MODIS
SCIENCE DATA SUPPORT TEAM
PRESENTATION

February 7, 1992

AGENDA

1. Action Items
2. MODIS Airborne Simulator (MAS)
3. Programming Code Review
4. SDST Schedule Discussion

ACTION ITEMS:

08/30/91 [Lloyd Carpenter and Team]: Draft a schedule of work for the next 12 months. Include primary events and milestones, documents to be produced, software development, MAS support, etc. (The schedule will be discussed at the meeting.) STATUS: Open. Due date 09/27/91.

12/06/91 [Liam Gumley]: Investigate a cataloguing scheme for the MAS data. Consider the Master Catalogue, PLDS and PCDS. STATUS: Open. Due date 02/14/92.

12/06/91 [Liam Gumley, Tom Goff, Ed Masuoka]: Develop a plan for storing and distributing MAS data. STATUS: Open. Due date 02/14/92.

01/03/92 [Ed Masuoka]: Check on the UCAR "copyright" as a first step in standardizing an SDST software copyright statement for code sharing. Check with legal. (Legal is developing the statement.) STATUS: Open. Due date 02/14/92.

01/03/92 [Team]: Check on the set of software engineering tools available in Code 530 to see if any of these would be of use to the SDST. STATUS: Open. Due date 02/14/92.

01/17/92 [Tom Goff]: Have a polished version (with peer review) of the file dump routine ready for the MODIS Science Team Meeting. STATUS: Open. Due date 04/01/92.

MODIS Airborne Simulator status (Liam Gumley)

Progress up to 6 February 1992

(1) MAS data processing status

Flight Date	Area covered during flight	MAS data received	INS data received	Processing status
11/12/91	Ferry flight to Houston	yes (subset)	yes	Complete (1 track)
11/14/91	Coffeyville, Kansas	yes	yes	Complete (16 tracks)
11/18/91	Coffeyville, Kansas	yes	yes	
11/21/91	Coffeyville, Kansas	yes	yes	
11/22/91	Coffeyville, Kansas	yes	yes	
11/24/91	Gulf coast, Texas/Louisiana	yes	yes	
11/25/91	Coffeyville, Kansas	yes	yes	
11/26/91	Coffeyville, Kansas	yes	yes	
12/03/91	Gulf coast, Texas/Louisiana	yes	yes	
12/04/91	Gulf coast, Texas/Louisiana	yes	yes	
12/05/91	Coffeyville, Kansas	yes	yes	21 of 29 tracks done
12/07/91	Coffeyville, Kansas	yes	yes	
11/16/91	Ground visible calibration	yes	-	
11/20/91	Ground visible calibration	yes	-	
11/23/91	Ground visible calibration	yes	-	

(2) MAS/INS clock synchronization

During this week, it was discovered that there was a discrepancy between the times recorded by the MAS and INS clocks. This was evidenced by turn data appearing at the end of processed flight tracks, when the turn data should not have been used. Examination of both the MAS and INS roll data showed that the MAS was recording the aircraft (and hence the imagery) rolling before the INS data recorded a roll (i.e. turning out of a straight line). It should be noted that neither the MAS nor INS roll data have been used until now in routine processing.

The plot attached overleaf shows the INS roll data lagging behind the MAS roll data by about 80 seconds. The data shown is INS data at full time resolution (sampled every 5 seconds) and the MAS data (sampled every 0.16 seconds) at the nearest whole second. The time period shown is from the middle of the flight and spans 358 seconds.

A cross correlation algorithm was coded and used to check whether the time lag could be detected analytically. The next plot attached shows the cross correlation for time lags from 0 to 335 seconds. The maximum correlation of approximately 0.97 is seen at a time lag of 80 seconds, in agreement with the previous plot. Interpolation of this data would allow the determination of the time lag more accurately.

Jeff Myers at Ames was called regarding this problem, and he said that a time offset is possible only if the MAS data system and the INS are running off their own separate internal clocks. In normal conditions, both the MAS and the INS receive time information from a GOES receiver on board the aircraft. However apparently during FIRE this system was not always available, and the MAS and INS ran on separate clocks for some missions. Jeff said there is a code in the INS data stream that indicates either GOES or internal clock usage (ASCII 'G' for GOES, ASCII 'I' for Internal). The MAS data stream contains no indication

as to which clock is used. A sample portion of the INS data set from 05-DEC-91 is attached which shows the internal clock code ('I') in the second column. This can be used to indicate the need to search for time offsets in the MAS vs INS clock time.

The problem was also discussed with Chris Moeller at Wisconsin. Chris has seen this problem before occasionally, however it is not that significant for his meteorological applications. In any case, it always shows up as a misalignment between the navigated MAS imagery and an overlaid map. This is corrected by simply adding a bias to the latitude and/or longitudes for that flight track. Chris therefore has no real need to look for this specific problem.

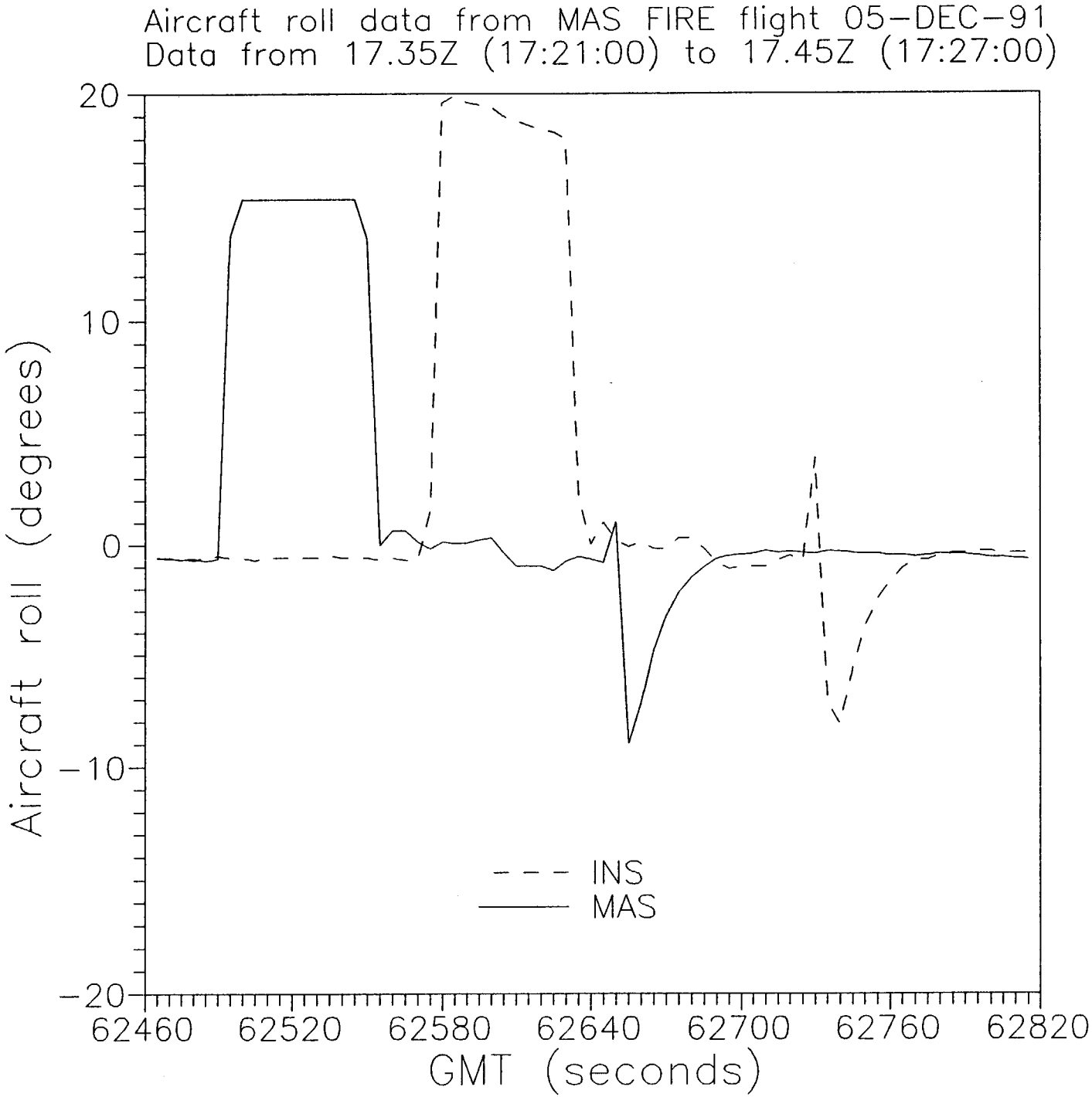
The question may well be asked "How do we know that either clock is correct?". It is important to realize that even though the INS clock is "wrong" (fast or slow), the INS location data recorded is still accurate (Ames has confirmed this via phone 2/6/92), albeit at the wrong time. Now since the INS data can be matched to the MAS data by using the aircraft roll information, it should always be possible to line up the MAS data stream with the INS data stream to better than 5 seconds accuracy (assuming interpolation as mentioned earlier). This can be done independent of the INS time, as long as the INS is still recording data every 5 seconds. This would allow the best possible geolocation accuracy, and the MAS time would have to be assumed to be correct.

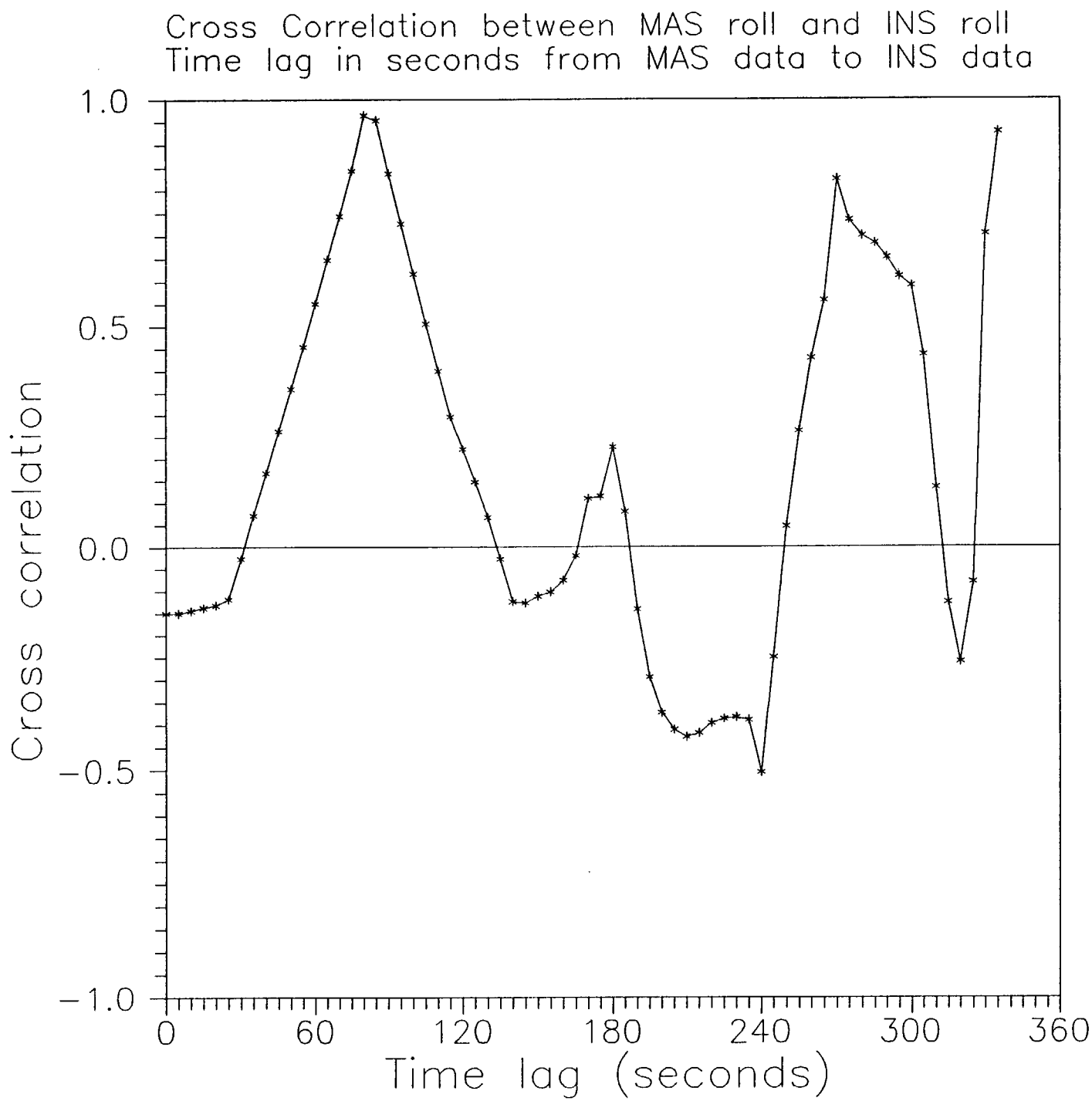
(3) MAS sensitivity changes with decreasing temperature

In the SDST handout of 2/24/92, it was mentioned that some work on characterizing the change in sensitivity of the MAS visible/near-IR channels with temperature would be worthwhile. Ames has now done some cold chamber tests of the MAS, and a subset of the results are plotted overleaf. Significant changes in the sensitivity of MAS FIRE channels 3, 4, 5 and 6 are seen. This gives further emphasis to the need for in-flight characterization of these sensitivity changes.

(4) MAS Users Group Meeting

An MAS Users Group Meeting will be held today (2/7/92) at 2:00pm in Room 271, Building 22.

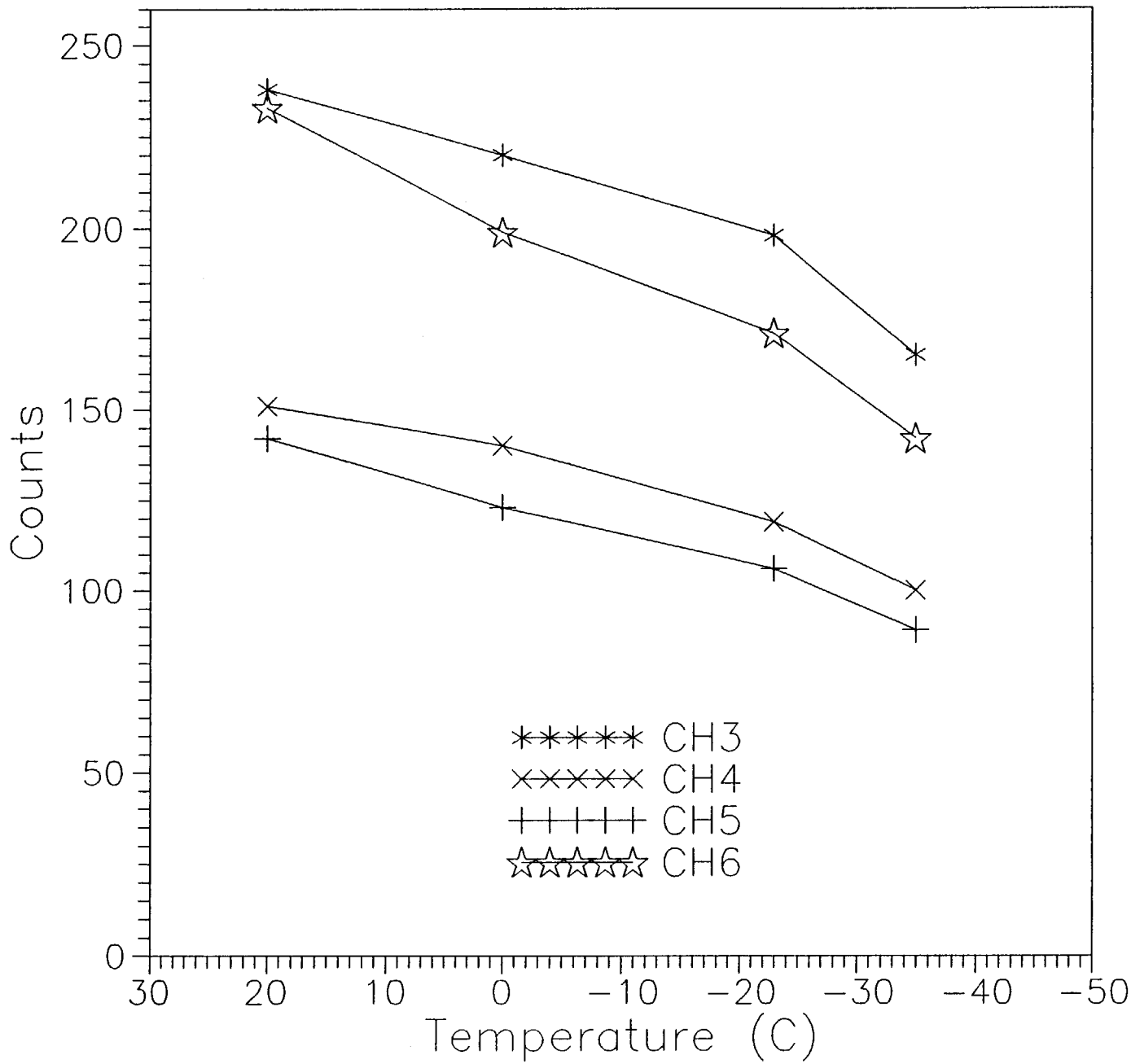

Aircraft roll data from MAS FIRE flight 05-DEC-91
Data from 17.35Z (17:21:00) to 17.45Z (17:27:00)
Aircraft roll (degrees)
20
10
0
-10
-20
62460
62520
62580
62640
62700
62760
62820
GMT (seconds)
INS
MAS


Cross Correlation between MAS roll and INS roll
Time lag in seconds from MAS data to INS data
1.0
0.5
0.0
−0.5
−1.0
Cross correlation
0
60
120
180
240
300
360
Time lag (seconds)

FLIGHT 92-036 DATE 12/05/91 A/C 709 TAKEOFF 14:30/05 LAND 21:15/05 RECORDER UNIT NUMBER 2

(GMT) T I M E DAY HH:MM:SS/DD	TIME	LATITUDE DEG MIN	LONGITUDE DEG MIN	ALTITUDE FEET	TRUE AIR SPD KNOT	PITCH DEG	ROLL DEG	TRUE HEADG DEG	N-S VEL KNOT	E-W VEL KNOT	VERT VEL FPS	GRND SPD KNOT	COMPUTED GROUND SPEED KNOT	WIND SPEED KNOT	WIND ANGLE DEG	VERT VEL FPS	COMPUTED ANGLE OF ATTACK DEG	OUTSIDE AIR TEMPERATURE DEGREE C
14:32:50/05	I	N29:35.8	W095:10.1	374	140	20.8	.7	174.1	-118	-0		119	119	26	34			-.2
14:32:55/05	I	N29:35.5	W095:10.1	824	145	20.4	-.5	174.1	-119	-3		120	120	31	35	90	22	-.2
14:33:00/05	I	N29:35.5	W095:10.1	1271	149	20.1	-.6	174.3	-121	-4		122	122	33	34	89	21	-.2
14:33:05/05	I	N29:35.2	W095:10.1	1730	152	18.7	1.1	175.7	-122	-3		124	124	34	26	92	21	-.2
14:33:10/05	I	N29:34.5	W095:10.8	2183	156	18.2	.8	178.6	-124	-5		127	127	32	15	91	20	-.2
14:33:15/05	I	N29:34.5	W095:10.8	2622	160	17.9	.3	180.1	-127	-8		128	128	34	13	88	19	-.2
14:33:20/05	I	N29:34.2	W095:10.8	3063	160	16.4	.3	182.5	-128	-11		130	130	33	5	88	19	-.2
14:33:25/05	I	N29:34.2	W095:10.8	3467	165	15.0	-.4	184.5	-130	-11		134	134	35	357	81	17	-.2
14:33:30/05	I	N29:33.9	W095:10.8	3855	170	14.1	-4.1	185.2	-134	-13		140	140	35	356	78	16	-.2
14:33:35/05	I	N29:33.9	W095:10.8	4243	177	14.9	-4.2	184.9	-139	-11		144	144	37	353	78	15	-.2
14:33:40/05	I	N29:33.5	W095:10.8	4679	180	17.8	-5.3	183.5	-144	-5		146	146	36	351	87	17	-.2
14:33:45/05	I	N29:33.2	W095:10.8	5185	179	17.7	-2.8	181.9	-146	3		142	142	34	345	101	20	-.2
14:33:50/05	I	N29:33.2	W095:10.8	5653	164	17.2	-7.2	179.6	-142	11		144	144	24	337	94	20	-.2
14:33:55/05	I	N29:32.9	W095:10.8	6065	158	16.3	-9.1	173.9	-144	14		146	146	13	11	82	18	-.2
14:34:00/05	I	N29:32.9	W095:10.8	6436	159	16.4	-10.4	169.9	-144	26		147	147	13	9	74	16	-.2
14:34:05/05	I	N29:32.5	W095:10.8	6809	160	14.9	-11.0	165.1	-141	41		150	150	13	2	75	16	-.2
14:34:10/05	I	N29:32.2	W095:10.8	7154	161	13.4	-11.4	159.9	-139	57		154	154	12	355	69	15	-.2
14:34:15/05	I	N29:32.2	W095:10.5	7469	162	12.0	-13.1	154.7	-136	74		161	161	11	336	63	13	-.2
14:34:20/05	I	N29:31.9	W095:10.5	7742	168	10.7	-11.4	147.6	-134	92		171	171	8	346	55	11	-.2
14:34:25/05	I	N29:31.9	W095:10.1	8010	176	9.1	.9	140.6	-126	115		179	179	10	340	54	10	-.2
14:34:30/05	I	N29:31.5	W095:10.1	8281	183	8.7	-.1	139.8	-127	128		189	189	17	325	54	10	-.2
14:34:35/05	I	N29:31.5	W095:09.8	8552	193	10.4	.1	139.8	-135	133		196	196	15	326	54	11	-.2
14:34:40/05	I	N29:31.2	W095:09.8	8887	198	12.5	.5	140.5	-142	136		199	199	15	317	67	12	-.2
14:34:45/05	I	N29:30.9	W095:09.5	9308	199	14.5	-.6	141.4	-144	137		197	197	17	311	84	15	-.2
14:34:50/05	I	N29:30.9	W095:09.2	9776	196	15.2	-.8	141.0	-145	134		194	194	13	305	94	16	-.2
14:34:55/05	I	N29:30.6	W095:08.8	10211	192	14.9	-.7	141.3	-141	132		191	191	15	306	87	16	-.2
14:35:00/05	I	N29:30.6	W095:08.8	10629	189	14.5	-.7	141.7	-139	131		189	189	17	305	84	15	-.2
14:35:05/05	I	N29:30.2	W095:08.5	11035	184	14.9	-3.0	142.3	-137	130		187	187	19	297	81	15	-.2
14:35:10/05	I	N29:30.2	W095:08.5	11447	183	14.6	-2.8	141.6	-135	129		184	184	18	297	82	16	-.2
14:35:15/05	I	N29:29.9	W095:08.2	11806	178	13.8	-.8	140.0	-130	131		184	184	18	291	72	14	-.2
14:35:20/05	I	N29:29.6	W095:07.8	12151	181	12.8	-.9	139.5	-127	133		184	184	19	302	69	13	-.2
14:35:25/05	I	N29:29.6	W095:07.8	12501	185	11.3	.2	139.9	-127	133		186	186	20	315	70	13	-.2
14:35:30/05	I	N29:29.2	W095:07.5	12839	188	10.6	-.3	140.5	-128	135		189	189	23	317	68	12	-.2
14:35:35/05	I	N29:29.2	W095:07.2	13148	188	10.1	.1	140.3	-131	136		192	192	21	310	62	11	-.2
14:35:40/05	I	N29:28.9	W095:07.2	13422	193	9.4	.8	140.7	-134	138		196	196	22	314	55	11	-.2
14:35:45/05	I	N29:28.9	W095:06.8	13710	199	9.1	.8	141.5	-138	140		201	201	23	318	58	11	-.2
14:35:50/05	I	N29:28.6	W095:06.5	14010	203	9.8	.7	142.3	-143	141		204	204	24	315	60	10	-.2
14:35:55/05	I	N29:28.6	W095:06.5	14349	204	11.5	-1.4	143.2	-148	141		205	205	25	310	68	11	-.2
14:36:00/05	I	N29:28.2	W095:06.2	14728	209	11.3	-1.6	142.4	-149	141		204	204	21	321	76	12	-.2
14:36:05/05	I	N29:27.9	W095:05.9	15102	210	12.2	-1.4	142.4	-147	141		203	203	23	325	75	12	-.2
14:36:10/05	I	N29:27.9	W095:05.9	15502	210	12.1	-1.3	142.1	-146	142		201	201	24	326	80	13	-.2
14:36:15/05	I	N29:27.6	W095:05.5	15891	210	11.9	-1.3	142.5	-143	142		199	199	28	329	78	13	-.2
14:36:20/05	I	N29:27.6	W095:05.2	16259	206	11.5	-1.8	142.3	-141	142		198	198	27	325	74	12	-.2
14:36:25/05	I	N29:27.3	W095:05.2	16600	204	10.8	-.2	142.0	-139	142		198	198	27	324	68	11	-.2
14:36:30/05	I	N29:26.9	W095:04.9	16918	202	10.0	-.7	142.5	-138	142		199	199	30	320	64	11	-.2
14:36:35/05	I	N29:26.9	W095:04.5	17227	203	8.1	-1.1	142.1	-139	142		202	202	27	320	62	10	-5.6
14:36:40/05	I	N29:26.6	W095:04.2	17477	205	7.5	-1.9	141.7	-142	145		207	207	26	318	50	8	-.2
14:36:45/05	I	N29:26.6	W095:04.2	17733	209	6.8	-3.2	140.2	-144	148		212	212	22	319	51	8	-.2
14:36:50/05	I	N29:26.3	W095:03.9	17968	211	6.1	-3.9	138.4	-147	154		219	219	18	307	47	8	-.2
14:36:55/05	I	N29:26.3	W095:03.6	18186	218	5.1	.5	136.6	-148	161		226	226	15	311	44	7	-.2
14:37:00/05	I	N29:25.9	W095:03.2	18392	223	5.4	.7	136.9	-152	168		233	233	19	308	41	6	-.2
14:37:05/05	I	N29:25.6	W095:03.2	18613	228	5.8	1.0	137.7	-158	172		239	239	21	300	44	7	-.2
14:37:10/05	I	N29:25.6	W095:02.9	18845	232	6.8	1.6	138.9	-164	174		244	244	24	296	46	7	-.2
14:37:15/05	I	N29:25.3	W095:02.6	19134	235	8.2	1.2	139.6	-170	175		245	245	24	290	58	8	-.2
14:37:20/05	I	N29:25.0	W095:02.2	19466	236	9.4	1.6	140.0	-174	173		245	245	22	290	67	11	-8.2

MAS FIRE configuration Ames cold chamber test data
Source is light box with spectralon panel 29-JAN-92
NOTE: No change observed in CH2 (data from Pat Grant)



Programming Code Reviews
Thomas E. Goff
6 February, 1992

tgoff on GSFC mail,
teg@LTPIRIS2.GSFC.NASA.GOV,
or (301) 982-3704

* **Programming Coding Practices** - We at RDC have had an informal internal review (PDR) of the fdump utility in order to establish a more formal procedure for reviews of final MODIS code to be generated in the future. From these preliminary reviews, a guide of suggested and manditory items to be included in a check list of code written by ourselves will be generated. At this point, external reviews within the MODIS SDST would be desirable.

* **Automatic scripting language** - As mentioned previously, I would like to obtain the Perl language and manuals for evaluation as a tool to produce automatic documentation from source listings. Access to this manual and other manuals will be necessary to perform our assigned tasks.

* **Remote communications to Goddard** - We at RDC have obtained the ncsa telnet program for PC's and a beta release of a slip protocol program to allow direct communications with the various machines at Goddard. The Goddard network people have been contacted to obtain an IP address for our PC's so we can test the feasibility of dial-up lines for more capability with screen editors, etc. This could lead to full X-window support via phone access. A political problem at Goddard exists because slip via a PC serial port has no interface card (hardware address), no rib manager (not connected to a rib), and no fixed location (dial up from anywhere). These are being worked out but will require a policy decision at GSFC networking to become reality. Contact has been made with the network support people who have provided access to the NCCS slip connection thereby allowing us to test the various networking connections.

DISCUSSION ITEMS

* A partial listing of the file dumping utility (FDUMP) is included for comments on the in-line documentation.

c:\modis\status.wp

```
/*Header:       FDUMP.C   a Generic File Dumping Utility                h*/

/*Version: 1.1, Version 1, Revision 1, 4 February 1992                  v*/

/*Legaleze: <my version until a genuine version is approved>

      Copyleft 1992, National Aeronautics and Space Administration,
      Goddard Space Flight Center, MODIS Science Data Support Team

    This document is considered to be of no value and therefore can be
 freely distributed without due compensation.  This software is provided
 without support and without obligation on the part of NASA/GSFC/MODIS/SDST
 to assist in its use, correction, modification, or enhancement.  Permission
 to use, copy, modify, and distribute this software and its documentation
 for any purpose without fee is hereby granted.  We request that this
 permission notice appear in supporting documentation, that the name of
 NASA/GSFC/MODIS/SDST not be used in advertising or publicity pertaining
 to the distribution of the software, and that all modifications to this
 software be documented and delimited by comments within this source listing.
 NASA/GSFC/MODIS/SDST makes no representations about the suitability of
 this software for any purpose.

    THIS SOFTWARE IS PROVIDED "AS IS" WITH NO EXPRESS OR IMPLIED WARRANTIES
 OF ANY KIND INCLUDING WARRANTIES OF DESIGN, MERCHANTIBILITY AND FITNESS FOR
 A PARTICULAR PURPOSE, OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE
 PRACTICE.

                                                                         c*/

/*Descriptor:
    If you have ever had unknown characters in your files that can not be
    displayed by a terminal server, or were added by file transfer programs
    among differing computers, then here is some help!  This program will
    display the contents of a file in a mnemonic format.  In other words,
    Byte characters will be displayed in an understandable form.  Other
    data formats can also be selected to display the file contents.
    The user has the option of selecting the beginning byte location
    within the file and the length of the dump in Bytes.  The data within
    the file can be displayed in the user's choice of ASCII, EBCDIC,
    hexidecimal, octal, integer (16 and 32 bit), floating point (32 or
    64 bit), or the special mnemonic character-equivalent format.  This
    mnemonic is handy for finding characters that are not normally
    displayable, such as <CR> for carrage returns or <XOFF> for the
    xoff (control-s or DC1) character.

    See also: the companion program: replace.c, which can fix up these
      undisplayable characters by replacing (for example) a <CR><LF> pair
      by a <LF> character, or replace all <HT> (tab) character by <b>
      (blanks). This program allows user input in mnemonic form.

    See also: a program to display all user input characters in mnemonic
      form: showchar.c is also available.  This program is highly machine
      dependent
                                                                         d*/

/*Author, sponsor:

    National Aeronautical and Space Administration (NASA)
    Goddard Space Flight Center (GSFC)
    Moderate Resolution Imaging Spectrometer (MODIS)
    Science Data Support Team (SDST)
    code 920.0, Greenbelt, MD, 20771

    Thomas E. Goff, Research and Data Systems Corporation
    7855 Walker Drive, Suite 460, Greenbelt, MD, 20770
    (301) 982-3704, Tgoff on GSFC Mail
    tegaltpiris2.gsfc.nasa.gov

    Provided under the auspices of the NASA EOS MODIS instrument
    Science Data Support Team, and used as an example of portable
    C code, stylistically structured, with self documenting procedures.

    Please report any bugs, enhancements, comments, machine ports, or
    whatever to the author at either address.
                                                                         a*/

/*USAGE:
```

```
    fdump [-<options>] [+<start_Byte>] [#<how_many_Bytes>] <file_name>
      where: -<options> can be:
             h for a hex dump (default)
             d for a decimal dump of each Byte
             o for an octal dump of each Byte
             a for an ASCII dump
             e for an EBCDIC dump
             m for the special mnemonic dump
             i16 (or i*2) for a 16 bit signed integer dump (2 Bytes)
             i32 (or i*4) for a 32 bit signed integer dump (4 Bytes)
             f32 (or r*4) for a 32 bit floating point dump (4 Bytes)
             f64 (or r*8) for a 64 bit floating point dump (8 Bytes)

             +<start_Byte> is the optional offset from the beginning of
                the file to begin dumping (0 is the first byte)

             #<how_many_Bytes> is the optional number of Bytes to dump

         and <file_name> is the file to be dumped.

   NOTES:
      - extended characters (ASCII > 127) are displayed with a leading ^
          and may create more than 80 characters per line on your display.
      - don't forget to escape special characters such as the * character
          from your shell (command processor).  (The method for escaping
          a character from special interpretation is machine dependent.)
          See the examples below:

   EXAMPLES: fdump -m #1000 fdump.c
                  to see the record terminators in the first 1000 Bytes
                  of this source file.
             fdump '-r*4' +100 myfile.dat
                  to see the file contents of myfile starting at Byte 100
                  in floating point (real) 32 bit form on a UNIX system.
             fdump `-i16` data.wk!
                  to dump a file in integer format on a PC running MS-DOS 5.0
                  and the 4DOS command.sys interpreter.

   COMPLETION CODES:
         0 - normal completion
        -1 - usage exit
        -2 - illegal number of files in the command line
        -3 = illegal option combinations
        -4 = cannot open the input file
        -5 = file size is smaller than the requested starting byte offet
   -100... = internal errors, bugs in the code
                                                                        u*/
/*Tested machines:  <This program has been tested on the following
                        computers, using the following compilers:>
         SGA personal IRIS V.3
           cc -o fdump fdump.c
         DEC VAX VMS 5.3
           cc fdump
           link fdump.obj
         PC clones with PowerC 2.1.2
           pc /e fdump
         PC clones with Borland C++ 1.0
           (IDE)
         Sun Sparc2, SUN 4.1
           cc -o fdump fdump.c
                                                                        t*/
/*Keys: < key words > file; dump; display functions; file dump;         k*/

/*Revision:    < revision history in reverse order >
    TEG, 31 Jan 92 - added r*4, r*8, i*2, and i*4 dump options
    TEG, 15 Jan 92 - clean up of comments & copyrights
    TEG, Dec 91 - initial release
                                                                        r*/
/*Bugs: < bugs / future enhancements >\
      - look into the i32 dump when performed on PC's: cannot get a
          true 32 bits, only 16 bits. Maybe a memory model problem!
      - add mneumonics for ASCII extended character set: 128 - 255
      - add a blocked output format option: i.e. for 3x3 matrix.
      - video enhancements (inverse video) for mnemonic would be a
          nice touch (machine dependent).
```

```
      - *** add a Unicode/ISO character dumping option. ***
                                                                    b*/
#include <stdio.h>
#include <ctype.h>

#define HUGE 2147483647          /* the largest int number on this machine */
                               /* (use 32767 for PC's , 2147483647 otherwise) */
#define BufferSize 8640          /* This is a 'magic' number (n*2160) */
                                 /* hint: lcd of {60,16,10,9,8,6,5,4} */
#define HEX 0x01
#define DEC 0x02
#define OCT 0x03
#define ASC 0x10
#define EBC 0x04            /* these are the internal flags for display modes */
#define MNU 0x20
#define I16 0x30
#define I32 0x40
#define F32 0x50
#define F64 0x60

unsigned char ebcdic2ascii[256] = {      /* ASCII to EBCDIC conversion chart */
      0,   1,   2,   3, 156,   9, 134, 127, 151, 141, 142,  11,
     12,  13,  14,  15,  16,  17,  18,  19, 157, 133,   8, 135,
     24,  25, 146, 143,  28,  29,  30,  31, 128, 129, 130, 131,
    132,  10,  23,  27, 136, 137, 138, 139, 140,   5,   6,   7,
    144, 145,  22, 147, 148, 149, 150,   4, 152, 153, 154, 155,
     20,  21, 158,  26,  32, 160, 161, 162, 163, 164, 165, 166,
    167, 168,  91,  46,  60,  40,  43,  33,  38, 169, 170, 171,
    172, 173, 174, 175, 176, 177,  93,  36,  42,  41,  59,  94,
     45,  47, 178, 179, 180, 181, 182, 183, 184, 185, 124,  44,
     37,  95,  62,  63, 186, 187, 188, 189, 190, 191, 192, 193,
    194,  96,  58,  35,  64,  39,  61,  34, 195,  97,  98,  99,
    100, 101, 102, 103, 104, 105, 196, 197, 198, 199, 200, 201,
    202, 106, 107, 108, 109, 110, 111, 112, 113, 114, 203, 204,
    205, 206, 207, 208, 209, 126, 115, 116, 117, 118, 119, 120,
    121, 122, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219,
    220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231,
    123,  65,  66,  67,  68,  69,  70,  71,  72,  73, 232, 233,
    234, 235, 236, 237, 125,  74,  75,  76,  77,  78,  79,  80,
     81,  82, 238, 239, 240, 241, 242, 243,  92, 159,  83,  84,
     85,  86,  87,  88,  89,  90, 244, 245, 246, 247, 248, 249,
     48,  49,  50,  51,  52,  53,  54,  55,  56,  57, 250, 251,
    252, 253, 254, 255 };

char *mneu[] = {                          /* ASCII to Mnemonic character array */
                "<NUL>","<SOH>","<STX>","<ETX>","<EOT>",
                "<ENQ>","<ACK>","<BEL>","<BS>", "<HT>",
                "<LF>", "<VT>", "<FF>", "<CR>", "<SO>",
                "<SI>", "<DLE>","<XON>","<DC2>","<XOFF>",
                "<DC4>","<NAK>","<SYN>","<ETB>","<CAN>",
                "<EM>", "<SUB>","<ESC>","<FS>", "<GS>",
                "<RS>", "<US>", " ",    "!",    "\"",
                "#",    "$",    "%",    "&",    "\'",
                "\(",   "\)",   "\*",   "\+",   "\,",
                "\-",   ".",    "/",    "0",    "1",
                "2",    "3",    "4",    "5",    "6",
                "7",    "8",    "9",    ":",    ";",
                "<",    "=",    ">",    "?",    "@",
                "A",    "B",    "C",    "D",    "E",
                "F",    "G",    "H",    "I",    "J",
                "K",    "L",    "M",    "N",    "O",
                "P",    "Q",    "R",    "S",    "T",
                "U",    "V",    "W",    "X",    "Y",
                "Z",    "\[",   "\\",   "\]",   "^",
                "_",    "`",    "a",    "b",    "c",
                "d",    "e",    "f",    "g",    "h",
                "i",    "j",    "k",    "l",    "m",
                "n",    "o",    "p",    "q",    "r",
                "s",    "t",    "u",    "v",    "w",
                "x",    "y",    "z",    "{",    "|",
                "}",    "~",    "<DEL>"                 };

union {                                   /* file contents data structure */
        unsigned char buff[BufferSize];
        short sbuff[BufferSize/2];
```

```
        long lbuff[BufferSize/4];
        float fbuff[BufferSize/4];
        double dbuff[BufferSize/8];
} *p ;

main(argc,argv)
int argc;                                   /* the number of command arguments */
char **argv;                                     /* array of command arguments */
{
 char filename[256];                                 /* the file to be dumped */
 FILE *in;                                             /* system file pointer */
 char flag;                                       /* user requested dump format */
 int argf;                           /* number of arguments successfully parsed */
 int startB = 0;                         /* user requested starting Byte offset */
 int numB = 0;                        /* user requested number of Bytes to dump */
 register int i;                                       /* local generic index */

 flag = 0x00;                               /* internal flag for the dump mode */

 if(argc == 1)                                   /* check for only one argument */
 {
   usage_and_die();
   exit(-1);
 }

 flag |= HEX;

 argf=1;
 for( i=1; i<argc; i++)
 {                                      /* let's look for command line arguments */
   if(argv[i][0] == '-')
   {                                          /* look for - options */
     ++argf;
     switch (argv[i][1])
     {
     case 'h':
         flag = HEX;
         break;
     case 'd':
         flag = DEC;
         break;
     case 'o':
         flag = OCT;
         break;
     case 'a':
         flag = ASC;
         break;
     case 'e':
         flag = EBC;
         break;
     case 'm':
         flag = MNU;
         break;
     case 'i':
         if((argv[i][2] == '1') && (argv[i][3] == '6'))
         {
           flag = I16 ;
           break;
         }
         if((argv[i][2] == '3') && (argv[i][3] == '2'))
         {
           flag =I32 ;
           break;
         }
         if((argv[i][2] == '*') && (argv[i][3] == '2'))
         {
           flag = I16;
           break;
         }
         if((argv[i][2] == '*') && (argv[i][3] == '4'))
         {
           flag = I32;
           break;
         }
     case 'f':
```

```
        if((argv[i][2] == '3') && (argv[i][3] == '2'))
        {
          flag = F32 ;
          break;
        }
        if((argv[i][2] == '6') && (argv[i][3] == '4'))
        {
          flag =F64 ;
          break;
        }
    case 'r':
        if((argv[i][2] == '*') && (argv[i][3] == '4'))
        {
          flag = F32;
          break;
        }
        if((argv[i][2] == '*') && (argv[i][3] == '8'))
        {
          flag = F64;
          break;
        }

    default: printf("\n*** Unknown option: %s ??\n",argv[1]);
             usage_and_die();
             exit(-1);
    }                       /* end if( - options ) */

                              /* check for illegal option combinations   */
    if((argv[i][2] == 'e') && (argv[i][3] == 0) && (flag & 0x03)) flag |= EBC;

    else if( argv[i][2]!=  0 && argv[i][2]!='a' && flag!=I16 &&
                flag!=I32 &&       flag!=F32 && flag!=F64 )
    {
      printf("\n*** Illegal combination: %s ??\n",argv[1]);
      usage_and_die();
      exit(-1);
    }
  }                       /* end for(i) */
  if(argv[i][0] == '+')
  {                           /* check for starting Byte request */
    ++argf;
    if ( (sscanf(argv[i]+1,"%d",&startB) == 0) || (startB < 0) )
    {
      printf("\n*** Invalid starting Byte specified: %d ?\n",startB);
      usage_and_die();
      exit(-1);
    }
  }                       /* end if ( + options ) */
  if(argv[i][0] == '#')
  {                              /* check for Byte dump limit */
    ++argf;
    if ( (sscanf(argv[i]+1,"%d",&numB) == 0) || (numB <= 0) )
    {
      printf("\n*** Invalid number of Bytes specified: %d ?\n",numB);
      usage_and_die();
      exit(-1);
    }
  }                       /* end if ( # options ) */
}
if((argc-argf) > 1)
{                       /* check for valid file specification */
  printf("\n*** Only one file at a time - please\n");
  usage_and_die();
  exit(-2);
}
strcpy(filename,argv[argc-1]);

if((flag & ASC) && (flag & EBC))
{
  printf("\n*** Cannot do ASCII & EBCDIC together\n");
  usage_and_die();
  exit(-3);
}

in = fopen(filename,"rb");        /* (file must be opened in binary mode) */
```

```
 if(in == NULL)
 {
   printf("\nCannot open: '%s' for read. Exiting...\n",filename);
   exit(-4);
 }

 xdump(in,flag,startB,numB);
 fclose(in);
 exit(0);
}                    /* end of fdump */

/*------------------------------------------------------------ Usage And Die ---*/

usage_and_die()
{
 printf("\nUsage: fdump");
 printf(" [-h|d|o|a|e|m|i16|i32|f32|f64, [+nnn [#nnn]]] <file>\n");
 printf("       -h = hex dump (default)\n");
 printf("       -d = decimal dump\n");
 printf("       -o = octal dump\n");
 printf("       -a = ASCII dump\n");
 printf("       -e = EBCDIC dump\n");
 printf("       -m = mneumonic ASCII dump\n");
 printf("     -i16 = 'short' conversion\n");
 printf("     -i32 = 'long' conversion\n");
 printf("     -f32 = 'float' conversion\n");
 printf("     -f64 = 'double' conversion\n");
 printf("       +nnn = starting offset in Bytes\n");
 printf("       #nnn = length of dump in Bytes\n\n");
 printf("   Note: d/h/o include an ASCII dump,\n");
 printf("     de/he/oe include an EBCDIC dump,\n");
 printf("     a and e CANNOT be used together,\n");
 printf("     and -option precedence applies\n");
 exit(-1);
}                    /* end of Usage_and_die */

/*------------------------------------------------------------------ XDump ---*/

xdump(fp, flag, startB, numB)
FILE *fp;                                                  /* file pointer */
char flag;                                                    /* dump type */
int startB;                                          /* starting Byte offset */
int numB;                                          /* number of Bytes to dump */
{
 char str[100];                    /* first output string */
 char chr[100];                    /* second output string */
 char tmp[50];                     /* temporary string */
 register int i;                   /* indices */
 register int j;                   /* indices */
 int offset;                       /* printed byte offset from startB */
 int remainder;                    /* residue bytes to align buff w/ file */
 int step = 0;                     /* number of items per CRT line */
                                     /* this is also used as a flag for */
                                     /* the output display mode */
 int numread = 0;                  /* number of bytes from file read */
 int len;                          /* length in bytes of input buffer */
 int BytesLeft;                    /* number of remaining bytes to dump */
 int Bpw = 1;                      /* number of bytes per word */

 BytesLeft = numB;
 strcpy(str,"illegal flag");

                    /* Let's do the listing headers first */
 if((flag & 0x03) == HEX)
 {
   step = 16;
   sprintf(str," Offset                        Hex Value                          ");
   if(flag & EBC) strcat(str,"EBCDIC\n");
   else strcat(str,"ASCII\n");
   sprintf(chr," -------  ----------------------------------------------   ----------------\n");
 }
 if((flag & 0x03) == DEC)
 {
   step = 10;
   sprintf(str,"  Offset                     Decimal Value                    ");
```

```
  if(flag & EBC) strcat(str,"EBCDIC\n");
  else strcat(str,"ASCII\n");
  sprintf(chr," ---------     ------------------------------------     ----------\n");
}
if((flag & 0x03) == OCT)
{
  step = 8;
  sprintf(str,"    Offset                    Octal Value                  ");
  if(flag & EBC) strcat(str,"EBCDIC\n");
  else strcat(str,"ASCII\n");
  sprintf(chr," ------------     ------------------------------       --------\n");
}
if(flag == ASC)
{
  step = 60;
  sprintf(str,"  Offset                              ASCII Value\n");
  sprintf(chr," ---------    ------------------------------------------------------------\n");
}
if(flag == EBC)
{
  step = 60;
  sprintf(str,"  Offset                              EBCDIC Value\n");
  sprintf(chr," ---------    ------------------------------------------------------------\n");
}
if(flag == MNU)
{
  step = 1;
  sprintf(str,"  Offset                          Mneumonic ASCII equivalents\n");
  sprintf(chr," ---------     ------------------------------------------------------------\n");
}
if(flag == I16)
{
  step = 9; Bpw = 2;
  sprintf(str,"  Offset                            16 bit integers\n");
  sprintf(chr," ---------     ------------------------------------------------------------\n");
}
if(flag == I32)
{
  step = 5; Bpw = 4;
  sprintf(str,"  Offset                            32 bit integers\n");
  sprintf(chr," ---------     ------------------------------------------------------------\n");
}
if(flag == F32)
{
  step = 6; Bpw = 4;
  sprintf(str,"  Offset                            32 bit floats\n");
  sprintf(chr," ---------     ------------------------------------------------------------\n");
}
if(flag == F64)
{
  step = 4; Bpw = 8;
  sprintf(str,"  Offset                            64 bit floats\n");
  sprintf(chr," ---------     ------------------------------------------------------------\n");
}
printf("%s",str);
printf("%s",chr);
                        /* pull file reads until the starting Byte is found */
len = sizeof(p->buff);
p = (void *)malloc(BufferSize);
for(j=0; j < startB/len; j++)                          /* page up to startB */
  numread = fread(p->buff, sizeof(char), len, fp);
remainder = startB % len;
if(remainder > 0)                                    /* align buff w/ startB */
  numread = fread(p->buff, sizeof(char), remainder, fp);

if(startB > j*len + numread)
{
  printf("\n*** The file is smaller than %d Bytes\n",startB);
  exit(-5);
}
offset = startB;                        /* initialize printed offset counter */

for(j = 0; j < 100; j++)
  str[j] = 0;
```

```
                        /* Now do the actual file dumping:
      This is performed in two sections: mnemonic and other non-mnemonic. The
      mnemonic dump has a variable number of output characters per line, the
      other dumps are fixed. */
i=0;          /* index into char array, incremented for each character */
j=0;          /* index into output array, incremented by mnemonic size */
if(BytesLeft == 0)
  BytesLeft = HUGE;

if(flag == MNU)
{                        /* BEGIN the special Mnemonic dump */
  while(BytesLeft)
  {
    BytesLeft-- ;                 /* this is the number of bytes remaining */
    if(i == 0)
    {                        /* preload the buffer (more bytes to process) */
      numread = fread(p->buff, sizeof(char), len, fp);
      if(numread == NULL)
      {                                   /* EOF encountered */
        printf("%s\n",str);
        return;
      }
      if((numread < len) && (numread < BytesLeft))BytesLeft=numread-1;
    }
    if(j == 0)
    {                    /* begin a new output line */
      sprintf(str,"%10d     ",offset);
      j += 15;
    }
    if(p->buff[i] & 0x80)
    {                    /* check for extended (>127) characters */
      p->buff[i] ^= 0x80;   /* (exclusive or) */
      strcat(str,"^");
      j++;
    }
    strcat(str,mneu[p->buff[i]]);
    j += strlen(mneu[p->buff[i]]);
    if(j > 75)
    {                    /* start a new output line */
      printf("%s\n",str);
      for(j = 0; j < 100; j++) str[j]=0;
      j=0;
    }
    i++; if(i == len)i=0;
    offset++ ;
  }
  printf("%s\n",str);
}                        /* END of special mnemonic dump */
else
{                        /* BEGIN all other file dumping types */
                         /* (the type is determined by the step parameter) */

  do  /* while(numread/Bpw == sizeof(p->buff)) */
  {
    numread = fread(p->buff, sizeof(char), len, fp);
    if(numread == NULL)
    {
      printf("%s\n",str);
      return;
    }
    if ( (BytesLeft != 0) && (BytesLeft < len) ) numread = numB ;
    BytesLeft -= len;

    for(i = 0; i < numread/Bpw; i+=step)
    {                        /* i is the input buffer index */
      switch (step)          /* j is the within line index */
      {
      case 16:
          sprintf(str,"%7x   ",offset);
          break;
      case  4:
      case  5:
      case  6:
      case  9:
      case 10:
```

```
        sprintf(str,"%10d    ",offset);
        break;
    case 8:
        sprintf(str,"%12o      ",offset);
        break;
    case 60:
        sprintf(str,"%10d    ",offset);
        break;
    default:
        errtrap(1);
        break;
    }          /* end switch(step) */

    for(j = 0; (j < step) && (j+i < numread/Bpw); j++)
    {                        /* print the file content values.. */
      switch (step)
      {
      case 16:
          sprintf(tmp,"%2x ",p->buff[i+j]);
          strcat(str,tmp);
          break;
      case  4:
          sprintf(tmp,"%14.6E ",p->dbuff[i+j]);
          strcat(str,tmp);
          break;
      case  5:
          sprintf(tmp,"%11d ",(long)p->lbuff[i+j]);
          strcat(str,tmp);
          break;
      case  6:
          sprintf(tmp,"%10.3e ",p->fbuff[i+j]);
          strcat(str,tmp);
          break;
      case  9:
          sprintf(tmp,"%6hd ",p->sbuff[i+j]);
          strcat(str,tmp);
          break;
      case 10:
          sprintf(tmp,"%3d ",p->buff[i+j]);
          strcat(str,tmp);
          break;
      case 8:
           sprintf(tmp,"%3o ",p->buff[i+j]);
          strcat(str,tmp);
          break;
      case 60:
          if(flag & EBC) p->buff[i+j] = ebcdic2ascii[p->buff[i+j]];
          if(isprint(p->buff[i+j]))
          {
            sprintf(tmp,"%c",p->buff[i+j]);
            strcat(str,tmp);
          }
          else strcat(str,".");
          break;
      default:
          errtrap(2);
          break;
      }              /* end switch(step) */
    }                /* end for(j) ...to be continued */

    for(; j < step; j++)
    {                        /* print blanks for missing values */
      switch (step)
      {
      case 16:
          strcat(str,"   ");
          break;
      case 10:
          strcat(str,"    ");
          break;
      case 8:
          strcat(str,"    ");
          break;
      default:
          errtrap(3);
```

```
            break;
        }               /* end switch(step) */
      }                 /* end for(j ...continued) */

      switch (step)
      {                         /* add spacing blanks */
      case 16:
          strcat(str,"  ");
          break;
      case 10:
          strcat(str,"    ");
          break;
      case 8:
          strcat(str,"      ");
          break;
      default:
          errtrap(4);
          break;
      }                 /* end switch(step) */

      for(j = 0; (j < step) && (j+i < numread); j++)
      {                         /* print ASCII or EBCDIC at right margin */
        if( step==8 || step==10 || step==16 )
        {
          if(flag & EBC) p->buff[i+j] = ebcdic2ascii[p->buff[i+j]];
          if(isprint(p->buff[i+j]))
          {
            sprintf(tmp,"%c",p->buff[i+j]);
            strcat(str,tmp);
          }
         else strcat(str,".");
        }                       /* endif step */
      }                         /* end for(j) */
      printf("%s\n",str);
      offset += step;
    }                           /* end for(i) */
  } while(numread == sizeof(p->buff));
 }                          /* END of other file dumping types */
}                                       /* return from xdump */

/*------------------------------------------------------------------ errTrap ---*/
errtrap(number)
int number;                 /* error code number */
{
  printf("An error was trapped - number: %d\n",number);
  exit(-100-number);
}
```