COPY 1

DOT-FAA-CD-330-90/05

ACD-330-90-05

[redacted]

APR 17 [illegible]

TECHNICAL CENTER LIBRARY
ATLANTIC CITY, N.J. [illegible]

PROCEDURE FOR CHANGING DATA WORDS FOR THE BENDIX TEST BED MICROWAVE LANDING SYSTEM

Thomas D'Ottavi

FAA Technical Center
Concepts Analysis Division
Airborne Systems Technology Branch

#5 EZ 11/27/89

PROCEDURE FOR CHANGING DATA WORDS
FOR THE BENDIX TEST BED MICROWAVE LANDING SYSTEM

Introduction

The **SITEROMS** program was developed for the BENDIX MLS Test Bed at the FAA Technical Center, because of the different siting configurations at the Tech Center (whenever the system is moved), and the temporary installations at various airports to support the MLS Demo Program. The amount of man hours required to change both the Basic and the Aux Data words manually without errors was at least 8 to 16 hours. With SITEROMS a user can change the Basic/Aux data words in about a half hour or less.

In order to properly monitor the radiated Basic/Aux Data, it is imperative that the Monitor electronics contain the same data words as the Timing and Control. For this reason, when a change is made to a Basic or Auxiliary Data word, both the Timing and Control firmware and the Monitor firmware should be identically changed.

The structure of the software SITEROMS has been designed so that modification of Basic/Aux Data words will be straight-forward with only the data to be entered at the keyboard. The software will create the documentation (ie.. MBJY2A.DOC), the file to be used to program the 8755A EPROMS and the AZ Site RMS 2716 EPROM (ie.. MBJY2A.ROM), and the EL Site RMS 2716 EPROM (ie..MBJY2A.ERM).

Hardware Requirements

SITEROMS is designed to work with the IBM PC, XT, AT, and most IBM-compatible computers. The computer should have at least 512K of memory. Only one disk drive is needed.

To start up SITEROMS, the computer must be booted with version 2.0 (or higher) of DOS.

To program both the 8755A's, and the 2716 EPROMS required to change the site data, the following will be required:

BP Microsystems EPROM Programmer Model EP-1 or equivalent.
BP Microsystems Programming Head Model 1A.
BP Microsystems Programming software for EP-1
Hex File Editor
2 Blank 8755A EPROMS.
2 Blank 2716 EPROMS.

Changing Site Data

When moving a BENDIX MLS station to a new location, it is only necessary to reprogram and replace three SITE DATA EPROMS in the AZ site and one EPROM in the EL site. The EPROMS requiring modification are identified as:

AZ SITE:

REF.	TYPE	CIRCUIT BOARD NAME
A1A1A7-U4	8755A	MONITOR PROCESSOR BOARD
A1A1A3-U7	8755A	TIMING AND CONTROL PROCESSOR BOARD
A1A1A5-U14	2716	RMS BOARD

EL SITE:

REF.	TYPE	CIRCUIT BOARD NAME
A1A1A5-U14	2716	RMS BOARD

The program will prompt the user for:

MLS Site Location: (ie.. FAA TECH CENTER)
This is the data that will be used for the documentation header and the RMS site-id header.

Remarks: (ie.. SYSTEM LOCATED ON RWY. 31)
Enter any remarks the user wishes to appear in the documentation.

Effective Date: (ie.. 3 NOV 86)
Enter the current date of the firmware change.

Format: (1 = FAA-STD-022c 2 = RTCA/DO-198)
This is the document that the program will use for the Basic/Aux data words format.
(Appendix A for FAA-STD-022c)
(Appendix B for RTCA/DO-198)

Output to: (1= Printer 2 = Screen 3 = File)
Select 1, 2, or 3
1 = Printer or 2 = Screen to view the change.
3 = File to create both the documentation and the ROM file on disk.

If File was selected as the output the program will ask for:

Enter File Name: (ie.. MBJY2A)
This is the file name the program will create for both the documentation and the ROM file. The file name may include the path the user wishes the file to be written to.
(Use Table 1-1 for file name)

Table 1-1
GUIDE TO SOFTWARE NOMENCLATURE AND USAGE

The meaning of the software reference is made up of the following parameters:

MBJY2A_1

SITE ID

2 = 2 DEG BEAM WIDTH
1 = 1 DEG BEAM WIDTH

A = AZIMUTH

BLANK = NO REVISIONS
_1 = REVISION ONE (etc.....)

The example above indicates an EPROM used with a 2 DEG AZIMUTH ANTENNA with a site ID of MBJY with 1 revision.

The user will then be asked for the data of both Basic/Aux Data words, one at a time. If BAZ status is 0 (none) in Basic Data Word 2 you will not be asked for input to Basic Data Word 5 or Aux Data Word 4.

All distance inputs will be in meters, and the program will prompt user with choices for other inputs. Numbers can be inputted in real numbers, but will be set for smallest LSB and largest MSB selection for data word.

Programming Site Data EPROMS

After creating the documentation and ROM files with SITEROMS program, the user can now program the 8755A EPROMS with the ROM file and the EPROM programmer.

The following step-by-step instructions should be executed:

1. Plug the EP-1 power cord into a power outlet.Connect EP-1 EPROM Programmer's serial cable to your PC, Turn on both the computer and the EP-1. The EP-1 will perform a ROM test, RAM test, and serial port test when power is applied. The LED's will flash in sequence as these test are performed, leaving just the green power LED on. Install Programming Head 1A on EPROM socket by lifting the lever on the EP-1 socket. Place the head over the socket and press the socket lever down.

NOTE: Parts must not be inserted or removed when the EP-1 ACTIVE LED is on. Parts must be removed before turning the programmer power on or off.

2. Boot computer to 2.0 (or higher) of DOS.
   Insert the EP-1 software diskette into drive A:. If the EP-1 is connected to COM1:, type **EP 1** and press ENTER. If the EP-1 is connected to COM2:, type **EP 2** and press ENTER.
   Screen display:
   EP-1 EPROM Programmer V3.11
   Copyright 1985-1989 BP Microsystems
   >

3. Type **C** <Return> (Chip select).

4. Type **15** <Return> (Intel).

5. Type **36** <Return> (8755A)
   Screen display:
   Chip Type: Intel 8755A *1A
   File Address Range: 0 - 7FF
   8755A *1A>

6. Install a blank 8755A in the Programming Head socket. Identify pin 1 on your chip. It will have a dot or notch near it. Lift the handle on the socket, place the chip in the socket with pin 1 closest to handle. Press the socket lever down.

7. Type **B** <Return> (Blank Check)
   Screen display:
   O.K.
   8755A *1A>

8. Type **P** <Return> (Program from hex file)
   Screen display:
   Hex file to read:

9. Enter the name of the ROM file the user created with the program SITEROMS. ie.. MBJY2A.ROM <Return>(The name will be located at the bottom of the documentation.
   ie.. ROM File Name: MBJY2A.ROM) Be sure to enter the drive and path if the file is not in the default drive or path.
   (ie.. C\PROM\MBJY2A.ROM)
   The EP-1 will now program the 8755A EPROM.
   Screen display when finished:
   800 bytes programmed correctly; No errors.
   8755 *1A>

10. Remove EPROM and repeat steps 6 thur 9 with second 8755A EPROM.

11. Label 8755A EPROMS with file name and effective date.

12. Remove Programming Head 1A from EP-1 socket by lifting EP-1 socket handle and lifting Programming Head 1A off socket.

13. Type **C** <Return> (Chip Select)

14. Type **15** <Return> (Intel)

15. Type **4** <Return> (2716)
    Screen display:
    Chip Type: Intel 2716
    File Address Range: 0 - 7FF
    2716>

16. Install a blank 2716 in the EP-1 socket. Identify pin 1 on your chip. It will have a dot or notch near it. Lift the handle on the socket, place the chip in the socket with pin 1 away from you, and leave four holes empty at the top of the socket. Press the socket lever down.

17. Type **B** <Return> (Blank Check)
    Screen display:
    O.K.
    2716>

18. Type **P** <Return> (Program from hex file)
    Screen display:
    Hex file to read:

19. For AZ RMS EPROM enter file name the user created with the program SITEROMS with the extension of ".ROM", and for the EL RMS EPROM use file with the extension of ".ERM".
(ie.. for AZ use MBJY2A.ROM or for EL use MBJY2A.ERM)
Again be sure to enter the drive and path if the file is not in the default drive or path.
The EP-1 will now program the 2716 EPROM.
Screen display when finished:
800 bytes programmed correctly; No errors.

20. Remove EPROM and be sure to label with file name and mark if AZ or EL.

21. Repeat steps 16 thur 20.

22. Type Q <Return> (Quit)

23. Turn power off both your computer and EP-1 programmer.

Changing Site EPROMS

AZ Site:

Refer to Figure 1-1 for switch and board locations of Control Assembly A1A1.

1. Remove cover from Control Assembly A1A1.

2. LOC/REM control switch to LOC position.

3. ON/OFF control switch to OFF position.

4. Remove RMS Board A1A1A5 from Control Assembly A1A1.

5. Carefully Remove U14 (2716 EPROM) from socket making note of pin 1's location. (see Figure 1-2)

6. Install new 2716 EPROM in U14 socket. Be sure of location of pin 1.

7. Re-install RMS Board A1A1A5 in Control Assembly A1A1.

8. Remove Monitor Processor Board A1A1A3 from Control Assembly A1A1.

9. Carefully Remove U7 (8755A EPROM) from socket making note of pin 1's location. (see Figure 1-3)

10. Install new 8755A EPROM in U7 socket. Be sure of location of pin 1.

11. Re-install Monitor Processor Board A1A1A3 in Control Assembly A1A1.

12. Remove Timing and Control Processor Board A1A1A7 from Control Assembly A1A1.

13. Carefully Remove U4 (8755A EPROM) from socket making note of pin 1's location. (see Figure 1-4)

14. Install new 8755A EPROM in U4 socket. Be sure of location of pin 1.

15. Re-install Timing and Control Processor Board A1A1A7 in Control Assembly A1A1.

16. ON/OFF Control switch to ON.

<u>EL Site:</u>

1. Repeat steps 1 thur 7 above.

2. ON/OFF Control switch to ON.

Data Word Verification

The transmitted data words should be verified with a portable test set, flight inspection data panel, or other means to ensure that the data words are correct.

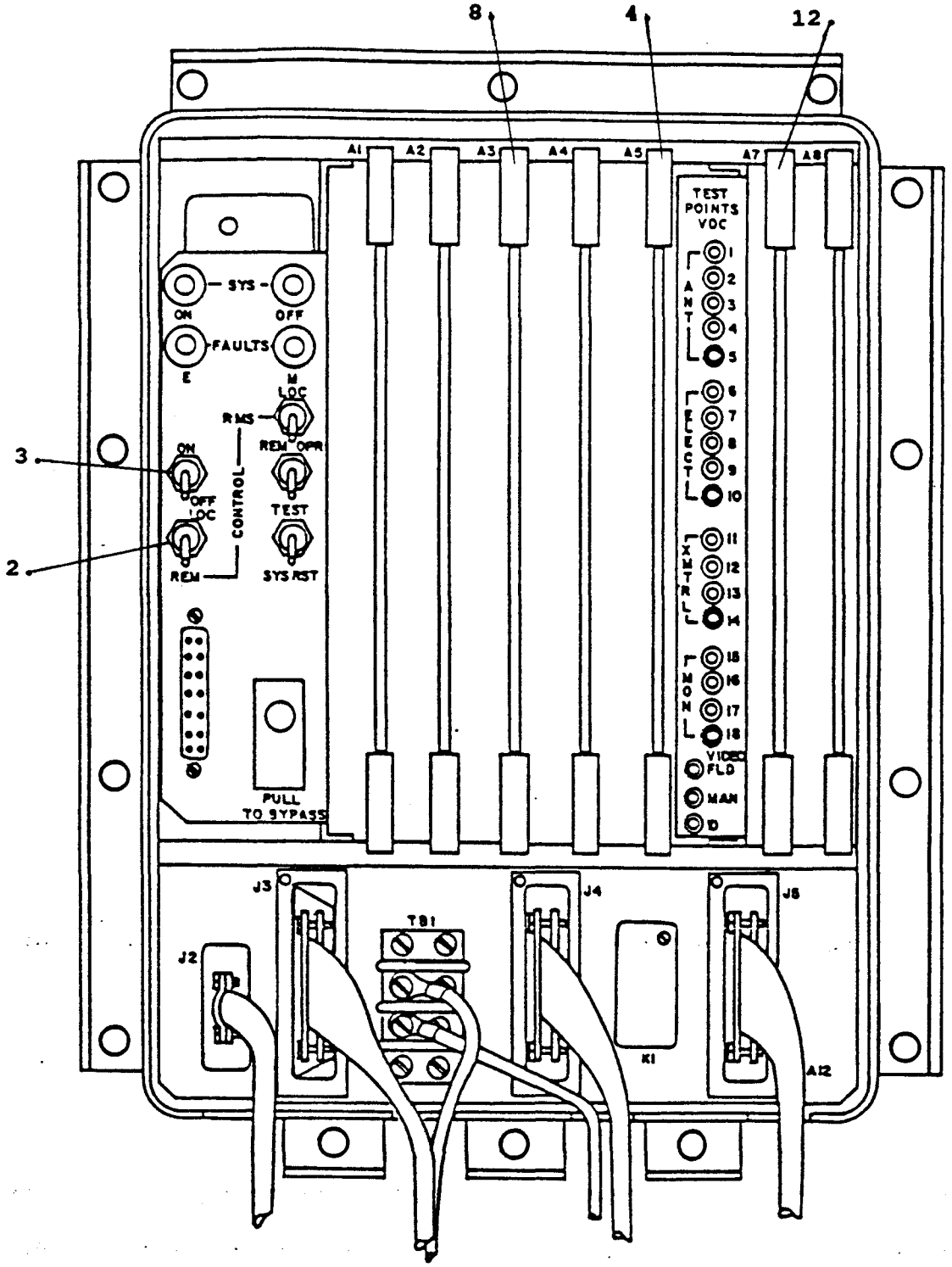


Figure 1-1 Control Assembly A1A1

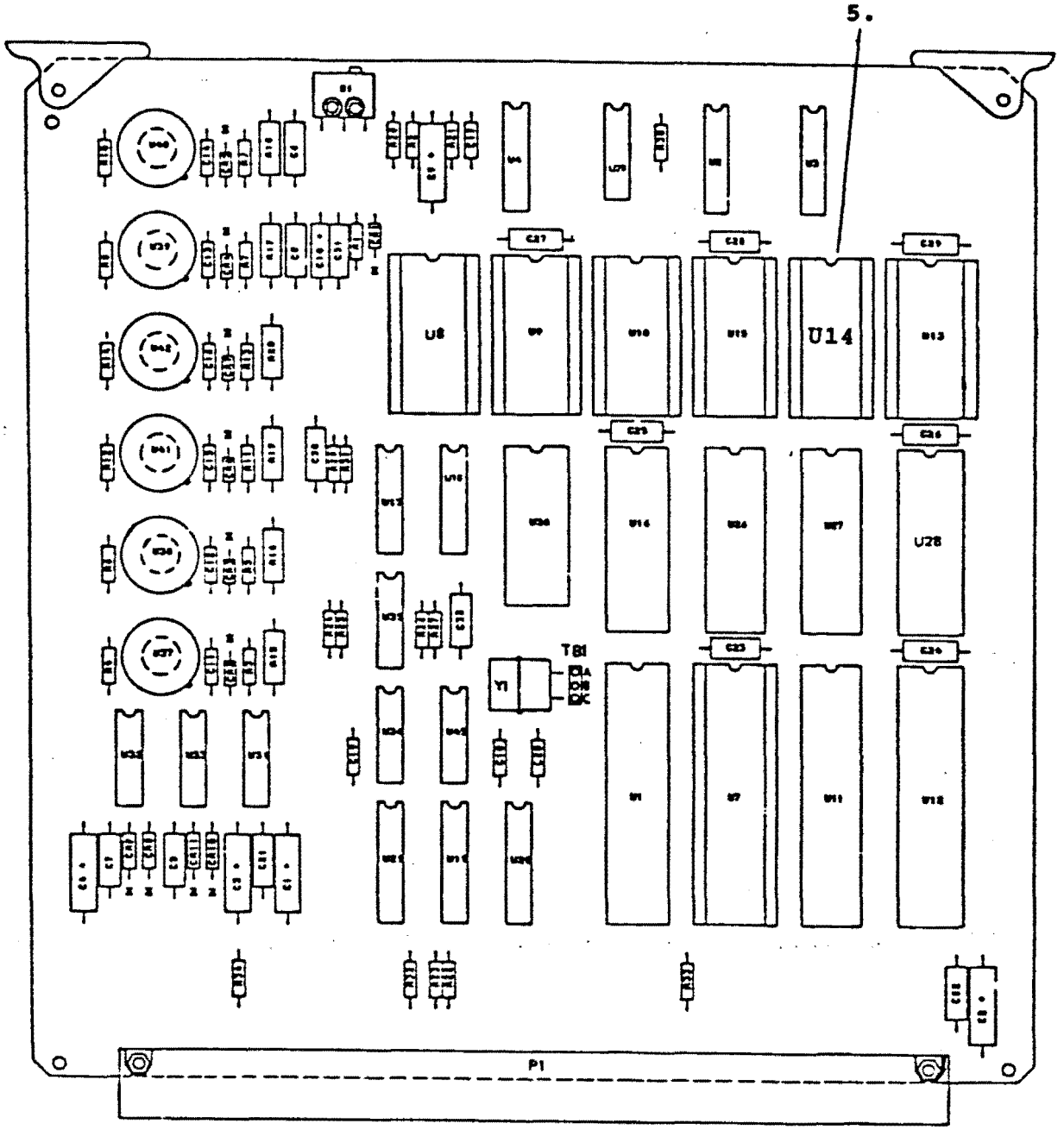


Figure 1-2 RMS Board A1A1A5

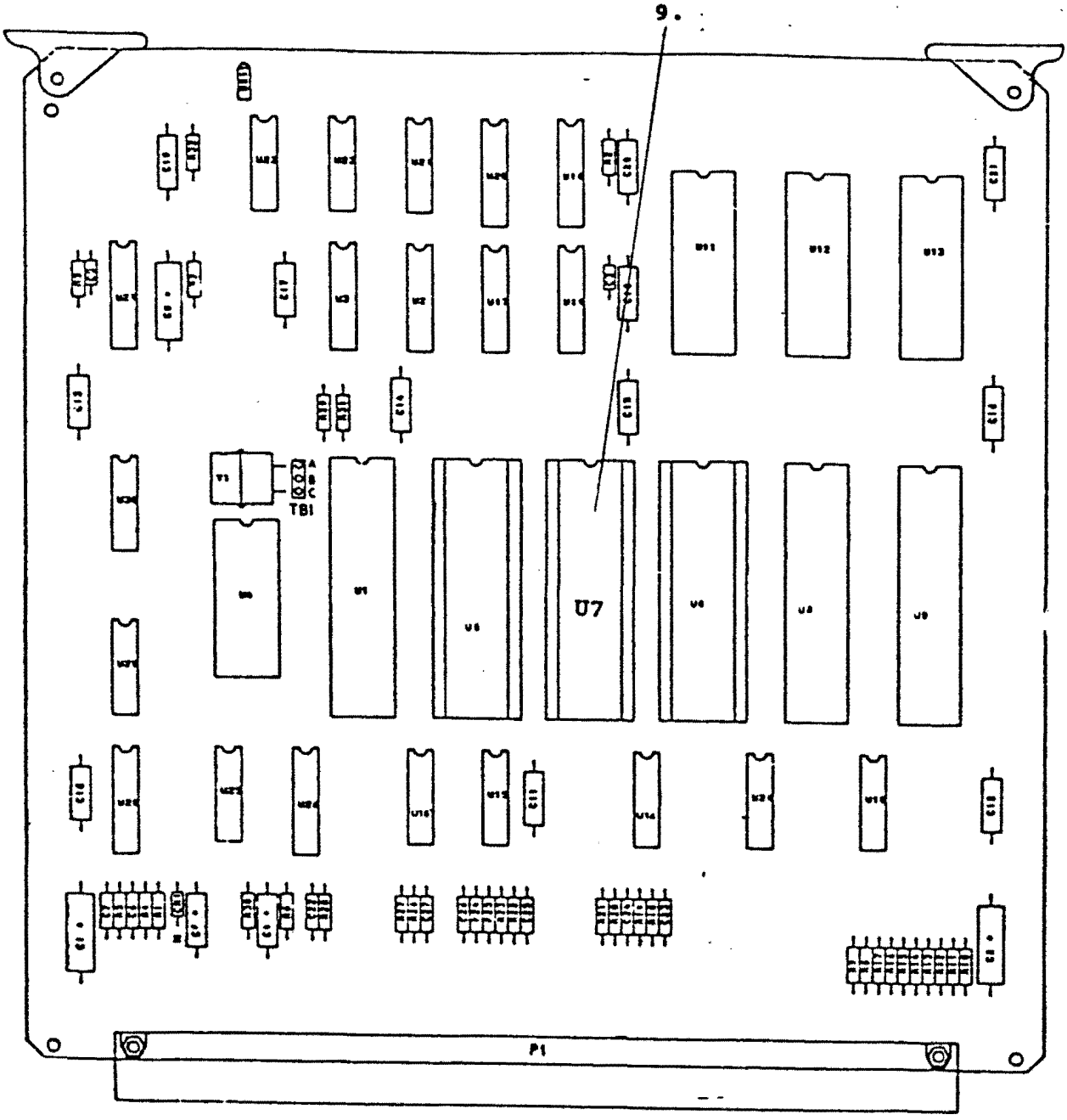


Figure 1-3 Monitor Processor Board

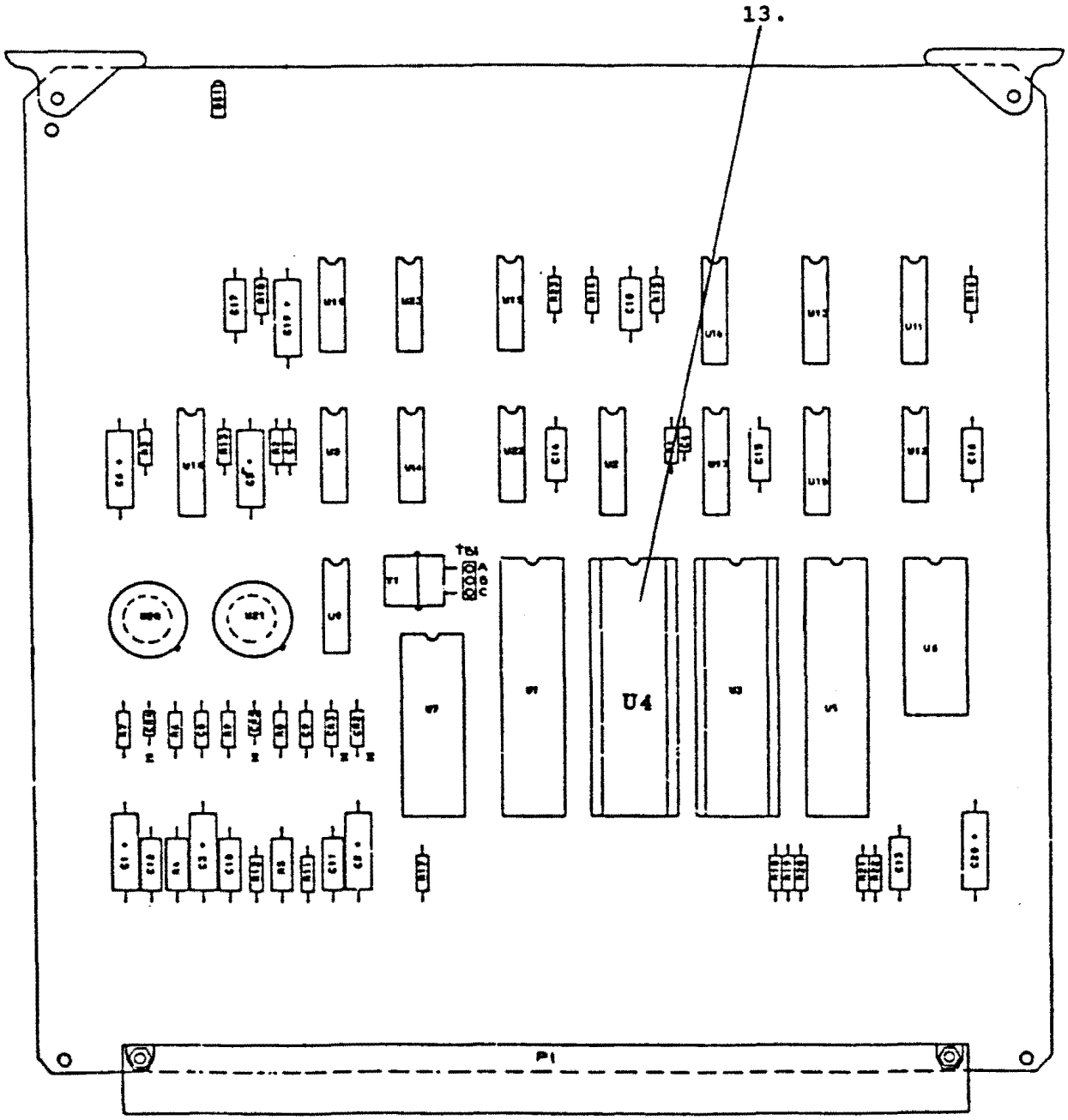


Figure 1-4 Timing and Control Processor Board Assembly A1A1A7

APPENDIX A

BASIC AND AUXILIARY DATA FORMAT

FAA-STD-022c

2.2 Information documents.- The following publications provide information on MLS operational requirements and avionics characteristics.

(a) International Civil Aviation Organization, International Standards and Recommended Practices, Aeronautical Telecommunications, Annex 10.

(b) FAA Handbook 8260.3 United States Standard for Terminal Instrument Procedures (TERPS).

(c) Radio Technical Commission for Aeronautics, Minimum Operational Performance Standards for Microwave Landing System Airborne Receiver Equipment, Doc. No. DO-177.

(d) Radio Technical Commission for Aeronautics, Minimum Operational Performance Standards for Airborne Distance Measuring Equipment (DME) Operating Within the Frequency Range of 960-1215 MHz, Doc. No. DO-189.

3. SYSTEM REQUIREMENTS, ANGLE AND DATA FUNCTIONS

3.1 Radiated signal characteristics

3.1.1 Channeling

3.1.1.1 Channel arrangement.- The MLS angle and data functions shall operate between 5031 MHz and 5090.7 MHz on any one of the 200 channels shown in Table 12, Appendix I.

3.1.1.2 Channel pairing with DME/P.- The channel pairing with the L-band ranging function shall be in accordance with Table 12, Appendix I.

3.2 Signal format organization

3.2.1 Angle guidance functions.- The organization of angle guidance functions shall be as shown in Figure 1. The function shall begin with a preamble time slot followed by time slots for sector signals, and scanning beam signals.

Figure 1. Angle Function Organization

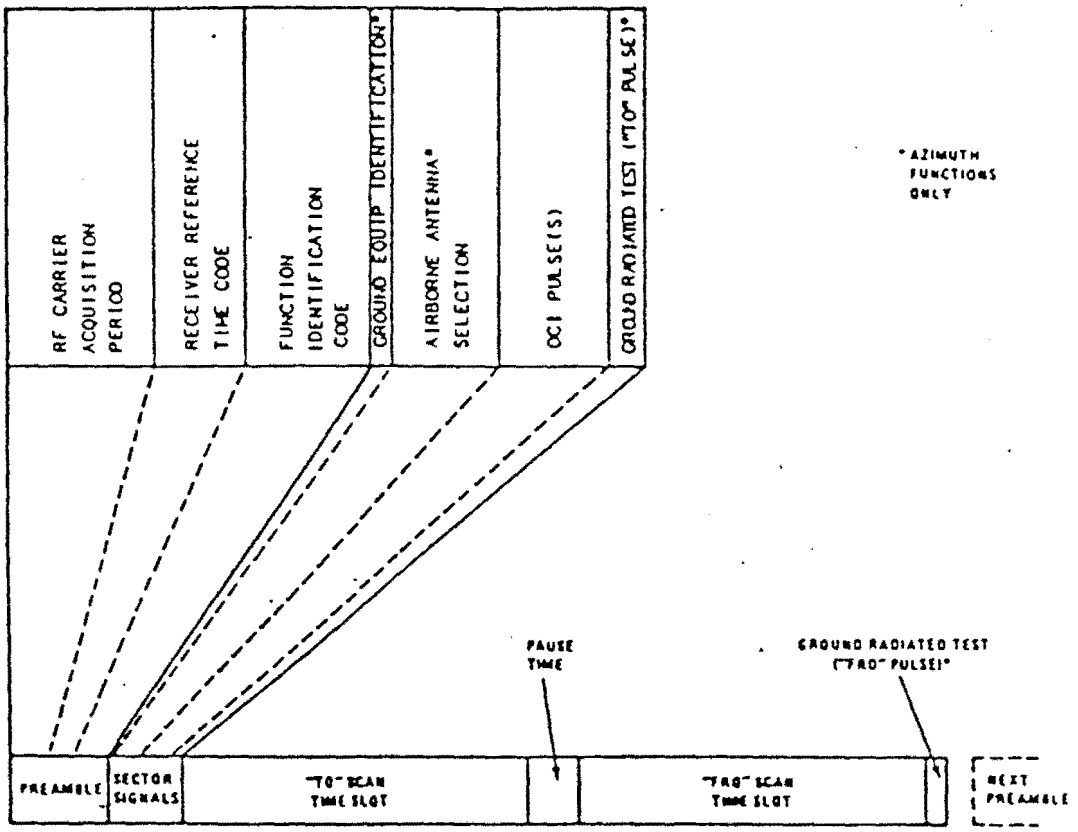
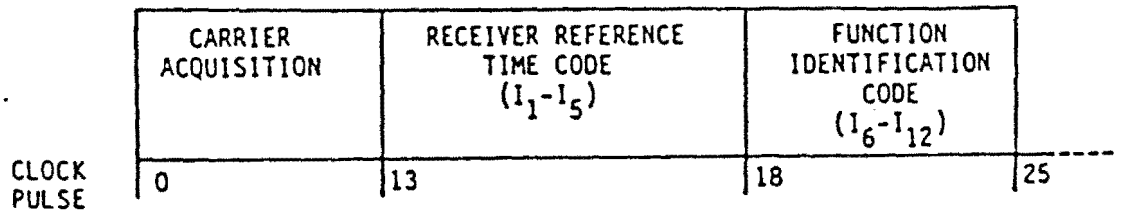


Figure 2. Preamble Organization



3.2.1.1 Preamble.- A preamble signal shall be transmitted throughout the applicable coverage sector to identify the particular function to follow. The preamble shall consist of a radio frequency carrier acquisition period, a receiver reference time code, and a function identification code as shown in Figure 2.

3.2.1.1.1 Carrier acquisition.- The preamble transmission shall begin with a period of unmodulated radio frequency carrier.

3.2.1.1.2 Receiver reference time code.- The receiver reference time code 11101 (bits I_1 to I_5), shall follow the carrier acquisition period. The time of the last phase transition midpoint in the code shall be the receiver reference time.

3.2.1.1.3 Function identification code.- A code for function identification shall follow the receiver reference time code. The code shall consist of the five information bits (I_6 to I_{10}) allowing identification of 31 different functions, plus two parity bits (I_{11} to I_{12}) as shown in Table 1.

Table 1. Function Identification Codes

FUNCTION	CODE I_6	I_7	I_8	I_9	I_{10}	I_{11}	I_{12}
APPROACH AZIMUTH	0	0	1	1	0	0	1
HIGH RATE APPROACH AZIMUTH	0	0	1	0	1	0	0
APPROACH ELEVATION	1	1	0	0	0	0	1
FLARE	0	1	1	0	0	0	1
BACK AZIMUTH	1	0	0	1	0	0	1
360° AZIMUTH	0	1	0	0	1	0	1
BASIC DATA 1	0	1	0	1	0	0	0
BASIC DATA 2	0	1	1	1	1	0	0
BASIC DATA 3	1	0	1	0	0	0	0
BASIC DATA 4	1	0	0	0	1	0	0
BASIC DATA 5	1	1	0	1	1	0	0
BASIC DATA 6	0	0	0	1	1	0	1
AUXILIARY DATA A	1	1	1	0	0	1	0
AUXILIARY DATA B	1	0	1	0	1	1	1
AUXILIARY DATA C	1	1	1	1	0	0	0

NOTE 1: The function identification codes have been chosen so that parity bits I_{11} and I_{12} satisfy the equations:

$$I_6 + I_7 + I_8 + I_9 + I_{10} + I_{11} = \text{EVEN}$$

$$I_6 + I_8 + I_{10} + I_{12} = \text{EVEN}$$

3.2.1.2 Sector signals

3.2.1.2.1 Azimuth functions.- The transmission format of any azimuth function shall include time slots for ground equipment identification, airborne antenna selection, out-of-coverage indication, and test pulses.

3.2.1.2.1.1 Morse Code identification.- The MLS Azimuth Equipment providing services for a particular runway shall be identified by a four character alphabetic designator starting with the letter M. This designator shall be encoded in International Morse Code, and transmitted in nominal 10-second intervals. The code shall be transmitted by the Approach Azimuth and Back Azimuth ground equipment within ±0.08 sec by the use of a DPSK Morse Code bit following the preamble. A "one" shall initiate a Morse Code symbol in the receiver and a "zero" shall terminate the symbol. The dot shall be a time duration of 0.13 to 0.16 second and the dash shall be 0.39 to 0.48 second. The time duration between Morse Code symbols shall be equal to that of one dot plus or minus 10 percent. The time duration between characters shall not be less than three dots. The Morse Code identification technique is shown in Figure 3.

NOTE: The ground equipment identification is also provided as a digital word as listed in Table 6 and transmitted on the data channel alternately between Approach and Back Azimuth coverages.

3.2.1.2.1.2 Airborne antenna selection signal.- A signal for airborne antenna selection shall be transmitted as a "zero" DPSK signal lasting for a six-bit period following the Morse Code bit. The signal shall be available throughout the coverage sector in which Approach or Back Azimuth guidance is provided.

NOTE: The signal provides an opportunity for the selection of the most appropriate antenna in a multiple antenna airborne installation.

3.2.1.2.1.3 Out-of-coverage indication pulses.- In the installed environment, where out-of-coverage indication pulses are used, they shall be: (1) greater than any guidance signal in the out-of-coverage sector, (2) at least 5 dB less than the clearance signal level within the adjacent clearance sector, and (3) at least 5 dB less than the scanning beam signal level within the proportional coverage region. The duration of each pulse measured at the half amplitude point shall be at least 100 microseconds, and the rise and fall times shall be less than 10 microseconds. It shall be permissible to sequentially transmit two pulses in each out-of-coverage indication time slot. Where pulse pairs are used, the duration of each pulse shall be at least 50 microseconds, and the rise and fall times shall be less than 10 microseconds. The transmission of out-of-coverage indication pulses from antennas with overlapping coverage patterns shall be separated by at least 10 microseconds.

Table 5b. Allowable DME/P Standard 1 System Errors (Cont'd)

LOCATION	MODE	PFE	CMN
5 NM to MLS Approach Reference Datum	FA	±85 m (279 ft) Reducing Linearly to ±30 m (100 ft)	18 m (60 ft)
See Note (2)	IA	±100 m (328 ft)	68 m (223 ft)
At MLS Approach Reference Datum and throughout Runway Coverage	FA	±30 m (100 ft)	18 m (60 ft)
5NM to Back Azimuth Reference Datum	IA/FA	±100 m (328 ft)	68 m (223 ft)
20 NM to 5NM From Back Azimuth Reference Datum	IA	±250 m (820 ft) Reducing Linearly to ±100m (328 ft)	68 m (223 ft) Reducing Linearly to 34m (111 ft)

Note: (1) Outside the Approach Azimuth and Back Azimuth coverage sectors the PFE shall not exceed ±375m (1230 ft) and the CMN shall not exceed 68m (223 ft).

(2) At distances from 5 nautical miles to the MLS approach reference datum and throughout the back azimuth coverage, the IA mode may be used when the FA mode is not operative.

3.6 Basic Data content.- Basic Data shall consist of the items specified in Table 6. Basic Data words shall be defined as follows:

(a) Approach azimuth to threshold distance shall represent the minimum distance between the approach azimuth antenna phase center and the vertical plane perpendicular to the centerline which contains the runway landing threshold.

(b) Approach azimuth proportional coverage limit shall represent the limit of the sector in which proportional Approach Azimuth guidance is transmitted.

NOTE: This limit does not represent the angle of the maximum flyable azimuth course. That will always be an angle less than the proportional limit and should be determined by flight inspection.

(c) Clearance signal type shall indicate the method of providing the azimuth clearance signal.

(d) Minimum glide path shall represent the minimum glide path as defined in 1.5.20.

(e) Back azimuth status shall represent the operational status of the Back Azimuth equipment.

(f) DME status shall represent the operational status of the DME equipment.

(g) Approach azimuth status shall represent the operational status of the Approach Azimuth equipment.

(h) Approach elevation status shall represent the operational status of the Approach Elevation equipment.

(i) Beamwidth shall represent, for a particular function, the antenna beamwidth as defined in 1.5.7.

(j) DME distance shall represent the minimum distance between the DME antenna phase center and the vertical plane perpendicular to the runway center line which contains the MLS datum point.

(k) Approach azimuth magnetic orientation shall represent the angle measured in the horizontal plane clockwise from Magnetic North to the zero-degree angle guidance radial of the Approach Azimuth antenna. The vertex of the measured angle shall be at the Approach Azimuth antenna phase center.

NOTE: For example, this data item would be encoded 090 for an approach Azimuth antenna serving runway 27, (assuming the magnetic heading is 270 degrees), when sited such that the zero degree radial is parallel to centerline.

(l) Back azimuth magnetic orientation shall represent the angle measured in the horizontal plane clockwise from Magnetic North to the zero-degree angle guidance radial of the Back Azimuth antenna. The vertex of the measured angle shall be at the Back Azimuth antenna phase center.

NOTE: For example, this data item would be encoded 270 for a Back Azimuth Antenna serving runway 27, (assuming the magnetic heading is 270 degrees), when sited such that the zero degree radial is parallel to centerline.

(m) Back azimuth proportional coverage limit shall represent the limit of the sector in which proportional Back Azimuth guidance is transmitted. The angle of the maximum flyable back azimuth course. That will always be an angle less than the proportional limit and should be determined by flight inspection.

(n) MLS ground equipment identification shall represent the last three characters of the system identification specified in 3.2.1.2.1.1. The characters shall be encoded in accordance with the 5-unit code of the International Alphabet No. 5 (IA-5) using bits B_1 through B_6.

NOTE: Bit B_7 of this code may be reconstructed in the airborne receiver by taking the complement of bit B_6.

Table 6. Basic Data Words

BASIC DATA WORD #1

Note 1: Transmit throughout the Approach Azimuth guidance sector at intervals of 1.0 second or less.

Note 2: The all zero state of the data field represents the lower limit of the absolute value of the coded parameter unless otherwise noted.

DATA BIT #	DATA ITEM DEFINITION	LSB VALUE	DATA BIT VALUE
1	Preamble	N/A	1
2			1
3			1
4			0
5			1
6			0
7			1
8			0
9			1
10			0
11			0
12			0
13	Approach azimuth to threshold distance (0m - 6300m)	100m	100m
14			200m
15			400m
16			800m
17			1600m
18			3200m
19	Approach azimuth proportional coverage limit (negative limit) (0° to -62°)	2°	-2°
20			-4°
21			-8°
22			-16°
23			-32°
24	Approach azimuth proportional coverage limit (positive limit) (0° to +62°)	2°	2°
25			4°
26			8°
27			16°
28			32°
29	Clearance signal type	N/A	0 = pulse; 1 = SB
30	Spare		Transmit zero
31	Parity: (13+14+15...+30+31 = 000)	N/A	N/A
32	Parity: (14+16+18...+30+32 = 009)	N/A	N/A

Table 6. Basic Data Words (cont)

BASIC DATA WORD #2

Note 1: Transmit throughout the Approach Azimuth guidance sector at intervals of 0.16 second or less.

Note 2: The all zero state of the data field represents the lower limit of the absolute range of the coded parameter unless otherwise noted.

DATA BIT #	DATA ITEM DEFINITION	LSB VALUE	DATA BIT VALUE
1	Preamble	N/A	1
2			1
3			1
4			0
5			1
6			0
7			1
8			1
9			1
10			1
11			0
12			0
13	Minimum glide path (2.0° to 14.7°)	0.1°	0.1°
14			0.2°
15			0.4°
16			0.8°
17			1.6°
18			3.2°
19			6.4°
20	Back azimuth status		see note 4
21	DME status		see note 6
22			see note 4
23	Approach azimuth status		see note 4
24	Approach elevation status		Transmit zero
25	Spare		Transmit zero
26	Spare		Transmit zero
27	Spare		Transmit zero
28	Spare		Transmit zero
29	Spare		Transmit zero
30	Spare		Transmit zero
31	Parity: (13+14+15...+30+31 = 000)	N/A	N/A
32	Parity: (14+16+18...+30+32 = 000)	N/A	N/A

Table 6. Basic Data Words (cont.)

BASIC DATA WORD #3

Note 1: Transmit throughout the Approach Azimuth guidance sector at intervals of 1.0 second or less.

Note 2: The all zero state of the data field represents the lower limit of the absolute range of the coded parameter unless otherwise noted.

DATA BIT #	DATA ITEM DEFINITION	LSB VALUE	DATA BIT VALUE
1	Preamble	N/A	1
2			1
3			1
4			0
5			1
6			1
7			0
8			1
9			0
10			0
11			0
12			0
13	Approach azimuth beamwidth	0.5°	0.5°
14	(0.5° - 4.0°) See note 7		1.0°
15			2.0°
16	Approach elevation beamwidth	0.5°	0.5°
17	(0.5° to 2.5°) See note 7		1.0°
18	Note: values greater than 2.5° are invalid		2.0°
19	DME distance	12.5m	12.5m
20	(0m to 6387.5m)		25.0m
21			50.0m
22			100.0m
23			200.0m
24			400.0m
25			800.0m
26			1600.0m
27			3200.0m
28	Spare		Transmit zero
29	Spare		Transmit zero
30	Spare		Transmit zero
31	Parity: (13+14+15...+30+31 = 000)	N/A	N/A
32	Parity: (14+16+18...+30+32 = 000)		

Table 6. Basic Data Words (cont.)

BASIC DATA WORD #4

Note 1: Transmit at intervals of 1.0 second or less throughout the Approach Azimuth guidance sector, except when Back Azimuth guidance is provided. See Note 8.

Note 2: The all zero state of the data field represents the lower limit of the absolute range of the coded parameter unless otherwise noted.

DATA BIT #	DATA ITEM DEFINITION	LSB VALUE	DATA BIT VALUE
1	Preamble	N/A	1
2			1
3			1
4			0
5			1
6			1
7			0
8			0
9			0
10			1
11			0
12			0
13	Approach azimuth magnetic orientation (0° to 359°)	1°	1°
14			2°
15			4°
16			8°
17			16°
18			32°
19			64°
20			128°
21			256°
22	Back azimuth magnetic orientation (0° to 359°)	1°	1°
23			2°
24			4°
25			8°
26			16°
27			32°
28			64°
29			128°
30			256°
31	Parity: (13+14+15...+30+31 = odd)	N/A	N/A
32	Parity: (14+16+18...+30+32 = odd)	N/A	N/A

Table 6. Basic Data Words (cont.)

BASIC DATA WORD #5

Note 1: Transmit only when Back Azimuth guidance is provided. See note 9.

Note 2: The all zero state of the data field represents the lower limit of the absolute range of the coded parameter unless otherwise noted.

DATA BIT #	DATA ITEM DEFINITION	LSB VALUE	DATA BIT VALUE
1	Preamble	N/A	1
2			1
3			1
4			0
5			1
6			1
7			1
8			0
9			1
10			1
11			0
12			0
13	Back azimuth proportional	2°	-2°
14	coverage negative limit		-4°
15	(0° to -42°)		-8°
16			-16°
17			-32°
18	Back azimuth proportional	2°	2°
19	coverage positive limit		4°
20	(0° to +42°)		8°
21			16°
22			32°
23	Back azimuth beamwidth	0.5°	0.5°
24	(0.5° to 4.0°) See note 7		1.0°
25			2.0°
26	Spare		Transmit zero
27	Spare		Transmit zero
28	Spare		Transmit zero
29	Spare		Transmit zero
30	Spare		Transmit zero
31	Parity: (13+14+15...+30+31 = 000)	N/A	N/A
32	Parity: (14+16+18...+30+32 - 000)	N/A	N/A

Table 6. Basic Data Words (cont.)

BASIC DATA WORD #6

Note 1: Transmit at intervals of 1.0 second or less throughout the Approach Azimuth guidance sector, except when Back Azimuth guidance is provided. See note 8.

DATA BIT #	DATA ITEM DEFINITION	LSB VALUE	DATA BIT VALUE
1	Preamble	N/A	1
2			1
3			1
4			0
5			1
6			0
7			0
8			0
9			1
10			1
11			0
12			1
(13 - 30)	MLS ground equipment identification (Note 3)		
13	Character 2	N/A	B1
14			B2
15			B3
16			B4
17			B5
18			B6
19	Character 3	N/A	B1
20			B2
21			B3
22			B4
23			B5
24			B6
25	Character 4	N/A	B1
26			B2
27			B3
28			B4
29			B5
30			B6
31	Parity: (13+14+15...+30+31 = 0 00)	N/A	N/A
32	Parity: (14+16+18...+30+32 = 0 00)	N/A	N/A

Table 6. Basic Data Words (cont)

NOTE 3: Characters are encoded using the International Alphabet Number 5 (IA-5), bits B_1 through B_6.

NOTE 4: Coding for status bit:

0 = Function not radiated, or radiated in test maintenance (not reliable for navigation).
1 = Function radiated in normal mode (for Back Azimuth, this also indicates that a Back Azimuth transmission follows).

NOTE 5: Data items which are not applicable to a particular ground equipment shall be transmitted as all zeros.

NOTE 6: Coding for status bits:

I_{21}	I_{22}	
0	0	DME transponder inoperative or not available.
1	0	Only IA mode or DME/N available.
0	1	FA mode, Standard 1, available.
1	1	FA mode, Standard 2, available.

NOTE 7: The value coded shall be the actual beamwidth (as defined in 1.5.7) rounded to the nearest 0.5°.

NOTE 8: When Back Azimuth guidance is provided, Data Words 4 and 6 shall be transmitted at intervals of 1.33 seconds or less throughout the Approach Azimuth coverage and 4 seconds or less throughout the Back Azimuth coverage.

NOTE 9: When Back Azimuth guidance is provided, Data Word 5 shall be transmitted at an interval of 1.33 seconds or less throughout the Back Azimuth coverage sector and 4 seconds or less throughout the Approach Azimuth coverage sector.

3.7 Auxiliary data.

3.7.1 Addresses.- The address codes of the auxiliary data words shall be as shown in Table 7.

3.7.2 Content.- Three function identification codes are reserved to indicate transmission of Auxiliary Data A, Auxiliary Data B and Auxiliary Data C. Auxiliary Data A contents that have been defined are specified below, Auxiliary Data B contents are reserved for future use, and Auxiliary Data C contents are reserved for national use.

3.7.2.1 Auxiliary data A content.- The data items specified in Table 8 are defined as follows:

(a) Approach azimuth antenna offset shall represent the minimum distance between the Approach Azimuth antenna phase center and the vertical plane containing the runway centerline.

(b) Approach azimuth to MLS datum point distance shall represent the minimum distance between the Approach Azimuth antenna phase center and the vertical plane perpendicular to the centerline which contains the MLS datum point.

(c) Approach azimuth alignment with runway center line shall represent the minimum angle between the Approach Azimuth antenna zero-degree guidance plane and the runway center line.

(d) Approach azimuth antenna coordinate system shall represent the coordinate system (planar or conical) of the angle data transmitted by the Approach Azimuth antenna.

(e) Approach elevation antenna offset shall represent the minimum distance between the Elevation antenna phase center and the vertical plane containing the runway centerline.

(f) MLS datum point to threshold distance shall represent the distance measured along the runway centerline from the MLS datum point to the runway threshold.

(g) Approach elevation antenna height shall represent the height of the Elevation antenna phase center relative to the height of the MLS datum point.

(h) DME offset shall represent the minimum distance between the DME antenna phase center and the vertical plane containing the runway centerline.

(i) DME to MLS datum point distance shall represent the minimum distance between the DME antenna phase center and the vertical plane perpendicular to the centerline which contains the MLS datum point.

(j) Back azimuth antenna offset shall represent the minimum distance between the Back Azimuth antenna phase center and the vertical plane containing the runway centerline.

(k) Back azimuth to MLS datum point distance shall represent the minimum distance between the Back Azimuth antenna and the vertical plane perpendicular to the centerline which contains the MLS datum point.

(l) Back azimuth antenna alignment with runway centerline shall represent the minimum angle between the Back Azimuth antenna zero-degree guidance plane and the runway centerline.

3.7.3 (Not used)

Table 8. Auxiliary Data

Word (See Note 6)	Data Content	Type of Data	Maximum Time Between Transmissions (Seconds)	Bits Used	Range Of Values	Least Significant Bit
A1	Preamble	digital	1.0	12		
	Address			8		
	Approach azimuth antenna offset			10	-511 m to +511m (see Note 3)	1 m
	Approach azimuth to MLS datum point distance			13	0 m to 8 191 m	1 m
	Approach azimuth antenna alignment with runway centerline			12	-20.47° to 20.47° (see Note 3)	0.01°
	Approach azimuth antenna coordinate system			1	(see Note 2)	
	Spare			13		
	Parity			7	(see Note 1)	
A2	Preamble	digital	1.0	12		
	Address			8		
	Approach elevation antenna offset			10	-511 m to +511 m (See Note 3)	1 m
	MLS datum point to threshold distance			10	0 m to 1 023 m	1 m
	Approach elevation antenna height			7	-6.3 m to +6.3 m (See Note 3)	0.1 m

Table 8. Auxiliary Data (Cont'd).

Word (See Note 6)	Data Content	Type of Data	Maximum Time Between Transmissions (Seconds)	Bits Used	Range Of Values	Least Significant Bit
A2 cont.	Spare			22		
	Parity			7	(See Note 1)	
A3	Preamble	digital (See Note 4)		12		
	Address			8		
	DME offset			10	-511 m to +511 m	1 m
	DME to MLS datum point distance			14	-8 191 m to +8 191 m (See Note 3)	1 m
	Spare			25		
	Parity			7	(See Note 1)	
A4	Preamble	digital (See Note 5)		12		
	Address			8		
	Back azimuth antenna			10	-511 m to +511 m (See Note 3)	1 m
	Back azimuth to MLS datum point distance			11	0 m to 2 047 m	1 m
	Back azimuth antenna alignment with runway centerline			12	-20.47° to 20.47° (See Note 3)	0.01°
	Spare			16		
	Parity			7	(See Note 1)	

Table 8. Auxiliary Data (Cont'd)

NOTE 2: Code for 56 is: 0 = conical
1 = planar

NOTE 3: The conv ion for the coding of negative numbers is as follows:

-MSB is t sign bit

0 = +
1 = -

- Other s represent the absolute value.

The convention fo he antenna location is as follows:

As viewed m the MLS approach reference datum looking toward the datum point, a pos ive number shall represent a location to the right of the runway cent line (lateral offset) or above the runway (vertical offset), or toward the stop end of the runway (longitudinal distance).

The convention for he antenna alignment is as follows:

As viewed f m above, a positive number shall represent clockwise rotation fr the runway centerline to the respective zero-degree guidance pl e.

NOTE 4: Data Word 3 is transmitted at intervals of 1.0 seconds or less throughou the approach Azimuth coverage sector, except when back Azimuth g dance is provided. Where back Azimuth is provided transmit at interv s of 1.33 seconds or less throughout the approach Azimuth sector an 4.0 seconds or less throughout the back Azimuth coverage sector.

NOTE 5: When back zimuth guidance is provided, transmit at intervals of 1.33 seco s or less throughout the back Azimuth coverage sector and 4.0 secon or less throughout the approach Azimuth coverage sector.

NOTE 6: The desig tion "A1" represents the function identification code for "Auxi ary Data A" and address code number 1.

4. GROUND EQUIPME REQUIREMENTS, ANGLE AND DATA FUNCTIONS

4.1 Function tran issions

NOTE: Requirements or the DME/P are contained in FAA-E-2721/3.

4.1.1 Function ti ng.- Event time slots for each angle and data function shall be as specif d in Tables: 14, 15, 16, 17, 18, and 19, Appendix II. The

APPENDIX B

BASIC AND AUXILIARY DATA FORMAT

RTCA/DO-198

This appendix presents a technical description of the MLS basic and auxiliary data words. The figures used in this appendix are extracted from the format proposed for incorporation into ICAO Annex 10, Volume I.

The information transmitted on the MLS, angle and data functions, are time shared on a single channel. Each function starts with a preamble that identifies that particular function, then the angle or data information is transmitted.

The preamble consists of three items. First there is a 0.832 millisecond period of unmodulated radio frequency carrier. Next is a five bit code to provide the receiver reference time code. These bits are labeled I_1 to I_5. The third part of the preamble uses bits I_6 through I_{10} to identify the function. Bits I_{11} and I_{12} provide parity checks for the identification codes. Table C-1 shows the identification codes for all of the angle and data functions.

TABLE C-1 FUNCTION IDENTIFICATION

Function	CODE I_6	I_7	I_8	I_9	I_{10}	I_{11}	I_{12}
Approach azimuth	0	0	1	1	0	0	1
High rate approach azimuth	0	0	1	0	1	0	0
Approach elevation	1	1	0	0	0	0	1
Flare elevation	0	1	1	0	0	0	1
Back azimuth	1	0	0	1	0	0	1
360 degree azimuth	0	1	0	0	1	0	1
Basic data 1	0	1	0	1	0	0	0
Basic data 2	0	1	1	1	1	0	0
Basic data 3	1	0	1	0	0	0	0
Basic data 4	1	0	0	0	1	0	0
Basic data 5	1	1	0	1	1	0	0
Basic data 6	0	0	0	1	1	0	1
Auxiliary data A	1	1	1	0	0	1	0
Auxiliary data B	1	0	1	0	1	1	1
Auxiliary data C	1	1	1	1	0	0	0

NOTE 1: The function identification codes have been chosen so that parity bits I_{11} and I_{12} satisfy the equations:

$$I_6 + I_7 + I_8 + I_9 + I_{10} + I_{11} = \text{EVEN}$$
$$I_6 + I_8 + I_{10} + I_{12} = \text{EVEN}$$

1.0 Basic Data

Basic data shall be encoded as 32-bit words consisting of a function preamble (12 bits) specified in Section 2.0, and data content as specified in Table C-2. Data containing digital information shall be transmitted with the least significant bit

first. The smallest binary number shall represent the lower absolute range limit with increments in binary steps to the upper absolute range limit specified in the tables.

Basic data shall consist of the items specified in Table C-2. Basic data word contents shall be defined as follows:

a. Approach azimuth to threshold distance shall represent the minimum distance between the approach azimuth antenna phase center and the vertical plane perpendicular to the centerline that contains the landing threshold.

b. Approach azimuth proportional coverage limit shall represent the limit of the sector in which proportional approach azimuth guidance is transmitted.

c. Clearance signal type shall indicate the method of providing the azimuth clearance signal.

d. Minimum glide path - The lowest angle of descent along the zero degree azimuth that is consistent with published approach procedures and obstacle clearance criteria.

e. Back azimuth status shall represent the operational status of the back azimuth equipment.

f. DME status shall represent the operational status of the DME equipment.

g. Approach azimuth status shall represent the operational status of the approach azimuth equipment.

h. Approach elevation status shall represent the operational status of the approach elevation equipment.

i. Beam width - The width of the scanning beam main lobe measured at the -3 dB points and defined in angular units on the antenna boresight, in the horizontal plane for the azimuth function and in the vertical plane for the elevation function.

j. DME distance shall represent the minimum distance between the DME antenna phase center and the vertical plane perpendicular to the runway centerline that contains the MLS datum point.

k. Approach azimuth magnetic orientation shall represent the angle measured in the horizontal plane clockwise from magnetic north to the zero-degree angle guidance radial originating

from the approach azimuth antenna phase center. The vertex of the measured angle shall be at the approach azimuth antenna phase center.

NOTE: For example, this data item would be encoded 090 for an approach azimuth antenna serving Runway 27 (assuming the magnetic heading is 270 degrees) when sited so that the zero degree radial is parallel to centerline.

l. Back azimuth magnetic orientation shall represent the angle measured in the horizontal plane clockwise from magnetic north to the zero-degree angle guidance radial originating from the back azimuth antenna. The vertex of the measured angle shall be at the back azimuth antenna phase center.

NOTE: For example, this data item would be encoded 270 for a back azimuth antenna serving Runway 27 (assuming the magnetic heading is 270 degrees) when sited so that the zero-degree radial is parallel to centerline.

m. Back azimuth proportional coverage limit shall represent the limit of the sector in which proportional back azimuth guidance is transmitted.

n. MLS ground equipment identification shall represent the last three characters of the system identification. The characters shall be encoded in accordance with International Alphabet No. 5 (IA-5) using bits b_1 through b_6.

NOTE: Bit b_7 of this code may be reconstructed in the airborne receiver by taking the complement of bit b_6.

TABLE C-2 BASIC DATA WORD #1

NOTE 1: Transmit throughout the approach azimuth guidance sector at intervals of 1.0 second or less.

NOTE 2: The all-zero state of the data field represents the lower limit of the absolute value of the coded parameter unless otherwise noted.

DATA BIT #	DATA ITEM DEFINITION	LSB VALUE	DATA BIT VALUE
1	Preamble (See Note 9)	N/A	1
2			1
3			1
4			0
5			1
6			0
7			1
8			0
9			1
10			0
11			0
12			0
13	Approach azimuth to threshold distance	100 m	100 m
14	(0 m to 6300 m)		200 m
15			400 m
16			800 m
17			1600 m
18			3200 m
19	Approach azimuth proportional coverage	2°	-2°
20	limit (negative limit)		-4°
21	(0° to -62°)		-8°
22			-16°
23			-32°
24	Approach azimuth proportional coverage	2°	2°
25	limit (positive limit)		4°
26	(0° to +62°)		8°
27			16°
28			32°
29	Clearance signal type	N/A	0 = pulse; 1 = SB
30	Spare		Transmit zero
31	Parity: (13+14+15+...+30+31 = ODD)	N/A	N/A
32	Parity: (14+16+18+...+30+32 = ODD)	N/A	N/A

TABLE C-2 BASIC DATA WORD #2

NOTE 1: Transmit throughout the approach azimuth guidance sector at intervals of 0.16 second or less.

NOTE 2: The all-zero state of the data field represents the lower limit of the absolute range of the coded parameter unless otherwise noted.

DATA BIT #	DATA ITEM DEFINITION	LSB VALUE	DATA BIT VALUE
1	Preamble (See Note 9)	N/A	1
2			1
3			1
4			0
5			1
6			0
7			1
8			1
9			1
10			1
11			0
12			0
13	Minimum glide path	0.1°	0.1°
14	(2.0° to 14.7°)		0.2°
15			0.4°
16			0.8°
17			1.6°
18			3.2°
19			6.4°
20	Back azimuth status		(see Note 4)
21	DME status		(see Note 5)
22			
23	Approach azimuth status		(see Note 4)
24	Approach elevation status		(see Note 4)
25	Spare		Transmit zero
26	Spare		Transmit zero
27	Spare		Transmit zero
28	Spare		Transmit zero
29	Spare		Transmit zero
30	Spare		Transmit zero
31	Parity: (13+14+15+...+30+31 = ODD)	N/A	N/A
32	Parity: (14+16+18+...+30+32 = ODD)	N/A	N/A

TABLE C-2 BASIC DATA WORD #3

NOTE 1: Transmit throughout the approach azimuth guidance sector at intervals of 1.0 second or less.

NOTE 2: The all-zero state of the data field represents the lower limit of the absolute range of the coded parameter unless otherwise noted.

DATA BIT #	DATA ITEM DEFINITION	LSB VALUE	DATA BIT VALUE
1	Preamble (See Note 9)	N/A	1
2			1
3			1
4			0
5			1
6			1
7			0
8			1
9			0
10			0
11			0
12			0
13	Approach azimuth beamwidth	0.5°	0.5°
14	(0.5° to 4.0°) (see Note 6)		1.0°
15			2.0°
16	Approach elevation beamwidth	0.5°	0.5°
17	(0.5° to 2.5°) (see Note 6)		1.0°
18	NOTE: Values greater than 2.5° are invalid		2.0°
19	DME distance	12.5m	12.5m
20	(0m to 6387.5m)		25.0m
21			50.0m
22			100.0m
23			200.0m
24			400.0m
25			800.0m
26			1600.0m
27			3200.0m
28	Spare		Transmit zero
29	Spare		Transmit zero
30	Spare		Transmit zero
31	Parity: (13+14+15+...+30+31 = ODD)	N/A	N/A
32	Parity: (14+16+18+...+30+32 = ODD)	N/A	N/A

TABLE C-2 BASIC DATA WORD #4

NOTE 1: Transmit at intervals of 1.0 second or less throughout the approach azimuth guidance sector, except where back azimuth guidance is provided. See Note 7.

NOTE 2: The all-zero state of the data field represents the lower limit of the absolute range of the coded parameter unless otherwise noted.

DATA BIT #	DATA ITEM DEFINITION	LSB VALUE	DATA BIT VALUE
1	Preamble (See Note 9)	N/A	1
2			1
3			1
4			0
5			1
6			1
7			0
8			0
9			0
10			1
11			0
12			0
13	Approach azimuth magnetic	1°	1°
14	orientation (0° to 359°)		2°
15			4°
16			8°
17			16°
18			32°
19			64°
20			128°
21			256°
22	Back azimuth magnetic	1°	1°
23	orientation (0° to 359°)		2°
24			4°
25			8°
26			16°
27			32°
28			64°
29			128°
30			256°
31	Parity: (13+14+15+...+30+31 = ODD)	N/A	N/A
32	Parity: (14+16+18+...+30+32 = ODD)	N/A	N/A

TABLE C-2 BASIC DATA WORD #5

NOTE 1: Transmit only where back azimuth guidance is provided. See Note 8.

NOTE 2: The all-zero state of the data field represents the lower limit of the absolute range of the coded parameter unless otherwise noted.

DATA BIT #	DATA ITEM DEFINITION	LSB VALUE	DATA BIT VALUE
1	Preamble (See Note 9)	N/A	1
2			1
3			1
4			0
5			1
6			1
7			1
8			0
9			1
10			1
11			0
12			0
13	Back azimuth proportional coverage	2°	-2°
14	negative limit		-4°
15	(0° to -42°)		-8°
16			-16°
17			-32°
18	Back azimuth proportional coverage	2°	2°
19	positive limit		4°
20	(0° to +42°)		8°
21			16°
22			32°
23	Back azimuth beamwidth	0.5°	0.5°
24	(0.5° to 4.0°) (see Note 7)		1.0°
25			2.0°
26	Back azimuth status		(see Note 4)
27	Spare		Transmit zero
28	Spare		Transmit zero
29	Spare		Transmit zero
30	Spare		Transmit zero
31	Parity: (13+14+15+...+30+31 = ODD)	N/A	N/A
32	Parity: (14+16+18+...+30+32 = ODD)	N/A	N/A

TABLE C-2 BASIC DATA WORD #6

NOTE: Transmit at intervals of 1.0 second or less throughout the approach azimuth guidance sector, except where back azimuth guidance is provided. See Note 7.

DATA BIT #	DATA ITEM DEFINITION	LSB VALUE	DATA BIT VALUE
1	Preamble (see Note 9)	N/A	1
2			1
3			1
4			0
5			1
6			0
7			0
8			0
9			1
10			1
11			0
12			1
(13-30)	MLS ground equipment identification (see Note 3)		
13	Character 2		B1
14			B2
15			B3
16			B4
17			B5
18			B6
19	Character 3		B1
20			B2
21			B3
22			B4
23			B5
24			B6
25	Character 4		B1
26			B2
27			B3
28			B4
29			B5
30			B6
31	Parity: (13+14+15+...+30+31 = ODD)	N/A	N/A
32	Parity: (14+16+18+...+30+32 = ODD)	N/A	N/A

NOTE 3: Characters are encoded using the International Alphabet Number-5 (IA-5):

NOTE 4: Coding for status bit:

0 - Function not radiated, or radiated in test mode (not reliable for navigation).
1 - Function radiated in normal mode (in Basic Data Word 2, the back azimuth status also indicates that a back azimuth transmission is to follow).

NOTE 5: Coding for status bits:

I_{21}	I_{22}	
0	0	DME transponder inoperative or not available.
1	0	Only IA mode or DME/N available.
0	1	FA mode, Standard 1, available.
1	1	FA mode, Standard 2, available.

NOTE 6: The value coded shall be the actual beamwidth rounded to the nearest 0.5 degree.

NOTE 7: Where back azimuth guidance is provided, data words 4 and 6 shall be transmitted at intervals of 1.33 seconds or less throughout the approach azimuth coverage and 4 seconds or less throughout the back azimuth coverage.

NOTE 8: Where back azimuth guidance is provided, data word 5 shall be transmitted at an interval of 1.33 seconds or less throughout the back azimuth coverage sector and 4 seconds or less throughout the approach azimuth coverage sector.

NOTE 9: The twelve data bits of the preamble are preceded by an 0.832 millisecond interval (13 clock pulses) of CW for carrier acquisition.

NOTE (GENERAL): Data items that are not applicable to a particular ground equipment shall be transmitted as all zeros.

2.0 Auxiliary Data

Three function identification codes are reserved to indicate transmission of Auxiliary Data A, Auxiliary Data B, and Auxiliary Data C. Auxiliary Data A contents are specified below, Auxiliary Data B contents are reserved for future use and Auxiliary Data C contents are reserved for national use. The address codes of the auxiliary data words shall be as shown in Table C-3.

The organization and timing of digital auxiliary data must be as specified in Table C-4a. Data containing digital information must be transmitted with the least significant bit first. Alphanumeric data characters must be encoded in accordance with the 7-unit code character set as defined by the American National Standard Code for Information Interchange (ASCII). An even parity bit is added to each character. Alphanumeric data must be transmitted in the order in which they are to be read. The serial transmission of a character must be with the lower order bit transmitted first and the parity bit transmitted last. The timing for alphanumeric auxiliary data must be as shown in Table C-4b.

The auxiliary data items specified in Table C-5 are defined as follows:

a. Approach azimuth antenna offset shall represent the minimum distance between the approach azimuth antenna phase center and the vertical plane containing the runway centerline.

b. Approach azimuth to MLS datum point distance shall represent the minimum distance between the approach azimuth antenna phase center and the vertical plane perpendicular to the centerline that contains the MLS datum point.

c. Approach azimuth alignment with runway centerline shall represent the minimum angle between the approach azimuth antenna zero-degree guidance plane and the runway centerline.

d. Approach azimuth antenna coordinate system shall represent the coordinate system (planar or conical) of the angle data transmitted by the approach azimuth antenna.

e. Approach azimuth antenna height shall represent the vertical location of the antenna phase center with respect to the MLS datum point.

f. Approach elevation antenna offset shall represent the minimum distance between the elevation antenna phase center and the vertical plane containing the runway centerline.

g. MLS datum point to threshold distance shall represent the distance measured along the runway centerline from the MLS datum point to the runway threshold.

h. Approach elevation antenna height shall represent the height of the elevation antenna phase center relative to the height of the MLS datum point.

i. MLS datum point elevation shall represent the datum point elevation relative to mean sea level (MSL).

j. Runway threshold height shall represent the vertical location of the intersection of the runway threshold and centerline with respect to the MLS datum point.

k. DME offset shall represent the minimum distance between the DME antenna phase center and the vertical plane containing the runway centerline.

l. DME to MLS datum point distance shall represent the minimum distance between the DME antenna phase center and the vertical plane perpendicular to the centerline that contains the MLS datum point.

m. DME antenna height shall represent the vertical location of the antenna phase center with respect to the MLS datum point.

n. Runway stop-end distance shall represent the distance along centerline between the runway stop-end and the MLS datum point.

o. Back azimuth antenna offset shall represent the minimum distance between the back azimuth antenna phase center and the vertical plane containing the runway centerline.

p. Back azimuth to MLS datum point distance shall represent the minimum distance between the back azimuth antenna and the vertical plane perpendicular to the centerline that contains the MLS datum point.

q. Back azimuth antenna alignment with runway centerline shall represent the minimum angle between the back azimuth antenna zero-degree guidance plane and the runway centerline.

r. Back azimuth antenna coordinate system shall represent the coordinate system (planar and conical) of the angle data transmitted by the back azimuth antenna.

s. Back azimuth antenna height shall represent the vertical location of the antenna phase center with respect to the MLS datum point.

t. Runway visual range (RVR) shall represent measurement of instrument RVR at touchdown zone, midpoint and stop end of the runway and trend values.

u. Surface wind shall represent speed and wind direction (magnetic).

TABLE C-3 AUXILIARY DATA WORD ADDRESS CODES

NO.	I_{13}	I_{14}	I_{15}	I_{16}	I_{17}	I_{18}	I_{19}	I_{20}	NO.	I_{13}	I_{14}	I_{15}	I_{16}	I_{17}	I_{18}	I_{19}	I_{20}
1.	0	0	0	0	0	1	1	1	33.	1	0	0	0	0	1	0	1
2.	0	0	0	0	1	0	1	0	34.	1	0	0	0	1	0	0	0
3.	0	0	0	0	1	1	0	1	35.	1	0	0	0	1	1	1	1
4.	0	0	0	1	0	0	1	1	36.	1	0	0	1	0	0	0	1
5.	0	0	0	1	0	1	0	0	37.	1	0	0	1	0	1	1	0
6.	0	0	0	1	1	0	0	1	38.	1	0	0	1	1	0	1	1
7.	0	0	0	1	1	1	1	0	39.	1	0	0	1	1	1	0	0
8.	0	0	1	0	0	0	1	0	40.	1	0	1	0	0	0	0	0
9.	0	0	1	0	0	1	0	1	41.	1	0	1	0	0	1	1	1
10.	0	0	1	0	1	0	0	0	42.	1	0	1	0	1	0	1	0
11.	0	0	1	0	1	1	1	1	43.	1	0	1	0	1	1	0	1
12.	0	0	1	1	0	0	0	1	44.	1	0	1	1	0	0	1	1
13.	0	0	1	1	0	1	1	0	45.	1	0	1	1	0	1	0	0
14.	0	0	1	1	1	0	1	1	46.	1	0	1	1	1	0	0	1
15.	0	0	1	1	1	1	0	0	47.	1	0	1	1	1	1	1	0
16.	0	1	0	0	0	0	1	1	48.	1	1	0	0	0	0	0	1
17.	0	1	0	0	0	1	0	0	49.	1	1	0	0	0	1	1	0
18.	0	1	0	0	1	0	0	1	50.	1	1	0	0	1	0	1	1
19.	0	1	0	0	1	1	1	0	51.	1	1	0	0	1	1	0	0
20.	0	1	0	1	0	0	0	0	52.	1	1	0	1	0	0	1	0
21.	0	1	0	1	0	1	1	1	53.	1	1	0	1	0	1	0	1
22.	0	1	0	1	1	0	1	0	54.	1	1	0	1	1	0	0	0
23.	0	1	0	1	1	1	0	1	55.	1	1	0	1	1	1	1	1
24.	0	1	1	0	0	0	0	1	56.	1	1	1	0	0	0	1	1
25.	0	1	1	0	0	1	1	0	57.	1	1	1	0	0	1	0	0
26.	0	1	1	0	1	0	1	1	58.	1	1	1	0	1	0	0	1
27.	0	1	1	0	1	1	0	0	59.	1	1	1	0	1	1	1	0
28.	0	1	1	1	0	0	1	0	60.	1	1	1	1	0	0	0	0
29.	0	1	1	1	0	1	0	1	61.	1	1	1	1	0	1	1	1
30.	0	1	1	1	1	0	0	0	62.	1	1	1	1	1	0	1	0
31.	0	1	1	1	1	1	1	1	63.	1	1	1	1	1	1	0	1
32.	1	0	0	0	0	0	1	0	64.	0	0	0	0	0	0	0	0

NOTE 1: Parity bits I_{19} and I_{20} are chosen to satisfy the equations:

$$I_{13} + I_{14} + I_{15} + I_{16} + I_{17} + I_{18} + I_{19} = \text{EVEN}$$

$$I_{14} + I_{16} + I_{18} + I_{20} = \text{EVEN}$$

TABLE C-4a

AUXILIARY DATA FUNCTION TIMING (DIGITAL)

Event	Event Time Slot Begins At: 15.625 kHz Clock Pulse (Number)	Time (milliseconds)
Preamble	0	0
Address Transmissions (Bits I_{13} - I_{20})	25	1.600
Data Transmission (Bits I_{21} - I_{69})	33	2.112
Parity Transmission (Bits I_{70} - I_{76})	82	5.248
End Function (Airborne)	89	5.696
End Guard Time; End Function (Ground)	--	5.900

TABLE C-4b

AUXILIARY DATA FUNCTION TIMING (ALPHANUMERIC)

Event	Event Time Slot Begins At: 15.615 kHz Clock Pulse (Number)	Time (milliseconds)
Preamble	0	0
Address Transmissions (Bits I_{13} - I_{20})	25	1.600
Data Transmission (Bits I_{21} - I_{69})	33	2.112
End Function (Airborne)	89	5.696
End Guard Time; End Function (Ground)	--	5.900

TABLE C-5 AUXILIARY DATA

WORD (See Note 6)	DATA CONTENT	TYPE OF DATA	MAXIMUM TIME BETWEEN TRANSMISSIONS (SECONDS)	BITS USED	RANGE OF VALUES	LEAST SIGNIFICANT BIT
A1	Preamble	Digital	1.0	12		
	Address			8		
	Approach azimuth antenna offset			10	-511 m to +511 m (see Note 3)	1 m
	Approach azumuth to MLS datum point distance			13	0 m to 8,191 m	1 m
	Approach azimuth antenna alignment with runway centerline			12	-20.47° to 20.47° (see Note 3)	0.01°
	Approach azimuth antenna coordinate system			1	(see Note 2)	
	Approach azimuth antenna height			7	-63 m to +63 m (see Note 3)	1 m
	Spare			6		
	Parity			7	(see Note 1)	
A2	Preamble	Digital	1.0	12		
	Address			8		
	Approach elevation antenna offset			10	-511 m to +511 m (see Note 3)	1 m
	MLS datum point to threshold distance			10	0 m to 1,023 m	1 m
	Approach elevation antenna height			7	-6.3 m to +6.3 m (see Note 3)	0.1 m
	MLS datum point elevation			13	-4,095 m to +4,095 m (see Note 3)	1 m
	Runway threshold height			7	-6.3 m to +6.3 m (see Note 3)	0.1 m
	Spare			2		
	Parity			7	(see Note 1)	
A3	Preamble	Digital	(see Note 4)	12		
	Address			8		
	DME offset			12	-2047 m to +2047 m (see Note 3)	1 m
	DME to MLS datum point distance			14	-8191 m to +8191 m (see Note 3)	1 m
	DME antennaa height			7	-63 m to +63 m (see Note 3)	1 m
	Runway stop-end distance			14	0 m to 16,383 m	1 m
	Spare			2		
	Parity			7	(see Note 1)	

TABLE C-5 AUXILIARY DATA
continued

WORD (See Note 6)	DATA CONTENT	TYPE OF DATA	MAXIMUM TIME BETWEEN TRANSMISSIONS (SECONDS)	BITS USED	RANGE OF VALUES	LEAST SIGNIFICANT BIT
A4	Preamble	Digital	(see Note 5)	12		
	Address			8		
	Back azimuth antenna			10	-511 m to +511 m (see Note 3)	1 m
	Back azimuth to MLS datum point distance			11	0 m to 2,047 m	1 m
	Back azimuth antenna alignment with runway centerline			12	-20.47° to 20.47° (see Note 3)	0.1°
	Back azimuth antenna coordinate system			1	(see Note 2)	
	Back azimuth antenna height			7	-63 m to +63 m (see Note 3)	1 m
	Spare			8		
	Parity			7	(see Note 1)	
A5	Preamble	Digital	1.0	12		
	Address			8		
	RVR (touchdown zone)			11	0 to 2,555 m (see Note 7)	5 m
	RVR (midpoint)			11	0 to 2,555 m (see Note 7)	5 m
	RVR (stop end)			11	0 to 2,555 m (see Note 7)	5 m
	Surface wind speed			7	0 to 127 knots	1 knot
	Surface wind direction (magnetic)			9	0 to 359°	1°
	Parity			7	(see Note 1)	

NOTE 1: Parity bits I_{70} to I_{76} are chosen to satisfy the equations which follow:

FOR BIT I_{70}

$$EVEN = (I_{13} + \ldots + I_{18}) + I_{20} + I_{22} + I_{24} + I_{25} + I_{28} + I_{29} + I_{31} + I_{32} + I_{33} + I_{35} + I_{36} + I_{38} + I_{41} + I_{44} + I_{45} + I_{46} + I_{50} + (I_{52} + \ldots + I_{55}) + I_{58} + I_{60} + I_{64} + I_{65} + I_{70}$$

FOR BIT I_{71}

$$EVEN = (I_{14} + \ldots + I_{19}) + I_{21} + I_{23} + I_{25} + I_{26} + I_{29} + I_{30} + I_{32} + I_{33} + I_{34} + I_{36} + I_{37} + I_{39} + I_{42} + I_{45} + I_{46} + I_{47} + I_{51} + (I_{53} + \ldots + I_{56}) + I_{59} + I_{61} + I_{65} + I_{66} + I_{71}$$

FOR BIT I_{72}

$$EVEN = (I_{15} + \ldots + I_{20}) + I_{22} + I_{24} + I_{26} + I_{27} + I_{30} + I_{31} + I_{33} + I_{34} + I_{35} + I_{37} + I_{38} + I_{40} + I_{43} + I_{46} + I_{47} + I_{48} + I_{52} + (I_{54} + \ldots + I_{57}) + I_{60} + I_{62} + I_{66} + I_{67} + I_{72}$$

FOR BIT I_{73}

$$EVEN = (I_{16} + \ldots + I_{21}) + I_{23} + I_{25} + I_{27} + I_{28} + I_{31} + I_{32} + I_{34} + I_{35} + I_{36} + I_{38} + I_{39} + I_{41} + I_{44} + I_{47} + I_{48} + I_{49} + I_{53} + (I_{55} + \ldots + I_{58}) + I_{61} + I_{63} + I_{67} + I_{68} + I_{73}$$

FOR BIT I_{74}

$$EVEN = (I_{17} + \ldots + I_{22}) + I_{24} + I_{26} + I_{28} + I_{29} + I_{32} + I_{33} + I_{35} + I_{36} + I_{37} + I_{39} + I_{40} + I_{42} + I_{45} + I_{48} + I_{49} + I_{50} + I_{54} + (I_{56} + \ldots + I_{59}) + I_{62} + I_{64} + I_{68} + I_{69} + I_{74}$$

FOR BIT I_{75}

$$EVEN = (I_{13} + \ldots + I_{17}) + I_{19} + I_{21} + I_{23} + I_{24} + I_{27} + I_{28} + I_{30} + I_{31} + I_{32} + I_{34} + I_{35} + I_{37} + I_{40} + I_{43} + I_{44} + I_{45} + I_{49} + (I_{51} + \ldots + I_{54}) + I_{57} + I_{59} + I_{63} + I_{64} + I_{69} + I_{75}$$

FOR BIT I_{76}

$$EVEN = I_{13} + I_{14} + \ldots + I_{75} + I_{76}$$

NOTE 2: Code for antenna coordinates is: 0 = conical, 1 = planar

NOTE 3: The convention for the coding of negative numbers is as follows:

- MSB is the sign bit

  0 = +
  1 = -

- Other bits represent the absolute value.

The convention for the antenna location is as follows:

> As viewed from the MLS approach reference datum looking toward the datum point, a positive number shall represent a location to the right of the runway centerline (lateral offset) or above the runway (vertical offset), or towards the stop end of the runway (longitudinal distance).

The convention for the antenna alignment is as follows:

> As viewed from above, a positive number shall represent clockwise rotation from the runway centerline to the respective zero-degree guidance plane.

NOTE 4: Data word A3 is transmitted at intervals of 1.0 second or less throughout the approach azimuth coverage sector, except when back azimuth guidance is provided. Where back azimuth is provided, transmit at intervals of 1.33 seconds or less throughout the approach azimuth sector and 4.0 seconds or less throughout the back azimuth coverage sector.

NOTE 5: Where back azimuth guidance is provided, transmit at intervals of 1.33 seconds or less throughout the back azimuth coverage sector and 4.0 seconds or less throughout the approach azimuth coverage sector.

NOTE 6: The designation "A1" represents the function identification code for "Auxiliary Data A" and address code number 1.

NOTE 7: The tenth and eleventh bits transmitted for each RVR value are used to provide trend information. The convention for coding is as follows:

	Tenth Bit	Eleventh Bit
Off	0	0
Decreasing	1	0
Equal	0	1
Increasing	1	1

APPENDIX C

C SOURCE CODE SITEROMS.C

```
/*************************************************************************/
/*                      PROGRAM SITEROMS.C     24 Nov 89                 */
/*                      By Thomas J. D'Ottavi    ACD-330                 */
/*************************************************************************/
#include <ctype.h>
#include <math.h>
#include <conio.h>
#include <dos.h>
#include <bios.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>

void strdel();
void get_dpsk();
void get_aux_dpsk();
void print_b_code();
void print_a_code();
void zero_data();
void get_bits();
void get_parity76();
int get_parity();
void m_code();
void get_morse();
void count();
void check_id(int);

char morse[65];
char basic_data[6][33],dpsk[6][33];
char aux_data[4][77],aux_dpsk[4][77];
unsigned char cnt=0;

FILE *output,*fptr,*fptr1;

void main()
{
   char temp_word[20];
   char dme_sta[3],temp_str[16];
   char locat[60],system_id[3],
        remark[60],edate[15],
        glide1[5],az_beam1[5],el_beam1[5],
        baz_beam1[6],el_lo1[6],dme_dist2[8],
        az_ant_align1[6],az_ant_hgt1[6],
        dme_mls_dd1[8],file_name[22],rom_name[22],
        el_rms[22];
   unsigned char az_cs[1],baz_cs[1];

   int k,word,len,az_thres,az_thres1,az_thres2;
   int ch,az_neg,az_pos,az_neg1,az_pos1,
       baz_neg,baz_neg1,clear,az_zero,az_zero1,
       baz_pos,baz_pos1,baz_zero,baz_zero1,
       mls_dpe,rwy_stop_dist;
```

```
int baz_sta,baz_sta5,az_sta,el_sta,disp;

double az_lo,az_lo1,az_mls_dp,az_mls_dp1,el_lo,
      glide,az_beam,dme_lo,dme_lo1,dme_mls_dd,el_mls_dp,
      el_beam,dme_dist,dme_dist1,dme_ant_hgt,
      dme_ant_hgt1,baz_beam,az_ant_align,
      az_ant_hgt,baz_lo,baz_lo1,baz_mls_dd,
      baz_mls_dd1,baz_ant_hgt,baz_ant_hgt1,baz_mls_dp,baz_mls_dp1,
      el_ant_hgt,el_ant_hgt1,mls_dp_elv,rwy_thr_hgt,baz_ant_align,
      baz_ant_align1,temp,xint;
register int i;

clrscr();
printf("\n\n\n\r MLS Site Location: ");
gets(locat);
printf("\n\n\r Remarks: ");
gets(remark);
printf("\n\n\r Effective date: ");
gets(edate);
printf("\n\n\r Format: ( 1 = FAA-STD-022c   2 = RTCA/DO-198 ): ");
gets(temp_str);
ch=atoi(temp_str);
printf("\n\n\r Output to: ( 1 = Printer   2 = Screen   3 = File ): ");
gets(temp_str);
disp=atoi(temp_str);
switch(disp){
   case 1 : output = stdprn; break;
   case 2 : output = stdout; break;
   case 3 : printf("\n\n\r Enter File Name: ");
            gets(file_name);
            strcpy(rom_name,file_name);
            strcat(rom_name,".ROM");
            strcpy(el_rms,file_name);
            strcat(el_rms,".ERM");
            fptr=fopen(rom_name,"w");
            fptr1=fopen(el_rms,"w");
            strcat(file_name,".DOC");
            output = fopen(file_name,"w"); break;
}
clrscr();
```

```
/*************************************************************************
 *                          basic data word # 1                          *
 ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ****/

   gotoxy(31,1);
   printf("BASIC DATA WORD  1");
   printf("\n\n\r AZ to threshold distance (0m to 6300m): ");
   gets(temp_str);
   az_thres=atoi(temp_str);
   printf("\n\n\r AZ proportional coverage (0 to -62): -");
   gets(temp_str);
   az_neg=atoi(temp_str);
   printf("\n\n\r AZ proportional coverage (0 to +62): +");
   gets(temp_str);
   az_pos=atoi(temp_str);
   printf("\n\n\r Clearance signal type (0 = pulse; 1 = SB): ");
   gets(temp_str);
   clear=atoi(temp_str);
   clrscr();

/*************************************************************************
 *                          basic data word # 2                          *
 ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ****/

   gotoxy(31,1);
   printf("BASIC DATA WORD  2");
   printf("\n\n\r Minium glide path (2.0 to 14.7): ");
   gets(temp_str);
   glide=atof(temp_str);
   printf("\n\n\r BAZ status (0 = none; 1 = present): ");
   gets(temp_str);
   baz_sta=atoi(temp_str);
   printf("\n\n\r DME status: ");
   printf("\n\r 0 0 = DME transponder inoperative or not available. ");
   printf("\n\r 1 0 = Only IA mode or DME/N available. ");
   printf("\n\r 0 1 = FA mode, Standard 1, available. ");
   printf("\n\r 1 1 = FA mode, Standard 2, available. ");
   gets(dme_sta);
   printf("\n\n\r AZ status (1 = ok; 0 = down): ");
   gets(temp_str);
   az_sta=atoi(temp_str);
   printf("\n\n\r EL status (1 = ok; 0 = down): ");
   gets(temp_str);
   el_sta=atoi(temp_str);
   clrscr();
```

```
/*************************************************************************
 *                          basic data word # 3                          *
 ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ****/

   gotoxy(31,1);
   printf("BASIC DATA WORD  3");
   printf("\n\n\r AZ beamwidth (0.5 to 4.0): ");
   gets(temp_str);
   az_beam=atof(temp_str);
   printf("\n\n\r EL beamwidth (0.5 to 2.5): ");
   gets(temp_str);
   el_beam=atof(temp_str);
   printf("\n\n\r DME distance to MLS datum (0m to 6387.5m): ");
   gets(temp_str);
   dme_dist=atof(temp_str);
   clrscr();

/*************************************************************************
 *                          basic data word # 4                          *
 ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ****/

   gotoxy(31,1);
   printf("BASIC DATA WORD  4");
   printf("\n\n\r AZ 0-deg. guidance plane (0 to 359): ");
   gets(temp_str);
   az_zero=atoi(temp_str);
   printf("\n\n\r BAZ 0-deg. guidance plane (0 to 359): ");
   gets(temp_str);
   baz_zero=atoi(temp_str);
   clrscr();

/*************************************************************************
 *                          basic data word # 5                          *
 ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ****/


   if(baz_sta==1){
      gotoxy(31,1);
      printf("BASIC DATA WORD  5");
      printf("\n\n\r BAZ proportional coverage (0 to -42): -");
      gets(temp_str);
      baz_neg=atoi(temp_str);
      printf("\n\n\r BAZ proportional coverage (0 to +42): +");
      gets(temp_str);
      baz_pos=atoi(temp_str);
      printf("\n\n\r BAZ beamwidth (0.5 to 4.0): ");
      gets(temp_str);
      baz_beam=atof(temp_str);
      printf("\n\n\r BAZ status (1 = ok; 0 = down): ");
      gets(temp_str);
      baz_sta5=atoi(temp_str);
```

```
      clrscr();
   }

/*************************************************************************
 *                          basic data word # 6                          *
 ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ****/

   gotoxy(31,1);
   printf("BASIC DATA WORD  6");
   printf("\n\n\r System ID: M-");
   gets(system_id);
   clrscr();

/*************************************************************************/
/*                                AUX DATA WORD 1                        */
/*** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ***/

   gotoxy(32,1);
   printf("AUX DATA WORD  1");
   printf("\n\n\r AZ ant lateral offset: ");
   gets(temp_str);
   az_lo=atof(temp_str);
   printf("\n\n\r AZ to MLS datum distance: ");
   gets(temp_str);
   az_mls_dp=atof(temp_str);
   printf("\n\n\r AZ ant alignment w/ centerline: ");
   gets(temp_str);
   az_ant_align=atof(temp_str);
   printf("\n\n\r AZ coordinate system (0 = conical  1 = planar): ");
   gets(az_cs);
   if(ch==2){
      printf("\n\n\r AZ ant height: ");
      gets(temp_str);
      az_ant_hgt=atof(temp_str);
   }
   clrscr();

/*************************************************************************/
/*                                AUX DATA WORD 2                        */
/*** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ***/

   gotoxy(32,1);
   printf("AUX DATA WORD  2");
   printf("\n\n\r EL ant lateral offset: ");
   gets(temp_str);
   el_lo=atof(temp_str);
   printf("\n\n\r MLS datum to threshold dist.: ");
   gets(temp_str);
   el_mls_dp=atof(temp_str);
   printf("\n\n\r EL height: ");
   gets(temp_str);
   el_ant_hgt=atof(temp_str);
```

```
   if(ch==2){
      printf("\n\n\r MLS datum point elev.: ");
      gets(temp_str);
      mls_dp_elv=atof(temp_str);
      printf("\n\n\r Runway threshold height: ");
      gets(temp_str);
      rwy_thr_hgt=atof(temp_str);
   }
   clrscr();

/*************************************************************************/
/*                                AUX DATA WORD 3                        */
/*** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ***/

   gotoxy(32,1);
   printf("AUX DATA WORD  3");
   printf("\n\n\r DME lateral offset: ");
   gets(temp_str);
   dme_lo=atof(temp_str);
   printf("\n\n\r DME to MLS datum distance: ");
   gets(temp_str);
   dme_mls_dd=atof(temp_str);
   if(ch==2){
      printf("\n\n\r DME ant height: ");
      gets(temp_str);
      dme_ant_hgt=atof(temp_str);
      printf("\n\n\r Runway stop-end dist.: ");
      gets(temp_str);
      rwy_stop_dist=atoi(temp_str);
   }
   clrscr();

/*************************************************************************/
/*                                AUX DATA WORD 4                        */
/*** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ** ***/


   if(baz_sta==1){
      gotoxy(32,1);
      printf("AUX DATA WORD  4");
      printf("\n\n\r BAZ ant lateral offset: ");
      gets(temp_str);
      baz_lo=atof(temp_str);
      printf("\n\n\r BAZ to MLS datum distance: ");
      gets(temp_str);
      baz_mls_dp=atof(temp_str);
      printf("\n\n\r BAZ ant alignment w/ centerline: ");
      gets(temp_str);
      baz_ant_align=atof(temp_str);
      if(ch==2){
         printf("\n\n\r BAZ ant coordinate system: ");
         gets(baz_cs);
```

```
        printf("\n\n\r BAZ ant height: ");
        gets(temp_str);
        baz_ant_hgt=atof(temp_str);
     }
     clrscr();
  }

/***************************  BASIC DATA WORD 1  ****************************/

  strcpy(basic_data[0],"111010101000");
  temp=fmod(az_thres,100.0);
  if(az_thres<0)
     az_thres=0;
  else if(az_thres>6300)
     az_thres=6300;
  az_thres2=az_thres;
  if((int)temp<50)
     az_thres-=(int)temp;
  else
     az_thres+=(100-(int)temp);
  az_thres1=az_thres;
  if(az_thres>=100)
     az_thres=az_thres/100;
  else
     az_thres=0;
  itoa(az_thres,temp_word,2);
  strrev(temp_word);
  len=strlen(temp_word);
  if(len<6)
     for(i=0;i<6-len;i++)
        strcat(temp_word,"0");
  az_thres=az_thres*100;
  strcat(basic_data[0],temp_word);
  if(az_neg<0)
      az_neg=0;
   else if(az_neg>62)
      az_neg=62;
  temp=fmod(az_neg,2.0);
  if((int)temp>0)
     az_neg+=1;
  az_neg1=az_neg;
  itoa(az_neg,temp_word,2);
  strrev(temp_word);
  len=strlen(temp_word);
  if(len<5)
     for(i=0;i<5-len;i++)
        strcat(temp_word,"0");
  else if(len>5)
     strdel(temp_word,len-5);
  strcat(basic_data[0],temp_word);
  if(az_pos<0)
     az_pos=0;
```

```
else if(az_pos>62)
   az_pos=62;
temp=fmod(az_pos,2.0);
if((int)temp>0)
   az_pos+=1;
az_pos1=az_pos;
itoa(az_pos,temp_word,2);
strrev(temp_word);
len=strlen(temp_word);
if(len<5)
   for(i=0;i<5-len;i++)
      strcat(temp_word,"0");
else if(len>5)
   strdel(temp_word,len-5);
strcat(basic_data[0],temp_word);
if(clear==1)
   strcpy(&temp_word[0],"10");
else
   strcpy(&temp_word[0],"00");
len=strlen(temp_word);
if(len>2)
   strdel(temp_word,len-2);
strcat(basic_data[0],temp_word);
temp=get_parity(0,31);
if(fmod(temp,2.0)>0)
   strcpy(&temp_word[0],"0");
else
   strcpy(&temp_word[0],"1");
temp=get_parity(0,32);
if(fmod(temp,2.0)>0)
   strcpy(&temp_word[1],"0");
else
   strcpy(&temp_word[1],"1");
len=strlen(temp_word);
if(len>2)
   strdel(temp_word,len-2);
strcat(basic_data[0],temp_word);
get_dpsk(0);

/****************************  BASIC DATA WORD 2  ************************/

strcpy(basic_data[1],"111010111100");
if(glide<2.0)
   glide=2.0;
else if(glide>14.7)
   glide=14.7;
   gcvt(glide,4,glide1);
glide=glide*10;
glide-=20.0;
itoa((int)glide,temp_word,2);
strrev(temp_word);
len=strlen(temp_word);
```

```
if(len<7)
   for(i=0;i<7-len;i++)
      strcat(temp_word,"0");
strcat(basic_data[1],temp_word);
glide=glide/10;
strcpy(&basic_data[1][19],(baz_sta==1)? "1" : "0");
strcat(basic_data[1],dme_sta);
strcpy(&basic_data[1][22],(az_sta==1) ? "1" : "0");
strcpy(&basic_data[1][23],(el_sta==1) ? "1" : "0");
for(i=24;i<30;i++)
   strcpy(&basic_data[1][i],"0");
temp=get_parity(1,31);
   strcpy(&temp_word[0],(fmod(temp,2.0)>0) ? "0" : "1");
temp=get_parity(1,32);

   strcpy(&temp_word[1],(fmod(temp,2.0)>0) ? "0" : "1");
len=strlen(temp_word);
if(len>2)
   strdel(temp_word,len-2);
strcat(basic_data[1],temp_word);
get_dpsk(1);

/*************************  BASIC DATA WORD 3  ***************************/

strcpy(basic_data[2],"111011010000");
if(az_beam<0.5)
   az_beam=0.5;
else if(az_beam>4.0)
   az_beam=4.0;
gcvt(az_beam,4,az_beam1);
az_beam=(az_beam*2)-1;
itoa((int)az_beam,temp_word,2);
strrev(temp_word);
len=strlen(temp_word);
if(len<3)
   for(i=0;i<3-len;i++)
      strcat(temp_word,"0");
strcat(basic_data[2],temp_word);
if(el_beam<0.5)
   el_beam=0.5;
else if(el_beam>2.5)
   el_beam=2.5;
gcvt(el_beam,4,el_beam1);
el_beam=(el_beam*2)-1;
itoa((int)el_beam,temp_word,2);
strrev(temp_word);
len=strlen(temp_word);
if(len<3)
   for(i=0;i<3-len;i++)
      strcat(temp_word,"0");
strcat(basic_data[2],temp_word);
dme_dist1=0;
```

```
if(dme_dist<0)
   dme_dist=0;
else if(dme_dist>6300.5)
   dme_dist=6300.5;
gcvt(dme_dist,6,dme_dist2);
if(dme_dist>=12.5)
   dme_dist=dme_dist/12.5;
else
   dme_dist = 0;
temp=modf(dme_dist,&xint);
if(temp>=.5)
   xint++;
itoa((int)xint,temp_word,2);
strrev(temp_word);
len=strlen(temp_word);
if(len<9)
   for(i=0;i<9-len;i++)
   strcat(temp_word,"0");
dme_dist1=((temp+xint)*12.5);
strcat(basic_data[2],temp_word);
strcat(basic_data[2],"000");
temp=get_parity(2,31);
if(fmod(temp,2.0)>0)
   strcpy(&temp_word[0],"0");
else
   strcpy(&temp_word[0],"1");
temp=get_parity(2,32);
if(fmod(temp,2.0)>0)
   strcpy(&temp_word[1],"0");
else
   strcpy(&temp_word[1],"1");
len=strlen(temp_word);
if(len>2)
   strdel(temp_word,len-2);
strcat(basic_data[2],temp_word);
get_dpsk(2);
dme_dist=dme_dist*12.5;

/***************************  BASIC DATA WORD 4  ************************/

strcpy(basic_data[3],"111011000100");
if(az_zero>=360)
   az_zero=0;
az_zero1=az_zero;
itoa(az_zero,temp_word,2);
strrev(temp_word);
len=strlen(temp_word);
if(len<9)
   for(i=0;i<9-len;i++)
      strcat(temp_word,"0");
strcat(basic_data[3],temp_word);
if(baz_zero>=360)
```

```
   baz_zero=0;
baz_zero1=baz_zero;
itoa(baz_zero,temp_word,2);
strrev(temp_word);
len=strlen(temp_word);
if(len<9)
   for(i=0;i<9-len;i++)
      strcat(temp_word,"0");
strcat(basic_data[3],temp_word);
temp=get_parity(3,31);
if(fmod(temp,2.0)>0)
   strcpy(&temp_word[0],"0");
else
   strcpy(&temp_word[0],"1");
temp=get_parity(3,32);
if(fmod(temp,2.0)>0)
   strcpy(&temp_word[1],"0");
else
   strcpy(&temp_word[1],"1");
len=strlen(temp_word);
if(len>2)
   strdel(temp_word,len-2);
strcat(basic_data[3],temp_word);
get_dpsk(3);

/****************************  BASIC DATA WORD 5  ************************/

strcpy(basic_data[4],"111011101100");
if(baz_sta==1){
   if(baz_neg<0)
       baz_neg=0;
    else if(baz_neg>42)
       baz_neg=42;
   temp=fmod(baz_neg,2.0);
   if((int)temp>0)
      baz_neg+=1;
   baz_neg1=baz_neg;
   itoa(baz_neg,temp_word,2);
   strrev(temp_word);
   len=strlen(temp_word);
   if(len<5)
      for(i=0;i<5-len;i++)
         strcat(temp_word,"0");
   else if(len>5)
      strdel(temp_word,len-5);
   strcat(basic_data[4],temp_word);
   if(baz_pos<0)
      baz_pos=0;
   else if(baz_pos>42)
      baz_pos=42;
   temp=fmod(baz_pos,2.0);
   if((int)temp>0)
```

```
      baz_pos+=1;
   baz_pos1=baz_pos;
   itoa(baz_pos,temp_word,2);
   strrev(temp_word);
   len=strlen(temp_word);
   if(len<5)
      for(i=0;i<5-len;i++)
         strcat(temp_word,"0");
   else if(len>5)
      strdel(temp_word,len-5);
   strcat(basic_data[4],temp_word);
   gcvt(baz_beam,4,baz_beam1);
   if(baz_beam<0.5)
      baz_beam=0.5;
   else if(baz_beam>4.0)
      baz_beam=4.0;
   baz_beam=(baz_beam*2)-1;
   itoa((int)baz_beam,temp_word,2);
   strrev(temp_word);
   len=strlen(temp_word);
   if(len<3)
      for(i=0;i<3-len;i++)
         strcat(temp_word,"0");
   strcat(basic_data[4],temp_word);
   strcpy(&basic_data[4][25],(baz_sta5==1)? "1" : "0");
   strcat(basic_data[4],"0000");
   temp=get_parity(4,31);
   strcpy(&temp_word[0],(fmod(temp,2.0)>0)?"0":"1");
   temp=get_parity(4,32);
   strcpy(&temp_word[1],(fmod(temp,2.0)>0)?"0":"1");
   len=strlen(temp_word);
   if(len>2)
      strdel(temp_word,len-2);
   strcat(basic_data[4],temp_word);
   if(baz_sta==0)
      for(i=0;i<32;i++)
      strcpy(&basic_data[4][i],"0");
   get_dpsk(4);
}

/***************************** BASIC DATA WORD # 6 *************************/

   strcpy(basic_data[5],"111010001101");
   strupr(system_id);
   get_bits(system_id[0],12);
   get_bits(system_id[1],18);
   get_bits(system_id[2],24);
   temp=get_parity(5,31);
   strcpy(&temp_word[0],(fmod(temp,2.0)>0)?"0":"1");
   temp=get_parity(5,32);
   strcpy(&temp_word[1],(fmod(temp,2.0)>0)?"0":"1");
   len=strlen(temp_word);
```

```
   if(len>2)
      strdel(temp_word,len-2);
   strcat(basic_data[5],temp_word);
   get_dpsk(5);

/****************************** AUX DATA WORD # 1 *************************/

   strcpy(aux_data[0],"11101111001000000111");
   az_lo1=az_lo;
   if(az_lo<-512)
      az_lo=-512;
   else if(az_lo>512)
      az_lo=512;
   if(az_lo<0)
      az_lo=fabs(az_lo)+512;
   temp=modf(az_lo,&xint);
   if(temp>.5)
      az_lo=xint+1;
    else
     az_lo=xint;
   itoa((int)az_lo,temp_word,2);
   strrev(temp_word);
   len=strlen(temp_word);
   if(len<10)
      for(i=0;i<10-len;i++)
         strcat(temp_word,"0");
   strcat(aux_data[0],temp_word);
   az_mls_dp1=az_mls_dp;
   if(az_mls_dp<0)
      az_mls_dp=0;
   else if(az_mls_dp>8191)
      az_mls_dp=8191;
   temp=modf(az_mls_dp,&xint);
   if(temp>.5)
      az_mls_dp=xint+1;
    else
     az_mls_dp=xint;
   itoa((int)az_mls_dp,temp_word,2);
   strrev(temp_word);
   len=strlen(temp_word);
   if(len<13)
      for(i=0;i<13-len;i++)
         strcat(temp_word,"0");
   strcat(aux_data[0],temp_word);
   gcvt(az_ant_align,5,az_ant_align1);
   if(az_ant_align< -20.47)
      az_ant_align= -20.47;
   else if(az_ant_align>20.47)
      az_ant_align=20.47;
   if(az_ant_align<0)
      az_ant_align=(fabs(az_ant_align)*100)+2048;
   else
```

```
      az_ant_align=az_ant_align*100;
   itoa((int)az_ant_align,temp_word,2);
   strrev(temp_word);
   len=strlen(temp_word);
   if(len<12)
      for(i=0;i<12-len;i++)
         strcat(temp_word,"0");
   strcat(aux_data[0],temp_word);
   strcat(aux_data[0],az_cs);
   if(ch==2){
      gcvt(az_ant_hgt,4,az_ant_hgt1);
      if(az_ant_hgt< -63.0)
         az_ant_hgt= -63.0;
      else if(az_ant_hgt>63.0)
         az_ant_hgt=63.0;
      if(az_ant_hgt<0.0)
         az_ant_hgt=fabs(az_ant_hgt)+64;
      if(az_ant_hgt<0)
         az_ant_hgt=0;
      itoa((int)az_ant_hgt,temp_word,2);
      strrev(temp_word);
      len=strlen(temp_word);
      if(len>7)
         strdel(temp_word,len-7);
      else if(len<7)
         for(i=0;i<7-len;i++)
            strcat(temp_word,"0");
      strcat(aux_data[0],temp_word);
      strcat(aux_data[0],"000000");
   }
   else
      strcat(aux_data[0],"0000000000000");
   get_parity76(0);
   get_aux_dpsk(0);
   if(az_ant_align>2047){
      az_ant_align-=2048;
      az_ant_align=az_ant_align-(az_ant_align*2);
   }
   az_ant_align=az_ant_align/100;

/****************************** AUX DATA WORD # 2 ************************/

   strcpy(aux_data[1],"11101111001000001010");
   gcvt(el_lo,4,el_lo1);
   if(el_lo<-512)
      el_lo=-512;
   else if(el_lo>512)
      el_lo=512;
   if(el_lo<0)
      el_lo=fabs(el_lo)+512;
   temp=modf(el_lo,&xint);
   if(temp>.5)
```

```
      el_lo=xint+1;
   else
      el_lo=xint;
   itoa((int)el_lo,temp_word,2);
   strrev(temp_word);
   len=strlen(temp_word);
   if(len<10)
      for(i=0;i<10-len;i++)
         strcat(temp_word,"0");
   strcat(aux_data[1],temp_word);
   if(el_mls_dp<0)
      el_mls_dp=0;
   else if(el_mls_dp>1023)
      el_mls_dp=1023;
   temp=modf(el_mls_dp,&xint);
   if(temp>.5)
      el_mls_dp=xint+1;
   else
      el_mls_dp=xint;
   itoa((int)el_mls_dp,temp_word,2);
   strrev(temp_word);
   len=strlen(temp_word);
   if(len<10)
      for(i=0;i<10-len;i++)
         strcat(temp_word,"0");
   strcat(aux_data[1],temp_word);
   el_ant_hgt1=el_ant_hgt;
   if(el_ant_hgt< -6.3)
      el_ant_hgt= -6.3;
   else if(el_ant_hgt>6.3)
      el_ant_hgt=6.3;
   if(el_ant_hgt<0.0)
      el_ant_hgt=(fabs(el_ant_hgt)*10)+64;
   else
      el_ant_hgt=el_ant_hgt*10;
   temp=modf(el_ant_hgt,&xint);
   if(temp>.5)
      el_ant_hgt=xint+1;
   else
      el_ant_hgt=xint;
   if(el_ant_hgt<1)
      el_ant_hgt=1;
   itoa((int)el_ant_hgt,temp_word,2);
   strrev(temp_word);
   len=strlen(temp_word);
   if(len<7)
      for(i=0;i<7-len;i++)
         strcat(temp_word,"0");
   strcat(aux_data[1],temp_word);
   if(ch==2){
      if(mls_dp_elv<0)
         mls_dp_elv=fabs(mls_dp_elv)+4096;
```

```
      temp=modf(mls_dp_elv,&xint);
      if(temp<=.5)
         mls_dp_elv=xint;
      else if(temp>.5)
         mls_dp_elv=xint+1;
      itoa((int)mls_dp_elv,temp_word,2);
      strrev(temp_word);
      len=strlen(temp_word);
      if(len<13)
         for(i=0;i<13;i++)
            strcat(temp_word,"0");
      strcat(aux_data[1],temp_word);
      mls_dpe=(int)mls_dp_elv;
      mls_dp_elv=xint+temp;
      if(rwy_thr_hgt< -6.3)
         rwy_thr_hgt= -6.3;
      else if(rwy_thr_hgt>6.3)
         rwy_thr_hgt=6.3;
      if(rwy_thr_hgt<0)
         rwy_thr_hgt=(fabs(rwy_thr_hgt)*10)+64;
      else
         rwy_thr_hgt=rwy_thr_hgt*10;
      itoa((int)rwy_thr_hgt,temp_word,2);
      strrev(temp_word);
      len=strlen(temp_word);
      if(len<7)
         for(i=0;i<7-len;i++)
            strcat(temp_word,"0");
      strcat(aux_data[1],temp_word);
      strcat(aux_data[1],"00");
   }
   else
      strcat(aux_data[1],"00000000000000000000");
   get_parity76(1);
   get_aux_dpsk(1);
   el_ant_hgt=el_ant_hgt/10;

/****************************** AUX DATA WORD # 3 *************************/

   strcpy(aux_data[2],"11101111001000001101");
   dme_lo1=dme_lo;
   if(ch==2){
      if(dme_lo>2047)
         dme_lo=2047;
      else if(dme_lo<-2047)
         dme_lo=-2047;
      if(dme_lo<0)
         dme_lo=fabs(dme_lo)+2048;
   }
   else
      if(dme_lo>511)
         dme_lo=511;
```

```
      else if(dme_lo<-511)
         dme_lo=-511;
      if(dme_lo<0)
         dme_lo=fabs(dme_lo)+512;
   temp=modf(dme_lo,&xint);
   if(temp>=.5)
      xint++;
   dme_lo=xint;
   itoa((int)dme_lo,temp_word,2);
   strrev(temp_word);
   len=strlen(temp_word);
   if(len< (ch == 2) ? 12 : 10)
      for(i=0;i<((ch == 2) ? 12 : 10) - len;i++)
         strcat(temp_word,"0");
   strcat(aux_data[2],temp_word);
   gcvt(dme_mls_dd,6,dme_mls_dd1);
   if(dme_mls_dd>8191)
      dme_mls_dd=8191;
   else if(dme_mls_dd< -8191)
      dme_mls_dd= -8191;
   if(dme_mls_dd<0)
      dme_mls_dd=fabs(dme_mls_dd)+8192;
   temp=0;
   temp=modf(dme_mls_dd,&xint);
   if(temp>=.5)
      xint++;
   dme_mls_dd=xint;
   itoa((int)dme_mls_dd,temp_word,2);
   strrev(temp_word);
   len=strlen(temp_word);
   if(len<14)
      for(i=0;i<14-len;i++)
         strcat(temp_word,"0");
   strcat(aux_data[2],temp_word);
   if(ch==2){
      dme_ant_hgt1=dme_ant_hgt;
      if(dme_ant_hgt>63)
         dme_ant_hgt=63;
      else if(dme_ant_hgt<-63)
         dme_ant_hgt=-63;
      if(dme_ant_hgt<0)
         dme_ant_hgt=fabs(dme_ant_hgt)+64;
      temp=modf(dme_ant_hgt,&xint);
      if(temp>.5)
         dme_ant_hgt=xint+1;
      else
         dme_ant_hgt=xint;
      itoa((int)dme_ant_hgt,temp_word,2);
      len=strlen(temp_word);
      if(len<7)
         for(i=0;i<7-len;i++)
            strcat(temp_word,"0");
```

```
      strcat(aux_data[2],temp_word);
      if(rwy_stop_dist>16383)
         rwy_stop_dist=16383;
      else if(rwy_stop_dist<0)
         rwy_stop_dist=0;
      itoa(rwy_stop_dist,temp_word,2);
      strrev(temp_word);
      len=strlen(temp_word);
      if(len<14)
         for(i=0;i<14-len;i++)
            strcat(temp_word,"0");
      strcat(aux_data[2],temp_word);
      strcat(aux_data[2],"00");
   }
   else
      strcat(aux_data[2],"00000000000000000000000000");
   get_parity76(2);
   get_aux_dpsk(2);
   if(dme_lo>((ch==2)?2047:512))
      dme_lo-=((ch==2)?2048:512);

/****************************** AUX DATA WORD # 4 *************************/

   strcpy(aux_data[3],"11101111001000010011");
   baz_lo1=baz_lo;
   if(baz_lo<-511)
      baz_lo=-511;
   else if(baz_lo>511)
      baz_lo=511;
   if(baz_lo<0)
      baz_lo=fabs(baz_lo)+512;
   temp=modf(baz_lo,&xint);
   if(temp>.5)
      baz_lo=xint+1;
    else
     baz_lo=xint;
   itoa((int)baz_lo,temp_word,2);
   strrev(temp_word);
   len=strlen(temp_word);
   if(len<10)
      for(i=0;i<10-len;i++)
         strcat(temp_word,"0");
   strcat(aux_data[3],temp_word);
   baz_mls_dp1=baz_mls_dp;
   if(baz_mls_dp<0)
      baz_mls_dp=0;
   else if(baz_mls_dp>2047)
      baz_mls_dp=2047;
   temp=modf(baz_mls_dp,&xint);
   if(temp>.5)
      baz_mls_dp=xint+1;
    else
```

```
  baz_mls_dp=xint;
itoa((int)baz_mls_dp,temp_word,2);
strrev(temp_word);
len=strlen(temp_word);
if(len<11)
   for(i=0;i<11-len;i++)
      strcat(temp_word,"0");
strcat(aux_data[3],temp_word);
baz_ant_align1=baz_ant_align;
if(baz_ant_align< -20.47)
   baz_ant_align= -20.47;
else if(baz_ant_align>20.47)
   baz_ant_align=20.47;
if(baz_ant_align<0)
   baz_ant_align=(fabs(baz_ant_align)*100)+2048;
else
   baz_ant_align=baz_ant_align*100;
itoa((int)baz_ant_align,temp_word,2);
strrev(temp_word);
len=strlen(temp_word);
if(len<12)
   for(i=0;i<12-len;i++)
      strcat(temp_word,"0");
strcat(aux_data[3],temp_word);
if(ch==2){
   strcat(aux_data[3],baz_cs);
   baz_ant_hgt1=baz_ant_hgt;
   if(baz_ant_hgt< -63.0)
      baz_ant_hgt= -63.0;
   else if(baz_ant_hgt>63.0)
      baz_ant_hgt=63.0;
   if(baz_ant_hgt<0.0)
      baz_ant_hgt=fabs(baz_ant_hgt)+64;
   if(baz_ant_hgt<0)
      baz_ant_hgt=0;
   itoa((int)baz_ant_hgt,temp_word,2);
   strrev(temp_word);
   len=strlen(temp_word);
   if(len<7)
      for(i=0;i<7-len;i++)
         strcat(temp_word,"0");
   strcat(aux_data[3],temp_word);
   strcat(aux_data[3],"00000000");
}
else
   strcat(aux_data[3],"0000000000000000");
get_parity76(3);
if(baz_sta==0)
   for(i=0;i<76;i++)
      strcpy(&aux_data[3][i],"0");
get_aux_dpsk(3);
```

```
/*****************************************************************************/
/*                                PRINT HEADER                               */
/*****************************************************************************/

   clrscr();
   if(disp==1){
      gotoxy(35,10);
      printf("PLEASE WAIT");
      gotoxy(29,12);
      printf("PRINTING DOCUMENTATION");
      fprintf(stdprn,"%c%c",27,54); /*  set character set II  */
   }
   fprintf(output,"\r                                BASIC / AUX DATA WORD");
   fprintf(output,"\n\r                                    content for");
   fprintf(output,"\n\r                                M-%s located at",system_id
   strupr(locat);
   len=strlen(locat);
   len=40-(int)(len/2);
   fprintf(output,"\n\r");
   for(i=0;i<len;i++)
      fprintf(output,"\x20");
   fprintf(output,"%s",locat);
   len=strlen(remark);
   len=40-(int)(len/2);
   fprintf(output,"\n\r");
   for(i=0;i<len;i++)
      fprintf(output,"\x20");
   fprintf(output,"%s",remark);
   fprintf(output,"\n\r");
   if(baz_sta==0){
      for(i=0;i<29;i++)
         fprintf(output,"\x20");
      fprintf(output,"(%s-deg.AZ, %s-deg.EL)",az_beam1,el_beam1);
   }
   else{
      for(i=0;i<22;i++)
         fprintf(output,"\x20");
      fprintf(output,"(%s-deg.AZ, %s-deg.EL, %s-deg.BAZ)",az_beam1,
                        el_beam1,baz_beam1);
   }
   switch(ch){
      case 1: fprintf(output,"\n\r                                                AS P
FAA-STD-022c");
               break;
      case 2: fprintf(output,"\n\r                                                AS P
RTCA/DO-198");
               break;
   }
   fprintf(output,"\n\r");
   len=strlen(edate);
   len=35-(int)(len/2);
   for(i=0;i<len;i++)
```

```
      fprintf(output,"\x20");
   fprintf(output,"effective: %s\n\n",edate);

/*************************************************************************/
/*                          PRINT  BASIC DATA WORD # 1                   */
/*************************************************************************/

   fprintf(output,"\r\tBASIC DATA WORD 1: ");
   for(i=0;i<32;i++){
      if(i==5||i==12||i==18||i==23||i==28||i==29||i==30)
         fprintf(output," ");
      fprintf(output,"%c",basic_data[0][i]);
   }
   fprintf(output,"\n\r\t              DPSK: ");
   for(i=0;i<32;i++){
      if(i==5||i==12||i==18||i==23||i==28||i==29||i==30)
         fprintf(output," ");
   fprintf(output,"%c",dpsk[0][i]);
      }
   fprintf(output,"\n\r\t              CODE: ");
   print_b_code(0);
   fprintf(output,"\n\n\r\t       Preamble: ");
   fprintf(output,"11101 0101000");
   fprintf(output,"\n\r\t       AZ to threshold distance: %dm = ",az_thres2);
   for(i=12;i<18;i++)
      fprintf(output,"%c",basic_data[0][i]);
   fprintf(output," (%dm.)",az_thres1);
   fprintf(output,"\n\r\t       AZ proportional coverage: -%d° = ",az_neg1);
   for(i=18;i<23;i++)
      fprintf(output,"%c",basic_data[0][i]);
   fprintf(output,"\n\r\t       AZ proportional coverage: +%d° = ",az_pos1);
      for(i=23;i<28;i++)
         fprintf(output,"%c",basic_data[0][i]);
   fprintf(output,"\n\r\t       Clearance signal type: ");
   fprintf(output,"%s",clear==1?"scanning = 1":"pulse = 0");
   fprintf(output,"\n\r\t       Spare: ");
   fprintf(output,"%c",basic_data[0][29]);
   fprintf(output,"\n\r\t       Odd parity: ");
   fprintf(output,"%c%c",basic_data[0][30],basic_data[0][31]);
   if(disp==2){
      getch();
      fprintf(output,"\n");
   }
/*************************************************************************/
/*                          PRINT  BASIC DATA WORD # 2                   */
/*************************************************************************/

   fprintf(output,"\n\n\n\r\tBASIC DATA WORD 2: ");
   for(i=0;i<32;i++){
      if(i==5||i==12||i==19||i==20||i==22||i==23||i==24||i==30)
         fprintf(output," ");
      fprintf(output,"%c",basic_data[1][i]);
```

```
   }
   fprintf(output,"\n\r\t                DPSK: ");
   for(i=0;i<32;i++){
      if(i==5||i==12||i==19||i==20||i==22||i==23||i==24||i==30)
         fprintf(output," ");
   fprintf(output,"%c",dpsk[1][i]);
   }
   fprintf(output,"\n\r\t                CODE: ");
   print_b_code(1);
   fprintf(output,"\n\n\r\t        Preamble: ");
   fprintf(output,"11101 0111100");
   fprintf(output,"\n\r\t        Minium glide path: ");
   fprintf(output,"%s° = ",glide1);
      for(i=12;i<19;i++)
         fprintf(output,"%c",basic_data[1][i]);
   fprintf(output,"\n\r\t        BAZ status: ");
   if(baz_sta==0)
      fprintf(output,"none = %d",baz_sta);
   else
      fprintf(output,"available = %d",baz_sta);
   fprintf(output,"\n\r\t        DME status: ");
   if(!strncmp(dme_sta,"00",2))
      fprintf(output,"not available = %s",dme_sta);
   else if(!strncmp(dme_sta,"10",2))
      fprintf(output,"Only IA mode = %s",dme_sta);
   else if(!strncmp(dme_sta,"01",2))
      fprintf(output,"FA mode, Standard 1 = %s",dme_sta);
   else if(!strncmp(dme_sta,"11",2))
      fprintf(output,"FA mode, Standard 2 = %s",dme_sta);
   fprintf(output,"\n\r\t        AZ status: ");
   if(az_sta==1)
      fprintf(output,"OK = %d",az_sta);
   else
      fprintf(output,"Down = %d",az_sta);
   fprintf(output,"\n\r\t        EL status: ");
   if(el_sta==1)
      fprintf(output,"OK = %d",el_sta);
   else
      fprintf(output,"Down = %d",el_sta);
   fprintf(output,"\n\r\t        Spare: ");
   for(i=24;i<30;i++)
      fprintf(output,"%c",basic_data[1][i]);
   fprintf(output,"\n\r\t        Odd parity: ");
   fprintf(output,"%c%c",basic_data[1][30],basic_data[1][31]);
   if(disp==2){
      getch();
      fprintf(output,"\n");
   }
/*************************************************************************/
/*                          PRINT  BASIC DATA WORD # 3                   */
/*************************************************************************/
   fprintf(output,"\n\n\n\r\tBASIC DATA WORD 3: ");
```

```
   for(i=0;i<32;i++){
      if(i==5||i==12||i==15||i==18||i==27||i==30)
         fprintf(output," ");
      fprintf(output,"%c",basic_data[2][i]);
   }
   fprintf(output,"\n\r\t                DPSK: ");
   for(i=0;i<32;i++){
      if(i==5||i==12||i==15||i==18||i==27||i==30)
         fprintf(output," ");
      fprintf(output,"%c",dpsk[2][i]);
   }
   fprintf(output,"\n\r\t                CODE: ");
   print_b_code(2);
   fprintf(output,"\n\n\r\t        Preamble: ");
   fprintf(output,"11101 1010000");
   fprintf(output,"\n\r\t        AZ beamwidth: ");
   fprintf(output,"%s° = ",az_beam1);
   for(i=12;i<15;i++)
      fprintf(output,"%c",basic_data[2][i]);
   fprintf(output,"\n\r\t        EL beamwidth: ");
   fprintf(output,"%s° = ",el_beam1);
   for(i=15;i<18;i++)
      fprintf(output,"%c",basic_data[2][i]);
   fprintf(output,"\n\r\t        DME distance to MLS datum: ");
   fprintf(output,"%sm. = ",dme_dist2);
   for(i=18;i<27;i++)
      fprintf(output,"%c",basic_data[2][i]);
   if(dme_dist1>=12.5)
      fprintf(output," (%0.0fm)",dme_dist);
   else if(dme_dist1>=6.5&&dme_dist1<=12.5)
      fprintf(output," (12.5m)");
   else
      fprintf(output," (0.0m)");
   fprintf(output,"\n\r\t        Spare: ");
   for(i=27;i<30;i++)
      fprintf(output,"%c",basic_data[2][i]);
   fprintf(output,"\n\r\t        Odd parity: ");
   fprintf(output,"%c%c",basic_data[2][30],basic_data[2][31]);
   if(disp==2){
      getch();
      fprintf(output,"\n");
   }
/*************************************************************************/
/*                          PRINT  BASIC DATA WORD # 4                   */
/*************************************************************************/

   fprintf(output,"\n\n\n\r\tBASIC DATA WORD 4: ");
   for(i=0;i<32;i++){
      if(i==5||i==12||i==21||i==30)
         fprintf(output," ");
      fprintf(output,"%c",basic_data[3][i]);
   }
```

```
   fprintf(output,"\n\r\t              DPSK: ");
   for(i=0;i<32;i++){
      if(i==5||i==12||i==21||i==30)
         fprintf(output," ");
      fprintf(output,"%c",dpsk[3][i]);
      }
   fprintf(output,"\n\r\t              CODE: ");
    print_b_code(3);
   fprintf(output,"\n\n\r\t        Preamble: ");
   fprintf(output,"11101 1000100");
   fprintf(output,"\n\r\t        AZ 0-degree guidance plane: ");
   fprintf(output,"%d = ",az_zero1);
   for(i=12;i<21;i++)
      fprintf(output,"%c",basic_data[3][i]);
   fprintf(output,"\n\r\t        BAZ 0-degree guidance plane: ");
   fprintf(output,"%d = ",baz_zero1);
   for(i=21;i<30;i++)
      fprintf(output,"%c",basic_data[3][i]);
   fprintf(output,"\n\r\t        Odd parity: ");
   fprintf(output,"%c%c\n",basic_data[3][30],basic_data[3][31]);
   disp==2 ? getch() : fprintf(output,"\f");
/*********************************************************************/
/*                          PRINT  BASIC DATA WORD # 5                 */
/*********************************************************************/

   fprintf(output,"\r\tBASIC DATA WORD 5: ");
   if(baz_sta==1)
      for(i=0;i<32;i++){
         if(i==5||i==12||i==17||i==22||i==25||i==26||i==30)
            fprintf(output," ");
         fprintf(output,"%c",basic_data[4][i]);
      }
   else
      fprintf(output,"(not radiated)");
   fprintf(output,"\n\r\t              DPSK: ");
   if(baz_sta==1)
      for(i=0;i<32;i++){
   if(i==5||i==12||i==17||i==22||i==25||i==26||i==30)
      fprintf(output," ");
   fprintf(output,"%c",dpsk[4][i]);
      }
   fprintf(output,"\n\r\t              CODE: ");
   print_b_code(4);
   fprintf(output,"\n\n\r\t        Preamble: ");
   fprintf(output,"11101 1101100");
   fprintf(output,"\n\r\t        BAZ proportional coverage: ");
   if(baz_sta==1){
      fprintf(output,"-%d° = ",baz_neg1);
      for(i=12;i<17;i++)
   fprintf(output,"%c",basic_data[4][i]);
      }
   fprintf(output,"\n\r\t        BAZ proportional coverage: ");
```

```
   if(baz_sta==1){
      fprintf(output,"+%d° = ",baz_pos1);
      for(i=17;i<22;i++)
   fprintf(output,"%c",basic_data[4][i]);
      }
   fprintf(output,"\n\r\t       BAZ beamwidth: ");
   if(baz_sta==1){
      fprintf(output,"%s° = ",baz_beam1);
      for(i=22;i<25;i++)
   fprintf(output,"%c",basic_data[4][i]);
      }
   fprintf(output,"\n\r\t       BAZ status: ");
   if(baz_sta==1)
      fprintf(output,"%s %d",baz_sta5==1 ? "OK = ":"DOWN = ",baz_sta5);
   fprintf(output,"\n\r\t       Spare: ");
   if(baz_sta==1)
      fprintf(output,"0000");
   fprintf(output,"\n\r\t       Odd parity: ");
   if(baz_sta==1)
      fprintf(output,"%c%c",basic_data[4][30],basic_data[4][31]);
   if(disp==2)
      getch();
/*************************************************************************/
/*                          PRINT  BASIC DATA WORD # 6                   */
/*************************************************************************/

   fprintf(output,"\n\n\n\r\tBASIC DATA WORD 6: ");
   for(i=0;i<32;i++){
      if(i==5||i==12||i==18||i==24||i==30)
         fprintf(output," ");
      fprintf(output,"%c",basic_data[5][i]);
   }
   fprintf(output,"\n\r\t             DPSK: ");
   for(i=0;i<32;i++){
      if(i==5||i==12||i==18||i==24||i==30)
         fprintf(output," ");
      fprintf(output,"%c",dpsk[5][i]);
   }
   fprintf(output,"\n\r\t             CODE: ");
   print_b_code(5);
   fprintf(output,"\n\n\r\t       Preamble: ");
   fprintf(output,"11101 0001101");
   fprintf(output,"\n\r\t       System ID: ");
   fprintf(output,"%c = ",system_id[0]);
   for(i=12;i<18;i++)
      fprintf(output,"%c",basic_data[5][i]);
   fprintf(output,"\n\r\t                  %c = ",system_id[1]);
   for(i=18;i<24;i++)
      fprintf(output,"%c",basic_data[5][i]);
   fprintf(output,"\n\r\t                  %c = ",system_id[2]);
   for(i=24;i<30;i++)
      fprintf(output,"%c",basic_data[5][i]);
```

```
   fprintf(output,"\n\r\t       Odd parity: ");
   fprintf(output,"%c%c",basic_data[5][30],basic_data[5][31]);
   if(disp==2)
      getch();
/*****************************************************************************/
/*                              PRINT  AUX DATA WORD # 1                     */
/*****************************************************************************/

   fprintf(output,"\n\n\n\r\tAUX DATA WORD 1: ");
   for(i=0;i<43;i++){
      if(i==5||i==12||i==20||i==30)
         fprintf(output," ");
      fprintf(output,"%c",aux_data[0][i]);
   }
   fprintf(output,"\n\r\t\t\t ");
   for(i=43;i<76;i++){
      if(i==55||i==56||i==63||i==69)
         fprintf(output," ");
      fprintf(output,"%c",aux_data[0][i]);
   }
   fprintf(output,"\n\r\t           DPSK: ");
   for(i=0;i<43;i++){
      if(i==5||i==12||i==20||i==30)
         fprintf(output," ");
      fprintf(output,"%c",aux_dpsk[0][i]);
   }
   fprintf(output,"\n\r\t\t\t\t ");
   for(i=43;i<76;i++){
      if(i==55||i==56||i==63||i==69)
         fprintf(output," ");
      fprintf(output,"%c",aux_dpsk[0][i]);
   }
   fprintf(output,"\n\r\t           CODE: ");
   print_a_code(0);
   fprintf(output,"\n\n\r\t       Preamble: ");
   fprintf(output,"11101 1110010");
   fprintf(output,"\n\r\t       Address code: ");
   for(i=12;i<20;i++)
      fprintf(output,"%c",aux_data[0][i]);
   fprintf(output,"\n\r\t       AZ ant lateral offset: %0.1fm. = ",az_lo1);
   for(i=20;i<30;i++)
      fprintf(output,"%c",aux_data[0][i]);
   fprintf(output," (%c%0.0fm.)",(az_lo1<0) ? '-' : ' ',az_lo);
   fprintf(output,"\n\r\t       AZ to MLS datum distance: %0.1fm. = ",az_mls_dp1
   for(i=30;i<43;i++)
      fprintf(output,"%c",aux_data[0][i]);
   fprintf(output," (%0.0fm.)",az_mls_dp);
   fprintf(output,"\n\r\t             AZ  ant  alignment  w/  centerline:  %s°
",az_ant_align1);
   for(i=43;i<55;i++)
      fprintf(output,"%c",aux_data[0][i]);
   fprintf(output,"(%0.2f°)",az_ant_align);
```

```
   fprintf(output,"\n\r\t       AZ coordinate system: ");
   if(strcmp(az_cs,"0")==0)
      fprintf(output,"conical = 0");
   else if(strcmp(az_cs,"1")==0)
      fprintf(output,"planar = 1");
   if(ch==2){
      fprintf(output,"\n\r\t       AZ ant height: %0.2fm. = ",az_ant_hgt);
      for(i=56;i<63;i++)
         fprintf(output,"%c",aux_data[0][i]);
      fprintf(output," (%sm.)",az_ant_hgt1);
      fprintf(output,"\n\r\t       Spare: 000000");
   }
   else
      fprintf(output,"\n\r\t       Spare: 0000000000000");
   fprintf(output,"\n\r\t       Parity: ");
   for(i=69;i<76;i++)
      fprintf(output,"%c",aux_data[0][i]);
   check_id(0);
   if(disp==2)
      getch();
/*****************************************************************************/
/*                              PRINT  AUX DATA WORD # 2                     */
/*****************************************************************************/

   fprintf(output,"\n\n\n\r\tAUX DATA WORD 2: ");
   for(i=0;i<40;i++){
      if(i==5||i==12||i==20||i==30)
         fprintf(output," ");
      fprintf(output,"%c",aux_data[1][i]);
   }
   fprintf(output,"\n\r\t\t\t ");
   for(i=40;i<76;i++){
      if(i==47||i==60||i==67||i==69)
         fprintf(output," ");
      fprintf(output,"%c",aux_data[1][i]);
   }
   fprintf(output,"\n\r\t           DPSK: ");
   for(i=0;i<40;i++){
      if(i==5||i==12||i==20||i==30)
         fprintf(output," ");
      fprintf(output,"%c",aux_dpsk[1][i]);
   }
   fprintf(output,"\n\r\t\t\t ");
   for(i=40;i<76;i++){
      if(i==47||i==60||i==67||i==69)
         fprintf(output," ");
      fprintf(output,"%c",aux_dpsk[1][i]);
   }
   fprintf(output,"\n\r\t           CODE: ");
   print_a_code(1);
   fprintf(output,"\n\n\r\t       Preamble: ");
   fprintf(output,"11101 1110010");
```

```
   fprintf(output,"\n\r\t       Address code: ");
   for(i=12;i<20;i++)
      fprintf(output,"%c",aux_data[1][i]);
   fprintf(output,"\n\r\t       EL ant lateral offset: %sm. = ",el_lol);
   for(i=20;i<30;i++)
      fprintf(output,"%c",aux_data[1][i]);
   fprintf(output," (%0.0fm.)",el_lo);
   fprintf(output,"\n\r\t            MLS  datum  to  threshold  dist.:  %0.0fm.
",el_mls_dp);
   for(i=30;i<40;i++)
      fprintf(output,"%c",aux_data[1][i]);
   fprintf(output,"\n\r\t       EL height: %0.2fm. = ",el_ant_hgtl);
   for(i=40;i<47;i++)
      fprintf(output,"%c",aux_data[1][i]);
   fprintf(output," (%c%0.1fm.)",(el_ant_hgtl<0) ? '-': ' ',el_ant_hgt);
   if(ch==2){
      fprintf(output,"\n\r\t       MLS datum point elev.: %0.1f = ",mls_dp_elv)
      for(i=47;i<60;i++)
         fprintf(output,"%c",aux_data[1][i]);
      fprintf(output," (%c%dm.)",(mls_dp_elv<0) ? '-' : ' ',mls_dpe);
      fprintf(output,"\n\r\t                Runway  threshold  height:  %0.1f
",rwy_thr_hgt/10);
      for(i=60;i<67;i++)
         fprintf(output,"%c",aux_data[1][i]);
   }
   fprintf(output,"\n\r\t       Spare: ");
   for(i=((ch==2)?67:47);i<69;i++)
      fprintf(output,"%c",aux_data[1][i]);
   fprintf(output,"\n\r\t       Parity: ");
   for(i=69;i<76;i++)
      fprintf(output,"%c",aux_data[1][i]);
   check_id(1);
   fprintf(output,"\n");
   disp==2 ? getch() : fprintf(output,"\f");
/*************************************************************************/
/*                          PRINT  AUX DATA WORD # 3                     */
/*************************************************************************/

   fprintf(output,"\r\tAUX DATA WORD 3: ");
   for(i=0;i<((ch==2)?46:44);i++){
      if(i==5||i==12||i==20||i==((ch==2)?32:30))
         fprintf(output," ");
      fprintf(output,"%c",aux_data[2][i]);
   }
   fprintf(output,"\n\r\t\t\t ");
   for(i=((ch==2)?46:44);i<76;i++){
      if(i==((ch==2)?53:51)||i==67||i==69)
         fprintf(output," ");
      fprintf(output,"%c",aux_data[2][i]);
   }
   fprintf(output,"\n\r\t            DPSK: ");
   for(i=0;i<((ch==2)?46:44);i++){
```

```
      if(i==5||i==12||i==20||i==((ch==2)?32:30))
         fprintf(output," ");
      fprintf(output,"%c",aux_dpsk[2][i]);
   }
   fprintf(output,"\n\r\t\t\t\t ");
   for(i=((ch==2)?46:44);i<76;i++){
      if(i==((ch==2)?53:51)||i==67||i==69)
         fprintf(output," ");
      fprintf(output,"%c",aux_dpsk[2][i]);
   }
   fprintf(output,"\n\r\t            CODE: ");
   print_a_code(2);
   fprintf(output,"\n\n\r\t        Preamble: ");
   fprintf(output,"11101 1110010");
   fprintf(output,"\n\r\t        Address code: ");
   for(i=12;i<20;i++)
      fprintf(output,"%c",aux_data[2][i]);
   fprintf(output,"\n\r\t        DME lateral offset: %0.1fm. = ",dme_lo1);
   for(i=20;i<((ch==2)?32:30);i++)
      fprintf(output,"%c",aux_data[2][i]);
   fprintf(output," (%c%0.1fm.)",(dme_lo1<0) ? '-' : ' ',dme_lo);
   fprintf(output,"\n\r\t        DME to MLS datum distance: %sm. = ",dme_mls_dd1
   for(i=((ch==2)?32:30);i<((ch==2)?46:44);i++)
      fprintf(output,"%c",aux_data[2][i]);
   fprintf(output," (%dm.)",(int)dme_mls_dd);
   if(ch==2){
      fprintf(output,"\n\r\t        DME ant height: %0.1fm. = ",dme_ant_hgt1);
      for(i=((ch==2)?46:44);i<53;i++)
         fprintf(output,"%c",aux_data[2][i]);
      fprintf(output," (%c%0.0fm.)",(dme_ant_hgt1<0) ? '-' : ' ',dme_ant_hgt);
      fprintf(output,"\n\r\t        Runway stop-end dist.: %dm. = ",rwy_stop_dist
      for(i=53;i<67;i++)
         fprintf(output,"%c",aux_data[2][i]);
   }
   fprintf(output,"\n\r\t        Spare: ");
   for(i=((ch==2)?67:44);i<69;i++)
      fprintf(output,"%c",aux_data[2][i]);
   fprintf(output,"\n\r\t        Parity: ");
   for(i=69;i<76;i++)
      fprintf(output,"%c",aux_data[2][i]);
   check_id(2);
   if(disp==2)
      getch();
/*************************************************************************/
/*                          PRINT  AUX DATA WORD # 4                     */
/*************************************************************************/

   fprintf(output,"\n\n\n\r\tAUX DATA WORD 4: ");
   if(baz_sta==1){
      for(i=0;i<41;i++){
         if(i==5||i==12||i==20||i==30)
            fprintf(output," ");
```

```
            fprintf(output,"%c",aux_data[3][i]);
      }
      fprintf(output,"\n\r\t\t\t\t ");
      for(i=41;i<76;i++){
         if(i==53||i==54||i==61||i==69)
            fprintf(output," ");
            fprintf(output,"%c",aux_data[3][i]);
      }
   }
   else
      fprintf(output,"(not radiated)");
   fprintf(output,"\n\r\t            DPSK: ");
   if(baz_sta==1){
      for(i=0;i<41;i++){
         if(i==5||i==12||i==20||i==30)
            fprintf(output," ");
            fprintf(output,"%c",aux_dpsk[3][i]);
      }
      fprintf(output,"\n\r\t\t\t\t ");
         for(i=41;i<76;i++){
            if(i==53||i==54||i==61||i==69)
            fprintf(output," ");
            fprintf(output,"%c",aux_dpsk[3][i]);
      }
   }
   fprintf(output,"\n\r\t            CODE: ");
   print_a_code(3);
   fprintf(output,"\n\n\r\t        Preamble: ");
   fprintf(output,"11101 1110010");
   fprintf(output,"\n\r\t        Address code: ");
   if(baz_sta==1)
      for(i=12;i<20;i++)
         fprintf(output,"%c",aux_data[3][i]);
   fprintf(output,"\n\r\t        BAZ ant lateral offset: ");
   if(baz_sta==1){
      fprintf(output,"%0.1fm. = ",baz_lo1);
      for(i=20;i<30;i++)
         fprintf(output,"%c",aux_data[3][i]);
      fprintf(output," (%c%0.0fm.)",(baz_lo1<0) ? '-' : ' ',baz_lo);
   }
   fprintf(output,"\n\r\t        BAZ to MLS datum distance: ");
   if(baz_sta==1){
      fprintf(output,"%0.1fm. = ",baz_mls_dp1);
      for(i=30;i<41;i++)
         fprintf(output,"%c",aux_data[3][i]);
      fprintf(output," (%0.0fm.)",baz_mls_dp);
   }
   fprintf(output,"\n\r\t        BAZ ant alignment w/ centerline: ");
   if(baz_sta==1){
      fprintf(output,"%0.2f° = ",baz_ant_align1);
      for(i=41;i<53;i++)
         fprintf(output,"%c",aux_data[3][i]);
```

```
   }
   if(ch==2){
      fprintf(output,"\n\r\t       BAZ coordinate system: ");
      if(baz_sta==1){
         if(strcmp(baz_cs,"0")==0)
            fprintf(output,"conical = 0");
         else if(strcmp(baz_cs,"1")==0)
            fprintf(output,"planar = 1");
      }
      fprintf(output,"\n\r\t       BAZ ant height: ");
      if(baz_sta==1){
         fprintf(output,"%0.2fm. = ",baz_ant_hgt1);
         for(i=53;i<61;i++)
            fprintf(output,"%c",aux_data[3][i]);
         fprintf(output," (%c%dm.)",baz_ant_hgt1<0 ? '-': ' ',(int)baz_ant_hgt)
      }
   }
   fprintf(output,"\n\r\t       Spare: ");
   if(baz_sta==1)
      for(i=((ch==2)?61:53);i<69;i++)
         fprintf(output,"%c",aux_data[3][i]);
   fprintf(output,"\n\r\t       Parity: ");
   if(baz_sta==1)
      for(i=69;i<76;i++)
         fprintf(output,"%c",aux_data[3][i]);
   check_id(3);
   strupr(rom_name);
   fprintf(output,"\n\n\n\r\tROM File Name: %s",rom_name);
   disp==2 ? getch() : fprintf(output,"\f");
/*********************************************************************/

   for(i=0;i<90;i++){
      fprintf(fptr,"00");
      fprintf(fptr1,"00");
      cnt++;
      count();
   }

   strcpy(morse,"111011100000");
   for(i=0;i<3;i++){
      get_morse(system_id[i]);
      strcat(morse,"00000");
   }
   len=strlen(morse);
   if(len<64)
      for(i=0;i<64-len;i++)
        strcat(morse,"0");
   m_code();
   fprintf(fptr,"5B5B5B2179001A28000000000000\n");
   fprintf(fptr,"00000004000000654000000000777979\n");
   fprintf(fptr,"79797979797979797979F0F000000000\n");
   fprintf(fptr,"00000000000000000000000000025448\n");
```

```
   fprintf(fptr,"4953204953204D2D");
   fprintf(fptr1,"5B5B5B2179001A28000000000000\n");
   fprintf(fptr1,"00000004000000654000000000777979\n");
   fprintf(fptr1,"79797979797979797979F0F000000000\n");
   fprintf(fptr1,"00000000000000000000000000025448\n");
   fprintf(fptr1,"4953204953204D2D");
   for(i=0;i<3;i++){
      fprintf(fptr,"%02X",system_id[i]);
      fprintf(fptr1,"%02X",system_id[i]);
   }
   fprintf(fptr,"2028");
   fprintf(fptr,"415A20\n");
   fprintf(fptr,"292C20");
   fprintf(fptr1,"2028");
   fprintf(fptr1,"454C20\n");
   fprintf(fptr1,"292C20");
   cnt=3;
   strcat(locat,"  ^D");
   len=strlen(locat);
   for(i=0;i<len;i++){
      fprintf(fptr,"%02X",locat[i]);
      fprintf(fptr1,"%02X",locat[i]);
      cnt++;
      count();
   }
   while(cnt!=16){
      fprintf(fptr,"FF");
      fprintf(fptr1,"FF");
      cnt++;
   }
   fprintf(fptr,"\n");
   fprintf(fptr1,"\n");
   for(i=0;i<113;i++){
      fprintf(fptr,"FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF\n");
      fprintf(fptr1,"FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF\n");
   }
   fcloseall();
   clrscr();
} /*  end main  */

void count()
{
   if(cnt==16){
      fprintf(fptr,"\n");
      fprintf(fptr1,"\n");
      cnt=0;
   }
}
void strdel(char str[],int n)
{
   strcpy(&str[n],&str[n+1]);
}
```

```
int get_parity(int word,int bit)
{
   int t;

   if(bit==31)
      t=basic_data[word][12]+basic_data[word][13]+basic_data[word][14]
        +basic_data[word][15]+basic_data[word][16]+basic_data[word][17]
        +basic_data[word][18]+basic_data[word][19]+basic_data[word][20]
        +basic_data[word][21]+basic_data[word][22]+basic_data[word][23]
        +basic_data[word][24]+basic_data[word][25]+basic_data[word][26]
        +basic_data[word][27]+basic_data[word][28]+basic_data[word][29];
   else if(bit==32)
      t=basic_data[word][13]+basic_data[word][15]+basic_data[word][17]
        +basic_data[word][19]+basic_data[word][21]+basic_data[word][23]
        +basic_data[word][25]+basic_data[word][27]+basic_data[word][29];
   return(t);
}

void get_dpsk(int word)
{
   int i;

   if(basic_data[word][0]=='1')
      strcpy(&dpsk[word][0],"1");
   else
      strcpy(&dpsk[word][0],"0");
   for(i=1;i<strlen(basic_data[word]);i++){
      if(basic_data[word][i]=='1'){
         if(dpsk[word][i-1]=='1')
            strcpy(&dpsk[word][i],"0");
         else
            strcpy(&dpsk[word][i],"1");
      }
      else if(dpsk[word][i-1]=='1')
         strcpy(&dpsk[word][i],"1");
      else
         strcpy(&dpsk[word][i],"0");
   }
}

void get_aux_dpsk(int word)
{
   int i;

   if(aux_data[word][0]=='1')
      strcpy(&aux_dpsk[word][0],"1");
   else
      strcpy(&aux_dpsk[word][0],"0");
   for(i=1;i<strlen(aux_data[word]);i++){
      if(aux_data[word][i]=='1'){
         if(aux_dpsk[word][i-1]=='1')
```

```
            strcpy(&aux_dpsk[word][i],"0");
         else
            strcpy(&aux_dpsk[word][i],"1");
      }
      else if(aux_dpsk[word][i-1]=='1')
         strcpy(&aux_dpsk[word][i],"1");
      else
         strcpy(&aux_dpsk[word][i],"0");
   }
}

void get_bits(char ch,int index)
{
   int i;

   for(i=index;i<index+6;i++){
      strcpy(&basic_data[5][i] ,(1 & ch) ? "1" : "0");
      ch >>= 1;
   }
}

void get_parity76(int word)
{
   int i,res,t;

 /*   bit 70   */

   res=0;
   for(i=12;i<18;i++)
      if(aux_data[word][i]=='1') res++;
   for(i=19;i<24;i+=2)
      if(aux_data[word][i]=='1') res++;
   if(aux_data[word][24]=='1') res++;
   if(aux_data[word][27]=='1') res++;
   if(aux_data[word][28]=='1') res++;
   if(aux_data[word][28]=='1') res++;
   for(i=30;i<33;i++)
      if(aux_data[word][i]=='1') res++;
   if(aux_data[word][34]=='1') res++;
   if(aux_data[word][35]=='1') res++;
   if(aux_data[word][37]=='1') res++;
   if(aux_data[word][40]=='1') res++;
   for(i=43;i<46;i++)
      if(aux_data[word][i]=='1') res++;
   if(aux_data[word][49]=='1') res++;
   for(i=51;i<55;i++)
      if(aux_data[word][i]=='1') res++;
   if(aux_data[word][57]=='1') res++;
   if(aux_data[word][59]=='1') res++;
   if(aux_data[word][63]=='1') res++;
   if(aux_data[word][64]=='1') res++;
   strcpy(&aux_data[word][69],(fmod((double)res,2.0)>0)? "1" : "0");
```

```
/*   bit 71   */

  res=0;
  for(i=13;i<19;i++)
     if(aux_data[word][i]=='1') res++;
  for(i=20;i<24;i+=2)
     if(aux_data[word][i]=='1') res++;
  if(aux_data[word][24]=='1') res++;
  if(aux_data[word][25]=='1') res++;
  if(aux_data[word][28]=='1') res++;
  if(aux_data[word][29]=='1') res++;
  for(i=31;i<34;i++)
     if(aux_data[word][i]=='1') res++;
  if(aux_data[word][35]=='1') res++;
  if(aux_data[word][36]=='1') res++;
  if(aux_data[word][38]=='1') res++;
  if(aux_data[word][41]=='1') res++;
  for(i=44;i<47;i++)
     if(aux_data[word][i]=='1') res++;
  if(aux_data[word][50]=='1') res++;
  for(i=52;i<56;i++)
     if(aux_data[word][i]=='1') res++;
  if(aux_data[word][58]=='1') res++;
  if(aux_data[word][60]=='1') res++;
  if(aux_data[word][64]=='1') res++;
  if(aux_data[word][66]=='1') res++;
  strcpy(&aux_data[word][70],(fmod((double)res,2.0)>0)? "1" : "0");

   /*   bit 72   */

  res=0;
  for(i=14;i<20;i++)
     if(aux_data[word][i]=='1') res++;
  for(i=21;i<26;i+=2)
     if(aux_data[word][i]=='1') res++;
  if(aux_data[word][26]=='1') res++;
  if(aux_data[word][29]=='1') res++;
  if(aux_data[word][30]=='1') res++;
  for(i=32;i<35;i++)
     if(aux_data[word][i]=='1') res++;
  if(aux_data[word][36]=='1') res++;
  if(aux_data[word][37]=='1') res++;
  if(aux_data[word][39]=='1') res++;
  if(aux_data[word][42]=='1') res++;
  for(i=45;i<48;i++)
     if(aux_data[word][i]=='1') res++;
  if(aux_data[word][51]=='1') res++;
  for(i=53;i<57;i++)
     if(aux_data[word][i]=='1') res++;
  if(aux_data[word][59]=='1') res++;
  if(aux_data[word][61]=='1') res++;
```

```
  if(aux_data[word][65]=='1') res++;
  if(aux_data[word][66]=='1') res++;
  strcpy(&aux_data[word][71],(fmod((double)res,2.0)>0)? "1" : "0");

 /*   bit 73   */

  res=0;
  for(i=15;i<21;i++)
     if(aux_data[word][i]=='1') res++;
  for(i=22;i<27;i+=2)
     if(aux_data[word][i]=='1') res++;
  if(aux_data[word][27]=='1') res++;
  if(aux_data[word][30]=='1') res++;
  if(aux_data[word][31]=='1') res++;
  for(i=33;i<36;i++)
     if(aux_data[word][i]=='1') res++;
  if(aux_data[word][37]=='1') res++;
  if(aux_data[word][38]=='1') res++;
  if(aux_data[word][40]=='1') res++;
  if(aux_data[word][43]=='1') res++;
  for(i=46;i<49;i++)
     if(aux_data[word][i]=='1') res++;
  if(aux_data[word][52]=='1') res++;
  for(i=54;i<58;i++)
     if(aux_data[word][i]=='1') res++;
  if(aux_data[word][60]=='1') res++;
  if(aux_data[word][62]=='1') res++;
  if(aux_data[word][66]=='1') res++;
  if(aux_data[word][67]=='1') res++;
  strcpy(&aux_data[word][72],(fmod((double)res,2.0)>0)? "1" : "0");

  /*   bit 74   */

  res=0;
  for(i=16;i<22;i++)
     if(aux_data[word][i]=='1') res++;
  for(i=23;i<28;i+=2)
     if(aux_data[word][i]=='1') res++;
  if(aux_data[word][28]=='1') res++;
  if(aux_data[word][31]=='1') res++;
  if(aux_data[word][32]=='1') res++;
  for(i=34;i<37;i++)
     if(aux_data[word][i]=='1') res++;
  if(aux_data[word][38]=='1') res++;
  if(aux_data[word][39]=='1') res++;
  if(aux_data[word][41]=='1') res++;
  if(aux_data[word][44]=='1') res++;
  for(i=47;i<50;i++)
     if(aux_data[word][i]=='1') res++;
  if(aux_data[word][53]=='1') res++;
  for(i=55;i<59;i++)
     if(aux_data[word][i]=='1') res++;
```

```
   if(aux_data[word][61]=='1') res++;
   if(aux_data[word][63]=='1') res++;
   if(aux_data[word][67]=='1') res++;
   if(aux_data[word][68]=='1') res++;
   strcpy(&aux_data[word][73],(fmod((double)res,2.0)>0)? "1" : "0");

   /*   bit 75   */

   res=0;
   for(i=12;i<17;i++)
      if(aux_data[word][i]=='1') res++;
   for(i=18;i<23;i+=2)
      if(aux_data[word][i]=='1') res++;
   if(aux_data[word][23]=='1') res++;
   if(aux_data[word][26]=='1') res++;
   if(aux_data[word][27]=='1') res++;
   for(i=29;i<32;i++)
      if(aux_data[word][i]=='1') res++;
   if(aux_data[word][33]=='1') res++;
   if(aux_data[word][34]=='1') res++;
   if(aux_data[word][36]=='1') res++;
   if(aux_data[word][39]=='1') res++;
   for(i=42;i<45;i++)
      if(aux_data[word][i]=='1') res++;
   if(aux_data[word][48]=='1') res++;
   for(i=50;i<54;i++)
      if(aux_data[word][i]=='1') res++;
   if(aux_data[word][56]=='1') res++;
   if(aux_data[word][58]=='1') res++;
   if(aux_data[word][62]=='1') res++;
   if(aux_data[word][63]=='1') res++;
   if(aux_data[word][68]=='1') res++;
   strcpy(&aux_data[word][74],(fmod((double)res,2.0)>0)? "1" : "0");

 /*   bit 76   */

   res=0;
   for(i=12;i<75;i++)
      if(aux_data[word][i]=='1') res++;
   strcpy(&aux_data[word][75],(fmod((double)res,2.0)>0)? "1" : "0");


}

void print_b_code(int word)
{
   int i,j1;
   unsigned char byte[9];

   j1=0;
   for(i=0;i<8;i++){
      byte[i]=0;
```

```
      if(dpsk[word][j1]=='1')
         byte[i]+=1;
      if(dpsk[word][j1+1]=='1')
         byte[i]+=2;
      if(dpsk[word][j1+2]=='1')
         byte[i]+=4;
      if(dpsk[word][j1+3]=='1')
         byte[i]+=8;
      j1+=4;
   }

   fprintf(output,"%01X%01XH,%01X%01XH,%01X%01XH,%01X%01XH",byte[1],byte[0],
                                                            byte[3],byte[2],
                                                            byte[5],byte[4],
                                                            byte[7],byte[6]);
   for(i=0;i<7;i+=2){
      fprintf(fptr,"%01X%01X",byte[i+1],byte[i]);
      fprintf(fptr1,"%01X%01X",byte[i+1],byte[i]);
      cnt++;
      count();
   }
}

void print_a_code(int word)
{
   int i,j1,byte[21];

   j1=0;
   for(i=0;i<20;i++){
      byte[i]=0;
      if(aux_dpsk[word][j1]=='1')
         byte[i]+=1;
      if(aux_dpsk[word][j1+1]=='1')
         byte[i]+=2;
      if(aux_dpsk[word][j1+2]=='1')
         byte[i]+=4;
      if(aux_dpsk[word][j1+3]=='1')
         byte[i]+=8;
      j1+=4;
   }
   byte[19]=0;
   fprintf(output,"%01X%01XH,%01X%01XH,%01X%01XH,%01X%01XH,%01X%01XH,%01X%01XH,
                  byte[1],byte[0],byte[3],byte[2],byte[5],byte[4],byte[7],
                  byte[6],byte[9],byte[8],byte[11],byte[10]);

   fprintf(output,"%01X%01XH,%01X%01XH,%01X%01XH,%01X%01XH",byte[13],byte[12],
                   byte[15],byte[14],byte[17],byte[16],byte[19],byte[18]);

   for(i=0;i<19;i+=2){
      fprintf(fptr,"%01X%01X",byte[i+1],byte[i]);
      fprintf(fptr1,"%01X%01X",byte[i+1],byte[i]);
      cnt++;
```

```
      count();
   }

}

void m_code()
{
   int i,j1,byte[17];

   j1=0;
   for(i=0;i<16;i++){
      byte[i]=0;
      if(morse[j1]=='1')
         byte[i]+=1;
      if(morse[j1+1]=='1')
         byte[i]+=2;
      if(morse[j1+2]=='1')
         byte[i]+=4;
      if(morse[j1+3]=='1')
         byte[i]+=8;
      j1+=4;
   }
   byte[16]=0;
   for(i=0;i<16;i+=2){
      fprintf(fptr,"%01X%01X",byte[i+1],byte[i]);
      fprintf(fptr1,"%01X%01X",byte[i+1],byte[i]);
      cnt++;
      count();
   }
}

void get_morse(char ch)
{
   int i,shft,temp;
   char bit[1];
   unsigned int mask;

   mask=0x0001;

   switch(ch){
      case 'A' : temp=0x1D; shft=5; break;
      case 'B' : temp=0x157; shft=9; break;
      case 'C' : temp=0x5D7; shft=11;break;
      case 'D' : temp=0x57;shft=7;break;
      case 'E' : temp=0x1;shft=1;break;
      case 'F' : temp=0x175;shft=9;break;
      case 'G' : temp=0x177;shft=9;break;
      case 'H' : temp=0x55;shft=7;break;
      case 'I' : temp=0x5;shft=3;break;
      case 'J' : temp=0x1DDD;shft=13;break;
      case 'K' : temp=0x1D7;shft=9;break;
      case 'L' : temp=0x15D;shft=9;break;
```

```
      case 'M' : temp=0x77;shft=7;break;
      case 'N' : temp=0x17;shft=5;break;
      case 'O' : temp=0x777;shft=11;break;
      case 'P' : temp=0x5DD;shft=11;break;
      case 'Q' : temp=0x1D77;shft=13;break;
      case 'R' : temp=0x5D;shft=7;break;
      case 'S' : temp=0x15;shft=5;break;
      case 'T' : temp=0x7;shft=3;break;
      case 'U' : temp=0x75;shft=7;break;
      case 'V' : temp=0x1D5;shft=9;break;
      case 'W' : temp=0x1DD;shft=9;break;
      case 'X' : temp=0x757;shft=11;break;
      case 'Y' : temp=0x1DD7;shft=13;break;
      case 'Z' : temp=0x577;shft=11;break;
   }
   for(i=0;i<shft;i++){
      strcpy(bit,(mask & temp)?"1":"0");
      strcat(morse,bit);
      mask <<=1;
   }
}
void check_id(int wrd)
{
   char *pos;

   if(pos=strstr(aux_data[wrd],"111010011001"))
      fprintf(output,"\n\rCAUTION !!! CONTAINS: APPROACH AZIMUTH ID AT POSITI
%u",(pos-&aux_data[wrd])+1);
   if(pos=strstr(aux_data[wrd],"111010010100"))
      fprintf(output,"\n\rCAUTION !!! CONTAINS: HIGH RATE APPROACH AZIMUTH ID
POSITION %u",(pos-&aux_data[wrd])+1);
   if(pos=strstr(aux_data[wrd],"111011100001"))
      fprintf(output,"\n\rCAUTION !!! CONTAINS: APPROACH ELEVATION ID AT POSITI
%u",(pos-&aux_data[wrd])+1);
   if(pos=strstr(aux_data[wrd],"111011001001"))
      fprintf(output,"\n\rCAUTION !!! CONTAINS: BACK AZIMUTH ID AT POSITI
%u",(pos-&aux_data[wrd])+1);
   if(pos=strstr(aux_data[wrd],"111010101000"))
      fprintf(output,"\n\rCAUTION !!! CONTAINS: BASIC DATA 1 ID AT POSITI
%u",(pos-&aux_data[wrd])+1);
   if(pos=strstr(aux_data[wrd],"111010111100"))
      fprintf(output,"\n\rCAUTION !!! CONTAINS: BASIC DATA 2 ID AT POSITI
%u",(pos-&aux_data[wrd])+1);
   if(pos=strstr(aux_data[wrd],"111011010000"))
      fprintf(output,"\n\rCAUTION !!! CONTAINS: BASIC DATA 3 ID AT POSITI
%u",(pos-&aux_data[wrd])+1);
   if(pos=strstr(aux_data[wrd],"111011000100"))
      fprintf(output,"\n\rCAUTION !!! CONTAINS: BASIC DATA 4 ID AT POSITI
%u",(pos-&aux_data[wrd])+1);
   if(pos=strstr(aux_data[wrd],"111011101100"))
      fprintf(output,"\n\rCAUTION !!! CONTAINS: BASIC DATA 5 ID AT POSITI
%u",(pos-&aux_data[wrd])+1);
```

```
   if(pos=strstr(aux_data[wrd],"111010001101"))
      fprintf(output,"\n\rCAUTION  !!!  CONTAINS:  BASIC  DATA  6  ID  AT  POSITI
%u",(pos-&aux_data[wrd])+1);
}
```