01	2EI	Test Program Status, Lower, Masked	CC (Note 6)	3

*FORMAT CODE: The number indicates the number of bytes. The letter(s) indicate the format type(s). See Fig. 6.

NOTES:
1. Condition code (CC1, CC0): 01 if positive, 00 if zero, 10 if negative.
2. Condition code is set to a meaningless value.
3. Condition code (CC1, CC0): 01 if R0 > r, 00 if R0 = r, 10 if R0 < r.
4. Condition code (CC1, CC0): 01 if r > V, 00 if r = V, 10 if r < V.
5. Index register must be register 3 or 3'.
6. Condition code (CC1, CC0): 00 if all selected bits are 1s, 10 if not all the selected bits are 1s.

PROGRAM STATUS WORD

PSU

7	6	5	4	3	2	1	0
S	F	II	Not Used	Not Used	SP2	SP1	SP0

S Sense
F Flag
II Interrupt Inhibit
SP2 Stack Pointer Two
SP1 Stack Pointer One
SP0 Stack Pointer Zero

PSL

7	6	5	4	3	2	1	0
CC1	CC0	IDC	RS	WC	OVF	COM	C

CC1 Condition Code One
CC0 Condition Code Zero
IDC Interdigit Carry
RS Register Bank Select
WC With/Without Carry
OVF Overflow
COM Logical/Arith. Compare
C Carry/Borrow

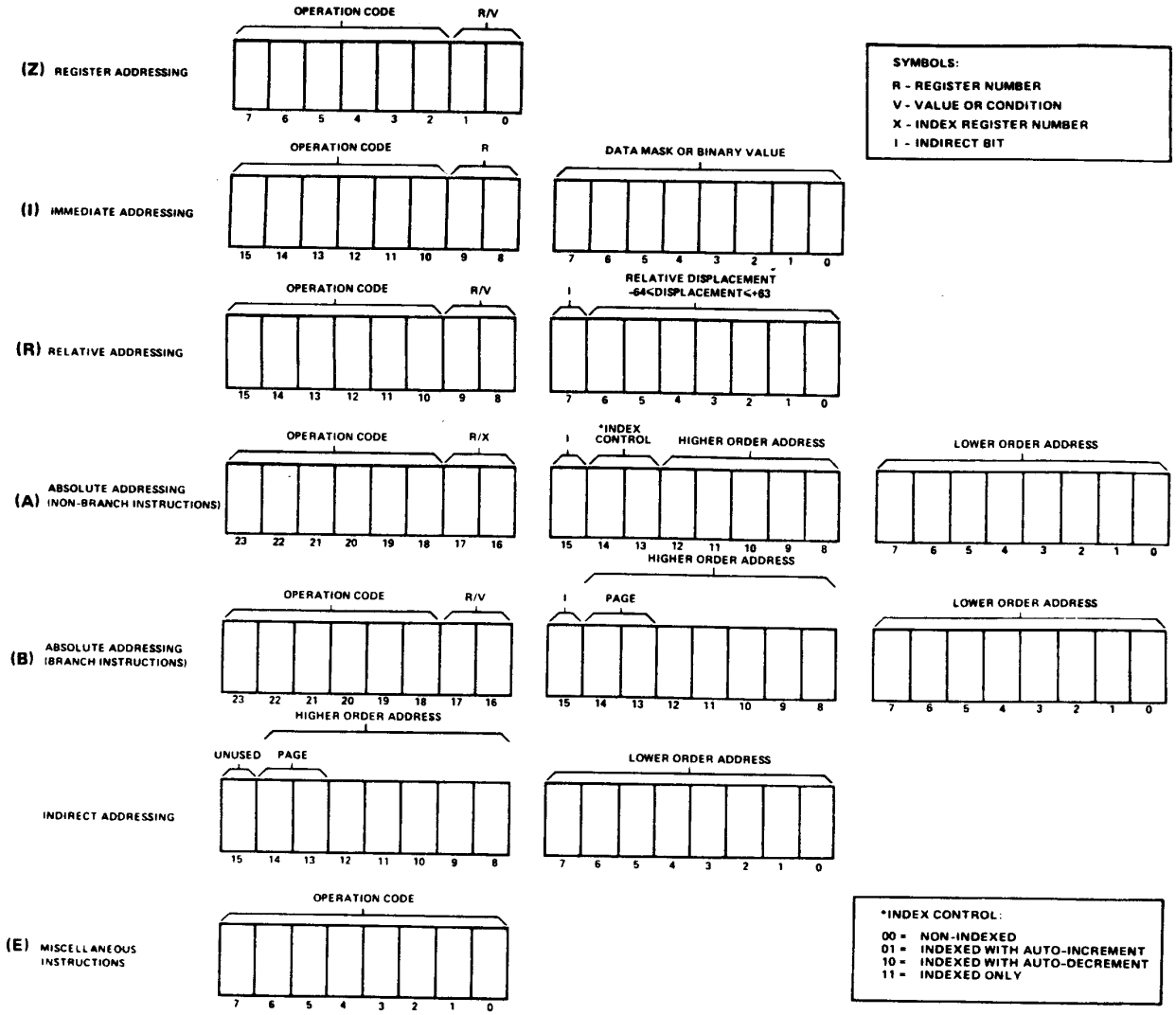


Figure 6. INSTRUCTION FORMATS

SUPPORT

DOCUMENTATION

The complete manual set is available in a durable 3-ring binder. The binder contains the Hardware Specifications, the Assembler Language Manual, the Software Simulator Manual, and a section called System Application Notes. Our update service provides customers with new application notes and updates to the manual set.

The **Hardware Specification Manual** includes a detailed description of the instruction set, the pin-outs, the AC and DC electrical characteristics, the Input/Output and memory interface signals with timing diagrams, the internal processor organization, and other useful information.

The **Assembler Language Manual** describes how to write programs in the 2650 symbolic assembly language, the pseudo-ops, and how to assemble a 2650 program. Additional information is presented on how to use the assembler program, how to interpret the output listings and how to load object modules.

The **Simulator Manual** describes the nature of the simulation program, how to write simulation commands and how to interpret the simulation output.

System Application Notes are included to help the user design with the 2650 processor. These notes present detailed technical information on various subjects of interest and apply to either programming, hardware configuration, or system concepts. This section will continue to grow.

Examples of Application Notes are:

- Serial I/O for the 2650
- Memory Interfaces
- How to use the Decimal Adjust instruction
- I/O Device Selection Methods
- A Minimal System Configuration

SOFTWARE SUPPORT

Signetics-developed software is available to both the batch processing user and the timesharing user. The Batch Assembler and Batch Simulator are written in standard FORTRAN and may be compiled and executed on most medium to large scale computer systems. Because of the modular design used, it is expected that many minicomputer users will also be able to utilize these programs. The main features of the programs are listed in Tables 3 and 4.

Signetics has also made the Batch Assembler, Batch Simulator and Interactive Simulator available on several international timesharing networks for those customers who wish to run these programs using a timesharing service.

When a customer chooses to follow the timesharing approach, he can also make use of the interactive version of the 2650 Simulator. With the Inter-

TABLE 3. ASSEMBLER FEATURES

- 2-Pass Assembler
- Diagnostic error messages
- Symbolic addressing including forward references
- Constant generation
- Pseudo-ops to aid programming
- Free format source code

TABLE 4. SIMULATOR FEATURES

- Cycle Counter for timing estimates
- Instruction fetch break points
- Operand fetch break points
- Trace facilities
- Snapshot dumps
- Patching facility
- Statistical information generated
- Easy-to-use command language
- Optionally selected start and end addresses
- Dynamic changes of simulated registers
- Optionally simulates ROM-RAM environment

active Simulator the software designer can utilize his timesharing terminal to dynamically alter his program and effectively reduce his program development time.

The Signetics 2650 Symbolic Assembly Language has been modeled after other assembly languages; because of this, the assembler is easy to learn and to use.

The Simulator programs are designed to aid the user in testing and correcting his programs. This approach is an alternative to dedicating hardware development tools to one or two programmers or designers for program development. The Simulator allows users to simulate the execution of programs without utilizing a processor. The Simulator utilizes the object module produced by the Assembler as input, and through use of appropriate simulator commands, can display and/or alter the internal registers of the simulated 2650 processor and the simulated memory contents.

The programs are usually delivered delivered on IBM compatible magnetic tape "mini-reels". All programs are in FORTRAN source code as card image records.

A growing Program Library is available to Signetics microprocessor users. We encourage users to submit all non-proprietary programs to Signetics to add to the program library so that we may make them available to other users.

PROTOTYPING HARDWARE

PROTOTYPING CARD

In order to develop a product using the Signetics 2650 microprocessor, both hardware and software must be designed. Recognizing that the basic needs of many of our customers for prototyping systems will be similar, Signetics has designed a prototyping card containing a basic microcomputer system. This card provides a starting point for the development of hardware interfaces while simultaneously providing a tool for software checkout.

The first Signetics prototyping card consists of a 2650 processor, ROM memory containing a loader and editor, RAM memory for program storage before committing to PROM or ROM, a TTY interface for easy access, a crystal-controlled clock and two input and output ports (8 bits each).

SYSTEM COMPATIBLE FAMILIES

The 2650 has been designed to interface directly with industry standard logic and memory families, particularly 7400 and 74LS00 logic families, TTL compatible 5V NMOS memories (Signetics' 2600 series) and bipolar memories (Signetics' 8200 and 82S00 series). Many interface circuits in the 8T00 family are particularly useful for constructing interfaces in 2650 systems.

Other logic families including 8200 TTL, 82S00 STTL and 4000 CMOS are compatible with the 2650. See Table 5.

TABLE 5. SYSTEM COMPATIBLE FAMILIES

Logic	7400, 8200	–	TTL
	74LS00	–	TTL-LS
	82S00	–	STTL
	4000	–	CMOS
Memory	2500	–	PMOS
	2600	–	NMOS
	7400, 8200	–	Bipolar TTL
	82S00	–	Bipolar STTL
Interface	8T00	–	TTL, STTL

CHAPTER II

2650 HARDWARE

FEATURES

GENERAL PURPOSE PROCESSOR
SINGLE CHIP
FIXED INSTRUCTION SET
PARALLEL 8-BIT BINARY OPERATIONS
40 PIN DUAL IN-LINE PACKAGE

N-CHANNEL SILICON GATE MOS TECHNOLOGY
TTL COMPATIBLE INPUTS AND OUTPUTS
SINGLE POWER SUPPLY OF +5 VOLTS
SEVEN GENERAL PURPOSE REGISTERS
RETURN ADDRESS STACK, 8 DEEP, ON CHIP

32K BYTE ADDRESSING RANGE
SEPARATE ADDRESS AND DATA LINES
VARIABLE LENGTH INSTRUCTIONS OF 1, 2, OR 3 BYTES
75 INSTRUCTIONS
MACHINE CYCLE TIME OF 2.4μsec
AT CLOCK FREQUENCY OF 1.25 MHz

DIRECT INSTRUCTIONS TAKE 2, 3 or 4 CYCLES
SINGLE PHASE TTL LEVEL CLOCK INPUT
STATIC LOGIC
TRI-STATE OUTPUT BUSSES
REGISTER, IMMEDIATE, RELATIVE, ABSOLUTE
INDIRECT, AND INDEXED ADDRESSING MODES
VECTOR INTERRUPT FORMAT

INTRODUCTION

GENERAL FEATURES

The 2650 processor is a general purpose, single chip, fixed instruction set, parallel 8-bit binary processor. A general purpose processor can perform any data manipulations through execution of a stored sequence of machine instructions. The processor has been designed to closely resemble conventional binary computers, but executes variable length instructions of one to three bytes in length. BCD Arithmetic is made possible through use of a special "DAR" machine instruction.

The 2650 is manufactured using Signetics' N-channel silicon gate MOS technology. N-channel provides high carrier mobility for increased speed and also allows the use of a single 5 volt power supply. Silicon gate provides for better density and speed. Standard 40 pin dual in-line packages are used for the processor.

The 2650 contains a total of seven general purpose registers, each eight bits long. They may be used as source or destination for arithmetic operations, as index registers, and for I/O transfers.

The processor can address up to 32,768 bytes of memory in four pages of 8,192 bytes each. The processor instructions are one, two, or three bytes long, depending on the instruction. Variable length instructions tend to conserve memory space since a one-or two-byte instruction may often be used rather than a three byte instruction. The first byte of each instruction always specifies the operation to be performed and the addressing mode to be used. Most instructions use six of the first eight bits for this purpose, with the remaining two bits forming the register field. Some instructions use the full eight bits as an operation code.

The most complex direct instruction is three bytes long and takes 9.6 microseconds to execute. This figure assumes that the processor is running at its maximum clock rate, and has an associated memory with cycle and access times of one microsecond or less. The fastest instruction executes in 4.8 microseconds.

The clock input to the processor is a single phase pulse train and uses only one interface pin. It requires a normal TTL voltage swing, so no special clock driver is required.

The Data Bus and Address signals are tri-state to provide convenience in system design. Memory and I/O interface signals are asynchronous so that Direct Memory Access (DMA) and multiprocessor operations are easy to implement.

The 2650 has a versatile set of addressing modes used for locating operands for operations. They are described in detail in the INSTRUCTIONS section of this manual.

The interrupt mechanism is implemented as a single level, address vectoring type. Address vectoring means that an interrupting device can force the processor to execute code at a device determined location in memory. The interrupt mechanism is described in detail in the FEATURES section of this manual.

APPLICATIONS

The ability of the semi-conductor industry to manufacture complete general purpose processors on single chips represents a significant technological advance which should prove to be of great benefit to digital systems manufacturers. In terms of chip size and density of transistors, the processors are simply extensions of the continually evolving MOS technology. But in terms of function provided, a significant threshold has been crossed.

By allowing designers to convert from hardware logic to programmed logic, the integrated processor provides several important advantages.

1. Logic functions may be implemented in memory bits instead of logic gates. The user then has greater access to the advantages of memory circuits. Memories use patterned circuitry and thus provide greater density and therefore greater economy.
2. Random logic implementations of complex functions are highly specialized and cannot be used in other applications. They are not often used in large volume. Programmed logic, on the other hand, relies on general purpose processor and memory circuits that are used in many applications. Thus, economies of volume are available for both the user and the manufacturer.
3. Because the functional specialization resides in the user's program rather than the hardware logic, changes, corrections and additions can be much easier to make and can be accomplished in a much shorter time.
4. With the programmed logic approach it is often possible to add new features and create new products simply by writing new programs.
5. The design cycle of a system using programmed logic can be significantly shorter than a similar system that attempts to use custom random logic. The debugging cycle is also greatly compressed.

A general purpose processor designed to implement programmed logic has many characteristics that allow it to do conventional computer operations as well. Many applications will specialize in programmed logic or in data processing, but some will take advantage of both areas. In a line printer application, for example, a processor can act primarily as a controller handling the housekeeping duties, control sequencing and data interfacing for the printer. It also might buffer the data or do some code conversions, but that is not its primary duty. On the other hand, in a text editing intelligent terminal, the processor is mainly concerned with data manipulation since it handles code translations, display paging, insertions, deletions, line justification, hyphenation, etc.

A point-of-sale type of terminal represents an application that combines both control and data processing activities for the processor. Coordinating the activities of the various devices and displays that make up the terminal is an important part of the job, as are the calculations that are essential to the operation of the machine.

INTERNAL ORGANIZATION

INTERNAL REGISTERS

The block diagram for the 2650 shows the major internal components and the data paths that interconnect them. In order for the processor to execute an instruction, it performs the following general steps:

1. The Instruction Address Register provides an address for memory.
2. The first byte of an instruction is fetched from memory and stored in the Instruction Register.
3. The Instruction Register is decoded to determine the type of instruction and the addressing mode.
4. If an operand from memory is required, the operand address is resolved and loaded into the Operand Address Register.
5. The operand is fetched from memory and the operation is executed.
6. The first byte of the next instruction is fetched.

The Instruction Register (IR) holds the first byte of each instruction and directs the subsequent operations required to execute each instruction. The IR contents are decoded and used in conjunction with the timing information to control the activation and sequencing of all the other elements on the chip. The Holding Register (HR) is used in some multiple-byte instructions to contain further instruction information and partial absolute addresses.

The Arithmetic Logic Unit (ALU) is used to perform all of the data manipulation operations, including Load, Store, Add, Subtract, And, Inclusive Or, Exclusive Or, Compare, Rotate, Increment and Decrement. It contains and controls the Carry bit, the Overflow bit, the Interdigit Carry and the Condition Code Register.

The Register Stack contains six registers that are organized into two banks of three registers each. The Register Select bit (RS) picks one of the two banks to be accessed by instructions. In order to accomodate the register-to-register instructions, register zero (RO) is outside the array. Thus, register zero is always available along with one set of three registers.

The Address Adder (AA) is used to increment the instruction address and to calculate relative and indexed addresses.

The Instruction Address Register (IAR) holds the address of the next instruction byte to be accessed. The Operand Address Register (OAR) stores operand addresses and sometimes contains intermediate results during effective address calculations.

The Return Address Stack (RAS) is an eight level, Last In, First Out (LIFO) storage which receives the return address whenever a Branch-to-Subroutine instruction is executed. When a Return instruction is executed, the RAS provides the last return address for the processor's IAR. The stack contains eight levels of storage so that subroutines may be nested up to eight levels deep. The Stack Pointer (SP) is a three bit wraparound counter that indicates the next available level in the stack. It always points to the current address.

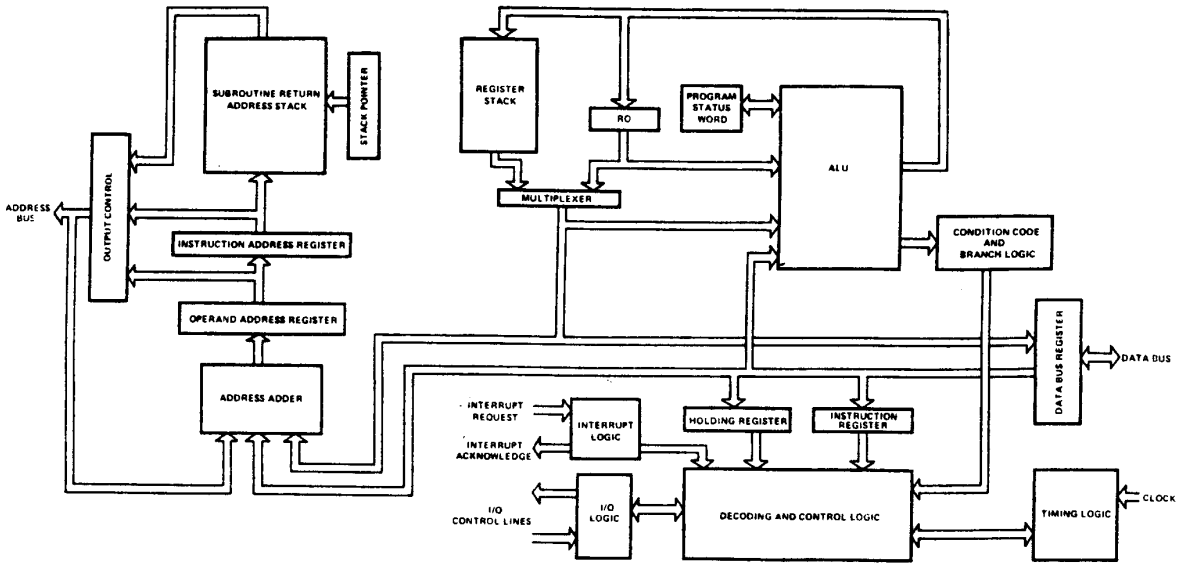


Figure 7. SIGNETICS 2650 BLOCK DIAGRAM

PROGRAM STATUS WORD

The Program Status Word (PSW) is a special purpose register within the processor that contains status and control bits. It is 16 bits long and is divided into two bytes called the Program Status Upper (PSU) and the Program Status Lower (PSL).

The PSW bits may be tested, loaded, stored, preset or cleared using the instructions which effect the PSW. The sense bit, however, cannot be set or cleared because it is directly connected to pin #1.

PSU	7	6	5	4	3	2	1	0
	S	F	II	Not Used	Not Used	SP2	SP1	SP0

S Sense
F Flag
II Interrupt Inhibit
SP2 Stack Pointer Two
SP1 Stack Pointer One
SP0 Stack Pointer Zero

PSL	7	6	5	4	3	2	1	0
	CC1	CC0	IDC	RS	WC	OVF	COM	C

CC1 Condition Code One
CC0 Condition Code Zero
IDC Interdigit Carry
RS Register Bank Select
WC With/Without Carry
OVF Overflow
COM Logical/Arithmetic Compare
C Carry/Borrow

SENSE (S)

The Sense bit in the PSU reflects the logic state of the sense input to the processor at pin #1. The sense bit is not affected by the LPSU, PPSU, or CPSU instructions. When the PSU is tested (TPSU) or stored into register zero (SPSU), bit #7 reflects the state of the sense pin at the time of the instruction execution.

FLAG (F)

The Flag bit is a simple latch that drives the Flag output (pin #40) on the processor.

INTERRUPT INHIBIT (II)

When the Interrupt Inhibit (II) bit is set, the processor will not recognize an incoming interrupt. When interrupts are enabled (II=0), and an interrupt signal occurs, the inhibit bit in the PSU is then automatically set. When a Return-and-Enable instruction is executed, the inhibit bit is automatically cleared.

STACK POINTER (SP)

The three Stack Pointer bits are used to address locations in the Return Address Stack (RAS). The SP designates the stack level which contains the current return address. The three SP bits are organized as a binary counter which is automatically incremented with execution of Branch-to-Subroutine instructions, and decremented with execution of Return instructions.

CONDITION CODE (CC)

The Condition Code is a two bit register which is set by the processor whenever a general purpose register is loaded or modified by the execution of an instruction. Additionally, the CC is set to reflect the relative value of two bytes whenever a compare instruction is executed.

The following table indicates the setting of the Condition Code whenever data is set into a general purpose register. The data byte is interpreted as an 8-bit, two's complement number.

Register Contents	CC1	CC0
Positive	0	1
Zero	0	0
Negative	1	0

For compare instructions the following table summarizes the setting of the CC. The data is compared as two 8-bit absolute numbers if bit #1, the COM bit, of the Program Status Lower byte is set to indicate "logical" compare (COM=1). If the COM bit indicates "arithmetic" (COM=0), the comparison instructions interpret the data bytes as two 8-bit two's complement binary numbers.

Register to Storage Compare Instruction	Register to Register Compare Instruction	CC1	CC0
Reg X Greater Than Storage	Reg 0 Greater Than Reg X	0	1
Reg X Equal to Storage	Reg 0 Equal to Reg X	0	0
Reg X Less Than Storage	Reg 0 Less Than Reg X	1	0

The CC is never set to 11 by normal processor operations, but it may be explicitly set to 11 through LPSL or PPSL instruction execution.

INTERDIGIT CARRY (DC)

For BCD arithmetic operations it is sometimes essential to know if there was a carry from bit #3 to bit #4 during the execution of an arithmetic instruction.

The IDC reflects the value of the Interdigit Carry from the previous add or subtract instruction. After any add or subtract instruction execution, the IDC contains the carry or borrow out of bit #3.

The IDC is also set upon execution of Rotate instructions when the WC bit in the PSW is set. The IDC will reflect the same information as bit #5 of the operand register after the rotate is executed. See Figure 8.

REGISTER SELECT (RS)

There are two banks of general purpose registers with three registers in each bank. The register select bit is used to specify which set of three general purpose registers will be currently used. Register zero is common and is always available to the program. An individual instruction may address only four registers, but the bank select feature effectively expands the available on-chip registers to seven. When the Register Select Bit is "0", registers 1, 2, & 3 in register bank #0 will be accessable, and when the bit is "1", registers 1, 2, & 3 in register bank #1 will be accessable.

WITH/WITHOUT CARRY(WC)

This bit controls the execution of the add, the subtract and the rotate instructions.

Whenever an add or a subtract instruction executes, the following bits are either set or cleared: Carry/Borrow (C), Overflow (OVF), and Interdigit Carry (IDC). These bits are set or reset without regard to the value of the WC bit. However, when WC=1, the final value of the carry bit affects the result of an add or a subtract instruction, i.e., the carry bit is either added (add instruction) or subtracted (subtract instruction) from the ALU.

Whenever a rotate instruction executes with WC=0, only the eight bits of the rotated register are affected. However, when WC=1, the following bits are also affected: Carry/Borrow (C), Overflow (OVF) and Interdigit Carry (IDC). The carry/borrow bit is combined with the 8-bit register to make a nine-bit rotate (see Figure 8). The overflow bit is set whenever the sign bit (bit 7) of the rotated register changes its value, i.e., from a zero (0) to a one (1) or from a one (1) to a zero (0). The interdigit carry bit is set to the new value of bit 5 of the rotated register.

OVERFLOW (OVF)

The overflow bit is set during add or subtract instruction executions whenever the two initial operands have the same sign but the result has a different sign. Operands with different signs cannot cause overflow. Example: A binary +124 (01111100) added to a binary +64 (01000000) produces a result of (10111100) which is interpreted in two's complement form as a −68. The true answer would be 188, but that answer cannot be contained in the set of 8-bit, two's complement numbers used by the processor, so the OVF bit is set.

Rotate instructions also cause OVF to be set whenever the sign of the rotated byte changes.

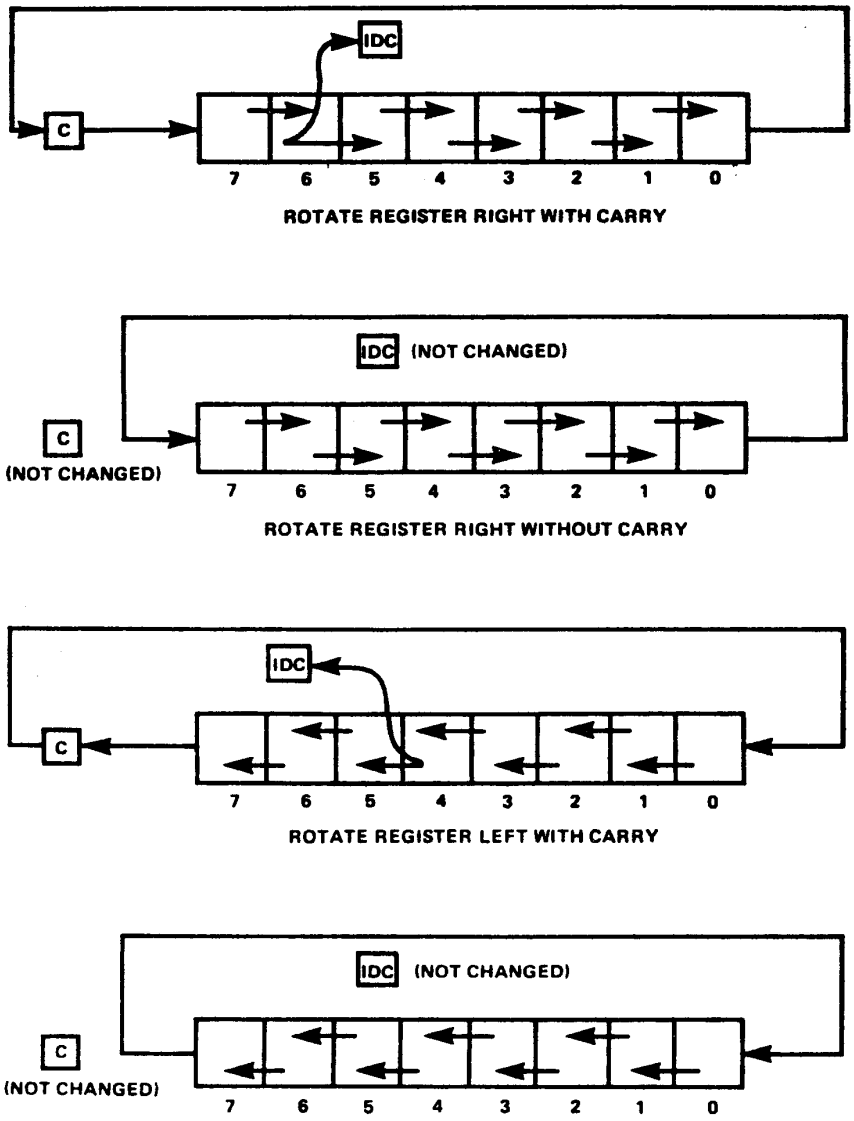


Figure 8.

COMPARE (COM)

The compare control bit determines the type of comparison that is executed with the Compare instructions. Either logical or arithmetic comparisons may be made. The arithmetic compare assumes that the comparison is between 8-bit, two's complement numbers. The logical compare assumes that the comparison is between 8-bit positive binary numbers. When COM is set to 1, the comparisons will be logical, and when COM is set to 0, the comparisons will be arithmetic. See Condition Code (CC).

CARRY (C)

The Carry bit is set by the execution of any add or subtract instruction that results in a carry or borrow out of the high order bit of the ALU. The carry bit is set to 1 by an add instruction that generates a carry, and a subtract instruction that does *not* generate a borrow. Inversely, an add that does not generate a carry causes the C bit to be cleared, and a subtract instruction that generates a borrow also clears the carry bit.

Even though a borrow is indicated by a zero in the Carry bit, the processor will correctly interpret the zero during subtract with borrow operations as in the following table.

Low Order bit Minuend	Low Order bit Subtrahend	Carry bit Borrow bit	Low Order Bit Result
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

The carry bit may also be set or cleared by rotate instructions as described earlier under "With/Without Carry".

To perform an Add with Carry or a Subtract with Borrow, the WC bit must be set.

MEMORY ORGANIZATION

The 2650 has a maximum memory addressing capability of 0_{10} —$32{,}767_{10}$ locations. As may be seen in the INSTRUCTIONS section of this manual, most direct addressing instructions have thirteen bits allocated for the direct address. Since thirteen bits can only address locations 0_{10} —$8{,}191_{10}$, a paging system was implemented to accomodate the entire address range.

The memory may be thought of as being divided into four pages of 8,192 bytes each. The addresses in each page range as in the following chart:

	START ADDRESS	END ADDRESS	
page 0	000000000000000	001111111111111	0_{10}—8191_{10}
page 1	010000000000000	011111111111111	8192_{10}—$16{,}383_{10}$
page 2	100000000000000	101111111111111	$16{,}384_{10}$—$24{,}575_{10}$
page 3	110000000000000	111111111111111	$24{,}576_{10}$—$32{,}767_{10}$

The low order 13-bits in every page range through the same set of numbers. These 13-bits are the same 13-bits addressed by non-branch instructions and are also the same 13-bits which are brought out of the 2650 on the address lines ADR0 — ADR12.

The high order two bits of the 15-bit address are known as the page bits. The page bits when examined by themselves also represent, in binary, the number of the memory page. Thus, the address 010000001101101 is known as address location 109_{10} in page 1. The page bits, corresponding to ADR13 and ADR14 are brought out of the 2650 on pins 19 & 18. These bits may be used for memory access when more than 8,192 bytes of memory are connected.

There are no instructions to explicitly set the page bits. They may be set through execution of direct or indirect, branch or branch-to-subroutine instructions. It may be seen that these instructions (see INSTRUCTION Section) have 15-bits allocated for address and when such an instruction is executed, the two high order address bits are set into the page bit latches in the 2650 processor and will appear on ADR13 and ADR14 during memory accesses until they are specifically changed.

For memory access from non-branch instructions, the 13-bit direct address will address the corresponding location within the current page only. However, the non-branch memory access instruction may access any byte in any page through indirect addressing which provides the full 15-bit address. In the case of non-branch instructions, the page bits are only temporarily changed to correspond to the high order two bits of the 15-bit indirect address used to fetch the argument byte. Immediately after the memory access, ADR13 & ADR14 will revert to their previous value.

The consequences of this page address system may be summarized by the following statements.

1. The RESET signal clears both page latches, i.e., ADR13 & ADR14 are cleared to zero.
2. All non-branch, direct memory access instructions address memory within the current page.
3. All non-branch, memory access instructions may access any byte of addressable memory through use of indirect addressing which temporarily changes the page bits for the argument access, but which revert back to their previous state immediately following instruction execution.
4. All direct and indirect addressing branch instructions set the page bits to correspond to the high order two bits of the 15 bit address.
5. Programs may *not* flow across page boundaries, they must branch to set the page bits.
6. Interrupts always drive the processor to page zero.

INTERFACE

SIGNALS

RESET

The RESET signal is used to cause the 2650 to begin processing from a known state. RESET will normally be used to initialize the processor after power-up or to restart a program. RESET clears the Interrupt Inhibit control bit, clears the internal interrupt-waiting signal, and initializes the IAR to zero. RESET is normally low during program execution, and must be driven high to activate the RESET function. The leading and trailing edges may be asynchronous with respect to the clock. The RESET signal must be at least three clock periods long. If RESET alone is used to initiate processing, the first instruction will be fetched from memory location page zero byte zero after the RESET signal is removed. Any instruction may be programmed for this location including a Branch to some program located elsewhere.

Processing can also be initiated by combining an interrupt with a reset. In this case, the first instruction to be executed will be at the interrupt address.

CLOCK

The clock signal is a positive-going pulse train that determines the instruction execution rate. Three clock periods comprise a processor cycle. Direct instructions are 2, 3, or 4 processor cycles long, depending on the specific type of instruction. Indirect addressing adds two processor cycles to the direct instruction times.

$\overline{\text{PAUSE}}$

The $\overline{\text{PAUSE}}$ input provides a means for temporarily stopping the execution of a program. When $\overline{\text{PAUSE}}$ is driven low, the 2650 finishes the instruction in progress and then enters the WAIT state. When $\overline{\text{PAUSE}}$ goes high, program execution continues with the next instruction. If $\overline{\text{PAUSE}}$ is turned on then off again before the last cycle of the current instruction begins, program execution continues without pause. If both $\overline{\text{PAUSE}}$ and $\overline{\text{INTREQ}}$ occur prior to the last cycle of the current instruction, the interrupt will be recognized, and an INTACK will be generated immediately following release of the $\overline{\text{PAUSE}}$. The next instruction to be executed will be a ZBSR to service the interrupt.

If an $\overline{\text{INTREQ}}$ occurs while the 2650 is in a WAIT state due to a $\overline{\text{PAUSE}}$, the interrupt will be acknowledged and serviced after the execution of the next normal instruction following release of the $\overline{\text{PAUSE}}$.

$\overline{\text{INTREQ}}$

The Interrupt Request input (normally high) is a means for external devices to change the flow of program execution. When the processor recognizes an $\overline{\text{INTREQ}}$, i.e., $\overline{\text{INTREQ}}$ is driven low, it finishes the instruction in progress, inserts a ZBSR instruction into the IR, turns on the Interrupt Inhibit bit in the PSU, and then responds with INTACK and OPREQ signals. Upon receipt of INTACK, the interrupting device may raise the $\overline{\text{INTREQ}}$ line and present a data byte to the processor on the DBUS. The required byte takes the same form as the second byte of a ZBSR instruction. Thus, the interrupt initiated Branch-to-Subroutine instruction may have a relative target address anywhere within the first or last 64 bytes of memory page 0. If indirect addressing is specified, a branch to any location in addressable memory is possible.

For devices that do not need the flexibility of the multiple target addresses, a byte of eight zeroes may be presented and will cause a direct subroutine branch to memory location zero in page zero. The relative address presented by the interrupting device is handled with a normal I/O read sequence using the usual interface control signals. The addition of the INTACK signal distinguishes the interrupt address operation from other operations that may take place as part of the execution of the interrupted instruction. At the same time that it acknowledges the $\overline{\text{INTREQ}}$, the processor automatically sets the bit that inhibits recognition of further interrupts. The Interrupt Inhibit bit may be cleared anytime during the interrupt service routine, or a Return-and-Enable instruction may be used to enable interrupts upon leaving the routine. If an $\overline{\text{INTREQ}}$ is waiting when the Interrupt Inhibit bit is cleared, it will be recognized and processed immediately without the execution of an intervening instruction.

$\overline{\text{OPACK}}$

The Operation Acknowledge signal is a reply from external memory or I/O devices as a response to the Operation Request signal from the processor. OPREQ is used to initiate an external operation. The affected external device indicates to the processor that the operation is complete by turning on the $\overline{\text{OPACK}}$ signal. This procedure allows asynchronous functioning of external devices.

If a Memory operation is initiated by the processor, the memory system will provide an $\overline{\text{OPACK}}$ when the requested memory data is valid on the Data Bus. If an I/O operation is initiated by the processor, the addressed I/O device may respond with an $\overline{\text{OPACK}}$ as soon as the write data is accepted from the Data Bus, or after the read operation is completed. However, in order to avoid slowing down the processor when using memories or I/O devices that are just fast enough to keep the processor operating at full speed the $\overline{\text{OPACK}}$ signal must be returned before the external operation is completed. Any $\overline{\text{OPACK}}$ that is returned within 600 nsec. following an OPREQ will not delay the processor. Data from a read operation can return up to 1000 nsec. after an OPREQ is sent and still be accepted by the processor without causing delays. If all devices will always respond within these time limits, the $\overline{\text{OPACK}}$ line may be permanently connected in the ON (low) state. Whenever an $\overline{\text{OPACK}}$ is not available within that time, the processor will delay instruction execution until the first clock following receipt of the $\overline{\text{OPACK}}$. All output line conditions remain unchanged during the delay and the processor does not enter the WAIT state. $\overline{\text{OPACK}}$ is true in the low state and false in the high state.

SENSE

The SENSE line provides an input line to the 2650 that is independent of the normal I/O Bus structures. The SENSE signal is connected directly to one of the bits in the Program Status Word. It may be stored or tested by an executing program. When a store (SPSU) or test (TPSU) instruction is executed, the SENSE line is sampled during the last cycle of the instruction.

Through proper programming techniques the SENSE signal may be used to implement a direct serial data input channel, or it may be used to present any bit of information that the designer chooses.

The SENSE input and FLAG output facilities provide the simplest method of communicating data in or out of the 2650 Processor as neither address decoding nor synchronization with other processor signals is necessary.

$\overline{\text{ADREN}}$

The Address Enable signal allows external control of the tri-state address outputs (ADR0-ADR12). When $\overline{\text{ADREN}}$ is driven high, the address lines are switched to their third state and show a high output impedance. This feature allows wired-OR connections with other signals. The ADR13 and ADR14 lines which are multiplexed with other signals are not affected by this signal.

When a system is not designed to utilize the feature, the $\overline{\text{ADREN}}$ input may be connected permanently to a low signal source.

$\overline{\text{DBUSEN}}$

The Data Bus Enable signal allows external control of the tri-state Data Bus output drivers. When $\overline{\text{DBUSEN}}$ is driven high, the Data Bus will exhibit a high output impedance. This allows wired-OR connection with other signals.

When a system is not designed to utilize this feature, the $\overline{\text{DBUSEN}}$ input may be permanently connected to a low signal source.

DBUS

The Data Bus signals form an 8-bit bi-directional data path in and out of the processor. Memory and I/0 operations use the Data Bus to transfer the write or read data to or from memory.

The direction of the data flow on the Data Bus is indicated by the state of the $\overline{\text{R}}$/W line. For Write operations, the output buffers in the processor output data to the bus for use by memory or by external devices. For Read operations, the buffers are disabled and the data condition of the bus is sensed by the processor. The output buffers may also be disabled by the $\overline{\text{DBUSEN}}$ signal.

The signals on the data bus are true signals, i.e., a one is a high level and a zero is low.

ADR

The Address signals form a 15 bit path out of the processor, and are used primarily to supply memory addresses during memory operations. The addresses remain valid as long as OPREQ is on so that no external address register is required. For extended I/O operations, the low order eight bits of the ADR lines are used to output the immediate byte of the instruction which typically is interpreted as a device address.

The 13 low order lines of the address are used only for address information. The two high order address lines are multiplexed with I/O control information. During memory operations, the lines serve as memory addresses. During I/O operations they serve as the D/$\overline{\text{C}}$ and E/$\overline{\text{NE}}$ control lines. Demultiplexing is accomplished through use of the Memory/$\overline{\text{IO}}$ Control line.

The line ADR0 carries the low order address bit, and ADR12 carries the high order address bit. The output drivers may be disabled by the $\overline{\text{ADREN}}$ signal.

The signals on the address bus are true, i.e., a one is a high level and a zero is low.

OPREQ

The Operation Request output is the coordinating signal for all external operations. The M/$\overline{\text{IO}}$, $\overline{\text{R}}$/W, E/$\overline{\text{NE}}$, D/$\overline{\text{C}}$ and INTACK lines are operation control signals that describe the nature of the external operation when the OPREQ line is true. The DBUS and ADR bus also should not be considered

valid except when OPREQ is in the high, or on state.

No output signals from the processor will change as long as OPREQ is on, with the exception of WRP. OPREQ will stay on until the external operation is complete, as indicated by the $\overline{\text{OPACK}}$ input. The processor delays all internal activity following an OPREQ until the $\overline{\text{OPACK}}$ signal is received.

INTACK

The Interrupt Acknowledge signal is used by the processor to respond to an external interrupt. When an $\overline{\text{INTREQ}}$ is received, the current instruction is completed before the interrupt is serviced. When the processor is ready to accept the interrupt it sets the INTACK to the high, or on, state along with OPREQ. The interrupting device then presents a relative address byte to the DBUS and responds with an $\overline{\text{OPACK}}$ signal. $\overline{\text{INTREQ}}$ may be turned off anytime following INTACK. INTACK will fall after the processor receives the $\overline{\text{OPACK}}$ signal.

M/$\overline{\text{IO}}$

The Memory/$\overline{\text{IO}}$ output is one of the operation control signals that defines external operations. M/$\overline{\text{IO}}$ indicates whether an operation is memory or I/O and should be used to gate Read or Write signals between memory or I/O devices.

The state of M/$\overline{\text{IO}}$ will not change while OPREQ is high.

The high state corresponds to Memory operation, and the low state corresponds to an I/O operation.

$\overline{\text{R}}$/W

The Read/Write output is one of the operation control signals that defines external operations. $\overline{\text{R}}$/W indicates whether an operation is *Read* or *Write*. It controls the nature of the external operation and indicates in which direction the DBUS is pointing. $\overline{\text{R}}$/W should not be considered valid until OPREQ is on and the state of the $\overline{\text{R}}$/W line does not change as long as OPREQ is on.

The high state corresponds to the Write operation, and the low state corresponds to the Read operation.

D/$\overline{\text{C}}$

The Data/Control Output is an I/O signal which is used to discriminate between the execution of the two types of one byte I/O instructions. There are four one byte I/O instructions; WRTC, WRTD, REDC, REDD. When Read Control or Write Control is executed, the D/$\overline{\text{C}}$ line takes on the low state which indicates Control ($\overline{\text{C}}$). When Read Data or Write Data is executed, the D/$\overline{\text{C}}$ line takes on the high state, indicating Data (D).

D/$\overline{\text{C}}$ should not be considered valid until (a) OPREQ is on and (b) M/$\overline{\text{IO}}$ indicates an I/O operation and (c) E/$\overline{\text{NE}}$ indicates a non-extended (one byte) operation, D/$\overline{\text{C}}$ is multiplexed with a high order address line. When the M/$\overline{\text{IO}}$ line is in the I/O state, the ADR14-D/$\overline{\text{C}}$ line should be interpreted as "D/$\overline{\text{C}}$". (When the M/$\overline{\text{IO}}$ line is in the M state, the ADR14-D/$\overline{\text{C}}$ line should be interpreted as memory address line #14.)

E/$\overline{NE}$

The Extended/Non-Extended output is the operation control signal that is used to discriminate between two byte and one byte I/O operations. Thus, E/$\overline{NE}$ indicates the presence or absence of valid information on the eight low order address lines during I/O operations.

E/$\overline{NE}$ should not be considered valid until (a) OPREQ is on and (b) M/$\overline{IO}$ indicates an I/O operation. E/$\overline{NE}$ is multiplexed with a high order address line. When the M/$\overline{IO}$ line is in the I/O state, the ADR13-E/$\overline{NE}$ line should be interpreted as "E/$\overline{NE}$". (When the M/$\overline{IO}$ line is in the M state, the ADR13-E/$\overline{NE}$ line should be interpreted as memory address bit #13.)

There are six I/O instructions; REDE, WRTE, REDC, REDD, WRTC, WRTD. When either of the two byte I/O instructions is executed (REDE, WRTE), the E/$\overline{NE}$ line takes on the high state or "Extended" indication. When any of the one byte I/O instructions is executed, the line takes on the low state or "non-extended" indication.

RUN/$\overline{WAIT}$

The RUN/$\overline{WAIT}$ output signal indicates the Run/Wait Status of the processor. The WAIT state may be entered by executing a HALT instruction or by turning on the $\overline{PAUSE}$ input. At any other time the processor will be in a RUN state.

When the processor is executing instructions, the line is in the high or RUN state; when in the WAIT state, the line is held low.

The HALT initiated WAIT condition can be changed to RUN by a RESET or an interrupt. The $\overline{PAUSE}$ initiated WAIT condition can be changed to RUN by removing the $\overline{PAUSE}$ input.

If a RESET occurs during a $\overline{PAUSE}$ initiated WAIT state and the $\overline{PAUSE}$ remains low; the processor will be reset, fetch one instruction from page zero byte zero and return to the WAIT state. When the $\overline{PAUSE}$ is eventually removed, the previously fetched instruction will be executed.

FLAG

The FLAG output indicates the state of the Flag bit in the PSW. Any change in the Flag bit is reflected by a change in the FLAG output. A one bit in the Flag will give a high level on the FLAG output pin. The LPSU, PPSU, and CPSU instructions can change the state of the Flag bit. The FLAG output is always a valid indication of the state of the Flag bit without regard for the status of the processor or control signals. Changes in the Flag bit are synchronized with the last cycle of the changing instruction.

WRP

The Write Pulse output is a timing signal from the processor that provides a positive-going pulse in the middle of each requested write operation (memory or I/O) and a high level during read operations. The WRP is designed to be used with Signetics 2606 R/W memory circuits to provide a timed Chip Enable signal. For use with memory, it may be gated with the M/$\overline{IO}$ signal to generate a Memory Write Pulse.

Because the WRP pulse occurs during any write operation, it may also be used with I/O write operations where convenient.

SIGNAL TIMING

The Clock input to the 2650 provides the basic timing information that the processor uses for all its internal and external operations. The clock rate determines the instruction execution rate, except to the extent that external memories and devices slow down the processor. Each internal processor cycle is composed of three clock periods as shown in Figure 9, 2650 TIMING DIAGRAMS.

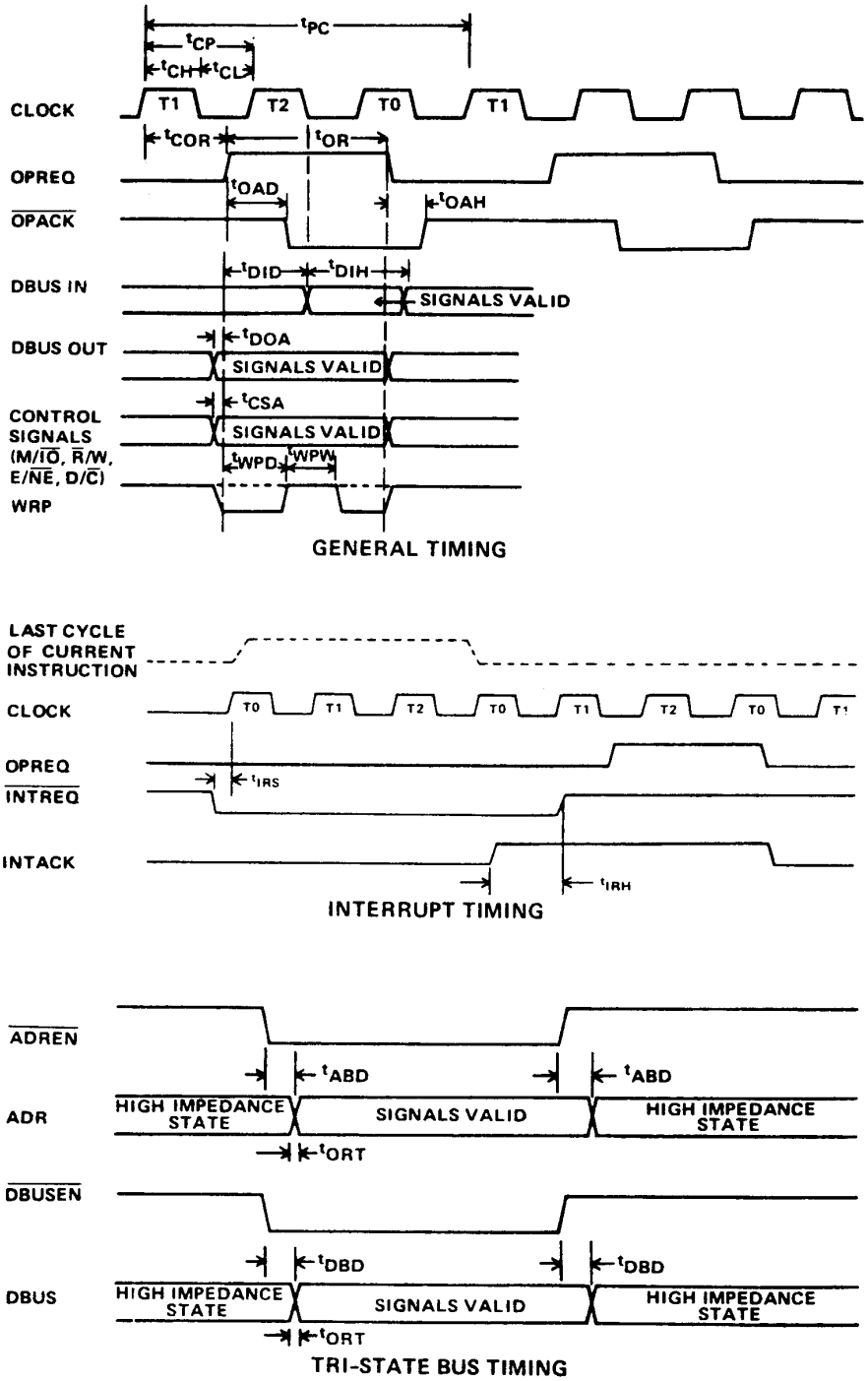


Figure 9. 2650 TIMING DIAGRAMS

OPREQ is the master control signal that coordinates all operations external to the processor. Many of the other signal interactions are related to OPREQ. The timing diagram assumes that the clock periods are constant and that $\overline{OPACK}$ is returned in time to avoid delaying instruction execution. In that case, OPREQ will be high for 1.5 clock periods (1/2 of t_{pc}) and then will be low for another 1.5 clock periods.

The interface control signals have been designed to implement asynchronous interfaces for both memory and input/output devices. The control signals are relatively simple and provide the following advantages: no external synchronizing is necessary, external devices may run at any data rate up to the processor's maximum I/O data rate, and because data signals are furnished with guard signals the external devices are often relieved of the necessity of latching information such as memory address.

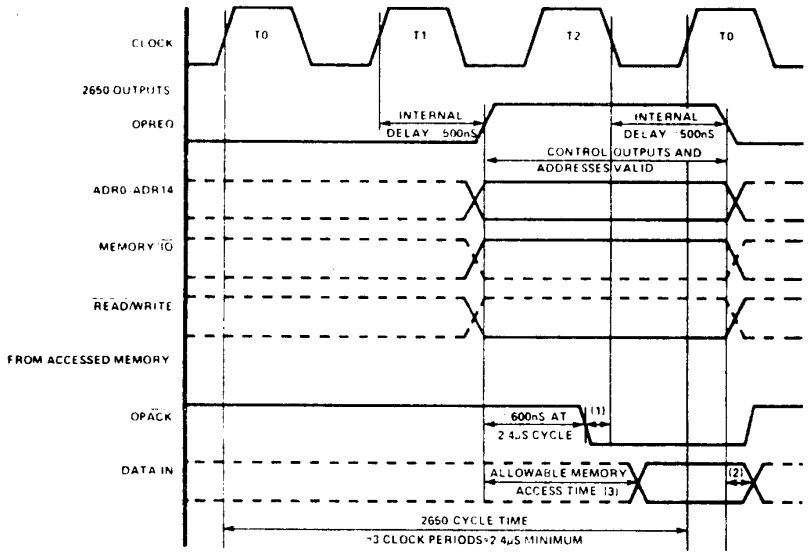
MEMORY READ TIMING

The following signals are involved in the processor's memory read sequence, as shown in Figure 10.

OPREQ	= Operation Request
DBUS0-DBUS7*	= Data Bus
ADR0-ADR12	= Address Bus
ADR13	= Address bit 13
ADR14	= Address bit 14
M/$\overline{IO}$	= Memory/Input-Output
$\overline{R}$/W	= Read/Write
$\overline{OPACK}$*	= Operation Acknowledge

The signals marked with an asterisk are sent from the memory device to the processor. The other signals are developed by the processor.

OPREQ is a guard signal which must be valid (high) for the other signals to have meaning. When reading main memory the 2650 simultaneously switches OPREQ to a high state, M/$\overline{IO}$ to M (memory), $\overline{R}$/W to $\overline{R}$ (Read), and places the memory address on lines ADR0-ADR14. Remember that



NOTES: (1) OPACK must go low at least 100 nS before the trailing edge of T2 in order not to slow down the 2650.
(2) DATA IN signals must be valid for 50nS after the trailing edge of OPREQ.
(3) Allowable memory access time is 1µs with 2 4µs cycle time.

Figure 10. MEMORY READ SEQUENCE

ADR13 & ADR14 are multiplexed with other signals and must be logically ANDed with OPREQ and M to be interpreted. Of course, ADR13 & ADR14 may be ignored if only page zero (8,192 bytes) is used.

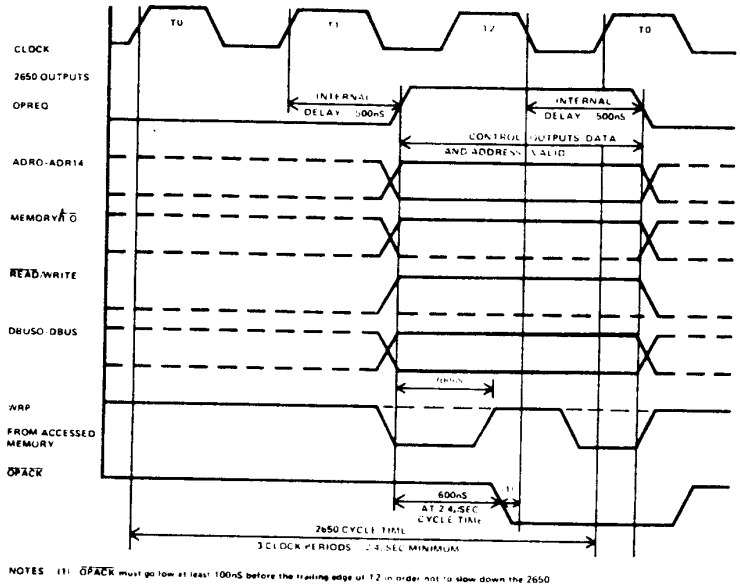
Once the memory logic has determined the simultaneous existance of the signals mentioned above, it places the true data corresponding to the given address location on the data bus (DBUS0 to DBUS7), and returns an $\overline{\text{OPACK}}$ signal to the processor. The processor, recognizing the $\overline{\text{OPACK}}$, strobes the data into the receiving register and lowers the OPREQ. This completes the memory read sequence.

If the $\overline{\text{OPACK}}$ signal is delayed by the memory device, the processor waits until it is received. OPREQ is lowered only after the receipt of $\overline{\text{OPACK}}$. The memory device should raise $\overline{\text{OPACK}}$ after OPREQ falls.

MEMORY WRITE TIMING

The signals involved with the processor's memory write sequence are similar to those used in the memory read sequence with the following exceptions: 1) the $\overline{\text{R}}$/W signal is in the W state and, 2) the WRP signal provides a positive going pulse during the write sequence which may be used as a chip enable, write pulse, etc.

Figure 11 demonstrates the signals that occur during a memory write.



NOTES (1) $\overline{\text{OPACK}}$ must go low at least 100nS before the trailing edge of T2 in order not to slow down the 2650

Figure 11. MEMORY WRITE SEQUENCE

INPUT/OUTPUT TIMING

The signal exchanges for I/O with external devices is very similar to the signaling for memory read/write. See the Features Section, INPUT/OUT PUT FACILITIES.

CRITICAL TIMES

Figure 9 describes the timing relationship between the various interface signals. The critical times are labeled and defined in the table of AC characteristics.

ELECTRICAL CHARACTERISTICS

PRELIMINARY AC CHARACTERISTICS

T_A=0°C to 70°C V_{CC}=5V±5% unless otherwise specified, see notes 1,2,3 & 4.

SYMBOL	PARAMETER	LIMITS		UNITS
		MIN	MAX	
t_{CH}	Clock High Phase	400	10,000	nsec
t_{CL}	Clock Low Phase	400	∞	nsec
t_{CP}	Clock Period	800	∞	nsec
t_{PC}[6]	Processor Cycle Time	2,400	∞	nsec
t_{OR}	OPREQ Pulse Width	$2t_{CH} + t_{CL} - 100$	∞	nsec
t_{COR}	Clock to OPREQ Time	100	700	nsec
t_{OAD}[7]	$\overline{\text{OPACK}}$ Delay Time	0	∞	nsec
t_{OAH}	$\overline{\text{OPACK}}$ Hold Time	0	∞	nsec
t_{CSA}	Control Signal Available	50		nsec
t_{DOA}	Data Out Available	50		nsec
t_{DID}[8]	Data in Delay	0	1000(8)	nsec
t_{DIH}[9]	Data in Hold	150		nsec
t_{WPD}	Write Pulse Delay	t_{CL}-100	t_{CL}-50	nsec
t_{WPW}	Write Pulse Width	t_{CL}	t_{CL}	nsec
t_{ABD}	Address Bus Delay		80	nsec
t_{DBD}	Data Bus Delay		120	nsec
t_{IRS}[10]	$\overline{\text{INTREQ}}$ Set up Time	0		nsec
t_{IRH}[10]	$\overline{\text{INTREQ}}$ Hold Time	0		nsec
t_{ORT}[5]	Output Buffer Rise Time		150	nsec

NOTES ON AC CHARACTERISTICS

1. See preceding timing diagrams for definition of timing terms.
2. Input levels swing between 0.65 volt and 2.2 volts.
3. Input signal transition times are 20ns.
4. Timing reference level is 1.5 volts.
5. Load is -100μA at 20pF.
6. A Processor Cycle time consists of three clock periods.
7. In order to avoid slowing down the processor, $\overline{\text{OPACK}}$ must be lowered 100ns before the trailing edge of T2 clock, if $\overline{\text{OPACK}}$ is delayed past this point, the processor will wait in the T2 state and sample $\overline{\text{OPACK}}$ on each subsequent negative clock edge until $\overline{\text{OPACK}}$ is lowered.
8. In order to avoid slowing the processor down, input data must be returned to the processor in 1μs or less time from the OPREQ edge, at a cycle time of 2.4μs.
9. Input data must be held until 50ns after OPREQ falls.
10. In order to interrupt the current instruction, $\overline{\text{INTREQ}}$ must fall prior to the first clock of the last cycle of the current instruction. $\overline{\text{INTREQ}}$ must remain low until INTACK goes high.

MAXIMUM GUARANTEED RATINGS[1]

Operating Ambient Temperature	0°C to +70°C
Storage Temperature	-65°C to + 150°C
All Input, Output, and Supply Voltages with respect to ground pin[3]	-0.5V to +6V
Package Power Dissipation[2]=IWPkg.	1.6W

PRELIMINARY 2650 DC ELECTRICAL CHARACTERISTICS

			LIMITS		
SYMBOL	PARAMETER	TEST CONDITIONS	MIN	MAX	UNIT
I_{LI}	Input Load Current	V_{IN} = 0 to 5.25V		10	μA
I_{LOH}	Output Leakage Current	ADREN, DBUSEN = 2.2V, V_{OUT} = 4V		10	μA
I_{LOL}	Output Leakage Current	ADREN, DBUSEN = 2.2V, V_{OUT} = 0.45V		10	μA
I_{CC}	Power Supply Current	V_{CC} = 5.25V, T_A = 0°C		100	mA
V_{IL}	Input Low		-0.6	0.8	V
V_{IH}	Input High		2.2	V_{CC}	V
V_{OL}	Output Low	I_{OL} = 1.6 mA	0.0	0.45	V
V_{OH}	Output High	I_{OH} = -100 μA	2.4	V_{CC}-0.5	V
C_{IN}	Input Capacitance	V_{IN} = 0V		10	pF
C_{OUT}	Output Capacitance	V_{OUT} = 0V		10	pF

Conditions: T_A = 0°C to 70°C, V_{CC} = 5V ±5%

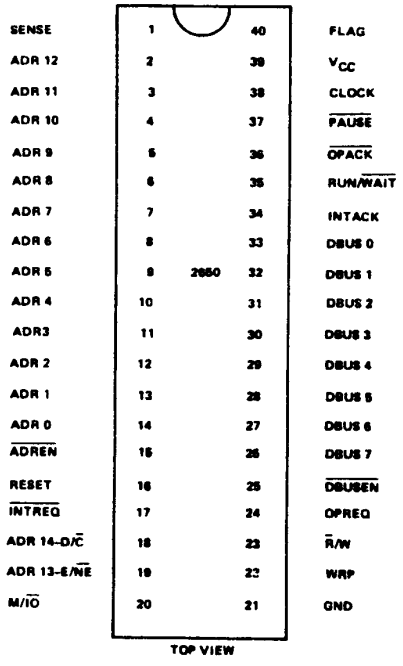
NOTES:

1. Stresses above those listed under "Maximum Guaranteed Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or at any other condition above those indicated in the operation sections of this specification is not implied.
2. For operating at elevated temperatures the device must be derated based on a +150°C maximum junction temperature and a thermal resistance of 50°C/W junction to ambient (40 pin IW package).
3. This product includes circuitry specifically designed for the protection of its internal devices from the damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying any voltages larger than the rated maxima.
4. Parameter valid over operating temperature range unless otherwise specified.
5. All voltage measurements are referenced to ground.
6. Manufacturer reserves the right to make design and process changes and improvements.
7. Typical values are at +25°C, nominal supply voltages, and nominal processing parameters.

INTERFACE SIGNALS

TYPE	PINS	ABBREVIATION	FUNCTION	SIGNAL SENSE
INPUT	1	GND	Ground	GND=0
INPUT	1	V_{CC}	+5 Volts ±5%	V_{CC}=1
INPUT	1	RESET	Chip Reset	RESET=1 (pulse), causes reset
INPUT	1	CLOCK	Chip Clock	
INPUT	1	$\overline{\text{PAUSE}}$	Temp. Halt execution	$\overline{\text{PAUSE}}$=0, temporarily halts execution
INPUT	1	$\overline{\text{INTREQ}}$	Interrupt Request	$\overline{\text{INTREQ}}$=0, requests interrupt
INPUT	1	$\overline{\text{OPACK}}$	Operation Acknowledge	$\overline{\text{OPACK}}$=0, acknowledges operation
INPUT	1	SENSE	Sense	SENSE=0 (low) or SENSE=1 (high)
INPUT	1	$\overline{\text{ADREN}}$	Address Enable	$\overline{\text{ADREN}}$=1 drives into third state
INPUT	1	$\overline{\text{DBUSEN}}$	Data Bus Enable	$\overline{\text{DBUSEN}}$=1 drives into third state
IN/OUT	8	DBUS0-DBUS7	Data Bus	DBUSn=0 (low), DBUSn=1 (high)
OUTPUT	13	ADR0-ADR12	Address 0 through 12	ADRn=0 (low), ADRn=1 (high)
OUTPUT	1	ADR13 or E/$\overline{\text{NE}}$	Address 13 or Extended/Non-Extended	Non-Extended=0, Extended=1
OUTPUT	1	ADR14 or D/$\overline{\text{C}}$	Address 14 or Data Control	Control=0, Data 1
OUTPUT	1	OPREQ	Operation Request	OPREQ=1, requests operation
OUTPUT	1	M/$\overline{\text{IO}}$	Memory/IO	IO=0, M=1
OUTPUT	1	$\overline{\text{R}}$/W	Read/Write	R=0, W=1
OUTPUT	1	FLAG	Flag Output	FLAG=1 (high), FLAG=0 (low)
OUTPUT	1	INTACK	Interrupt Acknowledge	INTACK=1, acknowledges interrupt
OUTPUT	1	RUN/$\overline{\text{WAIT}}$	Run/Wait Indicator	RUN=1, WAIT=0
OUTPUT	1	WRP	Write Pulse	WRP=1 (pulse), causes writing

PIN CONFIGURATION



TOP VIEW

FEATURES

INPUT/OUTPUT FACILITIES

The 2650 processor provides several mechanisms for performing input/output functions. They are flag and sense, non-extended I/O instructions, extended I/O instructions, and memory I/O. These four facilities are described below.

FLAG & SENSE I/O

The 2650 has the ability to directly output one bit of data without additional address decoding or synchronizing signals.

The bit labeled "Flag" in the Program Status Word is connected through a TTL compatible driver to the chip output at pin #40. The Flag output always reflects the value in the Flag bit.

When a program changes the Flag bit through execution of an LPSU, PPSU, or CPSU, the bit will be set or cleared during the last cycle of the instruction that changes it.

The Flag bit may be used conveniently for many different purposes. The following is a list of some possible uses:

1. A serial output channel
2. An additional address bit to increase addressing range.
3. A switch or toggle output to control external logic.
4. The origin of a pulse for polling chains of devices.

The Sense bit performs the complementary function of the Flag and is a single bit direct input to the 2650. The Sense input, pin #1 is connected to a TTL compatible receiver and is then routed directly to a bit position in the Program Status Word. The bit in the PSW always represents the value of the external signal. It may be sampled anytime through use of the TPSU or SPSU instructions.

This simple input to the processor may be used in many ways. The following is a list of some possible uses:

1. A serial input channel
2. A sense switch input
3. A break signal to a processing program
4. An input for yes/no signaling from external devices.

NON-EXTENDED I/O

There are four one byte I/O instructions; REDC, REDD, WRTC, and WRTD. They are all referred to as non-extended because they can communicate only one byte of data, either into or out of the 2650.

REDC and REDD causes the input transfer of one byte of data. They are identical except for the fact that the D/$\overline{C}$ Signal is in the D state for REDD and in the $\overline{C}$ state for REDC. Similarly, the instructions WRTC and WRTD cause an output transfer of one byte of data. The D/$\overline{C}$ line discriminates between the two pairs of input/output instructions. The D/$\overline{C}$ line can be used as a 1-bit device address in simple systems.

The read and write timing sequences for the one byte I/O instructions are the same as the memory read and write sequences with the following exceptions: the M/$\overline{IO}$ signal is switched to $\overline{IO}$, the D/$\overline{C}$ line becomes valid, E/$\overline{NE}$ is switched to $\overline{NE}$ (non-extended), and the Address bus contains no valid information.

The $\overline{NE}$ signal informs the devices outside the 2650 that a one byte I/O instruction is being executed. The D/$\overline{C}$ line indicates which pair of the one byte I/O instructions are being executed; D implies either WRTD or REDD, and $\overline{C}$ implies either WRTC or REDC. Finally, to determine whether it is a read or a write, examine the $\overline{R}$/W signal level.

Table 6 illustrates the sense of the interface signals. The "Signal Timing" section should be referenced for the exact timing relationships. It should be remembered that the control signals are not to be considered valid except when the OPREQ signal is valid.

TABLE 6. I/O INTERFACE SIGNALS

	OPREQ	M/$\overline{IO}$	$\overline{R}$/W	ADR13-E/$\overline{NE}$	ADR14-D/$\overline{C}$
MEMORY READ	T	M	$\overline{R}$	ADR13	ADR14
MEMORY WRITE	T	M	W	ADR13	ADR14
2 BYTE READ	T	$\overline{IO}$	$\overline{R}$	E	Don't Care
2 BYTE WRITE	T	$\overline{IO}$	W	E	Don't Care
1 BYTE CONTROL READ	T	$\overline{IO}$	$\overline{R}$	$\overline{NE}$	$\overline{C}$
1 BYTE CONTROL WRITE	T	$\overline{IO}$	W	$\overline{NE}$	$\overline{C}$
1 BYTE DATA READ	T	$\overline{IO}$	$\overline{R}$	$\overline{NE}$	D
1 BYTE DATA READ	T	$\overline{IO}$	W	$\overline{NE}$	D

EXTENDED I/O

There are two, two byte I/O instructions; REDE and WRTE. They are referred to as extended because they can communicate two bytes of data when they are executed. The REDE causes the second byte of the instruction to be output on the low order address lines, ADR0-ADR7, which is intended to be used as a device address while the byte of data then on the Data Bus will be strobed into the register specified in the instruction. The WRTE also presents the second byte of the instruction on the Address Bus, but a byte of data from the register specified in the instruction is simultaneously output on the Data Bus.

The two byte I/O instructions are similar to the one byte I/O instructions except: the D/$\overline{C}$ line is not considered, and the data from the second byte of the I/O instruction appears on the Address Bus all during the time that OPREQ is valid. The data on the Address Bus is intended to convey a device address, but may be utilized for any purpose.

Table 6 illustrates the sense of the interface signals for extended I/O instructions. Refer to "Signal Timing" section for exact timing relationships.

MEMORY I/O

The 2650 user may choose to transfer data into or out of the processor using the memory control signals. The advantage to this technique is that the data can be read or written by the program through ordinary instruction execution and data may be directly operated upon with the arithmetic instructions.

To make use of this technique, the designer has to assign memory addresses to devices and design the device interfaces to generate the same signals as memory.

A disadvantage to this method is that it may be necessary to decode more address lines to determine the device address than with other I/O facilities.

INTERRUPT MECHANISM

The 2650 has been implemented with a conventional, single level, address vectoring interrupt mechanism. There is one interrupt input pin. When an external device generates an interrupt signal ($\overline{\text{INTREQ}}$), the processor is forced to transfer control to any of 128 possible memory locations as determined by an 8-bit vector supplied by the interrupting device.

Of special interest is that the device may return a relative indirect address signal which causes the processor to enter an indirect addressing sequence upon receipt of an interrupt. This enables a device to direct the processor to execute code anywhere within addressable memory.

Upon recognizing the interrupt signal, the processor automatically sets the Interrupt Inhibit bit in the Program Status Word. This inhibits further interrupts from being recognized until the interrupt routine is finished executing and a Return-and-Enable instruction is executed or the inhibit bit is explicitly cleared.

When the inhibit bit in the PSW is set, the processor will not recognize an interrupt input. The Interrupt Inhibit bit may be set under program control (LPSU, PPSU) and is automatically set whenever the processor accepts an interrupt. The inhibit bit may be cleared in three ways:

1. By a RESET operation
2. By execution of an appropriate clear or load PSU instruction; (CPSU, LPSU)
3. By execution of a Return-and-Enable instruction.

The sequence of events for a normal interrupt operation is as follows:

1. An executing program enables interrupts.
2. External device initiates interrupt with the $\overline{\text{INTREQ}}$ line.
3. Processor finishes executing current instruction.
4. Processor sets inhibit bit.
5. Processor inserts the first byte of ZBSR (Zero Branch-to-Subroutine, Relative) instruction into the instruction register instead of what would have been the next sequential instruction.
6. Processor accesses the data bus to fetch the second byte of the ZBSR instruction.
7. Interrupting device responds to the Processor generated INTACK (Interrupt Acknowledge) by supplying the requested second byte.
8. The processor executes the Zero Branch-to-Subroutine instruction, saving the address of the instruction following the interrupted instruction in the RAS, and proceeds to execute the instruction at page 0, byte 0, or the address relative to page 0, byte 0 as given by the interrupting device.
9. When the interrupt routine is complete, a return instruction (RETC, RETE) pulls the address from the RAS and execution of the interrupted program resumes.

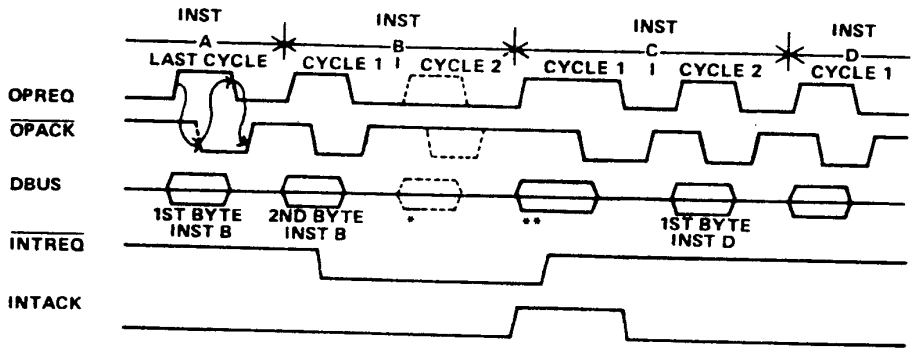
Since the interrupting device specifies the interrupt subroutine address in the standard relative address format, it has considerable flexibility with regard to the interrupt procedure. It can point to any location that is within +63 or −64 bytes of page zero, byte zero of memory. (Negative relative addresses wrap around the memory, modulo $8{,}192_{10}$ bytes.) The interrupting device also may specify whether the subroutine address is direct or indirect by providing a zero or one to DBUS #7 (pin #26). If the external device is not complex enough to exercise these options, it may respond to the INTACK operation with a byte of all zeroes. In such a case, the processor will execute a direct Branch-to-Subroutine to page zero, byte zero of memory.

The timing diagram in Figure 12 will help explain how the interrupt system works in the processor. The execution of the instruction labeled "A" has been proceeding before the start of this diagram. The last cycle of instruction "A" is shown. Notice that, as in all external operations, the OPREQ output eventually causes an $\overline{\text{OPACK}}$ input, which in turn allows OPREQ to be turned off. The arrows show this sequence of events. The last cycle of instruction "A" fetches the first byte of instruction "B" from Memory and inserts it into the Instruction Register.

Assume that instruction "B" is a two cycle, two byte instruction with no operand fetch (e.g., ADDI). Since the first byte has already been fetched by instruction "A", the first cycle of instruction "B" is used to fetch the second byte of instruction "B". Had instruction "B" not been interrupted, it would have fetched the first byte of the next sequential instruction during its second (last) cycle. The dotted lines indicate that operation.

Since instruction "B" is interrupted, however, the last cycle of "B" is used to insert the interrupt instruction (ZBSR) into the instruction register. Notice that the $\overline{\text{INTREQ}}$ input can arrive at any time. Instruction B is interrupted since $\overline{\text{INTREQ}}$ occured prior to the last (2nd) cycle of execution.

Instead of being the next sequential instruction following "B", instruction "C" is the completion of the interrupt. The first cycle of "C" is used to fetch the second byte of the ZBSR instruction from the DBUS as provided by the interrupting device. This fact is indicated by the presence of the INTACK control signal. The $\overline{\text{INTREQ}}$ may then be removed. When the device responds with the requested byte, it uses a standard operation acknowledge procedure ($\overline{\text{OPACK}}$) to so indicate to the processor. During the second cycle of instruction "C" the processor executes the ZBSR instruction, and fetches the first byte of instruction "D" which is located at the subroutine address.



* PROCESSOR INSERTS 1ST BYTE OF ZBSR INSTRUCTION. ADDRESS OF 1ST BYTE OF INSTC IS PUSHED INTO RETURN ADDRESS STACK.

** 2ND BYTE OF ZBSR (INTERRUPT VECTOR)

Figure 12. INTERRUPT TIMING

SUBROUTINE LINKAGE

The on-chip stack, along with the Branch-to-Subroutine and Return instructions provide the facility to transfer control to a subroutine. The subroutine can return control to the program that branched to it via a Return instruction.

The stack is eight levels deep which means that a routine may branch to a subroutine, which may branch to another subroutine, etc., eight times before any Return instructions are executed.

When designing a system that utilizes interrupts, it should be remembered that the processor jams a ZBSR into the IR and then executes it. This will cause an entry to be pushed into the on-chip stack like any other Branch-to-Subroutine instruction and may limit the stack depth available in certain programs.

When branching to a subroutine, the following sequence of events occurs:

1. The address in the IAR is used to fetch the Branch-to-Subroutine instruction and is then incremented in the Address Adder so that it points to the instruction following the subroutine branch.
2. The Stack Pointer is incremented by one so that it points to the next Return Address Stack location.
3. The contents of the IAR are stored in the stack at the location designated by the Stack Pointer.
4. The operand address contained in the Branch-to-Subroutine instruction (the address of the first instruction of the subroutine) is inserted into the IAR.

When returning from a subroutine, this sequence of events occurs:

1. The address in the IAR is used to fetch the return (RETC, RETE) instruction from memory.
2. When the return instruction is recognized by the processor, the contents of the stack entry pointed to by the Stack Pointer is placed into the IAR.
3. The Stack Pointer is decremented by one.
4. Instruction execution continues at the address now in the IAR.

CONDITION CODE USAGE

The two-bit register, called the Condition Code, is incorporated in the Program Status Word. It may be seen in the description of the 2650 instructions, that the Condition Code (CC) is specifically set by every instruction that causes data to be transferred into a general purpose register and it is also set by compare instructions.

The reason for this design feature is that after an instruction executes, the CC contains a modest amount of information about the byte of data which has just been manipulated. For example, a program loads register one with a byte of unknown data and the Condition Code setting indicates that the byte is positive, negative or zero. The negative indication implies that bit #7 is set to one.

Consequently, a data manipulation operation when followed by a conditional branch is often sufficient to determine desired information without resorting to a specific test, thus saving instructions and memory space.

In the following example, the Condition Code is used to test the parity of a byte of data which is stored at symbolic memory location CHAR.

```
EQ      EQU     0       THE EQUAL CONDITION CODE
CHAR    DATA    2       UNKNOWN DATA BYTE
WC      EQU     H'04'   THE WITH CARRY BIT
NEG     EQU     2       CC MASK
        CPSL    WC      CLEAR CARRY BIT
        LODI,R2 -8      SET UP COUNTER
        SUBZ    R0      CLEAR REG 0
        LODR,R1 CHAR    GET THE CHARACTER (cc is set)
LOOP    BCFR,NEG G01    IF NOT SET, DON'T COUNT (cc is
                          tested)
        ADDI,R0 +1      COUNT THE BIT
G01     RRL,R1          MOVE BITS LEFT (cc is set)
        BIRR,R2 LOOP    LOOP TILL DONE

*       FINISHED,TEST IF REG 0 HAS A ONE IN LOW ORDER
*       IF BIT #0 = 1, ODD PARITY. IF BIT #0 = 0, THEN EVEN.

        TMI,R0  H'01'
        BCTR,EQ ODD
EVEN    HALT
ODD     HALT
```

START-UP PROCEDURE

The 2650 processor, having no internal start-up procedure must be started in an orderly fashion to assure that the internal control logic begins in a known state.

Assuming power is applied to the chip and the clock input is running, the easiest way to start is to apply a Reset signal for at least three clock periods. When the RESET signal is removed the processor will fetch the instruction at page 0, byte 0 and commence ordinary instruction execution.

To start processing at a specific address, a more complex start-up procedure may be employed. If an Interrupt signal is applied initially along with the Reset, processing will commence at the address provided by the interrupting device. Recall that the address provided may include a bit to specify indirect addressing and therefore the first instruction executed may be anywhere within addressable memory. The Reset and Interrupt signal may be applied simultaneously and when the Reset is removed, the processor will execute the usual interrupt signal sequence as described in INTERRUPT MECHANISM. There is an example of a start-up technique in the System Application Notes.

INSTRUCTIONS

ADDRESSING MODES

An addressing mode is a method the processor uses for developing argument addresses for machine instructions.

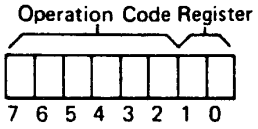
The 2650 processor can develop addresses in eight ways:

- Register addressing
- Immediate addressing
- Relative addressing
- Relative, indirect addressing
- Absolute addressing
- Absolute, indirect addressing
- Absolute, indexed addressing
- Absolute, indirect, indexed addressing

However, of these eight addressing modes, only four of them are basic. The others are variations due to indexing and indirection. The basic addressing mode of each instruction is indicated in parentheses in the first line of each detailed instruction description. The following text describes how effective addresses are developed by the processor.

REGISTER ADDRESSING

All register-to-register instructions are one byte in length. Instructions utilizing this addressing mode appear in this general format.

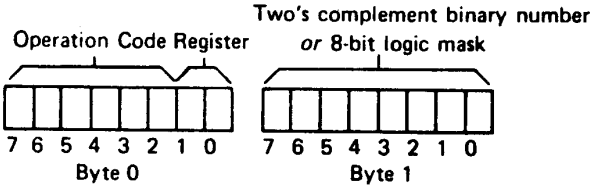


Since there are only two bits designated to specify a register, register zero always contains one of the operands while the other operand is in one of the three registers in the currently selected bank. Register zero may also be specified as the explicit operand giving instructions such as: LODZ R0.

In one byte register addressing instructions which have just one operand, any of the currently selected general purpose registers or register zero may be specified, e.g., RRL,R0.

IMMEDIATE ADDRESSING

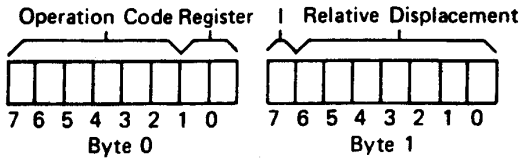
All immediate addressing instructions are two bytes in length. The first byte contains the operation code and register designation, while the second byte contains data used as the argument during instruction execution.



The second byte, the data byte, may contain a binary number or a logic mask depending on the particular instruction being executed. Any register may be designated in the first byte.

RELATIVE ADDRESSING

Relative addressing instructions are all two bytes in length and are memory reference instructions. One argument of the instruction is a register and the other argument is the contents of a memory location. The format of relative addressing instructions is:



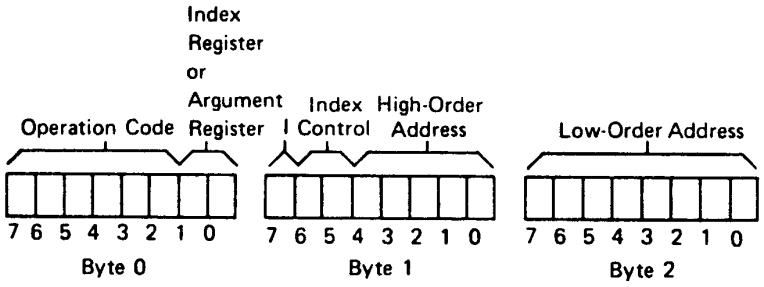
The first byte contains the operation code and register designation, while the second byte contains the relative address. Bits 0—6, byte 1, contain a 7-bit two's complement binary number which can range from −64 to +63. This number is used by the processor to calculate the effective address. The effective address is calculated by adding the address of the first byte following a relative addressing instruction to the relative displacement in the second byte of the instruction.

If bit 7, byte 1 is set to "1", the processor will enter an indirect addressing cycle, where the actual operand address will be accessed from the effective address location. See Indirect Addressing.

Two of the branch instructions (ZBSR, ZBRR) allow addressing relative to page zero, byte 0 of memory. In this case, values up to +63 reference the first 63 bytes of page zero and values up to -64 reference the last 64 bytes of page zero.

ABSOLUTE ADDRESSING FOR NON-BRANCH INSTRUCTIONS

Absolute addressing instructions are all three bytes in length and are memory reference instructions. One argument of the instruction is a register, designated in bits 1 and 0, byte 0; the other argument is the contents of a memory location. The format of absolute addressing instructions is:



Bits 4—0, byte 1 and 7—0, byte 2 contain the absolute address and can address any byte within the same page that the instruction appears.

The index control bits, bits #6 and #5, byte 1 determine how the effective address will be calculated and possibly which register will be the argument during instruction execution. The index control bits have the following interpretation:

Index Control Bit 6	Index Control Bit 5	Meaning
0	0	Non-indexed address
0	1	Indexed with auto-increment
1	0	Indexed with auto-decrement
1	1	Indexed only

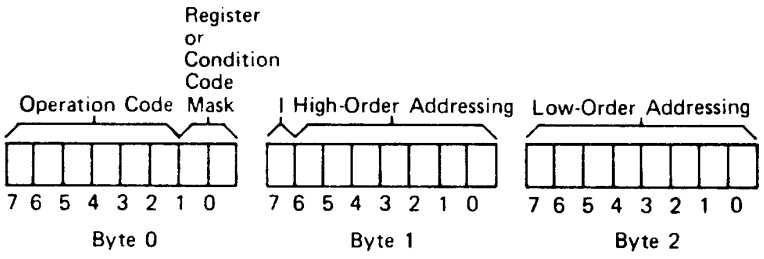
When the index control bits are 0 & 0, bits #1 and #0 in byte 0 contain the argument register designation and bits 0 to 4, byte 1 and bits 0 to 7, byte 2 contain the effective address. Indirect addressing may be specified by setting bit #7, byte 1 to a one.

When the index control bits are 1 & 1, bits #1 and #0 in byte 0 designate the index register and the argument register implicitly becomes register zero. The effective address is calculated by adding the contents of the index register (8-bit absolute integer) to the address field. If indirect addressing is specified, the indirect address is accessed and then the value in the index register is added to the indirect address. This is commonly called post indexing.

When the index control bits contain 0 & 1, the address is calculated by the processor exactly as when the control bits contain 1 & 1 *except* a binary 1 is added to the contents of the selected index register *before* the calculation of the effective address proceeds. Similarly, when the index control bits contain 1 & 0, a binary 1 is subtracted from the contents of the selected index register *before* the effective address is calculated.

ABSOLUTE ADDRESSING FOR BRANCH INSTRUCTIONS

The three byte, absolute addressing, branch instructions deviate slightly in format from ordinary absolute addressing instructions as shown below:



The notable difference is that bits 6 and 5, byte 1, are no longer interpreted as Index Control bits, but instead are interpreted as the high order bits of the address field. This means that there is no indexing allowed on most absolute addressing branch instructions, but indexed branches are possible through use of the BXA and BSXA instructions. The bits #6 and #5, byte 1, are used to set the current page register, thus enabling programs to directly transfer control to another page.

See the MEMORY ORGANIZATION, BXA and BSXA instructions, and INDIRECT ADDRESSING.

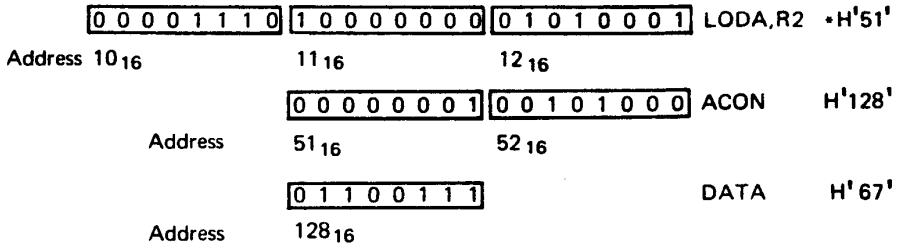
INDIRECT ADDRESSING

Indirect addressing means that the argument address of an instruction is not specified by the instruction itself, but rather the argument address will be found in the two bytes pointed to by the address field or relative address field, of absolute or relative addressing instructions. In the case of absolute addressing, the value of the index register is added to the indirect address *not* to the value in the address field of the instruction. In both cases, the processor will enter the indirect addressing state when the bit designated "I" is set to one. Entering the indirect addressing sequence adds two cycles (6 clock periods) to the execution time of an instruction.

Indirect addresses are 15-bit addresses stored right justified in two contiguous bytes of memory. As such, an indirect address may specify any location in addressable memory (0—32,767). The high order bit of the two byte indirect address is not used by the processor.

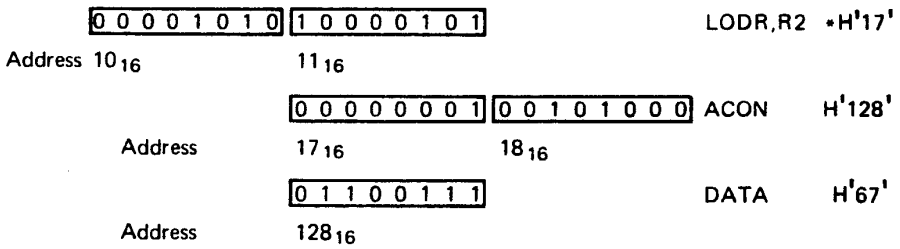
Only single level indirect addressing is implemented. The following examples demonstrate indirect addressing.

Example 1.



The LODA instruction in memory locations 10, 11, and 12 specifies indirect addressing (bit 7, byte 1, is set). Therefore, when the instruction is executed, the processor takes the address field value, H' 51', and uses it to access the two byte indirect address at 51 and 52. Then using the contents of 51 and 52 as the effective address, the data byte containing H' 67' is loaded into register 2.

Example 2.



In a fashion similar to the previous example, the relative address is used to access the indirect address which points to the data byte. When the LODR instruction is executed, the data byte contents, H' 67', will be loaded into register 2.

INSTRUCTIONS FORMAT EXCEPTIONS

There are several instructions which are detected by decoding the entire 8 bits of the first byte of the instruction. These instructions are unique and may be noticed in the instruction descriptions. Examples are: HALT, CPSU, CPSL.

Of this type of instruction, two operation codes were taken from otherwise complete sets thus eliminating certain possible operations. The cases are as follows:

(NOT OKAY)	STRZ	0	Storing register zero into register zero is not implemented, the operation code is used for NOP (no operation).
(OKAY)	NOP		
(NOT OKAY)	ANDZ	0	AND of register zero with register zero is not implemented, the operation code is used for HALT.
(OKAY)	HALT		

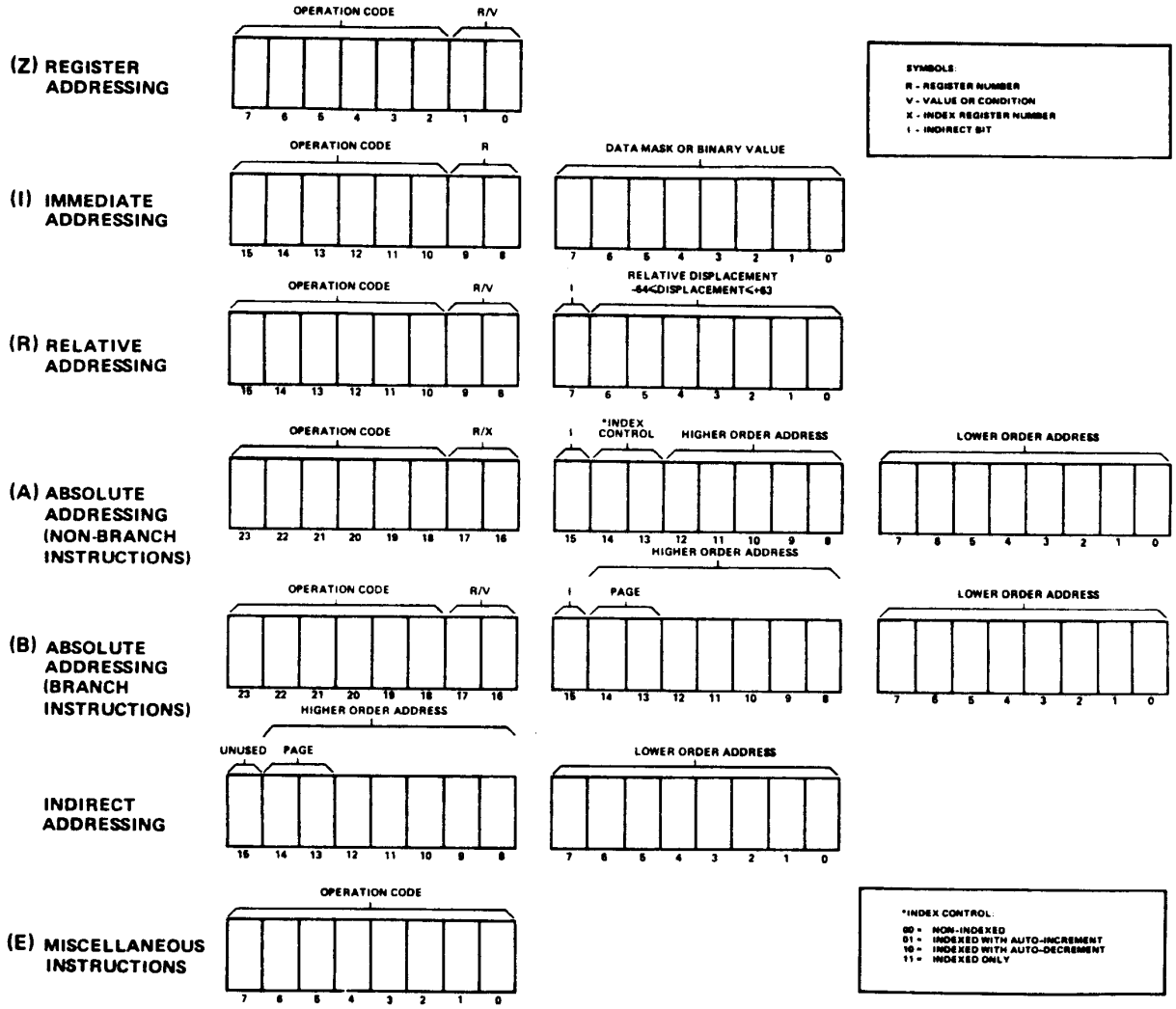


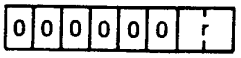
Figure 13. INSTRUCTION FORMATS

LOAD REGISTER ZERO

(Register Addressing)

Mnemonic LODZ r

Binary Coding



Execution Time 2 cycles (6 clock periods)

Description

This one-byte instruction transfers the contents of the specified register, r, into register zero. The previous contents of register zero are lost. The contents of register r remain unchanged.

When the specified register, r, equals 0, the operation code is changed to 60_{16} by the assembler. The instruction, 00000000, yields indeterminate results.

Processor Registers Affected CC

Condition Code Setting

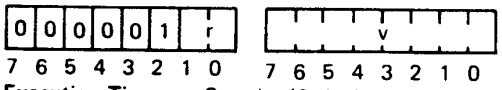
Register Zero	CC1	CC0
Positive	0	1
Zero	0	0
Negative	1	0

LOAD IMMEDIATE

(Immediate Addressing)

Mnemonic LODI,r v

Binary Coding



Execution Time 2 cycles (6 clock periods)

Description

This two-byte instruction transfers the second byte of the instruction, v, into the specified register, r. The previous contents of r are lost.

Processor Registers Affected CC

Condition Code Setting

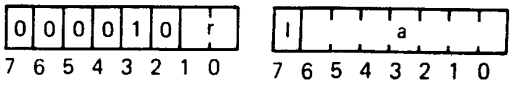
Register r	CC1	CC0
Positive	0	1
Zero	0	0
Negative	1	0

LOAD RELATIVE

(Relative Addressing)

Mnemonic LODR,r (*)a

Binary Coding



Execution Time 3 cycles (9 clock periods)

Description

This two-byte instruction transfers a byte of data from memory into the specified register, r. The data byte is found at the effective address formed by the addition of the a field and the address of the byte following this instruction. The previous contents of register r are lost. Indirect addressing may be specified.

Processor Registers Affected CC

Condition Code Setting

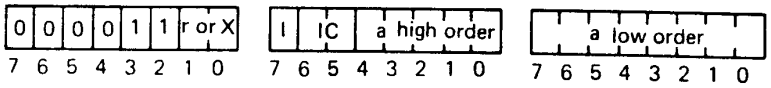
Register r	CC1	CC0
Positive	0	1
Zero	0	0
Negative	1	0

LOAD ABSOLUTE

(Absolute Addressing)

Mnemonic LODA,r (*)a(,X)

Binary Coding



Execution Time 4 cycles (12 clock periods)

Description

This three-byte instruction transfers a byte of data from memory into the specified register, r. The data byte is found at the effective address. If indexing is specified, bits 1 and 0, byte 0, indicate the index register and the destination of the operation implicitly becomes register zero. The previous contents of register r are lost.

Indirect addressing and/or indexing may be specified.

Processor Registers Affected CC

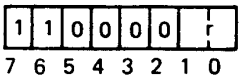
Condition Code Setting

Register r	CC1	CC0
Positive	0	1
Zero	0	0
Negative	1	0

STORE REGISTER ZERO (Register Addressing)

Mnemonic STRZ r

Binary Code



Execution Time 2 cycles (6 clock periods)

Description

This one-byte instruction transfers the contents of register zero into the specified register r. The previous contents of register r are lost. The contents of register zero remain unchanged.

Note: Register r may not be specified as zero. This operation code, '11000000', is reserved for NOP.

Processor Registers Affected CC

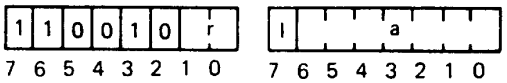
Condition Code Setting

Register r	CC1	CC0
Positive	0	1
Zero	0	0
Negative	1	0

STORE RELATIVE (Relative Addressing)

Mnemonic STRR,r (*)a

Binary Code



Execution Time 3 cycles (9 clock periods)

Description

This two-byte instruction transfers a byte of data from the specified register, r, into the byte of memory pointed to by the effective address. The contents of register r remain unchanged and the contents of the memory byte are replaced.

Indirect addressing may be specified.

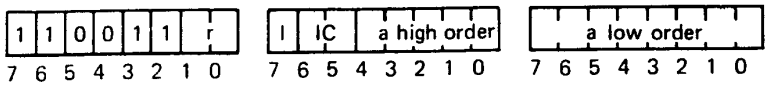
Processor Registers Affected None

Condition Code Setting N/A

STORE ABSOLUTE (Absolute Addressing)

Mnemonic STRA,r (*)a(,X)

Binary Code



Execution Time 4 cycles (12 clock periods)

Description

This three-byte instruction transfers a byte of data from the specified register, r, into the byte of memory pointed to by the effective address. The contents of register r remain unchanged and the contents of the memory byte are replaced.

Indirect addressing and/or indexing may be specified. If indexing is specified, bits 1 and 0, byte 0, indicate the index register and the destination of the operation implicitly becomes register zero.

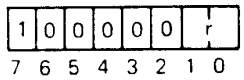
Processor Registers Affected None

Condition Code Setting N/A

ADD TO REGISTER ZERO (Register Addressing)

Mnemonic ADDZ r

Binary Code



Execution Time 2 cycles (6 clock periods)

Description

This one-byte instruction causes the contents of the specified register, r, and the contents of register zero to be added together in a true binary adder. The 8-bit sum of the addition replaces the contents of register zero. The contents of register r remain unchanged.

Note: Add with Carry may be effected. See Carry bit.

Processor Registers Affected C, CC, IDC, OVF

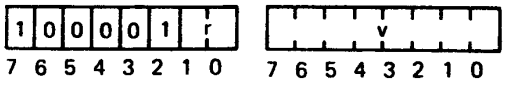
Condition Code Setting

Register Zero	CC1	CC0
Positive	0	1
Zero	0	0
Negative	1	0

ADD IMMEDIATE (Immediate Addressing)

Mnemonic ADDI,r v

Binary Coding



Execution Time 2 cycles (6 clock periods)

Description

This two-byte instruction causes the contents of register r and the contents of the second byte of this instruction to be added together in a true binary adder. The eight-bit sum replaces the contents of register r.

Note: Add with Carry may be effected. See Carry bit.

Processor Registers Affected C, CC, IDC, OVF

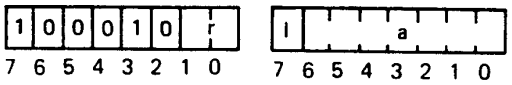
Condition Code Setting

Register r	CC1	CC0
Positive	0	1
Zero	0	0
Negative	1	0

ADD RELATIVE (Relative Addressing)

Mnemonic ADDR,r (*)a

Binary Coding



Execution Time 3 cycles (9 clock periods)

Description

This two-byte instruction causes the contents of register r and the contents of the byte of memory pointed to by the effective address to be added together in a true binary adder. The eight-bit sum replaces the contents of register r.

Indirect addressing may be specified.

Note: Add with Carry may be effected. See Carry bit.

Processor Registers Affected C, CC, IDC, OVF

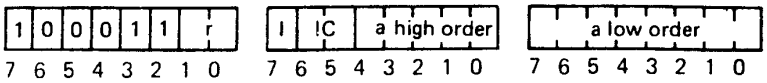
Condition Code Setting

Register r	CC1	CC0
Positive	0	1
Zero	0	0
Negative	1	0

ADD ABSOLUTE (Absolute Addressing)

Mnemonic ADDA,r (*)a(,X)

Binary Coding



Execution Time 4 cycles (12 clock periods)

Description

This three-byte instruction causes the contents of register r and the contents of the byte of memory pointed to by the effective address to be added together in a true binary adder. The eight-bit sum replaces the contents of register r.

Indirect addressing and/or indexing may be specified. If indexing is specified, bits 1 and 0, byte 0, indicate the index register and the destination of the operation implicitly becomes register zero.

Note: Add with Carry may be effected. See Carry bit.

Processor Registers Affected C, CC, IDC, OVF

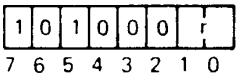
Condition Code Setting

Register r	CC1	CC0
Positive	0	1
Zero	0	0
Negative	1	0

SUBTRACT FROM REGISTER ZERO (Register Addressing)

Mnemonic SUBZ r

Binary Coding



Execution Time 2 cycles (6 clock periods)

Description

This one-byte instruction causes the contents of the specified register r to be subtracted from the contents of register zero. The result of the subtraction replaces the contents of register zero.

The subtraction is performed by taking the binary two's complement of the contents of register r and adding that result to the contents of register zero. The contents of register r remain unchanged.

Note: Subtract with Borrow may be effected. See Carry bit.

Processor Registers Affected C, CC, IDC, OVF

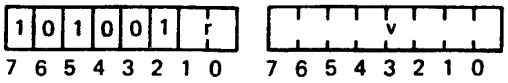
Condition Code Setting

Register Zero	CC1	CC0
Positive	0	1
Zero	0	0
Negative	1	0

SUBTRACT IMMEDIATE (Immediate Addressing)

Mnemonic SUBI,r v

Binary Code



Execution Time 2 cycles (6 clock periods)

Description

This two-byte instruction causes the contents of the second byte of this instruction to be subtracted from the contents of register r. The result of the subtraction replaces the contents of register r.

The subtraction is performed by taking the binary two's complement of the contents of the second instruction byte and adding that result to the contents of register r.

Note: Subtract with Borrow may be effected. See Carry bit.

Processor Registers Affected C, CC, IDC, OVF

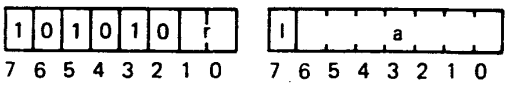
Condition Code Setting

Register r	CC1	CC0
Positive	0	1
Zero	0	0
Negative	1	0

SUBTRACT RELATIVE (Relative Addressing)

Mnemonic SUBR,r (*)a

Binary Code



Execution Time 3 cycles (9 clock periods)

Description

This two-byte instruction causes the contents of the byte of memory pointed to by the effective address to be subtracted from the contents of register r. The result of the subtraction replaces the contents of register r.

The subtraction is performed by taking the binary two's complement of the contents of the byte of memory and adding that result to the contents of register r.

Indirect addressing may be specified.

Note: Subtract with Borrow may be effected. See Carry bit.

Processor Registers Affected C, CC, IDC, OVF

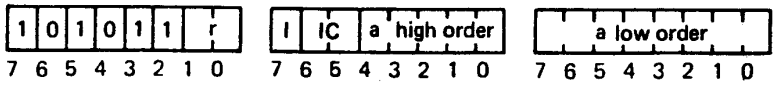
Condition Code Setting

Register r	CC1	CC0
Positive	0	1
Zero	0	0
Negative	1	0

SUBTRACT ABSOLUTE (Absolute Addressing)

Mnemonic SUBA,r (*)a(,X)

Binary Code



Execution Time 4 cycles (12 clock periods)

Description

This three-byte instruction causes the contents of the byte of memory pointed to by the effective address to be subtracted from the contents of register r. The result of the subtraction replaces the contents of register r.

The subtraction is performed by taking the binary two's complement of the contents of the memory byte and adding that result to the contents of register r.

Indirect addressing and/or indexing may be specified. If indexing is specified, bits 1 and 0, byte 0, indicate the index register and the destination of the operation implicitly becomes register zero.

Note: Subtract with Borrow may be effected. See Carry bit.

Processor Registers Affected C, CC, IDC, OVF

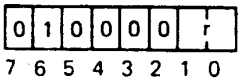
Condition Code Setting

Register r	CC1	CC0
Positive	0	1
Zero	0	0
Negative	1	0

AND TO REGISTER ZERO (Register Addressing)

Mnemonic ANDZ r

Binary Code



Execution Time 2 cycles (6 clock periods)

Description

This one-byte instruction causes the contents of the specified register, r, to be logically ANDed with the contents of register zero. The result of the operation replaces the contents of register zero. The contents of register r remain unchanged.

The AND operation treats each bit of the argument bytes as in the truth table below:

Bit (0-7)	Bit (0-7)	AND Result
0	0	0
0	1	0
1	1	1
1	0	0

Note: Register r may not be specified as zero. This operation code, '01000000', is reserved for HALT.

Processor Registers Affected CC

Condition Code Setting

Register Zero	CC1	CC0
Positive	0	1
Zero	0	0
Negative	1	0

AND IMMEDIATE (Immediate Addressing)

Mnemonic ANDI,r v

Binary Code

Execution Time 2 cycles (6 clock periods)

Description

This two-byte instruction causes the contents of the specified register r to be logically ANDed with the contents of the second byte of this instruction. The result of this operation replaces the contents of register r.

The AND operation treats each bit of the argument bytes as in the truth table below:

Bit (0-7)	Bit (0-7)	AND Result
0	0	0
0	1	0
1	1	1
1	0	0

Processor Registers Affected CC

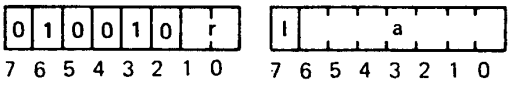
Condition Code Setting

Register Zero	CC1	CC0
Positive	0	1
Zero	0	0
Negative	1	0

AND RELATIVE (Relative Addressing)

Mnemonic ANDR,r (*)a

Binary Code



Execution Time 3 cycles (9 clock periods)

Description

This two-byte instruction causes the contents of the specified register r to be logically ANDed with the contents of the memory byte pointed to by the effective address. The result of this operation replaces the contents of register r.

The AND operation treats each bit of the argument bytes as in the truth table below:

Bit (0-7)	Bit (0-7)	AND Result
0	0	0
0	1	0
1	1	1
1	0	0

Processor Registers Affected CC

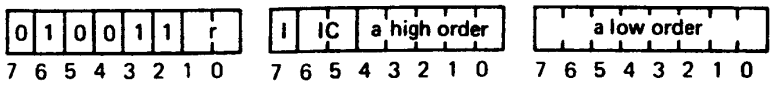
Condition Code Setting

Register Zero	CC1	CC0
Positive	0	1
Zero	0	0
Negative	1	0

AND ABSOLUTE (Absolute Addressing)

Mnemonic ANDA,r (*)a(,X)

Binary Code



Execution Time 4 cycles (12 clock periods)

Description

This three-byte instruction causes the contents of Register r to be logically ANDed with the contents of memory byte pointed to by the effective address. The result of the operation replaces the contents of register r.

The AND operation treats each bit of the argument bytes as in the truth table below:

Bit (0-7)	Bit (0-7)	AND Result
0	0	0
0	1	0
1	1	1
1	0	0

Indirect addressing and/or indexing may be specified. If indexing is specified, bits 1 and 0, byte 0, indicate the index register and the destination of the operation implicitly becomes register zero.

Processor Registers Affected CC

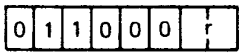
Condition Code Setting

Register Zero	CC1	CC0
Positive	0	1
Zero	0	0
Negative	1	0

INCLUSIVE OR TO REGISTER ZERO (Register Addressing)

Mnemonic IORZ r

Binary Code



Execution Time 2 cycles (6 clock periods)

Description

This one-byte instruction causes the contents of the specified register, r, to be logically Inclusive ORed with the contents of register zero. The result of this operation replaces the contents of register zero. The contents of register r remain unchanged.

The Inclusive OR operation treats each bit of the argument bytes as in the truth table below:

Bit (0-7)	Bit (0-7)	Inclusive OR Result
0	0	0
0	1	1
1	1	1
1	0	1

Processor Registers Affected CC

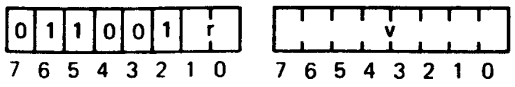
Condition Code Setting

Register Zero	CC1	CC0
Positive	0	1
Zero	0	0
Negative	1	0

INCLUSIVE OR IMMEDIATE (Immediate Addressing)

Mnemonic IORI,r v

Binary Code



Execution Time 2 cycles (6 clock periods)

Description

This two-byte instruction causes the contents of the specified register r to be logically Inclusive ORed with the contents of the second byte of this instruction. The result of this operation replaces the contents of register r.

The Inclusive OR operation treats each bit of the argument bytes as in the truth table below:

Bit (0-7)	Bit (0-7)	Inclusive OR Result
0	0	0
0	1	1
1	1	1
1	0	1

Processor Registers Affected CC

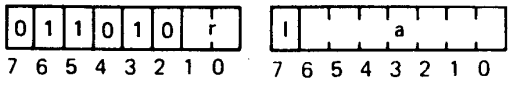
Condition Code Setting

Register r	CC1	CC0
Positive	0	1
Zero	0	0
Negative	1	0

INCLUSIVE OR RELATIVE (Relative Addressing)

Mnemonic IORR,r (*)a

Binary Code



Execution Time 3 cycles (9 clock periods)

Description

This two-byte instruction causes the contents of the specified register r to be logically Inclusive ORed with the contents of the memory byte pointed to by the effective address. The result of this operation replaces the previous contents of register r.

Indirect addressing may be specified.

The Inclusive OR operation treats each bit of the argument byte as in the truth table below:

Bit (0-7)	Bit (0-7)	Inclusive OR Result
0	0	0
0	1	1
1	1	1
1	0	1

Processor Registers Affected CC

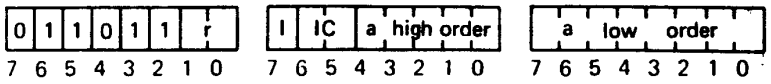
Condition Code Setting

Register r	CC1	CC0
Positive	0	1
Zero	0	0
Negative	1	0

INCLUSIVE OR ABSOLUTE (Absolute Addressing)

Mnemonic IORA,r (*)a(,X)

Binary Code



Execution Time 4 cycles (12 clock periods)

Description

This three-byte instruction causes the contents of register r to be logically Inclusive ORed with the contents of the memory byte pointed to by the effective address. The result of the operation replaces the previous contents of register r.

Indirect addressing and/or indexing may be specified. If indexing is specified, bits 1 and 0, byte 0, indicate the index register and the destination of the operation implicitly becomes register zero.

The Inclusive OR operation treats each bit of the argument bytes as in the truth table below:

Bit (0-7)	Bit (0-7)	Inclusive OR Result
0	0	0
0	1	1
1	1	1
1	0	1

Processor Registers Affected CC

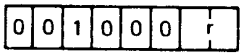
Condition Code Setting

Register Zero	CC1	CC0
Positive	0	1
Zero	0	0
Negative	1	0

EXCLUSIVE OR TO REGISTER ZERO (Register Addressing)

Mnemonic EORZ r

Binary Code



Execution Time 2 cycles (6 clock periods)

Description

This one-byte instruction causes the contents of the specified register r to be logically Exclusive ORed with the contents of register zero. The result of this operation replaces the contents of register zero. The contents of registet r remain unchanged.

The Exclusive OR operation treats each bit of the argument bytes as in the truth table below:

Bit (0-7)	Bit (0-7)	Exclusive OR Result
0	0	0
0	1	1
1	1	0
1	0	1

Processor Registers Affected CC

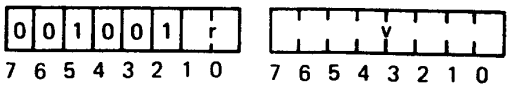
Condition Code Setting

Register Zero	CC1	CC0
Positive	0	1
Zero	0	0
Negative	1	0

EXCLUSIVE OR IMMEDIATE (Immediate Addressing)

Mnemonic EORI,r v

Binary Code



Execution Time 2 cycles (6 clock periods)

Description

This two-byte instruction causes the contents of the specified register r to be logically Exclusive ORed with the contents of the second byte of this instruction. The result of this operation replaces the previous contents of register r.

The Exclusive OR operation treats each bit of the argument bytes as in the truth table below:

Bit (0-7)	Bit (0-7)	Exclusive OR Result
0	0	0
0	1	1
1	1	0
1	0	1

Processor Registers Affected CC

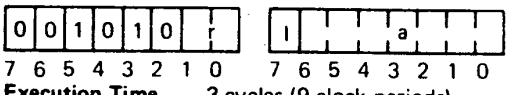
Condition Code Setting

Register r	CC1	CC0
Positive	0	1
Zero	0	0
Negative	1	0

EXCLUSIVE OR RELATIVE (Relative Addressing)

Mnemonic EORR,r (*)a

Binary Code



Execution Time 3 cycles (9 clock periods)

Description

This two-byte instruction causes the contents of the specified register r to be logically Exclusive ORed with the contents of the memory byte pointed to by the effective address. The result of this operation replaces the previous contents of register r.

Indirect addressing may be specified.

The Exclusive OR operation treats each bit of the argument bytes as in the truth table below:

Bit (0-7)	Bit (0-7)	Exclusive OR Result
0	0	0
0	1	1
1	1	0
1	0	1

Processor Registers Affected CC

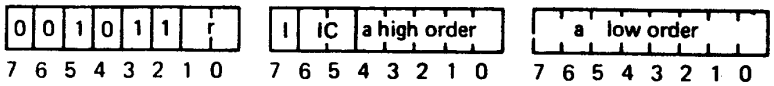
Condition Code Setting

Register r	CC1	CC0
Positive	0	1
Zero	0	0
Negative	1	0

EXCLUSIVE OR ABSOLUTE (Absolute Addressing)

Mnemonic EORA,r (•)a(,X)

Binary Code



Execution Time 4 cycles (12 clock periods)

Description

This three-byte instruction causes the contents of register r to be Exclusive ORed with the contents of the memory byte pointed to by the effective address. The result of the operation replaces the previous contents of register r.

Indirect addressing and/or indexing may be specified. If indexing is specified, bits 1 and 0, byte 0, indicate the index register and the destination of the operation implicitly becomes register zero.

The Exclusive OR operation treats each bit of the argument bytes as in the truth table below:

Bit (0-7)	Bit (0-7)	Exclusive OR Result
0	0	0
0	1	1
1	1	0
1	0	1

Processor Registers Affected CC

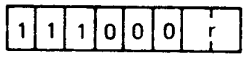
Condition Code Setting

Register r	CC1	CC0
Positive	0	1
Zero	0	0
Negative	1	0

COMPARE TO REGISTER ZERO (Register Addressing)

Mnemonic COMZ r

Binary Code



Execution Time 2 cycles (6 clock periods)

Description

This one-byte instruction causes the contents of the specified register r to be compared to the contents of register zero. The comparison will be performed in either "arithmetic" or "logical" mode depending on the setting of the COM bit in the Program Status Word.

When COM=1 (logical mode) the values will be interpreted as 8-bit positive binary numbers; when COM=0, the values will be interpreted as 8-bit two's complement numbers.

The execution of this instruction *only* causes the Condition Code to be set as in the following table.

Processor Registers Affected CC

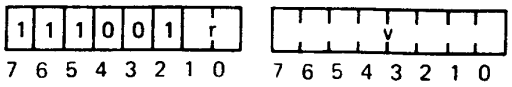
Condition Code Setting

	CC1	CC0
Register zero greater than Register r	0	1
Register zero equal to Register r	0	0
Register zero less than Register r	1	0

COMPARE IMMEDIATE (Immediate Addressing)

Mnemonic COMI,r v

Binary Code



Execution Time 2 cycles (6 clock periods)

Description

This two-byte instruction causes the contents of the specified register r to be compared to the contents of the second byte of this instruction. The comparison will be performed in either the "arithmetic" or "logical" mode depending on the setting of the COM bit in the Program Status Word.

When COM=1 (logical mode), the values will be treated as 8-bit positive binary numbers; when COM=0, the values will be treated as 8-bit two's complement numbers.

The execution of this instruction *only* causes the Condition Code to be set as in the following table.

Processor Registers Affected CC

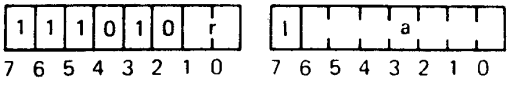
Condition Code Setting

	CC1	CC0
Register r greater than v	0	1
Register r equal to v	0	0
Register r less than v	1	0

COMPARE RELATIVE (Relative Addressing)

Mnemonic COMR,r (*)a

Binary Code



Execution Time 3 cycles (9 clock periods)

Description

This two-byte instruction causes the contents of the specified register r to be compared to the contents of the memory byte pointed to by the effective address. The comparison will be performed in either the "arithmetic" or "logical" mode depending upon the setting of the COM bit in the Program Status Word.

When COM=1 (logical mode), the values will be treated as 8-bit positive binary numbers; when COM=0, the values will be treated as 8-bit, two's complement numbers.

The execution of this instruction *only* causes the Condition Code to be set as in the following table.

Processor Registers Affected CC

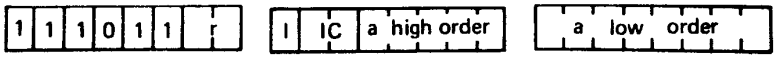
Condition Code Setting

	CC1	CC0
Register r greater than memory byte	0	1
Register r equal to memory byte	0	0
Register r less than memory byte	1	0

COMPARE ABSOLUTE (Absolute Addressing)

Mnemonic COMA,r (*)a(,X)

Binary Code



Execution Time 4 cycles (12 clock periods)

Description

This three-byte instruction causes the contents of register r to be compared to the contents of the memory byte pointed to by the effective address. The comparison will be performed in either the "arithmetic" or "logical" mode depending on the setting of the COM bit in the Program Status Word.

Where COM=1 (logical mode), the values will be treated as 8-bit, positive binary numbers; when COM=0 (arithmetic mode), the values will be treated as 8-bit, two's complement numbers.

Indirect addressing and/or indexing may be specified. If indexing is specified, bits 1 and 0, byte 0, indicate the index register and the destination of the operation implicitly becomes register zero.

The execution of this instruction *only* causes the Condition Code to be set as in the following table.

Processor Registers Affected CC

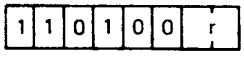
Condition Code Setting

	CC1	CC0
Register r greater than memory byte	0	1
Register r equal to memory byte	0	0
Register r less than memory byte	1	0

ROTATE REGISTER LEFT (Register Addressing)

Mnemonic RRL,r

Binary Code

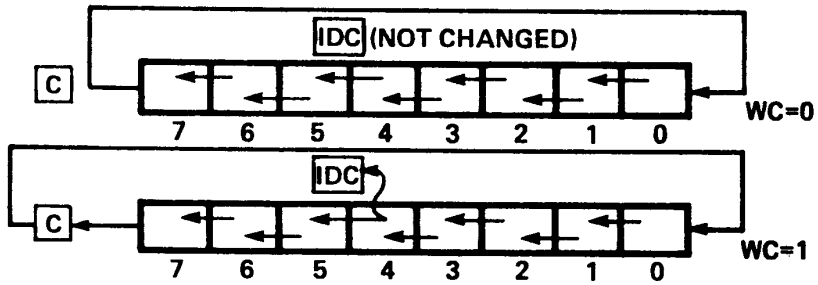


Execution Time 2 cycles (6 clock periods)

Description

This one-byte instruction causes the contents of the specified register r to be shifted left one bit. If the WC bit in the Program Status Word is set to zero, bit #7 of register r flows into bit #0; if WC=1, then bit #7 flows into the Carry bit and the Carry bit flows into bit #0.

Register bit #4 flows into the IDC if WC=1.



Note: Whenever a rotate causes bit #7 of the specified register to change polarity, the OVF bit is set in the PSL.

Processor Registers Affected C, CC, IDC, OVF

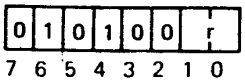
Condition Code Setting

Register r	CC1	CC0
Positive	0	1
Zero	0	0
Negative	1	0

ROTATE REGISTER RIGHT (Register Addressing)

Mnemonic RRR,r

Binary Code

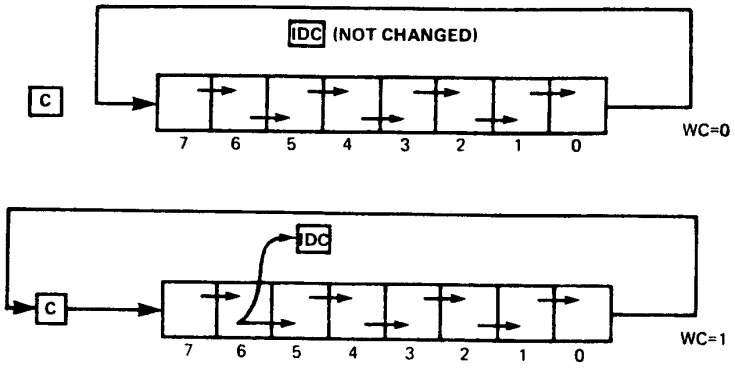


Execution Time 2 cycles (6 clock periods)

Description

This one-byte instruction causes the contents of the specified register r to be shifted right one bit. If the WC bit in the Program Status Word is set to zero, bit #0 of the register r flows into bit #7; if WC=1, then bit #0 of the register r flows into the Carry bit and the Carry bit flows into bit #7.

Register bit #6 flows into the IDC if WC=1.



Note: Whenever a rotate causes bit #7 of the specified register to change polarity, the OVF bit is set in the PSL.

Processor Registers Affected C, CC, IDC, OVF

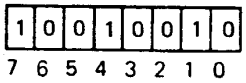
Condition Code Setting

Register r	CC1	CC0
Positive	0	1
Zero	0	0
Negative	1	0

LOAD PROGRAM STATUS, UPPER

Mnemonic LPSU

Binary Code



Execution Time 2 cycles (6 clock periods)

Description

This one-byte instruction causes the current contents of the Upper Program Status Byte to be replaced with the contents of register zero.

See Program Status Word description for bit assignments. Bits #4 and #3 of the PSU are unassigned and will always be regarded as containing zeroes.

Processor Registers Affected F, II, SP

Condition Code Setting N/A

LOAD PROGRAM STATUS, LOWER

Mnemonic LPSL

Binary Code

1	0	0	1	0	0	1	1
7	6	5	4	3	2	1	0

Execution Time 2 cycles (6 clock periods)

Description

This one-byte instruction causes the current contents of the Lower Program Status Byte to be replaced with the contents of register zero.

See Program Status Word description for bit assignments.

Processor Registers Affected CC, IDC, RS, WC, OVF, COM, C

Condition Code Setting

The CC will take on the value in bits #7 and #6 of register zero.

STORE PROGRAM STATUS, UPPER

Mnemonic SPSU

Binary Code

0	0	0	1	0	0	1	0
7	6	5	4	3	2	1	0

Execution Time 2 cycles (6 clock periods)

Description

This one-byte instruction causes the contents of the Upper Program Status Byte to be transferred into register zero.

See Program Status Word description for bit assignments. Bits #4 and #3 which are unassigned will always be stored as zeroes.

Processor Registers Affected CC

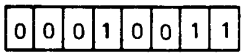
Condition Code Setting

Register Zero	CC1	CC0
Positive	0	1
Zero	0	0
Negative	1	0

STORE PROGRAM STATUS, LOWER

Mnemonic SPSL

Binary Code



Execution Time 2 cycles (6 clock periods)

Description

This one-byte instruction causes the contents of the Lower Program Status Byte to be transferred into register zero.

See Program Status Word description for bit assignments.

Processor Registers Affected CC

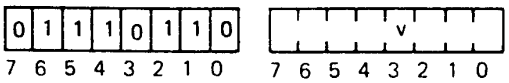
Condition Code Setting

Register Zero	CC1	CC0
Positive	0	1
Zero	0	0
Negative	1	0

PRESET PROGRAM STATUS UPPER, SELECTIVE (Immediate Addressing)

Mnemonic PPSU v

Binary Code



Execution Time 3 cycles (9 clock periods)

Description

This two-byte instruction causes individual bits in the Upper Program Status Byte to be selectively set to binary one. When this instruction is executed, each bit in the v field of the second byte of this instruction is tested for the presence of a one and if a particular bit in the v field contains a one, the corresponding bit in the status byte is set to binary one. Any bits in the status byte which are not selected are not modified.

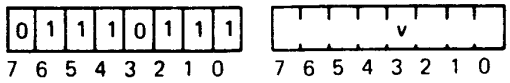
Processor Registers Affected F, II, SP

Condition Code Setting N/A

PRESET PROGRAM STATUS LOWER, SELECTIVE (Immediate Addressing)

Mnemonic PPSL v

Binary Code



Execution Time 3 cycles (9 clock periods)

Description

This two-byte instruction causes individual bits in the Lower Program Status Byte to be selectively set to binary one. When this instruction is executed, each bit in the v field of the second byte of this instruction is tested for the presence of a one and if a particular bit in the v field contains a one, the corresponding bit in the status byte is set to binary one. Any bits in the status byte which are not selected are not modified.

Processor Registers Affected CC, IDC, RS, WC, OVF, COM, C

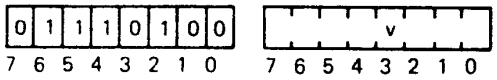
Condition Code Setting

The CC bits may be set by the execution of this instruction.

CLEAR PROGRAM STATUS UPPER, SELECTIVE (Immediate Addressing)

Mnemonic CPSU v

Binary Code



Execution Time 3 cycles (9 clock periods)

Description

This two-byte instruction causes individual bits in the Upper Program Status Byte to be selectively cleared. When this instruction is executed, each bit in the v field of the second byte of this instruction is tested for the presence of a one and if a particular bit in the v field contains a one, the corresponding bit in the status byte is cleared to zero. Any bits in the status byte which are not selected are not modified.

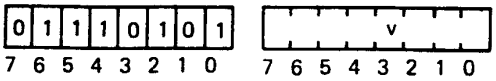
Processor Registers Affected F, II, SP

Condition Code Setting N/A

CLEAR PROGRAM STATUS LOWER, SELECTIVE (Immediate Addressing)

Mnemonic CPSL v

Binary Code



Execution Time 3 cycles (9 clock periods)

Description

This two-byte instruction causes individual bits in the Lower Program Status Byte to be selectively cleared. When this instruction is executed, each bit in the v field of the second byte of this instruction is tested for the presence of a one and if a particular bit in the v field contains a one, the corresponding bit in the status byte is cleared to zero. Any bits in the status byte which are not selected are not modified.

Processor Registers Affected CC, IDC, RS, WC, OVF, COM, C

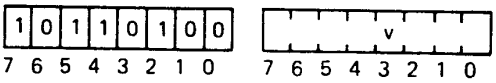
Condition Code Setting

The CC bits may be cleared by the execution of this instruction.

TEST PROGRAM STATUS UPPER, SELECTIVE (Immediate Addressing)

Mnemonic TPSU v

Binary Code



Execution Time 3 cycles (9 clock periods)

Description

This two-byte instruction tests individual bits in the Upper Program Status Byte to determine if they are set to binary one. When this instruction is executed, each bit in the v field of this instruction is tested for the presence of a one, and if a particular bit in the v field contains a one, the corresponding bit in the status byte is tested for a one or zero. The Condition Code is set to reflect the result of this operation.

If a bit in the v field is zero, the corresponding bit in the status byte is not tested.

Processor Registers Affected CC

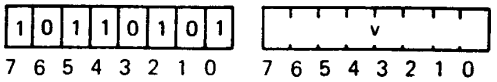
Condition Code Setting

	CC1	CC0
All of the selected bits in PSU are 1s	0	0
Not all of the selected bits in PSU are 1s	1	0

TEST PROGRAM STATUS LOWER, SELECTIVE (Immediate Addressing)

Mnemonic TPSL v

Binary Code



Execution Time 3 cycles (9 clock periods)

Description

This two-byte instruction tests individual bits in the Lower Program Status Byte to determine if they are set to binary one. When this instruction is executed, each bit in the v field of this instruction is tested for a one, and if a particular bit in the v field contains a one, the corresponding bit in the status byte is tested for a one or zero. The Condition Code is set to reflect the result of this operation.

Processor Registers Affected CC

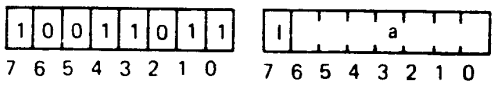
Condition Code Setting

	CC1	CC0
All of the selected bits in PSL are 1s	0	0
Not all of the selected bits in PSL are 1s	1	0

ZERO BRANCH RELATIVE (Relative Addressing)

Mnemonic ZBRR (*)a

Binary Code



Execution Time 3 cycles (9 clock periods)

Description

This two-byte unconditional relative branch instruction directs the processor to calculate the effective address differently than the usual calculation for the Relative Addressing mode.

The specified value, a, is interpreted as a relative displacement from page zero, byte zero. Therefore, displacement may be specified from -64 to +63 bytes. The address calculation is modulo 8192_{10}, so the negative displacement actually will develop addresses at the end of page zero. For example, ZBRR -8, will develop an effective address of 8184_{10}, and a ZBRR +52 will develop an effective address of 52_{10}.

This instruction causes the processor to clear, address bits 13 and 14, the page address bits; and to replace the contents of the Instruction Address Register with the effective address of the instruction. This instruction may be executed anywhere within addressable memory.

Indirect addressing may be specified.

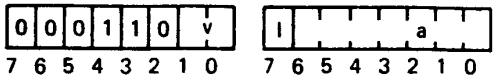
Processor Registers Affected None

Condition Code Setting N/A

BRANCH ON CONDITION TRUE, RELATIVE (Relative Addressing)

Mnemonic BCTR,v (*)a

Binary Code



Execution Time 3 cycles (9 clock periods)

Description

This two-byte conditional branch instruction causes the processor to fetch the next instruction to be executed from the memory location pointed to by the effective address only if the two-bit v field matches the current Condition Code field (CC) in the Program Status Word.

If the v field and CC field do not match, the next instruction is fetched from the location following the second byte of this instruction.

Indirect addressing may be specified.

If the v field is set to 3_{16}, an unconditional branch is effected.

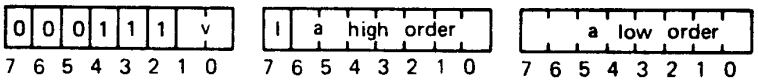
Processor Registers Affected None

Condition Code Setting N/A

BRANCH ON CONDITION TRUE, ABSOLUTE (Absolute Addressing)

Mnemonic BCTA,v (*)a

Binary Code



Execution Time 3 cycles (9 clock periods)

Description

This three-byte conditional branch instruction causes the processor to fetch the next instruction to be executed from the memory location pointed to by the effective address only if the two-bit v field matches the two-bit Condition Code field (CC) in the Program Status Word.

If the v field and CC field do not match, the next instruction is fetched from the location following the second byte of this instruction.

Indirect addressing may be specified.

If the v field is set to 3_{16}, an unconditional branch is effected.

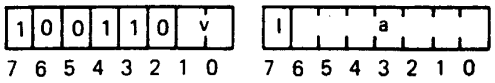
Processor Registers Affected None

Condition Code Setting N/A

BRANCH ON CONDITION FALSE, RELATIVE (Relative Addressing)

Mnemonic BCFR,v (*)a

Binary Code



Execution Time 3 cycles (9 clock periods)

Description

This two-byte branch instruction causes the processor to fetch the next instruction to be executed from the memory location pointed to by the effective address only if the two-bit v field does not match the two-bit Condition Code field (CC) in the Program Status Word. If there is no match, the contents of the Instruction Address Register are replaced by the effective address.

If the v field and CC field match, the next instruction is fetched from the location following the second byte of this instruction.

Indirect addressing may be specified.

The v field may not be set to 3_{16} as this bit combination is used for the ZBRR operation code.

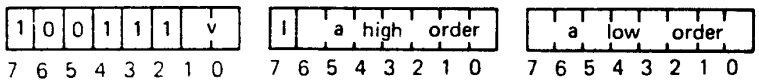
Processor Registers Affected None

Condition Code Setting N/A

BRANCH ON CONDITION FALSE, ABSOLUTE (Absolute Addressing)

Mnemonic BCFA,v (*)a

Binary Code



Execution Time 3 cycles (9 clock periods)

Description

This three-byte instruction causes the processor to fetch the next instruction to be executed from the memory location pointed to by the effective address only if the two-bit v field does not match the two-bit Condition Code field (CC) in the Program Status Word. If there is no match, the contents of the Instruction Address Register are replaced by the effective address.

If the v field and CC field match, the next instruction is fetched from the location following the second byte of this instruction.

Indirect addressing may be specified.

The v field may not be set to 3_{16} as this bit combination is used for the BXA operation code.

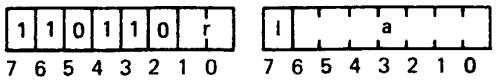
Processor Registers Affected None

Condition Code Setting N/A

BRANCH ON INCREMENTING REGISTER, RELATIVE (Relative Addressing)

Mnemonic BIRR,r (*)a

Binary Code



Execution Time 3 cycles (9 clock periods)

Description

This two-byte branch instruction causes the processor to increment the contents of the specified register by one. If the new value in the register is non-zero, the next instruction to be executed is taken from the memory location pointed to by the effective address, i.e., the effective address replaces the previous contents of the Instruction Address Register. If the new value in register r is zero, the next instruction to be executed follows the second byte of this instruction.

Indirect addressing may be specified.

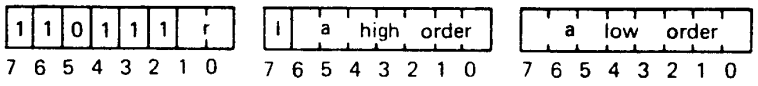
Processor Registers Affected None

Condition Code Setting N/A

BRANCH ON INCREMENTING REGISTER, ABSOLUTE (Absolute Addressing)

Mnemonic BIRA,r (*)a

Binary Code



Execution Time 3 cycles (9 clock periods)

Description

This three-byte branch instruction causes the processor to increment the contents of the specified register by one. If the new value in the register is non-zero, the next instruction to be executed is taken from the memory location pointed to by the effective address, i.e., the effective address replaces the previous contents of the Instruction Address Register. If the new value of register r is zero, the next instruction to be executed follows the second byte of this instruction.

Indirect addressing may be specified.

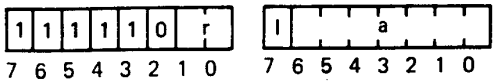
Processor Registers Affected None

Condition Code Setting N/A

BRANCH ON DECREMENTING REGISTER,RELATIVE (Relative Addressing)

Mnemonic BDRR,r (*)a

Binary Code



Execution Time 3 cycles (9 clock periods)

Description

This two-byte branch instruction causes the processor to decrement the contents of the specified register by one. If the new value in the register is non-zero, the next instruction to be executed is taken from the memory location pointed to by the effective address, i.e., the effective address replaces the previous contents of the Instruction Address Register. If the new value in register r is zero, the next instruction to be executed follows the second byte of this instruction.

Indirect addressing may be specified.

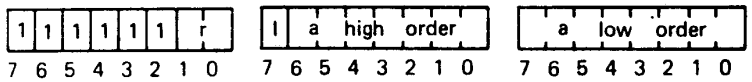
Processor Registers Affected None

Condition Code Setting N/A

BRANCH ON DECREMENTING REGISTER,ABSOLUTE (Absolute Addressing)

Mnemonic BDRA,r (*)a

Binary Code



Execution Time 3 cycles (9 clock periods)

Description

This three-byte instruction causes the processor to decrement the contents of the specified register by one. If the new value in the register is non-zero, the next instruction to be executed is taken from the memory location pointed to by the effective address, i.e., the effective address replaces the previous contents of the Instruction Address Register. If the new address in register r is zero, the next instruction to be executed follows the second byte of this instruction.

Indirect addressing may be specified.

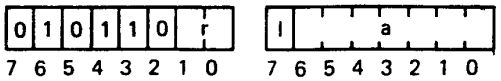
Processor Registers Affected None

Condition Code Setting N/A

BRANCH ON REGISTER NON-ZERO, RELATIVE (Relative Addressing)

Mnemonic BRNR,r (*)a

Binary Code



Execution Time 3 cycles (9 clock periods)

Description

This two-byte branch instruction causes the contents of the specified register r to be tested for a non-zero value. If the register contains a non-zero value, the next instruction to be executed is taken from the location pointed to by the effective address, i.e., the effective address replaces the current contents of the Instruction Address Register.

If the specified register contains a zero value, the next instruction is fetched from the location following the second byte of this instruction.

Indirect addressing may be specified.

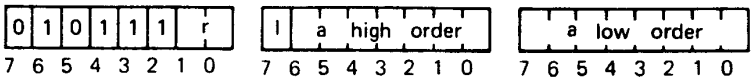
Processor Registers Affected None

Condition Code Setting N/A

BRANCH ON REGISTER NON-ZERO, ABSOLUTE (Absolute Addressing)

Mnemonic BRNA,r (*)a

Binary Code



Execution Time 3 cycles (9 clock periods)

Description

The three-byte branch instruction causes the contents of the specified register r to be tested for a non-zero value. If the register contains a non-zero value, the next instruction to be executed is taken from the location pointed to by the effective address, i.e., the effective address replaces the contents of the Instruction Address Register.

If the specified register contains a zero value, the next instruction is fetched from the location following the third byte of this instruction.

Indirect addressing may be specified.

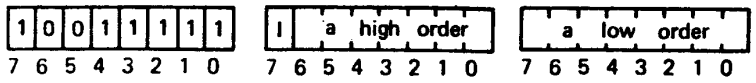
Processor Registers Affected None

Condition Code Setting N/A

BRANCH INDEXED, ABSOLUTE (Absolute Addressing)

Mnemonic BXA (*)a,X

Binary Code



Execution Time 3 cycles (9 clock periods)

Description

This three-byte branch instruction causes the processor to perform an unconditional branch. Indexing is required and register #3 must be specified as the index register because the entire first byte of this instruction is decoded by the processor. When executed, the content of the Instruction Address Register (IAR) is replaced by the effective address.

If indirect addressing is specified, the value in the index register is added to the indirect address to calculate the effective branch address.

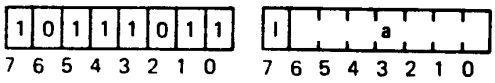
Processor Registers Affected None

Condition Code Setting N/A

ZERO BRANCH TO SUBROUTINE, RELATIVE (Relative Addressing)

Mnemonic ZBSR (*)a

Binary Code



Execution Time 3 cycles (9 clock periods)

Description

This two-byte unconditional subroutine branch instruction directs the processor to calculate the effective address differently than the usual calculation for the Relative Addressing mode.

The specified value a is interpreted as a relative displacement from page zero, byte zero. Therefore, displacement may be specified from -64 to +63 bytes. The address calculation is modulo 8192_{10}, so the negative displacement will develop addresses at the end of page zero. For example, ZBSR -10, will develop an effective address of 8182_{10}, and ZBSR 31 will develop an effective address of 31_{10}.

This instruction causes the processor to clear the page address bits, address bits 14 and 13, and may be executed anywhere within addressable memory.

Indirect addressing may be specified.

When executed, this instruction causes the Stack Pointer to be incremented by one, the address of the byte following this instruction is pushed into the Return Address Stack (RAS), and control is transferred to the effective address.

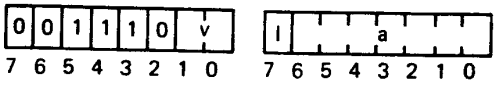
Processor Registers Affected SP

Condition Code Setting N/A

BRANCH TO SUBROUTINE ON CONDITION TRUE, RELATIVE (Relative Addressing)

Mnemonic BSTR,v (*)a

Binary Code



Execution Time 3 cycles (9 clock periods)

Description

This two-byte conditional subroutine branch instruction causes the processor to perform a subroutine branch *only* if the two-bit v field matches the current Condition Code field (CC) in the Program Status Word. If the fields match, the Stack Pointer is incremented by one and the current contents of the Instruction Address Register, which points to the byte following this instruction, is pushed into the Return Address Stack. The effective address replaces the previous contents of the IAR.

If the v field and CC field do not match, the next instruction is fetched from the location following the second byte of this instruction and the SP is unaffected.

Indirect addressing may be specified.

If v is set to 3_{16}, the BSTR instruction branches unconditionally.

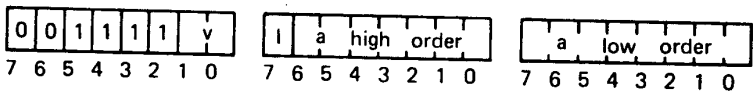
Processor Registers Affected SP

Condition Code Setting N/A

BRANCH TO SUBROUTINE ON CONDITION TRUE,ABSOLUTE (Absolute Addressing)

Mnemonic BSTA,v (*)a

Binary Code



Execution Time 3 cycles (9 clock periods)

Description

This three-byte conditional subroutine branch instruction causes the processor to perform a subroutine branch only if the two-bit v field matches the current Condition Code Field (CC) in the Program Status Word. If the fields match, the Stack Pointer is incremented by one and the current contents of the Instruction Address Register, which points to the byte following this instruction is pushed into the Return Address Stack. The effective address replaces the previous contents of the IAR.

If the v field and the CC field do not match, the next instruction is fetched from the location following the third byte of this instruction and the Stack Pointer is unaffected.

Indirect addressing may be specified.

If v is set to 3_{16}, the BSTA instruction branches unconditionally.

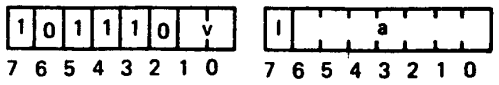
Processor Registers Affected SP

Condition Code Setting N/A

BRANCH TO SUBROUTINE ON CONDITION FALSE, RELATIVE (Relative Addressing)

Mnemonic BSFR,v (•)a

Binary Code



Execution Time 3 cycles (9 clock periods)

Description

This two-byte conditional subroutine branch instruction causes the processor to perform a subroutine branch *only* if the two-bit v field does *not* match the current Condition Code field (CC) in the Program Status Word. If the fields do not match, the Stack Pointer is incremented by one and the current content of the Instruction Address Register, which points to the location following this instruction, is pushed into the Return Address Stack. The effective address replaces the previous contents of the IAR.

If the v field and the CC match, the next instruction is fetched from the location following this instruction and the SP is unaffected.

Indirect addressing may be specified.

The v field may not be coded as 3_{16} because this combination is used for the ZBSR operation code.

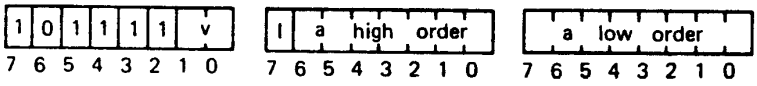
Processor Registers Affected SP

Condition Code Setting N/A

BRANCH TO SUBROUTINE ON CONDITIONFALSE, ABSOLUTE (Absolute Addressing)

Mnemonic BSFA,v (•)a

Binary Code



Execution Time 3 cycles (9 clock periods)

Description

This three-byte conditional subroutine branch instruction causes the processor to perform a subroutine branch *only* if the two-bit v field does *not* match the current Condition Code (CC) in the Program Status Word. If the fields do not match, the Stack Pointer is incremented by one and the current content of the Instruction Address Register, which points to the location following this instruction, is pushed into the Return Address Stack. The effective address replaces the previous contents of the IAR.

If the v field and the CC match, the next instruction is fetched from the location following this instruction and the SP is unaffected.

Indirect addressing may be specified.

The v field may not be coded as 3_{16} as this combination is used for the BSXA operation code.

Processor Registers Affected SP

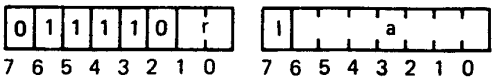
Condition Code Setting N/A

BRANCH TO SUBROUTINE ON NON-ZERO REGISTER, RELATIVE

(Relative Addressing)

Mnemonic BSNR,r (*)a

Binary Code



Execution Time 3 cycles (9 clock periods)

Description

This two-byte subroutine branch instruction causes the contents of the specified register r to be tested for a non-zero value. If the register contains a non-zero value, the next instruction to be executed is taken from the location pointed to by the effective address. Before replacing the contents of the Instruction Address Register with the effective address, the Stack Pointer (SP) is incremented by one and the address of the byte following the instruction is pushed into the Return Address Stack (RAS).

If the specified register contains a zero value, the next instruction is fetched from the location following this instruction.

Indirect addressing may be specified.

Processor Registers Affected SP

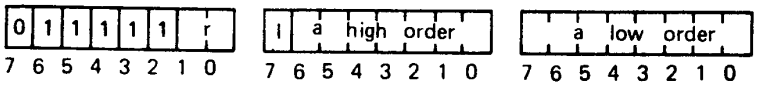
Condition Code Setting N/A

BRANCH TO SUBROUTINE ON NON-ZERO REGISTER, ABSOLUTE

(Absolute Addressing)

Mnemonic BSNA,r (*)a

Binary Code



Execution Time 3 cycles (9 clock periods)

Description

This three-byte subroutine branch instruction causes the contents of the specified register r to be tested for a non-zero value. If the register contains a non-zero value, the next instruction to be executed is taken from the location pointed to by the effective address. Before replacing the current contents of the Instruction Address Register (IAR) with the effective address, the Stack Pointer (SP) is incremented by one and the address of the byte following the instruction is pushed into the Return Address Stack (RAS).

If the specified register contains a zero value, the next instruction is fetched from the location following this instruction.

Indirect addressing may be specified.

Processor Registers Affected SP

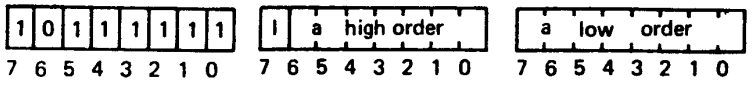
Condition Code Setting N/A

BRANCH TO SUBROUTINE INDEXED, ABSOLUTE, UNCONDITIONAL

(Absolute Addressing)

Mnemonic BSXA (•)a,X

Binary Code



Execution Time 3 cycles (9 clock periods)

Description

This three-byte instruction causes the processor to perform an unconditional subroutine branch. Indexing is required and register #3 must be specified as the index register because the entire first byte of this instruction is decoded by the processor.

Execution of this instruction causes the Stack Pointer (SP) to be incremented by one, the address of the byte following this instruction is pushed into the Return Address Stack (RAS), and the effective address replaces the contents of the Instruction Address Register.

If indirect addressing is specified, the value in the index register is added to the indirect address to calculate the effective address.

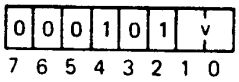
Processor Registers Affected SP

Condition Code Setting N/A

RETURN FROM SUBROUTINE, CONDITIONAL

Mnemonic RETC,v

Binary Code



Execution Time 3 cycles (9 clock periods)

Description

This one-byte instruction is used by a subroutine to conditionally effect a return of control to the program which last issued a subroutine branch instruction.

If the two-bit v field in the instruction matches the Condition Code field (CC) in the Program Status Word, the following action is taken: The address contained in the top of the Return Address Stack replaces the previous contents of the Instruction Address Register (IAR), and the Stack Pointer is decremented by one.

If the v field does not match CC, the return is not effected and the next instruction to be executed is taken from the location following this instruction.

If v is specified as 3_{16}, the return is executed unconditionally.

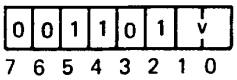
Processor Registers Affected SP

Condition Code Setting N/A

RETURN FROM SUBROUTINE AND ENABLE INTERRUPT,CONDITIONAL

Mnemonic RETE,v

Binary Code



Execution Time 3 cycles (9 clock periods)

Description

This one-byte instruction is used by a subroutine to conditionally effect a return of control to the program which last issued a subroutine branch instruction. Additionally, if the return is effected, the Interrupt Inhibit (II) bit in the Program Status Word is cleared to zero, thus enabling interrupts. This instruction is mainly intended to be used by an interrupt handling routine because receipt of an interrupt causes a subroutine branch to be effected and the Interrupt Inhibit bit to be set to 1. The interrupt handling routine must be able to return and enable simultaneously so that the interrupt routine cannot be interrupt unless that is specifically desired.

If the two-bit v field in the instruction matches the Condition Code field (CC) in the Program Status Word, the following action is taken: The address contained in the top of the Return Address Stack (RAS) replaces the previous contents of the Instruction Address Register (IAR), the Stack Pointer is decremented by one and the II bit is cleared to zero.

If the v field does not match CC, the return is not effected and the next instruction to be executed is taken from the location following this instruction.

If v is specified as 3_{16}, the return is executed unconditionally.

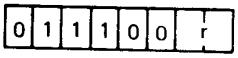
Processor Registers Affected SP , II

Condition Code Setting N/A

READ DATA (Register Addressing)

Mnemonic REDD,r

Binary Code



Execution Time 2 cycles (6 clock periods)

Description

This one-byte input instruction causes a byte of data to be transferred from the data bus into register r. Signals on the data bus are considered to be true signals, i.e., a high level will be set into the register as a one.

When executing this instruction, the processor raises the Operation Request (OPREQ) line, simultaneously switching the M/$\overline{IO}$ line to $\overline{IO}$ and the $\overline{R}$/W to $\overline{R}$ (Read). Also, during the OPREQ signal, the D/$\overline{C}$ line switches to D (Data) and the E/$\overline{NE}$ switches to $\overline{NE}$ (Non-extended).

See Input/Output section of this manual.

Processor Registers Affected CC

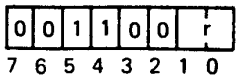
Condition Code Setting

Register r	CC1	CC0
Positive	0	1
Zero	0	0
Negative	1	0

READ CONTROL (Register Addressing)

Mnemonic REDC,r

Binary Code



Execution Time 2 cycles (6 clock periods)

Description

This one-byte input instruction causes a byte of data to be transferred from the data bus into register r. Signals on the data bus are considered to be true signals, i.e., a high level will be set into the register as a one.

When executing this instruction, the processor raises the Operation Request (OPREQ) line, simultaneously switching the M/$\overline{IO}$ line to $\overline{IO}$, the $\overline{R}$/W line to $\overline{R}$ (Read), the D/$\overline{C}$ line to $\overline{C}$ (Control), and the E/$\overline{NE}$ line to $\overline{NE}$ (Non-extended).

See Input/Output section of this manual.

Processor Registers Affected CC

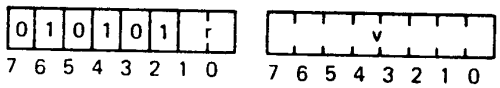
Condition Code Setting

Register r	CC1	CC0
Positive	0	1
Zero	0	0
Negative	1	0

READ EXTENDED (Immediate Addressing)

Mnemonic REDE,r v

Binary Code



Execution Time 3 cycles (9 clock periods)

Description

This two-byte input instruction causes a byte of data to be transferred from the data bus into register r. During the execution of this instruction, the content of the second byte of this instruction is made available on the address bus. Signals on the data bus are true signals, i.e., a high level is interpreted as a one.

During execution, the processor raises the Operation Request (OPREQ) line, simultaneously placing the contents of the second byte of the instruction on the address bus. During the OPREQ signal, the M/$\overline{IO}$ line is switched to $\overline{IO}$, the $\overline{R}$/W line to $\overline{R}$ (Read), line and the E/$\overline{NE}$ line to E (Extended).

See Input/Output section of this manual.

Processor Registers Affected CC

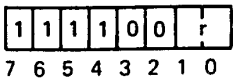
Condition Code Setting

Register r	CC1	CC0
Positive	0	1
Zero	0	0
Negative	1	0

WRITE DATA (Register Addressing)

Mnemonic WRTD,r

Binary Code



Execution Time 2 cycles (6 clock periods)

Description

This one-byte output instruction causes a byte of data to be made available to an external device. The byte to be output is taken from register r and made available on the data bus. Signals on the data bus are true signals, i.e., high levels are ones.

When executing this instruction, the processor raises the Operation Request (OPREQ) line and simultaneously places the data on the Data Bus. Along with the OPREQ, the M/$\overline{IO}$ line is switched to $\overline{IO}$, the $\overline{R}$/W signal is switched to W (Write), and a Write Pulse (WRP) is generated. Also, during the valid OPREQ signals, the D/$\overline{C}$ line is switched to D (Data) and the E/$\overline{NE}$ line is switched to $\overline{NE}$ (Non-extended).

See Input/Output section of this manual.

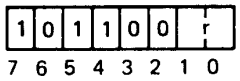
Processor Registers Affected None

Condition Code Setting N/A

WRITE CONTROL (Register Addressing)

Mnemonic WRTC,r

Binary Code



Execution Time 2 cycles (6 clock periods)

Description

This one-byte output instruction causes a byte of data to be made available to an external device.

The byte to be output is taken from register r and made available on the data bus. Signals on the data bus are true signals, i.e., high levels are ones

When executing this instruction, the processor raises the Operation Request (OPREQ) line and simultaneously places the data on the Data Bus. Along with the OPREQ signal, the M/$\overline{IO}$ line is switched to $\overline{IO}$, the $\overline{R}$/W signal is switched to W (Write), the D/$\overline{C}$ line is switched to $\overline{C}$ (Control), the E/$\overline{NE}$ is switched to $\overline{NE}$ (Non-extended), and a Write Pulse (WRP) is generated.

See the Input/Output section of this manual.

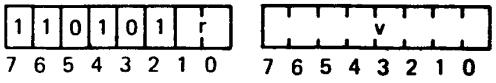
Processor Registers Affected None

Condition Code Setting N/A

WRITE EXTENDED (Immediate Addressing)

Mnemonic WRTE,r v

Binary Code



Execution Time 3 cycles (9 clock periods)

Description

This two-byte output instruction causes a byte of data to be made available to an external device. The byte to be output is taken from register r and is made available on the data bus. Simultaneously, the data in the second byte of this instruction is made available on the address bus. The second byte, v, may be interpreted as a device address.

Signals on the busses are true levels, i.e., high levels are ones.

When executing this instruction, the processor raises the Operation Request (OPREQ) line and simultaneously places the data from register r on the data bus and the data from the second byte of this instruction on the address bus. Along with OPREQ, the M/$\overline{IO}$ line is switched to $\overline{IO}$, the $\overline{R}$/W line is switched to W (Write), the E/$\overline{NE}$ line is switched to E (Extended), and a Write Pulse (WRP) is generated.

See the Input/Output section of this manual.

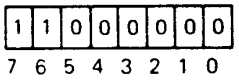
Processor Registers Affected None

Condition Code Setting N/A

NO OPERATION

Mnemonic NOP

Binary Code



Execution Time 2 cycles (6 clock periods)

Description

This one-byte instruction causes the processor to take no action upon decoding it. No registers are changed, but fetching and executing a NOP instruction requires two processor cycles.

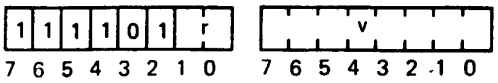
Processor Registers Affected None

Condition Code Setting N/A

TEST UNDER MASK IMMEDIATE (Immediate Addressing)

Mnemonic TMI,r v

Binary Code



Execution Time 3 cycles (9 clock periods)

Description

This two-byte instruction tests individual bits in the specified register r to determine if they are set to binary one. During execution, each bit in the v field of the instruction is tested for a one, and if a particular bit in the v field contains a one, the corresponding bit in register r is tested for a one or zero. The condition code is set to reflect the result of the operation.

If a bit in the v field is zero, the corresponding bit in register r is not tested.

Processor Registers Affected CC

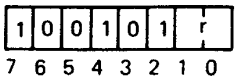
Condition Code Setting

	CC1	CC0
All of the selected bits are 1s	0	0
Not all of the selected bits are 1s	1	0

DECIMAL ADJUST REGISTER (Register Addressing)

Mnemonic DAR,r

Binary Code



Execution Time 3 cycles (9 clock periods)

Description

This one-byte instruction conditionally adds a decimal ten (two's complement negative six in a four-bit binary number system) to either the high order 4 bits and/or the low order 4 bits of the specified register r.

The truth table below indicates the logical operation performed. The operation proceeds based on the contents of the Carry (C) and Interdigit Carry (IDC) bits in the Program Status Word. The C and IDC remain unchanged by the execution of this instruction.

This instruction allows BCD sign magnitude arithmetic to be performed on packed digits by the following procedure.

BCD Addition:
1. add 66_{16} to augend
2. perform addition of addend and augend
3. perform DAR instruction

BCD Subtraction:
1. perform subtraction (2's complement of subtrahend is added to the minuend)
2. perform DAR instruction

Since this operation is on sign-magnitude numbers, it is necessary to establish the sign of the result prior to executing in order to properly control the definition of the subtrahend and minuend.

Carry	Interdigit Carry	Added to Register r
0	0	AA_{16}
0	1	$A0_{16}$
1	1	00_{16}
1	0	$0A_{16}$

Processor Registers Affected CC

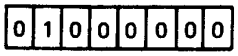
Condition Code Setting

The Condition Code is set to a meaningless value.

HALT, ENTER WAIT STATE

Mnemonic HALT

Binary Code



Execution Time 2 cycles (6 clock periods)

Description

This one-byte instruction causes the processor to stop executing instructions and enter the WAIT state. The RUN/$\overline{\text{WAIT}}$ line is set to the WAIT state.

The only way to enter the RUN state after a HALT has been executed, is to reset the 2650 or to interrupt the processor.

Processor Registers Affected None

Condition Code Setting N/A

APPENDIXES

APPENDIX A

MEMORY INTERFACE

Figure 22 shows a complete interface between the 2650 and a 256 x 8 R/W random access memory. Since the memory chips are MOS they can be driven directly by the address lines and the control lines. The gates shown are assumed to be standard 7400 series TTL so that some signal buffering is assumed to be necessary. If CMOS or 74LS gates are used, some of the buffering inverters may not be necessary. The same is true of the data bus. Depending on the number and nature of the I/O devices being interfaced, it may or may not be necessary to buffer the data bus.

Because the data in and data out signals for the memory chips are bussed together, care must be taken to avoid overlap of drivers on the data bus. In this example, the problem is solved by using the write pulse into the memory as the chip select input instead of using the $\overline{R}/W$ line as is conventionally done. The $\overline{R}/W$ output from the processor is a level and is valid when Operation Request is true. Write Pulse from the processor is gated with the OPREQ and M/$\overline{IO}$ signals to assure proper operation.

For a large memory the next address line (ADR8) could be gated into the chain that generates the chip select signals, with similar write pulse generation for the higher order memory.

The $\overline{OPACK}$ signal is assumed to be false for the duration of all memory operations. This eliminates some gating from that control input. No problems will be encountered with this approach as long as the memories are fast enough for the clock speed being used with the processor. At a cycle time of 2.4μs, data must be returned to the processor by 1μs or less time from the OPREQ leading edge.

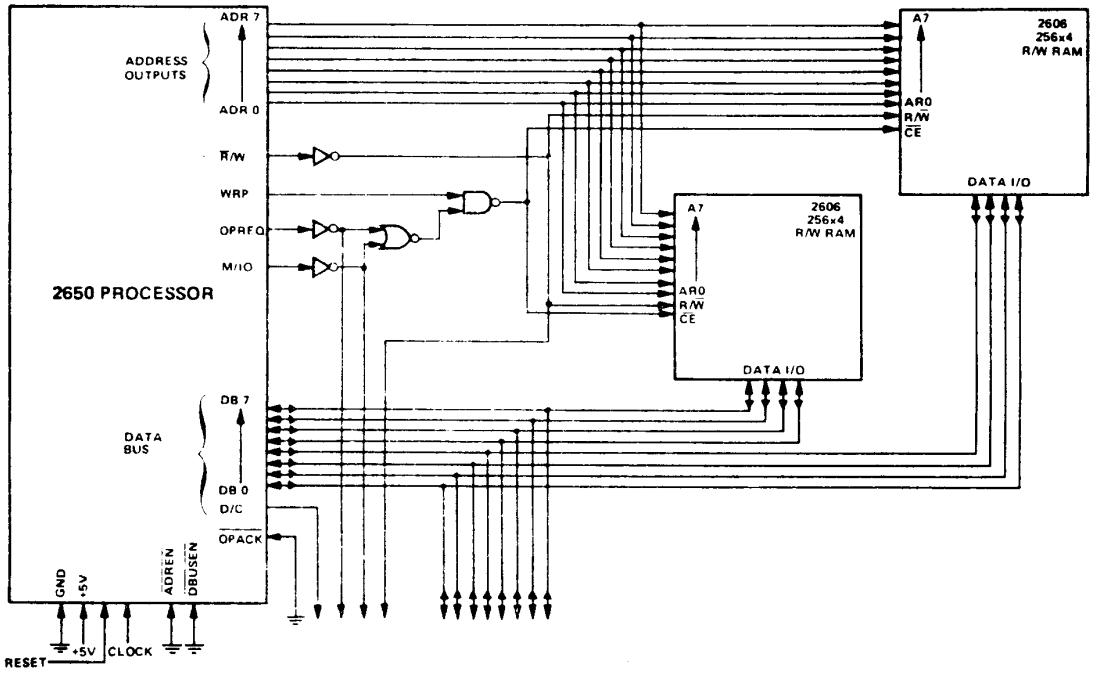


Figure 22.

APPENDIX B

I/O INTERFACE

Figure 23 shows one of many possible methods for buffering the data bus and interfacing it to several devices. There are advantages to be gained by using the Signetics 8T26. It has a PNP input buffer that keeps its low input level current at 200μA instead of 1.6mA. This lightens the load on the processor bus drivers and allows the processor to interface to several 8T26's if necessary. The 8T26 has four complete driver/receiver pairs in a package, so two packages can fully buffer the 8-bit data bus.

The control signals generated for use with I/O interfaces are very straightforward. Combining M/$\overline{IO}$ with OPREQ generates a signal that can often be used conveniently at the I/O devices instead of having each device derive the signal individually. In the figure it is gated with the Read/Write information in order to control the bus buffer.

Each I/O device must handle four basic processor interface functions:

(a) bus interface
(b) data transfer logic
(c) device selection logic
(d) transfer acknowledge logic

Depending on the nature of the complete system and the particular I/O device, these functions can be either extremely simple or fairly complex.

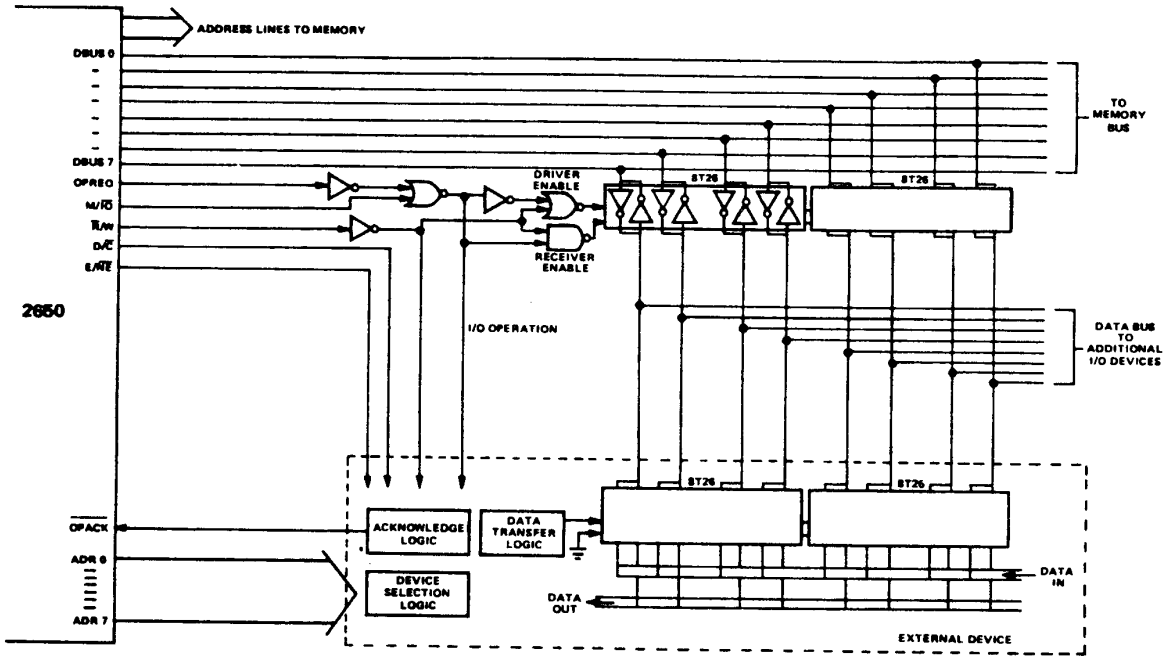


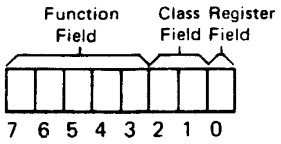
Figure 23.

APPENDIX C

INSTRUCTIONS, ADDITIONAL INFORMATION

The 2650 uses variable length instructions that are one, two or three bytes long. The instruction length is determined by the nature of the operation being performed and the addressing mode being used. Thus, the instruction can be expressed in one byte when no memory operand addressing is necessary, as with register-to-register or rotate instructions. On the other hand, for direct addressing instructions, three bytes are allocated. The relative and immediate addressing modes allow two-byte instructions to be implemented.

The 2650 uses explicit operand addressing; that is, each instruction specifies the operand address. The first byte of each 2650 instruction is divided into three fields and specifies the operation to be performed, the addressing mode to be used and, where appropriate, the register or condition code mask to be used.



The CLASS field specifies the instruction group, the major address mode and the number of processor cycles required for each instruction. The CLASS field also specifies, with one exception, the number of bytes in the instruction. The following table shows the specifications for each class.

CLASS FIELD	INSTRUCTION GROUP	ADDRESS REGISTER	BYTE LENGTH	DIRECT CYCLES
0	Arithmetic	Register	1	2
1	Arithmetic	Immediate	2	2
2	Arithmetic	Relative	2	3
3	Arithmetic	Absolute	3	4
4	Control (inc. rotate)		1	2
5	Control		1-2	3
6	Branch	Relative	2	3
7	Branch	Absolute	3	3

Within the arithmetic groups (classes 0, 1, 2, and 3) the function field specifies one of the eight operations as follows:

FUNCTION FIELD	ARITHMETIC OPERATION
0	LOAD
1	EXCLUSIVE OR
2	AND
3	INCLUSIVE OR
4	ADD
5	SUBTRACT
6	STORE
7	COMPARE

Within the branch group (classes 6 and 7) the function field specifies one of eight operations as follows:

FUNCTION FIELD	BRANCH OPERATION
0	Branch On Condition True
1	Branch To Subroutine On Condition True
2	Branch On Register Non-Zero
3	Branch To Subroutine On Register Non-Zero
4	Branch On Condition False
5	Branch To Subroutine On Condition False
6	Branch On Incrementing Register
7	Branch On Decrementing Register

There is very little pattern to the use of the function field within the control group (classes 4 and 5).

The register field is used to specify the index register, to specify the operand source register, to specify the destination register, or a condition code mask. For the register-to-register and the indexed instructions, register zero is implicitly assumed to be the source or the destination of the instruction. For all other instructions that involve a register, the register field allows any of four registers to be specified, except for indexed branch instructions which require that register 3 be specified.

Conditional branch instructions utilize the 2-bit register field as a condition code mask field. A few instructions use the register field as part of the operation code and consequently allow no variation in register usage.

APPENDIX D

INSTRUCTION SUMMARY

SIGNETICS 2650 PROCESSOR

ALPHABETIC LISTING

HEX	OP	Pg.
8C	ADDA	57
8D		
8E		
8F		
84	ADDI	56
85		
86		
87		
88	ADDR	56
89		
8A		
8B		
80	ADDZ	55
81		
82		
83		
4C	ANDA	61
4D		
4E		
4F		
44	ANDI	60
45		
46		
47		
48	ANDR	60
49		
4A		
4B		
41	ANDZ	59
42		
43		
9C	BCFA	75
9D		
9E		

HEX	OP	Pg.
98	BCFR	75
99		
9A		
1C	BCTA	74
1D		
1E		
1F		
18	BCTR	74
19		
1A		
1B		
FC	BDRA	77
FD		
FE		
FF		
F8	BDRR	77
F9		
FA		
FB		
DC	BIRA	76
DD		
DE		
DF		
D8	BIRR	76
D9		
DA		
DB		
5C	BRNA	78
5D		
5E		
5F		
58	BRNR	78
59		
5A		
5B		

HEX	OP	Pg.
BC	BSFA	81
BD		
BE		
B8	BSFR	81
B9		
BA		
7C	BSNA	82
7D		
7E		
7F		
78	BSNR	82
79		
7A		
7B		
3C	BSTA	80
3D		
3E		
3F		
38	BSTR	80
39		
3A		
3B		
BF	BSXA	83
9F	BXA	79
EC	COMA	67
ED		
EE		
EF		

HEX	OP	Pg.
E4	CØMI	66
E5		
E6		
E7		
E8	CØMR	66
E9		
EA		
EB		
E0	CØMZ	65
E1		
E2		
E3		
75	CPSL	72
74	CPSU	71
94	DAR	89
95		
96		
97		
2C	EØRA	65
2D		
2E		
2F		
24	EØRI	64
25		
26		
27		
28	EØRR	64
29		
2A		
2B		
20	EØRZ	63
21		
22		
23		

HEX	OP	Pg.
40	HALT	90
6C	IØRA	63
6D		
6E		
6F		
64	IØRI	62
65		
66		
67		
68	IØRR	62
69		
6A		
6B		
60	IØRZ	61
61		
62		
63		
0C	LØDA	53
0D		
0E		
0F		
04	LØDI	52
05		
06		
07		
08	LØDR	53
09		
0A		
0B		
00	LØDZ	52
01		
02		
03		

HEX	OP	Pg.
93	LPSL	69
92	LPSU	68
C0	NØP	87
77	PPSL	71
76	PPSU	70
30	REDC	85
31		
32		
33		
70	REDD	84
71		
72		
73		
54	REDE	85
55		
56		
57		
14	RETC	83
15		
16		
17		
34	RETE	84
35		
36		
37		
D0	RRL	67
D1		
D2		
D3		

HEX	OP	Pg.
50	RRR	68
51		
52		
53		
13	SPSL	70
12	SPSU	69
CC	STRA	55
CD		
CE		
CF		
C8	STRR	54
C9		
CA		
CB		
C1	STRZ	54
C2		
C3		
AC	SUBA	59
AD		
AE		
AF		
A4	SUBI	58
A5		
A6		
A7		
A8	SUBR	58
A9		
AA		
AB		
A0	SUBZ	57
A1		
A2		
A3		

HEX	OP	Pg.
F4	TMI	88
F5		
F6		
F7		
B5	TPSL	73
B4	TPSU	72
B0	WRTC	86
B1		
B2		
B3		
F0	WRTD	86
F1		
F2		
F3		
D4	WRTE	87
D5		
D6		
D7		
9B	ZBRR	73
8B	ZBSR	79

NUMERIC LISTING

HEX	OP	Pg.
00	LØDZ	52
01		
02		
03		
04	LØDI	52
05		
06		
07		
08	LØDR	53
09		
0A		
0B		
0C	LØDA	53
0D		
0E		
0F		
12	SPSU	69
13	SPSL	70
14	RETC	83
15		
16		
17		
18	BCTR	74
19		
1A		
1B		
1C	BCTA	74
1D		
1E		
1F		
20	EØRZ	63
21		
22		
23		

HEX	OP	Pg.
24	EØRI	64
25		
26		
27		
28	EØRR	64
29		
2A		
2B		
2C	EØRA	65
2D		
2E		
2F		
30	REDC	85
31		
32		
33		
34	RETE	84
35		
36		
37		
38	BSTR	80
39		
3A		
3B		
3C	BSTA	80
3D		
3E		
3F		
40	HALT	90
41	ANDZ	59
42		
43		

HEX	OP	Pg.
44	ANDI	60
45		
46		
47		
48	ANDR	60
49		
4A		
4B		
4C	ANDA	61
4D		
4E		
4F		
50	RRR	68
51		
52		
53		
54	REDE	85
55		
56		
57		
58	BRNR	78
59		
5A		
5B		
5C	BRNA	78
5D		
5E		
5F		
60	IØRZ	61
61		
62		
63		
64	IØRI	62
65		
66		
67		

HEX	OP	Pg.
68	IØRR	62
69		
6A		
6B		
6C	IØRA	63
6D		
6E		
6F		
70	REDD	84
71		
72		
73		
74	CPSU	71
75	CPSL	72
76	PPSU	70
77	PPSL	71
78	BSNR	82
79		
7A		
7B		
7C	BSNA	82
7D		
7E		
7F		
80	ADDZ	55
81		
82		
83		
84	ADDI	56
85		
86		
87		

HEX	OP	Pg.
88	ADDR	56
89		
8A		
8B		
8C	ADDA	57
8D		
8E		
8F		
92	LPSU	68
93	LPSL	69
94	DAR	89
95		
96		
97		
98	BCFR	75
99		
9A		
9B	ZBRR	73
9C	BCFA	75
9D		
9E		
9F	BXA	79
A0	SUBZ	57
A1		
A2		
A3		

HEX	OP	Pg.
A4	SUBI	58
A5		
A6		
A7		
A8	SUBR	58
A9		
AA		
AB		
AC	SUBA	59
AD		
AE		
AF		
B0	WRTC	86
B1		
B2		
B3		
B4	TPSU	72
B5	TPSL	73
B8	BSFR	81
B9		
BA		
BB	ZBSR	79
BC	BSFA	81
BD		
BE		
BF	BSXA	83

HEX	OP	Pg.
C0	NØP	87
C1	STRZ	54
C2		
C3		
C8	STRR	54
C9		
CA		
CB		
CC	STRA	55
CD		
CE		
CF		
D0	RRL	67
D1		
D2		
D3		
D4	WRTE	87
D5		
D6		
D7		
D8	BIRR	76
D9		
DA		
DB		
DC	BIRA	76
DD		
DE		
DF		
E0	CØMZ	65
E1		
E2		
E3		
E4	CØMI	66
E5		
E6		
E7		
E8	CØMR	66
E9		
EA		
EB		
EC	CØMA	67
ED		
EE		
EF		
F0	WRTD	86
F1		
F2		
F3		
F4	TMI	88
F5		
F6		
F7		
F8	BDRR	77
F9		
FA		
FB		
FC	BDRA	77
FD		
FE		
FF		

2650 INSTRUCTIONS

ORGANIZED BY FUNCTION

LOAD/STORE		Pg.
00	LØDZ	52
01		
02		
03		
04	LØDI	52
05		
06		
07		
08	LØDR	53
09		
0A		
0B		
0C	LØDA	53
0D		
0E		
0F		
C1	STRZ	54
C2		
C3		
C8	STRR	54
C9		
CA		
CB		
CC	STRA	55
CD		
CE		
CF		

ARITHMETIC		Pg.
80	ADDZ	55
81		
82		
83		
84	ADDI	56
85		
86		
87		
88	ADDR	56
89		
8A		
8B		
8C	ADDA	57
8D		
8E		
8F		
A0	SUBZ	57
A1		
A2		
A3		
A4	SUBI	58
A5		
A6		
A7		
A8	SUBR	58
A9		
AA		
AB		
AC	SUBA	59
AD		
AE		
AF		
60	IØRZ	61
61		
62		
63		
64	IØRI	62
65		
66		
67		

ARITHMETIC		Pg.
68	IØRR	62
69		
6A		
6B		
6C	IØRA	63
6D		
6E		
6F		
20	EØRZ	63
21		
22		
23		
24	EØRI	64
25		
26		
27		
28	EØRR	64
29		
2A		
2B		
2C	EØRA	65
2D		
2E		
2F		
41	ANDZ	59
42		
43		
44	ANDI	60
45		
46		
47		
48	ANDR	60
49		
4A		
4B		
4C	ANDA	61
4D		
4E		
4F		

BRANCH		Pg.
18	BCTR	74
19		
1A		
1B		
1C	BCTA	74
1D		
1E		
1F		
98	BCFR	75
99		
9A		
9C	BCFA	75
9D		
9E		
58	BRNR	78
59		
5A		
5B		
5C	BRNA	78
5D		
5E		
5F		
D8	BIRR	76
D9		
DA		
DB		
DC	BIRA	76
DD		
DE		
DF		
F8	BDRR	77
F9		
FA		
FB		
FC	BDRA	77
FD		
FE		
FF		
9F	BXA	79
9B	ZBRR	73

SUBROUTINE BRANCH		Pg.
38	BSTR	80
39		
3A		
3B		
3C	BSTA	80
3D		
3E		
3F		
B8	BSFR	81
B9		
BA		
BC	BSFA	81
BD		
BE		
78	BSNR	82
79		
7A		
7B		
7C	BSNA	82
7D		
7E		
7F		
BF	BSXA	83
BB	ZBSR	79

SUBROUTINE RETURN		
14	RETC	83
15		
16		
17		
34	RETE	84
35		
36		
37		

COMPARE		Pg.
E0	CØMZ	65
E1		
E2		
E3		
E4	CØMI	66
E5		
E6		
E7		
E8	CØMR	66
E9		
EA		
EB		
EC	CØMA	67
ED		
EE		
EF		

INPUT/OUTPUT		
30	REDC	85
31		
32		
33		
70	REDD	84
71		
72		
73		
B0	WRTC	86
B1		
B2		
B3		
F0	WRTD	86
F1		
F2		
F3		
54	REDE	85
55		
56		
57		
D4	WRTE	87
D5		
D6		
D7		

PROGRAM STATUS MANIPULATION

		Pg.
92	LPSU	68
93	LPSL	69
12	SPSU	69
13	SPSL	70
74	CPSU	71
75	CPSL	72
76	PPSU	70
77	PPSL	71
B4	TPSU	72
B5	TPSL	73

ROTATE INSTRUCTIONS

D0	RRL	67
D1		
D2		
D3		
50	RRR	68
51		
52		
53		

MISCELLANEOUS

		Pg.
C0	NØP	87
40	HALT	90
F4	TMI	88
F5		
F6		
F7		
94	DAR	89
95		
96		
97		

APPENDIX E

SUMMARY OF 2650 INSTRUCTION MNEMONICS

In these tables parentheses are used to indicate options. In no case are they coded in any instruction. The following abbreviations are used:

r — register expression, must evaluate to $0 \leqslant r \leqslant 3$.
v — value expression
* — indirect indicator
a — address expression
x — index register expression
X — index register expression with optional auto-increment or auto-decrement

NOTE:
— the use of the indirect indicator is always optional.
— when an index register expression is specified, it can be followed by ',+' or ',-' which indicates use of auto-increment or auto-decrement of the index register. Example:

```
LODA,0      DPR,R3,+
```

BXA, BSXA are exceptions and do not permit auto-increment or auto-decrement.
— even though an address expression is specified in a hardware relative addressing instruction, the assembler develops it into a value of $(-64 \leqslant V \leqslant +63)$.
— a memory reference instruction which requires indexing may use only register 0 as the destination of the operation.
— if an index register expression is used with either the BXA or BSXA instructions it must specify index register #3 (either register bank) for indexing. Any other value in the index field will produce an error during assembly. However, it is not necessary to use an index register expression with these instructions; a blank in this field will default to register 3.

LOAD/STORE INSTRUCTIONS			Length (bytes)
LODZ	r	Load Register Zero	1
LODI,r	v	Load Immediate	2
LODR,r	(*)a	Load Relative	2
LODA,r	(*)a(,X)	Load Absolute	3
STRZ	r	Store Register Zero	1
STRR,r	(*)a	Store Relative	2
STRA,r	(*)a(,X)	Store Absolute	3

ARITHMETIC INSTRUCTIONS			
ADDZ	r	Add to Register Zero	1
ADDI,r	v	Add Immediate	2
ADDR,r	(*)a	Add Relative	2
ADDA,r	(*)a(,X)	Add Absolute	3
SUBZ	r	Subtract from Register Zero	1
SUBI,r	v	Subtract Immediate	2
SUBR,r	(*)a	Subtract Relative	2
SUBA,r	(*)a(,X)	Subtract Absolute	3

LOGICAL INSTRUCTIONS			
ANDZ	r	And to Register Zero	1
ANDI,r	v	And Immediate	2
ANDR,r	(*)a	And Relative	2
ANDA,r	(*)a(,X)	And Absolute	3
IORZ	r	Inclusive or to Register Zero	1
IORI,r	v	Inclusive or Immediate	2
IORR,r	(*)a	Inclusive or Relative	2
IORA,r	(*)a(,X)	Inclusive or Absolute	3
EORZ	r	Exclusive or to Register Zero	1
EORI,r	v	Exclusive or Immediate	2
EORR,r	(*)a	Exclusive or Relative	2
EORA,r	(*)a(,X)	Exclusive or Absolute	3

COMPARISON INSTRUCTIONS			
COMZ	r	Compare to Register Zero	1
COMI,r	v	Compare Immediate	2
COMR,r	(*)a	Compare Relative	2
COMA,r	(*)a(,X)	Compare Absolute	3

ROTATE INSTRUCTIONS			Length (bytes)
RRR,r		Rotate Register Right	1
RRL,r		Rotate Register Left	1

BRANCH INSTRUCTIONS			
BCTR,v	(*)a	Branch on Condition True Relative	2
BCFR,v	(*)a	Branch on Condition False Relative	2
BCTA,v	(*)a	Branch on Condition True Absolute	3
BCFA,v	(*)a	Branch on Condition False Absolute	3
BRNR,r	(*)a	Branch on Register Non-Zero Relative	2
BRNA,r	(*)a	Branch on Register Non-Zero Absolute	3
BIRR,r	(*)a	Branch on Incrementing Register Relative	2
BIRA,r	(*)a	Branch on Incrementing Register Absolute	3
BDRR,r	(*)a	Branch on Decrementing Register Relative	2
BDRA,r	(*)a	Branch on Decrementing Register Absolute	3
BXA	(*)a(,x)	Branch Indexed Absolute, Unconditional	3
ZBRR	(*)a	Zero Branch Relative, Unconditional	2

SUBROUTINE BRANCH/RETURN INSTRUCTIONS			
BSTR,v	(*)a	Branch to Subroutine on Condition True, Relative	2
BSFR,v	(*)a	Branch to Subroutine on Condition False, Relative	2
BSTA,v	(*)a	Branch to Subroutine on Condition True, Absolute	3
BSFA,v	(*)a	Branch to Subroutine on Condition False, Absolute	3
BSNR,r	(*)a	Branch to Subroutine on Non-Zero Register, Relative	2
BSNA,r	(*)a	Branch to Subroutine on Non-Zero Register, Absolute	3
BSXA	(*)a(,x)	Branch to Subroutine, Indexed, Unconditional	3
RETC,v		Return From Subroutine, Conditional	1
RETE,v		Return From Subroutine and Enable Interrupt, Conditional	1
ZBSR	(*),a	Zero Branch to Subroutine Relative, Unconditional	2

PROGRAM STATUS INSTRUCTIONS			
LPSU		Load Program Status, Upper	1
LPSL		Load Program Status, Lower	1
SPSU		Store Program Status, Upper	1
SPSL		Store Program Status, Lower	1
CPSU	v	Clear Program Status, Upper, Selective	2
CPSL	v	Clear Program Status, Lower, Selective	2
PPSU	v	Preset Program Status, Upper, Selective	2
PPSL	v	Preset Program Status, Lower, Selective	2
TPSU	v	Test Program Status, Upper, Selective	2
TPSL	v	Test Program Status Lower, Selective	2

INPUT/OUTPUT INSTRUCTIONS			
WRTD,r		Write Data	1
REDD,r		Read Data	1
WRTC,r		Write Control	1
REDC,r		Read Control	1
WRTE,r	v	Write Extended	2
REDE,r	v	Read Extended	2

MISCELLANEOUS INSTRUCTIONS			
HALT		Halt, Enter Wait State	1
DAR,r		Decimal Adjust Register	1
TMI,r	v	Test Under Mask Immediate	2
NOP		No Operation	1

APPENDIX F

NOTES ABOUT THE 2650 PROCESSOR

1. AUTO-INCREMENT, DECREMENT of index register. This feature is optional on any instruction which uses indexing with the exception of BXA and BSXA. The increment or decrement occurs before the index register is added to the displacement in the instruction.
2. The contents of registers when used for indexing are considered to be unsigned absolute numbers. Consequently, index registers can contain values from 0 to 255. They "wrap-around" so that the number following 255 is 0.
3. Only absolute addressing instructions can be indexed.
4. The Branch on Incrementing Register or Decrementing Register instructions perform the increment or decrement before testing for zero. The only time the branch address is not taken, is when the register contains zero.
5. All hardware relative addressing is implemented as modulo 8K and therefore relative addressing across the top of a page boundary will result in a physical address near the bottom of the page being accessed. For example:

   $1FFC_{16}$ LODR,R2 $+16

   This instruction results, during execution, in accessing the byte at location 000C in the same page as the instruction. Similarly, negative relative addresses from near the bottom of a page may result in an effective address near the top of the page.
6. Page boundaries cannot be indexed across.
7. Data can always be accessed across a page boundary through use of relative indirect or absolute indirect addressing modes.
8. The only way to transfer control to a program in some other page is to branch absolute or branch indirectly to the new page. Program execution cannot flow across a page boundary.
9. Unconditional branch or branch to subroutine instructions are coded by specifying a value of 3 in the register/value field of BSTA, BSTR, BCTA or BCTR. Example:

```
UN      EQU     3
        • • •
        • • •
        • • •
        BSTA,UN PAL
        BCTR,3  LOOP
```

   Unconditional branches on conditions false (BCFA, BCFR) are not allowed.

APPENDIX G

ASC II AND EBCDIC CODES

This table presents the only characters that the assembler will recognize in an A or E type constant and their equivalent codes in hexadecimal.

VALID CHARACTERS	EBCDIC CODE	ASC II CODE	VALID CHARACTERS	EBCDIC CODE	ASC II CODE
0	F0	30	V	E5	56
1	F1	31	W	E6	57
2	F2	32	X	E7	58
3	F3	33	Y	E8	59
4	F4	34	Z	E9	5A
5	F5	35	blank	40	20
6	F6	36	.	4B	2E
7	F7	37	(	4D	28
8	F8	38	+	4E	2B
9	F9	39	|	4F	7C
A	C1	41	&	50	26
B	C2	42	!	5A	21
C	C3	43	$	5B	24
D	C4	44	*	5C	2A
E	C5	45	)	5D	29
F	C6	46	;	5E	3B
G	C7	47	¬ or ~	5F	7E*
H	C8	48	-	60	2D
I	C9	49	/	61	2F
J	D1	4A	,	6B	2C
K	D2	4B	%	6C	25
L	D3	4C	— or ←	6D	5F*
M	D4	4D	>	6E	3E
N	D5	4E	?	6F	3F
O	D6	4F	:	7A	3A
P	D7	50	#	7B	23
Q	D8	51	@	7C	40
R	D9	52	'	7D	27
S	E2	53	=	7E	3D
T	E3	54	"	7F	22
U	E4	55	<	4C	3C

*may have different graphic symbols on different computer systems

APPENDIX H

COMPLETE ASCII CHARACTER SET

(MSB) b_7				0	0	1	1	1	1
			b_6	1	1	0	0	1	1
			b_5	0	1	0	1	0	1
b_4	b_3	b_2	b_1						
0	0	0	0	SP	0	@	P	`	p
0	0	0	1	!	1	A	Q	a	q
0	0	1	0	"	2	B	R	b	r
0	0	1	1	#	3	C	S	c	s
0	1	0	0	$	4	D	T	d	t
0	1	0	1	%	5	E	U	e	u
0	1	1	0	&	6	F	V	f	v
0	1	1	1	'	7	G	W	g	w
1	0	0	0	(	8	H	X	h	x
1	0	0	1	)	9	I	Y	i	y
1	0	1	0	*	:	J	Z	j	z
1	0	1	1	+	;	K	[	k	{
1	1	0	0	,	<	L	\	l	|
1	1	0	1	–	=	M	]	m	}
1	1	1	0	.	>	N	↑	n	~
1	1	1	1	/	?	O	←	o	DEL