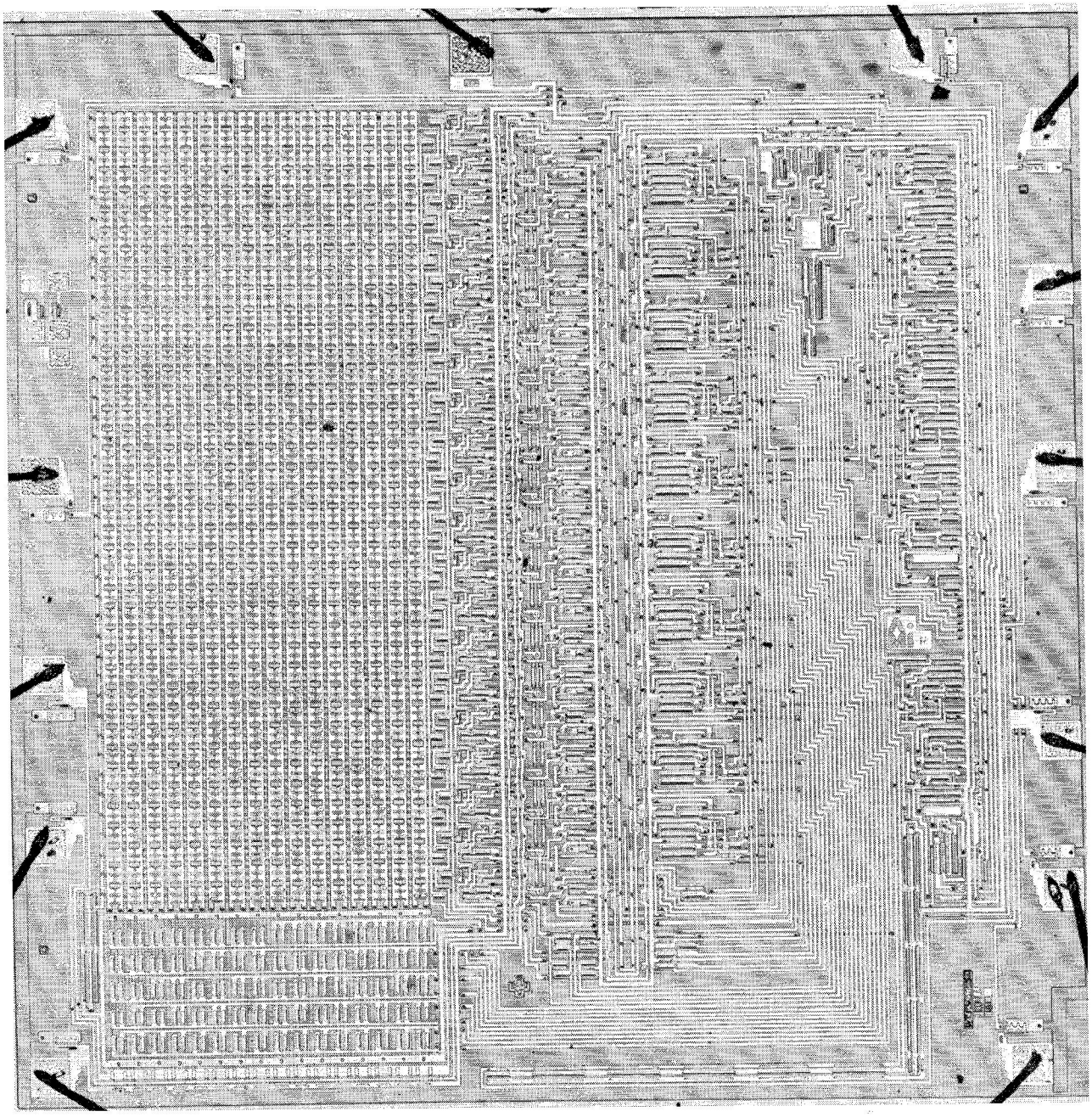


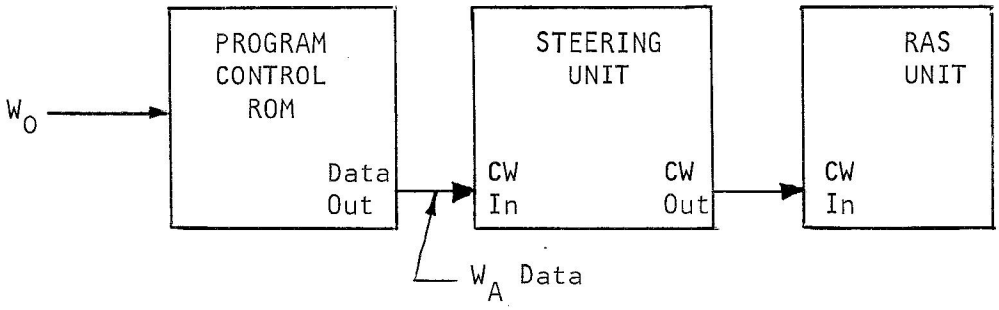
F-13228

Figure 1-20. Read-Only Memory PN 944125, 3268 Devices

Module 1	Module 2	Module 3
1 -- PMU	1 -- PDU	1 -- SLF
1 -- SL	1 -- SL	1 -- SL
1 -- RAS	1 -- RAS	1 -- RAS
3 -- Data ROM's	1 -- Data ROM	2 -- Data ROM's
3 -- Program Control ROM's	3 -- Program Control ROM's	3 -- Program Control ROM's

Each arithmetic unit has an associated steering unit and RAS unit. This is not an absolutely necessary association but was chosen to add flexibility to the programming. Figures 1-22 and 1-23 illustrate two other possibilities of circuit arrangements. It is important to note that the hardware arrangement is flexible to the given problem statement. If many multiplies are needed, then two or more PMU's could be used instead of one. The same can be true for any of the circuits.

The ROM's are used in two different modes: the data ROM's access data during the W_0 cycle, and the control ROM's access data during the W_A cycle. The difference being determined by which timing signal W_A (for W_0 accessed ROM's) or W_0 (for W_A accessed ROM's) is connected to the increment input to the ROM's. The control word path for all three modules is connected as shown below.



The first five bits (LSB) for the 20-bit control word go to the RAS, and the last 15 bits are left in the SL. The SLF unit accepts the same four bits of the control word that selects its RAS registers. The breakdown of the control words is as follows:

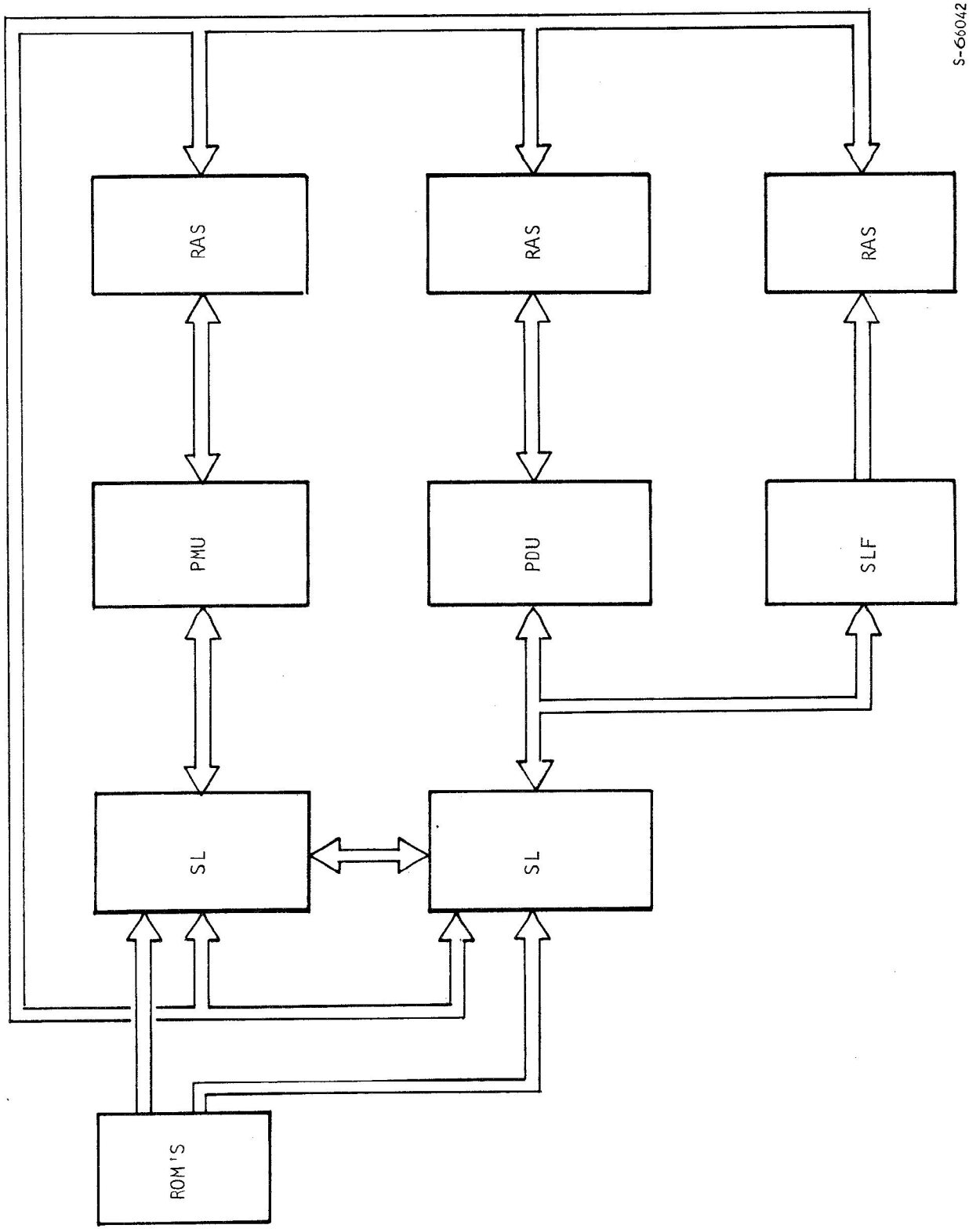


Figure 1-22. Circuit Arrangement 1

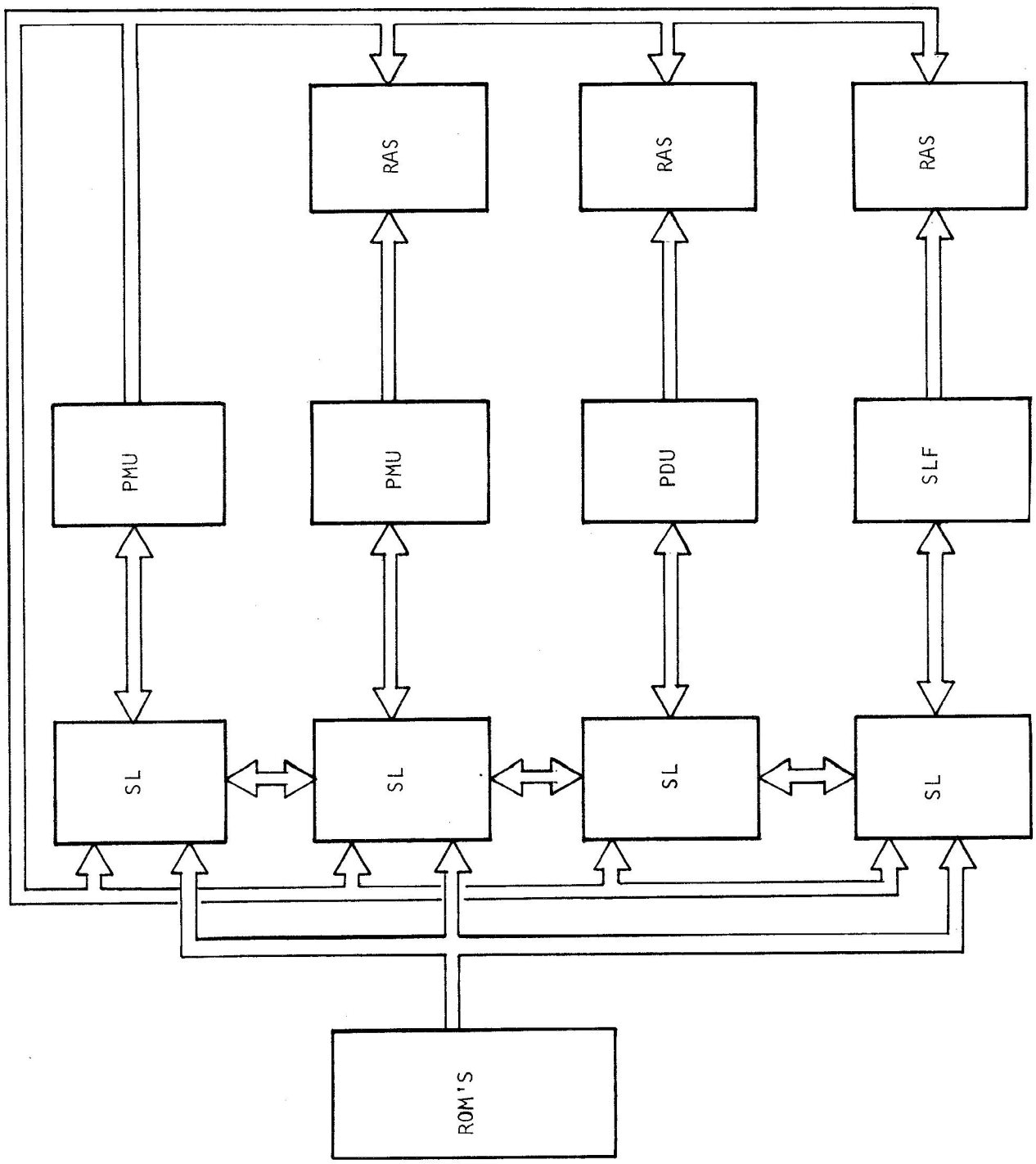
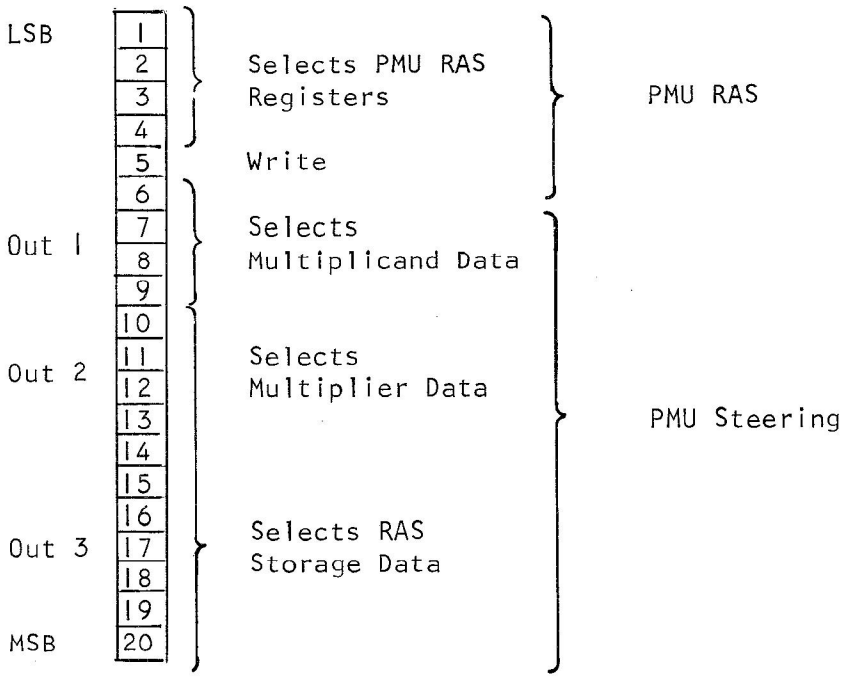
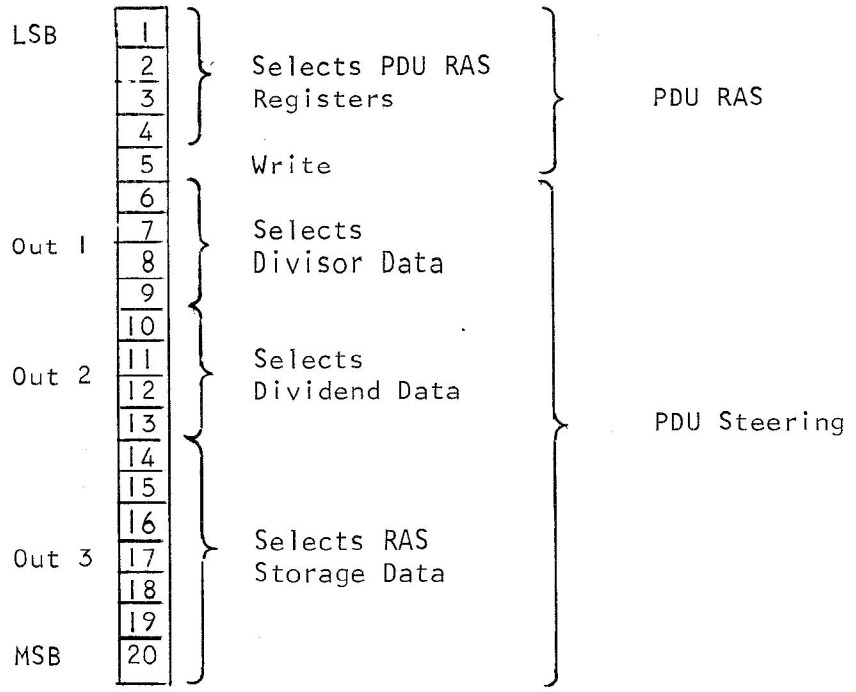


Figure 1-23. Circuit Arrangement 2

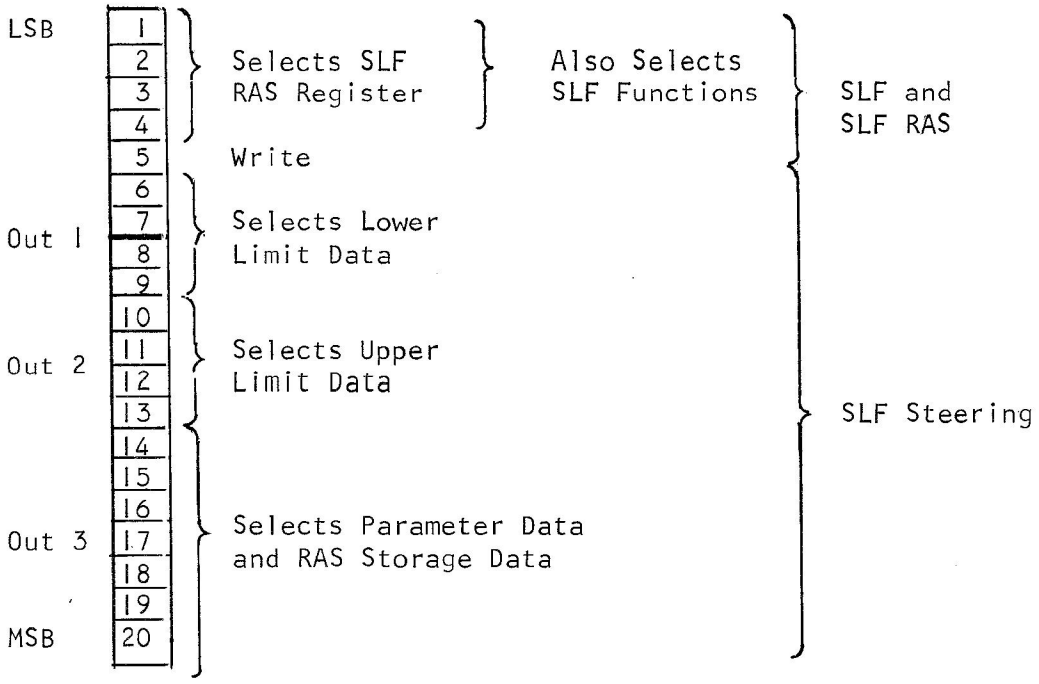
Module 1 Control Word



Module 2 Control Word

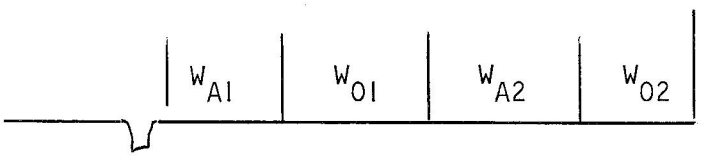


Module 3 Control Word



Data storage ROM's can be treated like the outputs from the other circuits. They are connected to the inputs of the steering units. The total number of ROM's used in each group is a function of the total amount of storage required and the update rate of the system. The number of arithmetic units is a function of the amount of calculation and the update rate of the system. As shown, the hardware is flexible in two directions: computational load and update rate. This flexibility is required to handle the various types of systems to be implemented. Figure 1-24 illustrates one method of a flexible hardware arrangement. To illustrate all hardware arrangements is not usefully possible. Each unique arrangement depends upon the programming requirements.

Since we have discussed the interconnection of the hardware circuits, a few computer operations can be discussed and how the circuits function together described. The following operation sequence will be used in the discussion.



FRAME MARK — BEGINNING OF COMPUTATIONAL CYCLE (FRAME)

The frame mark signal will set all ROM's to their initial address.

During W_{A1}, the three groups of control word ROM's will simultaneously shift out a 20-bit control word. These control words will select the necessary data paths to transfer the appropriate data to become M, R, D, Z, U, P, L, or data inputs to the RAS's. The appropriate RAS register will be selected simultaneously.

During W_{01}, all of the arithmetic, RAS, and data ROM outputs are transferring data to the inputs of the steering unit. Since the control words have already selected the appropriate data paths, the data will continue through the steering and back to the inputs of the arithmetic and/or RAS units.

During W_{A2}, the second set of control words are transferred to the steering, RAS, SLF units, and the arithmetic units are performing arithmetic operations on the data transferred to them during W_{01}.

During W_{02}, the outputs, whether it be computed arithmetic answers (product, quotient, limit, etc.) or stored RAS data, are again transferred to the steering units.

A complete cycle of following data through the processor circuits has now been accomplished. This cycle is repeated 512 times in the F-14A CADC and constitutes one frame. System programming must then be discussed if meaningful data are to be applied to the data that are being transferred around and through the circuits. Section 2 will cover the software programming of the processor.

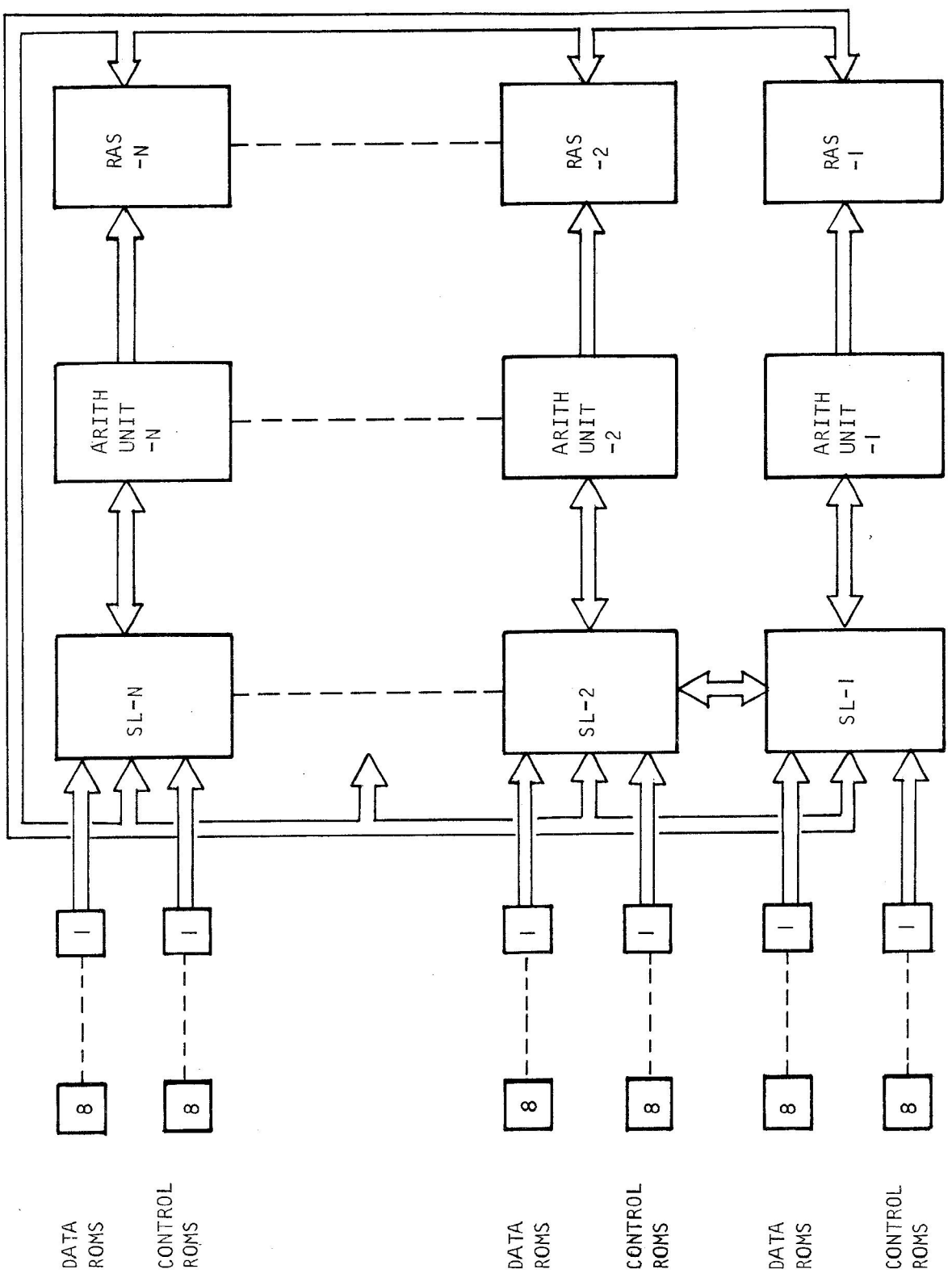


Figure I-24. General Circuit Arrangement

SECTION 2

SOFTWARE PROGRAMMING

The software for the F-14A CADC essentially consists of knowing how to organize and code the various control words to accomplish the appropriate data transfers. To accomplish this, the desired mathematical equation should first be converted to a hardware oriented mathematical model, and subsequently transformed to a time-hardware diagram. At this stage, the control words are ready to be coded. After coding has been completed, simulation of the entire loop verifies the desired results. This section will discuss in detail the above procedure and how to use it. Figure 2-1 shows the flow diagram of this procedure.

MATHEMATICAL MODEL AND SCALING CONSIDERATIONS

The system problem statement is defined in terms of mathematical equations. An example of this is as the following polynomial:

Input Y, Output E

$$E = \sum_{i=0}^{3} a_i X^i = X^3 a_3 + X^2 a_2 + X^1 a_1 + a_0 = \text{3rd order polynomial}$$

where,

$$X = \frac{Y\Big|_{LI}^{L2} + B}{C}$$

This represents a third order polynomial whose independent variable is conditioned (B), expanded (C), and limited (LI, L2). This example will be used throughout the entire software procedure.

The math model should be appropriately scaled such that no calculations will exceed $\pm (1-2^{-19})$. These are the maximum positive and negative values obtainable in the processor. A few rules that should be followed when doing the scaling are as follows:

(a) When adding or subtracting, both numbers must be scaled to the same number. The sum or difference will then be scaled to this same number.

Example:

$$\frac{0.65}{A} + \frac{0.2}{A} = \frac{0.85}{A}$$

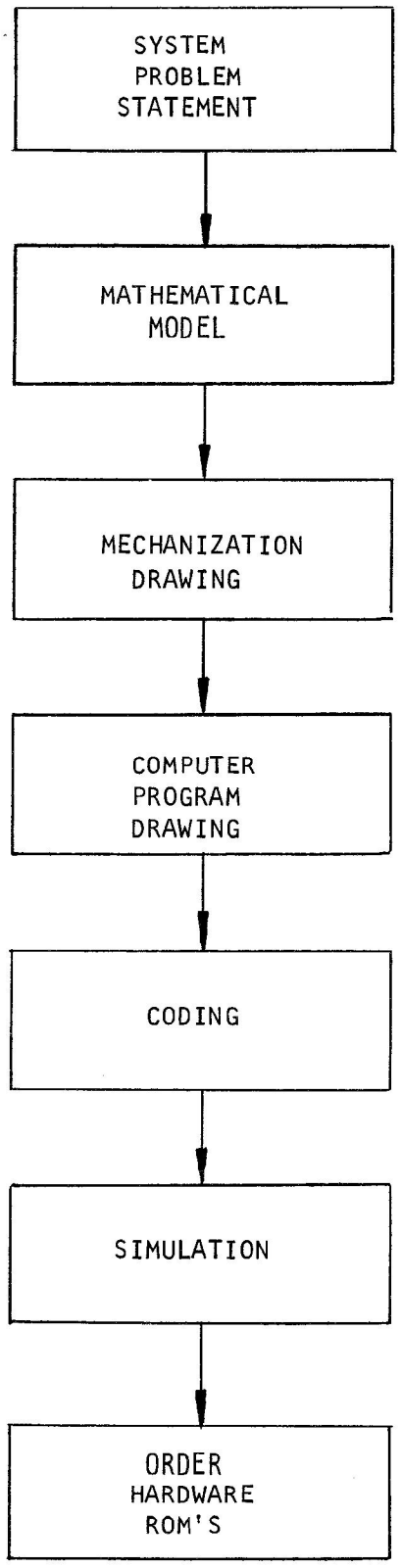


Figure 2-1. Software Flow Diagram

$$\frac{0.65}{A} - \frac{0.2}{A} = \frac{0.45}{A}$$

where

A = scaling factor

Care must be taken such that the sum does not exceed $+(1-2^{-19})$ or the difference exceeds $-(1-2^{-19})$. If this condition exists, it is called overflow.

(b) When multiplying two numbers, the scaling of the product will be the product of the input scalings.

Example:

$$\frac{0.65}{A} \times \frac{0.2}{B} = \frac{0.13}{AB}$$

(c) When dividing two numbers, the scaling of the quotient will be the quotient of the input scalings.

Example:

$$\frac{0.2}{A} \div \frac{0.8}{B} = \frac{0.25}{A/B}$$

Care must be taken such that the scaled dividend $\left(\frac{0.2}{A}\right)$ is always greater than or equal to the scaled divisor $\left(\frac{0.8}{B}\right)$. If the dividend and divisor are equal in magnitude, then the divide algorithm will produce a maximum positive or maximum negative quotient.

$$\frac{+A}{+A} = 1-2^{-19} = \max + \qquad \frac{+A}{-A} = -(1-2^{-19}) = \max -$$

Now, take the original example and apply some scaling factors to it.

Suppose,

Output $0 \leq E \leq 10$

Input $0 \leq Y \leq 30$

Then the scaled math model might look like this.

$$\frac{E}{15} = \sum_{i=0}^{3} \frac{a_i}{15} \left(\frac{X}{18}\right)^i$$

where,

$$\frac{X}{18} = \frac{\begin{array}{c|c} \frac{Y}{36} & \begin{array}{c}\frac{L2}{36} \\ \frac{L1}{36}\end{array} \end{array} + \frac{B}{36}}{\frac{C}{2}}$$

There are cases where the scaling might require addition of arithmetic operations to the original math definitions. The following example will show this.

Suppose,

$$\left(\frac{W}{2}\right)\left(\frac{Y}{10}\right) + \frac{X}{4}$$

If the above equation is used as is, then

$$\frac{WY}{20} + \frac{X}{4}$$

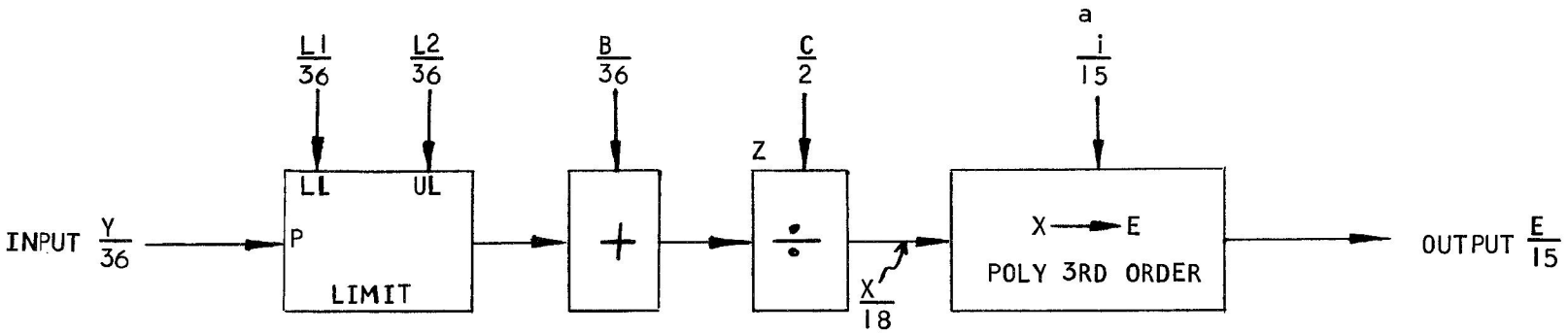
This would result in the invalid addition of two numbers with different scaling factors. A divide operation to rescale $\frac{WY}{20}$ would have to be added as shown.

$$\frac{\frac{WY}{20}}{\frac{4}{20}} + \frac{X}{4} = \frac{WY}{4} + \frac{X}{4} = \frac{WY + X}{4}$$

It must be remembered that all arithmetic results must always be less than 1.

MECHANIZATION DRAWING

After the math model and scaling is complete, the next step will be to make a mechanization drawing. This drawing transforms the mathematics one step closer to the hardware. The mechanization drawing is a step-by-step layout of the math model with blocks that represent hardware. The priority of processing the calculations or the intermediate storage of data is not shown on this drawing. Figure 2-2 shows the original example transformed to a mechanization drawing.



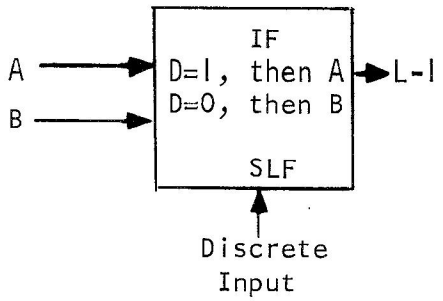
Each block represents a function that the system can perform.

S-66046

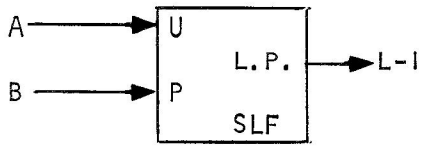
Figure 2-2. Mechanization Drawing, Example

Other blocks that may be used are:

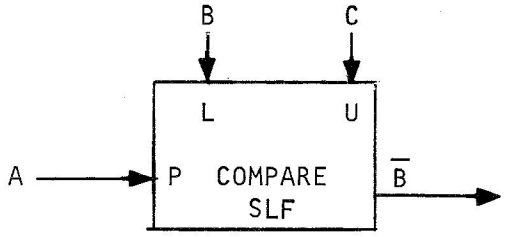
1. "IF" transfer based on discrete input to SLF



2. Logical product



3. Comparison of A and B for discrete output



Each SLF combination (described in Section 1, SLF) can have a block that represents its function.

COMPUTER PROGRAM DRAWING

The computer program drawing transforms the mechanization drawing into a time dependent flow of calculations. All data transfers are shown with respect to each other. The use of each arithmetic unit at the various OP times are shown. Each arithmetic module (Figure 2-3) is shown as a row of 512 blocks, each block representing one OP time. The PMU module is on top with the PDU module in the middle and the SLF module at the bottom. Under each row of blocks are positions for 16 horizontal lines, each line representing one RAS register. As data is stored from operation to operation, it is shown as a horizontal line extending from the operation that the data is written to the operation that the data is read. ROM data constants are shown entering

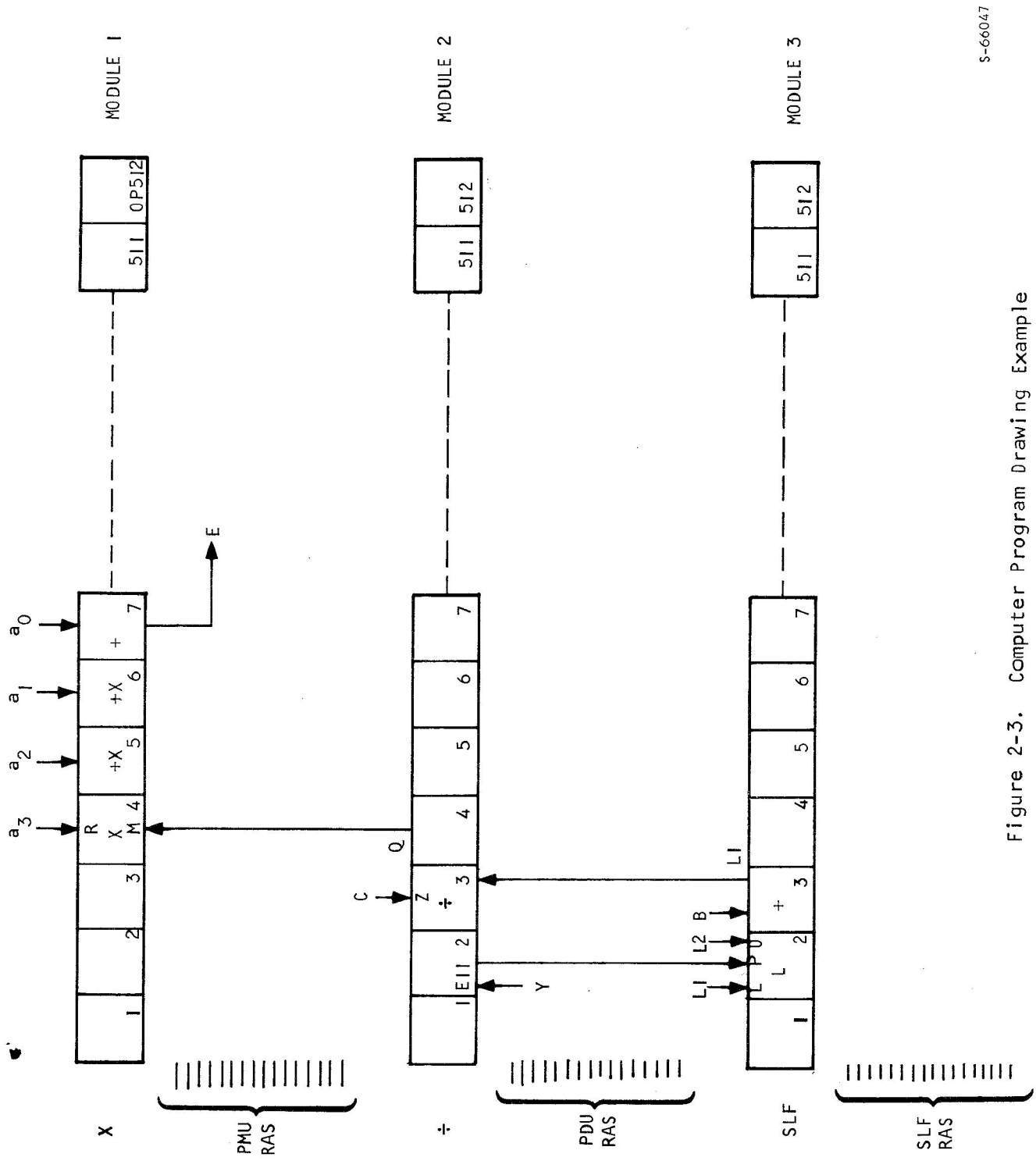


Figure 2-3. Computer Program Drawing Example

the blocks from the top. As each constant is accessed from the ROM, an arrow is drawn. Figure 2-3 shows the original example as it would be programmed on the program drawing. This program layout is not unique and can change when incorporated with other parts of the software program. The following describes the actions for each operation.

Operation 2

(1) Input Y is transferred through PDU steering and SLF steering to the P_{input} of the SLF.

(2) Upper and lower limits are transferred from ROM's through the SLF steering to the SLF.

Operation 3

(1) The limited result appears on SLF Line 1 output and is transferred through the SLF steering (E12 + E8) to form a sum with B.

(2) B is transferred from a ROM to the SLF steering to form a sum with the limited result (Line 1).

(3) The sum is then transferred through the PDU steering to the dividend.

(4) The divisor is transferred from a ROM through the PDU steering to the PDU.

Operation 4

(1) Proper quotient appears on the PDU output and is transferred through the PMU steering to the multiplicand.

(2) The a_3 constant for the polynomial is transferred from a ROM through the PMU steering to the PMU.

Operation 5

(1) Product is transferred through PMU steering and is summed with the a_2 constant.

(2) This sum appears on PMU steering output 3 and is transferred through PMU steering output 1 to the multiplier.

(3) The multiplicand is transferred out of the PMU through the PMU steering and back to the PMU multiplicand.

Operation 6

(1) This is the same as Operation 5 except with the a_1 constant from the ROM.

Operation 7

(1) Product is transferred through PMU steering and is summed with the a_0 constant.

(2) This sum appears on PMU steering output 3 and is written into a PMU RAS register. This is the result E.

Additional examples from actual F-14A CADC program are presented in the Programming Examples subsection. These examples will illustrate the different functions and methods of programming. Section 3 (Computer Aids) will discuss and show the computer program drawing, which has been drawn by a CalComp plotter.

CODING

After the complete problem statement has been laid out as a computer program drawing, the next step is to code the program into instructions or control words. The coding is presently performed in what is classically called machine language. This is coding in binary that is directly usable by the machine. Coding is performed OP by OP for each module. The code words were previously defined under SLF (Section 1). Figure 2-4 (a through g) shows the example problem coded onto prepared forms.

EXECUTIVE CONTROL

The executive control is used to control various parts of the F-14A CADC system. In general, it is used to control the inputs (A/D, discretes), the outputs (D/A, digital data, discrete), and the processor.

The executive control consists of an associated group of ROM's, supplying a 20-bit control word every operation time. Each bit has a preassigned task based upon the hardware configuration. The assignment for each of the bits is as follows:

MSG	20	Discrete Input Enable
	19	SLF Control Enable
	18	SLF Data-2 Enable
	17	SLF Data-1 Enable
	16	PDU Control Enable
	15	PDU Data Enable
	14	PMU Control Enable
	13	PMU Data Enable
	12	Built-in Tests (All Except Qc)
	11	A/DC Shift Out (A/DC Output Register)
	10	Update Input (A/DC MUX) (16)

PMU
STEERING CODING FORM

Name_____

Date_____

OP	STORAGE TO SELECTED				WRITE	M SELECTED TO OUTPUT 1				R SELECTED TO OUTPUT 2				SELECTED TO OUTPUT 3				Σ_1	Σ_2		
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	COMMENTS
1																					
2																					
3						0	0	1	0	0	1	1	1								Q+0 → M ROM → R
4						1	0	1	0	0	0	0	0	1	0	0	0	1			M → M 12 + 8 → R
5						1	0	1	0	0	0	0	0	1	0	0	0	1			M → M 12 + 8 → R
6						1	0	1	0	0	0	0	0	1	0	0	0	1			M → M 12 + 8 → R
7	0	1	0	0	1									1	0	0	0	0			12 + 8 → RAS

Figure 2-4(a). Coding of Example Problem

PDU
STEERING CODING FORM

Name_____

Date_____

OP	STORAGE TO SELECTED				WRITE	Z SELECTED TO OUTPUT 1				D SELECTED TO OUTPUT 2				SELECTED TO OUTPUT 3				Σ_1	Σ_2	COMMENTS
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	
1																					
2														1	0	1	0				Y → SLF
3						0	1	1	1	0	1	0	0								ROM → Z LI → D
4														1	0	0	0	0			Q + 0 → M
5																					
6																					
7																					

Figure 2-4(b). (Continued)

SLF
STEERING CODING FORM

Name_____

Date_____

OP	STORAGE TO SELECTED				WRITE	L SELECTED TO OUTPUT 1				U SELECTED TO OUTPUT 2				P SELECTED TO OUTPUT 3				Σ_1	Σ_2		COMMENTS
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	
1																					
2	0	1	0	0	0	0	1	1	1	1	0	0	0	0	0	1	0				Y → P ROM → LL, UL
3	0	1	0	0	0									1	0	0	0	1			LI + B → D
4																					
5																					
6																					
7																					

Figure 2-4(c). (Continued)

PMU
CONSTANT ROM

Name________________

Date________________

OP	SIGN		1	2	3	4	5	6	7	8	COMMENTS
1		•									
2		•									
3	-	•	1	2	5	8	9	6	3	3	-a3
4	0	•	4	2	1	1	1	1	1	1	-a2
5	0	•	0	0	0	0	0	0	0	0	-a1
6	-	•	2	3	4	5	6	7	8	1	-a0
7		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									

Figure 2-4(d). (Continued)

PDU
CONSTANT ROM

Name________________

Date________________

OP	SIGN		1	2	3	4	5	6	7	8	COMMENTS
1		•									
2		•									
3		•	5	0	0	0	0	0	0	0	-c
4		•									
5		•									
6		•									
7		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									

Figure 2-4(e). (Continued)

SLF (E8)
CONSTANT ROM

Name________________

Date________________

OP	SIGN		1	2	3	4	5	6	7	8	COMMENTS
1		•									
2	0	•	1	2	3	4	5	6	7	8	-L1
3	-	•	4	6	8	2	2	1	1	1	-8
4		•									
5		•									
6		•									
7		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									

Figure 2-4(f). (Continued)

SLF (E9)
CONSTANT ROM

Name________________

Date________________

OP	SIGN		1	2	3	4	5	6	7	8	COMMENTS
1		•									
2	0	•	8	8	8	9	9	9	8	8	-L2
3		•									
4		•									
5		•									
6		•									
7		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									
		•									

Figure 2-4(g). (Continued)

	9	Wing Sweep D/A Update
	8	Digital Outputs
	7	Update D/A's (Except Wing Sweep); Update IFF; Update Switches
	6	BITE Control
	5	Sensor Memory - Ps
	4	Sensor Memory - Pt
	3	Counter Shift Out Ps; Counter Shift Out Pt; Frame Mark, Built-in Qc, Executive Reset (16), Internal Switches
	2	Spare
LSB	1	Parity

The following describes each bit and how it presently interfaces with the hardware. A discussion on Simulation (see Computer Aids, Section 3) and its relationship with each bit is also included.

Bit 1

(a) Hardware--This bit will be such that the number of transitions from 0 → 1 and 1 → 0 is odd.

(b) Simulation--The simulation will determine the state of this bit.

Bit 2

(a) Hardware--Spare

(b) Simulation

Bit 3

(a) Hardware--This should be a 40-bit period signal lasting W_A and W_0. Each occurrence of this bit will determine 1 of 16 conditions. The bits may occur consecutively.

1. Counter shift out Psi
2. Counter shift out Pti
3. Frame mark
4. DT2 (internal switches)
5. DT3 (internal switches)
6. DT4 (internal switches)
7. Built-in Qc test
8. Executive reset

9 through 16

(b) Simulation--The simulation will verify that this bit always occurs 16 times throughout the frame. The order of the above conditions will be an input to the simulation.

Bit 4

(a) Hardware--This should be a 40-bit period signal lasting W_A and W_0. Each time this bit occurs, a data word will be shifted out of the sensor memory (Pt) during W_0.

(b) Simulation--The simulation should verify that it does not occur more than 24 times or simultaneous with bit 5 or bit 13.

Bit 5

(a) Hardware--This should be a 40-bit period signal lasting W_A and W_0. Each time this bit occurs, a data word will be shifted out of the sensor (Ps) during W_0.

(b) Simulation--The simulation should verify that this bit does not occur more than 24 times or simultaneous with bit 4 or bit 13.

Bit 6

(a) Hardware--This bit should control the BITE control logic. It will be a W_A and W_0 length signal. It should always be programmed for 1 OP only and 3 OPS after EB9 and for D/A associated with Outputs 44 and 47.

This bit should be programmed at least 8 more times with 3 OPS after EB7.

(b) Simulation--The simulation should verify that it lasts for 1 OP and that it comes on always and only 3 operations after bit 9 and for bit 7 occurrences associated with Outputs 44 and 47.

This simulation should check that this bit occurs at least 8 more times with 3 OPS after EB7.

Bits 6 and 7 cannot occur together.

Bit 7

(a) Hardware--This bit will be a 1 every time an output from the processor needs to be updated, such as D/A and switches.

The output discretes and data will be picked up by one occurrence of this bit. Each time this bit occurs, it will indicate a different condition; i.e., different D/A or switch. The total number of conditions will be less than 50.

This bit should be programmed for one OP only each time a new update is necessary.

If an EB6 occurs after an EB7, then the next EB7 must be 4 OPS after the last EB7.

(b) Simulation--The simulation should verify that this bit lasts only 1 OP time each time it occurs. The bit cannot recur for 4 operations.

Bits 6 and 7 cannot occur together.

The order of the above conditions will be an input to the simulation.

The simulation should verify that if an EB6 occurs after an EB7, then the next EB7 must be 4 OPS after the last EB7.

Bit 8

(a) Hardware--This bit indicates that data from the PDU-RAS are to be used as a digital output. At the same time, the PDU data ROM will be shifting out the I.D. code.

This bit should occur for 1 OP at a time, and there should be a minimum of a 4 OP gap between occurrences of this bit.

(b) Simulation--The simulation should verify that this bit occurrence lasts for only 1 OP and that a minimum gap of 4 OPS is between each occurrence. Bit 15 must occur when bit 8 does.

Bit 9

(a) Hardware--This is a 40-bit period signal lasting W_A and W_0. The occurrence of this bit will update the wing sweep D/A (Output 40).

(b) Simulation--The simulation should verify that this bit lasts only 1 OP each time it occurs. The bit should not occur more than one time throughout the frame.

Bit 10

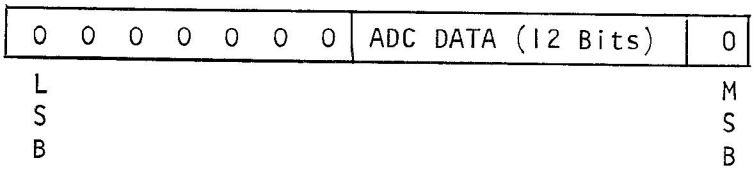
(a) Hardware--When this bit goes to a logical "1," the ADC will step its multiplexer. This bit should last for only 1 OP time.

(b) Simulation--The simulation should verify that this bit lasts for only 1 OP at each occurrence.

Bits 10 and 11 should alternate in occurrence.

Bit 11

(a) Hardware--When this bit goes to a logical "1," the data from the ADC output register will be shifted out during W_0 as follows:



This bit will go to a logical "1" at least 6 (n+6) operations after bit 10 went to a "1."

(b) Simulation--The simulation should verify that this bit lasts only one operation at each occurrence and occurs not before 6 operations after the occurrence of bit 10.

Bits 10 and 11 should alternate in occurrence.

Bit 12

(a) Hardware--This bit indicates that the SLF comparator ($\overline{B}$) output has the result of an internal check. One bad check must be programmed, and as many good checks as desired. Operation 511 should not be used.

(b) Simulation--The simulation should verify that when this bit occurs, the result is as predicted.

The predicted result and OP number should be an input to the simulation. Operation 511 should not be used.

Bits 13 through 19

(a) Hardware--When this bit is a logical "1," the ROM associated with it will be enabled. When this bit is a logical "0," the ROM associated with it will be disabled and the last accessed 20-bit sequence from the data and control ROM's will be accessed.

Bit 13	PMU	Data	Enable
Bit 14	PMU	Control	Enable
Bit 15	PDU	Data	Enable
Bit 16	PDU	Control	Enable
Bit 17	SLF	Data-1	Enable
Bit 18	SLF	Data-2	Enable
Bit 19	SLF	Control	Enable

This will be discussed further in this section.

(b) Simulation--The simulation should use these bits to control the named ROM's. Bit 15 must always occur with bit 8.

Bit 20

(a) Hardware--When this bit is a logical "1" the ROM associated with should be stepped to its next position. The contents of the register is sampled by the SLF D input at $T_{19} \cdot W_A$.

When this bit is not a "1," the discrete output will be zero.

(b) Simulation--The simulation should use this bit to step the discrete register.

Executive bits 13 through 19 are assigned to control the processor to enable and disable the associated groups of control and data ROM's. The reason for having such a control is to minimize the usage of the processor ROM's. The association of the computer program drawing to the ROM's is as follows:

If the multiplier (PMU) module was used 100 percent of the time, the PMU control and data ROM's would have to be incremented every operation. This means that each associated group of ROM's (control and data) would be stepped 512 times. This would be 512/128 = 4 ROM's for each associated group (28 ROM's total for all seven groups). Now suppose that only half of the control words and only half of the data words are needed sporadically throughout the program. This would allow two alternatives. The first alternative would be to increment the ROM's each operation and to "pad" the words not used with zeros. This would provide maximum programming flexibility and maximum ROM storage. The second alternative would be to provide an outside source that would increment the ROM's only when they are needed. This would provide minimum programming flexibility and minimum ROM storage. This is essentially what the executive bits 13 through 19 do. When a system is being considered, a considerable number of ROM's can be saved using this method. In the F-14A CADC, 14 ROM's were saved at the cost of 4 executive ROM's. This is a total saving of 10 ROM's. Before actual implementation of a system, the system engineer would have to make a tradeoff of programming flexibility versus ROM cost.

SOFTWARE SYSTEMS AND CONSIDERATIONS

As has been shown, the software for this system is closely associated with the hardware arrangement. This fact is related to two factors: (1) the flexible arrangement of the hardware and its interconnects and (2) the present means of coding by machine language. The latter could be eliminated by developing a mnemonic language to aid the programmer of the system. The former is an advantage of the system in that the system becomes efficiency adaptable to many problem statements.

Software and Hardware Relationship

After the program mechanization has been completed, an estimate is made on how many of each arithmetic operation and logical function must be performed. From this estimate and the required update rate of the calculations, the number of arithmetic modules can be determined, and the interconnect of the hardware defined. The most efficient way to determine this interconnect is to connect the circuits as the computer program drawing is made. This consists of determining which circuit outputs are connected to which steering inputs. As each data path is needed in the program, it is added to the interconnection. When the program drawing is completed, the total interconnect will be completely laid out. Any inputs to the steering that have not been connected are not needed and can be grounded. After this step is completed, the program drawing can be reprogrammed with the final interconnect if greater efficiency in programming and memory storage is desired.

An important consideration when programming the processor is the propagation delays encountered when transferring data from circuit-to-circuit. As can be seen, the path the output data takes until it gets transferred to an input is programmed. In the F-14A CADC, the maximum time allowed for this transferring is the ϕ 2 clock time (approximately 2.0 μsec). The programmer that makes the computer program drawing must consider the length of each path he programs such that it does not exceed the width of the ϕ 2 clock. The maximum allowable delays out of, through, and into each circuit are listed in the source control drawing of each circuit. The simulation discussed under Computer Aids, Section 3, provides a means of keeping a cumulative total of the propagation delays for each data path.

The software for the F-14A CADC uses arithmetic and logical functions as follows:

Multiplies	300
Divides	105
Adds or subtracts	225
Limits	75
Square roots	4
And/or	20
If transfer	55
Discrete inputs	25
Discrete outputs	21
A/D	15
D/A	23

For the present hardware configuration, this represents about 50 percent of its total computational capability, i.e., maximum number of multiplies is 512, maximum number of divides is 512, etc. This capability should not be confused

$$X = \frac{\left[\left(\frac{Psi}{Pti}\right)_{1.3}\right]^{L2}_{L1} - \frac{K_4}{1.3}}{K_5}$$

$$Z = \frac{\left[\left(\frac{Psi}{Pti}\right)_{1.3}\right]^{L4}_{L3} - \frac{K_6}{1.3}}{K_7}$$

where

K_4, K_6 = Preconditioning offset constant

K_5, K_7 = Preconditioning scaling constant

with the subject of programming flexibility. As previously discussed, programming flexibility depends on the "pad" locations in the ROM's.

Suppose a module uses 3 ROM's (384 words). If the program requires only 2-1/2 ROM's (320 words), then it could be said that the hardware has capability for 64 more words. Thus, programming flexibility is shown in the computer program drawing and how the spare 64 words are spaced throughout the program is illustrated. If they are at the end of the program, then minimum reprogram flexibility is available because the executive words need reprogramming to make use of these spares. If the spares are spaced logically throughout the program, then maximum reprogram flexibility could be obtained, thus requiring minimum executive reprogramming.

In this system, as in most cases, maximum efficiency and flexibility can only be obtained through the complete knowledge of both the hardware and the software. Programming ingenuity is a big factor if minimum hardware is the objective.

PROGRAMMING EXAMPLES

This section consists primarily of the programming examples obtained from the F-14A CADC program. For each example given, a mathematical model, a mechanization drawing, and a program drawing will be provided. The coding of each example will serve as an exercise to the reader.

Angle of Attack (Figures 2-5 and 2-6)

Indicated angle of attack is corrected to true angle of attack by the following equation.

$$\alpha_T = \underbrace{f_1\left(\frac{Psi}{Pti}\right)}_{\text{SLOPE}}\alpha_i + \underbrace{f_2\left(\frac{Psi}{Pti}\right)}_{\text{INTERCEPT}}$$

$$\frac{\alpha_T}{K_1} = \left[\frac{\left(\frac{\sum_{i=0}^{6} a_i X^i}{K_2}\right)}{K_3}\right]\left(\frac{\pm\alpha_i}{50}\right) + \overbrace{\left(\frac{\sum_{i=0}^{6} b_i Z^i}{K_1}\right)}$$

where,

K_1 = Normalizing scaling constant of function α_T

K_2 = Normalizing scaling constant of polynomial

K_3 = Mechanized scaling factor

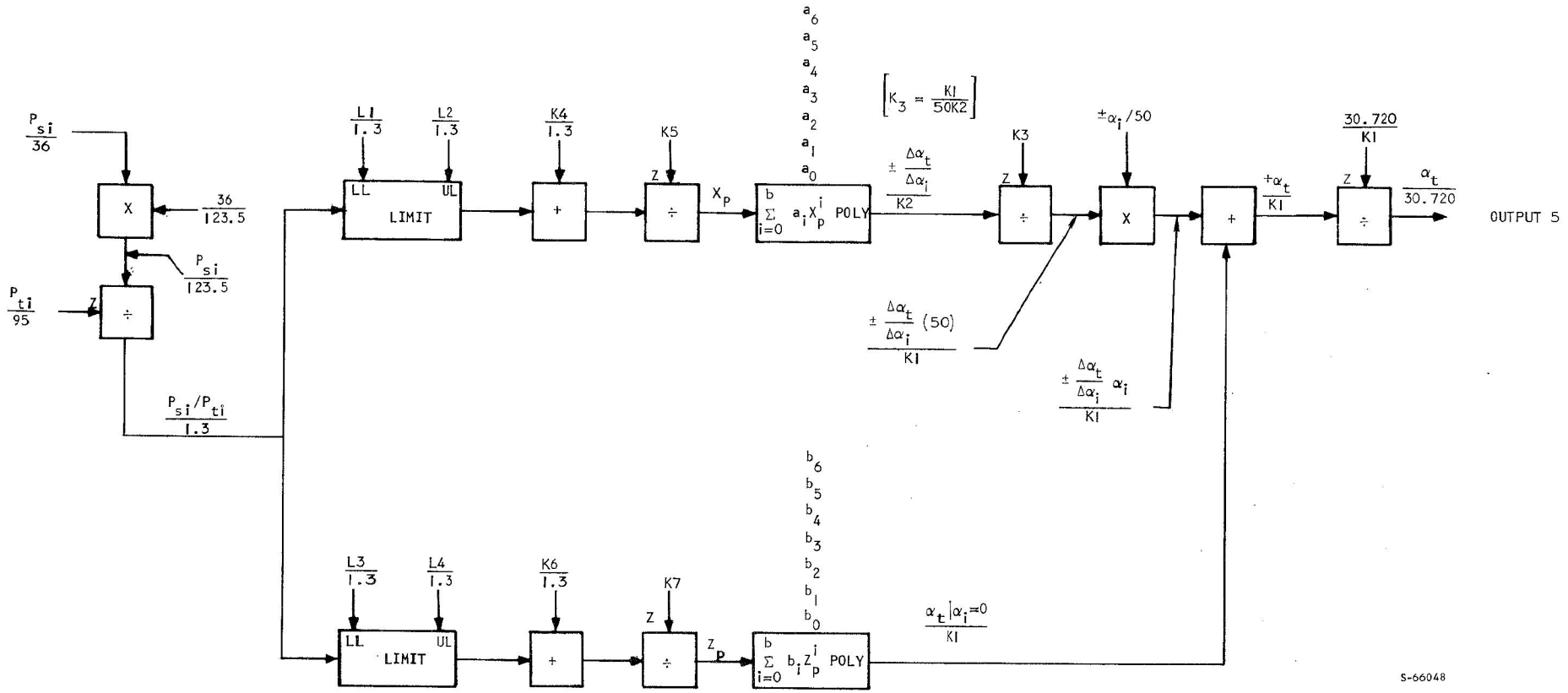


Figure 2-5. Angle-of-Attack Correction Mechanization

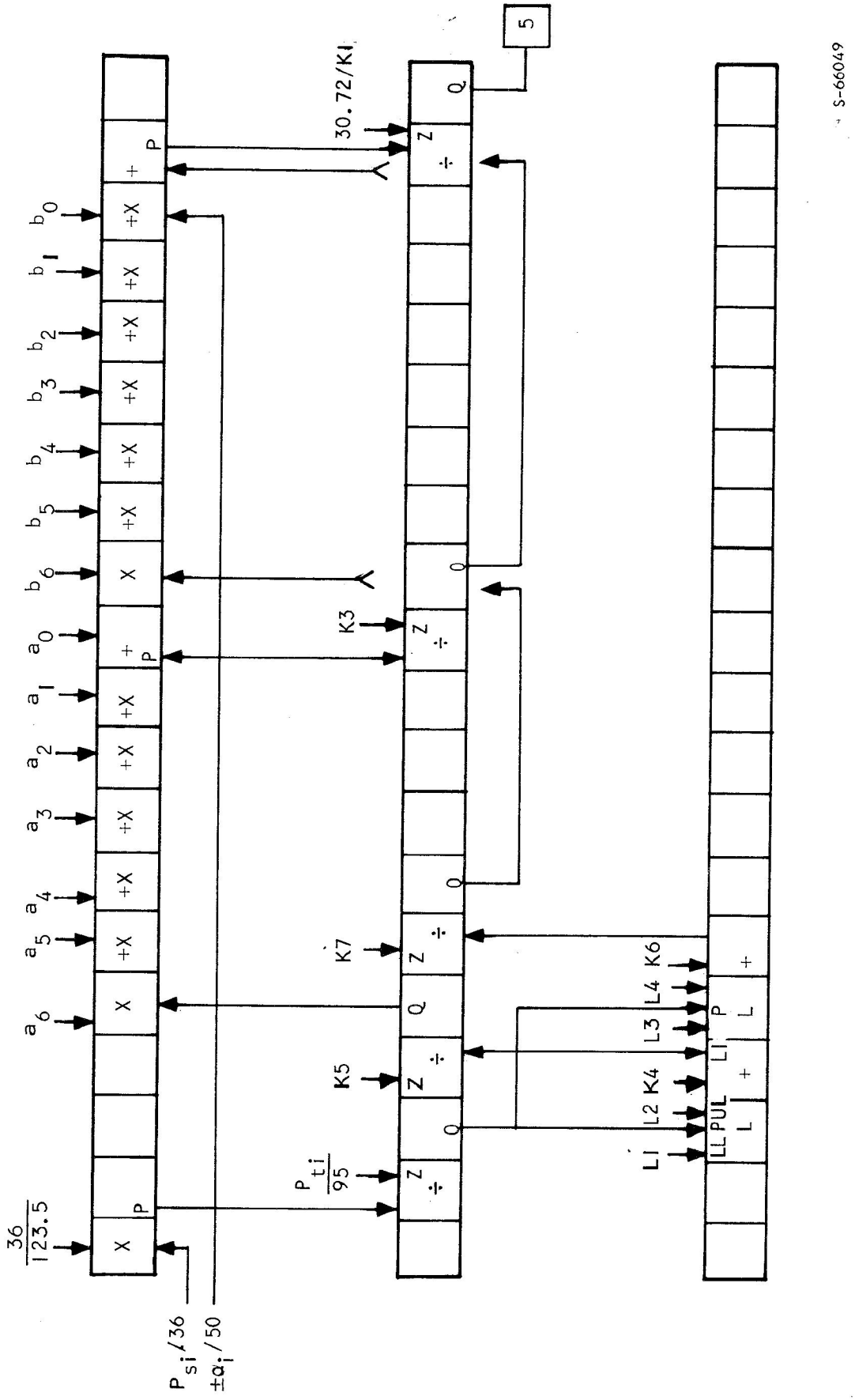


Figure 2-6. Angle-of-Attack Correction Program Drawing

Engine Pressure Ratio (EPR) (Figures 2-7 and 2-8)

Output 38A, B

$$EPR = \frac{P_{T7}}{P_T} - EPR_{STD} + f_1$$

$$= \frac{P_{T7}}{P_T} + f_2$$

where

f_1 = f(ambient temperature) per specification, Figure 3.6

f_2 = $-EPR_{STD} + f_1$ (mechanized with 4 zones as shown below)

EPR_{STD} = EPR from f_1 at 59^0F (518.67^0R)

1. Zone 1

Temperature range T_1 ^0R to T_2 ^0R

EPR range E_1 to E_2

$$\text{Slope} = \frac{E_2 - E_1}{T_2 - T_1} = SI$$

$$f_1 = T_{T(input)}\Big]_{TI}^{T5} \times SI + EPR\ (SI) \text{ at } 0^0R \text{ (offset)}$$

$$f_2 = T_{T(input)}\Big]_{TI}^{T5} \times SI + EPR\ (SI) \text{ at } 0^0R - EPR\ STD = K2$$

$$\frac{f_2}{KI} = \frac{T_T}{4900}\Bigg]_{\frac{TI}{4900}}^{\frac{T5}{4900}} \left(\frac{4900\ (SI)}{KI}\right) + \frac{K2}{KI}$$

2. Zone 2

Temperature range $T_2\ ^{\circ}R$ to $T_3\ ^{\circ}R$

EPR range E2 to E3

$$\text{Slope} = \frac{E3 - E2}{T_3 - T_2} = S2$$

$$f_2 = T_{T(\text{input})}\Big]_{TI}^{T5} \times S2 + EPR\ (S2)\ \text{at}\ 0^{\circ}R - EPR\ STD = K3$$

$$\frac{f_2}{KI} = \frac{T_T}{4900}\Bigg]_{\frac{TI}{4900}}^{\frac{T5}{4900}} \left(\frac{4900\ (S2)}{KI}\right) + \frac{K3}{KI}$$

3. Zone 3

Temperature range $T_3\ ^{\circ}R$ to $T_4\ ^{\circ}R$

EPR range E3 to E4

$$\text{Slope} = \frac{E4 - E3}{T_4 - T_3} = S3$$

$$f_2 = T_{T(\text{input})}\Big]_{TI}^{T5} \times S3 + EPR\ (S3)\ \text{at}\ 0^{\circ}R - EPR\ STD = K4$$

$$\frac{f_2}{KI} = \frac{T_T}{4900}\Bigg]_{\frac{TI}{4900}}^{\frac{T5}{4900}} \left(\frac{4900\ (S3)}{KI}\right) + \frac{K4}{KI}$$