TECHNICAL MEMORANDUM (NASA) 43

DEMONSTRATION PROGRAM FOR OMEGA RECEIVER PROTOTYPE MICROCOMPUTER DATA PROCESSING

Using the prototype Omega receiver developed for the NASA Joint University Program plus a digital interface to a commercial microcomputer, a software routine to demonstrate receiver operation is described.

by

Robert W. Lilley
Avionics Engineering Center
Department of Electrical Engineering
Ohio University
Athens, Ohio 45701

November 1976

Supported by

National Aeronautics and Space Administration
Langley Research Center
Hampton, Virginia
Grant NGR 36-009-017

(NASA-CR-149226) DEMONSTRATION PROGRAM FOR OMEGA RECEIVER PROTOTYPE MICROCOMPUTER DATA PROCESSING (Ohio Univ.) 11 p HC A02/MF A01 CSCL 17G

N77-12029

Unclas
G3/04 55800

I. INTRODUCTION

The Omega project group of the Ohio University Avionics Engineering Center is evaluating the JOLT(TM) microcomputer, based on the MOS Technology 6502 processor chip, for use in Omega navigation systems. The computer program described in this memorandum was prepared in hand-assembled code for the JOLT, and makes use of the hardware interface unit [1] which attaches the JOLT to the Ohio University Prototype Omega Receiver. [2]

The program prints two Omega LOP values and plots the LOP values. The LOP's are selected by the front panel switches on the Prototype Receiver.

This paper provides program documentation; reference is made to other Technical Memoranda for descriptions of the receiver and interface hardware.

II. PROGRAM DESCRIPTION

A program flow chart appears in Figure 1. Initialize code first sets interrupt addresses for the Non-Maskable Interrupt (NMI) in the JOLT processor. The "current-time-slot" flag is set to station D, which is the first data made available after the receiver attains sync. The processor status byte is set to provide binary processing with interrupts enabled, a carriage return is given to initialize the teleprinter, and a jump is performed to enter the program loop to wait for an interrupt.

The program loop operates continuously, testing the interrupt flag in memory to determine whether a complete Omega sequence has been received (stations A-H). When one complete ten-second sequence has elapsed, as determined by the receipt of eight interrupts from the receiver (one for each Omega time-slot), the program loop branches to the print routine.

The interrupt service routine is called whenever an NMI interrupt is received from the receiver-computer interface. [1] The interrupt signal signifies that an Omega time-slot has passed, and that a new data value for that Omega station is available. The interrupt routine reads the receiver status word and determines whether the current time-slot is the A slot. If so, the interrupt flag, which is also the data index pointer, is reset to zero. The program then reads the phase data word from the interface and stores the data in the A data word in the interrupt data buffer. The status word is stored in the A word in the status buffer. If the time-slot is not A, the interrupt flag/pointer is incremented by one to index the phase and status to the proper buffer words for later use by the print routine.

The interrupt routine executes once per Omega time-slot and eventually produces an eight-word data block, containing, in order, phase data for each Omega station. It is important to point out that the Omega Prototype Receiver selects a maximum of four Omega stations at a time. Therefore, an additional eight-word data buffer is produced, containing the receiver status word for each time slot. Reference [1] describes the status word in detail.

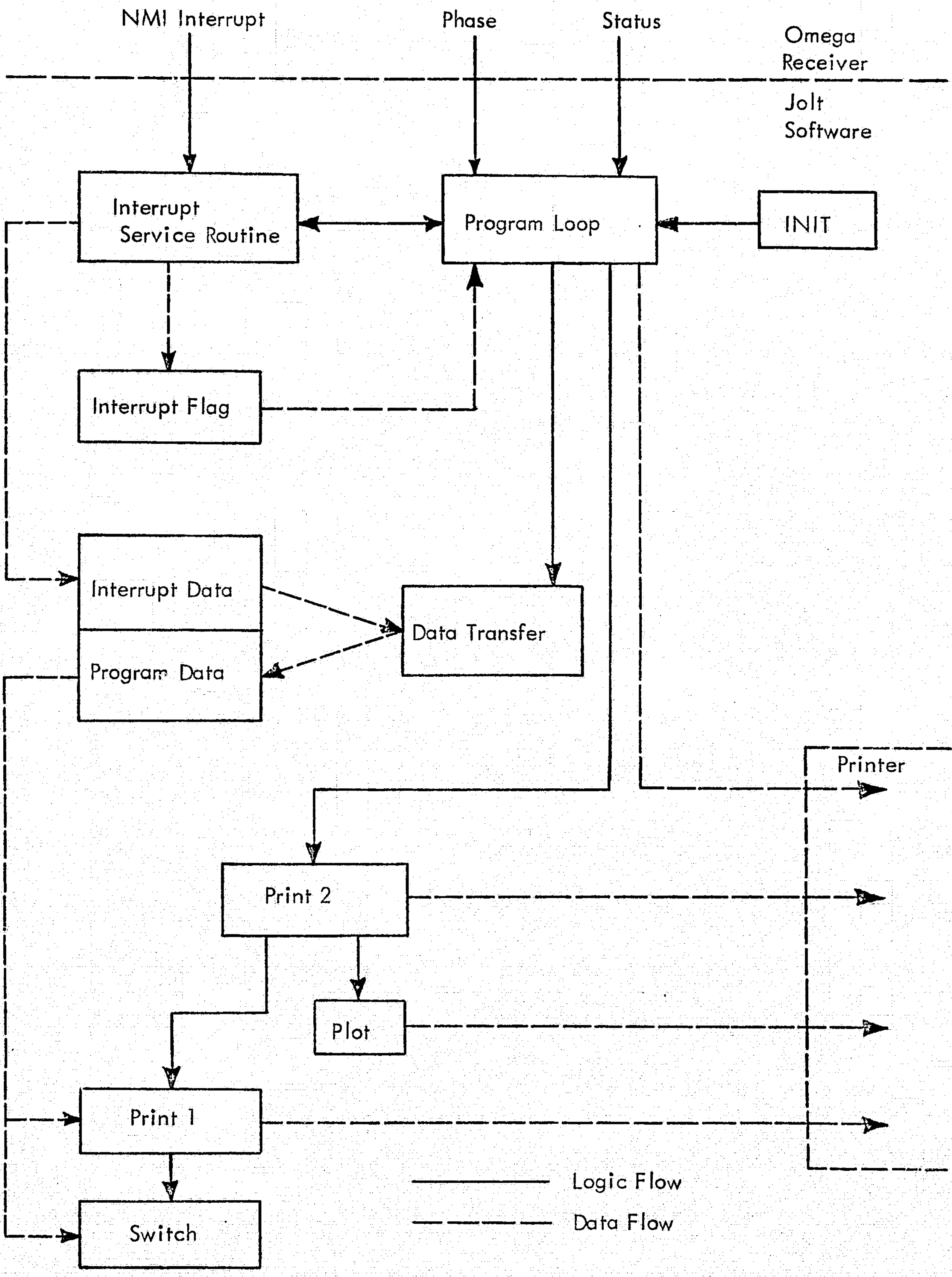

Figure 1. JOLT Interface Software Flow Chart.

For program description purposes, it is sufficient to say that the status word contains information on receiver selector switch positions and a bit which signifies the A time-slot, for data synchronization.

When the interrupt routine returns an interrupt flag of 8 (Station H) to the program loop routine, the latter clears the interrupt flag to avoid multiple executions of the print routine, and it calls DATA to copy the interrupt and status buffers into print storage to avoid alteration by the interrupt routine during printing. Then PRINT1 is called twice to print the two sections of the output. A carriage return and line feed are given to reset the teleprinter for the next line, and the routine branches back to the program loop to wait for another interrupt flag.

The PRINT2 subroutine calls PRINT1 twice to obtain two phase values making up an LOP. It then subtracts these values and prints the LOP value on the teleprinter. It then calls the PLOT subroutine to cause a plot of the LOP value to be printed.

Figure 2 shows a portion of the program output in actual operation. The C-D and G-D LOPs are selected on the Omega Receiver. Graphs are plotted modulo 16 spaces due to paper size. Note the occasional spacing errors. These are introduced by Omega interrupts occurring during operation of the JOLT computer monitor routines. These interrupts disturb the monitor software timing routines which drive the teleprinter. For this demonstration routine, this occasional problem was neglected; future designs involve a printer interface using parallel logic attached to the JOLT data bus. Since JOLT monitor software will not perform in-line timing operations for the parallel-connected printer, this interrupt interference will vanish.

The software routine, in combination with the Omega Receiver Interface,[1] demonstrates that a commercially-available microprocessor can accept Omega data from the Prototype Receiver and can produce Omega LOP information as input for navigation processing. Development continues in this software area under the Tri-University program.

III. PROGRAM LISTING

The software reported here is hand-assembled code for the JOLT (TM) microcomputer. The routines intercommunicate by use of the JOLT Jump-to-Subroutine command and the hardware stack implemented in the MOS Technology 6502 CPU chip used in the JOLT.

A. Initialization. This routine initializes the NMI interrupt vector at locations FFFA and FFFB to point to address 0000, sets the interrupt flag for Omega time-slot D (03), initializes the stack pointer to FF_{16} and prints a carriage return-line feed. Control is then passed to the program loop at location 0020.

```
C=2D,D=0F,'.CP=1E          *          C=2D,D=0F,'.P=1E          *
C=33 D=14,'.CP=1F          *          C=32,D=14,'.CP=1F          *
C=3B,D=16,'.CP=25            *        C=34,D=16,'.CP=1F          *
C=02 D=1B,'.CP=27            *        C=33,D=1B,'.CP=1B         *
C=03 D=20,'.CP=23           *         C=34,D=20,'.CP=14        *
C=01 D=23,'.CP=1E          *          C=33,D=23,'.CP=15        *
C=02 D=25,'.CP=1D          *          G=33,D=25,'.CP=13       *
C=06 D=2B,'.CP=1B         *           G=39,D=2B,'.CP=0E      *
C=05 , D=2F,'.CP=16       *            G=39,D=2F,'.CP=0A     *
C=01 D=32,'.CP=0F      *              G=3F,D=32,'.CP=0C      *
C=3D D=35,'.CP=03    *                G=39,D=35,'.CP=04 *
C=35 D=3A,'.CP=3B                 *   G=37,D=3A,'.CP=3D                   *
C=2F D=3E,'.CP=30               *     G=37,D=3E,'.CP=39                  *
C=2C D=02,'.CP=2A             *       G=34,D=02,'.CP=32                *
C=2F D=08,'.CP=26            *        G=31,D=08,'.C=29               *
C=2F D=08,'.CP=27            *        G=2D,D=08,'.CP=25              *
C=2D,D=0C,'.CP=21           *         G=2D,D=0C,'.CP=21             *
C=29 D=11,'.CP=13         *           G=33,D=11,'.CP=22             *
C=29 D=14,'.CP=15        *            G=32,D=14,'.CP=1E            *
C=29 D=13,'.CP=11       *             G=37,D=13,'.CP=1F            *
C=20 D=1A,'.CP=06 *                   G=37,D=1A,'.CP=1D            *
C=17 D=21,'.CP=36                *    G=36,D=21,'.CP=15          *
C=11 D=23,'.CP=2E              *      G=3D,D=23,'.CP=1A           *
C=13 D=26,'.CP=32               *     G=00,D=26,'.CP=1A           *
C=16 D=2A,'.CP=2C              *      G=04,D=2A,'.CP=1A           *
C=1B, D=2C,'.CP=2F             *      G=09,D=2C,'.CP=1D            *
C=20 D=2F,'.CP=31               *     G=04,D=2F,'.CP=15          *
C=1D D=27,'.CP=36                *    G=05,D=27,'.CP=1E            *
C=19 D=26,'.CP=33               *     G=01,D=26,'.CP=1B           *
C=16, D=2B,'.CP=2B            *       G=02,D=2B,'.CP=17          *
C=16 D=30,'.CP=26            *        G=00,D=30,'.CP=10         *
C=1A D=34,'.CP=26            *        G=02,D=34,'.CP=0E        *
C=19 D=38,'.CP=21           *         G=3F,D=38,'.CP=07 *
C=1D D=3B,'.CP=22           *         G=39,D=3B,'.CP=3E                   *
C=1F D=3C,'.CP=23           *         G=37,D=3C,'.C=3B                  *
C=1F D=02,'.CP=1D          *          G=34,D=02,'.CP=32                *
C=19 D=07,'.CP=12      *              G=2E,D=07,'.CP=27              *
C=21 D=08,'.CP=19         *           G=2C,D=08,'.CP=24              *
C=1E D=0B,'.CP=13      *              G=29,D=0B,'.CP=1E            *
C=1E D=11,'.CP=0D    *                G=27,D=11,'.CP=16          *
C=1D D=14,'.CP=09   *                 G=23,D=14,'.CP=14          *
C=1A D=15,'.CP=05 *                   G=23,D=15,'.CP=13         *
C=1 6 D=1D,'.CP=39                 *  G=27,D=1D,'.CP=0A    *
C=12 D=20,'.CP=32               *     G=25,D=20,'.CP=05 *
C= 14,D=25,'.CP=2F              *      G=26,D=25,'.CP=01*
C=14 D=25,'.CP=2F              *      G=24,D=25,'.CP=3F                   *
C=15 , D=2C,'.CP=29            *       G=21,D=2C,'.CP=35                 *
C=16 D=30,'.CP=26            *        G=23,D=30,'.CP=33                *
C=1D D=34,'.CP=29             *       G=1C,D=34,'.CP=28              *
C=1A D=37,'.CP=23           *         G=1B,D=37,'.CP=24             *
C=1C, D=3C,'.CP=20          *         G=1A,D=3C,'.CP=1E           *
C=1E D=3,'.C=20           *          G=1A,D=3E,'.CP=1C          *
C=20 D=01,'.CP=1F           *         G=1F,D=01,'.CP=1E           *
C=1E D=03,'.CP=16        *            G=1D,D=08,'.CP=15          *
C=1E , D=0B,'.CP=13      *             G=1C,D=0B,'.CP=11         *
C=1 G D=0C,'.CP=10       *            G=1E,D=0C,'.CP=12          *
```

ORIGINAL PAGE IS
OF POOR QUALITY

Figure 2. Sample Output Data.

Address	Program Code	Comment
0100	A9 00	Zero Accumulator (AR)
0102	8D FA FF	Store
0105	8D FB FF	Store
0108	A9 03	03 to AR
010A	85 64	Interrupt Flag at 0064
010C	A2 FF	FF in X Register (XR)
010E	9A	FF to Stack Pointer
010F	D8	Set Binary Mode
0110	20 8A 72	CR LF
0113	4C 20 00	To Program Loop

After loading the interface software, address 0100 should be the starting point for execution, to insure correct initialization.

B. Program. The purpose of the Program Loop is to test the interrupt flag to determine whether one full set of Omega measurements have been received. If so (if the flag equals 8, indicating reception of time-slots A through H) print routines are called. If the complete sequence has not been received, the program loop continues cycling.

Address	Program Code	Comment
0020	→ A5 64	Get Interrupt Flag to AR
0022	C9 08	Compare 8 to AR
0024	DO FA	If AR not 8, loop to 0020
0026	A9 00	Zero AR
0028	85 64	Clear Interrupt Flag
002A	20 50 01	Call Data Transfer Routine
002D	20 70 00	Call PRINT2 for LOP 1
0030	20 70 00	Call PRINT2 for LOP 2
0033	20 8A 72	Print CR LF
0036	4C 20 00	Reenter Program Loop

C. Data Transfer Routine. Since Omega interrupts occur during data printout, it is necessary to transfer primary buffer data for Omega phase and status to a secondary buffer. The print routines act on secondary data, allowing real-time update of the primary buffer by the receiver interrupt process.

Address	Program Code	Comment
0150	98	Y Register (YR) to AR
0151	48	Save AR on stack
0152	A0 00	Zero YR
0154	→ B9 38 01	Get Primary Buffer (Y index)
0157	99 28 01	Put Secondary (Y index)
015A	C8	YR = YR + 1
015B	CO10	Compare YR to 16
015D	DO F5	If YR less than 16 loop to 0154
015F	68	Get AR from stack
0160	A8	AR to YR
0161	60	Return from Subroutine

D. Interrupt Service Routine. When the Omega receiver creates a hardware interrupt pulse on the NMI line of the JOLT computer, the program counter is set to address 0000 due to the vector inserted at address FFFA and FFFB by the initialization routine described earlier. The interrupt service routine reads Omega data and status for the current time-slot and places this data in the Primary Data Buffer, indexed by the Omega time-slot number. The interrupt flag is set to the number of the current time-slot. When completed, the routine issues a return instruction, which returns to the interrupted instruction in the JOLT software.

Address	Program Code	Comment
0000	48	Save AR on stack
0001	98	YR to AR
0002	48	Save AR on stack
0003	A4 64	Get Interrupt flag in YR
0005	AD 00 80	Get status from receiver (8000)
0008	0A	Shift AR left 1 bit
0009	48	Save AR on stack
000A	90 02	If Carry = 0, not A time-slot
000C	A0 00	A time-slot; zero YR
000E	AD 00 90	Read receiver phase (9000)
0011	99 40 01	Store Phase (YR index)
0014	68	Get AR from stack
0015	99 38 01	Store Status (YR index)
0018	C8	YR = YR + 1
0019	84 64	Store YR in Interrupt Flag
001B	68	Get AR from stack
001C	A8	AR to YR
001D	68	Get AR from stack
001E	40	Return from Interrupt

Note that the routine reads memory mapped location 8000 to get Omega status and 9000 to get phase data. These are hard-wired locations in the Omega Receiver Interface. The high-order Status bit is on during the A time-slot, also due to receiver hardware. The bit is used for software synchronization of the time-slot index with hardware time-slot generation.

E. PRINT2 - Print/Plot Control Routine. PRINT2 calls a group of other routines to accomplish the actual printing and plotting tasks for one Omega LOP using data from the Omega Prototype Receiver. First, PRINT1 prints the first phase value and then the second. The values are subtracted and the LOP value is printed by PRINT2. The PLOT routine is then called to provide a graph of the LOP value on the teleprinter.

Address	Program Code	Comment
0070	48	Save AR on stack
0071	8A	XR to AR
0072	48	Save AR
0073	20 B0 00	Call PRINT1 (get phase in AR)
0076	85 A7	Save phase value
0078	20 B0 00	Call PRINT1 (get phase in AR)
007B	85 A8	Save phase value
007D	A5 A7	First phase value to AR
007F	38	Set Carry Bit

0080	E5 A8	Subtract phase values to get LOP
0082	29 3F	"and" off two high-order bits
0084	85 A6	Save LOP value
0086	A9 4C	"L" to AR
0088	20 C6 72	Print
008B	A9 4F	"0" to AR
008D	20 C6 72	Print
0090	A9 50	"P" to AR
0092	20 C6 72	Print
0095	A9 3D	"=" to AR
0097	20 C6 72	Print
009A	A5 A6	Get LOP in AR
009C	20 B1 72	Print
009F	20 70 01	Call PLOT Routine
00A2	68	Retrieve AR from stack
00A3	AA	Restore XR
00A4	68	Pull AR from stack
00A5	60	Return to calling program
00A6	(FF)	LOP save byte
00A7	(FF)	Phase value 1 save
00A8	(FF)	Phase value 2 save

F. PRINT1 - Phase Value Print Routine. PRINT1 is called by PRINT2 to obtain output of a single, labeled phase value (e.g. C = 04). PRINT1 returns the phase value to the calling program in the AR in addition to printing the value. PRINT1 calls SWITCH to obtain the identification of LOP switch settings from the Omega Receiver status word. The switch setting determines the label applied to the phase value by PRINT1. PRINT1 is called twice for each LOP.

Address	Program Code	Comment
00B0	8A	XR to AR
00B1	48	Save AR on stack
00B2	98	YR to AR
00B3	48	AR to stack
00B4	20 40 00	Call SWITCH: get current setting in AR
00B7	A8	AR to Y (switch data)
00B8	B9 20 01	Get CHAR(YR) for label in AR
00BB	20 C6 72	Print it
00BE	A9 3D	Get "=" in AR
00C0	20 C6 72	Print it
00C3	B9 30 01	Get Phase (YR) from Data Area in AR
00C6	29 3F	"and" off high order 2 bits
00C8	85 D9	Save phase value
00CA	20 B1 72	Print phase value
00CD	A9 2C	"," in AR
00CF	20 C6 72	Print
00D2	68	Get AR from stack
00D3	A8	AR to YR
00D4	68	Get AR from stack

00D5	AA	AR to XR
00D6	A5 D9	Get phase value in AR
00D8	60	Return to calling program
00D9	(FF)	Save byte for phase value

G. SWITCH - Get Omega Receiver Switch Position Labels. SWITCH is called twice for each LOP printed. The SWITCH routine expects four calls for each print line (two LOPs). In four calls, the SWITCH routine returns the switch positions for the time-slot selector switches on the Omega Prototype Receiver from top to bottom. Routine returns a hex value of 00 for a switch selecting the "A" time-slot, and increasing values up to 07 for a switch selecting the "H" time-slot.

Address	Program Code	Comment
0040	8A	XR to AR
0041	48	Save AR on stack
0042	98	YR to AR
0043	48	Save AR on stack
0044	A9 00	Zero to AR
0046	A8	Zero to YR
0047	85 63	Store Zero
0049	B9 28 01	Get Status Word (YR) in AR
004C	0A	AR left one bit
004D	99 28 01	Store shifted status (YR)
0050	B0 0C	If Carry 0, loop; if 1 store YR
0052	C8	YR = YR + 1
0053	C0 08	Compare YR:8
0055	D0 F2	If YR = 8, continue
0057	68	Get AR from stack
0058	A8	AR to YR
0059	68	Ger AR from stack
005A	AA	AR to XR
005B	A5 63	Get Output in AR
005D	60	Return to caller
005E	84 63	Carry = 1; store output
0060	4C 52 00	Loop
0063	(FF)	Output Save Byte
0064	(FF)	Interrupt Flag Storage Byte

Each selector switch places one bit in the Status Word if it is current in the time-slot being processed by the receiver. The SWITCH routine successively shifts the Status word left, checking the Carry bit to determine which bit(s) are on for the current time-slot.

The interrupt flag is stored here for use by other routines already described.

H. PLOT - Teleprinter Plot Routine. PLOT accepts one value of phase for each call, and prints one "*" character after a number of spaces determined by the four high-order bits of the phase input value. The result is a graph, 16 spaces wide, of the phase values making up one LOP. PLOT is called once for each call to PRINT2 (once per LOP).

Address	Program Code	Comment
0170	A5 A6	Get LOP value in AR
0172	48	Save AR on stack
0173	8A	XR to AR
0174	48	Save AR on stack
0175	A5 A6	Get LOP value in AR
0177	4A	Shift AR right 1 bit
0178	4A	Shift AR right 1 bit
0179	84 A9	Save Shifted AR
017B	A2 00	Zero XR
017D	E4 A9	Compare X to LOP value
017F	F0 0B	If equal, print
0181	20 77 73	Space
0184	E8	XR = XR + 1
0185	E0 10	Compare XR to 16
0187	F0 0F	If X = 16, Done
0189	4C 7D 01	Loop
018C	8A	XR to AR
018D	48	Save AR on stack
018E	A9 2A	Get "*" to AR
0190	20 C6 72	Print
0193	68	Get AR from stack
0194	AA	AR to XR
0195	4C 84 01	Loop
0198	68	Get AR from stack
0199	AA	AR to XR
019A	68	Get AR from stack
019B	60	Return to caller

I. DATA - Data Storage Area. This storage area provides for "live" data storage as interrupts are detected, and for program storage of stable data blocks after the receipt of 8 time-slots.

Address	Program Code	Comment
0120	41 42 43 44	Characters A, B, C, D
0124	45 46 47 48	Characters E, F, G, H
0128	(FF FF FF FF)	Program Storage:
012C	(FF FF FF FF)	8-byte Status Buffer
0130	(FF FF FF FF)	Program Storage:
0134	(FF FF FF FF)	8-byte Phase Buffer
0138	(FF FF FF FF)	Interrupt Storage:
013C	(FF FF FF FF)	8-byte Status Buffer
0140	(FF FF FF FF)	Interrupt Storage:
0144	(FF FF FF FF)	8-byte Phase Buffer

IV. REFERENCES

[1] Lilley, R. W., "A Microprocessor Interface for the Ohio University Prototype Omega Navigation Receiver, Technical Memorandum (NASA) 32, Avionics Engineering Center, Department of Electrical Engineering, Ohio University, Athens, Ohio, August, 1976.

[2] "Prototype Omega Receivers for Use in Data Collection and Evaluation", Final Report, EER 28-1, Avionics Engineering Center, Department of Electrical Engineering, Ohio University, Athens, Ohio, September, 1976.