FAIRCHILD SEMICONDUCTOR

PPS 25

Programmed Processor System Preliminary Users Manual

MADE IN FAIRCHILD

OCTOBER 25, 1972

PPS 25 PROGRAMMED PROCESSOR SYSTEM

A. GENERAL DESCRIPTION

I. INTRODUCTION

MOS/LSI technology was first applied commercially in electronic desk top calculators. In the late 1960's, virtually every calculator manufacturer had new electronic designs on the market that utilized MOS/LSI chips. The low cost potential of MOS/LSI made possible what is now acknowledged to be a revolution in calculator system design.

The advances of MOS/LSI technology in the 1970's have made possible both increased circuit complexity and performance. In this decade a similar equipment revolution is foreseen in the minicomputer and data processor systems field where required computation speed and logic density are higher than in most calculator systems.

To meet the needs of this market, Fairchild developed a family of micro-programmed MOS/LSI processor blocks called the PPS 25 (Programmable Processor System). Equipment engineers have long recognized the desirability and flexibility of designing digital systems using a micro-programmed processor as the heart of the design. Such systems are very practical in that they can be readily programmed around specific system requirements. Unfortunately, the size and cost of minicomputers to date have limited their application to relatively large systems. However, with the availability of the PPS 25, small systems can be readily and economically designed. In fact, it is now possible for the equipment designer to develop small programmable digital processing systems for under $50.

II. FEATURES OF THE PPS 25 SYSTEM

The PPS 25 system was designed to fill the gap between intermediate to upper-end calculators and minicomputers. It substitutes a few MOS/LSI packages where normally several hundred TTL MSI or SSI packages would be required to implement the system function.

The PPS 25 system has a versatile instruction set which permits the system to perform a wide variety of different functions. The basic operating features of the system are summarized below:

BCD Serial/Parallel Processing Unit with 95 Instructions

- 25-Digit Serial, 4-Bit Parallel Organization
- 62.5 μs Word Time
- 2.5 μs Bit Time
- Programmable Time Enable Patterns Provide Versatile Data Field Selection Within the 25-Digit Word
- 4-Level Subroutine Nesting
- Both 2 and 3-Way Conditional Branch Structure
- External Interrupt Capability
- High Speed Operation Assured by:
  - Basic Serial/Parallel Arithmetic Unit
  - Overlapped Fetch and Execute Instruction Cycle
  - Separate Micro-instruction and Data Busses
- Up to Seven 25-Digit Memory Registers Available with Arbitrary Expansion Provided
- Up to 26 Programmable ROMs -- 256 x 12 Bits Each
- Input Keyboard Capability for Up to 61 Keys and 32 Mode Switches
- Standard Output Display Chip Available for 16-Digit Display or Communications Interface with Provision for Other Custom Outputs
- Versatile Data Buss Structure Provides Flexible I/O Interface Expansion

III. APPLICATIONS OF THE PPS 25 SYSTEM

The PPS 25's flexible set of 95 instructions permits the system to perform a wide variety of functions. Examples of systems where the PPS 25 can be employed are:

UPPER-END SCIENTIFIC CALCULATORS

Scientific calculators formed with these MOS/LSI blocks handle up to 25-digit numbers. They can be programmed to handle either fixed point, floating point, or scientific notation, and, in their basic form, will add, subtract, multiply, divide and take square root. They also can perform complex arithmetic functions such as sine, cosine, and log. The programmable ROM's provide a great number of different capabilities. Multiply, for example, can be performed with any one of a number of algorithms. Furthermore, the word length need not be 25 digits. The blocks can be used to build a calculator with any number of digits up to 25.

Internally, the chips handle 25-digit words (stored in four parallel 25-bit shift registers). The word length actually used and the format within the word are entirely optional. A posible format for a calculator with sceintific notation would be to devote three digits to the exponent and its sign, leaving 21 digits for the mantissa, and one digit for its sign.

All numbers are stored and processed in binary coded decimal form. These four bits are processed in parallel fashion, while the digits are processed in series. The system can be clocked at a maximum rate of 400 kHz. At this rate, the digit and word times are 2.5 and 62.5 μs respectively.

Up to 61 keys and 32 mode switches can be handled by standard keyboard interface circuits, and up to 16 digits of display can be controlled by the output chip.

CONTROL SYSTEMS

Because of the micro-program flexibility, the PPS 25 system can be used in process control systems, vending machine systems, medical instrumentation, numerical machine control, traffic light controllers, and in virtually all medium speed programmed control systems.

COMPUTER SYSTEMS

The PPS 25 system is also organized for high speed calculations and data processing. Basic features -- serial/parallel word structure, overlapped fetch and execute, and separate data and micro-instruction busses -- were designed in. This was done to extend the application of the set into the fields of electronic point-of-sale terminals, cash registers banking terminals, small business machines, etc. For example, the addition of two 25-digit numbers can be executed in 62.5 μs, and the multiplication of similar numbers can be accomplished in less than 50 ms.

PERIPHERAL SYSTEMS

This micro-programmed system is also ideally configured for many peripheral applications. If keyboard entry is required, keyboard devices provide a direct interface, with full n-key rollover and anti-bounce on up to 61 keys, plus 32 mode switches. Other forms of input data can be fed directly into the data source buss. If digit display of up to 16 digits is required, the output communication device provides the necessary interface; of multiple dedices can serve as a general output communication link.

If special interfaces are required, such as printers or magcard readers, custom interfaces can be provided directly from the destination buss or ROMs can be utilized to emulate the desired interface within the structure of the system.

B. SPECIFIC DESCRIPTION

I. THE PPS 25 CONSISTS OF 4 BASIC ELEMENTS:

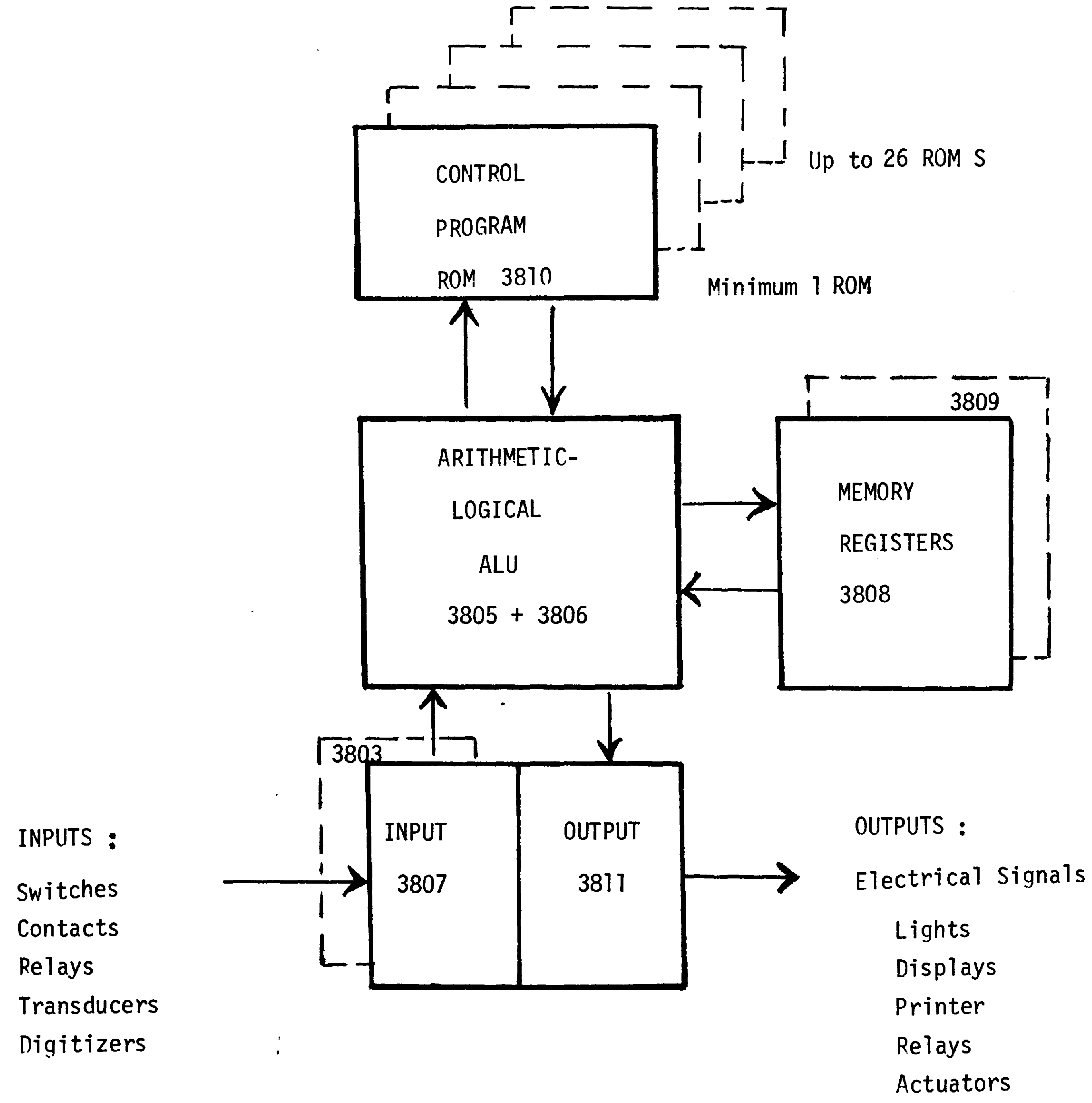


Figure 1 -

BASIC PPS 25 ELEMENTS

THE FUNCTIONS OF THE 4 ELEMENTARY BLOCKS ARE DESCRIBED AS FOLLOWS:

1. The ALU is called the Arithmetic and Logical Unit. This is the brain. It performs all the arithmetic and logical operations as well as provides timing, synchronization, addressing, and supervision to the rest of the system. Its functions are performed by the 3805 Arithmetic Chip and the 3806 Function and Timing Chip.

2. The next most important and essential unit is the Control ROM. Its function is to store, under the name Program, all the minute steps of information called "INSTRUCTION" that the ALU needs to perform a meaningful task. The ALU calls on the control ROM to request the next instruction. This is called the Addressing cycle; the ROM, in turn, sends back ONE INSTRUCTION to the ALU.
   One control ROM can store 256 program steps.
   The ROM control function can be implemented with as many as twenty-six 3810 chips, but as few as one (ONE IS MINIMUM) is required.

3. The Memory Block permits the storage of data in the form of digits, each 4 bits binary coded (BCD Digits); 25 such digits can be stored in each register. There is a total of 6 registers available: 3 registers on chip 3808, 3 registers on chip 3809. The term "register" is arbitrary; each register could very well, indeed, store 2 words of 12 digits or 3 words of 8 digits, allowing a total data storage of: twelve 12-digit words or eighteen 8-digit words.

It should be noted that the Arithmetic Chip also contains one 25-digit register and that a simple system only requiring counting, i.e., increasing or decreasing a number by 1 or 2 or 3, etcetera, can be implemented without register chips. On the other hand, systems requiring more than 6 registers can be implemented with MOS/LSI customs techniques.

4. The Input/Output Block is the part of the system that permits communication between the processor and the user. An Input Encoding Device is the chip 3807 (also referred to as Keyboard Encoder Chip). Thirty-one momentary contact closures can be input at the key matrix (8 "R" Lines and 4 "D" lines); 16 static switches can be input at the mode switch matrix (8 "S" lines and 2 "D" lines). The words "switch" and "contact" do not necessarily imply keybutton having a mechanical switch; a MOS or Bipolar transistor correctly connected can function as an input switch to either matrix.

An Input Expander Chip 3803 is also available; it doubles the number of input lines available.

The output function is provided by the 3811 chip. It is more specifically designed to interface with a numeric display of up to 16 digits. However, its outputs are available in BCD code for the data and encoded for the 16-digit output selection, and could easily be translated into any desirable code for data processing.

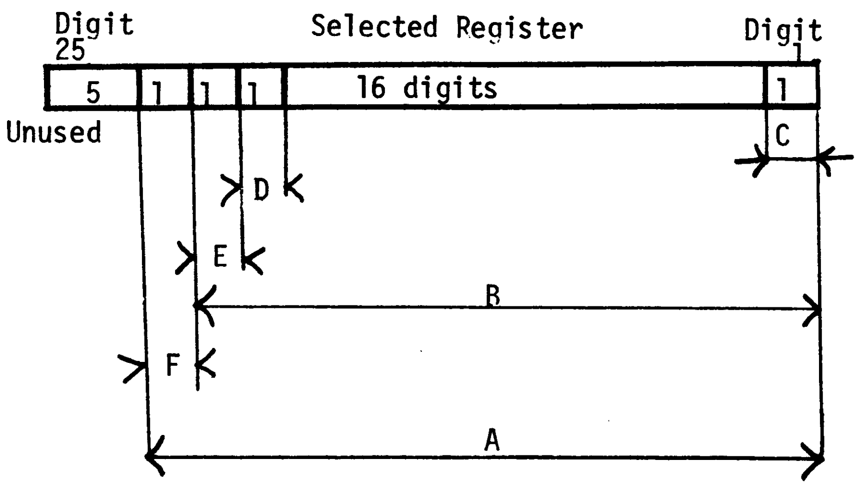
It should be pointed out that the types of input and output are not limited to the present devices. Other modes of input and output can be utilized; some examples will be given later.

II. PPS 25 OPERATION - CONCEPTS AND FEATURES:

The ALU Block gives to the user the following tools to implement (program) a digital system: See the examples below, and the block diagram of Fig. 2.

1. Control over the data registers configuration. The 25 digits of the data register(s) can be arranged or monitored by 6 different , overlapping,or non-overlapping time intervals, effectively breaking up the selected registers into individual Time Enable during which the data is manipulated under control of the program.

Example 1:



TE : A. Total Field Used

B. Total Number Field = 19 Digits

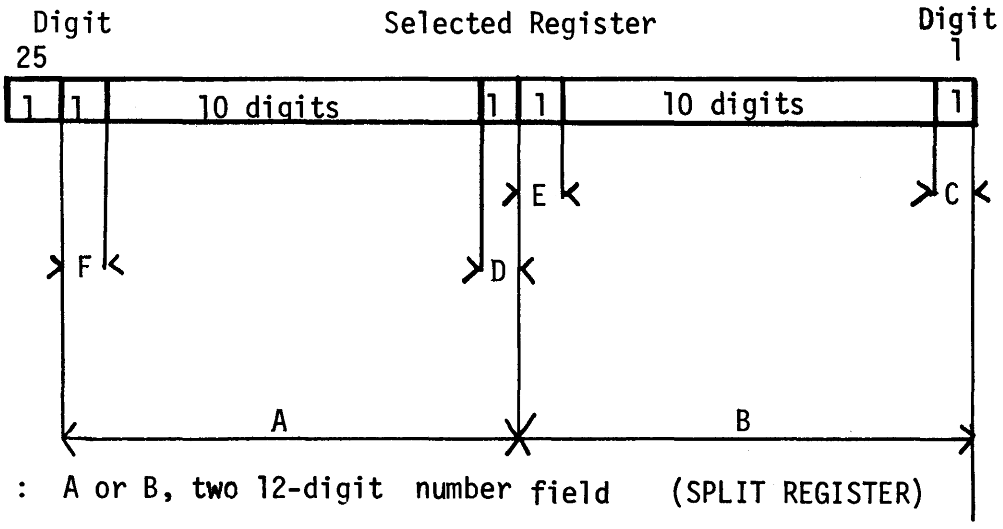
C. Least Significant Digit

D. Most Significant Digit

E. Sign of Number

F. Decimal Point Counter (Up to 9)

Example 2:



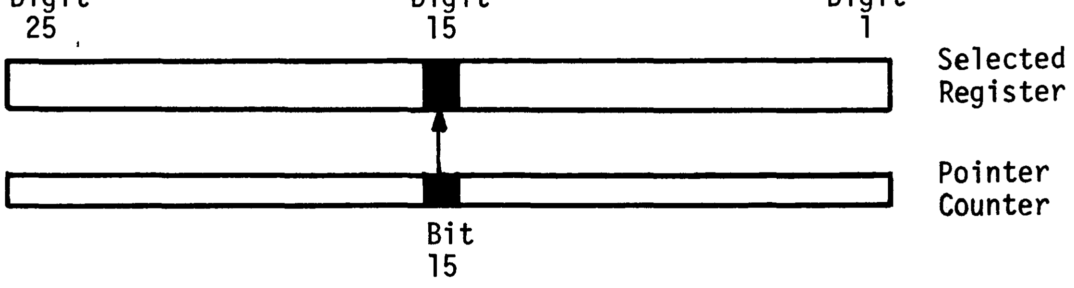
TE : A or B, two 12-digit number field (SPLIT REGISTER)

C or D, Least significant digit of each number

E or F, Most significant digit of each number

In addition to these 6 Time Enables which are mask programmable, a digit time indicator is programmable to any one of the 25 digits (one at the time only). It is controlled by a 25 position pointer counter.

Example 3:



Digit 15 above is available for data manipulation by the arithmetic unit.

The use of the Time Enable or of the Pointer effectively limits the data operated upon to the time interval chosen, i.e., selecting Time Enable A in Example 2 will limit all data operation to word A or 12 digits.

2. An Arithmetic Unit that will operate on one digit, or, on up to 25 digits in one ALU cycle (62.5 microseconds per instruction execution cycle). The operations are performed in decimal arithmetic, with interdigit carry or borrow if necessary. If carry or borrow extends past the TE, the FLAG and/or BORROW signals indicate this condition. In the simplest form any digit of any register can be used as an up and down decimal counter. The instructions: Increment and Decrement will perform this function. Other instructions will perform the arithmetic operation such as Add, Subtract, Complement, as well as Data Movement, Left Shift and Right Shift. Data can be transferred with a MOVE instruction to and from any register to another register via the data busses. X0, X1, X2, X3 and Y0, Y1, Y2, Y3.

Another mode of operation would be to transfer data to and from the "A" register in the ALU to an external bulk memory via the data busses. Data can be non-decimal (modulo 16) if only data movement, right shift and shift load immediate are used.

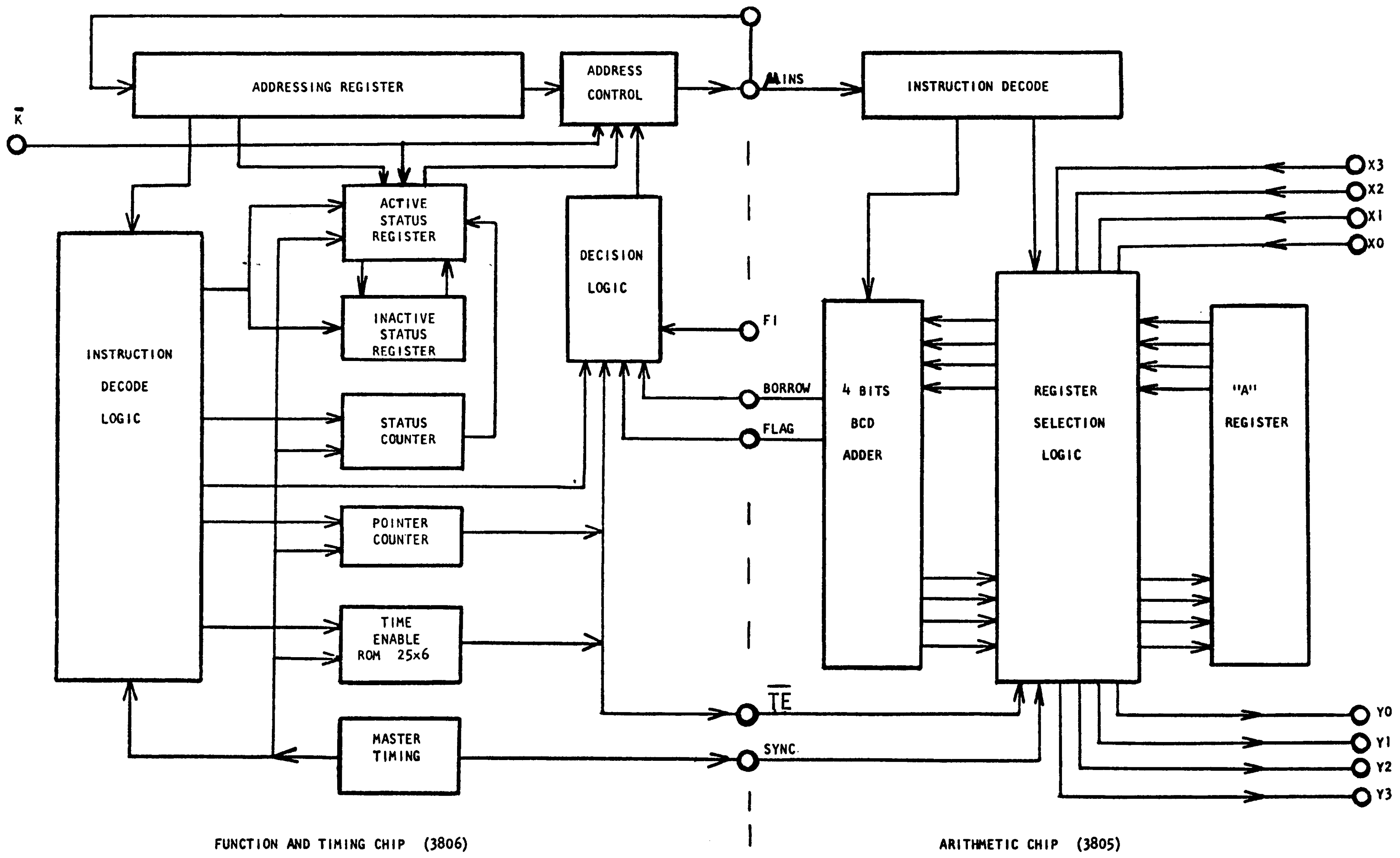


FIGURE 2. BLOCK DIAGRAM OF A.L.U.

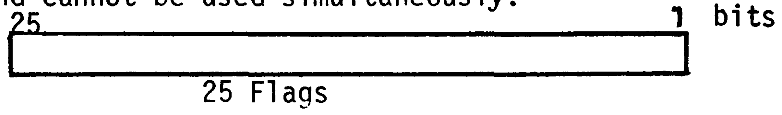
3. An Address Register, also referred to as program counter, holds the 8* bits of address to be used to call the ROM to send a new instruction to the ALU. The address, 8 bit* as well as the instruction, are transmitted between the ALU and the control ROM on a single line, the Micro-Instruction Line, during one (25 clocks) ALU cycle. The address cycle, 8 bit*; and the instruction cycle, 12 bits, are time coded (multiplexed). The timing synchronization between the ALU and the ROM is assured by the SYNC liné.

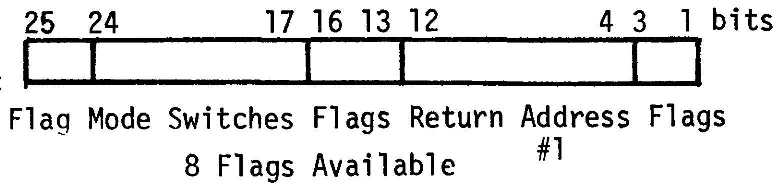
4. Two 25-bit Status Registers are available to store control information as follows:
   One status register is active at all times in the ALU; the other is inactive or surrogate and can be exchanged with the active register by a program instruction.
   Each register can be used for three different purposes:
   4.1. Each register can be used to store up to 25 individual Flags. A Flag is equivalent to a Flip-Flop. It can be CLEARED, SET, RESET. Its state can be used to control other logical paths "Interrogated".
   4.2. In each status register, 8 of the above Flags can store the condition On or Off of 8 switches externally connected to the Input Encoding Chip 3807 at the Mode Switch Matrix. Therefore, the condition of those switches become a Flag that can be Interrogated.

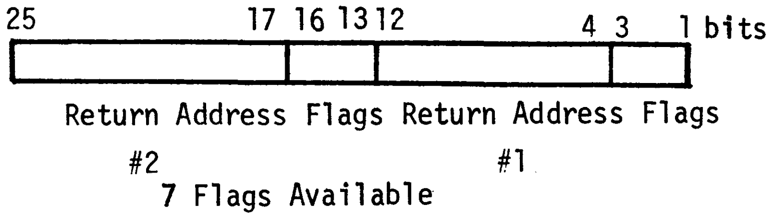
*Note: Actually 9 bits are reserved for future expansion of addressing.

4.3. Each status register can be used to store 2 program addresses, 9 bits each, that can be used as return address for subroutine calls. (Up to 4-level subroutine nesting)

Some of the conditions above are mutually exclusive and cannot be used simultaneously:

Ex. 1:


Ex. 2:

25 24 17 16 13 12 4 3 1 bits
Flag Mode Switches Flags Return Address Flags
8 Flags Available
#1

Ex. 3:

25 17 16 13 12 4 3 1 bits
Return Address Flags Return Address Flags
#2
#1
7 Flags Available

The same status bits cannot be used simultaneously for 2 or 3 different functions. However, they can be used sequentially in the same program in the three different modes.

5. The ALU can call upon up to 25 control ROM to deliver programs or portions of a program. Each control ROM has 256 addresses with each address containing one Instruction. Ten ROMs equal 2560 instructions (that's a lot of programming capability). Switching control ROM is accomplished very simply under program control by an instruction called ROM SELECT.

6. The Control ROM, besides being programmed to contain Instructions ordering the ALU to perform Arithmetic or Logical operation, can be used to control an infinite number of devices implemented with discrete, SSI, MSI, LSI, etc. A total of 95 instructions, are available: 30 ALU Instructions, 63 I/O Instructions.
   The Micro-Instruction line contains the necessary information and only needs to be decoded at the proper time with the timing reference supplied by the SYNC. line.

   Possible use of the I/O Instructions are provided with the Input Encoding chips:

   Device 3807 - 5 Instructions

   Device 3803 - 5 Instructions

   or output chip:

   Device 3811 - 8 instructions
7. The ALU has a direct input line $\overline{K}$ (see figure 2) which permits an external device to be input in the following manners:

7.1. Direct access to the address control logic, were 8 bits properly coded can be used as address modification (see IV-4).

7.2. Direct access to the active status register, were 8 bits properly coded, can be stored as flags (see 2.2.1 and IV-4).

C. PPS 25 - INSTRUCTION SET:

Two classes of instructions are available, which are listed in the table on the next page.

1. Arithmetic Instructions
2. Non-arithmetic Instructions

1. ARITHMETIC INSTRUCTIONS:

These instructions involve data manipulation and therefore make use of the registers and Time Enable.

The format of the Arithmetic Instruction is as follows:

11 10 9	8 7 6	5 4 3	2 1 0	Bits
OPERATION CODE	DESTINATION REGISTER	SOURCE REGISTER	TIME ENABLE CODE	
3 bits	3 bits	3 bits	3 bits	

1.1 The Time Enable is designated by a 3-bit code. There are six programmable patterns, each called up by a different code. There is no restriction on this programming; the different Time Enable can be overlapping, adjacent or separated. The Time Enable could encompass the entire word, only a few digits, or it can just cover one digit. Code 7 calls up a pointer as time-enable. By definition, this is a 1-character time and can select any one of the 25 digits in a word.

Time Enable Code 0 calls out a class of non-arithmetic instructions.

1.2 The registers are designated by a 3-bit code. Assignments are as follows:

Binary Code	Octal Code	Register
000	0	Accumulator
100	4	B Register on Memory Chip #1 (3808)
101	5	C Register on Memory Chip #1
110	6	D Register on Memory Chip #1
001	1	E Register on Memory Chip #2 (3809)
010	2	F Register on Memory Chip #2
011	3	G Register on Memory Chip #2
111	7	H Dummy Register used as destination register for test instructions. (See description of index type instructions.)

Any register can be selected as either a source or a destination register. Designating the source register as X and the destination register as Y.

1.3 The 3-bit operation code selects one of the following eight instructions:

Binary Code	Octal Code	Function	Mnemonic	Operation
000	0	BCD Addition	ADD	The contents of the source register are added to the accumulator and the result is transferred to the destination register. **
		000 | Y | X | TE		A + X → Y
001	1	BCD Subtraction	SUB	The contents of the source register are subtracted from the accumulator and the result is stored in the destination register. **
		001 | Y | X | TE		A - X → Y
		001 | Y | 000 | TE		A - A → Y i.e. clear Y

This instruction must not be used when there is a possiblility of the accumulator containing illegal codes (i.e., power is turned on).

Binary Code	Octal Code	Function	Mnemonic	Operation
010	2	Transfer (Move)	MOV	The contents of the accumulator are transferred to the destination register. The contents of the source register replace the original contents of the accumulator. The content of the source register is not affected.
		$\boxed{010}\boxed{Y}\boxed{X}\boxed{TE}$		$A \rightarrow Y; \quad X \rightarrow A$
		$\boxed{010}\boxed{Y}\boxed{000}\boxed{TE}$		$A \rightarrow Y$
		$\boxed{010}\boxed{000}\boxed{X}\boxed{TE}$		$X \rightarrow A$
011	3	Complementation	COM	The contents of the source register are complemented (subtracted from zero) and transferred to the destination register. **
		$\boxed{011}\boxed{Y}\boxed{X}\boxed{TE}$		$0 - X \rightarrow Y$ (Result is the ten's complement of **X**)
100	4	Increment	INC	The contents of the source register are incremented by one and the result is stored in the destination register. **
		$\boxed{100}\boxed{Y}\boxed{X}\boxed{TE}$		$X + 1 \rightarrow Y$
101	5	Decrement	DEC	The contents of the source register are decremented by one and the result stored in the destination register. **
		$\boxed{101}\boxed{Y}\boxed{X}\boxed{TE}$		$X - 1 \rightarrow Y$

** The contents of the source and accumlator registers are not changed by the instruction operation.

Binary Code	Octal Code	Function	Mnemonic	Operation
110	6	Left Shift	LSH	The contents of the selected register are shifted left one digit. The digit at the right-most edge of the time enable pattern is zeroed.
		$\boxed{110}\boxed{Y}\boxed{X}\boxed{TE}$		Left Shift X → Y
		Clear (X = A, Y ≠ A)	CLR	The contents of the selected register are cleared.
		$\boxed{110}\boxed{Y}\boxed{000}\boxed{TE}$		0 → Y (Y ≠ A)
111	7	Right Shift	RSH	The contents of the selected register are shifted right one digit. The digit at the left-most edge of the time enable pattern is zeroed. As with all operations, digits not selected by the time enable pattern are unaffected.
		$\boxed{111}\boxed{Y}\boxed{X}\boxed{TE}$		Right Shift X → Y

One additional instruction is permitted if the shift instructions above are modified as follows:

11_	6 or 7	Shift Load Immediate	SLI	The contents of the accumulator are shifted left and 4-bits of the instruction are loaded as data into the digit at the right-most edge of the time enable pattern.

		$\boxed{11a}\boxed{111}\boxed{bcd}\boxed{TE}$		Load (a,b,c,d) into digit
				a = BCD Value 8
				b = BCD Value 4
				c = BCD Value 2
				d = BCD Value 1

Note: In the above description the term "contents" refers to the digits selected by the time enable pattern.

2. NON-ARITHMETIC INSTRUCTIONS:

Bits 0.1.2 of instruction are always zero.

These instructions are divided into three subclasses.

1. Index Instructions (Address modification)
2. Control Instructions
3. Input/Output Instructions

2.1 Index Instructions:

These instructions are used to modify the program address. Normally the address register is incremented by one after each instruction. Index instructions allow modification of the address register in order to provide branching.

The format of the Index Instruction is as follows:

11	10	9	8	7	6	5	4	3	2	1	0	Bits
–	–	–	–	–	–	–	–	1	0	0	0	

8 bits (4 to 11) are reserved for address modification. Two types of address modification are possible.

2.1.1 Unconditional Branching (BRU)

During execution of this instruction, the bits 4 through 10 are used to indicate the magnitude of the jump; bit 11 indicates the sign, "0" forward jump, "1" backward jump; the longest forward jump is over 127 addresses, Code 0 1 1 1 1 1 1 1 . The longest backward jump is to the 126th address before the test, code 1 0 0 0 0 0 0 0 . 1 1 1 1 1 1 1 1 is an illegal code. This instruction can be used after any instruction to perform an unconditional branching with the exception of the following 6 instructions where it would result in a conditional branching. SUB,COM, INC, DEC, Interrogate status (ISB), Interrogate pointer (IPT), see sections 2.1.2 and 2.2.1.

2.1.2 Conditional Branching (BOC)

The conditional branching results from having the index instruction code following an Arithmetic Test

SUB	Subtract	$A-X \longrightarrow Y$
COM	Complement	$0-X \longrightarrow Y$
INC	Increment	$X+1 \longrightarrow Y$
DEC	Decrement	$X-1 \longrightarrow Y$

The destination register Y can be specified to be "H" dummy register (code 7), in order to provide a non-destructive test. Interrogate instructions also result in a BOC if followed by the index instruction. Two modes of conditional branching can be selected by the following instructions:

	11			0	
BR3	0 0 0	0 0 1	1 0 0	0 0 0	Octal Code 0140

Selects a 3-way conditional branching.

BR2	0 0 0	0 0 0	0 1 0	0 0 0	Octal Code 0020

Selects a 2-way conditional branching.

The control instruction needs to be used only once at the beginning of the selection of the 2-way or 3-way mode.

2.1.2.1 3-way conditional branching selected:

During execution of the index instruction one of three branches will occur:

Condition 1: Bits 8 thru 11 ** of the index instruction are added to the addressing register as the instruction address is being incremented.

Condition 2: Bits 4 thru 7 ** of the index instruction are added to the addressing register as the instruction address is being incremented.

Condition 3: The address register is incremented by one with no modification.

These conditions are summarized in the following table:

	Condition 1	Condition 2	Condition 3
Type of Test Instruction	Bits 8-11** Modify Address	Bits 4-7** Modify Address	Address is Not Modified
A - X (SUB)	$A < X$	$A > X$	$A = X$
0 - X (COM)	$X \neq 0$		$X = 0$
X - 1 (DEC)	$X = 0$	$X \neq 0$ or 1	$X = 1$
X + 1 (INC)	$X + 1 = 0$	$X + 1 \neq 0$	

** Each 4-bit modifier has a sign bit and a maximum magnitude of 7.

0000	The shortest forward jump, calling up the subsequent address.
0111	The largest forward jump, jumping <u>over</u> seven addresses.
1111	An illegal index that would call up the index word again.
1110	The shortest backward jump, calling up the test instruction again.
1101	Jumps back to one address prior to the test, thus forming a useful loop.
1000	The largest backward jump, calling up the sixth address before the test.

2.1.2.2 2-way conditional Branching Selected:

During execution of the index instruction one of two branches will occur:

Condition 1: Bits 4 through 11* of the index instruction are added to the addressing register as the instruction address is being incremented.

Condition 2: The address register is incremented with no modification. These conditions are summarized in the following table:

	Condition 1	Condition 2
Type of Test Instruction	Bits 4-11* Modify Address	Address is Not Modified
A - X (Sub)	A < X	A ≥ X
0 - X (COM)	X ≠ 0	X = 0
X - 1 (DEC)	X = 0	X ≠ 0
X + 1 (INC)	X + 1 = 0	X + 1 ≠ 0

* The 8 bit modifier has a sign bit, and a maximum magnitude of ± 128.

00000000	The shortest forward jump calling up the subsequent address
01111111	The largest forward jump junping <u>over</u> 127 addresses
11111111	Illegal Index
11111110	Shortest backward jump calling up the test instruction again
11111101	Jumps back to 1 address prior to the test thus forming a useful loop
10000000	The largest backward jump calling up the 126th address before the test

2.2 CONTROL INSTRUCTIONS;

2.2.1 Status Register Instructions;

These instructions will either change or interrogate the content of the status register bit or bits specified by the code.

EXCHANGE STATUS REGISTER (EXS)

```
 11                      0
|0 0 0|0 0 1|1 1 0|0 0 0|   Octal Code 0160
```

Execution of this instruction replaces the content of the active register by the content of the surrogate register and vice-versa.

CLEAR STATUS REGISTER (CSR)

```
 11                      0
|1 1 0|1 0 0|1 0 0|0 0 0|   Octal Code 6440
```

Execution of this instruction resets all 25 bits of active status register to "0".

SET STATUS "FLAG" (SSB)

```
 11     7                0
|- - -|- - 1|0 0 0|0 0 0|
```

The five bits 7 through 11 represent one of 25 codes specifying which status bit is to be set to a "1". The code table is shown in figure 3. Caution: Not to use, after a test or an interrogate. See, conditional status complement (CSB)

RESET STATUS "FLAG" (RSB)

```
 11     7                0
|- - -|- - 1|1 0 0|0 0 0|
```

The five bits 7 through 11 represent one of 25 codes specifying which status bit is to be reset at a "0". The code table is shown in Figure 3.

STORE MODES SWITCHES (STM)

```
 11     7                0
|1 1 0|1 0 1|1 0 0|0 0 0|   Octal Code 6540
```

This instruction will cause the Timing and Control chip 3806 to accept 8 bits of data presented to the K input of this chip and to store them in bits 17 through 24 of the status register. Previous data in these bits are destroyed. Once stored the bits can be interrogated individually as flags.

INTERROGATE STATUS "FLAG" (ISB)

```
 11      7         0
|- - -|- - 0|0 1 0|0 0 0|
```

This instruction is used to test a particular bit in the active status register for "1" or "0". The five bits 7 through 11 represent one of 25 codes specifying which status bit is to be interrogated (see Figure 3). If the result of the test is TRUE (Bit was "1"), it will be interpreted as condition 1 of a conditional branching (BOC) and will follow the rules for address modification explained in sections 2.1.2.1 and 2.1.2.2 depending on the mode selected. If the result of the test if FALSE (Bit was "o") it will be interpreted as Condition 2 of a BOC and will follow the same rules as above.

CONDITIONAL STATUS COMPLEMENT (CSB)

```
 11      7         0
|- - -|- - 1|0 0 0|0 0 0|
```

This instruction will cause the status bit specified by the five bits 7 through 11 to be complemented (reversed) under the following conditions: The previous instruction was a test (SUB, COM, DEC, INC) or an interrogate (ISB or IPT) and the result was either a condition 1 (see tables BOC section 2.1.2.1 and 2.1.2.2)or TRUE for interrogate instructions.
Exception: COMPL cannot be used, if the previous instruction included T25 in the time enable pattern and the status bit to be complemented was specified as bit 2.

STORE ADDRESS # 1 (SA1) Subroutine entry

```
 11                0
|1 1 0|1 0 1|0 1 0|0 0 0|   Octal Code 6520
```

This instruction causes the current address A in the addressing register to be stored in bits 4 through 12 of the status register. Incrementing of the address takes place just prior to storage so that A + 1 is actually stored.

RECALL ADDRESS #1 (RA1) Subroutine return

```
 11                0
|1 1 0|1 0 0|0 1 0|0 0 0|   Octal Code 6420
```

This instruction recalls the 8 bits that were stored by the SA1 instruction and places them back in the addressing register while it is being incremented so that the address register now contains A + 2 as the next address. Also, the address A + 2 is sent back for storage in the status register to replace the previously stored address.

STORE ADDRESS #2 (SA2) Subroutine entry 2nd level

11			0
1 1 0	1 0 1	0 0 0	0 0 0

Octal Code 6500

This instruction causes the current address A to be stored in bits 17 through 25 of the status register as address, A + 1. Caution: Bits 17-24 are also used to store mode switch data. Attention must be given in programming to insure that the return address is not destroyed by a store mode instruction.

RECALL ADDRESS #2 (RA2) Subroutine return 2nd level

11			0
0 0 0	0 0 1	0 0 0	0 0 0

Octal Code 0100

This instruction recalls the address stored in bits 17 through 25 of the status register, and places them in the addressing register while it is incremented. Address register now contains A + 2. It should be noted that the recall address #2 could be used for another purpose beside subroutine return. If it follows a store mode instruction (STM), the data entered in bits 17 through 24 of the status register will be transferred to the address register, selecting an address in the range of 256 locations. This gives control over the address register to some input device.

2.2.2 Pointer Counter Instructions;

These instructions will either change or interrogate the content of one bit in the pointer counter.

SET POINTER (SPT)

11	7		0
- - -	- - 1	0 1 0	0 0 0

This instruction will cause the pointer counter to be set to one of 25 positions (each of the possible states of the pointer counter is indicating a digit position). The bits 7 through 11 represent one of 25 codes. The code table is shown in Figure 3

POINTER LEFT (PLF)

11			0
0 0 0	0 0 0	1 0 0	0 0 0

Octal Code 0040

The execution of this instruction causes the pointer to move one position to the left (i.e., the pointer counter is incremented by one). Position 1 refers to the right-most digit of a register, position 25 refers to the left-most digit. If a PLF instruction is executed when the pointer is in position 25, it will be moved to position 1.

POINTER RIGHT (PRT)

0 0 0	0 0 1	0 1 0	0 0 0	Octal Code 0120

The execution of this instruction causes the pointer to move one position to the right (i.e., the pointer counter is decremented by one). If a PRT instruction is executed when the pointer is in position 1, it will be moved to position 25.

INTERROGATE POINTER (IPT)

11	7		0
- - -	- - 0	1 0 0	0 0 0

This instruction is used to determine if the pointer is in the position specified by the instruction code. The five bits 7 through 11 represent one of 25 code specifying which digit position is tested for pointer position (see Figure 3). If the result of the test is TRUE (pointer was at position indicated), it will be interpreted as condition 1 of a conditional branching (BOC) and will follow the rules for address modification explained in sections 2.1.2.1 and 2.1.2.2 depending on the mode selected.

If the result of the test is FALSE (no pointer at position indicated), it will be interpreted as condition 2 of a BOC and will follow the same rules as above.

2.2.3 General Control Instructions ;

NO OPERATION (NOP)

0 0 0	0 0 0	0 0 1	0 0 0	Octal Code 0010

	Bits 7-11	← Bits 4-6					→	Bits 0-3
		000	100	110	001	101	010	
NO.	Code	ROM SELECT	SET STATUS	RESET STATUS	INT STATUS	SET POINTER	INT POINTER	
0*	00000	ROM 0	---	---	---	---	---	0000
1	11101	ROM 1	Bit 1	Bit 1	Bit 1	Pos. 1	Pos. 1	
2	00001	ROM 2	Bit 2	Bit 2	Bit 2	Pos. 2	Pos. 2	↑
3	00010	ROM 3	Bit 3	Bit 3	Bit 3	Pos. 3	Pos. 3	
4	00100	ROM 4	Bit 4	Bit 4	Bit 4	Pos. 4	Pos. 4	
5	01001							
6	10010							
7	00101							
8	01011							
9	10110	∘	∘	∘	∘	∘	∘	
10	01100	∘	∘	∘	∘	∘	∘	
11	11001	∘	∘	∘	∘	∘	∘	
12	10011							
13	00111							
14	01111							
15	11111							
16	11110							
17	11100							
18	11000							
19	10001							
20	00011							↓
21	00110							
22	01101							
23	11011							
24	10111							
25	01110	ROM 25	Bit 25	Bit 25	Bit 25	Pos. 25	Pos. 25	

FIGURE 3 Code Conversion Table for Referencing the Pointer, Status Register and ROM's.

(* Applies only to ROM Select.)

ROM SELECT INSTRUCTION (ROM)

```
 11     7           0
|- - -|- - 0|0 0 0|0 0 0|
```

The ROM normally selected upon initial power turn on is ROM 0. The ROM select instruction gives control to the ROM which number is specified by the code represented by the five bits 7 through 11 of the instruction, see code table figure 3 , A ROM becomes selected by this method and remains selected until the next ROM select instruction. The ROM which now has control continues with the subsequent address.

At the initial turn-on the instruction in location 0 can be a NOP to guarantee that.the program starts in location 1.

2.3 Input - Output Instructions

The format of the input - output commands (XIO) is as follows:

```
 11        6 5        0 Bits
|- - -|- - -|1 1 0|0 0 0|
```

Bits 0 to 5 (octal code 60) define the instruction as being an XIO command. Bits 6 through 11 define 1 of 64 possible commands. 63 codes are available for XIO commands (status register exchange use one code: 0160).

A listing of XIO commands for the devices 3807, 3803 and 3811 is given as follows:

2.3.1 XIO Commands For Keyboard Chips:

3807		3803		COMMAND
MNEMONIC	OCTAL CODE	MNEMONIC	OCTAL CODE	
XIO 16	2060	XIO 24	3060	Send Key Address
XIO 32	4060	XIO 0	0060	Send first 8 bits of Mode Switch Data
XIO 40	5060	XIO 8	1060	Send second 8 bits of Mode Switch Data
XIO 48	6060	XIO 48	6060	Set Alarm
XIO 56	7060	XIO 56	7060	Reset Alarm

SEND KEY ADDRESS: (XIO 16)

This command causes the 3807 to respond, causing the 3806 to increment the next ROM address by the amount by the key which has been depressed (if any). If no key has been depressed, the ROM address will be incremented by one.

SEND KEY ADDRESS: (XIO 24)

This command has the same effect as above except the 3803 responds rather than the 3807.

SEND 1ST 8 BITS OF MODE SWITCH DATA: (XIO 32)

This command causes the 3807 to present the status of the mode switches connected to D3 to the 3806 for storage in the status register. To store this data in 3806, XIO 32 must be followed by STORE MODES command.

SEND 1ST 8 BITS OF MODE SWITCH DATA: (XIO 0)

This command has the same effect as above except the 3803 responds rather than the 3807.

SEND 2ND 8 BITS OF MODE SWITCH DATA: (XIO 40)

This command causes 3807 to present the status of the mode switches connected to D2 to the 3806 for storage in the status register. To store this data in 3806 XIO 40 must be followed by STORE MODES command.

SEND 2ND 8 BITS OF MODE SWITCH DATA: (XIO 8)

This command has the same effect as above except 3803 responds rather than 3807.

SET ALARM (XIO 48)

This command causes 3807 and 3803 to be set to the alarm state.

RESET ALARM: (XIO 56)

This command causes 3807 and 3803 to be reset from the alarm state.

2.3.2 XIO Commands For Output Chip (3811)

MNEMONIC	OCTAL CODE	COMMAND
XIO 3	0360	Memory On
XIO 5	0560	Memory Off
XIO 7	0760	Learn Sign
XIO 9	1160	Learn D.P. Position
XIO 17	2160	Display Disable
XIO 33	4160	Display Enable
XIO 61	7560	Blank Enable
XIO 63	7760	Blank Disable

MEMORY ON: (XIO 3)

This command causes the M output (3811) to be set

MEMORY OFF: (XIO 5)

This command causes the M output (3811) to be reset.

LEARN SIGN: (XIO 7)

This command causes Y0 to be sampled and stored by 3811 on the following frame during time enable. The normal convention is Y0 = 0 +, Y0 = 1-.

LEARN DP POSITION (XIO 9)

This command causes the following time enable pattern to be stored on 3811. This stored time enable pattern defines the position(s) the decimal point(s).

DISPLAY DISABLE: (XIO 17)

This command disables (forcesLOW) all outputs of the 3811 except M which is not effected. It also prevents 3811 from accepting data.

DISPLAY ENABLE: (XIO 33)

This command readies the 3811 to accept data.

BLANK ENABLE: (XIO 61)

This command causes the K0, K1, K2, K3, to be set HIGH (VSS) and $\overline{\text{BLK}}$, DP to be set LOW It does not disable any inputs of 3811.

BLANK DISABLE: (XIO 63)

This command causes $\overline{\text{BLK}}$ to be set HIGH and allows K0, K1, K2, K3 and DP to resume their programmed values based on the accepted data and accepted decimal point information.

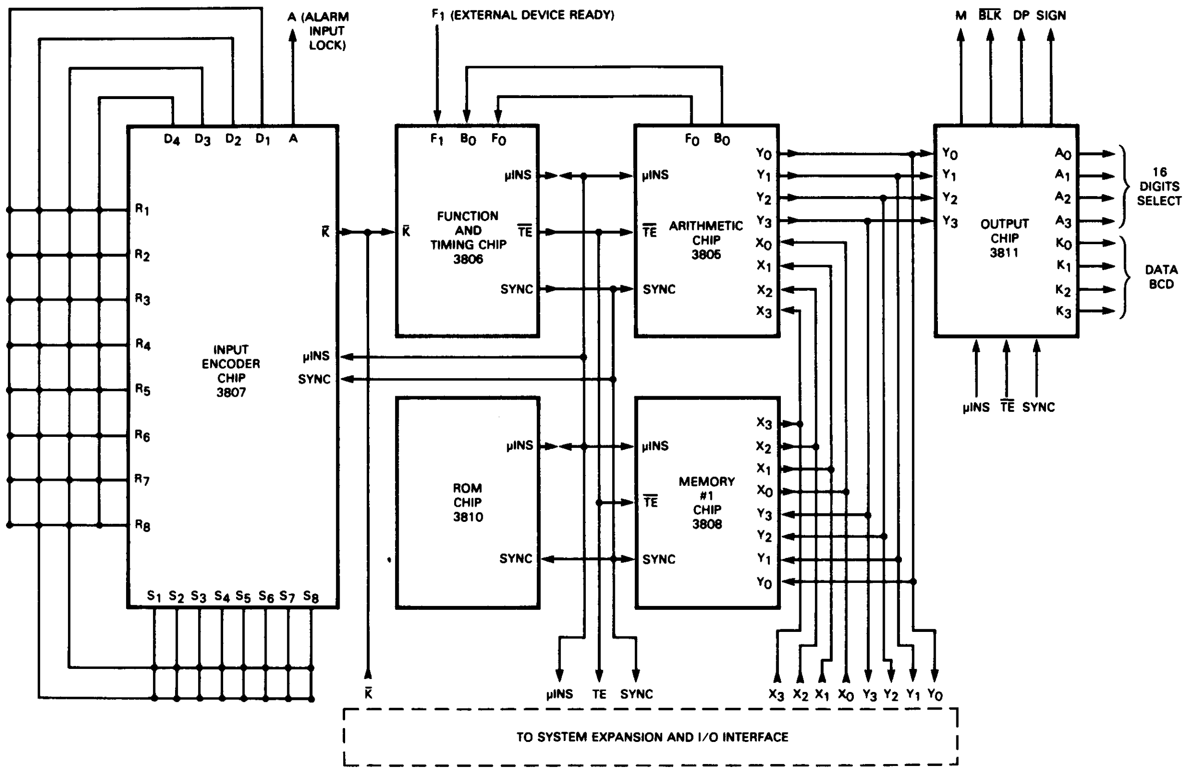


FIGURE 4: BLOCK DIAGRAM OF A TYPICAL (6 CHIPS) PPS 25 SYSTEM

C THE PPS 25 SYSTEM TIMING:

A typical system consisting of 6 chip is illustrated by figure 4.

The communication lines between the 6 chips: Micro Instruction, Time Enable, SYNC , X(0,1,2,3), Y(0,1.2,3) and K are all referenced for their operation to the timing of one cycle of the ALU as follows:

The system timing is organized for a 25 clocks cycle (see figure 5). One clock period T contains 1 cycle of Ø1 and 1 cycle of Ø2; at 400 kHz one clock period is equal to 2.5 microseconds and one ALU cycle is equal to 62.5 microseconds. All interchip signals are synchronous and change state following the 0 to 1 transition of Ø2 (1 is defined as Active LOW negative logic). A"T"time also begins following a 0 to 1 transition of Ø2. There are 25 T times labelled T_1 to T_{25}

1. SYNC Line Timing:

Every ALU cycle, the SYNC line makes a 1 to 0 transition at T11 and a 0 to 1 transition at T25.

2. Micro-Instruction Timing:

The Micro-Instruction line is bi-directionnal and its timing is as follows:
Time:

T4-T11	8 bits ROM address is transmitted by function and timing chip (3806)
T12	Additional ROM address bit provided for expansion of ROM address to 9 bits (512 words)
T14-T25	12 bits instruction is transmitted by the selected ROM (3810)
T1,T2,T3&T	Not used

3. Time Enable Timing:

The TE line determines the field(s) of data to be manipulated. T14-T16 of the previous instruction cause $\overline{TE}$ to be set in the range of T1 to T25 of the next instruction. $\overline{TE}$ is Active HIGH (VSS=1).

4. $\overline{K}$ Line Timing:

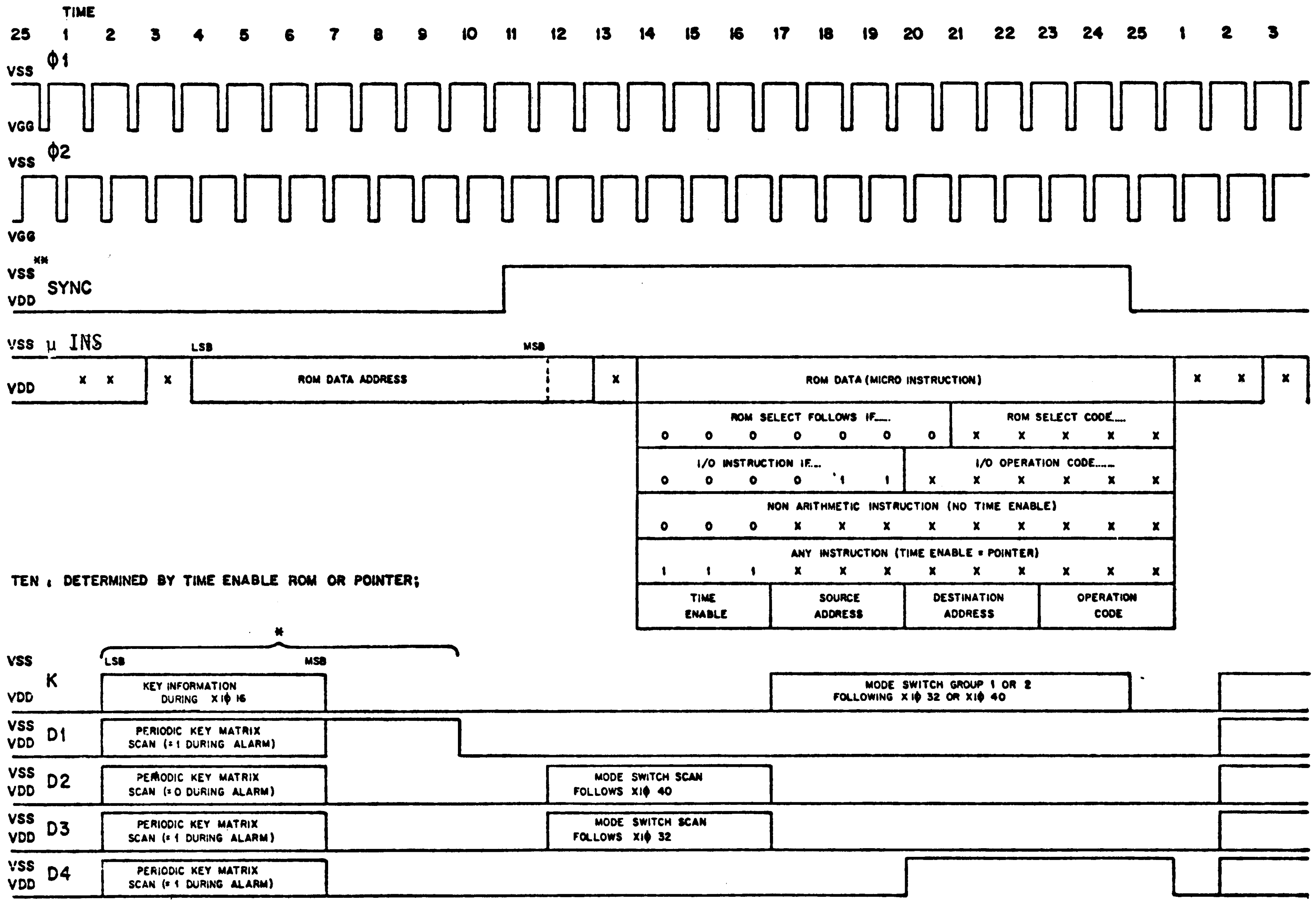
Any information on the $\overline{K}$ line during times T2-T9 will be added to the ROM address during the same ALU cycle at T4-T11. Execution of a store mode command causes the content of the $\overline{K}$ line at times T17-T24 to be stored in the active status register. $\overline{K}$ is Active HIGH (VSS=1).

5. Data Bus Lines Timing:

   Y0,Y1,Y2,Y3; X0,X1,X2,X3 carry all data transfers. A data transfer spans T1-T25 on the X0-X3 buss is delayed 1T time on the Y0-Y3 buss to T2-T1. The instruction causing a given data transfer was transmitted during T14-T25 of the previous ALU cycle.

6. Key Switch Matrix and Mode Switch Timing:

   The key switch matrix is scanned by the combination of the lines D1,D2,D3,D4 and R1,R2,R3,R4,R5,R6,R7,R8. Every ALU cycle (25 clock times) a new intersection in the matrix is scanned. The actual scanning takes place at the times T2-T6. The mode switch matrix is scanned by the combination of the lines D2,D3 and S1,S2, S3, S4, S5, S6,S7, S8. The mode switch group #1 is controlled by line D3 and scanned at times T12-T16. The mode switch group #2 is controlled by line D2 and scanned at times T12-T16.


TIME
25 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 1 2 3
VSS ϕ1 VGG
VSS ϕ2 VGG
VSS ** SYNC VDD
VSS μ INS VDD
X X X
LSB ROM DATA ADDRESS MSB
X
ROM DATA (MICRO INSTRUCTION)
X X X
ROM SELECT FOLLOWS IF.... 0 0 0 0 0 0 0
ROM SELECT CODE.... X X X X X
I/O INSTRUCTION IF... 0 0 0 0 1 1
I/O OPERATION CODE..... X X X X X X
NON ARITHMETIC INSTRUCTION (NO TIME ENABLE) 0 0 0 X X X X X X X X X
ANY INSTRUCTION (TIME ENABLE = POINTER) 1 1 1 X X X X X X X X X
TIME ENABLE
SOURCE ADDRESS
DESTINATION ADDRESS
OPERATION CODE
TEN : DETERMINED BY TIME ENABLE ROM OR POINTER;
*
VSS K VDD
LSB MSB
KEY INFORMATION DURING X1ϕ 16
MODE SWITCH GROUP 1 OR 2 FOLLOWING X1ϕ 32 OR X1ϕ 40
VSS D1 VDD
PERIODIC KEY MATRIX SCAN (=1 DURING ALARM)
VSS D2 VDD
PERIODIC KEY MATRIX SCAN (=0 DURING ALARM)
MODE SWITCH SCAN FOLLOWS X1ϕ 40
VSS D3 VDD
PERIODIC KEY MATRIX SCAN (=1 DURING ALARM)
MODE SWITCH SCAN FOLLOWS X1ϕ 32
VSS D4 VDD
PERIODIC KEY MATRIX SCAN (=1 DURING ALARM)

* THE 1's COMPLEMENT OF THIS VALUE IS ADDED TO THE PROGRAM REGISTER. THE SUM APPEARS AS THE ROM DATA ADDRESS.

** VSS = 0, VDD = 1, EITHER VALUE = X

FIGURE 5.0 SYSTEM TIMING PPS 25

D. DESCRIPTION OF PPS 25 ASSEMBLER (PROCASM)

1.0 INTRODUCTION

The assembler software is a 2-pass absolute symbolic assembler for the instruction set of the Fairchild PPS 25 Building Blocks. It is written in USA Standard FORTRAN. Input consists of a title or deck of symbolic statements, and output is a listing of the assembled codes and their corresponding locations, and one or more files of data to be processed by other components of the Fairchild CAD Software System (e.g., assembled output can be directed to FAIRSIM, ROM, etc.).

2.0 LANGUAGE

The assembler language is similar in form to the FAIRSIM and FAIRGEN network description languages. Each statement is entered on a single card (or card image in a file), and may be coded in a free format; that is, fields within a statement do not have to start at any special column location (except for the LABEL field). Each statement has four fields - LABEL, OPERATION, OPERAND, AND COMMENTS.

Character Set:

The characters which may be used in coding assembler statements consist of the following: Alphabetics (A - Z and $ and &); Numerics (0 - 9); Special Characters (blank, comma, plus sign, minus sign, and asterisk).

2.1 LABEL FIELD

A LABEL, if present, must begin in column one of the card, otherwise, column one must be blank. A LABEL may consist of up to four characters, the first of which must be one of the alphabetics. The remaining characters may be alphabetics or numerics.

2.2 OPERATION FIELD

The OPERATION field follows the LABEL field and must be separated from it by at least one blank space. The OPERATION field must contain one of the PPS 25 operation mnemonics (i.e., ADD, DEC BRU, etc.) or one of the assembler instructions listed in Section 4.0.

2.3 OPERAND FIELD

The OPERAND field follows the OPERATION field and must be separated from it by at least one blank space. The OPERATION field may contain one or more expressions, separated by commas. The OPERAND field may not contain imbedded blanks. An OPERAND expression consists of one or more terms connected by the operators plus (+) signifying addition, and minus (-) signifying subtraction. Each term may be either a LABEL, a decimal number, or an asterisk which signifies the value of the current address of the instruction.

2.4 COMMENTS FIELD

After the OPERAND field, and separated from it by at least one blank space, the programmer may enter comments. The COMMENTS field is ignored by the assembler.

3.0 INSTRUCTION MNEMONICS

3.1 Register Reference Type

X = Source Register

Y = Destination Register

T = Time Enable Code

ADD	Y, X, T	A(T) + X(T) → Y(T)
SUB	Y, X, T	A(T) - X(T) → Y(T)
COM	Y, X, T	0 - X(T) → Y(T)
DEC	Y, X, T	X(T) - 1 → Y(T)
INC	Y, X, T	X(T) + 1 → Y(T)
MOV	Y, X, T	A(T) → Y(T); X(T) → A(T)

RSH	Y, X, T	RIGHT SHIFT X(T) → Y(T)
LSH	Y, X, T	LEFT SHIFT X(T) → Y(T)
CLR	X, T	0 → X(T)
SLI	Z, T	LEFT SHIFT A(T) + Z → A(T)
	Z	Represents one data character its value being defined in hexadecimal representation (0 to 9 and A to F)

3.2 Index Type

BR3	Selects 3-way conditional branching
(ADD1, ADD2	Designate Symbolic Addresses)
BRU ADD1	Branch unconditionally to ADD1.
BOC ADD1, ADD2	Branch to ADD1 if Condition 1 was satisfied by the preceding test instruction. Branch to ADD2 if Condition 2 was satisfied. If neither condition was satisfied, execute the next instruction.
NOP	No operation.

Type of Interrogate or Test Instruction	Branch to ADD1 IF	Branch to ADD2 IF	Execute Next Instruction IF
A - X	$A < X$	$A > X$	$A = X$
0 - X	$X \neq 0$	----	$X = 0$
X - 1	$X = 0$	Otherwise	$X = 1$
X + 1	$X + 1 = 0$	$X + 1 \neq 0$	-----
ISB NN	Bit NN is a "1"	Bit NN is a "0"	-----
IPT NN	Pointer _is_ at NN	Pointer is _not_ at NN	-----

3-WAY CONDITIONAL BRANCH TABLE

BR2	Selects 2-way conditional branching
BOC ADD1	Branch to ADD1 if Condition 1 was satisfied by the preceeding test instruction. Otherwise execute next instruction.

Type of Interrogate or Test Instruction	Branch to ADD1 IF	Execute Next Instruction IF
A - X	$A < X$	$A \geq X$
0 - X	$X \neq 0$	$X = 0$
X - 1	$X = 0$	$X \neq 0$
X + 1	$X + 1 = 0$	$X + 1 \neq 0$
ISB NN	Bit NN is a "1"	Bit NN is a "0"
IPT NN	Pointer <u>is</u> at NN	Pointer <u>is not</u> at NN

2-WAY CONDITIONAL BRANCH TABLE

3.3 Status Register Reference Type

SSB	NN	SET STATUS BIT NN
RSB	NN	RESET STATUS BIT NN
ISB	NN	INTERROGATE STATUS BIT NN
CSB	NN	CONDITIONAL STATUS COMPLIMENT BIT NN
CSR		CLEAR STATUS REGISTER
EXS		EXCHANGE STATUS REGISTER
STM		STORE MODES

3.4 Pointer Reference Type

SPT	NN	SET POINTER TO POSITION NN
IPT	NN	INTERROGATE POINTER
PLF		MOVE POINTER LEFT
PRT		MOVE POINTER RIGHT

3.5 Subroutine Call Type

SA1	STORE ADDRESS NUMBER 1
RA1	RECALL ADDRESS NUMBER 1
SA2	STORE ADDRESS NUMBER 2
RA2	RECALL ADDRESS NUMBER 2

3.6 ROM Select

ROM	NN	SELECTION ROM NN

3.7 I/O Control

XIO	NN	OUTPUT I/O INSTRUCTION NN

4.0 ASSEMBLER INSTRUCTIONS

In addition to the mnemonic codes which are the instruction set of the PPS 25 building blocks, the assembler recognizes and implements the following assembler instructions:

HED - Set Heading

The contents of the card following the OPERATION code through Column 72 are copied and used as a heading on the output listing. In addition, a skip to the top of a page is executed.

SPC - Space Listing

The expression in the OPERAND field, if present, is evaluated and the output listing is spaced the corresponding number of lines.

SKP - Skip to Top of Page

The listing is immediately skipped to the top of a new page.

ORG - Set Origin

The expression in the OPERAND field is evaluated and used as the location counter value. A LABEL, if present, is assigned the value of the new location.

5.0 Example of a listing of SOURCE STATEMENTS, ASSEMBLED CODES and their corresponding locations.

```
LOC.    CODE                    SOURCE STATEMENT

        0001     ES             EQU 1
        0005     T              EQU 5
        0001     F              EQU 1
        0004     SY             EQU 4
        0000     A              EQU 0
        0004     B              EQU 4
        0002     C              EQU 2
        0007     H              EQU 7
        0007     PTR            EQU 7
0000    0010                    NOP
0001    6101                    CLR 1,1,F
0002    2101                    MOV 1,0,F
0003    7200                    ROM 1
0004    0010                    NOP
0005    0010                    NOP
0006    0010                    NOP
0007    0010                    NOP
0010    0010                    NOP
0011    0010                    NOP
0012    0010                    NOP
0013    0010                    NOP
0014    6520                    SA1
0015    3430                    BRU BR1
0016    0010                    NOP
0017    7620                    ISB 15
0020    0030                    BOC B3,B4
0021    7740     B3             RSB 15
0022    5060     B4             XIO 40
0023    6540                    STP
0024    6440                    CSR
0025    3211                    COM 2,1,F
0026    0211                    ADD 2,1,F
0027    1211                    SUB 2,1,F
0030    5227                    DEC 2,2,PTR
0031    1240                    IPT
0032    0030                    BOC B5,B6
0033    6223     B5             LSH  2,2,3
0034    7223     B6             RSH  2,2,3
0035    0120                    PRT
0036    0010                    NOP
0037    0060                    XIO 0
0040    7060                    XIO 56
0041    2300                    SSB 5
0042    5720                    SPT 24
0043    4007                    INC 0,0,PTR
0044    0260                    XIO 2
0045    0360                    XIO 3
0046    0460                    XIO 4
0047    5460                    XIO 44
0050    0560                    XIO 5
```

FAIRCHILD
SEMICONDUCTOR