APOLLO

R-467

THE COMPLEAT SUNRISE
BEING A DESCRIPTION OF PROGRAM SUNRISE
(SUNRISE 33 - NASA DWG #1021102)

September 1964

MIT

CAMBRIDGE 39, MASSACHUSETTS

APOLLO

Approved: Milton B. Trageser Date: 9/17/64
MILTON B. TRAGESER, DIRECTOR
APOLLO GUIDANCE AND NAVIGATION PROGRAM

Approved: Roger B. Woodbury Date: 9/18/64
ROGER B. WOODBURY, DEPUTY DIRECTOR
INSTRUMENTATION LABORATORY

R-467

THE'COMPLEAT SUNRISE
BEING A DESCRIPTION OF
PROGRAM SUNRISE
(SUNRISE 33 - NASA DWG #1021102)

September 1964

R. Battin (SGA)
R. Crisp (ISS)
A. Green (DDG)
T. J. Lawton (SGA)
C. A. Muntz (SGA)
J. Rocchio (DDG)
E. Smally (DDG)

MIT INSTRUMENTATION LABORATORY

CAMBRIDGE 39, MASSACHUSETTS

COPY # 305

ACKNOWLEDGMENT

This report was prepared under DSR Project 55-191, sponsored by the Manned Spacecraft Center of the National Aeronautics and Space Administration through Contract NAS 9-153.

R-467

TIIE COMPLEAT STJNRISE
BEING A DESCRIPTION OF PROGRAM SUNRISE
(SUNRISE 33 - NASA DWG # 1021102)

ABSTRACT

SUNRISE (SUNRISE 33 - NASA DWG # 1021102) is a computer program which has been prepared for the Apollo Guidance Computer and is intended for operation with, at least, Apollo Guidance and Navigation Systems 4 through 7. The objectives of SUNRISE are to:

a. provide the groundwork in terms of input/output, control and utility programs for all subsequent Apollo Guidance Computer programs;

b. provide the necessary tools for proofing guidance systems 4 through 7;

c. demonstrate the efficacy of an all-digital simulation as a checkout tool; and

d. demonstrate rapid turn around time for manufacture and checkout of core ropes.

PREFACE

The development of Apollo Guidance Computer (AGC) programs for the manned lunar landing mission is, of necessity, an evolutionary process. This is true because as the design of hardware, the mission profile, and the man/machine interface evolve, each change must be reflected in the AGC program. Further, the Guidance Computer Programming state-of-the-art has undergone significant development over the last several years.

The first noteworthy group of AGC programs was the ECLIPSE series. These were developed for use with the computer sub-system and were used chiefly to evaluate the Display/Keyboard design. This series of programs became available for testing in late **1963.**

SUNRISE represents the first series of programs at the G&N system level. It incorporates

1) Complete Input/Output Control Programs
2) Complete Executive Control Programs
3) Some Mission Programs
4) System Test Programs
5) Restart Capabilities (after transient failure)

Future programs will be built on the foundation *of* SUNRISE. More and more mission programs (e.g. Entry, Thrust Vector Control, etc.) will be incorporated without the necessity of changing the basic executive, Input/Output or restart philosophy embodied in SUNRISE. Great confidence in the efficacy *of* these basic tools will be gained by their operation over a period *of* months.

Each future AGC program series will be obtained by adding to its immediate predecessor. Hence, there is an orderly progression of program series leading to the final goal where at each step, the invaluable operating and check-out experience is not lost.

TABLE OF CONTENTS

INTRODUCTION	1
1. Mission Programs	1
2. Executive Control	2
3. Input/Output Control	2
4. Auxiliary Routines	3
INTERPRETER	5
1. General Description	5
2. Addressing	13
3. Operation Codes	15
4. The VAC Area	16
5. Summary of Operation Codes	17
6. Description of Operation Codes	20
7. Miscellaneous Instructions	33
8. YUL System Formats	39
9. Formats Within AGC Words	40
EXECUTIVE	43
1. Priority Control of Jobs	43
2. Time Sharing of Erasable Storage	45
3. Computer Activity Display	46
WAITLIST	47
MASTER CONTROL	48
1. Program Control From the Keyboard	48
2. Mission Program Restart After GO	48
3. **Major** Mode Display	50
FRESH START AND RESTART	51
DOWNRUPT PROCESSOR	52

T4RUPT OUTPUT CONTROL 55
1. Relay Driving 57
2. IMU CDU Driving 57
3. Optics CDU Driving 58
4. IMU Mode Sampling (KSAMP) 58
5. Optics Mode Sampling (OPTSAMP) 59
6. Down Telemetry Monitor 59
7. Failure Signal Monitor 59

MODE SWITCHING AND MARK 62
1. IMUZERO 64
2. IMUCOARS 64
3. IMUFINE 65
4. IMUATTC 65
5. IMUREENT 65
6. IMULOCK 65
7. IMURECOR. 66
8. IMUSTALL 66
9. OPTZERO 66
10. OPTCOARS 66
11. OPTTRKON 66
12. SCTMARK 66
13. SXTMARK 66
14. MKRELEAS 68
15. OPTSTALL 68
16. CDUINC 69
17. IMUPULSE 69

EXTENDED VERBS FOR MODING 71

AGC SELF-CTIECK 73
1. CCSCHK 73
2. PTY+ERAS 74
3. SCCHK 74
4. MPCHK 75

5. SUCHK 75
6. DVCHK 75
7. TS+-CHK 75

ALARM PROCESSOR 77
1. Alarm 77
2. Abort 77

INTERBANK COMMUNICATION 80
1. BANKCALL 80
2. SWCALL 80
3. POSTJUMP 80
4. BANKJUMP 81
5. DATACALL 81
6. MAKECADR 81

ORBITAL INTEGRATION 82
1. Encke's Method 82
2. Disturbing Acceleration 86
3. Error Transition Matrix 89
4. Numerical Integration Method 91

PRELAUNCH ALIGNMENT 93
1. Initialization 93
2. IMU Moding 93
3. Vertical Erection 94
4. Gyrocompassing 95

RTB ROUTINES 99
1. LOADTIME 99
2. FRESHPD 99
3. ZEROVAC 99
4. CDULOGIC 99
5. 1STO2S 100

6. READPIPS 100
7. PULSEIMU 100
8. SGNAGREE 100
9. TRUNLOG 100

SYSTEM TEST 101
1. SEXTANT 101
2. The Accelerometer Test 101
3. Operating Procedures for Automatic Test in Program SUNRISE 103

IN-FLIGHT ALIGNMENT SUBROUTINES 106
1. CALCGTA 106
2. ARCTRIG 107
3. SMNB 107
4. NBSM 108
5. CALCGA 109
6. SXTNB 109
7. AXISGEN 110
8. CALCSXA 111
9. SXTANG 112

PINBALL 113
1. Input/Output Interfaces of Keyboard and Display System Program 114
2. Keyboard Use of Keyboard and Display System 118
3. Internal Use of Keyboard and Display Program 126
4. Alarms and Special Controls 132
5. Input/Output Codes 135
6. Verbs 137
7. Nouns 141

LIST OF FIGURES

Interpreter Formats	**39-40**
Flow Chart for Interpreter Dispatcher	**42**
DOWNRUPT Logic	54
T4RUPT Sequencing	56
IMU Mode Sampling	60
Optics Mode Sampling	**61**
Vertical Erection Loop	**97**
Gryocompassing Loop	98
Display Panel Layout	**116**

INTRODUCTION

SUNRISE (SUNRSE 33 - NASA DWG # 1021102) is a computer program which has been prepared for the Apollo Guidance Computer and is intended for operation with, at least, Apollo Guidance and Navigation Systems 4 through 7. The objectives of SUNRISE are to:

a. provide the groundwork in terms of input/output, control and utility programs for all subsequent Apollo Guidance Computer programs;

b. provide the necessary tools for proofing guidance systems 4 through 7;

c. demonstrate the efficacy of an all-digital simulation as a checkout tool; and

d. demonstrate rapid turn around time for manufacture and checkout of core ropes.

SUNRISE consists of approximately 18 program sections which occupy about 10,000 words of computer storage. Each section is designed to perform one or more distinct functions which are described in some detail in this report. An appreciation of this document requires a certain familiarity with the overall Apollo Guidance and Navigation System, and especially, MIT/IL Report R-393 which describes the logical organization of the Apollo Guidance Computer. The various program sections fall logically into Pour categories:

1. Mission Programs

Mission programs are so-called because they perform major activities necessary for the accomplishment of Apollo mission objectives. The three mission programs included in SUNRISE are:

a. Orbital Integration - to solve the second order differential equations describing the motions of a body subject only to the forces of gravity.

b. Pre-launch Platform Alignment - to align the Apollo Inertial Measurement Unit stable member to a desired orientation while this device is located on the surface of the earth.

c. System Test Programs - to measure various parameters of the guidance system and verify that its performance is within specifications.

2. Executive Control

Executive control programs coordinate and synchronize AGC activity to guarantee orderly and timely execution of required tasks. The programs in this category are:

a. Master Control - to initiate, terminate, and supervise restart of all mission programs. The restart function takes place after a transient failure of the computer.

b. Executive - to supervise the execution of all programs which do not operate in the interrupt mode. Each program is assigned a priority and the Executive routine allows the highest priority program to operate at any given time (execution of the higher priority job commences within 20 ms). Up to seven programs may he in various states of completion.

c. Waitlist - to supervise the execution of programs which must operate at a specific time. All execution controlled by the waitlist must take place in the interrupt mode. Request for lengthy computation (4 ms or greater) may be sent to the Executive under Waitlist control for subsequent execution. Up to six programs may be simultaneously under control of the Waitlist.

3. Input/Output Control

Input/ output control programs receive signals from various devices for processing and send commands for control and display purposes. Most of the data processed by these programs comes from or is to be routed to mission programs.

a. Down-Link Servicing - to transmit data to the NAA down-telemetry programmer for subsequent transmission to the ground.

b. T4RUPT Output Control - to service output devices which require high frequency attention. Some of the functions performed by this program are to:

1) Drive the IMU CDU's to align the platform or steer the spacecraft,

2) Drive the Optics CDU's to position the Sextant and Telescope,

3) Update the electroluminescent display panel,

4) Mode switch and monitor the IMU and Optics, and

5) Verify the correct operation of the Down-Link servicing program.

c. Mode Switching and MARK Routines - to supervise the input/output execution performed by T4RUPT as it relates to the IMU and optics moding. In addition, the MARK function for optical measurements is supervised by this program. Mission programs operate devices through this routine rather than through T4RUPT.

d. Keyboard and Display Program (Pinball) - to perform various functions in response to commands from the keyboard and to display information as a result of these commands or commands from other programs.

e. Extended Verbs - to provide an appendage to Pinball which allows various test and check-out functions to be performed with the AGC.

4. Auxiliary Routines

Auxiliary routines comprise all other programs necessary for efficient operation of the computer. In particular:

a. List Processing Interpreter - to allow compact and readable programs to be prepared at the expense of computer operational speed. It contains explicit double precision vector and matrix operations and provides the AGC with many of the characteristics of a large commercial computer.

b. In-Flight Alignment Subroutines - to provide the framework for alignment of the Inertial Measurement Unit, This consists of interpretive subroutines required mainly for geometric transformations.

c. Interbank Communication - to allow convenient communication among otherwise isolated banks of fixed memory.

d. Fresh Start - to initialize certain programs necessary for control of the computer. Execution of this program should only be necessary just after the power is turned on.

e. RTB Routines - to provide an appendage to the Interpreter which effectively increases its operation code and provides a convenient link between the basic and interpretive languages.

f. Alarm Processor - to provide a method of displaying an alarm to the operator and forcing a computer restart if necessary.

g. Self-Check - to verify proper performance of most AGC control pulses during the period of time which the computer would otherwise be idle. This routine has been designed to indicate many of the sources of trouble, in the event of a computer failure.

INTERPRETER

1. General Description

In most single address interpretive systems, the address of the operand is stored in the same word as the operation code. If the word length of the computer is short, however, addressing problems become difficult when one can only directly address a small fraction of machine memory. As all 15 bits are necessary to address any of the AGC's 25,600 registers, an interpretive system which allows a full AGC word for the operand address seemed more desirable. One such system might consist of three 5-bit operation codes packed into one word, with three words of operand address immediately following. This, however, would mean an increase of 33% in storage requirements for interpretive programs - an increase we cannot afford.

A solution to this storage requirement is the use of list-processing techniques. In such a notation, an interpretive program consists of lists called "equations". Each equation consists of a string of operators followed by a string of addresses to be used by the operators. Savings begin to appear when we make the rule that generally the first address should be used to load an accumulator, Thus, by eliminating any load instructions from the order code, space is conserved; since the vacant entries in the order code can be replaced by more complex and diverse operations, programs should consist of fewer operations.

The same savings are available in storing operations. If an address is left over at the end of an equation it will be understood that the result is to be stored in the location specified by the left over address.

As an example consider the coding of $C = A + B$. This is coded as follows

Old Stvle		As List	
DCA	**A**	**DAD**	***0***
DAD	B		**A**
DTS	**C**		B
		STORE	**C**

For a more complete example consider

$$E = \frac{A + COS[B + SQRT(A^2 + B)]}{C}$$

Old Style		As List	
DCA	A	DSQ	3
DMP	A	DAD	SQRT
DAD	B	DAD	*COS*
ITC*	SQRT	DAD	DDV
DAD	B		A
ITC*	*COS*		€3
DAD	A		B
DDV	*C*		A
DTS	E		C
		STORE	E

Note that in this example only 10% more space is needed to obtain the desired addressing capabilities (10% less if the indirect addresses for SQRT and COS which must be in the block are included.) The notation used in the new interpreter is, a form of Polish Notation. Two 7 bit operation codes are stored per word followed by a string of extended (14 bit) addresses, The integer in place of the first right-hand operation code tells the interpreter where the address string starts and is equal to the number of additional words of operation codes.

One point should be made clear with the so-called load- and store-addresses. No loading is done until the first operation specifies that it needs a double or triple precision number or a vector. Thus, if it is the first operation in an equation, DAD loads a DP quantity using the load address. Similarly, **VAD** loads a vector and TAD a TP quantity. Storing operations work in much the same way, selecting the type of quantity to be stored by the nature of the last operation performed.

In the coding of some equations, temporary results must be stored which are of no further use. For example, in the calculation of **X = AR** + BS, AR must be calculated and held temporarily while BS is computed. The product AR will

be of no further use. For this purpose, the interpreter provides a temporary area in the form of a push-down list which is referenced by an economical implied-address scheme. To store a result in this temporary area, one need only omit a store address from the end of an equation. Hence, while

```
DMP     0
        A
        B
STORE   C
```

computes $C = AB$,

```
DMP     0
        A
        B
```

computes AB and puts it in the push-down list.

To recall an operand from the push-down list, any one of three methods can be used.

a. No Address Given - If an operator finds no address in the address string, it takes its argument from this temporary list, Thus

```
DMP     1
DAD
        X
        Y
```

computes XY, adds the contents of the push-down list, and since no store address is given stores the result in the push-down list.

b. Inactive Address - A special type of address known as an inactive address is used to reference the push-down list without running out of addresses. Thus, $X = Y + SQRT\ (Y^2 + A^2)$ is coded as

```
DAD     SQRT
DAD
        Y
        -
        Y
STORE   X
```

where - is recognized by YUL as the inactive address. (This is assembled as a - 0 where addresses are usually positive. See section on addresses for details).

c. Store Address Distinction - To allow the programmer to draw from the push-down list just before storing without using an inactive address, a special format for store addresses is employed (see addressing section for format details). Thus operand addresses are distinguishable from store addresses, and when an operator finds a store address instead of an operand address it pushes up.

To complete our earlier example X = AR + BS is coded as

```
DMP     0
        A
        R
DMP     1
DAD
        B
        S
STORE   X
```

where the STORE operation code tells YUL to make a store address for X. By convention, any address to be used in storing a quantity must be a STORE address. As another example, X = AB + CD + EF becomes

```
DMP       0
          A
          B
DMP       I
DAD
          C
          D
DMP       1
DAD
          E
          F
STORE     X
```

Sometimes more than one such temporary storage register is needed in the computation of certain expressions. Consider

$$X = \frac{A^2 + B^2}{C^2 + D^2}$$

Clearly

```
DSQ       0
          D
DSQ       1
DAD
          C
```

leaves $D^2 + C^2$ in the push-down list. However another temporary location is needed for the computation of the numerator. The push-down structure provides this extra space. Thus a certain number of quantities can be retained in this list, the receipt of another quantity causing the "pushing down" of the others. When arguments are retrieved, the last one inserted (or alternatively the one on top of the stack) will be delivered. Thus the list has a List-In-First-Out (LIFO), behavior.

The complete coding of the above goes as follows:

```
DSQ     0
        D
DSQ     1
DAD
        C
DSQ     0
        A
DSQ     1
DAD     DDV
        €3
STORE   X
```

Here the store address distinction has caused the last DAD and DDV operators to reference the push-down list.

At the current time 32 registers are being allowed for the push-down list. Thus this list can accommodate up to 16 double precision quantities, 10 triple precision quantities, 5 vectors, or any combination totaling 32 words.

With this type of push-down capability extremely long equations may be calculated without storing any temporary results outside the push-down list. Often, however it is more advantageous to calculate intermediate results which can be used a number of times. This is not possible with the push-down list since once an argument has been used it is not accessible later. For this purpose a temporary block is provided in the VAC area along with the push-down list. At the present time, 20 registers are included which can be referenc'ed as ordinary erasable registers with the addresses 0 - 31D. (These addresses usually refer to specials and centrals but these are not available to interpretive programs.)

As an illustration consider

$$\overline{X} = [(\overline{A} + \overline{B}) \cdot \overline{C}] (\overline{A} + \overline{B}) + (\overline{A} + \overline{B}) X\overline{C}$$

which is most economically coded as

```
VAD       0
          A
          B
STOKE     0
NOLOD     1
DOT       VXSC
          C
          0
VXV       1
VAD
          0
          C
STORE     X
```

This temporary storage should be used to store everything but the outputs of the program: i. *e*. only those quantities which will be used by other programs **or** the program itself the next time it is run. This is necessary to consolidate temporaries and conserve erasable storage.

To summarize then, each interpretive program is provided with a VAC area consisting of a 6 word vector accumulator, a **32** word push-down list referenced by implied address techniques or by addresses 0 - **31.** (See description of VAC area for details).

Included in the order code are provisions for 16 unary operations; that is, operations not requiring an operand, but only an accumulator. For example, **DAD** is a binary operation because two arguments are used; SQRT is unary because it only uses one, the contents of an accumulator. Among these are SIN, COS, SQRT, VSQ, ABS, UNIT, etc. See the description of the order code for details. Loading and storing of results is the same, with *Y* = SIN(X) written as

```
SIN       0
          X
STORE     Y
```

To increase addressing capabilities two index registers are available to the interpretive programmer. Under control of bits in the operation code, the contents of one of these registers (selected by a bit in the address) will be subtracted from a 13 bit sub-address in the operand address. The index registers themselves may contain a positive or negative 14 bit address. This resultant address is then used to locate the operand.

Of course, indexed addresses do not change the contents of the index register involved. For index register modification a number of instructions are included. Index register manipulations may be done at anytime in an equation as they do not alter accumulators or temporaries,

Index registers also provide a means of doing simple single precision "bookkeeping" in the interpretive mode. Thus these registers serve a two-fold purpose. For example, if a counter is to, be decremented by two and then used as an address, we have

```
LXA, 1          1
INCR, 1         SXA, 1
                COUNT
                2
                COUNT
```

If this counter refers to an element in a list (i. e., is the negative of the desired address) of which we want the SIN, this can be coded in Polish as

```
LXA, 1          2
INCR, 1         SXA, 1
SIN*
                COUNT
                2
                COUNT
                0, 1
STORE           X
```

*Denotes indexed address

2. Addressing

The basic address used by the interpreter is a 14 bit quantity, called a Polish address. As opposed to basic coding, then, a Polish address can directly reference about half of the 25, *600* registers in the AGC memory, An exception to the above is that addresses *0* - 42 are reserved for temporary storage (see details on VAC area), and are relative to the beginning of the temporary area, The Polish address may reference the VAC area, all of general erasable, and all of banks 21 - 34 of fixed memory.

Many interpretive instructions can use one of the two index registers to modify a given address. Thus if *so* specified by a bit in the operation code, the 14 bit address will be interpreted as a 13 bit sub-address with the remaining bit specifying the index register to be used. This index register will be subtracted from the given 13 bit address to form an effective Polish address,, which is then used to locate the desired operand. This address modification does not alter the contents of the index register used. The formats in which both types are stored in the AGC may be found in the format summary.

Index registers may contain complete Polish addresses or the negative of such an address. However, any effective address resulting from index register modification must, in general, be positive. An exception to this is an indexed shift count which **may** be positive or negative, the sign specifying the direction of the shift. (See description of VSLT and TSLT). Thus, index registers may contain any address from $+37777_8$ to -37777_8 or in short, any single precision quantity.

Addresses in the range 32000_8 - 37777_8 are recognized to be store addresses. If the address is in this range, the low-order 10 bits are used as an erasable address and the results stored directly. Store addresses whose 10 bit erasable portion is in the range 0 - 42 refer to the temporary block, as before. If the address is in the range 34000 - 37777 the low-order 11 bits are interpreted as a 10 bit erasable address and a 1 bit index register tag. An effective address is formed as before with addresses 0 - 42 being treated as usual.

The final type of address, the inactive address, is assembled by the YUL system as a - 0, which distinguishes it from the others. (See format summary).

Two special cases arise which are not legal Polish addresses - the last register in the last bank - register 71777_8, and register 1777_8 - the last erasable storage location - as either a store address or an operand address. This seems little loss as they can be reserved for programs written in basic coding. Finally, while the 14 bit address eliminates most of the bank-switching problems associated with shorter addresses, the only restriction is that program flow must not cross bank boundaries. To continue a program once it has reached the end of bank, then, one must transfer control (with an ITC, say) to the beginning of the next bank. Two other restrictions apply to addressing: a) that every equation have at least one address (STORE addresses included) and b) that the first address in an equation be algebraically positive (e.g., not a negative index increment).

3. Operation Codes

The Polish operation code is a seven bit quantity - stored two per word. This operation code is further broken up into 2 prefix bits and a five bit selection code. The four possible prefixes are as follows:

00	Binary operation with direct operand address
10	Binary operation with indexed operand address
01	Miscellaneous operation (modify index registers, etc.)
11	Unary operation (no operand address required)

Generally, the binary operations refer to an accumulator and an operand and hence binary operators always require an address. Unary operations refer to an accumulator only and no operand addresses are needed. Miscellaneous operations are generally those which do not reference or effect an accumulator.

In binary and miscellaneous operation codes, the five bit selection code is used to specify one of 32 possible operations. In unary operation codes, the high-order 4 bits in the five bit selection code specify one of 16 unary operations, the remaining bit required only if the unary operator must load an accumulator before execution. In this case, it specifies whether the load address is direct or indexed. On the other hand, if a binary operator must load an accumulator before execution and both the load address and operand address are either indexed or direct, no special precaution need be taken. If they are not the same type, the accumulator must be loaded with a DMOVE, TMOVE, or VMOVE order, which ever is appropriate.

4. The VAC Area

Each interpretive program will have a block of 43 registers available for various purposes. They are referred to by addresses 0 - **42,** as explained earlier. A map of this area is as follows

0 - 31 Push-down List	32 - register push-down list and temporary storage
32 - 37 VAC	6 - register vector accumulator
38, 39, X1, *x 2*	Index Registers X1, X2
40, 41, S1, S2	Step Registers S1, S2
42, QPRET	Return Address, QPRET

The function of each portion was explained earlier. However, registers need not be used for the indicated purposes only. For example, if no index registers are being used and more temporary block is needed, the pairs of registers 38 - **39** and **40** - 41 can each be used to store double precision quantities.

Similarly, if no vector operations are being performed and more push-down list is needed, one can continue to "push-down" into registers *32* - **37,** accommodating three additional double-precision quantities, The symbols VAC, X1, X2, S1, S2, and QPRET are defined in the YUL language interpreter deck as 32D, 38D, 39D, 40D, 41D and 42D and may be freely referred to.

5. Summary of Operation Codes

The following operations may have indexed addresses, load a DP quantity if the load indicator is on, and cause a DP quantity to be stored at the end of an equation.

Binary:	DAD	DP Add
	DSU	DP Subtract
	BDSU	Backwards Subtract
	DMP	DP Multiply
	DMPR	DP Multiply and Round
	DDV	DP Divide
	RDDV	Backward Divide
	TSRT	Shift Right
	TSLT	Shift Left
	TSLC	Normalize
	SIGN or SGN	Set Sign
Unary:	SIN	
	COS	
	ASIN	Arcsin
	ACOS	Arccos
	SQRT	
	DSQ	Square C(MPAC)
	DMOVE	
	ABS	Absolute Value

The following operators may have indexed address, load a DP quantity if the load indicator is on, and cause a DP quantity to be stored only if a store address is given.

Binary:	BPL	Branch Plus
	BZE	Branch Zero
	BMN	Branch Minus
	BHIZ	Branch if Hi-Order Zero

The following operators may have indexed addresses, load a vector if the load indicator is on, and cause a vector to be stored at the end of an equation.

Binary:	VAD	Vector Add
	VSU	Vector Subtract
	BVSU	Backwards Subtract
	VXV	Vector Cross
	MXV	Matrix Vector
	VXM	Vector Matrix
	VPROJ	Vector Projection
Unary:	UNIT	
	VMOVE	
	ABVAL	Vector Length
	VSQ	Square of Vector Length

The following operations may have indexed addresses, but do not load an accumulator if the load indicator is on, and do not cause anything to be stored unless a store address is given.

Binary:	ITC	Transfer Control
	BOV	Branch on Overflow
	STZ	Store Zero

The following operations may have indexed and cause a TP quantity to be loaded or stored if desired.

Binary:	TAD	TP Add
	TSU	TP Subtract
Unary:	TMOVE	
	TP	Declare Triple Precision

The following operations may have indexed addresses. Their loading and storing characteristics are miscellaneous and described in the order code description.

Binary:	VXSC	Vector Times Scalar
	DOT	Dot Product
Unary:	COMP	Complement
	ROUND	
	VDEF	Vector Define
	SMOVE	Single Precision Load

The following operators must have direct addresses, do not cause loading of an accumulator if the load indicator is on, nor do they cause anything to be stored unless a store address is given.

Miscellaneous:	EXIT	
	RTB	Return to Basic
	AXT	Address to Index True
	AXC	Address to Index Complemented
	LXA	Load Index from the Address
	LXC	Load Index Complemented
	SXA	Store Index in the Address
	XCHX	Index Register Exchange
	INCR	Increment Index
	XAD	Index Register Add from Erasable
	XSU	Index Register Subtract from Erasable
	AST	Address to Step True
	TIX	Transfer on Index
	NOLOD	Turn off Load Indicator
	ITA	Transfer Address
	ITCI	Transfer Control Indirect
	SWITCH	Change Switch Bit
	TEST	Test Switch Bit
	LODON	Turn on Load Indicator
	ITCQ	Return using QPRET

6. Description of Operation Code

The following is a description of the 72 instructions presently available with the interpreter. A rough estimate of the average execution time is given, although the times quoted are probably good to 10%.

Load and Store Times

Times required for loading and storing are as follows:

Loading	DP	1.2 ms
	TP	1.25 ms
	VECTOR.	2.0 ms
Storing	DP	1.0 ms
	TP	1.1 ms
	VECTOR	1.85 ms

ITC X Interpretive Transfer Control

Begin the equation starting at X, leaving in QPRET the Polish address of the beginning of the next equation. ITC must be the last (or only) operator in an equation. X must refer to fixed memory.

Timing: 1.4 ms Average

STZ Store Zero

Store a single precision zero in X. If STZ is the last operator in the current equation and no store address is given, begin the next equation without storing MPAC or VAC in the push-down list.

Timing: 1.2 ms Average

HOV X Branch on Overflow

If the overflow indicator is on, turn it off, and begin the equation at X. If not, continue as in *STZ*.

Timing: 1.2 ms Average

Comments: Overflow in the following operations turn on the overflow indicator: DAD, DSU, BDSU, DDV, BDDV, TAD, TSU, TSLT, MXV, VXM, VAD, VSU, BVSU, DOT, VXV, ROUND, VSLT, UNIT, and VPROJ.

DAD X Double Precision Add

C(MPAC, MPAC + 1) + C(X, X + 1) replace C(MPAC, MPAC + 1). Turn on the overflow indicator if such occurs. DAD MPAC will not work; use TSLT 1 instead.

Timing: 1.65 ms Average

DSU X Double Recision Subtract

C(MPAC, MPAC + 1) - C(X, X + 1) replace C(MPAC, MPAC + 1). Tuen on overflow indicator if necessary.

Timing: 1.75 ms Average

BDSU X Backwards Double Precision Subtract

C(X, X + 1) - C(MPAC, MPAC + 1) replace C(MPAC, MPAC + 1). Turn on overflow indicator if necessary.

Timing: 1.8 ms Average

DMP X Double Precision Multiply

C(MPAC, MPAC + 1) times C(X, X + 1) replace C(MPAC, MPAC + 1, MPAC + 2); i.e., a triple precision result is obtained. However, only a double precision result will be stored unless a TP operation is the last in the equation, thus declaring the mode triple precision just before storing.

Timing: 2.6 ms Average

DSQ Double Precision Square Operation

C(MPAC, MPAC + 1) are squared, leaving a triple precision result in MPAC, MPAC + 1, MPAC + 2. Ordinarily a double precision result will be saved in a later storing operation, unless the TP code is used as in DMP.

Timing: 1.95 ms Average

DMPR X Double Precision Multiply and Round

C(MPAC, MPAC + 1) times C(X, X + 1) replace C(MPAC, MPAC + 1, MPAC + 2). The result is then rounded to double precision, setting C(MPAC + 2) to zero in the process.

Timing: 2.9 ms Average

DDV X Double Precision Divide

If C(MPAC, MPAC + 1) < C(X, X + 1), C(MPAC, MPAC + 1) divided by C(X, X + 1) replace C(MPAC, MPAC + 1). If C(X, X + 1) ≤ C(MPAC, MPAC + 1) turn on the overflow indicator and exit, not necessarily leaving C(MPAC, MPAC + 1) as they were initially.

Timing: 3.7 ms Average

BDDV X Backwards Double Precision Divide

If C(MPAC, MPAC + 1) > C(X, X + 1), C(X, X + 1) divided by C(MPAC, MPAC + 1) replace C(MPAC, MPAC + 1). If C(X, X + 1) ≥ C(MPAC, MPAC + 1) turn on the overflow indicator and exit without necessarily leaving C(MPAC, MPAC + 1) as it was initially. Otherwise the same as DDV.

Timing: 3.9 ms Average

BPL X Branch Plus

If C(MPAC, MPAC + 1, MPAC + 2) ≤ 0 begin the equation at X. Otherwise continue with the present equation. If there are no more operations in the current equation and no store address is given, do not store MPAC in the push-down list, but being the next equation immediately.

Timing: 1.35 ms Average

BZE X Branch Zero

If C(MPAC, MPAC + 1, MPAC + 2) = 0 begin the equation at X. Otherwise continue as in BPL. +0 and -0 are always equivalent in the interpreter.

Timing: 1.35 ms Average

BMN X Branch Minus

If C(MPAC, MPAC + 1, MPAC + 2) < 0 begin the equation at X. Otherwise continue as in BPL.

Timing: 1.35 ms Average

BHIZ X Branch if Hi Order *Zero*

If C(MPAC) = 0 begin the equation at X. Otherwise continue as in BPL.

Timing: 1.25 ms Average

Comment: This can be used as a scaling check, branching if | C(MPAC, MPAC + 1, MPAC + 2) | < 1/16, 384. This can also be used to test single precision quantities which might have been set to zero (by STZ, say).

SIGN —X Set Sign of MPAC

If C(X) ≥ 0, no operation occurs. If C(X) < 0 complement MPAC or VAC, which ever is appropriate. SIGN loads DP if the load indicator is on.

Timing: 1.75 ms Average

Comment: X must be in erasable

SIN Double Precision Sin Function

If α = C(MPAC, MPAC + 1) initially, SIN leaves 1/2 SIN $(2\pi\alpha)$ in MPAC, MPAC + 1.

Timing: 11.10 ms Average

COS Double Precision Cosine Function

If α = C(MPAC, MPAC + 1) initially, COS leaves 1/2 COS $(2\pi\alpha)$ in MPAC, MPAC + 1.

Timing: 11.95 ms Average

ASIN Double Precision $\sin^{-1}$ Function

If α = C(MPAC, MPAC + 1) initially, ASIN leaves $(1/2\pi)\ \sin^{-1}(2\alpha)$ MPAC, MPAC + 1. The principal values lie between $-\frac{1}{4}$ and $\frac{1}{4}$ $(-\pi/2$ and $\pi/2$ scaled). This is the inverse of SIN.

Timing: 18.55 ms Average

ACOS Double Precision $\cos^{-1}$ Function

If α = C(MPAC, MPAC + 1) initially, ACOS leaves $(1/2\pi)\ \cos^{-1}(2\alpha)$ in MPAC, MPAC + 1. The principal values lie between 0 and 1/2 (0 and π scaled). This is the inverse of COS.

Timing: 18.35 ms Average

SQRT Double Precision Square Root Function

If α = C(MPAC, MPAC + 1, MPAC + 2), SQRT leaves $\sqrt{\alpha}$ in MPAC, MPAC + 1. If in double precision C(MPAC + 2) is set to zero initially.

Timing: 4.25 ms Average

The following instructions have miscellaneous loading and storing characteristics, explained with each operation,

VXSC X Double Precision Vector Times Scalar

If the load indicator is on load a vector into VAC using the first address. Leave in VAC the product of the vector C(VAC, . . ., VAC + 5) and C(X, X + 1) where X is the second address. If the load indicator is off, and the current mode is vector, leave in VAC, . . ., VAC + 5 the product of the vector C(VAC, . . ., VAC + 5) and C(X, X + 1). If the current mode is double precision the scalar is taken as C(MPAC, MPAC + 1) and the product of the vector C(X, . . ., X + 5) and the scalar in MPAC is left in VAC, . . ., VAC + 5. In any case, a vector is stored if at the end of an equation.

Timing: 4.8 ms Average

DOT X Vector Dot Product

The dot product of C(VAC, . . ., VAC + 5) and C(X, . . ., X + 5) replace C(MPAC, MPAC + 1, MPAC + 2). Turn on the overflow indicator is such occurs.

Timing: 6.1 ms Average

Comments: Ordinarily only a double precision result will be stored. However, a triple precision result can be saved, if desired, by using the TP instruction as in DMP.

DMOVE Double Precision Move

This operation simply uses the first address to load MPAC, MPAC + 1 with the first address and proceed. DMOVE, of course, must be the first loading instruction (i. *e.*, not an index instruction) in an equation. This is useful for accomplishing A = B, etc.

TMOVE Triple Precision Move

Same as DMOVE except a triple precision number is loaded into MPAC, MPAC + 1, MPAC + 2.

TP Declare Triple Precision

Set the mode to triple precision. This is used for supplying triple precision arguments to and for getting triple precision results from DMP, DSQ, DOT, TSRT, TSLT, and TSLC.

SMOVE Single Precision Mode

Same as DMOVE, only set C(MPAC + 1) = 0.

Comments: This is particularly useful for picking up single precision numbers left by input/output routines and changing them to double precision.

VMOVE Double Precision Vector Move

Same as DMOVE, only a double precision vector is loaded into VAC, ..., VAC + 5.

VDEF Vector Define

VDEF loads the vector (V_0, V_1, V_2) into VAC, ..., VAC + 5 where it is understood that V_0 is in MPAC, MPAC + 1, that V_1 is on top of the push-down list, and that V_2 is just below V_1 in the push-down list.

Timing: 1.5 ms Average

Comments: This operator can be used to set up a vector whose components must be individually computed, and also to store three double precision quantities provided they can be stored consecutively and the computation of any of the three does not depend on any that precede it, This is necessary since any preceeding results are in the push-down list, where they should not be referenced until the **VDEF** is executed. Thus if one wishes to compute

$$X = \textit{COS}\ (\textbf{ANGLE})$$
$$\text{DOUBLE} = \textbf{2A}$$
$$\text{BSQ} = B^2$$

where X, DOUBLE, and BSQ are to be stored in the temporary block he should write

```
DSQ     0
        B
TSLT    0
        A
        1
COS     1
VDEF
        ANGLE
STORE   0
```

Thua **we** have saved two words of program by **storing** the three in one vector storate operation rather than three double precision storing operations.

ROUND Round to Double Precision

Round C(MPAC, MPAC + 1, MPAC + 2) to double precision, setting C(MPAC + 2) = 0 in the process. Turn on the overflow indicator if such occurs.

Timing: .75 ms Average

Comments: ROUND will not load an accumulator as it is anticipated that it will be used after shifts, etc., when MPAC is already loaded. If one wishes A = ROUND (B) he should write

```
TMOVE    1
ROUND
         B
STORE    A
```

ABS Absolute Value

if in double-or triple-precision leave |C(MPAC, MPAC + 1, MPAC + 2)| in MPAC, MPAC + 1, MPAC t 2.

Timing: .9 ms Average

ABVAL Vector Length

Leave half the length of the vector in VAC, ..., VAC + 5 as C(MPAC, MPAC + 1, MPAC + 2).

Timing: 10.85 ms Average

COMP Complement

Complement C(MPAC, MPAC + 1, MPAC + 2) if in double-, or triple-precision, or complement C(VAC, ..., VAC + 5) if in vector mode.

Timing: 1.7 ms Average

Comments: If COMP is the first operation in an equation, a double precision number will be loaded. If one wishes to load a vector and then complement it, the vector must be loaded with VMOVE.

rs.

TSRT X Triple Precision Right Shift

Shift C(MPAC, MPAC + 1, MPAC + 2) right X places, where X is interpreted as an integer $0 \leq X \leq 42$. If a triple precision result is to be stored, a TP instruction should be the last in the equation as in DMP. Similarly, if a triple precision number is to be loaded into the accumulator (presumably to be shifted), the equation must begin with a TMOVE instruction. Ordinarily, double precision numbers are used.

Timing: 1.9 + .25N ms Average

where N = number of 14 bit shifts required

TSLT X Triple Precision Shift Left

2)|

Same as TSRT, except C(MPAC, MPAC + 1, MPAC + 2) is shifted left X places, In addition turn on the overflow indicator if any ones are shifted out of the high end of the triple accumulator and lost. If the address is indexed, and a negative shift count -N results from the index operation, shift right N places,

Timing: 1.15 + .35 X ms Average

TSLC X Triple Precision Shift Left and Count

Shift C(MPAC, MPAC + 1, MPAC + 2) left until $|C(MPAC)| \geq 1/2$. Store in X a single precision integer equal to the number of shifts required. Triple precision arguments may be loaded and stored using the TP instruction as in TSRT.

Timing: 1.25 + .45N ms Average

N = number of shifts required

The following instructions cause a triple precision number to be loaded into C(MPAC, MPAC + 1, MPAC + 2) if the load indicator is on. Similarly, these instructions cause a triple precision result to be stored at the end of an equation.

TAD X Triple Precision Add

C(MPAC, MPAC + 1, MPAC + 2) plus C(X, X + 1, X + 2) replace C(MPAC, MPAC + 1, MPAC + 2). Turn on the overflow indicator if such occurs.

Timing: 1.7 ms Average

TSU X Triple Precision Subtract

C(MPAC, MPAC + 1, MPAC + 2) - C(X, X + 1, X + 2) replace C(MPAC, MPAC + 1, MPAC + 2). Turn on the overflow indicator if necessary.

Timing: 1.8 ms Average

The following operations cause a double-precision vector to be loaded into VAC if the load indicator is on. Similarly, they cause a DP vector to be stored at the end of an equation.

VAD X Vector Add

C(VAC, ..., VAC + 5) + C(X, ..., X + 5) replace C(VAC, ..., VAC + 5). Turn on the overflow indicator if overflow occurs in the calculation of any component. VAD VAC will not work; use USLT 1 instead.

Timing: 2.4 ms Average

VSU X Vector Subtract

C(VAC, ..., VAC + 5) - C(X), ..., X + 5) replace C(VAC, ..., VAC + 5). Turn on the overflow indicator if necessary as in VAD

Timing: 2.5 ms Average

BVSU X Backwards Vector Subtract

C(X, . . . , X + 5) - C(VAC, . . . , VAC + 5) replace C(VAC, . . . , VAC + 5). Turn on the overflow indicator if necessary as in VAD.

Timing: 4.35 ms Average

VXV X Vector Cross Product

The cross product C(VAC, . . . , VAC + 5) * C(X, . . . , X + 5) replaces C(VAC, . . . , VAC + 5). Turn on the overflow indicator if such occurs in the calculation of any component.

Timing: 10.95 ms Average

VPROJ X Vector ,Projection

The vector projection [(VAC, . . . , VAC + 5). (X, . . . , X + 5)] (VAC, . . , VAC + 5) replaces VAC.

Timing: **9.9** ms Average

VSLT X Vector Shift Left

Shift each D P component of the vector C(VAC, . . , VAC + 5) left X places, turning on the overflow indicator if any significant bits are lost in any component. X may range from *0* to 28. If X is the negative result of an indexed address, shift right as in TSRT.

Timing: **1.2** + **1.2N** ms Average

Comments: The VSLT code must be used to double C(VAC, . . . , VAC + 5); VAD VAC will not work.

VSHT X Vector Shift Right

Shift each DP component of the vector C(VAC, ..., VAC + 5) right X places, where X may range from 1 to 28. Round each component in the process.

Timing: 3.75 ms Average

MXV X Double Precision Matrix Times Vector

The matrix C(X, ..., X + 17) premultiplies the vector C(VAC, ..., VAC + 5) leaving the result in C(VAC, ..., VAC + 5). Turn on the overflow indicator if such occurs in the calculation of any component.

Timing: 19.1 ms Average

VXM X Double Precision Vector Times Matrix

The matrix C(X, ..., X + 17) postmultiplies the vector C(VAC, ..., VAC + 5), leaving the result in VAC, ..., VAC + 5. Turn on the overflow indicator if such occurs in the calculation of any component.

Timing: 19.0 ms Average

UNIT Double Precision Vector Uniting

The vector VAC, ..., VAC + 5 is divided by its absolute value, leaving the resulting unit vector in VAC, ..., VAC + 5. If a vector has two zero components, the non-zero component is set to $1 - 2^{-28}$, as in the divide routine.

Timing: 22.75 ms Average

Comments: If not, the overflow indicator *is* turned on and C(VAC) are unspecified.

7. Miscellaneous Instructions

The following instructions do not affect loading or storing of any accumulator, nor do they change the current mode of operation. Thus, they may be placed at convenience anywhere in the equation without altering double- or triple-precision results or vector results. One exception to the above is that if a miscellaneous instruction is the last in an equation and no store address is given, an accumulator will not be stored in the push-down list.

Miscellaneous instructions must have direct addresses since the index information is used to address an index register directly rather than to use it to modify an address.

AXT, 1 X
AXT, 2 **X** Address to Index True

Load the given Polish address X directly into the specified index register, The address may be in the range - 37777 to + 37776.

Timing: **1.1** ms Average

Comments: Two operation codes are required as the index tag is incorporated in the operation code rather than the address.

AXC, 1 X
AXC, 2 X Address to Index Complemented

Load the complement of the given address into the specified index register.

Timing: **1.1** ms Average

LXA, 1 **X**
LXA, 2 **X** Load Index from the Address

The single precision word in the erasable register X are loaded into the specified index register.

Timing: 1.15 ms Average

Comments: X must be erasable or an error will result in general. If one desires to load from a fixed register AXT would do. The word at X may either be a Polish address or a single precision "bookkeeping" quantity.

LXC, 1 X
LXC, 2 X Load Index from the Address Complemented

Same as LXA only the complement of the word at Z is loaded into the specified index register.

Timing: 1.1 ms Average

SXA, 1 X
SXA, 2 X Store Index in the Address

Store the indicated index register in erasable register X.

Timing: 1.15 ms Average

XCHX, 1 X
XCHX, 2 X Index Register Exchange

Exchange the contents of the indicated index register and the erasable register X.

Timing: 1.15 ms Average

INCR, 1 X
INCR, 2 X Increment Index Register

X is interpreted as a number between - 37777 and + 37776 and added to the specified index register. Any overflows which may arise are ignored.

Timing: 1.15 ms Average

XAD, 1 X

XAD, 2 X Index Register Add from Erasable

The contents of register X are added to the specified index register, ignoring any overflows that may occur. X must he in erasable.

Timing: 1.2 ms Average

XSU, 1 X

XSU, 2 X Index Register Subtract from Erasable

The contents of X are subtracted from the specified index register, ignoring any overflows which might occur. X must be in erasable.

Timing: **1.2** ms Average

AST, **1** X

AST, **2** X Address to Step True

X is interpreted as a nurnber from -37777 to + 37776 and loaded into the specified step register.

Timing: **1.1** ms Average

TIX, 1 X

TM, 2 X Transfer to Index

If the contents of the indicated step register can be subtracted from the indicated index register without driving the index register to zero or negative, the subtraction is performed and the equation at X is begun. If not, the index register is left unchanged and no branching takes place.

Timing: **1.25** ms Average

EXIT Leave Interpretive Mode

Return to basic coding starting at the end of the current equation. The

EXIT order must be the last (or only) order in the equation in which it appears.

Timing: .95 ms Average

RTB X Return to Basic at X

Return to basic coding at Polish address X. RTB need not be the last instruction in its equation.

Timing: 1.1 ms Average

Comments: Using the RTB instruction, one can call a basic subroutine without leaving the interpreter. Thus, if the subroutine exists with TC **DANZIG** the interpretive order immediately following the RTB order is executed. In this case one need not return as before:

```
TC      INTPRET
ITCQ    0
```

See program section "RTB Routines" for details and examples.

NOLOD No Load

Turn off the load indicator and continue.

Timing: .a5 Average

Comments: This can be used when the accumulator contains the desired quantity after having stored it at the end of the previous equation.

ITA X Interpretive Transfer of Address

C(QPRET) are stored in register X.

Timing: 1.0 ms Average

ITCI X Interpretive Transfer Control Indirect

Begin the equation starting at the Polish address stored at X.

Timing: .9 ms Average

Comments: These two instructions enable the interpreter to save, and later use, the return address when a called subroutine performs and ITC instructions.

SWITCH X Switch Bit

Complement switch X, where $1 \leq X \leq 15$, $17 \leq X \leq 31$, or $33 \leq X \leq 47$.

Timing: 1.4 ms Average

TEST X, Y Test Bit

Test switch X (X defined above) and if zero, begin the equation at Y. If one, continue as usual. The two address occupy two consecutive AGC words.

Timing: 1.4 ms Average

ITCQ Return

Begin the equation whose address is in QPRET. This is an interpretive return from subroutine.

Timing: 1.15 ms Average

LODON Load Indicator On

Turn the load indicator on without storing MPAC or **VAC.**

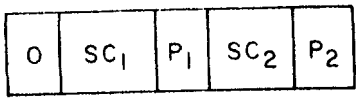
Timing: 1.15 ms Average

8. R 001 YUL SYSTEM FORMATS
R 002 --- ------ -------

	003	(SYMBOL) OPCOD	N	TO BEGIN AN EQUATION. HERE, N = NUMBER
A	004			OF ADDITIONAL OPERATOR WORDS AND MUST BE
A	005			LESS THAN 2560 SYMBOL IF DESIRED
	006	OPCOD	OPCOD	GENERAL FORMS OF OPERATOR WORDS. AN
	007	OPCOD*	OPCOD	ASTERICK DENOTES AN INDEXED OPERAND
	008	OPCOD	OPCOD*	ADDRESS OR IN THE CASE OF UNARY OPER-
	009	OPCOD*	OPCOD*	ATIONS AN INDEXED LOAD ADDRESS. OPCOD IS
	010	OPCOD		ANY OPERATION SUCH AS DAD, TSRT, UNIT,
	011	OPCOD*		OR AXT,1.
	012		ADDRESS	POLISH OPERAND ADDRESSES. THESE MAY TAKE
	013	P	ADDRESS	ON VALUES FROM +37777 TO -37776.
	014		SADDRESS,T	INDEXED POLISH OPERAND ADDRESSES. T IS
	015	P	SADDRESS,T	AN INDEX REGISTER TAG, 1 DENOTING X1 AND
A	016			2 X2. SADDRESS IS ANY ADDRESS LESS THAN
A	017			20000 WITH POSITIVE SIGN IMPLIED
	018	STORE	EADDRESS	STORE ADDRESSES. EADDRESS IS AN ADDRESS
A	019			IN ERASABLE - IE, LESS THAN 2000
	020	STORE	EADDRESS,T	INDEXED STORE ADDRESS
A	021			IT IS RECOMMENDED FOR CLARITY THAT A 2
A	022			BE PUNCHED IN COLUMN 8 OF STORE ADDRESS
A	023			CARDS SO THEY WILL DOUBLE-SPACE AFTER
A	024			PRINTING.

9. FORMATS WITHIN AGC WORDS

OPERATOR WORDS

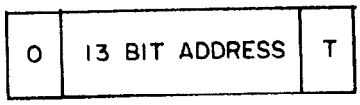


SC: SELECTION CODE
P: PREFIX BITS

ERECT OPERAND ADDRESS

RANGE:
−37776, +37777

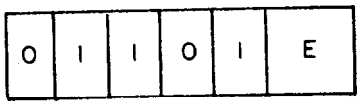
INDEXED OPERAND ADDRESS



T = INDEX TAG

RANGE: 0,+17777

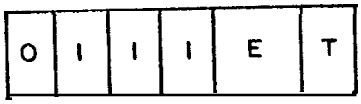
DIRECT STORE ADDRESS



E=ERASABLE ADDRESS

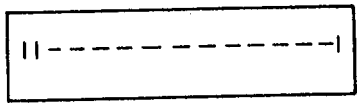
RANGE: 0, + 1777

INDEXED STORE ADDRESS



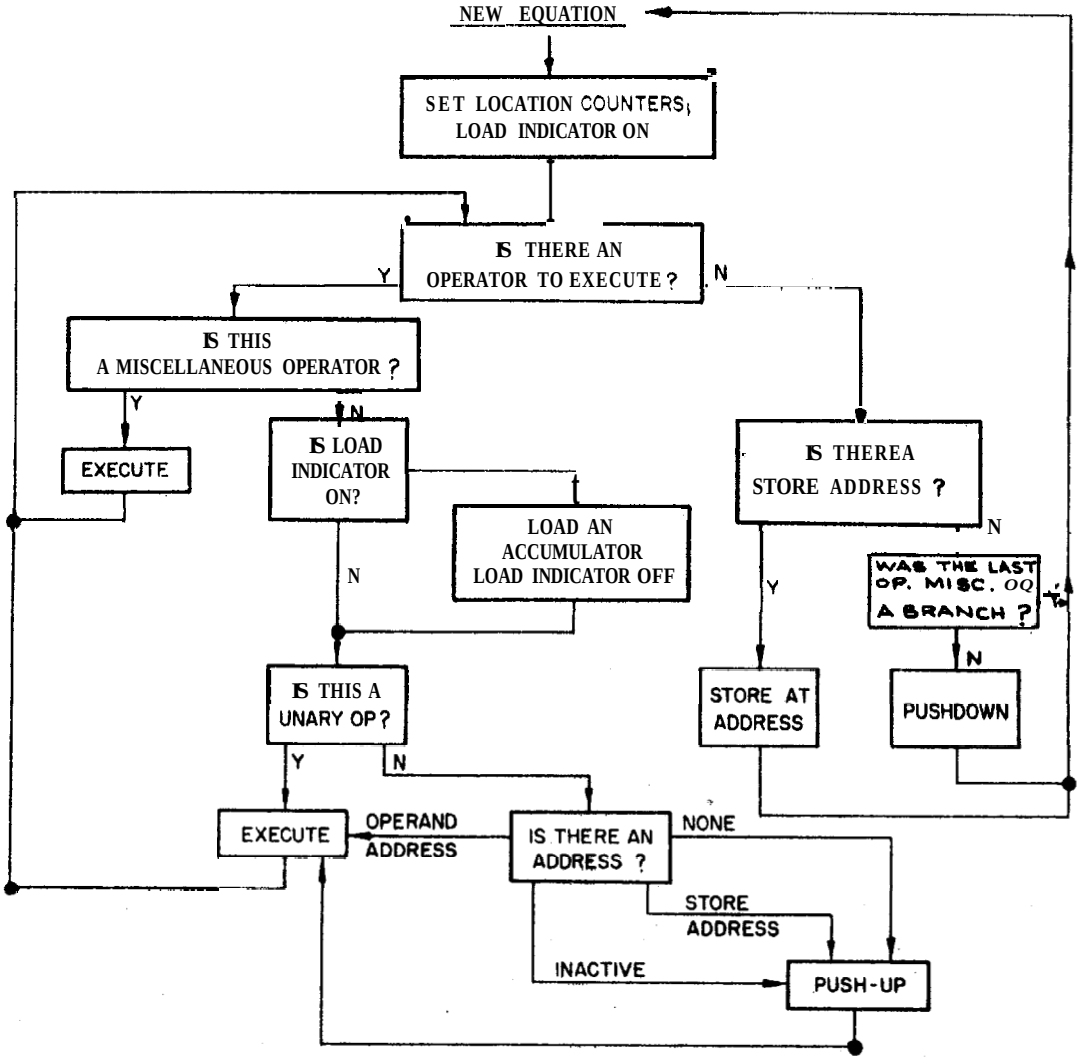
RANGE: 0,+ 1777

INACTIVE ADDRESS



Polish addresses are stored as positive numbers, with one added to them. Operator words are complemented with one added to them, however. In general, then, operand addresses are positive algebraicly and operators negative so that they can be distinguished from each other. A one is added since words are partially decoded by a CCS instruction; the CCS subtracts one from the word on execution. Note that any integers to be loaded into index registers must be Polish addresses and not DEC or OCT addresses.

FLOW CHART FOR DISPATCHER



EXECUTIVE

All AGC programs operate under control of the Executive routine except those which are executed during the interrupt mode. Executive controlled programs are called "jobs" in contradistinction to so-called "tasks" which are controlled by the Waitlist routine and are completed during interrupt time. The functions of the Executive are to control priority of jobs, to permit time sharing of erasable storage and to maintain the "Computer Activity" display.

1. Priority Control of Jobs

The functions of the Executive routine are four fold: initiation of job control, programmed interruption of a job, programmed suspension and resumption of a **job,** and termination of a job. They are briefly described in the following,

a. Initiation of Job Control - A job may be initiated (usually during an interrupt) in one of two ways, depending on the type of job. If a job contains any interpretive sections or if it is in basic language requiring a large amount of temporary storage (as many as **40** words) the calling sequence is as follows (in interrupt or with interrupt inhibited):

L	CAF	PRIORITY
L + 1	TC	FINDVAC
L + 2	CADR	JOBCADR

where PRIORITY contains the priority of the new job in bits **14 - 10** and JOBCADR is its starting location, The first instruction executed should be in basic language. As part of the FINDVAC sequence, the interpretive overflow indicator is turned off and the push-down pointer initially set to the beginning of the assigned Work Area.

If a **job** *is* written only in basic language and does not require much temporary storage the calling sequence is identical with that for FINDVAC except that the instruction at **L + 1** should be TC NOVAC. If the new job has the highest active priority **(i.*e.*,** it has the largest number in bits **14 - 10** of

its priority word) its execution will begin within 20 ms as explained in the next section.

b. Programmed Interruption of a Job - All programs under executive control must examine register NEWJOB periodically to see if a job of higher priority is awaiting execution. NEWJOB is examined at least every 20 ms with an average of about 10 ms between interrogations. The interrogation sequence is as follows:

```
L       CCS     NEWJOB
L + 1   TC      CHANG 1
```

The contents of NEWJOB are positive non-zero if a job of higher priority is present and + 0 if not. If the current job is interrupted in this manner, control will be returned to L + 2 when the interrupted job again becomes the job of highest active priority.

Interrogation of NEWJOB is performed by the Interpreter when running in the interpretive mode. NEWJOB is examined roughly every interpretive instruction in this mode.

b. Programmed Suspension and Resumption of a Job - When a job is geared to the occurrence of certain external events (e. g. until the completion of an input-output operation) and must wait a period of time until the event occurs, it may be suspended or "put to sleep". Temporary storage is left intact through the period of inactivity. The calling sequence is as follows:

```
L       CAF     WAKECADR
L + 1   TC      JOBSLEEP
```

where **WAKECADR** contains the complete address to which the job will return when it is awakened. Active jobs of lower priority are permitted to run while a higher priority job is "sleeping".

When the anticipated event occurs the inactive, or "sleeping" job is "awakened", as follows (in interrupt or with interrupt inhibited):

```
L       CAF     WAKECADR
L + 1   TC      JOBWAKE
```

Control is returned to L + 2 after job activation; if the job has the highest priority of all active jobs under executive control, it will be resumed after the next NEWJOB interrogation.

If the JOBWAKE sequence is executed, but the suspended job is not to be found (already awake or not put to sleep) control is returned to L + 2 with no action taken. If C(LOCCTR) are positive non-zero, then a job was awakened; otherwise C(LOCCTR) = + 0. This feature allows the first of several events to awaken a job without prior knowledge of which event will occur first.

d. Termination of a Job - To terminate a job the following calling sequence is executed:

```
L       TC      ENDOFJOB
```

This sequence releases the erasable memory allocated to this job and scans for the active job of highest priority to which control is given. Note that there is always an active job to be performed. In the absence of all others the "Dummy Job" of priority 1 is executed. The Dummy Job is the only job which never goes to ENDOFJOB.

2. Time Sharing of Erasable Storage

The erasable memory allocated by the Executive routine may be classified as: Job Register Sets, Work Areas (sometimes referred to as VAC areas), and Executive Temporaries.

a. Job Register Sets - Every job has an associated 8-register Job Register Set. This set contains, in a packed format:

The priority of the job
Location of the assigned Work Area (if used)
The three register multiple precision accumulator
Location for starting, resuming, or awakening
Various frequently referenced Interpretive Parameters e. g. overflow indicator, push-down pointer, etc.

Eight such sets are provided allowing a total of seven jobs and the Dummy Job to be simultaneously under executive control. The job presently being

executed is in the first of these eight sets. During programmed interruption, suspension, or termination of a job, the job being readied for execution has its Job Register Set moved to the top of the stack. In the case of programmed interruption or suspension, the Job Register Set of the job being removed is placed in one of the seven register set storage areas for future recall. Note that the physical movement of register sets allows access without the need of indexing.

b. Work Areas - Jobs initiated by the FINDVAC sequence are assigned a 43 word Work Area. Any job which contains interpretive sections must have such an area assigned. There are five Work Areas available which means that up to five interpretive jobs may be simultaneously under executive control.

A non-interpretive job may use all 43 registers as temporary storage in any way required.

For interpretive jobs the Work Area is organized as follows (relative addresses):

0 - 31	Push-down list
32 - 37	Vector accumulator
38 - 39	Index registers
40 - 41	Step registers
42	Interpretive Q register (QPRET)

These Work Areas are not swapped as are Job Register Sets and are therefore not directly addressable. The Executive maintains in register FIXLOC the base address of any assigned Work Area so that basic jobs may reference the Work Area with "INDEX FIXLOC" instructions.

c. Executive Temporaries - A block of 28 temporary registers is available to programs running under the Executive. The contents of these registers are not preserved during programmed interruption or suspension as are the Job Register Sets and Work Areas but are, however, protected between interrogations of register NEWJOB. Thus, this block may be considered as short-term temporary storage.

3. Computer Activity Display

The Executive routine controls the computer activity light. This display panel light is off only when the Dummy Job is the job of highest active priority.

WAITLIST

The function of the Waitlist routine is to provide timing control for other program sections. Programs operating under control of Waitlist are referred to as "tasks" in contradistinction to "jobs". Tasks are run in the interrupt mode and are therefore of short duration, i.e. less than 4 ms. If the Waitlist is to initiate a lengthy computation, e.g, a steer law computation, then the task will contain an Executive routine call *so* that the computation itself is performed as a job.

The Waitlist program derives its timing from counter TIME3. Overflows from this counter trigger Interrupt one. On occurrence of this interrupt, control is transferred to the task or tasks which are waiting to be performed, A maximum of six different tasks can be simultaneously under control of the Waitlist.

The calling sequence, which must be made in the interrupt mode or with interrupt inhibited, is as follows

L	CAF	DT
L + 1	TC	WAITLIST
L + 2	CADR	TASKCADR

where DT is the time from now at which the task is to be performed with C(DT) given at 10 ms/bit. C(DT) must be less than 12000_{10} or 2 minutes. TASKCADR. is the CADR. of the task to be performed and return is to L+3.

Since the frequency of the TIME3 counter overflows is 100 times per second, there will be an average uncertainty of 5 ms in the timimg of a task, which is called by a program that is not itself under Waitlist control, due to lack of knowledge of the exact time of the next overflow following such a call. However, when the initiating program is part of a task, the time of the next overflow is accurately known. Thus, self-perpetuating tasks have negligible uncertainty in their repetition rate.

At the conclusion of any task, the instruction TC TASKOVER.must be executed to terminate the task, call any additional task6 which might be due and finally to RESUME sequence.

MASTER CONTROL

The Master Control program consists of subroutines to control mission programs from the keyboard, to restart mission programs after a GO sequence (parity fail, power fail, etc.) and to maintain the "Major Mode" display on the DSKY. Although the Master Control routine in SUNRISE is of a preliminary nature, the expectation is that it will provide the necessary experience to coordinate future mission programs during the evolution of AGC programs.

1. Program Control from the Keyboard

The capability to start, stop, and change the mode of mission programs manually is provided by Master Control in cooperation with Pinball (Keyboard & Display program). To initiate manual program control, "Verb 37 Enter" is executed, followed by two octal digits, **X** and ***Y***, and a second Enter. Here **X** is a pre-assigned program number (e.g., ***0*** for Prelaunch) and Y the command. If Y = 0 the program will terminate (usually within a second). The remaining values of Y **(1-7)** may be used to start the program in different modes or alter its mode if already running. The function associated with these values **(1-7)** is assigned by the program and varies among mission programs.

If the referenced program is not operating when the command is entered, it will be started in the mode indicated by digit ***Y;*** if running, it will be honored shortly after the second "Enter". The response time in this case varies from program to program, but is usually less than 1 second.

In SUNRISE, two programs are run under Master Control: Prelaunch Alignment and Orbital Integration. The manual options available with each program are explained with the respective program description.

2. Mission Program Restart After GO

A GO Sequence results from the detection of such malfunctions as a parity failure, a TC trap, a counter failure, an interrupt lock or a power failure. The effect, from a programmer's point of view, of a ***GO*** sequence is to

a) Set output registers to zero

b) Knock down interrupt-in-progress flip-flops and clear interrupt chain

c) Do a TC 2030 .

GO cannot be inhibited by interrupt in progress or INHINT.

To implement restart of mission programs from a transient failure of the type cited, each such program is divided into as many as thirty steps or "phases". A table containing the current phase of all mission programs is kept by Master Control so that in the event of a GO sequence, these programs (if running) may be restarted at the beginning of the appropriate phase (not necessarily the phase in which the program was operating when the GO sequence occurred). This table, known as the Phase Table, is stored in triplicate and mission programs are restarted only if all three copies agree. If any disagreement is found in the Phase Tables, a "Fresh Start" is executed with no restart of mission programs.

It should be noted that Master Control does not itself effect the restart of the programs that were operating. Instead, there is associated with each program under Master Control the complete address of a routine which, using the phase information, will restart the program. This restart routine is considered a part of the mission program with which it is associated.

The Phase Table may be read or altered by major mission programs using a set of subroutines provided by Master Control. To sample the phase of a program running under Master Control, the calling sequence is

```
L       TC     GETPHASE
L + 1   OCT    OOOOX
```

where X is the program number as explained in subsection 1. The requested phase appears in bits 1-5 of the A register. Return is to L+2 if a terminate request has been given (e.g. by a keyboard command) and to L+3 otherwise.

To change the phase of a program, the following subroutine is provided (called under Executive control only):

```
L       TC     PHASCHNG
L + 1   OCT    OZZOX
```

where X is as above and ZZ is the new phase number. The number of the previous phase (or one which was keyed in) appears in the low 5 bits of MPAC. Return is to L+2 or L+3 as in PHASCHNG.

To terminate a program running under Master Control the following sequence is executed instead of TC ENDOFJOB

```
L       TC      MAJEXIT
L + 1   OCT     OOOOX
```

This, of course, must be performed under Executive control and is actually a prologue to ENDOFJOB.

3. Major Mode Display

Master Control is the only program which services the Major Mode Display. SUNRISE contains three major programs: Orbital Integration, Pre-launch Platform Alignment, and System Test. Considering the 2 digit major mode display as a six bit binary number, the identification is bit one, two and three, respectively. Thus, if Prelaunch Alignment is operating, the display will read 01. For Orbital Integration the designation is 02 and for System Test, 04. If more than one of these programs is operating, the display will contain the "or" of the activity bits; that is, for Integration and Prelaunch running together, the display will read 03.

FRESH START AND RESTART

The restart portion of this program section is described under program section "Master Control". The fresh start function is performed is response to a keyboard entry or if, after a GO sequence, the restart program finds disagreement in the phase table.

The purpose of Fresh Start is to initialize most programs. In particular, Executive and Waitlist are cleaned out; the Phase Table is cleaned out; all relays under computer control are knocked down; T4RUPT is initialized; DOWNRUPT is initialized; several Keyboard and Display registers are initialized; and counters TIME 3 and TIME 4 are reset.

Theoretically, Fresh Start need be performed only once, specifically, when the computer is first turned on.

DOWNRUPT PROCESSOR

The DOWNRUPT processor is initiated on receipt of an interrupt from the NAA down-telemetry programmer. This device sends pulses *to* the interrupt chain at a rate of either 10 pps or 50 pps. For either telemetry rate, the downlink information has the following characteristics:

1. The list of general AGC erasable registers (data words) includes thirty or more registers, with different lists for different mission phases. No words are omitted in a complete pass through the list.

2. An ID word appears every four data words.

3. Relay words (AGC relay manipulations) and Input Character Words (Keyboard, Mark, and Uplink information) are telemetered whenever they occur and are freely interspersed among the data words. A relay word change is sent down in the next available telemetry cycle. An Input Character word is handled the same way provided no relay word has taken precedence. Note that at either *50* pps or **10** pps all relay changes will be telemetered.

Since relay banks are switched no more frequently than once every **120** ms, only nine relay word changes can occur per second, Likewise, because of the up-telemetry rate, no more than ten input characters can occur per second. Accordingly, at least **30** Data/ID words will be sent in each second at the 50 pps telemetry rate.

The ,down-telemetry format and the decoding and program logic are given in the following:

ID Word Format

Word Order bits	=	*0*
Bits 15-11	=	*00000*
Bits 10-1	=	Data Index

Data Word Format

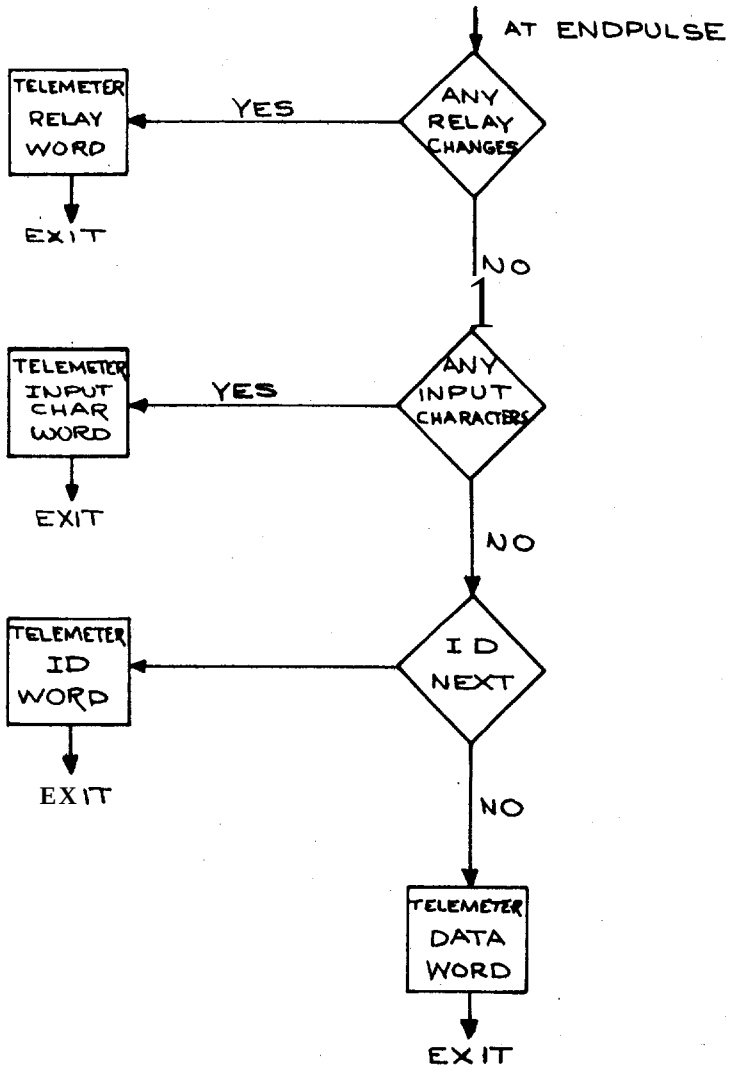
Word Order bits	=	1
Bits 15-1	=	Data

Relay Word Format

Word Order bits	=	*0*
Bits 15-12	=	Relay Word Address $\neq$ *0000*
Bits 11-1	=	Relay Settings

Input Character Word Format

Word Order Bits	=	*0*
Bits 15-12	=	*0000*
Bit 11	=	1
Bits 10-8	=	Unused
Bit 7	=	**MARK** Button
Bit 6	=	0 for KBD 1 for UPLINK
Bits 5-1	=	Keyboard or UPLINK character.


AT ENDPULSE
ANY RELAY CHANGES
YES
TELEMETER RELAY WORD
EXIT
NO
ANY INPUT CHARACTERS
YES
TELEMETER INPUT CHAR WORD
EXIT
NO
I D NEXT
TELEMETER ID WORD
EXIT
NO
TELEMETER DATA WORD
EXIT

T4RUPT OUTPUT CONTROL

Constant and relatively high-rate output control is needed to operate several G&N subsystems. The following functions must be serviced at this constant rate:

1. The 14 banks of R and C relays may he switched one bank at a time with about 20 ms allowed for the latching to take place. The driving circuits are de-activated at the end of this time. One bank may be switched each 120 ms.

2. An IMU CDU may be driven (if required) every 60 ms, so that each IMU CDU is sampled about 5 times per second.

3. An optics CDU may be driven (if required) every 480 ms, so that each optics CDU is sampled about once per second.

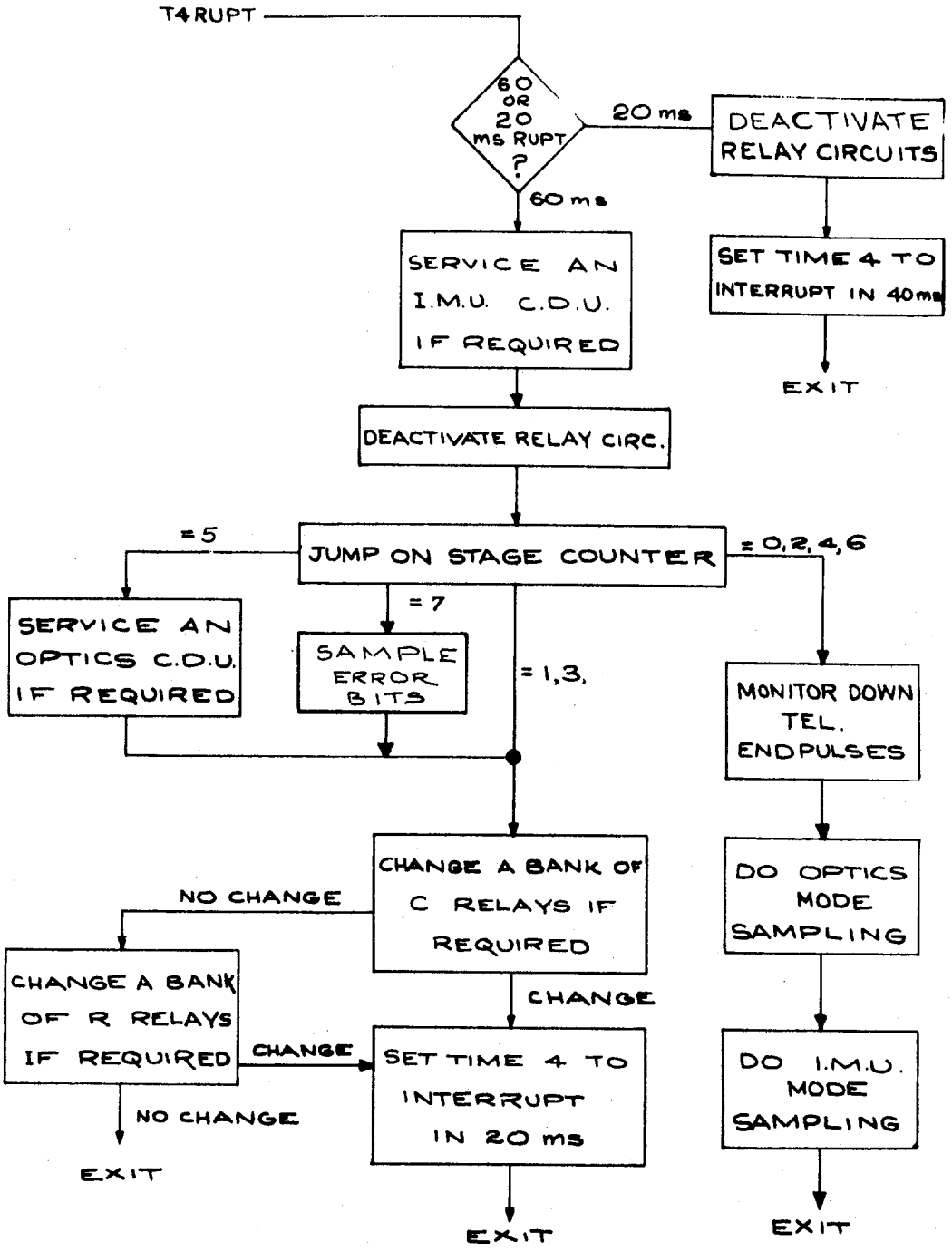
4. IMU Mode switches are sampled every 120 ms and appropriate action is taken upon receiving a mode change indication. This program also verifies that computer-requested mode changes take place successfully.

5. Optics Mode switches are sampled every 120 ms in the same manner as IMU Mode sampling.

6. The presence of Down-Tel Endpulses is verified every 120 ms and action is taken so that DOWNRUPT may detect the presence of an abnormally high rate of Endpulses.

7. The IMU fail, CDU fail, and PIPA fail signals are sampled each 480 ms. An alarm is triggered or not depending on the mode of the IMU.

To implement this, overflow from the preset counter TIME4 triggers interrupt 3 (T4RUPT) every 60 ms. However, if a relay change is requested, TIME4 is set to interrupt in 20 ms. The relay driving circuits are then de-activated and TIME4 is set to interrupt in 40 ms, reestablishing the normal 60 ms pattern. The allocation of tasks each 60 ms interrupt is shown in the accompanying flow chart.


T4RUPT
60 OR 20 ms RUPT ?
20 ms
DEACTIVATE RELAY CIRCUITS
60 ms
SERVICE AN I.M.U. C.D.U. IF REQUIRED
SET TIME 4 TO INTERRUPT IN 40ms
EXIT
DEACTIVATE RELAY CIRC.
=5
JUMP ON STAGE COUNTER
=0,2,4,6
=7
=1,3,
SERVICE AN OPTICS C.D.U. IF REQUIRED
SAMPLE ERROR BITS
MONITOR DOWN TEL. ENDPULSES
CHANGE A BANK OF C RELAYS IF REQUIRED
NO CHANGE
CHANGE
DO OPTICS MODE SAMPLING
CHANGE A BANK OF R RELAYS IF REQUIRED
CHANGE
SET TIME 4 TO INTERRUPT IN 20 ms
DO I.M.U. MODE SAMPLING
NO CHANGE
EXIT
EXIT
EXIT

1. Relay Driving

The state of the 11 banks of R relays is stored in registers DSPTAB ,through DSPTAB + 10D; the state of the three banks of C relay is stored in DSPTAB + 11D through DSPTAB + 13D. If a change is to be made in bank N of R relays, the correct setting of the relays is placed in bits 1 - 11 of DSPTAB + N - 1 and the entire word complemented to show a change must be made. The contents of register NOUT must be incremented by 1, showing the number of banks of R relays left to be changed. NOUT is decreased by one when DSPOUT (a section of T4RUPT) changes a bank of R relays. Thus, a zero contents of NOUT indicated no changes are to be made.

Changes in the C relays are indicated in a different manner. The word has a one in bit 15 but the state of the relays is stored direct, not complemented. No register such as NOUT need be incremented.

If a relay bank is changed, the relay word is stored in DISPBUF to be telemetered the next TM cycle. DOWNRUPT then sets DISPBUF to + 0 and telemeters the relay word. DISPBUF is then ignored until it becomes non-zero again.

2. IMU CDU Driving

The status of the IMU CDUs are indicated by register CDUIND and are as follows:

C(CDUIND) = 0, 1, or 2 indicates the AGC is driving, respectively, the x, y, or z IMU CDU.

C(CDUIND) < - 0 indicates the AGC is not driving these CDU's but that they are reserved; i.e., they should not be driven. This setting exists during fine align, for example.

C(CDUIND) = - 0 indicates the IMU CDU's are in the "available" state, i.e., they are not being driven by the AGC and are not reserved.

If the AGC is driving these CDU's each is serviced every 180 ms. Three desired angle are supplied in registers THETAD through THETAD + 2. These desired angles must be supplied in 2's complement arithmetic; i.e., + 0 ≠ - 0, conforming to the special CDU counter arithmetic which makes all 2^{15} counter states algebraically distinct., Two gains should be supplied in registers KG and KH

The command to a CDU ($\Delta\theta$) is computed as follows:

$$\Delta\theta_N = KG\,(\theta_D - \theta_A) + KH\,(\Delta\theta_{N-1})$$

where θ_D is the desired angle and θ_A is the value of the CDU counter when the command was generated. Note that there is an error of as many as KG/1 bits in the actual angle as compared to the desired angle.

3. Optics CDU Driving

The status of the optics CDU's is indicated by OPTIND, following the same conventions as CDUIND. Here, the positive values are 0 and 1, for OPTX and OPTY. Desired angles should be supplied in registers DESOPTX and DESOPTX + 1 in 2's complement arithmetic. Here, however, the gain is always 1 and the command is computed as

$$\Delta\theta = \theta_D - \theta_A$$

4. IMU Mode Sampling (KSAMP)

a. Computer Commands - When the computer is mode switching the IMU, the expected status of bits 1 - 7 in register IN 3 is left in erasable location DESKSET (Desired K Setting). The mode sampling program makes sure that IN 3 and DESKSET are the same or are going to be the same and passes the information back to the main program via erasable location WASKSET (Was K Setting) as follows:

C(WASKSET)	> + 0	Normal operating conditions = [C(DESKSET)]
	< - 0	Astronaut command to ignore output of mode sampling
	□ + 0	System is about to switch
	= - 0	System has failed

b. Manual Commands - The computer monitors the mode the IMU is presently in and sets the appropriate C relays so that if the TRNSW is set

to computer, the mode will remain unchanged. In addition to this, the computer cooperates with a manual request to zero encoders. The flow of this program is presented in the attached diagram.

5. Optics Mode Sampling (OPTSAMP)

a. Computer Commands - When the computer is mode switching the optics the expected status of bits 10, 12 - 14, of IN 3 is left in erasable location DESOPSET (Desired Optics Setting). The mode sampling program makes sure that IN 3 and DESOPSET agree or otherwise takes appropriate action as indicated in the accompanying logic flow diagram.

b. Manual Commands - The action taken here is almost identical with the IMU case. (See flow diagram).

6. Down Telemetry Monitor

Each 120 ms erasable location TELCOUNT is set to 7. The DOWNRUPT program decrements this number at each interrupt time. Before the location is set to 7 a check is made that it is currently less than 7 *so* that at least one Endpulse has been received in the last 120 ms. If this is not the case then the TM Fail light is turned on,

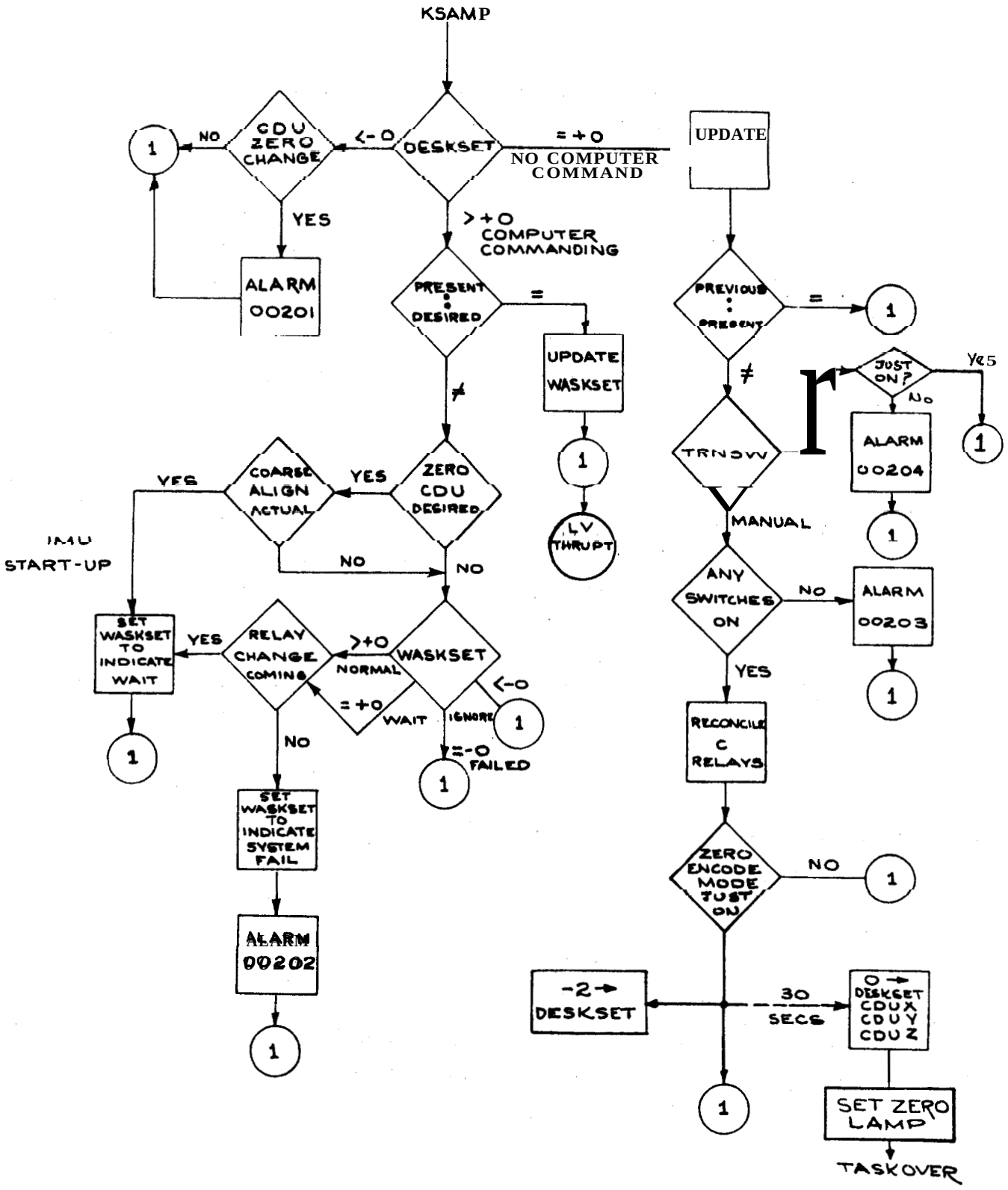
The DOWNRUPT program itself makes sure that the TELCOUNT does not pass through zero. If it does, then the TM Fail light is turned on and the BLOCK ENDPULSE bit is set. This condition corresponds to a rate in excess of 56 Endpulses/sec.

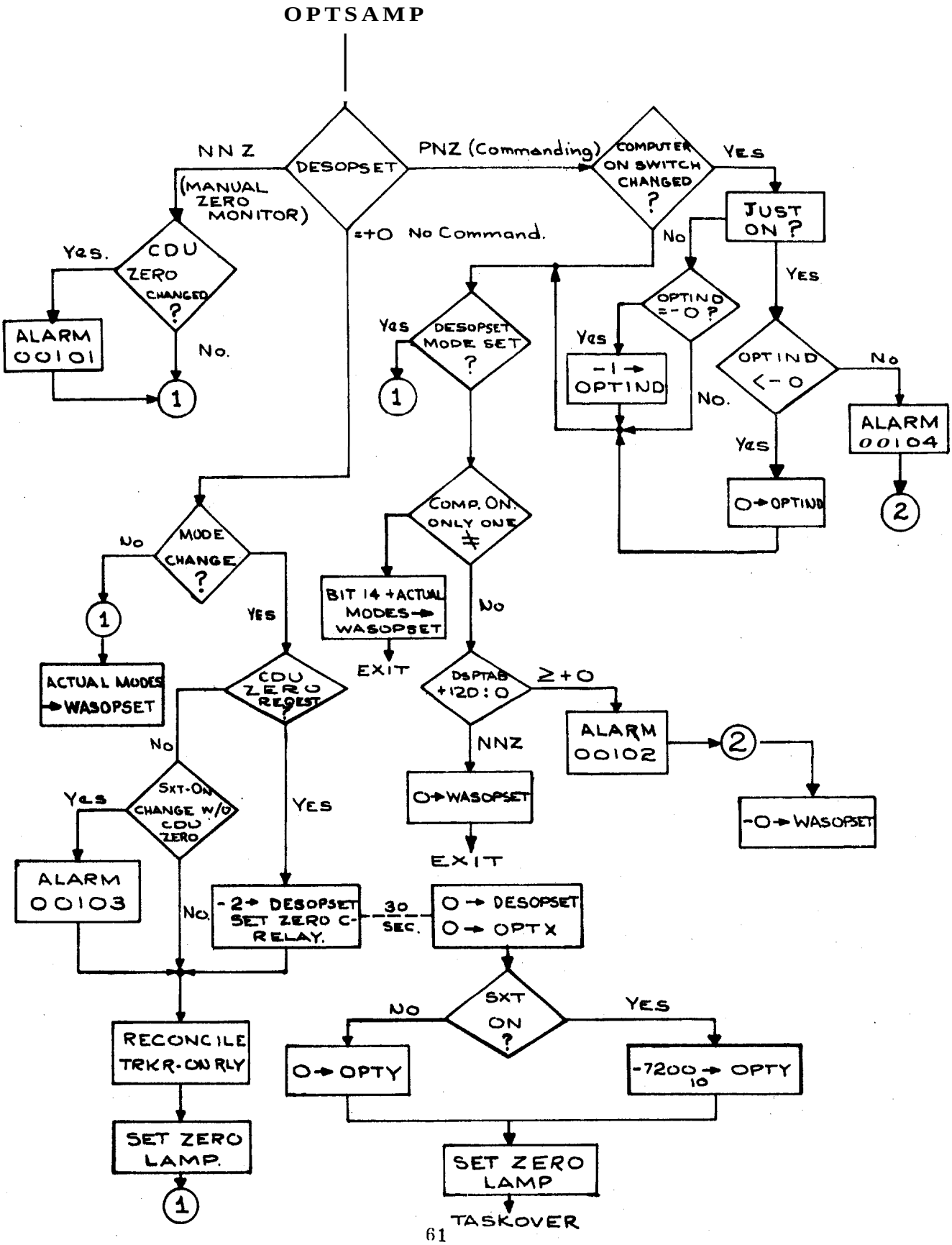
7. Failure Signal Monitor

Each 480 ms the CDU Fail, PIPA Fail, and IMU Fail signals (Bits 10 - 12 of IN 2) are examined. If any of them has just become a one, the corresponding light on the IMU panel is energized subject to the following provisions:

a) CDU Fail lamp is not lit in Zero Encode Mode
b) IMU Fail lamp is not lit in Coarse Align Mode

Mission programs can monitor this status of these signals by inspecting C(OLDERR) which is non-zero if a failure has occurred. The failure monitor can be bypassed by setting C(OLDERR) < - 0.


KSAMP
DESKSET
<-0
=+0
NO COMPUTER COMMAND
>+0 COMPUTER COMMANDING
CDU ZERO CHANGE
NO
YES
1
ALARM 00201
PRESENT : DESIRED
=
≠
UPDATE WASKSET
1
LV THRUPT
ZERO CDU DESIRED
YES
COARSE ALIGN ACTUAL
YES
NO
NO
IMU START-UP
SET WASKSET TO INDICATE WAIT
1
RELAY CHANGE COMING
YES
>+0 NORMAL
=+0 WAIT
WASKSET
<-0 IGNORE
=-0 FAILED
1
1
NO
SET WASKSET TO INDICATE SYSTEM FAIL
ALARM 00202
1
UPDATE
PREVIOUS : PRESENT
=
1
≠
JUST ON?
Yes
No
ALARM 00204
1
TRNOVV
MANUAL
1
ANY SWITCHES ON
NO
ALARM 00203
1
YES
RECONCILE C RELAYS
ZERO ENCODE MODE JUST ON
NO
1
-2→ DESKSET
30 SECS
0→ DESKSET CDUX CDUY CDUZ
1
SET ZERO LAMP
TASKOVER


OPTSAMP
DESOPSET
NNZ
(MANUAL ZERO MONITOR)
PNZ (Commanding)
=+0 No Command.
COMPUTER ON SWITCH CHANGED ?
YES
JUST ON ?
YES
No
CDU ZERO CHANGED ?
Yes.
No.
ALARM 00101
1
DESOPSET MODE SET ?
Yes
OPTIND =-0 ?
Yes
-1 → OPTIND
No.
OPTIND <-0
No
Yes
ALARM 00104
2
O→OPTIND
MODE CHANGE ?
No
YES
ACTUAL MODES →WASOPSET
COMP. ON: ONLY ONE ≠
BIT 14 +ACTUAL MODES→ WASOPSET
EXIT
No
DSPTAB +120 : 0
≥+0
NNZ
ALARM 00102
O→WASOPSET
EXIT
-O→WASOPSET
CDU ZERO REQEST ?
No
Yes
SXT-ON CHANGE W/O CDU ZERO
Yes
ALARM 00103
No.
-2→ DESOPSET SET ZERO C-RELAY.
30 SEC.
O → DESOPSET
O → OPTX
SXT ON ?
NO
YES
O→ OPTY
-7200 10 → OPTY
RECONCILE TRKR-CNTRLY
SET ZERO LAMP.
SET ZERO LAMP
TASKOVER

MODE SWITCHING AND MARK

The following set of AGC program is used to select various modes of the IMU and Optics, and to control use of these sub-systems. The following routines currently comprise this set:

IMUZERO	Zeros IMU CDU's
IMUCOARS	Coarse Aligns the IMU
IMUFINE	Switches IMU to Fine Align Mode
IMUATTC	Switches IMU to S/C Attitude Control Mode
IMUREENT	Switches IMU to Re-entry attitude Control
IMULOCK	Locks IMU CDU's
IMURECOR	Re-coarse Align the IMU (entered from Fine Align)
IMUSTALL	Waiting Routine used in conjunction with all of above IMU routines
OPTZERO	Zeros Optics CDU's
OPTCOARS	Coarse Aligns Optics (no automatic fine align)
OPTTRKON	Optics Star Tracker On
SCTMARK	Requests N Scanning Telescope Marks
SXTMARK	Requests N Sextant Marks
MKRELEAS	Releases Mark System
OPTSTALL	Waiting Routine used in conjunction with all of the above optics routines
CDUINC	Increments a 2's complement number by a desired amount
IMUPULSE	Pulse torques the IMU a specified amount

The appropriate mode switching routine is called by

```
L       TC BANKCALL
L + 1   CADR (SUBRO)
```

following which it is possible to continue useful computation while the mode switch proceeds independently. Eventually one must effect the following call before subsequent IMU (OPTICS) Mode Switching:

```
L       TC BANKCALL
L + 1   CADR IMUSTALL (OR OPTSTALL)
L + 2   TC (Error)
```

The appropriate stall routine will return control to L + 3 on successful completion of the mode switch. If waiting is required, the main program is put to sleep (with VAC protection, as usual). On completion of the mode switch, detection of any unwanted error signal results in return of control to L + 2 where appropriate action may be taken.

The Mark and Mark release routines are used in conjunction with the optics system. The calling sequence

```
L       CAF N(≤ 6)
L + 1   TC BANKCALL
L + 2   CADR SCTMARK (or SXTMARK)
```

prepares the computer to accept N depressions of the MARK button by the astronaut. This program sets up a VAC area for storage of up to 6 accepted MARKS.

Eventually the main program must command

```
L       TC BANKCALL
L + 1   CADR OPTSTALL
L + 2   TC Error
```

Return is to L + 3 after N marks have been accepted. The main program is put to sleep while waiting for all Marks to be made. Unwanted OPTICS Mode changes will result in a return of control to L + 2 where appropriate action may be taken.

On completion of MARK data processing the main program must command

```
L       TC BANKCALL
L + 1   CADR MKRELEAS
```

to release the VAC area and free the optics system.

The Mode Switching and Mark subroutines will now be described in functional outline' form.

1. IMUZERO

a. CDUIND is tested. If it contains +0, then the IMU is considered available. Otherwise, alarm 301 is fired and the job is terminated. The IMU is declared not available by setting C(CDUIND) = -1 to lock out IMU usage by all other programs.

b. The C relay output table is set to switch the IMU mode. C(DESKSET) is set to allow checking of the mode switch by T4RUPT. The zero-encode light is lit.

c. A WAITLIST call is made for a 30 second delay.

d. On expiration of the WAITLIST call, the CDU counters are zeroed.

e. Executes the ENDIMU sequence to service the main program (see IMUSTALL).

f. Tests for CDU failure indication and, if present, lights the CDU Fail lamp and sets the IMUSTALL routine for an error return.

2. IMUCOARS

The desired CDU angles must be left in THETAD, THETAD + 1, and THETAD + 2 and desired gains in KG and KH before going to IMUCOARS.

a. The C relay output table is set to switch the IMU mode and C(DESKSET) is set to allow checking of the mode switch by T4RUPT. The zero encode light is serviced (turned off unless the OPT CDU's are being zeroed).

b. A 200 ms WAITLIST call is made to allow sufficient time for relay closure and verification by T4RUPT.

c. On completion of the above C(WASKSET) is tested to verify successful mode switch. The T4RUPT IMU CDU drive is enabled by setting C(CDUIND) = 0.

d. A 30 second WAITLIST call is made to allow sufficient time for the CDU loops to settle.

e. Upon expiration of the WAITLIST call, the ENDIMU sequence is executed. A test is made for CDU failure indication and if present, the IMUSTALL routine is set for an error return.

3. IMUFINE

a. The C relay output table is set to switch the IMU mode. C(DESKSET) is set to allow checking of the mode switch by T4RUPT.

b. A 90 second WAITLIST call is made to allow sufficient time for the mode switch and gyro re-centering. The T4RUPT CDU drive routine is disabled by setting C(CDUIND) = -1, the reserved but inactive state.

c. On expiration of the WAITLIST call, the ENDIMU sequence is executed. A test is made for the presence of IMU or CDU failure indications and if present, the appropriate lamp(s) are lit. IMUSTALL is set for error return.

4. IMUATTC

The desired CDU angles and CDU loop gains must be set before entering this mode as in Coarse Align.

a. The C relay output table is set to switch the IMU mode. C(DESKSET) is set to allow checking of the mode switch by T4RUPT.

b. A *200* ms WAITLIST call is made to allow sufficient time for relay closure and verification by T4RUPT. CDU activity is disabled by setting C(CDUIND) = -1.

c. Upon expiration of the WAITLIST call, the T4RUPT CDU drive is enabled by setting C(CDUIND) = *0*.

Note: No check is made of IMU and CDU failure indications since, in this mode, the main program must check the status of these signals periodically.

5. IMUREENT

This routine functions in an identical manner to IMUATTC.

6. IMULOCK

At expiration of the Waitlist call, the ENDIMTJ sequence is executed, The rest of the routine is identical to IMUCOARS from paragraph C.

7. IMURECOR

At expiration of the WAITLIST call the desired CDU angles THETAD, +1, +2 are set to the actual CDU angles. The rest of the routine is identical to IMUCOARS from paragraph c.

8. IMUSTALL

a. C(IMUCADR) is tested and if found to be > + 0 or < -1, the result is Alarm 301 and termination of the job. A +0 indicates mode switching is incomplete and the job is put to sleep with the Wake Address in IMUCADR. A -1 indicates the switch has been successfully completed and a -0 indicates failure.

b. If the switch has taken place, successfully or not, C(IMUCADR) is set to +0 and return is made to L + 2 or L + 3 for failure or success.

On entrance to the LMUSTALL ENDIMU sequence from a mode switching program, C(IMUCADR) is tested. If ≠ + 0, a job had been put to sleep and so is awakened. If = + 0 (the only other possibility), no job need be awakened. Furthermore, C(WASKSET) is checked for successful completion of the mode switch. If unsuccessful, then exit from IMUSTALL will be to L + 2 rather than L + 3. If no job was awakened, C(IMUCADR) is set to -1 for successful completion and to -0 for failure.

9. OPTZERC
10. OPTCOARS
11. OPTTRKON

These routines are not available at this time due to a lack of sub-system interfaces. They will be included in later programs. These functions can be accomplished manually for the time being.

12. SCTMARK

13. SXTMARK

a. C(MARKSTAT) is tested and if found to be +0, the Mark button is available. If > + 0, MARK button is busy, Alarm 105 is fired and the job terminated.

b. An idle VAC area is found. If none is available, Alarm 104 is fired and the job aborts, The address of the idle VAC area is placed in MARKSTAT and the VAC area is declared busy.

c. The desired SXT-ON switch setting (1 for SXTMARK, *0* for SCTMARK) and the number of Marks desired are stored in MARKSTAT as follows:

Bits	
1 - 9	VAC Area Address
10	= *0* Initially and after every MK ACCEPT = 1 if MARK since last MK ACCEPT
11	SXT-ON Switch Desired State
12-14	Number of MK ACCEPTS wanted
15	= 0

These states exist while MARKS are being accepted. After acceptance of the N^{th} MARK, bits **10-15** are set to zero.

d. Another job is requested via the executive to "paste up" verb **51** with the flash on to indicate that a MARK is wanted. The flash is turned off after the N^{th} Mark has been accepted.

When a MARK interrupt is received, the following registers are read in the order listed and stored as indicated relative to the next register available in the VAC area:

OPTICS **Y**	VAC_n + 7K + 5
OPTICS **X**	VAC_n + 7K + **3**
CDU **Y**	VAC_n + 7K + **2**
CDU *Z*	VAC_n + 7K + **4**
CDU **X**	VAC_n + 7K + ***6***
TIME **2**	VAC_n + 7K + ***0***
TIME **1**	VAC_n + 7K + **1**

where n specifies the VAC area being used, and K is the number of **MARKS** accepted to date. The SXT-ON switch is then checked to determine that it is in the desired position **as** stored in MARKSTAT. If it is not, then Alarm **106** is fired **and** the **MARK** information is not retained, However, if the manual ignore state is present in WASOPSET; i. e. C(WASOPSET) < *-0,* this discrepancy is ignored. Finally, bit **10** of MARKSTAT is set to **1** to

show a MARK received since last MK ACCEPT (or initialization of MARKSTAT).

When a MK ACCEPT is received (by VERB 52, Enter; soon to be replaced by a special key), the contents of MARKSTAT are tested. If zero, i.e. no REQUEST N MARKS has been done, Alarm 107 is fired and no further action is taken. On the other hand, if C(MARKSTAT) # 0 but N MARK ACCEPT's have been received or no MARK has been made since the last MARK ACCEPT, Alarm 110 is fired and no further action is taken. If no alarms are to be triggered, bit 10 of MARKSTAT is set to zero to show a MK ACCEPT since last MARK. Then index K is set to K + 1, i.e., the VAC Area pointer is moved to the next seven slots.

If more MARK ACCEPTS remain to be made, no further action is taken. Otherwise a job is initiated to turn off the flash. Control is then transferred to the ENDOPT sequence of OPTSTALL with the high six bits of MARKSTAT set to zero; i.e., MARKSTAT contains the address of the VAC Area used for MARK storage.

14. MKRELEAS

When the main program has control returned from OPTSTALL after a SCTMARK or SXTMARK and has finished processing the MARK data, the following must be performed:

```
L       TC BANKCALL
L + 1   CADR MKRELEAS
```

with return to L + 2. No call to OPSTALL is necessary. This sequence releases the VAC area and sets C(MARKSTAT) = 0.
Note: If a MARK is made with C(MARKSTAT) = 0, the OPTICS X, OPTICS Y, and TIME 1 counters are displayed and no further action is taken. When this takes place the VERB is 06, the Noun is 56.

15. OPTSTALL

This program is completely analogous *to* IMUSTALL with the substitution of the register name OPTCADR for IMUCADR and WASOPSET for WASKSET. The routines are completely independent and may be used concurrently.

16. CDUINC

If the execution of the main program requires the addition of a 1's complement number to a 2's complement counter, the following is commanded:

```
L       TC BANKCALL
L + 1   CADR CDUINC
```

with the 1's complement number in TEM2 and the address of the 2's complement quantity in ADDRWD. Return is to L + 2 with the 2's complement sum in the location whose address is in.ADDRWD. The sum is also available in A.

17. IMUPULSE

If the IMU is in the Fine Align mode, the IRIG's may be pulse-torqued by commanding

```
L       TC BANKCALL
L + 1   CADR IMUPULSE
```

with the double precision gyro demands in GYROD to GYROD + 5 for the x, y, and z gyros, respectively. Units of the demands are the same as the gyro, i.e. $2\pi/2^{20}$ radians/bit. The DP demand need not have sign agreement.

The gyros are pulsed in the order y, z, x (inner, middle, outer) with the last pulse delivered with negative sign. If the demand D is positive, D + 2 pulses are sent followed within 20 ms by 2 negative pulses. If D is negative 2 positive pulses are sent out followed immediately by D-2 negative pulses. Note that a zero demand produces + 2 followed by -2 pulses to assure uniform drift rates on all gyros.

A positive pulse is defined as a pulse which results in a positive rotation about the gyro output axis.

The augmentation of the pulse train by two in each direction is designed to reduce the effect of pulse rise time on the scale factor. The effect of the first two pulses is greatly reduced by the relatively slow rise

time of the torque circuits. The initial two pulses are canceled by the last two pulses and the intervening pulses have a fairly constant scale factor.

The gyros are always left in the same state (negative) to yield good drift repeatability.

Calls to IMUPULSE terminate with a call to IMUSTALL for idling and error checking as usual.

EXTENDED VERBS FOR MODING

A number of noun-verb combinations are provided in this program section for manual moding of the IMU and OPTICS. For the most part, they simply call routines described in "Mode Switching and Mark" and "Pinball". It should be understood that these verbs are provided primarily for laboratory use and that mission programs do their own moding.

1. Verb 40 Noun 20 - Zero IMU CDU's

   Call to IMUZERO followed by a call to IMUSTALL.

2. Verb 40 Noun 55 - Zero Optics CDU's

   Not available in SUNRISE since the required interface connections are not yet available.

3. Verb 41 Noun 20 - Coarse Align IMU

   Call for a load of desired gimbal angles (V25 N22) followed by calls to IMUCOARS and IMUSTALL when the angles have been entered,

4. Verb 41 Noun 55 - Coarse Align Optics

   Call for a load of desired optics shaft and trunion angles (V24 N57) followed by an enabling of the T4TUPT Optics drive by placing +0 in OPTIND.

5. Verb 42 - Fine Align the IMU

   Call for a load of gyro torquing angles (V25 N67) followed by calls to IMUFINE, IMUSTALL, IMUPULSE, and IMUSTALL in that order.

6. Verb 43 - Lock IMU

   Call IMULOCK followed by IMUSTALL.

7. Verb 44 - Attitude Control

   Call for desired CDU angles (V25 N22) followed by calls to IMUATTC and IMUSTALL.

8. Verb 45 - Re-entry Control

   Call for a load of desired CDU angles (V25 N22) followed by calls to IMUREENT and IMUSTALL.

9. Verb 46 - Return to Coarse Align

Call IMURECOR followed by IMUSTALL.

10. Verb 47 - Optics Tracker On

Not available in SUNRISE.

11. Verb 50 - Please Perform

Used to display a command to the operator. Depression of ENTER signifies requested action performed.

12. Verb 51 - Please MARK

Request for operator to MARK.

13. Verb 52 - MARK ACCEPT

Performs the same function as a depression of the MARK ACCEPT button. The verb will be provided until the button is placed on the G&N panel.

14. Verb 53 - Release IMU

Makes IMU available to other programs.

15. Verb 54 - Torque Gyros

Calls for a load of torquing angles (V25 N67) followed by calls to IMUPULSE and IMUSTALL. This call assumes the IMU is in Fine Align (as opposed to V42 above) and does not call IMUFINE.

AGC SELF-CHECK

The routine SELF-CHECK in SUNRISE was written to exercise most of the control pulses in the AGC **4** computer. Most of the control pulses in an instruction are used every time that particular instruction is used; however, the functions that some of these pulses perform are not utilized until some time later. A systematic method is used to check the existence of pulses that perform such functions, **As** an example, the pulses that write a data or instruction word back into erasable after it has been used are not checked until that data or instruction word is used again. It can be assumed that all control pulses are checked unless otherwise stated.

Some control pulses serve no useful purpose but they appear in various memory cycles because they are utilized on the same line in other memory cycles and it was not economical in terms of computer design to omit them. These pulses are WSC pulse on line 9 of TC, **NDXl,** MP3, RUPT3, and STD2 memory cycles and RSC pulse on line 11 of NDXO, CCSl, and line 7 of NDXl, MP3 memory cycles,

There are two memory cycles and a few control pulses located in other memory cycles that cannot be checked without external signals from outside the computer, Specifically, the control pulses in the SHINC and SHANC memory cycles, the WOVR pulse in the MINC memory cycle, and the WOVI pulse on line 11 of the NDXl memory cycle can not be checked by means of program only.

A short description of each subroutine of SELF-CHECK is given below together with the general type of control pulses checked by each subroutine. No particular effort has been made to check the pulses used in connection with the S and *Z* registers and the NISQ pulse, Some of these pulses are used in every memory cycle and the fact that SELF-CHECK is itself successfully performed insures the existence of these pulses,

1. CCSCHK

The CCS instruction is checked first because this instruction is used to test the remaining computer instructions. This subroutine checks that CCS performs the four required branches correctly and

that C(A) is correct after each branch. Three TC instructions were added at the end of this subroutine to specifically check the RZ, WQ , RSC pulses.

Most of the CCS control pulses and half of the TC control pulses are checked by this subroutine.

2. PTY+ERAS

The first half of this subroutine checks the ability of instructions to replace correctly data words back into erasable memory, as well as checking and generating correct parity for these data words,
The second part checks the ability of instructions to replace correctly instruction words back into erasable memory, as well as checking and generating correct parity for these instruction words.

This subroutine checks the operation of two groups of pulses at the same time. The WG - RG, WB - RB, WG pulses which control reading out of erasable memory and writing back into erasable memory. The WP - GP, TP - RP2 pulses which generate the correct parity and test for correct parity.

All of the parity pulses used in connection with the parity register, except those in the TS instruction and those used in the PINC and MINC memory cycles, are checked by this subroutine. The parity control pulses not so checked are checked by the TS+-CHK subroutine,

3. SCCHK

The SCCHK routine specifically checks the existence of the RSC and WSC pulses. The LP register (address *0003)* is utilized for this purpose. Normally the correct operation of the instruction is not of importance here. For example, when a CS LP is performed, it is not of interest if the clear and subtract operation is performed correctly. What is important is that the word (number) in LP has been shifted once. The mask instruction is exceptional because it lacks an WSC pulse; thus c(LP) does not get shifted. Another exception occurs at the end of the routine when *CCS* is used to assertain if the data word in LP has been shifted the correct number of times.

4. MPCHK

The MPCHK subroutine checks most of the pulses associated with the multiply instruction. The number ±37777 is multiplied by ±00002 and the contents of the A register and the contents of the LP register checked in each of the four multiplications. The c(A) = ±00001, the high order product and c(LP) = ±37776, and the low order product.

5. SUCHK

The SUCHK subroutine checks most of the pulses associated with the subtract instruction. The number +37777 is subtracted from +37776 and the c(A) checked for -1.

6. DVCHK

This subroutine checks most of the pulses associated with the divide instruction, The number ± 1/4 is divided by ±3/8 and the contents of the A register, Q register, and LP register are checked for each of the four divisions. The c(A) = ±25252, the quotient and c(Q) = 67777 = -|remainder| The c(LP) = + 1 for + sign of quotient, 40000 for + ÷ - division, and 40001 for - ÷ + division.

7. TS+-CHK

The ability of the computer to perform a PINC and MINC operation on overcounter is checked by this subroutine. The skip on overflow feature of the TS instruction is also checked.

This subroutine checks most of the pulses used by the transfer to storage instruction and all of the pulses in the PINC and MINC memory cycles except the WOVR. pulse,

Associated with each subroutine, are one or two bits. At the end of the successful completion of the subroutine, or first half if two bits are assigned, the assigned bit is put in an erasable register called OKREG. Thus, if SELF-CHECK is not successfully completed, the contents of OKREG determines the number of parts of the routine completed. This part of SELF-CHECK might not be required in later versions,

The following action is taken by the computer if an error is found while performing SELF-CHECK:

1. c(Q) is put in SFAIL register.
2. PROGRAM ALARM LIGHT is turned on,
3. Display octal 1102 (V01 N31) in R1 of DSKY and place in FAILREG register.
4. c(SMODE) is set to zero.

As mentioned previously, SELF-CHECK is part of the backup idle loop and thus has the lowest possible priority. SELF-CHECK can be initiated using **DSKY** or the monitor of the computer test set. Verb **21** and noun **27** have to be utilized to enter from the DSKY; this action calls for a load of the SMODE register. The number loaded into the SMODE register controls the manner in which SELF-CHECK will be used in the backup idle loop. **A** negative number entry will cause the SELF-CHECK routine to be performed continuously until a job of higher priority 'comes along; the contents **of** counters SCOUNT and SCOUNT+1 will indicate the number of successful iterations performed. **A** positive number entry **will** indicate the number ***of*** times this routine will be performed before being dropped from the backup idle loop. SELF-CHECK is eliminated from the backup idle loop when the contents of SMODE **is** +zero.

ALARM PROCESSOR

This program section facilitates the display of certain alarm messages on the DSKY. Two routines are available: ALARM and ABORT,

1. ALARM

When a failure is detected which does not necessarily require a Fresh Start a call to ALARM may be made to display an error code. The calling sequence is:

```
L        TC     ALARM
L + 1    OCT + AAANN
```

where AAA is the 8 bit number of the general area (e.g. 001 for OPTICS interface, 002 for IMU interface, etc.) and NN the 6 bit alarm number within that area. (A tabulation of current alarms is given in the accompanying table.) Return is to L + 2.

a. If C(FAILREG) = 0, the error code at L + 1 is stored in FAILREG and a new job is initiated whose sole purpose is to display FAILREG. The priority of this job is 37, so it will be initiated at the next CCS NEWJOB. The PROGRAM ALARM lamp is turned on.

b. If C(FAILREG) ≠ 0, a failure has already occured. Bit 15 is set to a one and the error code at L + 1 is ignored. Bit 15 is, thus, the "mulitple failures" bit.

c. If C(FAILREG) < - 0, multiple failures have occured and no action is taken.

Note: FAILREG is set to zero and the PROGRAM ALARM is turned off on ERROR RESET or "Fresh Start".

2. ABORT

When a failure which can only be remedied by a Fresh Start occurs (e.g. Executive or WAITLIST table overflow) a call to ABORT is made to initiate the Fresh Start. The sequence is:

```
L        TC     ALARM
L + 1    OCT + AAANN
```

The PROGRAM ALARM light and FAILREG are set as in ALARM, and the routine terminates with a programmed TCA trap:

```
WHIMPER     TC      WHIMPER (Not with a bang. . . .)
```

A GOJAM sequence is thus triggered which initiates a program restart at location **2030.** See description of program section Fresh Start and Restart for details on restarting.

Table of Error Codes for ALARM::' and ABORT

Optics Sub-system

00101	Zero **CDU** switch altered before. expiration of **30** second wait
00102	Computer unable to achieve desired OPTICS mode
00103	SXT-ON switch turned on with OPTICS not in **ZERO CDU** mode
*00***104**	No VAC areas available for MARKS
00105.	Internal MARK request with MARK system busy
00106	SXT-ON switch not in desired state at MARK time or MARK with all requested MARKS accepted
00107	MARK ACCEPT with MARK system not.in use
00110	MARK ACCEPT with all requested MARKS accepted, or no MARKS since intialization or last MARK ACCEPT

IMU Sub-system

00201	**CDU** zero switch altered before expiration of **30** second wait

*All are ALARMS unless otherwise stated

00202	Computer unable to achieve desired mode
00203	No IMU Mode indicated to computer
00204	Mode switch with TRNSW in computer control, but computer not commanding
Major Mode Conflict	
00301	Two programs trying to use the IMU
Procedural Difficulty	
00401	Desired gimbal angles yield gimbal lock (i.e. MGA $\geq 60^{\circ}$)
00402	Star out of field of view
Computer Hardware Malfunctions	
01101.	RUPT2 Occured
01102	AGC Self Test error (Q in SFAIL)
List Overflows (All Abort)	
01201	Executive overflow - No VAC areas
01202	Executive overflow - no core sets
01203	WAITLIST overflow - too many tasks
01204	Slight variation of 01203
01205	Master Control overflow - too many jobs waiting
01206	Keyboard & Display waiting line overflow
01207	No **VAC** area for MARKS

INTERBANK COMMUNICATION

This utility program section allows transfer of information and/or control between switched Banks in the AGC. The section consists of six routines. It should be noted that the "call" routines may be used only under Executive control. However, the "jump" routines may be used in the interrupt mode as well.

1. BANKCALL

Allows transfer of control from any switchable band to any other (including fixed-fixed) with accumulator protection going and coming. Usage is as follows:

```
L       TC BANKCALL
L + 1   CADR SUBRO
```

where **SUBRO** is the location of the destination. The program **SUBRO** terminates with **TC SWRETURN** to return control to **L + 2.** Alternatively, the subroutine may execute **TC Q** since C(Q) is set to **SWRETURN.**

2. SWCALL

Similar to **BANKCALL** except that the **CADR** of the destination arrives **in** the accumulator. The accumulator is protected on returning via **SWRETURN.** Usage is as follows:

```
L       CAF CADR SUBRO
L + 1   TC SWCALL
```

3. POSTJUMP

Permits **a** uni-directional jump to **another bank** with accumulator protected. Usage is **as** follows:

```
L       TC POSTJUMP
L + 1   CADR DESTIN
```

where **DESTIN** is the destination location.

4. BANKJUMP

Similar to POSTJUMP except that the CADR of the destination arrives in the accumulator. Usage is as follows:

```
L       CAF CADR SUBRO
L + 1   TC BANKJUMP
```

5. DATACALL

Fetches the contents of a location in a switchable bank. Usage is as follows:

```
L       CAFDATACADR
L + 1   TC DATACALL
```

Return is to L + 2 with the contents of the location whose CADR is in DATACADR in the accumulator.

6. MAKECADR

Constructs, from information set by BANKCALL or SWCALL, the CADR of the return address and places it in location ADDRWD. Usage is as follows:

```
L       TC MAKECADR
```

Return is to L + 1 with the CADR in ADDRWD.

This routine is particularly useful for subroutines which require parameters given in the calling sequence. The CADR of the parameters is formed by MAKECADR and the parameters themselves are then fetched by the DATACALL routine.

This program section is used during a coasting period of an Apollo mission to compute position and velocity of the spacecraft.

Position and velocity will be maintained in the computer in non-rotating rectangular coordinates and will be referenced to either the earth or the moon, An earth centered equatorial coordinate system is used when the vehicle is outside of the lunar sphere of influence. Inside of this sphere the center of coordinates will coincide with the center of the moon. The extrapolation of position and velocity will be made by a direct numerical integration of the equations of motion.

The basic equation may be written in the form

$$\frac{d^2}{dt^2}\,\underline{r}_{PV} + \frac{\mu_P}{r^3_{PV}}\,\underline{r}_{PV} = \underline{a}_d \qquad (1)$$

where $\underline{r}_{PV}$ is the vector position of the vehicle with respect to the primary body P which is either the earth or moon and μ_P is the gravitational constant of P. The vector $\underline{a}_d$ is the vector acceleration which prevents the motion of the vehicle from being precisely a conic with P at the focus.

Encke's Method

If a_d is small compared with the,central force field, direct use of **Eq. (1)** is inefficient. Therefore, the integration will be accomplished using the technique of differential accelerations attributed to Encke.

At time t_0 the position and velocity vectors $\underline{r}_{PV}(t_0)$ and $\underline{v}_{PV}(t_0)$ define an osculating orbit. The vector difference $\underline{\delta}(t)$ satisfies the following differential equation

$$\frac{d^2}{dt^2}\,\underline{\delta} = \frac{\mu_P}{r^3_{PV(C)}}\left[\left(1 - \frac{r^3_{PV(C)}}{r^3_{PV}}\right)\underline{r}_{PV} - \underline{\delta}\right] + \underline{a}_d \qquad (2)$$

subject to the initial conditions

$$\underline{\delta}(t_0) = 0 \qquad \frac{d}{dt}\underline{\delta}(t_0) = \underline{\nu}(t_0) = 0$$

where $\underline{r}_{PV(C)}$ is the osculating conic position vector. The numerical difficulties which would arise from the evaluation of the coefficient of $\underline{r}_{PV}$ in **Eq. (2)** may be avoided as follows. Since

$$\underline{r}_{PV}(t) = \underline{r}_{PV(C)}(t) + \underline{\delta}(t) \tag{3}$$

it follows that

$$-\frac{r_{PV(C)}}{r^3_{PV}}\underline{r} = -f(q_C) = 1 - (1+q_C)^{3/2}$$

where

$$q_C = \frac{(\underline{\delta} - 2\underline{r}_{PV}) \cdot \underline{\delta}}{r^2_{PV}} \tag{4}$$

The function f(q) may be conveniently evaluated from

$$f(q) = q\,\frac{3 + 3q + q^2}{1 + (1+q)^{3/2}} \tag{5}$$

Encke's method may now be summarized as follows:

1. Position in the osculating orbit is calculated from

$$\underline{r}_{PV(C)}(t) = \left[1 - \frac{x^2}{r_{PV}(t_0)}\, C(\alpha_0 x^2)\right] \underline{r}_{PV}(t_0)$$

$$+ \left[(t - t_0) - \frac{x^3}{\sqrt{\mu_P}}\, S(\alpha_0 x^2)\right] \underline{v}_{PV}(t_0) \qquad (6)$$

where

$$\alpha_0 = \frac{2}{r_{PV}(t_0)} - \frac{v_{PV}(t_0)^2}{\mu_P} \qquad (7)$$

and x determined as the root of Kepler's equation in the form

$$\sqrt{\mu_P}(t - t_0) = \frac{\underline{r}_{PV}(t_0) \cdot \underline{v}_{PV}(t_0)}{\sqrt{\mu_P}}\, x^2\, C(\alpha_0 x^2)$$

$$+ \left[1 - r_{PV}(t_0)\alpha_0\right] x^3\, S(\alpha_0 x^2) + r_{PV}(t_0)\, x \qquad (8)$$

The special transcendental functions S and C are defined by

$$S(x) = \frac{1}{3!} - \frac{x}{5!} + \frac{x^2}{7!} - \ldots$$

$$C(x) = \frac{1}{2!} - \frac{x}{4!} + \frac{x^2}{6!} - \ldots$$

2. Deviations from the osculating orbit are then obtained by a numberical integration of

$$\frac{d^2}{dt^2}\, \underline{\delta}(t) = - \frac{\mu_P}{r^3_{PV(C)}(t)} \left[f(q)\, \underline{r}_{PV}(t) + \underline{\delta}(t)\right] + \underline{a}_d(t) \qquad (9)$$

The first term on the right hand side of the last equation must remain small, i.e. of the same order as $\underline{a}_d(t)$, if the method is to be efficient. As the deviation vector $\underline{\delta}$ grows in magnitude, this term will eventually increase in size. Therefore, in order to maintain the efficiency, a new osculating orbit should be defined by the true position and velocity. The process of selecting a new conic orbit from which to calculate deviations is called rectification. When rectification occurs, the initial conditions for the differential equation for $\underline{\delta}$ are again zero and the right hand side is simply the perturbation acceleration $\underline{a}_d$ at the time of rectification.

3. The position vector $\underline{r}_{PV}(t)$ is computed from Eq. *(3)* using Eq. *(6)*. The velocity vector $\underline{v}_{PV}(t)$ is then computed from

$$\underline{v}_{PV}(t) = \underline{v}_{PV(C)}(t) + \underline{\nu}(t) \tag{10}$$

where

$$\underline{v}_{PV(C)}(t) = \frac{\sqrt{\mu_P}}{r_{PV}(t_0)\, r_{PV(C)}(t)} \left[\alpha_0 x^3 S(\alpha_0 x^2) - x\right] \underline{r}_{PV}(t_0)$$

$$+ \left[1 - \frac{x^2}{r_{PV(C)}(t)} C(\alpha_0 x^2)\right] \underline{v}_{PV}(t_0) \tag{11}$$

Disturbing Acceleration

The form of the disturhing acceleration $\underline{a}_d$ to be used depends on the phase of the mission. In earth orbit only the gravitational anomalies arising from the non-spherical shape of the earth need be considered. During trans-lunar and trans-earth flight, the gravitational attraction of the sun and the secondary body Q (either earth or moon) are relevant forces. In lunar orbit, it may be necessary to consider forces arising from the non-spherical shape of the moon. A summary of the various cases appears below.

a. Earth orbit

$$\underline{a}_d = -\frac{\mu_E}{r_{EV}^2} \sum_{k=2}^{4} J_k \left(\frac{r_{eq}}{r_{EV}}\right)^k \left[P'_{k+1}(\cos\phi)\,\underline{i}_{EV} - P'_k(\cos\phi)\,\underline{i}_z\right] \qquad (12)$$

where

$$P'_2(\cos\phi) = 3\cos\phi$$

$$P'_3(\cos\phi) = \frac{1}{2}(15\cos^2\phi - 3)$$

$$P'_4(\cos\phi) = \frac{1}{3}(7\cos\phi\, P'_3 - 4P'_2)$$

$$P'_5(\cos\phi) = \frac{1}{4}(9\cos\phi\, P'_4 - 5P'_3)$$

are the derivatives of second and third order Legendre polynomials;

$$\cos\phi = \underline{i}_{EV} \cdot \underline{i}_z$$

is the cosine of the angle ϕ between the unit vector $\underline{i}_{EV}$ in the direction of $\underline{r}_{EV}$ and the unit vector $\underline{i}_z$ in the direction of the north pole; r_{eq} is the equatorial radius of the earth; and J_2, J_3, J_4 are the coefficients of the second, third and fourth harmonics of the earth's potential function. The subscript E denotes that the center of the earth is the center of coordinates.

b. Trans-lunar and trans-earth flight

$$\underline{a}_d = -\frac{\mu_Q}{r_{QV}^3}\left[f(q_Q)\,\underline{r}_{PQ} + \underline{r}_{PV}\right] - \frac{\mu_S}{r_{SV}^3}\left[f(q_S)\,\underline{r}_{PS} + \underline{r}_{PV}\right] \tag{13}$$

where the subscripts Q and S denote the secondary body and the sun respectively, Thus, for example, $\underline{r}_{PS}$ is the position vector of the sun with respect to the primary body. The arguments q_O are calculated from

$$q_{(\,)} = \frac{(\underline{r}_{PV} - 2\underline{r}_{P(\,)}) \cdot \underline{r}_{PV}}{r^2_{P(\,)}} \tag{14}$$

and the function f from **Eq.** (5).

Ephemeris data for the positions of the moon relative to the earth $\underline{r}_{EM}$ and the sun relative to the earth moon barycenter $\underline{r}_{BS}$ are required as functions **of** time. The position of the sun relative to the primary planet $\underline{r}_{PS}$ is then computed from

$$\underline{r}_{PS}(t) = \underline{r}_{BS}(t) + \frac{\mu_Q}{\mu_P + \mu_Q}\,\underline{r}_{PQ} \tag{15}$$

In the vicinity of the lunar sphere of influence a change in Origin of coordinates is made. Thus

$$\begin{aligned} \underline{r}_{PV}(t) - \underline{r}_{PQ}(t) &= \underline{r}_{QV}(t) \rightarrow \underline{r}_{PV}(t) \\ \underline{v}_{PV}(t) - \underline{v}_{PQ}(t) &= \underline{v}_{QV}(t) \rightarrow \underline{v}_{PV}(t) \end{aligned} \tag{16}$$

c. Lunar orbit

$$\underline{a}_d = -\frac{3\mu_M r_m^2 C'}{2r_{MV}^4}\left\{\left\{\frac{B-A}{C}\left[1-5(\underline{i}_{MV}\cdot\underline{i}_\eta)^2\right]+\frac{C-A}{C}\left[1-5(\underline{i}_{MV}\cdot\underline{i}_\zeta)^2\right]\right\}(\underline{i}_{MV}\cdot\underline{i}_\xi)\underline{i}_\xi\right.$$

$$+\left\{\frac{B-A}{C}\left[3-5(\underline{i}_{MV}\cdot\underline{i}_\eta)^2\right]+\frac{C-A}{C}\left[1-5(\underline{i}_{MV}\cdot\underline{i}_\zeta)^2\right]\right\}(\underline{i}_{MV}\cdot\underline{i}_\eta)\underline{i}_\eta \qquad (17)$$

$$\left.+\left\{\frac{B-A}{C}\left[1-5(\underline{i}_{MV}\cdot\underline{i}_\eta)^2\right]+\frac{C-A}{C}\left[3-5(\underline{i}_{MV}\cdot\underline{i}_\zeta)^2\right]\right\}(\underline{i}_{MV}\cdot\underline{i}_\zeta)\underline{i}_\zeta\right)$$

where A, B, C are the principal moments of inertia of the moon, r_m is the radius of the moon, C' is C divided by the product of the mass of the moon and the square of its radius, $\underline{i}_\xi$, $\underline{i}_\eta$, $\underline{i}_\zeta$ are the selenographic coordinate unit vectors, and $\underline{i}_{MV}$ is the unit vector in the direction of $\underline{r}_{MV}$.

Error Transition Matrix

Position and velocity vectors of the vehicle as maintained in the computer are only estimates of the true values. As a part of the navigation technique, it is necessary also to maintain statistical data in the computer to aid in the processing of navigation measurements.

If $\underline{\epsilon}(t)$ and $\underline{v}(t)$ are the errors in the estimates of the position and velocity vector, respectively, then the six dimensional correlationmatrix $E(t)$ is defined by

$$E(t) = \begin{pmatrix} \overline{\underline{\epsilon}(t)\,\underline{\epsilon}(t)^T} & \overline{\underline{\epsilon}(t)\,\underline{v}(t)^T} \\ \overline{\underline{v}(t)\,\underline{\epsilon}(t)^T} & \overline{\underline{v}(t)\,\underline{v}(t)^T} \end{pmatrix} \tag{18}$$

However, it is more convenient to utilize a matrix $W(t)$, called the error transition matrix, and defined by

$$E(t) = W(t)\,W(t)^T \tag{19}$$

Extrapolation of the matrix $W(t)$ is made by direct numerical integration of the differential equation

$$\frac{dW}{dt} = \begin{pmatrix} O & I \\ G(t) & O \end{pmatrix} W \tag{20}$$

where $G(t)$ is the three dimensional gravity gradient matrix. The matrices I and O are the three dimensional identity and zero matrices respectively. If the W matrix is partitioned as

$$W = \begin{pmatrix} \underline{w}_1 & \underline{w}_2 & \cdots & \underline{w}_6 \\ \frac{d\underline{w}_1}{dt} & \frac{d\underline{w}_2}{dt} & \cdots & \frac{d\underline{w}_6}{dt} \end{pmatrix} \tag{21}$$

then the extrapolation may be accomplished by successively integrating the vector differential equations

$$\frac{d^2}{dt^2}\,\underline{w}_i = G(t)\,\underline{w}_i \qquad i = 1, 2, \ldots, 6$$

The G(t) matrix for trans-lunar and trans-earth flight is given by

$$G(t) = \frac{\mu_E}{r^5_{EV}(t)}\left[3\,\underline{r}_{EV}(t)\,\underline{r}_{EV}(t)^T - r^2_{EV}(t)\,I\right] + \frac{\mu_M}{r^5_{MV}(t)}\left[3\,\underline{r}_{MV}(t)\,\underline{r}_{MV}(t)^T - r^2_{MV}(t)\,I\right]$$

Thus, the differential equations for the $\underline{w}_i(t)$ vectors are simply

$$\frac{d^2}{dt^2}\,\underline{w}_i = \frac{\mu_E}{r^3_{EV}(t)}\left\{3\left[\underline{i}_{EV}(t)\cdot\underline{w}_i(t)\right]\underline{i}_{EV}(t) - \underline{w}_i(t)\right\} + \frac{\mu_M}{r^3_{MV}(t)}\left\{3\left[\underline{i}_{MV}(t)\cdot\underline{w}_i(t)\right]\underline{i}_{MV}(t) - \underline{w}_i(t)\right\} \tag{22}$$

$$i = 1, 2, \ldots, 6$$

Numerical Integration Method

The extrapolation of navigation data requires the solution of seven second order vector differential equations, specifically Eqs. (9) and (22). These are all special cases of the form

$$\frac{d^2}{dt^2} \underline{y} = \underline{f}(\underline{y}) \tag{23}$$

in which the right hand side is a function of the independent variable and time only. Nystrom's method is particularly will suited to this form and gives an integration method of fourth order accuracy. The second order system is written

$$\begin{aligned} \frac{d}{dt} \underline{y} &= \underline{z} \\ \frac{d}{dt} \underline{z} &= \underline{f}(\underline{y}) \end{aligned} \tag{24}$$

and the formulae are summarized below

$$\begin{aligned} \underline{y}_{n+1} &= \underline{y}_n + \underline{\phi}(\underline{y}_n)\,\Delta t \\ \underline{z}_{n+1} &= \underline{z}_n + \underline{\psi}(\underline{y}_n)\,\Delta t \\ \underline{\phi}(\underline{y}_n) &= \underline{z}_n + \frac{1}{6}(\underline{k}_1 + 2\underline{k}_2)\,\Delta t \\ \underline{\psi}(\underline{y}_n) &= \frac{1}{6}(\underline{k}_1 + 4\underline{k}_2 + \underline{k}_3) \\ \underline{k}_1 &= \underline{f}(\underline{y}_n) \\ \underline{k}_2 &= \underline{f}(\underline{y}_n + \frac{1}{2}\underline{z}_n\,\Delta t + \frac{1}{8}\underline{k}_1(\Delta t)^2) \\ \underline{k}_3 &= \underline{f}(\underline{y}_n + \underline{z}_n\,\Delta t + \frac{1}{2}\underline{k}_2(\Delta t)^2) \end{aligned} \tag{25}$$

For efficient use of computer storage as well as computing time the computations are performed in the following order.

a. Equation (9) is solved using the Nystrom formulae (25). The position of the sun and moon are required at times t_n, $t_n + \frac{1}{2}$ At, t_n + At to be used in the evaluation of the vectors $\underline{k}_1$, $\underline{k}_2$, $\underline{k}_3$ respectively. It is necessary to preserve the values of the vectors $\underline{r}_{EV}$ and $\underline{r}_{EM}$ at these times for use in the solution of **Eqs. (22).**

b. Equations **(22)** are solved one set at a time using formulae **(25)** together with the values of $\underline{r}_{EV}$ and $\underline{r}_{EM}$ which resulted from the first step.

PRELAUNCFI ALIGNMENT

The prelaunch alignment program is designed to erect the IMU *so* that the stable member *x* axis points along the local vertical and to align the IMU *so* that the stable member z axis points along a specified azimuth in the horizontal plane. This is accomplished by instrumenting a second order control loop through the AGC. During the erection phase both the y and z PIPA input channels are handled identically. During the alignment, or gyrocompassing, phase the z channel is maintained as before (with a gain change) and the y channel is used to perform the azimuth alignment and to hold the vertical orientation of the x axis at the same time. The two phases are detailed in the accompanying block diagrams.

The program may be logically divided into the following four sections:

1. Initialization

The program is called by the "change major mode" verb. The operator has the option of accepting a set of standard initial conditions or, alternatively, of specifying desired initial conditions prior to the call. The quantities to be specified are:

Azimuth - measured clockwise from north and given as a fraction scaled by $2n$.

Initial Gimbal Angles - in CDU units; i. e. $2\pi/2^{15}$ radians/bit.

Gyro Drift Compensation - in IRIG pulse torque units/.5sec.

Latitude - scaled by $2n$.

The standard initial conditions are zero gimbal angles and drifts, 90° E azimuth, and latitude of MIT/IL-7 (42° 21' 51.189").

2. IMU Moding

The following sequence takes place using the routines described under section "Mode Switching and Mark Routines" :

a. **CDU** zeroing for **30** seconds

b. Coarse align to specified gimbal angles **for** *60* seconds

c. Fine align for 90 seconds with no pulse torquing **of** the gyros.

3. Vertical Erection

For the next 5 min. (assuming no operator intervention) the y and z PIPA inputs are filtered and fed back to the z and y gyros, respectively, in such a way as to level the y-z plane. The PIPA's are read and the gyros pulse-torqued at a frequency of 2 cps . The loop approximates a second order system with a time constant of 50 seconds and a damping factor of .5.

The equations are :

$$(INT)_y = (INT)Y + (\Delta v)_y$$

$$\theta_z = K_1 (\Delta v)_y + K_2 (INT)_y$$

$$(INT)_z = (INT)_z + (\Delta v)_z$$

$$\theta_y = K_1 (\Delta v)_z + K_2 (INT)_z$$

$$(INT)_E = (INT)_E + (\Delta v)_E$$

$$\theta_{STH} = K_1 (\Delta v)_E + K_2 (INT)_E$$

where Δv and θ are PIPA outputs and gyro pulse-torquing commands, respectively. Here the subscripts E and STH refer to fictitious easterly and southerly axes. Then the commands θ_x, θ_Y, θ_z are computed by adding an earth-rate compensation as follows:

$$\theta_x = \theta_x + \sin X \, \Omega \Delta t$$

$$\theta_{STH} = \theta_{STH} - \cos X \, \Omega \Delta t$$

$$\theta_y = e_Y + \sin \mathbb{C} \, \theta_E + \cos \psi \, \theta_{STH}$$

$$\theta_z = \theta_z + \cos \psi \, \theta_E - \sin \psi \, \theta_{STH}$$

where

Δt = cycle time

λ = latitude

ψ = azimuth

Ω = earth's rotation rate

Note that, since the azimuth is not controlled during vertical erection, the θ_E command is set to zero.

4. Gyrocompassing

After 5 minutes have elapsed (again, assuming no operator intervention) control of the azimuth is initiated. Vertical erection is maintained about the southerly axis, with K 1 and K2 taking on new values. A gyrocompassing technique is used to detect spurious rate commands to the stable member about an easterly axis. These rates are used to close the azimuth loop and to maintain a single order vertical erection about the easterly axis.

The equations are

$$\theta_{STH} = K_1' (\Delta v)_E + K_2' (\Delta v)_E - \cos \lambda \, \Omega \Delta t$$

$$(\Delta v)_E = (\Delta v)_E + \sin \psi \, (\Delta v)_Y + \cos \psi \, (\Delta v)_z$$

$$INT_E = INT_E + (\Delta v)_E$$

$$(\Delta v)_{STH} = \cos \psi \, (\Delta v)_Y - \sin \psi \, (\Delta v)_z$$

$$(FILTER) = C_2 \, (FILTER) + C_1 \, (\Delta v)_{STH}$$

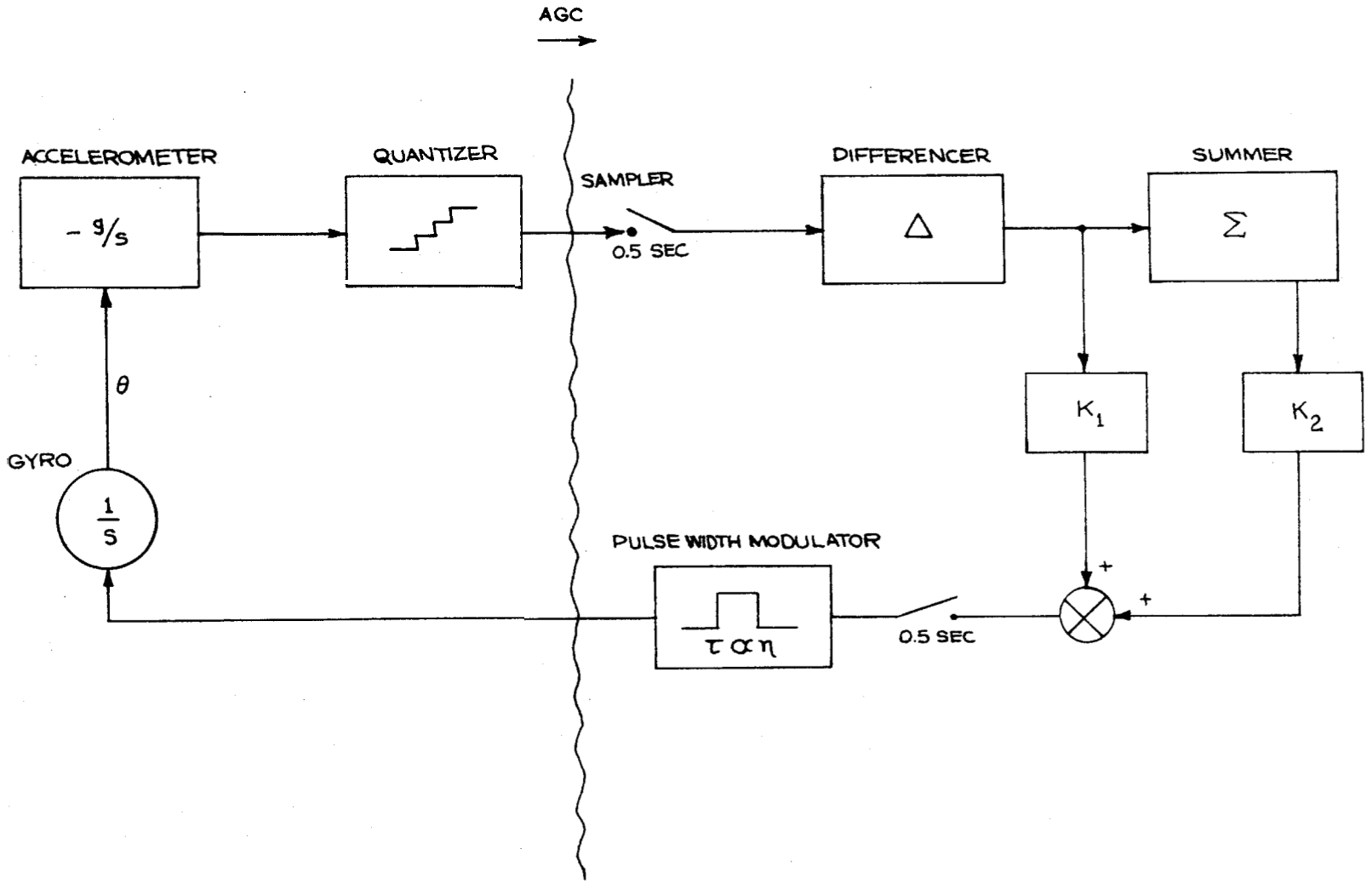
$$\theta_x = \theta_x + C_3 \, (FILTER) + \sin \lambda \, \Omega \Delta t$$

$$\theta_E = C_4 \, (FILTER)$$

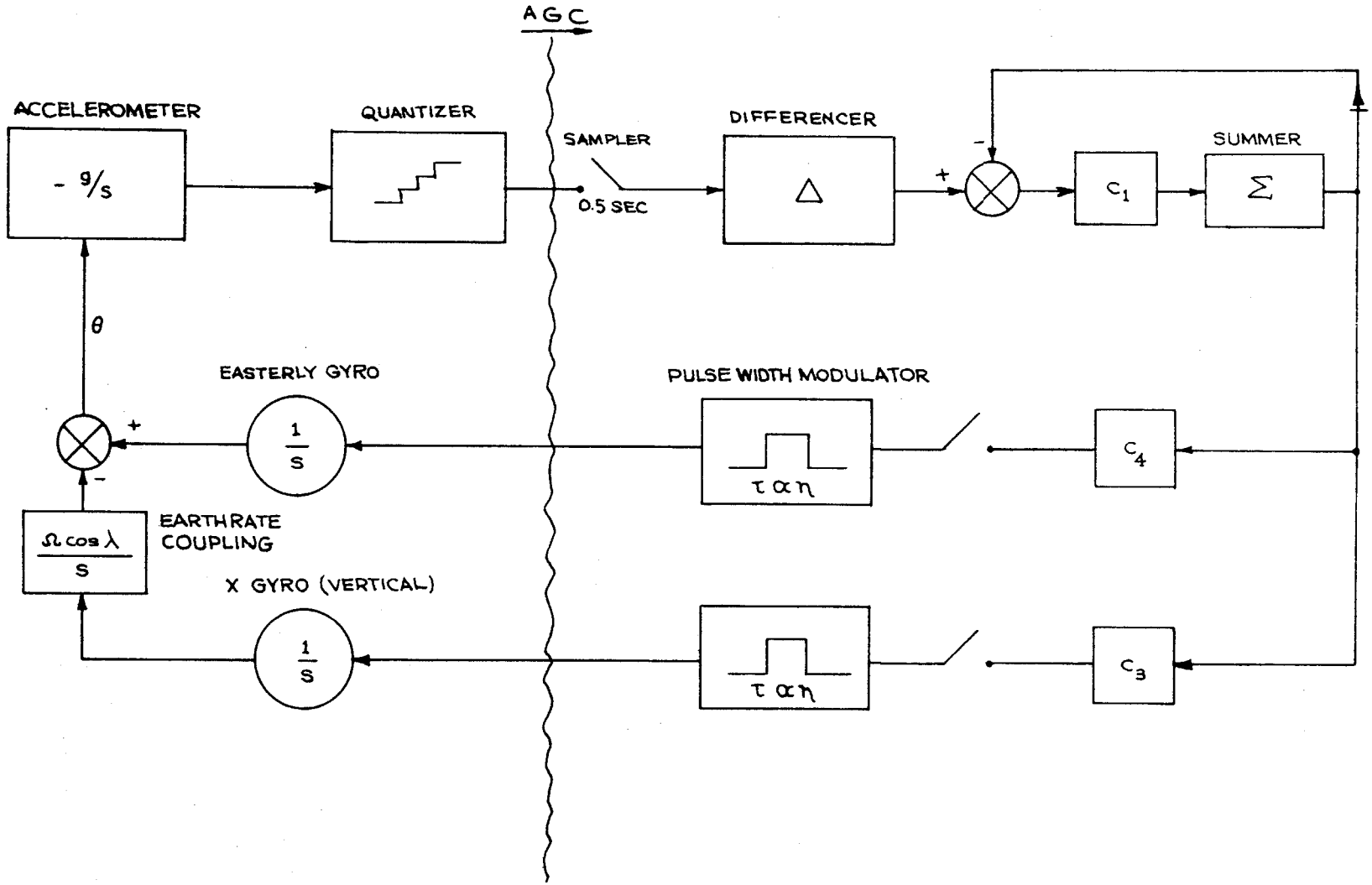
$$\theta_y = \sin \psi \, \theta_E + \cos \psi \, \theta_{STH}$$

$$\theta_z = \cos \psi \, \theta_E - \sin \psi \, \theta_{STH}$$

The loop approximates a critically damped second order system with a time constant of 1000 seconds.



VERTICAL ERECTION LOOP



GYROCOMPASSING LOOP

RTB ROUTINES

The RTB instruction (Return To Basic) allows the interpretive instruction set to be augmented by adding basic language subroutines which may be called by this single interpretive instruction. Those appearing in SUNRISE are as follows:

1. LOADTIME'"

RTB LOADTIME causes the contents of the DP clock TIME 2 and TIME 1 to be placed in MPAC, MPAC+1 with MPAC+2 set to zero.

2. FRESHPD

RTB FRESHPD causes the interpretive push-down pointer to be set to the top of the list; i. e. register *0.*

3. ZEROVAC

RTB ZEROVAC clears the first **38** registers of the assigned work area to +0. C(MPAC) are destroyed in the processes.

4. *C*DULOGIC*

RTB CDULOGIC is used to convert a single precision 2's complement angle counter (CDU's) to a DP 1's complement number scaled in revolutions. Usage is as follows:

```
SMOVE   1
RTB
            CDU (XYZ)
            CDULOGIC
```

where **CDU** *(XYZ)* is a register containing a CDU reading (not the CDU counter itself).

---

* Starred 'routines will cause the 'result of their computation to be entered in the push-down list if they appear at the end of an equation. The remainder will not push-down even if at the end *of* an equation with no store address. given.

5. 1STO2S*

RTB 1STO2S is the inverse of CDULOGIC: i.e., a DP 1's complement angle scaled in revolutions (arriving in MPAC, MPAC+1) is converted to a 2's complement single precision angle scaled in half revolutions. The answer is left in MPAC with MPAC+1 set to zero.

6. READPIPS*

RTB READPIPS causes the three PIPA counters to be read into the major part of the three components of the vector accumulator with the minor parts set to zero in the process.

7. PULSEIMU

RTB PULSEIMU does a call to IMUPULSE to initiate gyro torquing but does not do a call to IMUSTALL. This latter call must be provided by the user when appropriate.

8. SGNAGREE*

RTB SGNAGREE forces sign agreement for the various parts of the triple precision number arriving in MPAC, MPAC+1, and MPAC+2.

9. TRUNLOG*

RTB TRUNLOG is an epilogue to CDULOGIC for the OPT Y (trunion) CDU with its two scale factors. The following sequence produces the 1's complement trunion angle scaled in revolutions:

```
SMOVE   1
RTB     RTB
        C(OPTY)
        CDULOGIC
        TRUNLOG
```

where again C(OPTY) is a register containing a reading of OPTY and not the CDU counter itself.

SYSTEM TEST

SUNRISE incorporates system test programs LGNTEST AND PIPTEST. LGNTEST is basically a SEXTANT - IMU alignment test, which may also he used to measure certain IRIG performance parameters. PIPTEST is an accelerometer test which places each accelerometer input axis in turn both up and down *the* local gravity vector. The tests are described together with the manual operating procedures necessary to run the system correctly.

IMU stable member orientations are programmed into SUNRISE. The position of the **X** and ***Y*** stable member axes in these orientations are given by the following vectors in reference earth coordinates (Up, South, and East)

ORIENTATION #	XSM AXIS	YSM **AXIS**
1	(1, 0, 0)	(0, 1, 0)
2	(-1, 0, 0)	(0, 0, 1)
3	(0, 0, 1)	(1, 0, 0)
4	(0, 1, 0)	(-1, 0, 0)
5	(0, 1, 0)	(0, 0, 1)
6	(0, 0, 1)	(0, 1, 0)
7	(0, 0, 1)	$(\sqrt{2}/2, -\sqrt{2}/2, 0)$
8	$(\sqrt{2}/2, \sqrt{2}/2, 0)$	(0, 0, 1)
9	$(\sqrt{2}/2, -\sqrt{2}/2, 0)$	$(\sqrt{2}/2, \sqrt{2}/2, 0)$

These orientations will allow the determination of **all** PIPA parameters and all IRIG coefficients, when used with the appropriate procedures.

1. SEXTANT - Stable Member Alignment Test

In this test the IMU stable member is coarse and fine aligned with respect to the sight lines of two collimators fixed in the laboratory. The alignment is checked by observation of the local gravity vector with the accelerometers. Clearly no information about misalignment around the gravity vector is obtained in a single test; thus a minimum of two tests must be made with the stable member in different orientations. The effects of earth rotation are eliminated by making observations at three times and extracting the

term due to misalignment.

A description of program LGNTEST follows:

a. Desiredstahlc:member orientation is initialized by setting POSITON to +1.

b. The display is grabbed to prevent other internal calls.

c. LATITUDE is initialized and displayed together with POSITON. Encoders are zeroed.

d. Program waits in ENDIDLE for load or proceed.

e. Navigation base position is sound by use of the sextant in the subroutine FINDNB2.

f. Wait for completion of zero encode mode in IMUSTALL.

g. Do coarse and fine alignment of stable member by subroutine PUTPOS.

h. Initialize earth rotation vector, time and PIPA counters in subroutine LGNINIT.

i. Wait 30 seconds in WAITLOP. Every 10 seconds apply earth rate torquing to stable member.

j. Read the PIPA counters corresponding to the PIPA's in the horizontal plane, and time.

k. Wait 300 seconds in WAITLOP. Every 10 seconds apply earth rate torquing in azimuth only.

1. Read the PIPA counters, time.

m. Wait 300 seconds in WAITLOP. Every 10 seconds apply rate torquing in azimuth only.

n. Read the PIPA counters, time.

o. Display the time between the second two sets of readings, DT2, then the corresponding PIPA increments DELE2, DELSB.

p. Display the time between the second two sets of readings, DT2, then the corresponding PIPA increments DELEB, DELS2.

q. Program returns to step #c.

2. The Accelerometer Test

In the accelerometer test the IMU stable member is aligned with respect to known collimator lines of sight. After coarse and fine alignment the IMU is pulse torqued with both rate and vertical erection commands to align the required axis to the vertical and to maintain its position there. After a suitable

settling time, the arrival of the next velocity increment from the accelerometer under test is timed. After a further period of time, corresponding to the PIPA counter accumulation of some 15,000 increments, the time of arrival of the next velcoity increment is again recorded with the contents of the PIPA counter.

The total velcoity increment is multiplied by the nominal scale factor and divided by the time to give the observed value of the acceleration due to gravity, which may be compared with the known value to determine the accuracy of the accelerometer.

The test is repeated with each accelerometer along and opposite to the gravity vector.

A description of program PIPTEST follows:

a. The display is grabbed to prevent other internal calls.

b. Navigation base position is found by use of subroutine FINDNB which calls up sextant marks on the collimators.

c. The first orientation is chosen by setting POSITION to +1.

d. IMU mode is set to zero encode.

e. The operator checks latitude and position and may reset them if desired.

f. Various conditions are initialized in subroutine LHNINIT.

g. Wait in WAITLOP for 300 seconds; every 10 seconds vertical erection and earth rate torquing are performed.

h. The next PIPA pulse (or the one after that in case of interrupt) is timed by the routine CATCHAPP.

i. Wait in **WAITLOP** for 90 seconds; continue earth rate torquing and vertical erection every **10** seconds.

j. The next PIPA pulse is timed by CATCHAPP.

k. The results are computed and displayed.

1. Program returns to step e with position set to the next desired stable member orientation. When all six positions have been covered the program is terminated manually.

3. Operating Procedures for Automatic Tests in Program SUNRISE

a. Alignment Test Operation

1) Switch transfer switch to computer control.

2) Depress keys "Verb 70 Enter" (NOTE: mode changes to Zero Encode).

3) Observe "Verb 06 - Noun 66" flashing. **If** latitude and position, as displayed in R1-R2 and **R3** respectively, are correct then proceed to step **4.** To change latitude execute "Verb 21 Noun 02 Enter, 0763 Enter, major part of double precision latitude Enter, Verb 22 Noun **01** Enter, 0763 Enter, minor part of double precision latitude Enter". To change stable member position execute "Verb 21 Noun 02 Enter, 1205 Enter, position number (corresponding to preceeding table of orientations) Enter".

4) If display is now correct proceed by commanding "Verb 33 Enter".

5) Observe the mark command "Verb 51" flashing. Align SEXTANT line of sight with the #1 collimator (SOUTH); when aligned depress "Mark". If satisfied with mark command "Verb 52 Enter".

6) Align SEXTANT line-of-sight to **#4** collimator (SOUTHEAST). Depress "Mark". If satisfied with mark command "Verb 52 Enter". Flash should turn off (During the above procedures and after, mode should change through Zero Encode and Coarse Align to Fine Align).

7) Observe "Verb 06 Noun 66, R3 = 1000" flashing. Record DT1 in R1 and R2 (seconds and fractions of seconds). Key in "Verb 33 Enter" R3 changes to 1001 flashing. Record DELE1 in R1. Key in "Verb 33 Enter". R3 changes to 1002. Record DELS1 in R1. Key in "Verb 33 Enter". R3 changes to 1003. Record DT2 in R1 and R2 (seconds and fractions of seconds). Key in "Verb 33 Enter". R3 changes to 1004. Record DELE2 in R1. Key in "Verb 33 Enter". R3 changes to 1005. Record DELS2 in R1.

8) Switch transfer switch to Manual. Switch to Coarse Align (observe Program Alarm Light); return **CDU** counters to zero manually. Switch to Fine Align. Return transfer switch to computer control; key in "Verb 33 Enter".

9) Observe display of latitude and new position, "Verb 06 Noun 66". If required, continue from step #3. When required results have been obtained, terminate by keying in "Verb 34 Enter".

b. PIPA Test Operation

1) Turn transfer switch to Computer Control.

2) Start test by keying in "Verb 71 Enter".

3) Computer should request mark by "Verb 51" flashing. Align SEXTANT line of sight to #1 collimator. When aligned depress "Mark", and if good accept by keying "Verb 52 Enter". Now align SEXTANT line of sight to #4 collimator. Depress "Mark" and accept by keying "Verb 52 Enter".

4) Observe mode change to Zero Encode. Observe "Verb 06 Noun 66" flashing.

5) If latitude and position, as displayed in R1-R2 and R3 respectively, are correct then proceed to step 6. To change latitude execute "Verb 21 Noun 02 Enter, 0763 Enter, major part of double precision latitude Enter, Verb 22 Noun 01 Enter, 0763 Enter, minor part of double precision latitude Enter". To change stable member position execute "Verb 21 Noun 02 Enter, 1205 Enter, position number (corresponding to preceeding table of orientations) Enter".

6) If display is now correct, proceed by keying "Verb 33 Enter"; flash should go out. Observe mode changes through coarse align to fine align.

7) Observe "Verb 06, Noun 66, R3 = position + 1, 000" flashing. Record local *G* as measured by the relevant accelerometer displayed in R1 and R2. Display CM/SEC/SEC and fractions thereof.

8) Switch the transfer switch to manual --switch to Coarse Align and return CDU counters to zero. Observe Program Alarm Light. Switch to Fine Align and return transfer switch to Computer Control.

9) Key in "Verb 33 Enter". Observe "Verb 06 Noun 66" flashing. Continue test from step #5. When all six PIPA sensitivities have been measured, terminate by keying "Verb 34 Enter".

IN-FLIGHT ALIGNMENT SUBROUTINES

This program section is composed of a set of nine interpretive subroutines to be used in the alignment program. They are fundamentally important for the geometrical problem of handling the many coordinate axes needed in the alignment process. Each of the subroutines is called with an interpretive ITC.

1. CALCGTA - Calculate Gyro Torquing Angles

In the fine align procedure, after the present platform orientation is determined, the torquing angles required to move the platform into the desired orientation must be computed. This is achieved as follows:

Let $\underline{x}_D$, $\underline{y}_D$, and $\underline{z}_D$ be the desired stable member axes referred to present stable member orientation. The rotations are performed in three steps: (1) rotating through θ_Y about the y axis, yielding $\underline{x}'_D$, $\underline{y}'_D$, $\underline{z}'_D$; (2) rotating through θ_Z about the z' axis, yielding $\underline{x}''_D$, $\underline{y}''_D$, $\underline{z}''_D$; (3) and finally rotating through θ_X about the x'' axis, yielding $\underline{x}''_D$, $\underline{y}''_D$, $\underline{x}''_D$. The relevant equations are as follows:

$$\underline{z}'_D = \text{UNIT}(-x_{D,3},\ 0,\ x_{D,1})$$

$$\sin\theta_y = z'_{D,1}$$

$$\cos\theta_y = x_{D,1}$$

$$\theta_y = \text{ARCTRIG}(\sin\theta_y,\ \cos\theta_Y$$

$$\sin\theta_z = x_{D,2}$$

$$\cos\theta_z = z'_{D,3}\,x_{D,1} - z'_{D,1}\,x_{D,3}$$

$$\theta_z = \text{ARCTRIG}(\sin\theta_z,\ \cos\theta_z)$$

$$\cos\theta_x = \underline{z}'_D \cdot \underline{z}_D$$

$$\sin\theta_x = \underline{z}'_D \cdot \underline{y}_D$$

$$= \text{ARCTRIG}(\sin\theta_x, \cos\theta_x)$$

The inputs to this subroutine are the three coordinate axes of the desired stable member orientation referred to the present stable member orientation (See subroutine AXISGEN for the technique used in computing these inputs.) These are written as three half-unit vectors contained in locations XDSMPR, YDSMPR, ZDSMPR. The outputs are the three torquing angles to be applied in order about the y, z, and *x* axes of the stable member, respectively and are stored in locations IGC, MGC, and OGC. Note that these rotations are not commutative but the computation is consistent with the ordering of the outputs from subroutine IMPULSE (q. v.). The torquing angles are scaled $(2\pi/2^{28})$ radians /bit and *so* must be right-shifted eight places before transmission to IMUPULSE, whose inputs must be scaled as gyro pulses; i. e. $(2\pi/2^{20})$ radians / bit.

2. ARCTRIG - Precision Inverse Trigonometric Functions

Given sin 8 and cos 8, this program computes 8. Either $\sin^{-1}$ or $\cos^{-1}$ is chosen *so* as to yield maximum accuracy. Inputs to the subroutine are double precision $\sin\theta$ and cos 8 scaled by **1/4** and stored in SINTH and COSTH, respectively. The output, θ, is scaled in revolutions and stored in THETA on return.

3. SMNB - Stable Member to Navigation Base Transformation

Let Q_1, Q_2, and Q_3 be the following rotation matrices:

$$Q_1 = \begin{pmatrix} \cos IG & 0 & -\sin IG \\ 0 & 1 & 0 \\ \sin IG & 0 & \cos IG \end{pmatrix}$$

$$Q_2 = \begin{pmatrix} \cos MG & \sin MG & 0 \\ -\sin MG & \cos MG & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$Q_3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos OG & \sin OG \\ 0 & -\sin OG & \cos OG \end{pmatrix}$$

where IG, MG, and OG represent the inner, middle and outer gimbal angles, respectively.

Given a vector $\underline{v}_{SM}$ in stable member coordinates, this program calculates

$$\underline{v}_{NB} = Q_3 Q_2 Q_1 \underline{v}_{SM}$$

On entering, the argument should be placed in **VAC** and the result will be found there on return. (Usually $\underline{v}_{SM}$ is a half-unit vector.) The CDU angle readings representing the gimbal angles should be stored sequentially in order **x, y, z** (outer, inner, middle). The base address of these three registers should be in S1 upon entry. On exit $\underline{v}_{NB}$ is left in **VAC.**

4. NBSM - Navigation Base to Stable Member

This routine is the inverse of SMNB. It calculates

$$\underline{v}_{SM} = Q_1^T Q_2^T Q_3^T \underline{v}_{NB}$$

using the same **CDU** conventions as SMNB.

5. CALCGA-Calculate Gimbal Angles

Given a stable member orientation and a Navigation Base orientation both referred to the same coordinate system, this routine calculates the corresponding gimbal angles. The procedure is as follows:

$$\underline{a}_{MG} = \text{UNIT}\,(\underline{x}_{NB} \times \underline{y}_{SM})$$

$$\cos OG = \underline{a}_{MG} \cdot \underline{z}_{NB}$$

$$\sin OG = \underline{a}_{MG} \cdot \underline{y}_{NB}$$

$$OG = \text{ARCTRIG}\,(\sin OG,\ \cos OG)$$

$$MG = \sin^{-1}\,(\underline{y}_{SM} \cdot \underline{x}_{NB})$$

$$\cos IG = \underline{a}_{MG} \cdot \underline{z}_{SM}$$

$$\sin IG = \underline{a}_{MG} \cdot \underline{x}_{SM}$$

$$IG = \text{ARCTRIG}\,(\sin IG,\ \cos IG)$$

Inputs are half-unit vectors along the three stable member axes and the three navigation base axes stored in XSM, YSM, ZSM, XNB, YNB, and ZNB, respectively. The output is left in OGC, MGC. IGC in 2's complement single precision. A middle gimbal angle in excess of 60° will cause an alarm.

6. SXTNB - Sextant to Navigation Base Transformation

Let PA and SA be the precision and shaft angles, respectively. SXTNB computes the following:

$$\underline{s}_{NB} = \begin{pmatrix} \sin PA \cos SA \\ \sin PA \sin SA \\ \cos PA \end{pmatrix}$$

On arrival, **X1** (interpretive index register 1) should contain the complement of the address of a MARK storage area (see "Mode Switching and MARK routines"); i.e., with sampled OPTICS CDU angles in relative addresses **3** and 5. The output is a half-unit vector stored in STARM.

7. AXISGEN - Coordinate Axes Generator

Given two half-unit vectors (usually star vectors), $\underline{s}_A$ and $\underline{s}_B$, expressed in two coordinate systems, denoted by primed and unprimed characters, i.e., $\underline{s}'_A$, $\underline{s}'_B$, $\underline{s}_A$, $\underline{s}_B$, this program computes the half-unit vectors $\underline{x}_D$, $\underline{y}_D$, $\underline{z}_D$ which are the primed coordinate system axes referred to the unprimed coordinate system, This is accomplished by defining two ortho-normal coordinate sets, one in each system, in the following manner:

$$\underline{u} = \underline{s}_A$$

$$\underline{v} = \text{UNIT}(\underline{s}_A \times \underline{s}_B)$$

$$\underline{w} = \underline{u} \times \underline{v}$$

$$\underline{u}' = \underline{s}'_A$$

$$\underline{v}' = \text{UNIT}(\underline{s}'_A \times \underline{s}'_B)$$

$$\underline{w}' = \underline{u}' \times \underline{v}'$$

Then

$$\underline{x}_D = u'_1 \underline{u} + v'_1 \underline{v} + w'_1 \underline{w}$$

$$\underline{y}_D = u'_2 \underline{u} + v'_2 \underline{v} + w'_2 \underline{w}$$

$$\underline{z}_D = u'_3 \underline{u} + v'_3 \underline{v} + w'_3 \underline{w}$$

The inputs consist of the four half-unit vectors stored as follows:

$\underline{s}_A$ in 6 - 11 of the work area

$\underline{s}_B$ in 12 - 17 of the work area

$\underline{s}'_A$ in STARAD

$\underline{s}'_B$ in STARAD + 6

The outputs are left in the XDC, YDC, and ZDC as half-unit vectors, the inputs being destroyed in the process.

8. CALCSXA - Calculate Sextant Angles

Given a half-unit star vector $\underline{s}_{SM}$ in stable member coordinates, this routine computes the angles SA and PA required to position the optics such that the line of sight lies along the star vector.

$$\underline{s}_{NB} = \text{SMNB}(\underline{s}_{SM})$$

Let

$$\underline{s}_{NB} = (s_1, s_2, s_3)$$

$$\underline{s}'_{NB} = \text{UNIT}(s_1, s_2, 0) = (\cos\theta, \sin\theta, 0)$$

$$\text{SA} = \text{shaft angle}$$

$$= \text{ARCTRIG}(\cos\theta, \sin\theta)$$

$$\text{PA} = \sin^{-1}(\underline{s}_{NB} \cdot \underline{s}'_{NB})$$

The input is a half-unit vector in location STAR. The outputs are SA left in SAC scaled $2\pi / 2^{29}$ radians /bit (half revolutions) and **PA** left in **PAC** scaled $2n / 2^{31}$ radians /bit (eighth revolutions). SMNB is called by this routine. An alarm is generated if the star is out of the field of view in the present orientation of the optics.

9. SXTANG - Calculate Sextant Angles

This routine is identical to CALCSXA except that the inputs are the star vector and navigation base axis vectors given in half unit vectors referred to the same coordinate system. The inputs should be left in **STAR,** XNB, YNB, and ZNB, respectively, the outputs being the same as CALCSXA.

PINBALL

The Keyboard and Display System Program, referred to as PINBALL, processes digital information exchanged between the Apollo Guidance Computer (AGC) and the computer operator. The initiation for these exchanges is primarily caused by operator action, but information exchange can be initiated by internal computer programs.

The modes of operation are as follows:

1. Display of Internal Data. Both a one-shot display and a periodically updating display (called monitor) are provided.
2. Loading External Data. As each numerical character is punched in, it is displayed in the appropriate Display Panel location.

The data involved in both loading and display can be presented in either octal or decimal, as the operator indicates. If decimal is chosen, the appropriate scale factors are supplied by the program.

3. Program Calling and Control. The Keyboard System is used to initiate a class of routines which are concerned with neither loading nor display. Typical of this class are test routines and system routines. Certain of these require instructions from the operator to determine whether to stop or continue at a given point.
4. Changing Major Mode. The initiation of large scale mission phases can be commanded by the operator.

The inputs to the Keyboard and Display Program are from the direct-wire keyboard and the remote keyboard which transmits via Uplink. Both of these are character-by-character inputs. In addition, there are internal machine program requests of the Keyboard and Display Program.

1. Input/Output Interfaces of Keyboard and Display System Program

The external inputs to the Keyboard and Display System Program are the direct-wire Keyboard and the Uplink. The output is the Electroluminescent Display Panel.

1.1 Direct-wire Keyboard

The Keyboard contains the following characters: VERB, NOUN, +, -, the numerical characters from *0* through 9, CLEAR, ENTER, ERROR LIGHT RESET, and KEY RELEASE. Each of the characters is represented by a 5 bit binary code (see Section 5). The Keyboard code is transmitted to the computer over a **5** wire link and is placed into bits 1-5 of INO.

Each depression of a Keyboard button activiates INTERRUPT #4 (KEYRUPT), as well as placing the key code into INO. This KEYRUPT program picks up the key code and enters a request to the Executive Routine for the program which decodes and digests the key code (CHARIN). Then a RESUME **is** executed, terminating the KEYRUPT.

1.2 UPLINK Inputs

The Uplink is the digital telemetry system which sends information from the ground to the airborne computer. Each time a word is received by the Uplink, INTERRUPT #5 (UPRUPT) is activated. UPRUPT picks up the transmitted code (these codes are the same as **key** codes) and enters a request to the Executive Routine for the program which decodes and digests the key code (CHARIN). Then a RESUME in executed, terminating the UPRUPT. Note that CHARIN makes no distinction between inputs from the Keyboard and inputs from the Uplink.

There is a toggle switch which is used either to accept Uplink inputs, or to block the Uplink, In the blocked position, the operator has chosen not to accept any keyboard type of input from the ground.

1.3 The MARK Button

When the MARK button is depressed, **a** "1" is placed in bit **15** of INO **and** KEYRUPT is activated.' The **MARK** part **of** KEYRUPT picks up

and stores seven quantities: three IMU angles, two Optics angles, and double precision time. The KEYRUPT is then terminated with a RESUME.

1.4 The Display Panel

1.41 Description

The Display Panel consists of 24 electroluminescent sections arranged as in Fig. 1-1. Each section is capable of displaying any decimal character or remaining blank, except the 3 sign sections (R1S, R2S, R3S). These display a plus sign, a minus sign, or a blank. The numerical sections are grouped to form 3 data display registers, each of 5 numerical characters; and 3 control display registers, each of 2 numerical characters. The data display registers are referred to as R1, R2, R3. The control display registers are known as Verb, Noun, and Major Mode.

The Major Mode display register is used to indicate which phase of the mission or large system program is operating. The Verb and Noun display registers are used to indicate the activity of a smaller class of programs, such as displays, loads, etc. These may be initiated by keyboard action, or from within the computer by program action.

1.42 Activation

Each Display Panel character is controlled by a group of 5 latching relays. Once these relays are activated, the appropriate character remains visible on the Display Panel until the state of these relays is changed. The 5 bit relay codes for each numerical character are listed in Section 5.

All the information necessary to operate the Display Panel is transmitted from the computer through output register OUTO. Two Display Panel characters are activated by OUTO at a time. Bits 1-5 (bit 1 is the low order bit) of OUTO operate the right character of the selected pair; bits 6-10 operate the left character of the pair. Bit 11 is used for special one bit functions, such as signs, flash, etc. Bits 12-15 (bit 15 is the high order bit) which are known as the Relayword code, select the appropriate pair of Display Panel characters. See Section 5 for details.

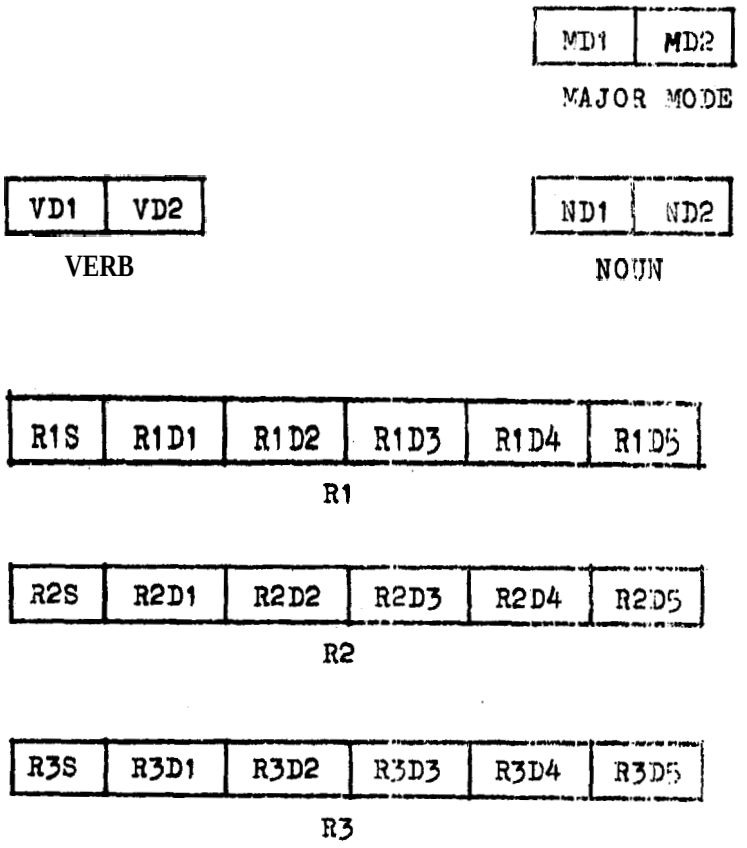


Figure 1-1. Display Panel

1.43 Timing

The 15 bit word which activates a pair of characters in the Display Panel is placed into OUTO by DSPOUT, a program operating in the Interrupt mode. DSPOUT is part of a larger Interrupt program called T4RUPT (INTERRUPT #3). T4RUPT is activated approximately once every 60 milleseconds. Every second T4RUPT (or approximately 120 milleseconds) T4RUPT calls DSPOUT. Thus a new group of Display Panel relays can be selected approximately every .12 seconds.

1.44 Display Panel Buffer

DSPOUT obtains the 15 bit word, to be placed in OUTO from an 11 register buffer called DSPTAB. Each DSPTAB register contains the information to activate a pair of numerical characters (and perhaps a special single bit function, such as sign) in the Display Panel. By retaining this information, the DSPTAB reflects the present state of the entire Display Panel. Thus it is possible to compare new information with that already displayed and to place new data in OUTO only when they differ.

2. Keyboard Use of Keyboard and Display System

The Keyboard and Display System Program can be initiated either by external manipulation of the Keyboard or by internal computer program action, The emphasis of this chapter will he on Keyboard use.

2. 1 Keyboard Operating Format..

The basic language of communication between the operator and the Keyboard and Display System is a pair of words known as Verb and Noun. Each of these is represented by a 2 character octal number. The Verb code indicates what action is to be taken (operation); the Noun code indicates to what this action is applied (operand). Typical Verbs are those for displaying, loading, etc. Nouns usually refer to a group of erasable registers within the computer memory. The Verb codes are listed in Section **6;** the Noun codes in Section 7.

2. 11 Standard Verb-Noun Activation Procedure

The standard procedure for the execution of a keyboard operation consists of a sequence of 7 key depressions.

[VERB] V_1V_2 [NOUN] N_1N_2 [ENTER]

The VERB key depression blanks the Verb lights on the Display Panel (VD1 and VD2), and clears the Verb code register within the computer. The next two numerical characters punched in are interpreted as the Verb code (V_1, V_2 in octal). Each of these characters is displayed in the Verb lights on the Display Panel as it is punched in. The NOUN key operates similarly for the NOUN lights and NOUN code register.

The depression of the ENTER key causes the performance of the Verb-Noun combination appearing in the lights at the time of depression. Thus it is not necessary to follow any order in punching in the Verb or Noun code. They may be done in reverse order, or an old Verb or old Noun may be used without repunching it.

No action is ever taken in performing the Verb-Noun combination until ENTER is pressed. If an error is noticed in either the Verb code or the Noun code before the ENTER is pressed, correction is simple. Merely press the VERB or NOUN key and repunch the originally intended code, without necessarily changing the other. Only when the operator has verified that the desired Verb and Noun codes are in the lights should he press the ENTER key and execute the Verb-Noun combination.

2.12 Further Procedure for Data Loading

If the Verb-Noun combination being executed requires data to he loaded by the operator, the flash will be turned on after the Enter which initiated the Verb-Noun execution. This flash turns the Verb and Noun lights off and on about once per second. Data is loaded in 5 character words and is displayed character-by-character in one of the 5 position data display registers R1, R2, or R3 as it is keyed in. Numerical data is assumed to be octal, unless the 5 character data word is preceded by a plus or minus sign. Then it is considered decimal. Decimal data must be loaded in full 5 numerical character words (no zeros may be suppressed); octal data may be loaded with high order zeros suppressed. If decimal is used for any component of a multi-component Load Verb, it must be used for all components of that Verb. No mixing of octal and decimal data is permitted for different components of the same Load Verb.

The ENTER key must be pressed after each data word. This tells the program that the numerical word being punched in is complete. The flash is turned off after the last ENTER of a loading sequence is pressed,

2.13 Acceptance of Keys

The numerical keys, the **CLEAR,** and the sign keys are rejected if struck after completion (final ENTER) of a data display or data load Verb. At such time, only the Verb, Noun, Enter, Error Light Reset, or Key Release are accepted, Thus the data keys are accepted only after the control keys have instructed the program to accept them.

Similarly the +, - keys are accepted just before the first numerical character of R1, R2, or R3 is punched in, and at no other time. The *8* or

9 key is accepted only while loading a data word into R1, R2, or R3 which was preceded by a + or - sign.

If more than two numerical characters are punched in while loading the Verb, Noun, or Major Mode code; or more than five numerical characters, while loading a data word; the excess characters are not accepted.

2.14 Release of Keyboard and Display System

The Keyboard and Display System Program can be used by internal computer programs as a subroutine (see Chapter 3). However, any operator keyboard action (except Error Light Reset) makes the Keyboard and Display System Program busy to internal routines. The operator has control of the Keyboard and Display System until he wishes to release it. Thus he is assured that data he wishes to observe will not be replaced by internally initiated data displays. In general, it is recommended that the operator release the Keyboard and Display System for internal use when he has temporarily finished with it. This is done by pressing the Key Release button. (Verb 35, Release Display System, is included for early Display Panels which had no Key Release button.) For more details, see Section 3.

2.2 General Verb, Noun Conventions

2.21 Noun Conventions

A Noun code can refer to a device, a group of computer erasable registers, a group of counter registers, or may serve merely as a label. A label Noun refers to no particular computer register, but conveys information by its Noun code number only. The group of registers to which a Noun code refers may be a group of 1, 2, or 3 members, These are generally referred to as 1, 2, or 3 component Nouns (the component is understood as a component member of the register group to which the Noun refers, and not necessarily in the sense of components of a vector or multiprecision number, although this may be the case. The meaning of component should be clear for any particular Noun.) The machine addresses for the registers to which a Noun refers are stored within the computer in Noun Tables.

There are two classes of Nouns, Normal and Mixed. A Normal Noun is one whose component members refer to computer registers which have consecutive addresses, and whose component members all use the same scale factor when converted to decimal. A Mixed Noun is one whose component members refer to non-consecutive machine addresses, or whose component members require different scale factors when converted to decimal, or both.

The Nouns which refer to counter registers are primarily intended to be used with Display Verbs. They are not normally loaded, but the capability exists and may he useful in special applications. Some Nouns refer to registers intended for command numbers (i.e., a desired new angle command). These are intended primarily to be used with Load Verbs, but may be displayed if desired.

2.22 Verb Conventions

As discussed in Section 2.21, a single Noun code refers to a group of 1, 2, or 3 component members. It is the Verb code that determines which component member of the Noun group is operated on. Thus there are 5 different Load Verbs. See Section 6. Verb 21 is required for loading the first component of whatever Noun is used therewith; Verb 22 loads the second component of the Noun; Verb 23, the third component; Verb 24, the first and second components of the Noun; and Verb 25 loads all three components of the Noun. A similar component format is used in the Display and Monitor Verbs.

2.23 Decimal Conversion

When decimal is used in loading or displaying data, conversion is done by interpreting the number as a fraction. For other than fractional representations, scale factors are applied after the fractional conversion to binary when loading; before the fractional conversion to decimal when displaying. The scale factors for Nouns that are to handle decimal data are kept in Noun Tables within the computer. These Noun Tables also keep information on how many component members are associated with each Noun that handles decimal data.

When the Decimal Display Verb is employed, all the component members of the Noun being used are scaled as appropriate, converted to decimal, and displayed in the data display registers. This Decimal Display Verb is the only exception to the component selection feature of Verbs.

If decimal is used for loading data of any component of a multi-component Load Verb, it must be used for all components of that Verb. Thus no mixture of decimal and octal data is permitted for different components of the same Load Verb.

2.3 Data Loading

Whenever any data is to be loaded by the operator, the Verb and Noun lights flash. This flashing occurs when the loading of data is required by procedures initiated either by operator keyboard action or by internal program action. Also, the appropriate data display register (R1, R2, or R3) is blanked, and the internal computer storage register is cleared in anticipation of data loading. Each numerical character is displayed in the proper data display register position as it is punched in. Each of the data display registers (R1, R2, R3) can handle 5 numerical characters (sign is optional). If an attempt is made to key in more than 5 numerical characters in sequence, the sixth and subsequent characters are simply rejected. They do not write over the last valid character.

The +, - keys are accepted only just before the first numerical character of R1, R2, or R3 is punched in. The signs are simply rejected if punched at any other time. If the 8 or 9 key is punched at any time other than while loading a data word which was preceded by a + or - sign, it is rejected and the Check Fail Light is turned on.

As data is loaded, it is temporarily stored in intermediate buffers. It is not placed into its final destination, as specified by the Noun code, until the final ENTER of the load sequence is punched in.

2.31 The CLEAR Button

The CLEAR Button is used to remove errors in loading data as it is displayed in R1, R2, or R3. It does nothing to the Major Mode,

Noun, or Verb lights. (The Noun lights are blanked by the NOUN key; the Verb lights, by the VERB key.) In the following discussions, the term Clearing Function will be used to mean: hlanking the data display register of interest and placing +0 into the internal storage register associated with that display register.

For single component Load Verbs or "Machine Address to be Specified" Nouns, the CLEAR button depression performs the Clearing Function on whichever register is being loaded, provided that the CLEAR is punched before the data ENTER. Once the ENTER is depressed, the CLEAR does nothing. The only way to correct an error after the data ENTER for a single component Load Verb is to begin the Load Verb again.

For the 2 or 3 component Load Verbs, there is a backing up feature of CLEAR. The first depression of the CLEAR key performs the Clearing Function on whichever register is being loaded. (The CLEAR may be pressed after any character, but before its ENTER.) Consecutive depressions of CLEAR perform the Clearing Function on the data display register above the current one, until R1 is cleared. Any attempts to back-up beyond R1 are simply ignored.

The backing-up function of CLEAR operates only on whatever data is pertinent to the Load Verb which initiated the loading sequence. For example, if the initiating Load Verb was a load second component only, no backing-up action is possible.

2.32 Loading Sequence

The normal use of the flash is with a Load Verb. In multi-component load situations, the appropriate single component Load Verbs are flashed one at a time. Thus the computer always instructs the operator through a loading sequence. For example, consider a three component loading sequence. The operator (or the internal program, for that matter) initiates the sequence by punching in VERB = 25, "load 3 components of:" (any noun will do). The Verb code is changed to 21, "load first component of:," and the flash is turned on. Verb 21 continues to be flashed as the operator punches

in the first word of data. When the ENTER is pressed, the Verb code is changed to 22. Flashing continues. The operator punches in the second data word. When ENTER is pressed, the Verb code is changed to 23, "load third component," and the flash continues. The third data word is punched in. When ENTER is pressed, the flash is turned off, and all three data words are placed in the locations specified by the Noun. Notice that throughout the changing of the Verb codes, the Noun code was left unchanged.

There are two special cases when the flash is used with Verb other than Load Verbs.

Case I. Machine Address to be Specified

There is a class of Noun available to allow any machine address to be used. These are called "Machine Address to be Specified" Nouns. When the ENTER which causes the Verb-Noun combination to be executed senses a Noun of this type, the flash is immediately turned on. The Verb code is left unchanged. The operator should load the 5 octal character complete machine address of interest. It is displayed in R3 as it is punched in. If an error is made in loading the address, the CLEAR may be used to remove it. Pressing ENTER causes continuation in executing the Verb.

Case II. Change Major Mode

To change Major Mode, the sequence is:

VERB 37 ENTER

This causes the Noun display register to be blanked and the Verb code to be flashed. The 2 octal character Major Mode code should then be loaded. For verification purposes, it is displayed as it is loaded in the Noun display register. The ENTER causes the flash to be turned off, a request for the new Major Mode to be entered, and a new Major Mode code to be displayed in the Major Mode display register. This

new Major Mode display is usually the logical "OR" of the old and new Major Mode codes. (See a forthcoming report by C. A. Muntz for details on Major Modes.)

2.33 Conclusion of Loading

The flash is turned off only by three events: 1) the final ENTER of a load sequence, 2) the VERB = Terminate, 3) the VERB = Proceed Without Data. It is important to conclude every Load Verb by one of the above three, especially if the load was initiated by program action within the computer. If an internally initiated load is not concluded validly, the program that initiated it may never be recalled.

The "Proceed Without Data" Verb is used to indicate that the operator is unable or wishes not to supply the data requested, but wishes the initiating program to continue as best it can with old data. The "Terminate" Verb is used to indicate that the operator chooses not to load the requested data, and also wishes the requesting routine to cease operation. See Section 3 for further explanation.

2.4 Program Initiation

There is a class of programs of larger scope than the data loading and display programs which the Keyboard can call. This class is typically concerned with system testing and operation. Of course, some of these programs will need to display data or request the loading of data or commands by means of the Keyboard and Display Program. The details of this will be described in Section 3.

The group of Verbs that perform this general program calling is called the Extended Verbs. Their Verb codes are in the range 40-77. The pertinent programs are called directly upon ENTER as part of the Keyboard and Display Program Executive Job, and with its same priority. The Verbs that call programs of relatively short duration should be executed directly and then should end the Keyboard and Display Program with a TC ENTRET. Those of longer duration should place a request to the Executive Routine for the main body of the program and end the Keyboard and Display Program with a TC ENTRET, The Keyboard and Display System is released before the Extended Verbs (codes 40-77) are called, so that they may use the Keyboard and Display System Program.

2.5 Data Monitor

There is a class of Verbs which display data every one-half second. Once a Monitor Verb is executed, the data in the Display Panel continues to be updated until the Monitor is turned off.

The Monitor is turned off by: the Error Reset Button, Verb 34 (Terminate), an alarm condition within the Keyboard and Display System Program (see Section 4.1), an internal program initiation of the Keyboard and Display System Program (a NVSUB call that finds the Keyboard and Display System available), and by Fresh Start.

Monitor action is suspended (but not ended) by the depression of any key, except Error Reset. Monitor action continues after the Keyboard and Display System is released. Thus it is now possible to suspend a monitor while the operator loads some data, or requests another display; and,to return to the original monitor when his intervention is concluded. In monitor of "Machine Address to be Specified," the machine address is re-displayed after resumption of a suspended monitor.

If a second monitor is requested by the operator while an earlier monitor is still active, the second monitor takes over the first, which ends. Thus multiple monitors cannot occur.

3. Internal Use of Keyboard and Display Program

The Keyboard and Display System Program is available to other internal computer programs in its entirety for subroutine use. Use is limited to routines operating under Executive Routine control. Routines operating in the Interrupt Mode may not use the Keyboard and Display Program as a subroutine.

The Verb and Noun Codes themselves are used as the language for communicating between other internal programs and the Keyboard and Display System Program. Any Verb-Noun combination available to the Keyboard can be called by internal routines.

3.1 Interlocks for Internal Use of Keyboard and Display Program

It is necessary to place certain limits on when an internal routine may use the Keyboard and Display Program. This is necessary to prevent the situation in which an internally initiated procedure might write over data in the Display Panel that the operator has not had time to digest.

The two interlocks are: an operator/internal interlock, and an internal/internal interlock.

The operator/internal interlock (called DSPLOCK) is made busy to internal routines by any keyboard activity (except Error Light Reset). It remains busy until it is "released" by the Key Release button or by certain special Verbs. See Section **3. 32.**

The internal/internal interlock (called GRABLOCK) is made busy to other internal routines once any internal routine has "grabbed" the Display System. It remains busy until **10** seconds after the grabbing routine has "freed" it. These procedures will be described in detail in Section **3. 3.**

3. 2 Typical Calling Sequence for Internal Use of Keyboard and Display Svstem

A complete calling sequence for typical internal use of the Keyboard and Display System Program is:

```
L,      TC GRABDSP
L+1,    Return here if Display System is already grabbed.
L+2,    Return here means you have Display System
        grabbed.

        _____
        _____
        _____

M,      TC NVSUB (with Verb-Noun code in A register)
M+1,    Return here if Display System is busy (operator).
M+2,    Return here after execution of Verb/Noun.

        _____
        _____
        _____

        NVSUB may be used more than once while Display
        System is grabbed.

        _____
        _____
        _____

        TC FREEDSP
```

3.3 Details of Calling Keyboard and Display Program for Internal Use

The details of the Calling Sequence shown in Section 3.2 will be described here. Also the formats and return option will he explained.

3.31 "GRABBING" the Display System (GRABDSP)

Any internal routine that wishes to use the Keyboard and Display Program must first "GRAB" the Display System by executing a TC GRABDSP. (This routine is in directly accessible fixed memory.) If the TC GRABDSP was done at location L, the return is to L+1 if the System is already grabbed (GRABLOCK is busy). If GRABLOCK shows that the Keyboard and Display System is free, the return is to L+2, indicating that the calling routine now has the System "grabbed".

The decision for what to do if the Keyboard and Display System is found already grabbed is left to the calling routine. It may use an optional routine called GRABUSY, which enters it into a list waiting for the Keyboard and Display System to be freed. Or it may prefer to continue and attempt the TC GRABDSP again later.

The calling sequence for the optional GRABUSY is:

```
CAF WAKECADR
TC GRABUSY
```

GRABUSY is in directly accessible fixed memory. It puts the calling JOB to sleep, and enters the specified CADR into a waiting list. This CADR is the location at which you wish your JOB to wake up when the Display System is freed. The list is searched and the CADR's are called in the sequence they were inserted according to the following convention. After a TC FREEDSP, the internal/internal interlock (GRABLOCK) is kept busy for 10 seconds, after which a CADR is called from the list and the JOB waked up. This 10 second delay insures that all internally initiated information for display will be visible.

There is a special entrance to GRABUSY called PREGBSY for routines in banks that wish the CADR of 1+ (Location from which the TC PREGBSY was done) to be entered in the list.

3.32 Internal Calling of a Verb/Noun Combination (NVSUB)

The communicator for calling the Keyboard and Display Program is called NVSUB. It is in directly accessible fixed memory. Any Verb-Noun combination available to the Keyboard can be called by NVSUB. The format is to place a word in the A register with the 6 bit Verb code in bits 7-12; the 6 bit Noun code, in hits 1-6. The TC NVSUB causes the Verb code to he displayed in the Verb lights, the Noun code to be displayed in the Noun lights, and the Verb-Noun combination to be executed. Notice that Verb 00 and Noun 00 are both illegal codes, if punched in by the operator. Advantage is made of this fact in the following situation. If the program calling NVSUB wishes to display a Verb code without executing it, or to display a Noun code without executing it, the contents of A should be set *so* that the other 6 bits are all zero. Thus to display the Verb code only, bits 1-6 are set to *0,* and the Verb code is placed into bits 7-12. To display the Noun code only, bits 7-12 are set to 0, and the Noun code *is* placed into bits 1-6. The entire action of NVSUB is taken as a subroutine of the program that called NVSUB. Thus NVSUB is run as part of the same Executive Job that called NVSUB, and with the same priority.

If the calling routine wishes to use one of the "Machine Address to be Specified" Nouns, it is necessary only to preload MPAC + 2 with the appropriate Machine Address before the TC NVSUB.

The Keyboard and Display System is made busy (DSPLOCK) to internal users when there is any Keyboard activity (except Error Light Reset). It remains busy until it is released by the Key Release button or by the following Verbs: Release Display System (35), which is included for early versions of the Keyboard which have no Key Release button; Proceed Without Data (33); Terminate (34); Fresh Start (36), Request Executive (20), Request Waitlist (10); also by all the Verbs with codes 40-77 (these are the so-called Extended Verbs).

If the TC NVSUB was done at location M, the return is to M+1 if the Keyboard and Display System is busy (DSPLOCK is busy). If DSPLOCK shows that the System is available, the return is to M+2, after the Verb-Noun combination has been executed. If the Keyboard and Display System. is available, but in the execution of the Verb-Noun combination some alarm condition is found (such as use of an undefined Verb or Noun code, or certain

The decision for what to do if the Keyboard and Display System is found busy **is** left to the calling routine. It may use an optional routine called NVSUBUSY, which enters it into a list waiting for the Keyboard and Display System to be released. Or it may prefer to continue and attempt the TC NVSUB again later.

The calling sequence for the optional NVSUBUSY is:

```
CAF WAKECADR
TC NVSUBUSY
```

NVSUBUSY is in directly accessible fixed memory. It puts the calling JOB to sleep, enters the specified CADR on the top of the waiting list, and turns on the "Key Release" light, telling the operator to release the Display System for internal use. This CADR is the location at which you wish your JOB to wake up when the Display System is released.

The waiting list is searched by **all** internal occurrences of the "Free" Display System; but only by those external occurrences of the "Release" Display System which are in response to use of the NVSUBUSY routine.

There is a special entrance to NVSTJBUSY called PRENVBSY for routines in banks that wish the CADR of (location from which the TC PRENVBSY was done) -2 to be entered in the list.

3.33 "FREEING" the Display System (FREEDSP)

An internal routine that **has** successfully "grabbed" the Keyboard and Display System may execute an unlimited number of calls to NVSUB **before** "freeing" ths System to Other internal users by executing a ***TC*** FREEDSP. (This routine is in directly accessible fixed memory.) **If** the **TC** FREEDSP was done at location N, the return is to N+1.

After a TC FREEDSP, GRABLOCK is kept busy for 10 seconds, after which a CADR is called from the list and the corresponding JOB waked up. The CADR's are called in the order they were inserted, except that any CADR that may have been placed in the list by NVSURUSY is called first. This is done to ensure that a routine which has "grabbed" the System, but has found NVSUB busy (due to keyboard action by the operator) and has used NVSUBUSY will be called before other routines that find the system already grabbed.

3. 4 Keyboard and Display–System Release

The "Key Release" Light (also known as Release Display System) is turned on to indicate to the operator that some internal program has attempted to use NVSUR, and found the Keyboard and Display System busy. Thus the operator should release the Keyboard and Display System by pressing the Key Release button (or by performing Verb 35 on those early keyboards which have no Key Release button).

Note that the waiting list associated with GRABUSY and NVSUBUSY is searched by those occurrences of the "Release" Display System which are in response to the use of the NVSUBUSY. routine. If the Display System is released at any time other than following an internal use **of** NVSUBUSY, the list is not searched even if there are some CADR's in the list due to the use of GRABUSY. These will be called after the routine that has the System grabbed performs a Free Display System.

3. 5 Internally Initiated Loads

If the NVSUB called operation is a load, some additional facts should be pointed out. If the Keyboard and Display System is available, the Load Verb is displayed along with the Noun, the flash is turned on. When everything is set up to receive data from the operator, NVSUB returns ***to*** **M+2** of the calling routine. Now there is a relatively long time until the. completion of the load. The option of what to do at this point is left to the calling routine. There are essentially *two* courses of action.

3. 51 Initiating Programs that Keep Control

Some programs that place requests through NVSUB for data to be loaded will wish to retain control of the computer while the operator loads

the data, These can inspect the contents of an erasable register called LOADSTAT to determine when the data is complete. The conditions of LOADSTAT are:

+0	Waiting for data.
+NZ	The Verb "Proceed Without Data" has been keyed in.
-NZ	The Verb "Terminate" has been keyed in.
-0	The data load has been completed.

3.52 Initiating Programs that Relinquish Control

Some programs that place requests through NVSUB for data to be loaded will desire to give up control of the computer until the data has been loaded. These will be recalled as soon as the data load is completed, if they follow the procedure outlined below.

A routine called ENDIDLE is provided in directly accessible fixed storage. It puts the current Executive Job to sleep, and stores away the location from which the initiating routine did a TC ENDIDLE (call this location P). It also makes NVSUB busy to other internal programs. This allows other Executive Jobs to be run while the actual data loading is taking place.

Recall to the routine that called ENDIDLE is made at the end of the loading sequence, when the Verb "Terminate" is keyed in, or when the Verb "Proceed Without Data" is keyed in, The recall is to:

P+1	For "Terminate".
P+2	For "Proceed Without Data".
P+3	For Data In.

The recall is made by way of an Executive Routine request to wake up the routine that did the TC ENDIDLE. NVSUB is made available just before recall.

4. Alarms and Special Controls

The Keyboard and Display System Program makes use of the Program Check Fail Light (also known as Illegal Order). It is also concerned with the Key Release Light (also known as Release Display System), the Key Release Button (also known as Release Display System), the Flash, and the Error Light Reset Button. This section will describe the way the Keyboard and Display System Program makes use of these alarms and controls.

4. 1 Program Check Fail Light (Illegal Order)

The Program Check Fail Light is turned on by the Keyboard and Display Program when it encounters some improper operating condition. If the Keyboard and Display Program had been initiated by internal computer program (through NVSUB), it returns to M+l (where M is the location from which the TC NVSUB was performed). If the Keyboard was the initiator, it does a TC ENDOFJOB.

The conditions which cause the Keyboard and Display Program to turn on the Program Check Fail Light are:

1. An input code is received from the Keyboard other than those listed in Section,5.
2. The contents of the register used to determine which Display Panel character is to be lighted has exceeded its limit.
3. An 8 or 9 was punched in while loading a word that was not preceded by a + or - sign.
4. An undefined Verb or Noun code has been used. The Verb *00* or Noun *00* are always illegal if punched into the Keyboard.
5. A decimal number which is numerically too large for the scale factor assigned to the current Noun has been loaded through the Keyboard.
6. An attempt has been made to load a decimal number into a register whose scale factor is specified by the Noun as time in hours. (Note that time in seconds is a permissible load.)
7. The DP decimal display Verb has been used with a Mixed Noun.

4. 2 Key Release (Release Display System)

The illumination of the Key Release light indicates that some internal program has attempted to use the Keyboard and Display System

Program and found it busy. The operator should press the Key Release button (or execute Verb 35 in those keyboards which have no Key Release button).

4.3 The Verb, Noun Flash

The Verb and Noun lights are flashed to indicate that the operator should load data It is turned off by any of the usual load sequence conclusions:

1. Finishing the load sequence.
2. Punching in Verb 33, Proceed Without Data.
3. Punching in Verb 34, Terminate.

4.4 Error Light Reset

The Error Light Reset button serves two functions. By direct wire, it turns off the non-program computer alarm lights on the Display Panel. By program action, it takes the following action related to the Display Panel: the Program Check Fail light is turned off, the UPLINK Activity light is turned off, the Telemetry Fail light is turned off, the Program Alarm light is turned off, the DSPTAB buffer is checked for proper format and corrected if necessary. Also, the following program action not related to the Display Panel is taken: Endpulses are unblocked; the Errupt lock is reset; the C Relays corresponding to IMU Fail, CDU Fail, and PIPA Fail are reset.

4.5 Fresh Start

The Fresh Start Program is initiated either by the automatic restart circuitry or by Verb 36. It takes the following action in regard to the Keyboard and Display System: turns off the Display Panel Alarm lights, turns off the Key Release light, blanks the Display Panel Data and Control Registers, releases the Display System (DSPLOCK), frees the Display System (GRABLOCK), terminates the Monitor. It also initializes much of the program control software. See a memo by C. A. Muntz for these details.

5. Input/Output Codes

5.1 Keyboard Input Codes

These codes enter the computer through bits 1-5 of INO. The MSB is placed in bit 5; the LSB, in bit 1.

	MSB LSB
Blank	0 0 0 0 0
0	1 0 0 0 0
1	0 0 0 0 1
2	0 0 0 1 0
3	0 0 0 1 1
4	0 0 1 0 0
5	0 0 1 0 1
6	0 0 1 1 0
7	0 0 1 1 1
8	0 1 0 0 0
9	0 1 0 0 1
VERB	1 0 0 0 1
ERROR RESET	1 0 0 1 0
KEY RELEASE	1 1 0 0 1
+	1 1 0 1 0
-	1 1 0 1 1
ENTER	1 1 1 0 0
CLEAR	1 1 1 1 0
NOUN	1 1 1 1 1

5.2 5 Bit Relay Output Codes for Display Panel

These codes are placed in OUTO. There are two possible orientations. For the right character of a pair, the MSB is placed in bit 5; the LSB, in bit 1. For the left character of a pair, the MSB is placed in bit 10; the LSB, in bit 6.

	MSB LSB
Blank	0 0 0 0 0
0	1 0 1 0 1
1	0 0 0 1 1
2	1 1 0 0 1
3	1 1 0 1 1
4	0 1 1 1 1
5	1 1 1 1 0
6	1 1 1 0 0
7	1 0 0 1 1
8	1 1 1 0 1
9	1 1 1 1 1

5.3 Relayword Format for OUTO

BITS 15-12	BIT 11	BITS 10-6	BITS 5-1
1 0 1 1		MD1	MD2
1 0 1 0	FLASH	VD1	VD2
1 0 0 1		ND1	ND2
1 0 0 0	UPACT		R1D1
0 1 1 1	+R1S	R1D2	R1D3
0 1 1 0	-R1S	R1D4	R1D5
0 1 0 1	+R2S	R2D1	R2D2
0 1 0 0	-R2S	R2D3	R2D4
0 0 1 1		R2D5	R3D1
0 0 1 0	+R3S	R3D2	R3D3
0 0 0 1	-R3S	R3D4	R3D5

The Display Panel Output Buffer (DSPTAB) follows this same format with respect to the low order 11 bits.

6. Verbs

6.1 Verb List

VERB CODE	FUNCTION	DISPLAY LOCATION
00	Illegal	
01	Display (in octal) 1st. component of:	R1
02	Display (in octal) 2nd component of:	R1
03	Display (in octal) 3rd component of:	R1
04	Display (in octal) 1st and 2nd components of:	R1, R2
05	Display (in octal) lst, 2nd, and 3rd components of:	R1, R2, R3
06	Display (in decimal) all component (s) of:	As appropriate
07	DP Decimal Display	R1, R2
10	Enter Request to Waitlist	
11	Monitor (in octal) 1st component of:	R1
12	Monitor (in octal) 2nd component of:	R1
13	Monitor (in octal) 3rd component of:	R1
14	Monitor (in octal) 1st and 2nd components of:	R1, R2
15	Monitor (in octal) lst, 2nd, and 3rd components of:	R1, R2, R3
16	Monitor (in decimal) all component (s) of:	As appropriate
17	Monitor DP Decimal	R1, R2
20	Enter Request to Executive	
21	Write 1st component into:	R1
22	Write 2nd component into:	R2
23	Write 3rd component into:	R3
24	Write 1st and 2nd components into:	R1, R2
25	Write 1st, 2nd, and 3rd components into:	R1, R2, R3
26	Spare	
27	Spare	
30	Spare	
31	Bank Display	
32	Bump displays [c(R2) into R3, c(R1) into R2]	
33	Proceed Without Data	
34	Terminate	
35	Release Display System	
36	Fresh Start	
37	Change Major Mode to:	

EXTENDED VERBS

VEKB CODE	FUNCTION	DISPLAY LOCATION
40	Zero (Used with Noun **ICDU** or **OCDU** only)	
41	Coarse Align **(Used** with Noun **ICDU** or **OCDU** only)	
42	Fine Align IMU	
43	Lock **IMU**	
44	Set **IMU** to Attitude Control	
45	Set **IMU** to Re-entry Control	
46	Return **IMU** to Coarse Align	
47	Turn Optical Tracker On (Not in use yet)	
50	Please Perform	
51	Please Mark	
52	**MARK** Accept (until button is available)	
53	Free (used with Noun **ICDU** or OCDU only)	
54	Pulse Torque ***GY***RO's	
55	Spare	
56	Spare	
57	Spare	
60	Spare	
61	Spare	
62	Spare	
63	Spare	
64	Spare	
65	Spare	
66	Spare	
67	Spare	
70	Perform GYRO Drift **Test**	
71	Perform PIPA Scale Test	
72	Spare	
73	Spare	
74	Spare	
75	Spare	
76	Spare	
77	Spare	

6.2 Verb Descriptions

Verbs 01-05	Perform octal displays of data.
Verb 06	Performs decimal display of data. The scale factors, types of scale factor routines, and component information are stored within the machine for each Noun which is required to display in decimal.
Verb 07	Performs a double precision decimal display of data. It does no scale factoring. It merely performs a 10 character fractional decimal conversion of two consecutive erasable registers using R1 and R2 (the sign is placed in the R1 sign position; the R2 sign position is blank). It cannot be used with Mixed Nouns. Its intended use is primarily with "Machine Address to be Specified" Nouns.
Verb 10	Enters request to Waitlist Routine for any machine address with any delay. This Verb assumes that the desired number of 10 millesecond units of delay has been loaded into the low order bits of the Prio/Delay register (Noun 26). This Verb is used with the "Machine Address to be Specified" Noun. The complete address of the desired location is then punched in. See Section 2.32 Case I.
Verbs 11-17	Updata data displays every one-half second.
Verb 20	Enters request to Executive Routine for any machine address with any priority. This Verb assumes that the desired priority has been loaded into bits 10-14 of the Prio/Delay register (Noun 26). This Verb is used with the Noun "Machine Address to be Specified". The complete address of the desired location is then punched in. See Section 2.32, Case I.
Verbs 21-25	Perform data load. Octal quantities are unsigned. Decimal quantities are preceded by a + or - sign.
Verb 31	Bank Display. This Verb is included to permit displaying the contents of fixed memory in any bank. Its intended use is for checking program ropes and the BANK position of program ropes.
Verb 32	Display Shift. Useful for preserving an existing display of a quantity while displaying another quantity.
Verb 33	Proceed Without Data. Informs routine requesting data to be loaded that the operator chooses not to load fresh data, but wishes the routine to continue as best it can with old data. Final decision for what action should be taken is left to requesting routine.

Verb **34**	Terminate. Informs routine requesting data to be loaded that the operator chooses not to load fresh data, and wishes the routine to terminate. Final decision for what action should be taken is left to requesting routine. If Monitor is on, it is turned off.
Verb **35**	Releases Keyboard and Display System for internally initiated use. Performs the same function as the Key Release button. It is included for use with those early keyboards which have no Key Release button.
Verb **36**	Initializes the program contral software and the Keyboard and Display System Program. See Section **4. 5** for details.
Verb **37**	Changes to New Major Mode. See Section **2. 32,** Case II *for* details.

EXTENDED VERBS

Verb **40**	Must be used with Noun **20** (ICDU) or Noun **55** (OCDU) only.
Verb **41**	Must be used with Noun **20** (ICDU) or Noun **55** (OCDU) only.
Verbs **42-47**	Call programs that perform the indicated ***G*** and N System procedures.
Verb **50**	This Verb is used only by internal routines that wish the operator to perform a certain task. It should never.be keyed in by the operator. It is usually used with Noun **25** (Checklist). The coded number for the Checklist Item to be performed is displayed in register **R l** by the requesting routine. Once the operator has performed the requested action, he should press **ENTER** to indicate that the Checklist Item has been performed. **If** he wishes not to perform the requested action, he should key in the Verb "Proceed Without Data".
Verb **51, 52**	Verb **51** is used only by internal routines that wish the operator to **MARK.** It should never be keyed in by theoperator. **It is** usually used with Noun ***30*** (Star Numbers). The numbers of the stars to be marked are displayed in registers R1, R2, **R3** by the requesting routine.

Verb **51,** 52 (con't)	The operator should indicate completion of each valid mark by Verb 52 (Mark Accept) or by pressing the Mark Accept button when it is available. He may do several MARK's until he gets what he considers a good one before doing Mark Accept. He should never press ENTER with Verb 51.
Verb 53, **54**	Call programs that perform the indicated G and N System procedure.
Verb *70,* **71**	Call programs that perform the indicated G and N System Test.

7. Nouns

7. 1 Noun List

NOUN CODE	NORMAL NOUNS
00	Not in Use
01	Specify Machine Address (Fractional)
02	Specify Machine Address (Whole)
03	Specify Machine Address (Degrees)
04	Specify Machine Address (Hours)
05	Specify Machine Address (Seconds)
06	Specify Machine Address (GYRO Degrees)
07	Specify Machine Address (Y OPTICS Degrees)
10	Spare
11	Spare
12	Spare
13	Spare
14	Spare
15	Increment Machine Address
16	Time Seconds
17	Time Hours
20	ICDU
21	PIPA's
22	New Angles **I**
23	Delta Angles **I**
24	Delta Time (Seconds)

NOUN CODE	NORMAL NOUNS
25	Checklist
26	Prio/Delay
27	Self Test On/Off Switch
30	Star Numbers
31	Failreg
32	Spare
33	Spare
34	Spare
35	Spare
36	Spare
37	Spare
40	Spare
41	Spare
42	Spare
43	Spare
44	Spare
45	Spare
46	Spare
47	Spare
50	Spare
51	Spare
52	Spare
53	Spare
54	Spare

NOUN CODE	MIXED NOUNS
55	OCDU
56	Uncalled Mark Data (OCDU & Time (Seconds))
57	New Angles OCDU
60	ICDUX and Time (Seconds)
61	ICDUY and Time (Seconds)
62	ICDUZ and Time (Seconds)
63	OCDUX and Time (Seconds)
64	OCDUY and Time (Seconds)
65	Time (Hours and Seconds)
66	System Test Results
67	Delta GYRO Angles
70	Spare
71	Spare
72	Spare
73	Spare
74	Spare
75	Spare
76	Spare
77	Spare

7.2 Noun Scale Factors

The Keyboard and Display Routines are able to handle decimal/binary and binary/decimal scale factoring for the following situations.

1) fractional
2) whole
3) degrees (maximum ±359.99)
4) time in seconds (maximum +999.99)
5) time in hours (maximum +745.65). This may be used for display only. No loading of time in hours.
6) GYRO degrees (maximum ± 9.9999)
7) Y OPTICS degrees. There are two different ranges, selected according to bit 13 of register **WASOPSET.** Bit 13 = *0* for range I; but 13 = 1 for range II.

   Range I maximum + 179.99 degrees
   Range II maximum + 89.999 degrees
   (In range II, a bias of 19.775 degrees is added for display; subtracted for loads.)

In all cases, a single precision (5 decimal character) word is handled.

7.3 Noun Descriptions

Noun *00*	Not in use.
Nouns 01-07	The machine address for these nouns is not stored within the machine, but is supplied by the operator. This allows any address to be loaded, displayed, or monitored, When ENTER is pressed with one of these nouns, the Flash is activated, but the Verb is left unchanged. This indicates to the operator that he should type in the 5 character OCTAL address, followed by an ENTER. It is displayed in R3, and the Flash is turned off. Thereafter, the execution of the Verb continues, just as in the case of any other noun. Seven different "Machine Address to be Specified" nouns are provided *so* that any of the decimal scale factors may be applied to the contens of any machine address.
Noun 15	This is used to increment the machine address already specified. It is useful for loading or displaying a group of consecutive addresses. OCTAL only.

Nouns 16-17	Refer to the time counters TIME1, TIME2, and are not normally loaded. If decimal is desired, Noun 16 gives seconds modulo 1000 (maximum 999.99 seconds); Noun 17 gives hours up to a machine maximum of 745.65 hours. *(See* Note 1.)
Nouns 20	Refer to the 3 CDU counters. They are not normally loaded. If decimal is desired, the form is XXX.XX degrees (maximum of +-359.99). (See Note 1.)
Noun 21	Refers to the 3 PIPA counters, and is not normally loaded. In decimal, register contents are presented as whole decimal, representing the number of PIPA counts (maximum ± 16,383). (See Note 1.)
Noun 22	Refers to the desired platform angle registers (THETAD). (Decimal scale, ± 359.99 degrees maximum.)
Noun 23	Refers to the Temporary Data Buffer. (Decimal scale, ± 359.99 degrees maximum.)
Noun 24	Refers to the Temporary Data Buffer. (Decimal scale, +999.99 seconds maximum.)
Noun 25	Usually used with Verb 50, described in Section 6.2. Refers to Temporary Data Buffer. (If decimal is desired, whole scale is used.)
Noun 26	This noun is to be preloaded with the desired PRIORITY for use with VERB = 20 (REQUEST EXECUTIVE); with the desired DELAY when used with VERB = 10 (REQUEST WAITLIST). OCTAL is to be used for the PRIORITY, decimal whole scaling is provided for the DELAY.
Noun 27	+0 loaded into this Noun causes the idle Job to be done for Executive Backup. -NZ loaded into this Noun causes the AGC Self Check routine to be done for the Executive Backup indefinitely. +*NZ* loaded into this Noun causes the AGC Self Check routine to be done for the Executive Backup N times, where N is the number loaded. (If decimal is used, whole number scale.)
Noun *30*	Refers to the Temporary Data Buffer. Intended for star identification numbers. The decimal scale for each is whole. This Noun is usually used with Verb 51, described in Section *6.2.*

Note 1. Since this noun refers to counter registers, it is not normally loaded. However, the capability exists and loading may be useful in special applications.

23 JAN 197

Noun 31	Used to display system failure conditions. A coded number to identify the failure is displayed in R1. (Octal only)

MIXED NOUNS are those whose components refer to non-consecutive addresses or have different scale factors.

Noun 55	Refers to the OCDU counters. (Decimal scale: X OPTICS scale is + 359.99 degrees maximum. Y OPTICS scale is either + 179.99 degrees maximum, or + 89.999 degrees maximum. See Section 3.2.) (See Note 1.)
Noun 56	Refers to Temporary Data Buffer. Noun is used by internal routines to display the OCDU angles and time (seconds) when an unrequested Mark is performed. (Optics decimal scale is same as Noun 55.)
Noun 57	Refers to the desired optics angle registers (DESOPTX, DESOPTY). (Optics decimal scale is the same as Noun 55.)
Nouns 60-64	Refers to each of the ICDU and OCDU counters separately, and time counter (in seconds). The ICDU decimal scales are ± 359.99 degrees maximum, (Optic decimal scales are the same as Noun 55.) (See Note 1.)
Noun 65	Refers to TIME1, TIME2 counters. (Decimal scales: hours scale, + 745. 65 maximum; seconds scale, + 999.99 maximum.) (See Note 1.)
Noun 66	Refers to Temporary Data Buffer. Decimal scales are whole, fractional, whole.
Noun 67	Refers to the desired delta GYRO angle registers (GYROD). Decimal scale is ± 9. 9999 degrees maximum.