7322.3-17 71

# ABORT ELECTRONIC ASSEMBLY

# PROGRAMMING REFERENCE

APRIL 1966

Prepared By: H. L. Stiverson
H. L. Stiverson

Approved By: HB Grossman
H. B. Grossman

Approved By: D. L. Meginnity
D. L. Meginnity

## TABLE OF CONTENTS

## INTRODUCTION

This document is intended as a reference source of data needed for the preparation of Abort Electronic Assembly (AEA) programs. It contains information about the AEA such as its physical description, input and output capabilities, word formats, and operation of programmable orders. It also contains information about the LEM Abort Mission frequently referred to by the programmer such as input and output resolution and rates, and the use of input and output registers and discretes. Contained in the appendix section is information for the use of the AEA Assembly Program, diagrams of coordinate systems in use, and condensed forms of information contained in the body of the reference.

## 1.0 COMPUTER DESCRIPTION

The AEA is a small, high speed, general purpose computer with a substantial amount of special purpose input/output electronics. It employs a fractional two's complement, parallel arithmetic section and parallel data transfer. Instruction words are 18 bits in length, consisting of a 5 bit order code, an index bit, and a single 12 bit operand address. Data words are 18 bits in length including sign. The ferrite core memory system is partially hardwired, partially scratchpad and has a five microsecond cycle time. For purposes of explanation, the computer may be separated into the Memory, the Central Computer, and the Input/Output Section.

### 1.1 CENTRAL COMPUTER

In the Central Computer are eight data and control registers, an 18 bit parallel adder, two timing registers, and associated logic. The data and control registers are interconnected by a parallel data bus. Computer operations are executed by appropriately timed transfers of information between these registers, between the Memory and the M Register, and between the Accumulator and input/output registers. The Adder and three registers, the M Register, the Q Register and the Accumulator, form the basis for execution of arithmetic operations.

#### Adder

The arithmetic section is designed around an 18 bit three microsecond parallel adder. Two 18 bit registers, the Accumulator and the M Register furnish the inputs t the Adder. The sum generated by the Adder is loaded into the Accumulator. Shifting operations are implemented by displacing the sum generated by the Adder one bit right or left when loading it into the Accumulator.

#### Accumulator (A Register)

This 18 bit register communicates in parallel with the Adder and the Data-Bus. In addition, it communicates serially with the Q Register for shifting operations suc as multiplication, division, and double length shifts. The Accumulator holds the results of most arithmetic operations and is the register which is used to communicat with input/output registers.

#### Memory Register (M Register)

The M Register is an 18 bit register which is loaded from the Memory or from the Data Bus. It holds data which is being transferred between the Central Computer nd the Memory via the Data Bus. Data transferred from the Memory is held by the M Register as it is placed on the Data Bus. During this time, the data is also written back into the Memory from the M Register if the transfer of data is a read and restore operation. Data transferred to the Memory from the Central Computer is held by the M Register as it is written into the Memory. The M Register holds the multiplicand during multiplication, the divisor during division, the addend during addition, and the subtrahend during subtraction. Load Accumulator operations are accomplished by clearing the Accumulator and then adding the contents of the M Register to the Accumulat

### Multiplier-Quotient Register (Q Register)

The Q Register is an 18 bit register which communicates in parallel with the Data-Bus and serially with the Accumulator. The Q Register holds the least significant half of the double length product after multiplication, and initially holds the least significant half of the double length dividend for division. After division, it holds the unrounded quotient. For double length shifting operations, the Q Register is logically attached to the low-order end of the Accumulator. After execution of a TSQ instruction, the Q Register holds a transfer instruction with an address field set to one greater than the location of the TSQ instruction. If the Q Register is then stored by the routine which has been transferred into, a convenient means of returning to the main program is provided. For divide operations, an extra bit, Q18, is attached to the Q Register.

### Index Register

The Index Register is a three bit counter which is used for operand address modification. When an indexed instruction is executed, the effective operand address is computed by a logical OR operation between the Index Register and the three least significant bits of the operand address. When a TIX instruction is executed, the Index Register is decremented by one if it is greater than zero, and the next instruction is taken from the location specified by the address field of the TIX instruction. If the Index Register is zero, it is not decremented, and the next instruction is taken in sequence. The Index Register is loaded under program control from the least significant three bits of the data bus.

### Address Register

The Address Register is a 12 bit register which is loaded under computer control from the least significant 12 bits of the Data Bus. It holds the address of the memory location to which access has been requested by the Central Computer.

Operation Code Register

The Operation Code Register is a 5 bit register which is loaded under computer control from the 5 most significant bits of the data bus. It holds the 5 bit order code during its execution.

Program Counter

The Program Counter is a 12 bit counter which is loaded from the least significant 12 bits of the data bus by execution of a transfer instruction. It holds and generates instruction addresses in sequence.

Cycle Counter

The Cycle Counter is a five bit counter which is used to control shift instructions and certain long orders.

Timing

Timing is controlled by two registers, one eight bits in length, and the other three bits in length. These registers produce the timing signals required to control all operations of the Central Computer.

## 1.2 INPUT-OUTPUT SECTION

There are four basic types of registers in the input-output section. These registers operate independently of the central computer except when they are accessed during execution of an input or output instruction. All transfers of data between the central computer and the input-output registers are in parallel. The PGNS Euler Angles are accumulated in three 15 bit integrator registers. The integrator registers shift 15 bit positions upon receipt of each input pulse, serially adding or subtracting one from the previous count. The integrator registers are set to zero by an external signal. ASA gyro and accelerometer pulses are accumulated in six 11 bit counters which are set to zero when accessed by the central computer. The four bit DEDA Register, the 18 bit Input Telemetry Register, the 24 bit Output Telemetry Register, and the 15 bit output register time shared for altitude and altitude rate are all implemented as shift registers. The remaining registers, the D/A Converter registers and the discrete input and output registers, are static registers. The Discrete Output Register is changed by setting or resetting specific bits within the register with specific output instructions.

## 1.3 MEMORY

The AEA memory is a coincident-current, parallel, random access, ferrite core stack with a capacity of 4096 18 bit words. It is divided into two equal sections - temporary storage and permanent storage. The addresses of the temporary storage locations are 0 thru $3777_8$, and the addresses of the permanent storage locations are $4000_8$ thru $7777_8$.

Each core in the temporary storage memory is threaded by an X selection winding, a Y selection winding, a sense winding, and an inhibit winding. In the permanent storage memory, the inhibit winding is omitted, and the X selection winding passes only through cores which represent zeros.

The cycle time of the memory is five μsec.

## 2.0 COMPUTER INSTRUCTIONS

The AEA instruction word consists of a five bit operation code, an index bit, and a single 12 bit address field. Instruction and data formats are shown in Figure 1.

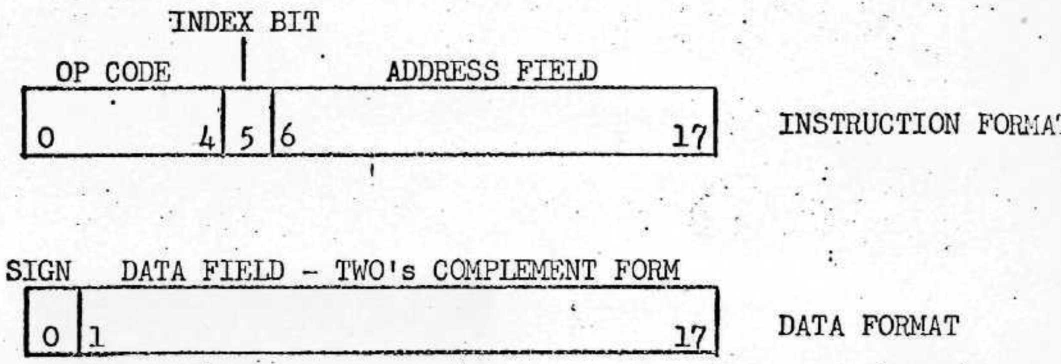


FIGURE 1. AEA INSTRUCTION AND DATA FORMATS.

If the index bit is a one, the address field of any instruction which accesses the Memory is modified by the Index Register. The effective operand address is computed by a logical OR operation between the three bits of the Index Register and the three least significant bits of the operand address.

The AEA executes 27 basic instructions. Four of these instructions, CLZ, ADZ, SUZ, and MPZ, do not include a memory restore cycle after data is read, leaving the addressed memory cell contents equal to zero. These instructions require less power than the corresponding instructions CLA, ADD, SUB, and MPR, and should be used whenever the contents of the addressed memory location need not be retained.

In the subsequent functional definition of AEA program instructions, each description is headed by the following information: A three letter mnemonic code used in symbolic programs to represent the five bit operation code; the letter Y or N to represent the address field in the description of the instruction; a two digit octal number which corresponds to the five bit operation code of the instruction word; and the time, in microseconds, required to execute the instruction. The lette Y, as used in the definitions, represents both the address field of the instruction and the memory location of the operand. This is only the case when no indexing is present. If indexing is present, the memory location which is actually addressed is Y modified by the Index Register. The execution times listed are nominal times. The actual times are obtained by dividing the nominal times by 1.024. Individual bit positions of a register are referred to by the letter which represents the register, followed by the bit position.

## 2.1 ARITHMETIC INSTRUCTIONS

Add

ADD Y 22 10 μsec

The contents of Y are algebraically added to the contents of the Accumulator. If overflow occurs, the overflow indicator is set. The contents of Y are unchanged.

Add and Zero

ADZ Y 32 10 μsec

The contents of Y are algebraically added to the contents of the Accumulator. If overflow occurs, the overflow indicator is set. The contents of Y are set to zero

Subtract

SUB Y 24 10 μsec

The contents of Y are algebraically subtracted from the contents of the Accumulator. The contents of Y are unchanged. If overflow occurs, the overflow indicator is set.

Subtract and Zero

SUZ Y 34 10 μsec

The contents of Y are algebraically subtracted from the contents of the Accumulator. The contents of Y are set to zero. If overflow occurs, the overflow indicator is set.

Multiply

MPY Y 06 70 μsec

The contents of the Accumulator are multiplied by the contents of Y. The most significant half of the double length product is placed in the Accumulator and the least significant half in the Q Register.

A special condition exists when the contents of Y are equal to minus one (Y0 = 1, Y1 thru Y18 = 0). In this case, the contents of the Accumulator will remain unchanged for all initial values of the Accumulator including minus one. The Q Register will contain zeros. If the Accumulator initially contains minus one and Y contains some value other than minus one, multiplication proceeds normally. In this case, the product will be represented by the two's complement of the contents of Y in the Accumulator, and zeros in the Q Register.

Multiply and Round

MPR Y 26 70 μsec

This instruction is identical to the MPY instruction with the exception that the most significant half of the double length product, held in the Accumulator, is rounded. Rounding is implemented as follows: If, after obtaining the double length unrounded product, bit Q1 is a one, a one is added into the least significant bit of the Accumulator. If bit Q1 is a zero, both the Q Register and the Accumulator remain unchanged.

Multiply and Zero

MPZ Y 36 70 μsec

This instruction is identical to the MPR instruction with the exception that the contents of Y are set to zero.

Divide

DVP Y 04 73 μsec

The contents of the double length dividend, the most significant half of which is contained in the Accumulator and the least significant half in the Q Register, is divided by the contents of Y. The sign of the Q Register is disregarded. The rounded quotient is held by the Accumulator and the unrounded quotient by the Q Register.

To prevent greater than fractional quotients, the dividend and divisor must be scaled such that the absolute magnitude of the dividend is less than that of the divisor when represented in the machine. When initial conditions which result in greater than fractional quotients are detected, the overflow indicator is set and the divide operation proceeds normally. One exception to this occurs when the contents of Y are positive and the Accumulator contains an equal negative value. The overflow indicator is not set and the quotient which results is minus one (A0 = 1, A1 thru A17 = 0). If the contents of Y are negative and the Accumulator contains an equal positive value, the overflow indicator is set. The result, however, is also minus one.

The last step of the division process is placing the quotient in the Accumulator and rounding it. Rounding is based upon a determination of the next bit of the quotient. If the additional bit is a one, a one in the least significant bit position is added to the quotient in the Accumulator. If the additional bit is a zero, the Accumulator remains unchanged and equal to the unrounded quotient retained in the Q Register. Rounding is inhibited when it causes overflow in the Accumulator. This condition exists when the Accumulator contains the maximum positive value (A0 = 0, A1 thru A17 = 1). It results from division of a quantity by a quantity of equal sign and only slightly larger magnitude.

Complement Accumulator

COM 60 16 µsec

The contents of the Accumulator are replaced by their two's complement. If the contents of the Accumulator are minus one (A0 = 1, A1 thru A17 = 0) or zero, the Accumulator remains unchanged.

Absolute Value of Accumulator

ABS 62 16 µsec

If the contents of the Accumulator are negative, they are replaced by their two's complement. If the contents of the Accumulator are possitive, zero, or equal to minus one (A0 = 1, A1 thru A17 = 0), the Accumulator remains unchanged.

## 2.2 LOAD AND STORE INSTRUCTIONS

Clear and Add

CLA Y 20 10 µsec

The Accumulator is loaded from Y. The contents of Y remain unchanged.

Clear, Add and Zero

CLZ Y 30 10 µsec

The Accumulator is loaded from Y. The contents of Y are set to zero.

Load Q Register

LDQ Y 14 13 µsec

The Q Register is loaded from Y. The contents of Y remain unchanged.

Store Accumulator

STO Y 10 13 µsec

The contents of the Accumulator are stored in Y. The Accumulator remains unchanged.

Store Q Register

STQ Y 12 13 µsec

The contents of the Q Register are stored in Y. The Q Register remains unchang

## 2.3 SHIFT INSTRUCTIONS

Accumulator Left Shift

ALS N 56 3N + 13 µsec

The contents of the Accumulator are shifted left N places. N is specified by bits 13 thru 17 of the instruction word. If any bit shifted from A1 changes A0, the overflow indicator is set. Zeros are shifted into A17, and bits shifted from A0 are lost.

Long Left Shift

| | | | |
|---|---|---|---|
| LLS | N | 52 | 3N + 13 μsec |

The contents of the Accumulator and bits one thru seventeen of the Q Register are left shifted as one register N places. N is specified by bits 13 thru 17 of the instruction word. The sign of the Q Register is made to agree with the sign of the Accumulator. If any bit shifted from A1 changes A0, the overflow indicator is set. Zeros are shifted into Q17, and bits shifted from A0 are lost.

Long Right Shift

| | | | |
|---|---|---|---|
| LRS | N | 54 | 3N + 13 μsec |

The contents of the Accumulator and bits one thru seventeen of the Q Register are right shifted as one register N places. N is specified by bits 13 thru 17 of the instruction word. The sign of the Q Register is made to agree with the sign of the Accumulator. Bits shifted into A0 are the same as A0, and bits shifted from Q17 are lost.

## 2.4 TRANSFER INSTRUCTIONS

Transfer

| | | | |
|---|---|---|---|
| TRA | Y | 40 | 10 μsec |

The next instruction is taken from Y.

Transfer and Set Q

| | | | |
|---|---|---|---|
| TSQ | Y | 72 | 16 μsec |

The contents of the Q Register are replaced by a transfer instruction with an address field set to one greater than the location of the TSQ instruction, and a one in the index bit position. The next instruction is taken from Y. If the Q Register is stored by the routine to which control has been transferred, a convenient means of returning to the main program is provided. The one in the index bit position of the transfer instruction is disregarded when the instruction is executed.

Transfer on Minus Accumulator

| | | | |
|---|---|---|---|
| TMI | Y | 46 | 10 μsec |

The next instruction is taken from Y if the contents of the Accumulator are negative. If the contents of the Accumulator are positive or zero, the next instruction is taken in sequence.

Transfer on Overflow

TOV Y 44 10 µsec

If the overflow indicator is set, the next instruction is taken from Y, and the overflow indicator is reset. If the overflow indicator is reset, the next instruction is taken in sequence.

## 2.5 INDEX INSTRUCTIONS

Address to Index

AXT N 50 13 µsec

The index register is set to N. N is specified by bits 14 thru 17 of the instruction word.

Test Index and Transfer

TIX Y 42 10 µsec

If the index register is greater than zero, it is decremented by one and the next instruction is taken from Y. If the index register is zero, it remains unchangec and the next instruction is taken in sequence.

## 2.6 TIMING INSTRUCTION

Delay

DLY Y 70

Execution is halted until a timing signal, which is generated at 20 millisecond intervals, is received. The next instruction is taken from Y. If the computer is not in the halt mode when the timing signal is generated, the Test Mode Failure Discrete is set. Execution of this instruction at the completion of each computation cycle provides a means of equalizing the time duration of computational cycles.

## 3.0 INPUT-OUTPUT INSTRUCTIONS

There are two operation codes for this set of instructions, one to address input registers and one to address output registers. The individual registers are addressed by variations in the address field of the input or output instruction word. Section 2.0 defines the format for AEA instructions. The following is a definition of the two input-output instructions:

Input

| | | | |
|---|---|---|---|
| INP | Y | 64 | 16 or 67 µsec |

The contents of the input register specified by Y are loaded into the Accumulator and the register is set to zero or remains unchanged depending upon the register which is selected. To facilitate resetting registers, the addresses of certain registers may be combined. When this is done, each of the input registers involved will be reset or will remain unchanged depending upon the registers which are addressed. When more than one input register is addressed simultaneously, the Accumulator will contain the logical OR of the contents of all of the registers which are addressed. Only addresses which have the same most significant octal character can be combined. To combine the addresses of two or more input registers, add their addresses together algebraically, excluding the most significant character. For exampl combining the addresses of all six of the eleven bit counters results in an address of 6176. Execution time is 67 µsec if a PGNS Angle Register is addressed, and 16 µsec for all other input registers.

The addresses of the individual input registers and input discrete bit positions are given in Table C and Table D of Appendix B.

Output

| | | | |
|---|---|---|---|
| OUT | Y | 66 | 13 µsec |

The contents of the Accumulator are transferred to the output register specified by Y unless Y is the address of a discrete output. If Y is the address of a discrete output, the discrete is set or reset as specified by Y. The addresses of certain groups of output registers and output discretes may be combined. Only addresses with the same most significant octal character may be combined. To combine the addresses of two or more output registers, add their addresses together algebraical excluding the most significant octal digit. When combining the addresses of two or more discrete outputs, this procedure cannot be followed for the set addresses. When adding the set addresses together, the most significant four binary bits are excluded. For example, combining the addresses of the first four discrete set outputs results in an address of 2760. When more than one output register is addressed simultaneously, each register is loaded from the bit positions of the Accumulator which normally load the specific register. Combinations of output discrete addresses cause each of the discretes involved to be set or reset depending upon the four most significant binary bits of the combination.

The addresses of the individual output registers and discretes are given in Table B and Table E of Appendix B.

## 3.1 INPUT REGISTERS

The AEA contains 13 input registers. They are the three fifteen-bit integrating registers for accumulating PGNS Euler Angles, the six eleven-bit counters for accumulati ASA gyro and accelerometer pulses, the eighteen-bit shift register for Downlink Telemetr the seven and eight-bit static registers for Discrete Inputs, and the four-bit shift register for DEDA input and output. Each of these registers is discussed separately in the following sections with emphasis on its function within the Abort Guidance Systen

### Integrator Registers

The three fifteen-bit integrator registers accumulate the PGNS Euler Angles $\theta$, $\phi$, and $\psi$. The inputs to these registers are in the form of pulse train signals which have a repetition rate which ranges from zero to 6.4 kpps. There are two inputs to each register, one for positive pulses and one for negative pulses. The registers shift fifteen-bit positions upon the receipt of each pulse train, serially adding or subtracting one from the previous count. Shifting is inhibited when a register is accessed by the Central Computer. The registers are reset only by receipt of an external signal. When read by an input instruction, the contents of the addressed register are loaded into bits 1 thru 15 of the Accumulator. Bits 16 and 17 of the Accumulator are set to zero.

The three PGNS Euler angles, which are the IMU gimbal angles, have the following characteristics:

a) Scale Factor - $360/2^{15}$ degrees per bit

b) Range - 15 bits

c) Format - two's complement

ASA Input Counters

The six 11-bit ripple counters accumulate incremental angular information about the vehicle x, y, and z axes and incremental velocity changes along the vehicle x, y, and z axes. The inputs to these registers are in the form of pulse train signals which have a repetition rate which ranges from zero to 64 kpps. The registers are incremented by one count upon the receipt of each pulse train. When read by an input instruction, the contents of the addressed register are loaded into bits 1 thru 11 of the Accumulator and the register is set to zero. Bits 12 thru 17 of the Accumulator are set to zero.

During any one millisecond period, the presence of 32 pulses represents a zero signal; the presence of 61 pulses represents the maximum positive signal; the presence of 3 pulses represents the maximum negative signal. The bias of 32,000 pulses per second is removed by the program after sampling the input register. To prevent overflow, the registers must be sampled at 20 msec intervals.

The nominal pulse weight of the incremental angular information is $2^{-16}$ radians/puls The nominal pulse weight of the velocity increments is (0.1/32) feet/second/pulse.

Downlink Telemetry Register

The Downlink Telemetry Register is an 18 bit shift register which receives data from the PGNS in serial form. The register retains the first 18 bits of a 40 bit downlink telemetry word which is shifted, under external control, at a rate of 51.2 kpps.

The data is transmitted by the PGNS at a basic rate of 50 words per second, every fifth word being an identification (ID) word. A stop pulse discrete is received from the PGNS to indicate the end of a transmitted word. To achieve proper synchronization with the data transmissions, the Stop Pulse Discrete is sampled twice each 20 msec interval with a minimum separation between samplings of 1 msec. When the program detects the Stop Pulse Discrete, it inputs the register. When the Downlink Telemetry Register is read with an input instruction, the 18 bits of the register are loaded into the Accumulator and the Stop Pulse Discrete is automatically reset.

Input of data through the Downlink Telemetry Register is initiated by a command via the DEDA. Upon receiving the command, the program searches for the ID word which signals the start of the block of data. Upon finding the correct ID word, the program reads in and stores the block of data, skipping over every fifth word which is an ID word. The number of words in the block of data is predetermined by system requirements. The downlink data words consist of a word order bit, a sign bit, 14 magnitude bits, and two additional bits.

The PGNS downlink word formats are shown in Fig. 2.

| Bit No. | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ID Word | 0 | 0 | 0 | 0 | 0 | 0 | X | X | X | X | X | X | X | X | X | X | Y | Y |
| Data Word | 1 | S | X | X | X | X | X | X | X | X | X | X | X | X | X | X | Y | Y |

X = 0 or 1

Y = Not Applicable

S = Sign Bit (0 or 1)

FIG. 2 PGNS DOWNLINK WORD FORMAT

Communication With Ground Support Equipment

The Downlink Telemetry Register is also used to transfer a block of computer words from the Ground Support Equipment (GSE) to the erasable portion of the AEA Memory. The number of 18 bit words in the block is determined by the GSE. Initiation of the information transfer is also controlled by the GSE.

The starting address, from which this block of information is to be stored, is specified by the first word which is transmitted to the computer. The format of the word, expressed in octal, is as follows:

$$77X_1X_2X_3X_4$$

Bits 0 thru 5 are an identification code, and bits 6 thru 17 specify the starting address.

Two input discretes (GSE 1 and GSE 2) and one output discrete (GSE 5) are used to control the transfer of information. GSE 1 is sampled periodically to determine if the GSE is requesting a loading operation. If the discrete is present, the first word is input from the Downlink Telemetry Register, and the starting address is extracted. The program then sets GSE 5 to acknowledge receipt of the word, and waits for GSE 1 to be reset. When GSE 1 is reset, the program resets GSE 5 and waits for an additional setting of GSE 1 to indicate that the next word has been transmitted to the Downlink Telemetry Register. If both GSE 1 and GSE 2 are set by the GSE, the program interprets he associated word as the last word of the block.

DATA Entry and Display Assembly (DEDA) Register

The DEDA Register is a four bit input and output register which communicates serially with the DEDA. Four bits of data are shifted in or out of the register by external shift pulses at the rate of 64 kpps. A one in the serial data is represented by the presence of a pulse and a zero by the absence of a pulse. Four bits of information are received serially from the DEDA by setting the DEDA-Shift-In Discrete Output. Four bits of information are transmitted serially to the DEDA by setting the DEDA-Shift-Out Discrete Output. Both of these discretes are reset automatically. A minimum period of 80 μsec must be allowed between settings of these discretes. Bits 1 thru 4 of the DEDA Register are transferred to bits 1 thru 4 of the Accumulator by addressing the DEDA Register with an input instruction. Addressing the DEDA Register with an output instruction transfers bits 14 thru 17 of the Accumulator to bits 1 thru 4 of the DEDA Register.

Information transfers to and from the DEDA Register are controlled by the Astronaut via the DEDA. The DEDA provides a means of transmitting mode commands, instructions, and data to the AEA, and read-out of data stored in the AEA Memory. 512 memory locations in the AEA may be addressed by the DEDA for read-out or storage of data. The DEDA keyboard consists of ten decimal digit pushbuttons, two sign pushbuttons,, and four control pushbuttons. The displays consists of a three digit address display, a sign plus five digit data display, and an operator error display. The four control pushbuttons, CLEAR, ENTER, READOUT, and HOLD, apply discretes of the same name to the AEA.

DEDA operation is initiated by depression of the CLEAR pushbutton. When the program determines that the Clear Discrete has been set, it prepares for a new operation and then checks for the presence of the Readout or Enter Discrete.

Data Readout

To initiate readout of a computer memory location, a three digit octal address is entered followed by depression of the READOUT pushbutton. Upon entry of each digit, it appears on the address display. When the program determines that the Readout Discrete has been set, it inputs, in four bit serial groups, the 9 bits of address information. The program then transmits to the DEDA, in four bit serial groups, the contents of the addressed memory location along with the address information. This information appears on the DEDA display. The scaling of the transmitted information is determined by its address as is its data format, which is either octal or binary code decimal. The contents of the specified location are transmitted at half second intervals unless the Hold discrete is received from the DEDA. If the Hold Discrete is received, the program stops transmitting data. Transmission is resumed from the same address if the Readout Discrete is again received. Data transmission is terminated following receipt of the Clear Discrete.

Data Entry

To enter data, a three digit address, a sign, and five digits of data are entered via the keyboard, followed by depression of the Enter pushbutton. When the program detects the Enter Discrete, it inputs, in four bit serial groups, 36 bits of address and data information. Depending on the address specified, the entering information may be a mode command, an instruction, or input data. Its address also specifies the scali of the data and whether it is to be treated as octal or binary coded decimal informatio

Operator Errors

If an operator error exists at the time the ENTER or READOUT pushbutton is depress subsequent four bit data transmissions from the DEDA will contain all ones. In normal operation no four bit transmission will contain all ones.

Data Format

Data transmissions in either direction are in four bit groups beginning at the mos significant end of the 36 bit word shown in Figure 3. Each 4 bit group is transmitted serially, most significant bit first.

| 1 | | 12 |
|---|---|---|
| 0 X X X | 0 X X X | 0 X X X |
| | ADDRESS | |

| 13 16 | 17 | | | | 36 |
|---|---|---|---|---|---|
| 0 0 0 X | * X X X | * X X X | * X X X | * X X X | * X X X |
| SIGN | | | DATA | | |

0 - Zero

X - One or Zero

* - One or Zero if format is BCD, Zero if format is octal

FIG. 3 36 BIT DEDA WORD

### Discrete Inputs

The discrete inputs are grouped into two words, Discrete Word One, which represents eight input discretes, and Word Two, which represents seven input discretes. All but the Stop Pulse discretes are buffered switch closures to computer ground. The Stop Pulse input lines set flip-flops in the computer. When addressed by an input instruction, the specified discrete word is set into the accumulator. The individual discretes occupy the bit positions shown in Table D of Appendix B, a zero representing the true state of the discrete. Unused bit positions of the Accumulator are set to zero.

Most of the individual discrete inputs are associated with data transfers via a particular register. The following list shows this association:

| | | |
|---|---|---|
| Downlink Telemetry Register | - | Downlink Telemetry Stop |
| | - | GSE 1 |
| | - | GSE 2 |
| Output Telemetry Register | - | Output Telemetry Stop |
| | - | GSE 3 |
| DEDA Register | - | DEDA Clear |
| | - | DEDA Hold |
| | - | DEDA Enter |
| | - | DEDA Readout |

Information on the use of these discretes may be found under the discussions of the associated registers.

Six discrete inputs are interrogated by the program to determine the system status. They are: Followup, Automatic, Abort, Abort Stage, Descent Engine On, and Ascent Engine On. The first four of these are Astronaut controls. The Descent Engine On Discrete and the Ascent Engine On Discrete are applied to the computer when the Descent Engine or the Ascent Engine, respectively, are firing.

## 3.2 OUTPUT REGISTERS

The AEA contains 13 output registers. They are the 10 static D/A converter registers used for displays and attitude control, the 14 bit shift register for controlling the altitude and altitude-rate displays, the 24 bit shift register for output telemetry, and the 4 bit shift register for communication with the Data Entry and Display Assembly. In addition, there are 11 discrete outputs. Each output register and discrete is discussed separately in the following paragraphs with emphasis on its use within the Abort Guidance System.

### Total Attitude Display Registers

The six Total Attitude Display Registers are 10 bit static registers which control D/A converters. They are used to output the sines and cosines of the Euler angles $\alpha$, $\beta$, and $\gamma$, for control of the Total Attitude Display. When addressed with an output instruction, the specific register is loaded from bits 0 thru 9 of the Accumulator. The computed data must be limited before outputting, and has the following characteristics:

a) Output Rate - 25 times per second.

b) Range - 9 bits plus sign.

c) Scale Factor - most significant bit equals 1/2.

d) Format - Sign plus magnitude with a positive sign represented by a zero.

Figure H in Appendix B shows the order of rotation of the Euler Angles $\alpha$, $\beta$, and $\gamma$, which relate the vehicle axes to the inertial axes with the gimbal sequence of the attitude display indicator.

### Attitude Error Registers

The three Attitude Error Registers are 10 bit static registers which control D/A converters. They are used to output the rotational attitude errors about the three body axes for control of the vehicle and for display. When addressed with an output instruction, the specific register is loaded from bits 0 thru 9 of the Accumulator. The computed data must be limited before outputting and has the following characteristics:

a) Output Rate - 25 times per second.

b) Computation Delay - Maximum of 10 msec between sampling of gyro inputs and the subsequent output of the attitude errors.

c) Range - 9 bits plus sign.

d) Scale Factor - Least significant bit equals 0.5113269 x $10^{-3}$ radians.

e) Format - Sign magnitude with a positive sign represented by a zero.

Lateral Velocity Register

The Lateral Velocity Register is a 9 bit static register which controls a D/A converter. It is used to output the lateral velocity, that velocity along the Y body axis, for the Navigation Displays. When addressed with an output instruction, the register is loaded from bits 0 thru 8 of the Accumulator. The computed data must be limited before outputting, and has the following characteristics:

a) Minimum Output Rate - 5 times per second.

b) Range - 8 bits plus sign.

c) Scale Factor - Most significant bit equals 100 ft/sec.

d) Format - Sign magnitude with a positive sign represented by a zero.

Altitude - Altitude Rate Register

The Altitude - Altitude Rate Register is a 15 bit output shift register which provides both serial data and shift pulses to the external equipment. The register is used to output both altitude and altitude rate to the Navigation Displays and, therefor has two output data lines and two output shift pulse lines. A one in the output data is represented by a pulse and a zero by the absence of a pulse. Fifteen bits of data are output, most significant bit first, with the shift pulses, at a rate of 64 kpps. Data output from this register is initiated by loading it from bits 0 thru 14 of the Accumulator with an output instruction. Before loading the register, the desired output lines must be selected with discrete outputs. If output to the Altitude Display is desired, the Altitude Rate Discrete must be reset and the Altitude Discrete set, prior to loading the register. If output to the Altitude Rate display is desired, the discrete output settings must be reversed. If both discretes are set, output on both sets of lines will occur. Successive outputs to the register must be spaced by a minimum of 270 μsec to allow the previous data to be shifted out.

The computed data must be limited before it is output and has the following characteristics:

Altitude

a) Minimum Output Rate - 5 times per second.

b) Range - 15 bits.

c) Scale Factor - Least significant bit equals 2.34 feet.

d) Format - Magnitude

Altitude Rate

a) Minimum Output Rate - 5 times per second.

b) Range - 14 bits plus sign.

c) Scale Factor - Least significant bit equals 0.5 ft/sec.

d) Format - Sign magnitude with a positive sign represented by a one.

Output Telemetry Register

The Output Telemetry Register is a 24 bit shift register which is used to output serial telemetry data. A one in the output data is represented by a pulse and a zero by the absence of a pulse. The 24 bits of data are shifted from the register, most significant bit first, by externally supplied shift pulses at a rate of 51.2 kpps. As the register is shifted, zeros are shifted into the least significant bit of the register. When the external equipment has supplied 24 shift pulses it sends a stop pulse which sets the Output Telemetry Stop Discrete. The 24 shift pulses, followed by the stop pulse, are sent to the computer at 20 msec intervals.

The Output Telemetry Register is loaded in two operations by execution of two different output instructions. An output instruction with 6200 in the address field resets the register and the Stop Discrete, and loads bits 0 thru 17 of the register from the Accumulator. An output instruction with 6100 in the address field loads bits 6 thru 23 of the register with the logical OR of the Accumulator and bits 6 thru 23 of the register.

The Output Telemetry Stop Discrete indicates that the previous word has been shifted out of the Output Telemetry Register. To assure proper synchronization with the external equipment, it is sampled twice in a 20 msec period with a minimum interval between the two samplings of 1 msec. When the program detects the presence of this discrete, it loads the Output Telemetry Register with the next word to be output, thereby resetting the discrete.

GSE Communication

The Output Telemetry Register, in conjunction with discrete input GSE 3, and discrete outputs GSE 4 and GSE 6, can be used to communicate with the Ground Support Equipment (GSE). When the GSE receives discrete output GSE 4, it unloads the Output Telemetry Register by transmitting 18 shift pulses to the AEA, thereby receiving the contents of the 18 most significant bits of the Output Telemetry Register. The GSE acknowledges receipt of the information by applying discrete input GSE 3 to the AEA. When the program detects this discrete, it resets discrete output GSE 4 and waits for discrete input GSE 3 to be reset. When discrete input GSE 3 is reset, the program loads the Output Telemetry Register with the next word and then sets discrete output GSE 4. The program indicates that the current word is the last word of the block by setting discrete output GSE 6 in coincidence with the usual setting of discrete output GSE 4.

### DEDA Register

A discussion of this register will be found in the section describing input registers.

### Discrete Outputs

The AEA contains 11 discrete output flip-flops. When addressed by an output instruction, the individual discrete is set or reset depending upon the address. The two DEDA discrete outputs cannot be reset by execution of an output instruction, but are automatically reset. Most of the individual discrete outputs are associated with data transfers via a particular register. The following list shows this associati

| | |
|---|---|
| Altitude - Altitude Rate Register | - Altitude |
| | - Altitude Rate |
| DEDA Register | - DEDA Shift In |
| | - DEDA Shift Out |
| Output Telemetry Register | - GSE 4 |
| | - GSE 6 |
| Downlink Telemetry Register | - GSE 5 |

Information on the use of these discretes may be found under the discussions of the associated registers.

The Test Mode Failure Discrete Output is automatically set if a Delay instruction is not being executed when the computer issues the 20 msec timing pulse. It is also used to indicate the detection of a fault by the self test program, and can be set or reset by addressing it with an output instruction.

The Ascent Engine On and Descent Engine On discrete outputs control the main engines when the Abort Guidance System is in control of the vehicle. An Ascent Engine On Command is issued by setting the Ascent Engine On Discrete and resetting the Descent Engine On Discrete. A Descent Engine On Command is issued by reversing the discrete settings.

### Carry Inhibit Discrete Output

The Ripple Carry Inhibit Discrete Output is used by self test programs to provide a more complete check of the computer adder. When set, it inhibits the normal path of carry propagation, thereby providing a method of testing the carry by-pass logic. This method of test is necessary because, under normal operating conditions, the carry by-pass logic is redundant, and therefore, its failure to produce a valid carry would go undetected.

The carry by-pass logic operates as follows: If either or both of the operands of each stage within the by-passed are ones, and if a carry into the least significant stage of the by-passed section is present, a carry is immediately generated out of the most significant stage of the by-passed section.

The Carry Inhibit Discrete, when set, inhibits carries out of the most significant stage of a by-passed section only when they are caused by a carry into this stage from the next most significant stage. For example: If both operands of the next most significant stage of a by-passed section are ones, and if one of the operands of the most significant stage is a one, a carry out of the most significant stage is not produced if the Carry Inhibit Discrete is set, unless the required conditions are present for a carry to be generated by the by-pass logic. However, if both operands of the most significant stage of a by-passed section are ones, a carry out of this stage is generated regardless of the state of the Carry Inhibit Discrete.

Figure 4 shows the carry by-pass system.

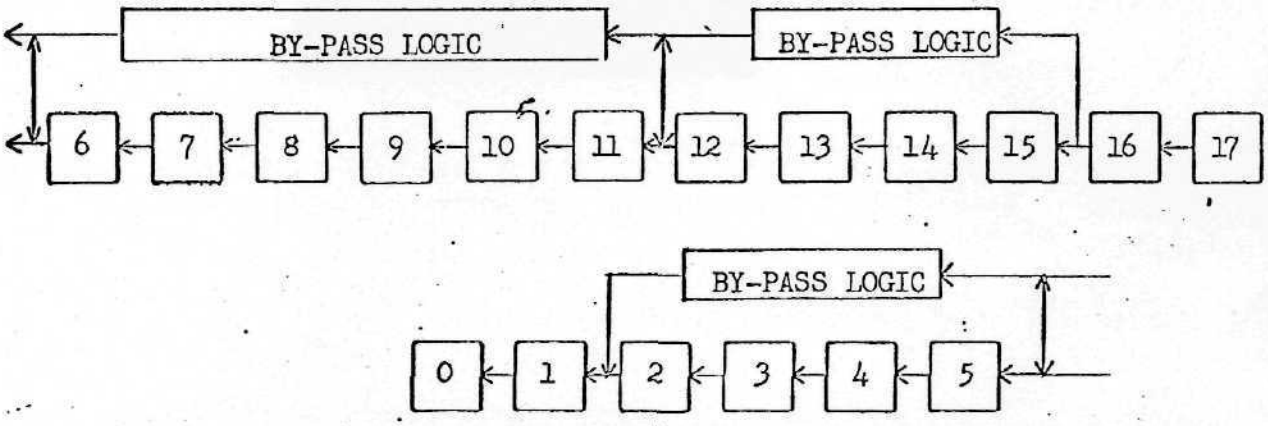


FIG. 4 CARRY BY-PASS SYSTEM

To test the ability of a section of by-pass logic to generate a carry, the following conditions are necessary:

1) Carry Inhibit Discrete Set

2) A carry out of the stage preceding least significant stage of by-passed

3) One, but not both, of the operands of the most significant stage of the by-passed section must be a one.

## 4.0 START-UP AND SHUT DOWN PROCEDURES

This section deals with automatic and programmed start-up and shut down procedur

### 4.1 AUTOMATIC START-UP SEQUENCE

When the computer is switched to the OPERATE mode, power is applied in a predetermined sequence which prevents loss of data from temporary storage. When the voltages are at their proper levels, initial conditions are established such that the computer obtains its first instruction from memory location $6000_8$. If this memory location does not contain an unconditional transfer (TRA, TSQ, or DLY), the next instruction will be taken from memory location $0001_8$. The automatic initialization sequence also resets the overflow indicator and the Engine On Discrete Output.

### 4.2 PROGRAMMED INITIALIZATION

When power is applied to the AEA, most of the registers and discretes in the Input/Output Section assume a random state. To prevent random outputs from the computer, and for proper execution of programs, it is necessary to initialize certair of these registers and discretes.

Before any instruction which involves use of the adder can be executed, the Carry Inhibit Discrete must be reset. All of the discrete outputs and the Carry Inhibit Discrete may be reset by executing two output instructions, one with 3050 in the address field, and one with 7077 in the address field.

The Downlink Telemetry Register and the DEDA Register must be initialized by inputting each of these registers.

The permanent memory must be initialized by accessing each of its locations at least one time before executing programs from it. Under normal conditions, initialization of the permanent memory is necessary only the first time it is operated. However, for maximum protection against loss of information, this initialization procedure should be performed as part of the start-up sequence.

### 4.3 AUTOMATIC SHUT DOWN SEQUENCE

When the power supply input voltage drops below the required level, the compute: shuts down operation systematically and restarts operation, as described in Section 4.1, when the required voltage is restored. To prevent loss of information from temporary storage when shutting down, the computer does not halt operation when a memory cycle is in process.

## ABBREVIATIONS AND SYMBOLS USED IN THE TEXT

### ABBREVIATIONS

| | | |
|---|---|---|
| AEA | - | Abort Electronic Assembly |
| AGS | - | Abort Guidance System |
| ASA | - | Abort Sensor Assembly |
| D/A | - | Digital to Analog |
| DEDA | - | Data Entry and Display Assembly |
| GSE | - | Ground Support Equipment |
| ID | - | Identification |
| IMU | - | Inertial Measurements Unit |
| LEM | - | Lunar Excursion Module |
| OP CODE | - | Operation Code |
| PGNS | - | Primary Guidance and Navigation System |

### SYMBOLS

Y - Represents a memory location or the address field of an instruction.

N - Represents the address field of an instruction.

A0, A1, ... A17 - Individual bit positions in the Accumulator.

Q0, Q1, ... Q17 - Individual bit positions in the Q Register.

Y0, Y1, ... Y17 - Individual bit positions of a memory location.

# APPENDIX A

# LEM ASSEMBLY PROGRAM

# APPENDIX A

This appendix provides all of the information necessary for the use of the LEM Assembly Program (LEMAP), a program which assembles symbolically coded programs into binary information for filling the AEA memory.

## CONTENTS

## I. INTRODUCTION

The LEM Assembly Program (LEMAP) consists of two sub-parts: the assembler and a card handling routine.

The card handling routine used by LEMAP is SCUFF ("Symbolic Compression of Unspecified Free Fields"). SCUFF provides the capability of compressing an original symbolic deck, of reading a compressed deck, and of reading a compressed deck and introducing symbolic modifications to it. Four general types of runs are possible:

1. Compression of a symbolic deck.
2. Modification and loading of a compressed deck.
3. The punching of a new compressed deck from a modified deck.
4. Punching of a symbolic deck from a compressed deck.

On each run which includes punching, LEMAP will provide a listing of the new deck, with a sequence number associated with each card. LEMAP may also, by use of a LIST card, provide a listing of an unmodified compressed deck in a non-punch run.

The compressed deck contains, in binary form, the BCI equivalent of the information contained on the original symbolic cards. The compressed deck is thus merely a more compact and manageable means of handling totally symbolic information. It provides essentially the same degree of flexibility of modification and rearrangement as a symbolic deck.

The Assembler requires two passes. On the first pass the following thin (not necessarily in the order given) are done:

1. Whenever an ORG pseudo-operation is encountered, the location count is set to the value appearing in the variable field (for an explanation of the location counter, see Section D in Chapter 2; also for more detailed informati about pseudo-operations, see Section B in Chapter 3). If the location counter is not set to the desired value by the programmer by using an ORG as the first instruction of his program, the location counter will start with the value 00C i.e., the first instruction of the program which requires a location in the L memory will be assigned to location 0000.

2. Whenever a LEM machine operation is encountered, the value of the location counter is increased by one, thus assigning an absolute location to LEM machine operation.

3. Whenever a data generating pseudo-operation, i.e., OCT or DEC, is encountered the location is increased by the number of words generated by the OCT or DEC pseudo-operation, thus assigning an absolute location for each data word generated by the DEC and OCT pseudo-operations.

4. Whenever a block storage allocating pseudo-operation, i.e., BSS or BES, is encountered the location counter is increased by the number appearing in the variable field, thus assigning an absolute location to this block of storage.

5. Whenever a location symbol is encountered, regardless of whether it is associated with a LEM machine operation or a pseudo-operation, it is stored away in the dictionary together with the current value of the location counter except for the pseudo-operations EQU, SYN, BES and DEFINE (for a detailed explanation of what is done for these pseudo-operations see Section B in Chapter 3).

6. A tape (logical unit SYSBU4) containing the BCI image of all the symbolic cards is written. This tape is then read during the second pass of the Assembler.

7. If requested, a compressed deck is punched.

On the second pass, the following things (not necessarily in the order given) are done:

1. All LEM machine operations are converted to the bit patterns used by the machine, i.e., the symbolic operation codes are replaced by the actual bit patterns used by the machine. The variable field of these operations is evaluated and converted to the bit patterns used by the machine. The assembled LEM word is then stored in the location set aside for it in the first pass.

2. The variable fields of all data generating pseudo-operations are converted to the actual bit patterns used by the machine and these data words are stored in the locations set aside for them in the first pass.

3. If a listing is requested, the Assembler lists the LEM location and the binary bit patterns generated together with the corresponding symbolic input cards. The LEM locations and generated bit patterns are printed as octal characters. For more information the reader is advised to examine the appended listing.

4. If requested, by a REFER card, a symbol reference table is printed. This symbol reference table gives the symbol name and its sequence number and

lists all the sequence numbers and mnemonics of instructions that refer to a given symbol. In addition, a "$" will be printed beside the mnemonic if the given symbol is referenced in a relative expression and an "*" will be printed if the tag field (index register field) is non-blank.

5. If requested, by a CPLPA or PA card, an absolute deck of the LEM program, which is to be used by the LEM Interpretive Computer Simulation Program is punched.

6. If requested, by a GO card, the guidance program will be left in 7094 core and the ICS will be read in.

## II. THE LEMAP LANGUAGE

### A. Basic Definitions

The LEMAP (LEM Assembly Program) language is made up of operation codes, symbols and integers which are arranged into an ordered series to form the symbolic instructions which the Assembler can recognize.

An Operation Code, which appears in the operation field of a symbolic instruction, is any one of the fixed set of mnemonic alphabetic codes which make up the vocabulary of the Assembler. These include all the basic LEM Compute (Abort Electronic Assembly) operations as well as the LEMAP pseudo-operations which are described in Chapter 3.

A Symbol is a name invented by the programmer which may appear either in the location field or the variable field of a symbolic instruction. In fact, placing a symbol in the location field of an instruction is the only way of defining a symbol. (See section on symbols in this chapter for a further discussion of symbols.)

An Integer is a string of numeric characters which will be interpretec as decimal (base 10) or octal (base 8), depending on the number of digits contair in the integer. An integer consisting of 3 or less digits will be interpreted as decimal and an integer consisting of 4 or more digits will be interpreted as octal with the following exceptions: (1) all integers in the variable field of the OCT, DEFINE and ORG pseudo-operations will always be interpreted as octal, and (2) all integers in the variable field of the pseudo-operation DEC will be interpreted as decimal.

B. Symbolic Card Format

A symbolic instruction consists of four major fields: location field, operation field, variable field, and comment field.

A program written in the LEMAP language is a succession of symbolic instructions punched one per card, in the following Symbolic Card Format.

The following are selectively punched or left blank.

1. Location Field (fixed length) occupies columns 1-6. Contains a location symbol or may be left blank. Column 7 is always blank.

2. Operation Field (fixed length) begins in column 8 and ends in or before column 14. It contains a mnemonic operation code.

3. The Variable Field (variable length). For LEM computer operations the variable field in its most general form consists of two subfields separated by a comma: address, tag. (Note that this is the reverse of the internal machin order, which is tag, address.)

In the following example:

| LOCATION | OP | VARIABLE FIELD |
|---|---|---|
| | TIX | ALPHA, 1 |

the operation TIX has an address of ALPHA and a tag of 1. (Note that 1 or 0 is the only legal tag because the LEM Computer has only one index register.)

One or more blank columns separate the variable field from the operation field. The variable field may begin in or following column 12, but in no case later than column 16, and it must end prior to column 64. The variable field cannot contain any blanks, since a blank signals its end. Hence, there may not be any blanks between subfields or within any subfield of the variable field.

Any valid expression may appear in any subfield of the variable field and will be evaluated according to the rules of expressions, as stated in Section E, except that only the rightmost bit of the tag subfield will be used, i.e., the tag subfield expression will be reduced modulo 2.

Certain pseudo-operations (non-LEM machine operations) in the Assembler which will be discussed in Chapter 3 require variable fields not of the type discussed in this section. The rules for specifying the variable field depend on the given pseudo-operation.

4. The Comment Field (variable length) begins with the first non-blank character following the blank character which terminates the variable field. All punching to the right of such a blank is considered to be a comment and has no effect on the processing of the source program. The comments are of course retained within the compressed deck.

## C. Symbols

A symbol (also referred to by the terms "location symbol" and "symbolic address") will consist of a string of one to six non-blank characters, at least one of which is non-numeric, and none of which is among the following set of seven characters:

| | |
|---|---|
| + (plus) | $ (dollar sign) |
| - (minus) | = (equal sign) |
| / (slash) | , (comma) |
| * (asterisk (see special use of "*" as a symbol in this chapter)) | |

For example,

A, A1, 12345X, (1), SQF2, and 12. are all legal symbols.

Every symbol used in the program must be defined exactly once. An error will be indicated by the Assembler if any symbol is defined more or less than once. It is generally desirable to use a location symbol for an instruction only if this symbol is needed to refer to that instruction from elsewhere in the program. The reason for this is that the Assembler, in processing the source program, keeps in core storage a "dictionary" (or symbol table) of location symbols, and there is a limit though reasonably large, as to the amount of a core storage which can be allotted for this purpose (2048 symbols at present).

## D. The Location Counter

Each entry in the dictionary compiled by the Assembly contains a location symbol and the "value" of the location symbol. This value is an absolute binary number denoting an actual machine cell in the LEM computer. In order that the Assembler may assign the proper value to each location symbol used in the source program, it uses a device called the "location counter". This can be set initially to an arbitrary value by the source program (see "ORG" under Pseudo-Operations, Chapter 3). Each time a LEM Computer instruction is encountered, the location counter is increased by one. (Certain pseudo-operations may result in no increase or an increase of more than one. For example, "END" and "EQU" have no effect on the location counter, whereas pseudo-operations such as "BSS" and "BES" may change the value of the location counter by more than one.) Whenever a location symbol is associated with the instruction being processed, an entry is made in the dictionary, taking the current value in the location counte for the value of the symbol.

E. Relative Expressions

A relative expression is the sum or difference of two or more symbols or constants. Consider the following coding:

```
LOCATION          OP          VARIABLE FIELD

ALPHA             TRA         BETA
                  CLA         GAMMA
                  SUB.        L (1)
STGAM             STO         GAMMA
                  TMI         DELTA
   .               .            .
   .               .            .
   .               .            .
```

Suppose the programmer wishes to transfer control to the instruction "CLA GAMMA". He may then write:

```
LOCATION          OP          VARIABLE FIELD

                  TRA         ALPHA+1

or alternately
he may write:
                  TRA         STGAM-2
```

Thus, he may refer to an unnamed instruction, "N", by using the symbol of an instruction somewhere in the vicinity of "N" and adding or subtracting an integer. The combination "ALPHA+1" or "STGAM-2" is called a relative expression.

### F. The Use of "*" as a Symbol

The asterisk may be used as a symbol. When employed in this way, it is regarded as a symbol whose value is the current value of the location counter. For example, the instruction,

| LOCATION | OP | VARIABLE FIELD |
|---|---|---|
| ALPHA | TRA | *+2 |
| is equivalent to | | |
| ALPHA | TRA | ALPHA+2 |

and represents a transfer to the second location following the TRA instruction. With this application of "*" in a program, one can avoid introducing superfluous symbols.

## III. DESCRIPTION OF LEMAP OPERATIONS

### A. Classification of LEMAP Operations

The LEMAP language includes all LEM (Abort Electronic Assembly) machine operations, and a group of pseudo-operations (non-machine operations).

A LEM machine operation generates an 18-bit binary machine word. The rules for specifying the location field and the variable field of a machine operation have already been discussed in Chapter II under Symbolic Card Format.

Unlike machine operations, some pseudo-operations may generate more than one LEM machine word and others may generate no words at all. All of the pseudo-operations of the Assembler will be described in detail in the remainder of this Chapter.

### B. Pseudo-Operations

Location Counter Control

The following three pseudo-operations, ORG, BSS and BES are used principally to control the contents of the location counter.

(1) ORG ("Origin")

If the programmer wishes the machine location of the first word in his program to be, say, $(1000)_8$, he may simply start his source program with the pseudo-operation:

| LOCATION | OP | VARIABLE FIELD |
|---|---|---|
| | ORG | 1000 |

No word is generated in the object program by this instruct but the effect is to direct the assembler to set the location counter to t value $(1000)_8$. If the instruction immediately following "ORG" is "ALPHA C BETA", then the symbol "ALPHA" will receive the value $(1000)_8$ in the dictic of location symbols, and the binary machine word which eventually results f "CLA BETA" will be ear-marked for location $(1000)_8$.

Note that the variable field of ORG consists of a single subfield and if more than one subfield is written, only the first will be used. The remaining subfields will be ignored and flagged as an error on th output listing.

If there is a symbol in the location field of an ORG instructi it will be assigned the value in the variable field. Notice that, in the exa

| LOCATION | OP | VARIABLE FIELD |
|---|---|---|
| ALPHA | ORG | 1000 |
| | CLA | BETA |

The same effect would have been obtained by writing:

| LOCATION | OP | VARIABLE FIELD |
|---|---|---|
| | ORG | 1000 |
| ALPHA | CLA | BETA |

On the other hand, if the programmer were to write:

| LOCATION | OP | VARIABLE FIELD |
|---|---|---|
| ALPHA | ORG | 1000 |
| GAMMA | CLA | BETA |

then "ALPHA" and "GAMMA" would both be entered into the dictionary each with the associated value of $(1000)_8$. (Another way of achieving such a "synonym" effect will be seen later in the pseudo-operations "SYN" and "EQU").

ORG instructions may appear anywhere in a program, not necessarily in the beginning. In fact, the Assembler does not require the presence of an ORG at the beginning, or anywhere within the source program.

If the programmer decides not to set the initial value of the location counter with ORG, the Assembler will assume the intent is to be begin the program at location $(0000)_8$.

Restriction: The variable field of an ORG instruction will always be interpreted as an octal integer and therefore should not contain any symbols or decimal integers.

(2) BSS ("Block Started by Symbol")

The BSS pseudo-operation is used to reserve a block of one or more words of memory within an object program for such purposes as "erasable storage". The length of the block reserved is given in the variable field, and if there is a symbol in the location field, this symbol refers to the first cell of the block. Consider the following examples:

| LOCATION | OP | VARIABLE FIELD |
|---|---|---|
| | BSS | 50 |

When the Assembler encounters this instruction, it will increase the location counter by $(50)_{10}$.

| LOCATION | OP | VARIABLE FIELD |
|---|---|---|
| ALPHA | OCT | 740 |
| BETA | BSS | 4 |
| GAMMA | OCT | 1677 |

Suppose the symbol ALPHA has been assigned to location 1001. Then the symbol BETA will be assigned to location 1002, and the symbol GAMMA will be assigned to location 1006, leaving four locations (1002, 1003, 1004 and 1005) for the block BETA.

(3) BES ("Block Ended by Symbol")

This pseudo-operation has exactly the same properties as "BSS" except that when it is used with a location symbol, that symbol is associated with the first word following the reserved block. In the example,

| LOCATION | OP | VARIABLE FIELD |
|---|---|---|
| ALPHA | BES | 50 |
| | CLA | BETA |

the symbol "ALPHA" gets associated with the instruction "CLA BETA". The programmer could have equivalently written:

| LOCATION | OP | VARIABLE FIELD |
|---|---|---|
|  | BES | 50 |
| ALPHA | CLA | BETA |

On the other hand, if the programmer writes:

| LOCATION | OP | VARIABLE FIELD |
|---|---|---|
| ALPHA | BES | 50 |
| GAMMA | CLA | BETA |

then "ALPHA" and "GAMMA" would both be entered into the dictionary, each with the associated value of the location counter at the time that the assembler processes the CLA instruction.

The effect of "ALPHA BES 50" is, then, to increase the location counter by $(50)_{10}$, and then to enter ALPHA into the dictionary with the resulting value in the location counter.

The examples given illustrate the use of BES with a symbol in the location field. If the location field had been left blank, BES would be exactly the same as BSS.

Pseudo-Operations for Relating Symbols

SYN, EQU and DEFINE are three pseudo-operations which serve the purely logical function of relating two or more symbols with respect to their value, or of simply assigning a value to a symbol.

(4) SYN, EQU ("Synonymous", "Equals")

In the LEMAP language, SYN and EQU are identical and may be used interchangeably. Hence, this discussion applies to both pseudo-operations.

We have seen that symbols may be defined by appearing in the location field of an instruction, and the symbol is assigned the current value of the location counter. Unlike ORG, BSS and BES, the pseudo-operations SYN and EQU do not affect the value of the location counter, but define a location symbol as being equivalent to the value of the expression in the variable field. Moreover, a SYN (or EQU) instruction is meaningless if it does not have a location symbol. The variable field of a SYN or EQU instruction should contain only one subfield, and this may be a symbol, an integer or an arithmetic expression.

Consider the following examples:

```
LOCATION        OP          VARIABLE FIELD

START           CLA         ALPHA
                SUB         BETA
                TMI         XI
                 .           .
                 .           .
                 .           .
NEG             AXT         7,1
XI              SYN         NEG
                CLA         GAMMA
                 .           .
                 .           .
                 .           .
                      (etc.)
```

The instruction "XI SYN NEG" states that the value of the symbol "NEG" should be assigned to the symbol "XI". The SYN instruction could ha been placed anywhere in the program.

```
LOCATION        OP          VARIABLE FIELD

ALPHA           SYN         BETA-GAMMA+17
```

The effect of the above instruction is to enter "ALPHA" in the dictionary with the value of the arithmetic expression in the variable field.

```
LOCATION        OP          VARIABLE FIELD

                 .           .
                 .           .
                 .           .
                LLS         SHIFT
SHIFT           SYN         35
                 .           .
                 .           .
                 .           .
```

In this example, the variable field expression is compl numeric, and gives the value $(35)_{10}$ to the symbol "SHIFT".

(5) DEFINE ("Octal Equals")

The pseudo-operation DEFINE is similar to SYN and EQU in the sense that the associated location symbol is defined as being equivalent the expression in the variable field, except that the variable field is alwa interpreted as an octal integer.

DEFINE is a useful pseudo-operation for defining symbols which are most meaningful when expressed in octal, such as an address in LEM

Data Generating Pseudo-Operations

The LEMAP language provides two pseudo-operations (DEC, OCT) which may be used to introduce words of data (often referred to as "constants" into a program. These data words will be converted into binary by the Assembl and are considered as much a part of the object program as the binary instruct: Negative data will be converted to 2's complement.

(6) DEC ("Decimal Data")

The DEC pseudo-operation is used to introduce into a progra words of data expressed as decimal numbers. A symbol or blank may appear in th location field. One or more subfields, each containing a decimal number, appea the variable field.

If there is a symbol in the location field, this symbol is entered into the dictionary with the current value of the location counter so t the first decimal number in the variable field may be referred to by this symbol

The subfields of the variable field are separated by commas. The number of subfields permissible is limited only by the restrictions that the last subfield must be terminated by a blank, and that the entire instruction must fit on one symbolic card, i.e., the variable field must end in or before card column 63.

Each subfield is converted to a binary word. These words are assigned to successively higher storage locations as the variable field is processed from left to right. Consecutive commas in the variable field, indicatin a null subfield, cause the number zero to be generated; as does a comma followed b a blank. Hence, the number of words generated is always one more than the number of commas in the variable field.

Every decimal number must be represented by a string of characters from the following set of 15 characters: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 (numeric characters), + (plus sign), - (minus sign), . (decimal point), E (exponent), and B (binary scaling).

Two types of decimal numbers can be used:

a) Integers (whole numbers)

b) Fixed point numbers

The sign of any decimal number is always specified by the first character, either "+" or "-". The "+" sign may be omitted, i.e., if no initial "+" or "-" appears in the string, the sign is assumed to be "+".

A decimal integer is composed of a string of exclusively numeric characters, possibly preceded by a plus sign or a minus sign, which is converted to a 17-bit binary number with sign (negative numbers will be converted to 2's complement). Thus, the decimal integer

-31

would be converted to the binary number whose octal representation is

777741

Both 0 and -0 are converted to octal number 000000. A decimal integer is distinguished from a fixed point decimal number by the fact that the letter B, the letter E and the decimal point are all absent. For all decimal integers, the position of the binary point is considered as being at the right hand end of the word, i.e., at B17.

A fixed point number has three components:

a) The principal part is written with or without a decimal point. The decimal point may appear at the beginning or end of the principal part, within the principal part, or may be omitted. If the decimal point is omitted. it is assumed to be located at the right hand end of the principal part.

b) The exponent part consists of the letter "E" followed by a signed or unsigned decimal integer. The exponent part may be absent. If present, it must follow the principal part, but may precede or follow the binary-place part.

c) The binary-place part consists of the letter "B" followed by a signed or unsigned decimal integer. The binary-place part must be present in a fixed point number, and must follow the principal part. If the number has an exponent part, then the binary-place part may precede or follow the exponent part.

A fixed point number is converted to a fixed point binary quantity which contains an understood binary point. The purpose of the binary-place part of the number is to specify the location of this understood binary point within the word. The number which follows the letter "B" specifies the number of binary places in the word to the left of the binary point (that is, the number of integral places in the word). The sign bit is not counted. Thus a binary-place part "0" specifies a 17-bit fraction. "B2" specifies 2 integral places and 15 fractional places. "B17" specifies a binary integer. "B-2" would specify a binary point located 2 places to the left of the leftmost bit of the word, that is, the word would contain the low-order 17 bits of a 19-bit binary fraction. The exponent part, if present, specifies a power of ten by which the principal part will be multiplied during conversion.

In the process of shifting the converted word to position the binary point, significant bits may be shifted past the right-hand end of the word and lost; no error will be indicated. However, if non-zero bits must be shifted past the left-hand end of the word, an error will be indicated on the output listing. Thus, the integral part of a fixed point number must be small enough to fit in the number of integral places allowed. Also, if the binary-place part is negative, the number must be an appropriately small fraction.

For example, the following fixed point numbers all specify the same configuration of bits; but not all of them specify the same location for the understood binary point:

```
22.5B5
11.25B4
1125B4E-2
1125.E-2B4
9B7E1
```

All of these fixed point numbers will be converted to the binary configuration whose octal representation is

```
264000
```

The 18-bit word size of the LEM computer cannot accommodate integers whose absolute value exceeds $2^{17}-1$. Hence, decimal numbers outside of this range should not be specified in a DEC pseudo-operation. If this restriction is violated, the number will be taken as zero and an error will be indicated on the output listing.

If the TRA instruction, in the following example, is assigned to location 1001, then the symbol DATA will be entered into the dictionary for location 1002. The five words (98,309.6B10,-5E-1B1,0,0) generated by DEC will occupy locations 1002-1006 and the symbol BETA will be assigned to location 1007.

```
LOCATION      OP        VARIABLE FIELD

ALPHA         TRA       GAMMA
DATA          DEC       98,309.6B10,-5E-1B1,,
BETA          BSS       7
```

### (7) OCT ("Octal Data")

The OCT pseudo-operation is used to introduce into a program, binary data expressed in octal form. A symbol or blanks may appear in the location field. One or more subfields, each containing a signed or unsigned octal integer, appears in the variable field.

If there is a symbol in the location field, this symbol is entered into the dictionary with the current value of the location counter, so that the first octal integer in the variable field may be referred to by this symbol.

The subfields of the variable field are separated by commas. The number of subfields permissible is limited only by the restrictions that the last subfield must be terminated by a blank, and that the entire instruction must fit on one symbolic card, i.e., the variable field must end in or before card column 63. Each subfield is converted to a binary word; these words are assigned to successively higher storage locations as the variable field is processed from left to right.

Consecutive commas in the variable field, indicating a null subfield, cause the number zero to be generated as does a comma followed by a blank. Hence, the number of words of data generated is always one more than the number of commas in the variable field.

Every octal integer must be represented by a string of characters from the following set of 10 characters: 0, 1, 2, 3, 4, 5, 6, 7 (numeric characters), + (plus sign), and - (minus sign).

The octal representation should consist of no more than 7 characters (a plus or minus sign and 6 numeric characters). Numbers preceded by a minus sign will be converted to their 2's complement. Thus, -2 and 777776 will both be converted to 777776. Both 0 and -0 are converted to 000000.

If the TRA instruction, in the following example, is assigned to location 1001, then the symbol DATA will be entered into the dictionary for location 1002; the five words (777777,0,-77,66,0) generated by OCT will occupy locations 1002-1006 and the symbol GAMMA will be assigned to location 1007.

| LOCATION | OP | VARIABLE FIELD |
|---|---|---|
| ALPHA | TRA | BETA |
| DATA | OCT | 777777,,-77,66, |
| GAMMA | BSS | 7 |

(8) END ("End")

"END" must be used as the last instruction of every source program to signal the Assembler that it has reached the end of the program.

If the location field of an "END" card contains a symbol, it will be ignored and will not be entered into the dictionary.

The variable field of an "END" card is ignored.

## IV. ERROR INDICATION

If certain error conditions are detected during the second pass of the assembler, an appropriate comment will be printed and the instruction will be flagged as an error. Whether or not a listing is requested, the card in error is printed and an error code will be printed at the extreme left of this line of print. A description of the error codes follows:

BD The variable field of data generating pseudo-operation is in error. Either too many bits are generated or the "B" field is missing in a fixed point decimal number.

| | |
|---|---|
| IADR | Illegal operand address (either negative or larger than LEM memory). |
| ILOC | Illegal LEM location (larger than LEM memory). |
| TM | Tag missing. The two instructions "AXT" and "TIX" always require a tag field. If the tag is missing, it will be inserted; however, the programmer should correct this instruction on the next run. |
| IOP | The operation field contains an illegal mnemonic. |
| U | The variable field contains an undefined symbol. |

In addition to the above error codes, the assembler will print a list of undefined symbols and a list of multiply defined symbols.

## V. DECK SET UP AND PRODUCTION PROCEDURES

LEMAP (LEM Assembly Program) allows modification of a current LEM program and the execution of an interpretive routine run in a single pass on the machine.

### A. Original Compilation Using LEMAP

(1) Input

The symbolic deck, preceded by either a CPL or a CPLPA control card. The symbolic deck may be followed by a GO card which will cause the reading in of the Interpretive Routine.

(2) Output

a) The compressed symbolic program punched in column binary form. This deck contains the entire symbolic deck in a highly compressed form.

b) A listing of the Assembly including a sequence number attached to each line in the symbolic coding.

c) If the CPLPA control card was used, a column binary absolute deck is also produced and is separated from the compressed deck by three (3) blank cards.

### B. Loading the Compressed Deck Without Modifications

(1) Input

The compressed deck, preceded by one of the following control cards; PA, PS, PSM, or ML. This control card may be followed by a REFER (or a

REFER and LIST in the case of ML). The compressed deck is followed by a blank card, and optionally a GO card.

(2) Output

a) The ML control card produces no output other than error diagnostics.

b) The PS control card results in the punching of a new compressed deck, and produces a new listing.

c) The PA control card produces a new absolute deck in addition to the PS output.

d) The PSM control card results in the punching of a symbolic deck and produces a listing.

e) The LIST control card causes a listing to be produced when used with an ML. Use with a PA, PSM, or PS has no effect.

f) The REFER control card causes a symbolic reference to be output when used with PS, PA, CPL, CPLPA, or ML and LIST.

C. Loading the Compressed Deck With Modifications

The modifications are preceded by a MOD control card and are followed by an ENDMOD control card. This modification deck immediately precedes the compressed deck. For output, see "Loading the Compressed Deck without Modificatic

(1) Modifications by Use of a Sequence Number in Card Columns 65-71

a) Replacements

An instruction punched in normal symbolic format, with the sequence number n in columns 65 through 71, will replace the line of coding bearir sequence number n.

b) Insertions

A symbolic instruction containing a non-integral sequence number in columns 65 through 71 will cause the instruction to be inserted such that the sequence numbers are kept in sort.

c) Rules for Punching Sequence Numbers

(i) All sequence numbers must be left justified in columns 65-71.

(ii) Integral sequence numbers less than 10 must be punched with a decimal point following the integer. Thus, seven must appear as 7., 7.0 or 7.00.

(iii) The last digit punched in a sequence number determines the "counting position", and if the sequence number on the following card consists of only one digit which is one greater than the counting position digit of the previous card, it is assigned a sequence number one greater than the previous card.

For example:

| | | |
|---|---|---|
| Card n | 7.2 | is given sequence number 7.2 |
| Card n+1 | 3 | is given sequence number 7.3 |
| Card n+2 | 4 | is given sequence number 7.4 |
| Card n+3 | 91 | is given sequence number 91 |
| Card n+4 | 2 | is given sequence number 92 |

In the following examples, blocks 1, 5, and 6 are due to rule (iii) above. Overlapping modifications occur in blocks 1 and 4. The blocks need not be in sort, for example; it is quite permissible for block 5 to follow block 2. Machine time saved if the modifications appear in ascending sequence is negligible.

| Block No. | Sequence No. (Punched) | Sequence No. (Assigned) |
|---|---|---|
| 1 | 5.0 | 5.0 |
| | 1 | 5.1 |
| | 2 | 5.2 |
| 2 | 73.1 | 73.1 |
| 3 | 14 | 14 |
| | 15 | 15 |
| 4 | 5.2 | 5.2 |
| | 5.3 | 5.3 |
| | 5.4 | 5.4 |
| 5 | 22.1 | 22.1 |
| | 2 | 22.2 |
| | 3 | 22.3 |
| | 4 | 22.4 |
| 6 | 65.21 | 65.21 |
| | 65.22 | 65.22 |
| | 3 | 65.23 |
| | 4 | 65.24 |
| 7 | 22.25 | 22.25 |

(2) Modifications by Use of the Pseudo-Operation ALTER

The ALTER pseudo-operation can be used to delete symbolic source program cards, insert additional symbolic source program cards, or both, depend on the form of the instruction.

There are two permissible forms for ALTER. The first is:

| LOCATION | OP | VARIABLE FIELD |
|---|---|---|
| | ALTER | $N_1$, $N_2$ |
| | | where $N_1$ and $N_2$ represent sequence numbers. |

This form indicates that the information corresponding to sequenc numbers $N_1$ through $N_2$ inclusive, is to be deleted from the program. If, in addition, symbolic cards immediately follow an ALTER of this form, they will be inserted into the program between $N_1$ - 1 and $N_2$ + 1. Since insertions are made as in an assembly, the words following $N_2$ are automatically adjusted and the numb of insertions need not be enual to the number of deletions.

The second form is:

| LOCATION | OP | VARIABLE FIELD |
|---|---|---|
| | ALTER | N |
| | | where N is a sequence numbe |

This form indicates that no deletions are to be made, and that the associated symbolic modification cards, i.e., the symbolic cards immediately following an ALTER of this form, are to be inserted between the symbolic instructi numbered N and N+1.

Restrictions:

a) In an ALTER instruction in the first form, $N_1$ must be less than or equal to $N_2$; otherwise an error will be indicated and the symbolic cards to be inserted will be ignored..

b) All cards to be inserted by an ALTER must be blank in columns 65-71. This is necessitated by the fact that each ALTER block is terminated by either another ALTER card or by a sequence number in columns 65-71.

c) A modification by an ALTER must not overlap a modification either by another ALTER or a modification of the type discussed in Section C (1) of this chapter.

Examples:

a) Suppose that it is desired to correct the instruction with the sequence number 15 in the following listing:

```
14.  ERR        CLA        N
15.             STO        ALPHA
16.             TRA        ERPRT
```

The instructions necessary to accomplish the correction are:

```
LOCATION        OP         VARIABLE FIELD

                ALTER      15, 15
                STO        BETA
```

Assuming that there are no modifications affecting the preceding instructions, after this change is made the listing will appear as:

```
14.  ERR        CLA        N
15.             STO        BETA
16.             TRA        ERPRT
```

b) Suppose that in the following listing, the instructions with sequence numbers 92 and 93 are to be deleted:

```
91.  NUMBER     EQU        24
92.  N1         EQU        12
93.  N2         EQU        0
94.  N3         DEC        1024
```

The required instruction is:

```
LOCATION        OP         VARIABLE FIELD

                ALTER      92, 93
```

Assuming that there are no modifications affecting the preceding instructions, after this change is made, the listing will appear as:

```
91.  NUMBER        EQU          24
92.  N3            DEC          1024
```

c) Suppose that it is desired to insert the instructions SUB A,1 and TMI NEG following the instruction which has the sequence number 9 in the listing below, without deleting any instructions.

```
 8.  CKN           CLA          N,1
 9.                ADD          CON
10.                STO          B,1
11.                TIX          CKN,1
```

The required instructions are:

```
LOCATION           OP           VARIABLE FIELD

                   ALTER        9
                   SUB          A,1
                   TMI          NEG
```

Assuming that there are no modifications affecting the preceding instructions, after this change is made, the listing will appear as:

```
 8.  CKN           CLA          N,1
 9.                ADD          CON
10.                SUB          A,1
11.                TMI          NEG
12.                STO          B,1
13.                TIX          CKN,1
```

Note: Of the three examples given, the 1st and 3rd could have been done in the following manner.

```
              LOCATION          OP           VARIABLE FIELD          SEQ

      Ex. a)                    STO          BETA                    15
      Ex. c)                    SUB          A,1                     9.1
                                TMI          NEG                     9.2
```

D. Control Cards

The control card operation code is punched left justified in column 8 through 15, the remainder of the card is ignored.

CPL — Results in the compilation of the symbolic deck, listing of the compilation, and punching of the compresse deck.

CPLPA — Results in the punching of an absolute column binary program deck in additon to the normal CPL output.

ML — Results in the loading of the compressed deck, and any modifications.

LIST — Produces a listing of the assembled (perhaps modified) program during an ML run. This card has no effect on CPL, CPLPA, PS, PSM, or PA runs.

PS — Produces a listing, and a new compressed deck with any modifications merged into place.

PA — Results in the punching of an absolute binary deck, in addition to the normal PS output.

PSM — Causes the punching of the (modified) program in symbolic form and produces an assembly listing.

REFER — Produces a symbolic reference table. Used only when normal assembly listing is to be produced.

MOD — This card must precede the modifications.

ENDMOD — This card must follow the last modification.

E. Notes

(1) Sequence numbers appearing on modification cards always refer to the sequence numbers appearing on the listing of the compressed deck with which they are used. Be extremely cautious of a listing produced on an ML run which had modifications.

(2) All information in CC1-64 is packed into the compressed deck; therefore, sequencing done in $CC < 65$ will cause a larger deck.

(3) On the DEC option: If either an E or a decimal point is used, a B must be specified; if none of these characters (E, ., or B) appear, an integer scaled at 17 will result.

(4) Numeric information appearing in the variable field of operatio codes will be considered decimal if three characters or less are punched, and octal if four characters or more are punched. The principal pseudo-ops are exceptions to this treatment. The variable field of ORG, DEFINE, and OCT is always treated as octal while that of DEC is always treated as decimal.

(5) If the first two symbolic cards of any deck that is to be compil have an asterisk in column 1, they will be used as header cards and will be printed at the top of each page of the LEMAP listing.

*
*

SAMPLE LISTING FOR THE LEM
ASSEMBLY PROGRAM (LEMAP)

```
                                04/28/66     PAGE      1.
        *
        *
        *    THERE ARE FOUR FORMS FOR THE OCTAL PORTION OF THE LEMAP
        *    LISTING.
        *
        *     (1) THE LEM MACHINE OPERATIONS HAVE THE FOLLOWING FORM
        *              BC EF  LLLL  OO R AAAA
        *              WHERE  B=X DRIVER
        *                     C=Y DRIVER
        *                     E=X SWITCH
        *                     F=Y SWITCH
        *                     LLLL=LOCATION OF THE GIVEN INSTRUCTION
        *                     OO=OPERATION CODE
        *                     R=INDEX REGISTER BIT
        *                     AAAA=VALUE OF THE ADDRESS FIELD
        *
        *     (2) THE DATA GENERATING PSEUDO-OPERATIONS, I.E., DEC AND OCT
        *          HAVE THE FOLLOWING FORM
        *              BC EF  LLLL  DDDDDD
        *              WHERE  B=X DRIVER
        *                     C=Y DRIVER
        *                     E=X SWITCH
        *                     F=Y SWITCH
        *                     LLLL=LOCATION OF THE GIVEN INSTRUCTION
        *                     DDDDDD=DATA GENERATED
        *
        *     (3) ALL NON DATA GENERATING PSEUDO-OPERATIONS, EXCEPT BSS
        *          AND BES HAVE THE FOLLOWING FORM
        *                                AAAA
        *              WHERE  AAAA=VALUE OF THE ADDRESS FIELD
        *
        *     (4) THE PSEUDO-OPERATIONS BSS AND BES HAVE THE FOLLOWING FORM
        *                                LLLL
        *              WHERE  LLLL=LOCATION OF THE FIRST CELL OF THE BLOCK
        *                             FOR A BSS
        *                 AND LLLL=LOCATION OF THE CELL IMMEDIATELY FOLLOWING
        *                             THE LAST CELL OF THE BLOCK FOR A BES
```

```
*                SAMPLE LISTING FOR THE LEM
*                ASSEMBLY PROGRAM (LEMAP)                                          7332.3-17
                                                   04/28/66     PAGE      2      Page A-26
                              40.    *
                              41.    *
                              42.    *
                              43.    *  CHECKSUM
                              44.    *    THE VALUE PRINTED AFTER THE END CARD AS THE CHECKSUM IS
                              45.    *    THE 2 S COMPLEMENT OF THE SUM, DISREGARDING OVERFLOW,
                              46.    *    OF THE CONTENTS OF LOCATIONS 4000(8) TO 7776(8),
                              47.    *    INCLUSIVE, AND IS STORED IN LOCATION 7777(8)
                              48.    *
                              49.    *
                              50.    *
                              51.    *
                              52.    *  THE FOLLOWING PROGRAM IS NOT MEANT TO BE EXECUTED. ITS PURPOSE
                              53.    *  IS TO FAMILIARIZE THE READER WITH THE FORMAT AND ERROR CODES
                              54.    *  OF THE LEMAP LISTING.
                              55.    *
                              56.    *
00 00  0000  010000           57.    X1     OCT      10000,1000
00 01  0001  001000
01 00  0002  777777           58.    A      OCT      -1,1,-3,3
01 01  0003  000001
02 00  0004  777775
02 01  0005  000003
03 00  0006  252525           59.           OCT      252525
03 01  0007  525252           60.           OCT      525252,-125252
04 00  0010  652526
04 01  0011  000000           61.    B      OCT      0
                    0012      62.    C      BSS      8
                    0022      63.    C1     BSS      8
                    0032      64.    C2     BSS      32
                              65.    *  THE NEXT INSTRUCTION SPECIFIES A NUMBER THAT IS TOO LARGE FOR
                              66.    *  THE LEM COMPUTER, I.E., IT CANNOT BE REPRESENTED BY 18 BITS.
*********************************   ILLEGAL SITUATION IN AREA OF ALTER NUMBER    67
       0072                   67.           OCT      1252525
                    0200      68.    M      DEFINE   200
                    0003      69.    IEQU   EQU      A+1
05 07  0073  50 1 0007        70.           AXT      B-A,1
```

```
    *                  SAMPLE LISTING FOR THE LEM
    *                  ASSEMBLY PROGRAM (LEMAP)
                                                    04/28/66    PAGE     3
    06 06  0074  20 1 0002     71.   L1    CLA     A,1
    06 07  0075  22 0 0000     72.         ADD     X1
    07 06  0076  10 1 0012     73.         STO     C,1
    07 07  0077  42 1 0074     74.         TIX     L1,1
    00 10  0100  50 1 0007     75.         AXT     B-A,1
    00 11  0101  20 1 0012     76.         CLA     C,1
    01 10  0102  46 0 0105     77.         TMI     NEG
    01 11  0103  42 1 0101     78.         TIX     *-2,1
    02 10  0104  40 0 0107     79.         TRA     NEG+2
    02 11  0105  14 1 0012     80.   NEG   LDQ     C,1
    03 10  0106  12 0 0011     81.         STQ     B
    03 11  0107  20 0 0011     82.         CLA     B
                               83.   * THE NEXT INSTRUCTION HAS AN UNDEFINED SYMBOL IN ITS ADDRESS
                               84.   * FIELD.
    ********************************  ILLEGAL SITUATION IN AREA OF ALTER NUMBER     85
 U  04 10  0110  46 0          85.         TMI     NEGC
    04 11  0111  50 1 0007     86.         AXT     7,1
    05 10  0112  20 1 0002     87.   L2    CLA     A,1
    05 11  0113  22 0 0001     88.         ADD     X1+1
    06 10  0114  10 1 0012     89.         STO     C,1
                               90.   * THE NEXT INSTRUCTION IS A TIX WITH THE TAG FIELD OMITTED.
    ********************************  ILLEGAL SITUATION IN AREA OF ALTER NUMBER     91
TM  06 11  0115  42 1 0112     91.         TIX     L2
    07 10  0116  20 0 0002     92.         CLA     A
    07 11  0117  22 0 0011     93.         ADD     B
    00 12  0120  10 0 0014     94.         STO     C+2
    00 13  0121  20 0 0011     95.         CLA     A+7
                               96.   * THE NEXT INSTRUCTION HAS A NEGATIVE ADDRESS FIELD.
    ********************************  ILLEGAL SITUATION IN AREA OF ALTER NUMBER     97
IADR 01 12  0122  10 0-0002    97.         STO     A-4
                       0310    98.         ORG     310
    14 10  0310  50 1 0007     99.   L3    AXT     7,1
    14 11  0311  20 1 0002    100.         CLA     A,1
    15 10  0312  06 0 0347    101.         MPY     DC1
    15 11  0313  46 0 0321    102.         TMI     *+6
    16 10  0314  42 1 0311    103.         TIX     L3+1,1
    16 11  0315  20 0 0347    104.         CLA     DC1
```

```
*                    SAMPLE LISTING FOR THE LEM
*                    ASSEMBLY PROGRAM (LEMAP)
                                                       04/28/66     PAGE    4
    17 10  0316  22 0 0370      105.            ADD    DC+10
    17 11  0317  10 0 0347      106.            STO    DC1
    10 12  0320  40 0 0310      107.            TRA    L3
    10 13  0321  10 0 0350      108.            STO    DC1+1
                                109.     *  THE NEXT INSTRUCTION IS AN AXT WITH AN ADDRESS FIELD THAT
                                110.     *  EXCEEDS THE CAPACITY OF THE INDEX REGISTER (3 BITS).
    ********************************      ILLEGAL SITUATION IN AREA OF ALTER NUMBER    111
ADR 11 12  0322  50 1 0012      111.            AXT    10,1
    11 13  0323  20 0 0005      112.     D      CLA    A+3
    12 12  0324  24 0 0006      113.            SUB    A+4
    12 13  0325  24 0 0012      114.            SUB    A+8
    13 12  0326  22 0 0014      115.     E      ADD    A+10
                       0002     116.     KING   EQU    A
    13 13  0327  20 0 0200      117.            CLA    M
    14 12  0330  10 0 0002      118.            STO    KING
                       0000     119.     IAD    DEFINE 10000
                                120.     *  THE NEXT INSTRUCTION HAS AN ADDRESS FIELD THAT EXCEEDS THE
                                121.     *  MEMORY CAPACITY OF THE LEM COMPUTER.
    ********************************      ILLEGAL SITUATION IN AREA OF ALTER NUMBER    122
ADR 14 13  0331  20 0 0000      122.            CLA    IAD
    15 12  0332  56 0 0001      123.     OCTOP1 ALS    1
    15 13  0333  10 0 0005      124.            STO    B-4
    16 12  0334  20 0 0003      125.            CLA    IEQU
    16 13  0335  10 0 0037      126.            STO    C2+5
                                127.     *  THE NEXT INSTRUCTION IS AN AXT WITH THE TAG FIELD OMITTED.
    ********************************      ILLEGAL SITUATION IN AREA OF ALTER NUMBER    128
M   17 12  0336  50 1 0007      128.            AXT    7
    17 13  0337  20 1 0002      129.     L4     CLA    A,1
    10 14  0340  14 1 0012      130.            LDQ    C,1
    10 15  0341  04 0 0376      131.            DVP    DC+16
    11 14  0342  10 1 0022      132.            STO    C1,1
    11 15  0343  42 1 0337      133.            TIX    L4,1
                                134.     *  THE NEXT INSTRUCTION HAS AN ILLEGAL MNEMONIC IN ITS OPERATI
                                135.     *  FIELD.
    ********************************      ILLEGAL SITUATION IN AREA OF ALTER NUMBER    136
IOP 12 14  0344     0 0002      136.            RNT    A
    12 15  0345  66 0 0015 -    137.            OUT    13
```

```
13 14  0346  40 0 7774      138.              TRA     L5
13 15  0347  000000         139.      DC1     DEC     0,0
14 14  0350  000000
14 15  0351  700000         140.              DEC     -.5B1
15 14  0352  700000         141.              DEC     -.5B+1
15 15  0353  700000         142.              DEC     -5E-1B1
16 14  0354  700000         143.              DEC     -5E-1B+1
16 15  0355  100000         144.              DEC     .5B1
17 14  0356  000725         145.      DC      DEC     +.3E5B23
17 15  0357  700000         146.              DEC     -.5B1,-.2121B-2,309.6B10
10 16  0360  446637
10 17  0361  115315
                            147.      * THE NEXT INSTRUCTION SPECIFIES A NUMBER THAT IS TOO LARGE FO
                            148.      * THE GIVEN BINARY SCALING.
********************************      ILLEGAL SITUATION IN AREA OF ALTER NUMBER    149
       0362                 149.              DEC     309.6B7
11 17  0363  331141         150.              DEC     .2121B-2
12 16  0364  446637         151.              DEC     -.2121B-2
12 17  0365  220626         152.              DEC     .1414B-2
13 16  0366  557152         153.              DEC     -.1414B-2
                            154.      * THE NEXT INSTRUCTION SPECIFIES A NUMBER THAT IS TOO LARGE FO
                            155.      * THE LEM COMPUTER, I.E., IT CANNOT BE REPRESENTED BY 18 BITS.
********************************      ILLEGAL SITUATION IN AREA OF ALTER NUMBER    156
       0367                 156.              DEC     262144
14 16  0370  072724         157.              DEC     15082.B+16
14 17  0371  231312         158.              DEC     .749E-1B-3
15 16  0372  776650         159.              DEC     -1200.B18
15 17  0373  014000         160.              DEC     3.B+6
16 16  0374  656373         161.              DEC     -.398E-1B-3
16 17  0375  700000         162.              DEC     -.5B1,-.2121B-2,309.6B10
17 16  0376  446637
17 17  0377  115315
20 00  0400  662463         163.              DEC     -309.6B10
20 01  0401  000142         164.              DEC     98,602,786
21 00  0402  001132
21 01  0403  001422
                            165.      * THE NEXT INSTRUCTION SPECIFIES A FIXED POINT NUMBER WITH THE
```

```
                                  166.     * . B FIELD (BINARY POINT) OMITTED.
 ********************************         ILLEGAL SITUATION IN AREA OF ALTER NUMBER    167
BD         0404                   167.              DEC     -.5
     22 01 0405  700000           168.              DEC     -.581,581,309.6B10
     23 00 0406  001105
     23 01 0407  115315
     24 00 0410  000142           169.              DEC     98,309.6B10,602
     24 01 0411  115315
     25 00 0412  001132
                         0423     170.     C3       BES     8
     21 03 0423  000155           171.              DEC     109,-.2121B10,778
     22 02 0424  777745
     22 03 0425  001412
     23 02 0426  614632           172.              DEC     -.45B0,309.6B10,309.6B10
     23 03 0427  115315
     24 02 0430  115315
     24 03 0431  115315           173.              DEC     309.6B10,-.5B1,-.2121B-2
     25 02 0432  700000
     25 03 0433  446637
     26 02 0434  700000           174.              DEC     -.5B1,-.2121B-2,309.6B10
     26 03 0435  446637
     27 02 0436  115315
     27 03 0437  700000           175.              DEC     -.5B1,-.2121B-2,309.6B10
     20 04 0440  446637
     20 05 0441  115315
                                  176.     *
                         3774     177.              ORG     3774
     76 36 3774  012156           178.              DEC     326.848B13
     76 37 3775  765622           179.              DEC     -326.848B13
     77 36 3776  002437           180.              DEC     .01B0
     77 37 3777  775341           181.              DEC     -.01B0
     00 40 4000  200000           182.              DEC     .5000025B0          NO ROUND
     00 41 4001  600000           183.              DEC     -.5000025B0
     01 40 4002  200001           184.              DEC     .500004B0           ROUND
     01 41 4003  577777           185.              DEC     -.500004B0
     02 40 4004  235361           186.              DEC     .173139971E15B48
     02 41 4005  542417           187.              DEC     -.173139971E15B48
```

```
*                    SAMPLE LISTING FOR THE LEM
*                    ASSEMBLY PROGRAM (LEMAP)
                                                        04/28/66     PAGE      7
  03 40   4006  064112          188.           DEC      .497291E-4B-12
  03 41   4007  713666          189.           DEC      -.497291E-4B-12
  04 40   4010  000000          190.           DEC      .3815E-5B17
  04 41   4011  000000          191.           DEC      -.3815E-5B17
  05 40   4012  000001          192.           DEC      .156B14
  05 41   4013  777777          193.           DEC      -.156B14
  06 40   4014  774761          194.           DEC      -.18488E-3B-6
  06 41   4015  003017          195.           DEC      .18488E-3B-6
  07 40   4016  252517          196.           DEC      .16665554B-2
  07 41   4017  525261          197.           DEC      -.16665554B-2
                       7774     198.           ORG      7774
  76 76   7774  50 1 0007       199.     L5    AXT      7,1
  76 77   7775  20 1 0002       200.           CLA      A,1
  77 76   7776  24 0 0001       201.           SUB      X1+1
  77 77   7777  10 1 0423       202.           STO      C5,1
                                203.     *  THE NEXT INSTRUCTION IS TO BE STORED IN A LOCATION THAT
                                204.     *  EXCEEDS THE MEMORY CAPACITY OF THE LEM COMPUTER.
  ********************************        ILLEGAL SITUATION IN AREA OF ALTER NUMBER     205
ILOC      0000  42 1 7775       205.           TIX      L5+1,1
                       0423     206.     C5    SYN      C3
                       0000     207.           END

  CHECKSUM=617766
```

```
    *               SAMPLE LISTING FOR THE LEM
    *               ASSEMBLY PROGRAM (LEMAP)
                                                  04/28/66      PAGE      8

UNDEFINED SYMBOLS

    NEGC
```

```
  *                   SAMPLE LISTING FOR THE LEM
  *                   ASSEMBLY PROGRAM (LEMAP)
                                                     04/28/66     PAGE      9

YMBOL DICTIONARY TABLE

 A          58          B          61          C          62          C1         63          C2         64
 C3         170         C5         206         D        X 112         DC         145         DC1        139
 E        X 115         IAD        119         IEQU       69          KING       116         L1         71
 L2         87          L3         99          L4         129         L5         199         M          68
 NEG        80          OCTOP1   X 123         X1         57
```

```
*                 SAMPLE LISTING FOR THE LEM
*                 ASSEMBLY PROGRAM (LEMAP)                    04/28/66     PAGE      10

       SYMBOL REFERENCE TABLE

7 X1                        72 ADD      88 ADD$    201 SUB$

8 A                         69 EQU$     70 AXT$*    71 CLA *    75 AXT$*    87 CLA *    92 CLA      95 CLA
                            97 STO$    100 CLA *   112 CLA$    113 SUB$    114 SUB$    115 ADD$    116 EQU
                           129 CLA *   136 OUT     200 CLA *

1 B                         70 AXT$*    75 AXT$*    81 STQ      82 CLA      93 ADD      95 CLA     124 STO

2 C                         73 STO *    76 CLA *    80 LDQ *    89 STO *    94 STO$    114 SUB     130 LDQ

3 C1                       132 STO *

4 C2                       126 STO$

8 M                        117 CLA

9 IEQU                      69 EQU     125 CLA

71 L1                       74 TIX *

80 NEG                      77 TMI      79 TRA$

87 L2                       91 TIX

99 L3                      103 TIX$*   107 TRA

12 D          NOT USED

15 E          NOT USED

16 KING                    118 STO

19 IAD                     122 CLA
```

```
    *                  SAMPLE LISTING FOR THE LEM
    *                  ASSEMBLY PROGRAM (LEMAP)
                                                  04/28/66       PAGE       11

123 OCTOP1    NOT USED

129 L4                   133 TIX *

139 DC1                  101 MPY      104 CLA      106 STO      108 STO$

145 DC                   105 ADD$     131 DVP$

170 C3                   206 SYN

199 L5                   138 TRA      205 TIX$*

206 C5                   202 STO *
```

```
 *                    SAMPLE LISTING FOR THE LEM
 *                    ASSEMBLY PROGRAM (LEMAP)
                                                       04/28/66     PAGE      12

  CHECK FOLLOWING SEQUENCE NUMBERS FOR ILLEGAL SITUATIONS
67     85     97    111    122    136    149    156    167    205
```

# APPENDIX B

## TABLES

## INSTRUCTION REPERTOIRE

| Mnemonic | Octal Code | Description | Execution Time (µsec) |
|---|---|---|---|
| ABS | 62 | Absolute Value of Accumulator | 16 |
| ADD | 22 | Add | 10 |
| ADZ | 32 | Add and Zero | 10 |
| ALS | 56 | Accumulator Left Shift | 3N + 13 |
| AXT | 50 | Address to Index | 13 |
| CLA | 20 | Clear and Add | 10 |
| CLZ | 30 | Clear, Add and Zero | 10 |
| COM | 60 | Complement Accumulator | 16 |
| DLY | 70 | Delay | |
| DVP | 04 | Divide | 73 |
| INP | 64 | Input | 16 or 67 |
| LDQ | 14 | Load Q Register | 13 |
| LLS | 52 | Long Left Shift | 3N + 13 |
| LRS | 54 | Long Right Shift | 3N + 13 |
| MPR | 26 | Multiply and Round | 70 |
| MPY | 06 | Multiply | 70 |
| MPZ | 36 | Multiply and Zero | 70 |
| OUT | 66 | Output | 13 |
| STO | 10 | Store Accumulator | 13 |
| STQ | 12 | Store Q Register | 13 |
| SUB | 24 | Subtract | 10 |
| SUZ | 34 | Subtract and Zero | 10 |
| TIX | 42 | Test Index and Transfer | 10 |
| TMI | 46 | Transfer on Minus Accumulator | 10 |
| TOV | 44 | Transfer on Overflow | 10 |
| TRA | 40 | Transfer | 10 |
| TSQ | 72 | Transfer and Set Q | 16 |
| | 00<br>02<br>16<br>74<br>76 | Unassigned Operation Codes | |

TABLE A

AEA OUTPUT REGISTERS

| NAME | TYPE OF REGISTERS | ADDRESS |
|---|---|---|
| sin θ | 9 Bits plus Sign | 2001 |
| cos θ | " " " " | 2002 |
| sin ∅ | " " " " | 2004 |
| cos ∅ | " " " " | 2010 |
| sin ψ | " " " " | 2020 |
| cos ψ | " " " " | 2040 |
| $E_x$ | " " " " | 6001 |
| $E_y$ | " " " " | 6002 |
| $E_z$ | " " " " | 6004 |
| Lateral Velocity | 8 Bits plus Sign | 6020 |
| Altitude, Altitude Rate | 14 Bits plus Sign | 6010 |
| Output Telemetry* | 24 Bit Shift Register | |
| Word 1 | Bits 0-17 | 6200 |
| Word 2 | Bits 6-23 | 6100 |
| DEDA | 4 Bit Shift Register | 2200 |

*Only Word 1 is used for GSE data output.

TABLE B

AEA INPUT REGISTERS

| Name | Type of Register | Address |
|---|---|---|
| $\theta$ (PGNS) | Integrating Reg. | 2001 |
| $\phi$ (PGNS) | " " | 2002 |
| $\psi$ (PGNS) | " " | 2004 |
| $\Delta V_x$ | 11 Bit Counter | 6020 |
| $\Delta V_y$ | " " " | 6040 |
| $\Delta V_z$ | " " " | 6100 |
| $\Delta \int q$ | " " " | 6002 |
| $\Delta \int p$ | " " " | 6010 |
| $\Delta \int r$ | " " " | 6004 |
| Downlink Telemetry | 18 Bit Shift Register | 6200 |
| Disc. Inp. Word 1 | 8 Bits | 2020 |
| Disc. Inp. Word 2 | 7 Bits | 2040 |
| DEDA | 4 Bit Shift Register | 2200 |

TABLE C

AEA DISCRETE INPUTS

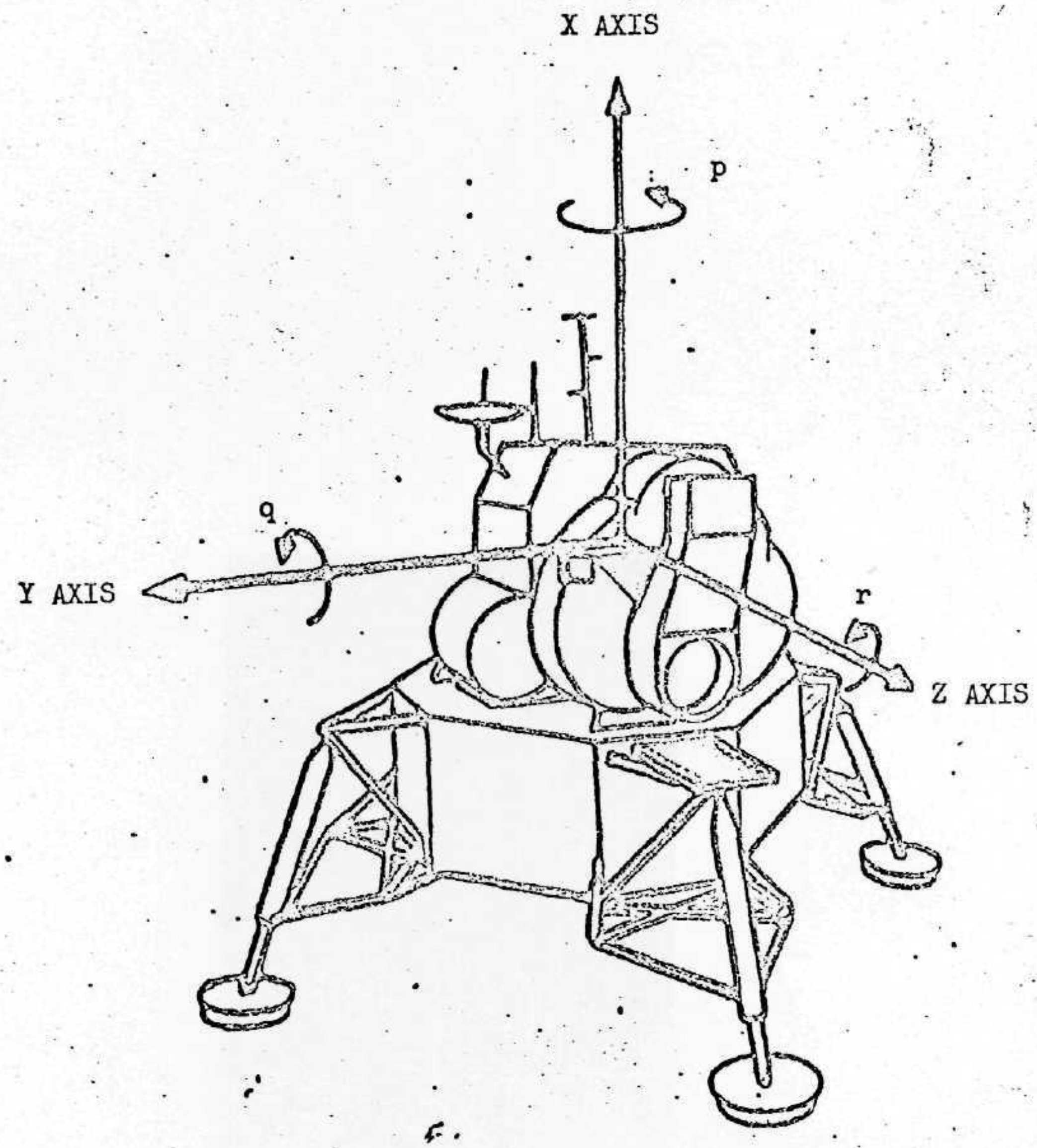
| BIT POSITION | DISCRETE WORD 1 | DISCRETE WORD 2 |
|---|---|---|
| 1 | Downlink Telemetry Stop | GSE Discrete 1 |
| 2 | Output Telemetry Stop | GSE Discrete 2 |
| 3 | Follow-Up | GSE Discrete 3 |
| 4 | Automatic | DEDA Clear |
| 5 | Descent Engine On | DEDA Hold |
| 6 | Ascent Engine On | DEDA Enter |
| 7 | Abort | DEDA Readout |
| 8 | Abort Stage | |

TABLE D

AEA DISCRETE OUTPUTS

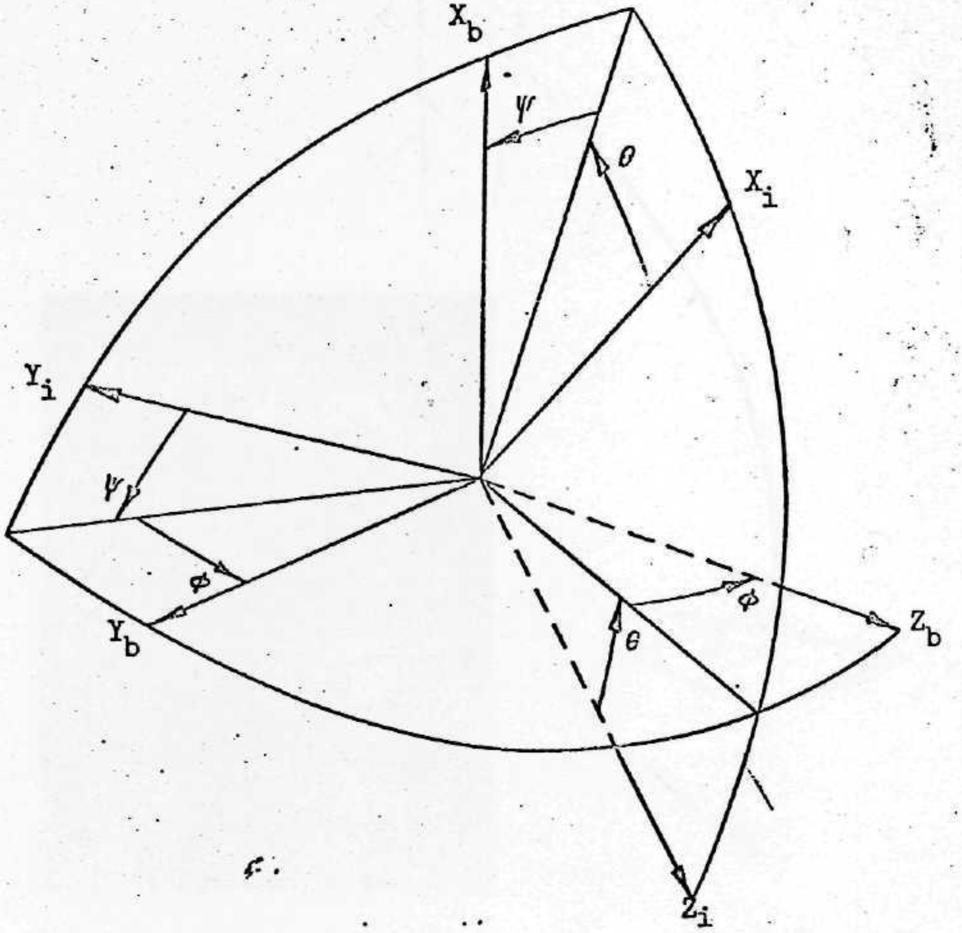
| NAME | SET | RESET |
|---|---|---|
| Ripple Carry Inhibit | 2410 | 3010 |
| Altitude | 2420 | 3040 |
| Altitude Rate | 2440 | 3040 |
| DEDA Shift In | 2500 | ---- |
| DEDA Shift Out | 2600 | ---- |
| GSE Discrete 4 | 6401 | 7001 |
| GSE Discrete 5 | 6402 | 7002 |
| GSE Discrete 6 | 6404 | 7004 |
| Test Mode Failure | 6410 | 7010 |
| Engine Off | 6420 | 7020 |
| Engine On | 6440 | 7040 |

TABLE E



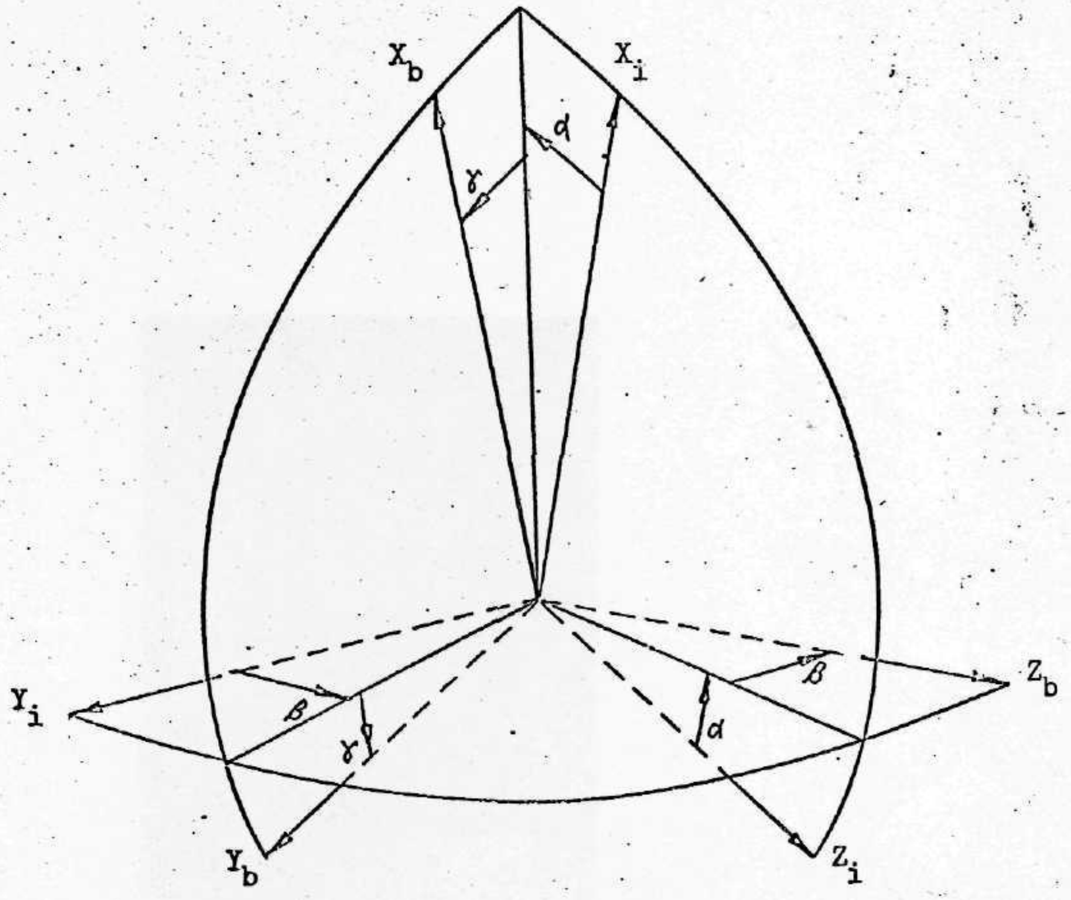
In Figure F, the direction of the arrows about the axes indicates positive angular rate (p, q, r) and displacement. The direction of the arrows along the axes indicates the direction of positive translational acceleration and velocity.

FIGURE F. VEHICLE REFERENCE AXES



Euler angles ($\theta$, $\psi$, $\phi$) are used to specify the vehicle reference frame ($X_b$, $Y_b$, $Z_b$) with respect to the PGNS inertial reference frame ($X_i$, $Y_i$, $Z_i$)

FIGURE G.



Euler angles (α, β, γ) used to specify the vehicle reference frame ($X_b$, $Y_b$, $Z_b$
with respect to the FDAI reference frame ($X_i$, $Y_i$, $Z_i$)

FIGURE H.

| | |
|---|---|
| 00 | |
| 01 | |
| 02 | |
| 03 | |
| 04 | DVP |
| 05 | |
| 06 | MPY |
| 07 | |
| 10 | STO |
| 11 | |
| 12 | STQ |
| 13 | |
| 14 | LDQ |
| 15 | |
| 16 | |
| 17 | |
| 20 | CLA |
| 21 | |
| 22 | ADD |
| 23 | |
| 24 | SUB |
| 25 | |
| 26 | MPR |
| 27 | |
| 30 | CLZ |
| 31 | |
| 32 | ADZ |
| 33 | |
| 34 | SUZ |
| 35 | |
| 36 | MPZ |
| 37 | |
| 40 | TRA |
| 41 | |
| 42 | TIX |
| 43 | |
| 44 | TOV |
| 45 | |
| 46 | TMI |
| 47 | |
| 50 | AXT |
| 51 | |
| 52 | LLS |
| 53 | |
| 54 | LRS |
| 55 | |
| 56 | ALS |
| 57 | |
| 60 | COM |
| 61 | |
| 62 | ABS |
| 63 | |
| 64 | INP |
| 65 | |
| 66 | OUT |
| 67 | |
| 70 | |
| 71 | |
| 72 | TSQ |