McIDAS

Man computer Interactive Data Access System

JULY 1985
Updated June 1987

DATA STRUCTURES

University of Wisconsin-Madison **SPACE SCIENCE AND ENGINEERING CENTER** 1225 W. Dayton St. Madison, Wis.

PREFACE

McIDAS Data Structures provides programmers with a guide to the data bases found on McIDAS. This manual provides basic information on structure, access, and use of most data files on the system, as well as to the more generalized underlying structures.

Chapter 1 provides information about the most basic level of files on McIDAS, the LW (Large Word array) file. The four more specific structures -- MD (Meteorological Data) files, areas, codicils, and grids, are discussed in Chapter 2. Chapters 3, 4, and 5 present the structures of individual data files.

TABLE OF CONTENTS

LW FILES

INTRODUCTION

The basic level of disk file access on McIDAS is the LW (Large Word array) file. Specialized file structures, including MD files, grids, and codicils, are superimposed upon the LW structure.

LW files are named collections of data viewed by the software as ordered arrays of up to 838Ø416 words (over 32M bytes), numbered from Ø. They are treated as large virtual arrays (i.e., contiguous blocks) of 4-byte words to which disk space is allocated dynamically, as needed. While the size of a file is potentially very large, it only takes up as much space as it actually needs at any given time. Because of this feature, LW files may also be referred to as virtual address space. There can be large gaps within a file that are empty (appear to be filled with the missing data code Z8Ø8Ø8Ø8Ø) and do not take up disk space.

McIDAS space is a collection of 1Ø24-word physical disk records called sectors residing on one or more disk volumes. Sectors have full word addresses. The left-most byte gives the volume number (Ø-3), and the right-most three bytes give the relative sector number (Ø-65535) within the volume. These sectors are treated abstractly as a single entity out of which are carved a large number of LW files.

LW files are composed of pages, units of 1024 words within the virtual address space. A page can contain data, mapped onto a sector in McIDAS space by means of the file's page tables, or it can be empty, not residing anywhere on disk but appearing to be full of missing data codes.

The status of each page recorded in a page table is null if the page is empty. Otherwise the page table contains the actual sector address of the data in the page.

To hold the page tables to exactly one sector in length, a file is divided into extents of maximum length 1Ø23*1Ø24=1Ø47552 words. A file can have up to eight extents to achieve its maximum size. It will usually have fewer because an extent has physical existence only if it contains data. A file has one page table for each extent that contains data.

PAGE TABLE STRUCTURE

Each page table is 1Ø24 words long:

Words	Contents
-1 to -1Ø24	page table for extent -1 of file (1st extent)
-1Ø25 to -2Ø48	page table for extent -2 of file (2nd extent)
.	.
.	.
.	.
-7169 to -8192	page table for extent -8 of file (last extent)

The first word of the table (word number -1Ø24 for the 1st table) records the last sector employed by this extent. It is used to decide when to allocate more space to the file. If this word is Z8Ø8Ø8Ø8Ø, then the file extent does not actually exist (no data). The second word gives the sector address of the first page of this extent, the third gives the sector address of the second page, etc.

MCTOC STRUCTURE

MCTOC is a single-extent LW file containing the McIDAS Table Of Contents. The first two pages are the disk free space table and the next eleven pages are the filename tables.

Words	Contents
Ø-2047	disk free space table
2Ø48-3Ø71	first filename table
3Ø72-4Ø95	second filename table
.	.
.	.
.	.
12288-13311	eleventh filename table

The page table for MCTOC's one and only extent is located at absolute sector address 1. Subroutine LWINIT, which resides in the scanner, reads this sector and picks up the address of the disk free space table and the filename tables.

DISK FREE SPACE TABLE STRUCTURE

Sectors are allocated in granules of sixteen sectors. Each granule is represented by a bit in the table. There are eight subtables of 256 words each (one per volume):

Words	Contents
Ø-255	subtable for volume Ø
56-511	subtable for volume 1
12-767	subtable for volume 2
68-1Ø23	subtable for volume 3
1024-1279	subtable for volume 4
1280-1535	subtable for volume 5
1536-1791	subtable for volume 6
1792-2047	subtable for volume 7

Each word of each subtable employs the rightmost 16 bits to record the status of 16 granules on that volume. Bit=1 means the granule has been allocated to some file and is unavailable. Bit=Ø means the granule is free and can be allocated. The first word controls the first sixteen granules, the second word controls the next sixteen granules, etc. Within each word, the lowest numbered granule is represented by the rightmost bit, the next highest numbered granule by the 2nd bit from the right, etc.

When a granule has been allocated to a file, its bit is set to 1 and the word containing its bit is then made negative. This is a signal to the garbage collector, a McIDAS program which runs periodically to gather up unused sectors so that they can be allocated as pages to McIDAS files. It tells the garbage collector not to return any granules to this word on its next pass. On this next pass, the garbage collector will set the word positive again so that granules can be returned to it on the following pass.

FILENAME TABLE STRUCTURE

Each of the eleven filename tables records the existence of up to 256 file extents. Each file extent has an entry that is four words (16 bytes) long:

Bytes	Contents
1-12	file name. If first character is '-' this file extent has been flagged for deletion. Only the garbage collector still knows of its existence. It records all granules occupied by the file as unavailable, then deletes the file so that its granules can be returned to free space on the following pass of the collector.
13-16	extent number and sector address of the page table for that extent. This is a full 32-bit word subdivided as follows (bits numbered from the right):

Bits	Contents
3Ø-28	extent number (Ø=extent -1, 1=extent -2, etc.)
27-24	volume number of page table
23-Ø	relative sector on volume of page table

If this word is positive, the corresponding slot in the table is full. When Ø or negative, the slot is empty and available to add a new file extent to the table. The garbage collector deletes file extents by making this word negative.

LW FILE ACCESS SUBPROGRAMS

Name	Member	Function
LWC	MRLWSUBS	create (optional-files created dynamically)
LWD	MRLWSUBS	delete
LWFILE	MRLWSUBS	check existence of extents
LWI	MRLWSUBS	read from an LW file
LWMOP	MRLWSUBS	close (optional-performed by system)
LWNAME	MRLWSUBS	check file name for validity
LWO	MRLWSUBS	write to an LW file

The LW level disk subroutines allow you to access McIDAS files as if they were large virtual arrays of 4-byte words, numbered starting from Ø. No

blocking or unblocking by the caller is required. This is performed automatically by the subprograms, with a large pool of dynamically managed buffers that is invisible to any calling routines.

It is not necessary to open or close files. It is not even necessary to create files. If you write on a nonexistent file, it is created at the conclusion of the program when you, or the system, call LWMOP. However, if you want to create a file, you may call LWC.

If you read a nonexistent or empty file, it appears to be filled with missing-data codes (all words Z8Ø8Ø8Ø8Ø).

CALLING SEQUENCES

NOTES: Arguments beginning with 'C' are CHARACTER*12. All others are integer.
All file names are 12 or fewer characters long.

LWC FUNCTION LWC(CFNAME)
input: CFNAME = filename
output: function value = Ø if file created
-1 if file already exists

LWD FUNCTION LWD(CFNAME)
input: CFNAME = filename
output: function value = Ø if file flagged for deletion
-1 if file does not exist

LWFILE FUNCTION LWFILE(CFNAME,IPAGE)
input: CFNAME = filename
IPAGE = extent (i.e., page number of page table for that extent) ranges from -1 to -8
if = Ø, check for existence of the file
output: function value = sector location of file extent's page table if positive
Ø if extent does not exist
-1 if extent or name is invalid

LWI FUNCTION LWI(CFNAME,IBEGWD,NWDS,IARR)
input: CFNAME = filename
IBEGWD = first virtual word in transfer (Ø-based)
NWDS = number of words to transfer
output: IARR = array to contain data
function value = Ø if last data transferred was from an allocated file page
1 if from a null page

LWMOP SUBROUTINE LWMOP(CFNAME)
input: CFNAME = filename

LWNAME FUNCTION LWNAME(CFNAME)
input: CFNAME = file name
output: function value = 1 if valid
Ø if valid but write protected
-1 if invalid

```
LWO     FUNCTION LWO(CFNAME,IBEGWD,NWDS,IARR)
          input:  CFNAME = filename
                  IBEGWD = first virtual word in transfer (Ø-based)
                  NWDS   = number of words to transfer
                  IARR   = array containing data to be written
          output: function value = Ø at all times.
```

WRITING WITH LOCKS

If it is possible that several programs in this or other partitions are going to write on a file concurrently (this is the normal case in a McIDAS environment), you must adhere to the following discipline or face unpredictable and unexplained loss of data integrity. A lock on the file is issued which will cause anyone else who is trying to modify the file to wait until you are done. Subroutine 'LOCK' locks a resource with an arbitrary 8-character name. For file locks the name will usually be the file name. It is important only that all programs competing for the resource agree on the name to lock. For example, all navigation files are locked under the umbrella name 'NAVILOCK'. As a result, when one navigation file is locked, they all are.

Because of the invisible page allocation table which is attached to every McIDAS file, and because of the cache buffering employed by LWI/LWO which causes actual I/O's to be done in an unpredictable order, instructions should be issued in the following order:

```
1) CALL LWMOP  ('filename')  flushes the buffer pool
2) CALL LOCK   ('filename')  locks the file name
3) LWI/LWO                   reads and writes from/to the file
4) CALL LWMOP  ('filename')  flushes the buffer pool
5) CALL UNLOCK ('filename')  release the file for the use of
                             other programs
```

Programs which only read a file need not use this procedure, but if they wish to be sure that the file is in a consistent state at the time of reading, they must follow the procedure to avoid executing while some other program is modifying the file.

GENERALIZED STRUCTURES

AREA DIRECTORY

Type: LW File
Size: 447936 words
Initialization:
Documented by: R. Dengel

WORD ALLOCATION

DATDIR contains the directory entries for areas. The directory for an area contains identification information describing important characteristics and parameters of the area data. There are 6999 global areas, numbered starting at 1. (This number may increase in the future.) Each entry is a 64-word (256-byte) integer array. The structure of a directory entry is as follows:

Words	Contents
1	area status = <Ø area does not exist Ø area exists
2	type = 4 new area structure (all bands interleaved)
3	satellite identification number (SSS)
4	nominal year and Julian day of image or sounding (YYDDD)
5	nominal time of image or sounding (HHMMSS)
6	Y-COOR - upper-left line in satellite coordinates
7	X-COOR - upper-left element in satellite coordinates
8	Z-COOR - upper-left z-coordinate. Not used for images.
9	Y-SIZE - number of lines in image
1Ø	X-SIZE - number of elements in image
11	Z-SIZE - number of bytes per data element (1 if image)
12	Y-RES - line resolution
13	X-RES - element resolution
14	Z-RES - z resolution (number of bands or channels)
15	Y-Z prefix - number of bytes, must be multiple of 4 (Ø if image)
16	project number that created the area
17	creation date (YYDDD)
18	creation time (HHMMSS)
19	filter map for soundings; a 13-bit vector. If a bit=1, there is data for that band in the file. The right-most bit is for band 1.
2Ø	image ID number - old reel number
21-24	identification (reserved); for radar 21, 22=CALL, 23=RANGE
25-32	up to 32 characters for comments
33	primary key of associated calibration codicil
34	primary key of associated navigation codicil
35	secondary key of associated navigation codicil. If primary key <Ø, use (SSSYYDDD,HHMMSS) to fetch system navigation.
36	validity code, must match line prefix
37-44	PDL. In packed-byte format if Mode AA image.
45	where band 8 came from if Mode AA-DS (step,dwell) image

Words	Contents
46	actual image start day (YYDDD)
47	actual image start time (HHMMSS)
48	actual starting scan
49	length of prefix documentation (DOC) section in bytes
5Ø	length of prefix calibration (CAL) section in bytes
51	length of prefix level (LEV) section in bytes (per band)
52	source type ('VISR','VAS ','AVHR','ERBE', etc.)
53	calibration type ('BRIT','RAW ','TEMP','RAD ', etc.)
54-64	(reserved)

AREA PROTECTION

WRITD and WRITDU (along with the area track allocation subprogram AQUIRE) implement area write/quit protection (AQP). When WRITD, WRITDU, or AQUIRE are called to change or delete an area, the area protection tables are checked to make sure that the current user (project) is authorized to make the change. Failure to pass the test results in abort 83 (UØ53). The operator (TØ) can override the check by setting UC(29) > Ø; console-started foreground tasks (chiefly tape programs) are always exempt from the check.

The area protection table lives at the back of the first (and only) extent of file DATDIR. The project number subtable begins at word 1Ø24ØØØ; it contains one word for each block of 16 areas.

Word	Contents
1Ø24ØØØ	project number assigned to block 1 (areas 1 - 15)
1Ø24ØØ1	project number assigned to block 2 (areas 16 - 31)
.	.
.	.
1Ø24ØØØ + n/16	project number assigned to area n

If the project number in the AQP table is Ø or negative, anyone is allowed to create or delete the areas. If the project number is positive, only programs running under that project number may alter or delete one of the corresponding areas.

The name/comments subtable begins at DATDIR word 1Ø25ØØØ. There are 8 words (32 bytes) in this subtable for each project number slot in the previous subtable. They contain a user name or comments referring to the ownership of the corresponding areas.

Both of the fields (project number and name/comments) are set by the system operator with the AQP command. Statistics concerning area usage are gathered by the MBAUS background program.

RELATED PROGRAMS AND SUBPROGRAMS

Name	Member	Function
AQP	MLAQP	assign project number to areas
COAREA	MRCOAREA	return area parameters from area directory
ENAREA	MRENAREA	write area parameters into area directory
MBAUS	MBAUS	list area disk space usage, background job
READD	MRREADD	read area directory
READDL	MRREADD	lock and read directory
REMARA	MRREMARA	remove area file and clear directory
WRITD	MRWRITD	write into area directory
WRITDL	MRWRITD	lock, write, and unlock directory
WRITDU	MRWRITD	write and unlock directory

CALLING SEQUENCES

All parameters are type integer.

COAREA SUBROUTINE COAREA (AREA,SS,DAY,TIME,LINE,ELEM,INCL,INCE,MEMO)
Return area parameters from area directory.

```
  input:  AREA  = area number
  output: SS    = satellite identification number
          DAY   = image day, YYDDD format
          TIME  = image time, HHMMSS format
          LINE  = satellite line coordinate of upper left-hand
                  corner
          ELEM  = satellite element coordinate of upper
                  left-hand corner
          INCL  = line resolution
          INCE  = element resolution
          MEMO  = 8-word memo array in directory
```

ENAREA SUBROUTINE ENAREA (AREA,SS,DAY,TIME,LINE,ELEM,INCL,INCE,MEMO)
Write area parameters into area directory.

```
  input:  AREA  = area number
  output: SS    = satellite identification number
          DAY   = date, YYDDD format
          TIME  = time, HHMMSS format
          LINE  = satellite line coordinate of upper left-hand
                  corner
          ELEM  = satellite element coordinate of upper
                  left-hand corner
          INCL  = line resolution
          INCE  = element resolution
          MEMO  = 8-word integer array to hold a memo on area
```

READD SUBROUTINE READD(ANUM,ENTRY)
Read area directory; should be used if you read only the directory.

```
  input:  ANUM  = area number
  output: ENTRY = 64-word integer array to contain directory
                  entry
```

READDL SUBROUTINE READDL(ANUM,ENTRY)
Lock and open directory only on the initial call. Subsequent calls to READDL read only.
input: ANUM = area number
output: ENTRY = 64-word integer array to contain directory entry

REMARA SUBROUTINE REMARA (AREA,EXIST)
Remove area file and clear directory.
input: AREA = area number
output: EXIST = Ø, if area did not exist; 1, if area existed

WRITD SUBROUTINE WRITD(ANUM,ENTRY)
Write into area directory. Does not unlock or force a write to disk. It is necessary to use WRITD when many entries must be written.
input: ANUM = area number
output: ENTRY = 64-word integer array containing directory entry

WRITDL SUBROUTINE WRITDL(ANUM,ENTRY)
Similar to WRITDU, but does a lock/read/write/unlock.
input: ANUM = area number
output: ENTRY = 64-word integer array containing directory entry

WRITDU SUBROUTINE WRITDU(ANUM,ENTRY)
Forces a write to disk by closing the directory, then unlocks the directory, freeing it for use by another program that calls READDL. If many writes are done, use WRITD until the last write, then you must use WRITDU.
input: ANUM = area number
output: ENTRY = 64-word integer array containing directory entry

DIGITAL IMAGERY STORAGE

Type: Area
Documented by: R. Dengel

INTRODUCTION

The "digital area," or "area," stores satellite imagery. The data stored in a digital area are arranged in a two-dimensional array. Values are referenced by line number (beginning with 1 at the top of the image) and element number (beginning with 1 at the left of each line). Each line begins with a prefix containing documentation information; the length and content depend upon the type of data in the area. There are 6999 global areas, numbered starting at 1 (this number may increase in the future).

The volume of data stored in an area is typically on the order of 1ØØ times as great as that in an LW file, necessitating a slightly different structure for efficient access. Like LW files, data is stored as a continuous stream of bytes. Unlike LW files, the physical blocking is done by tracks to optimize disk access.

Associated with the system of digital areas is an LW file called the "area directory" (DATDIR). The directory entry for a given area contains identification information describing important characteristics and parameters of the area data. The documentation for DATDIR begins on page 2-2.

READING AREAS

Examples:

```
1.   CALL OPNARA(AREA)    <or   IS=OPNARA(AREA) >

     CALL READA(AREA,LINE,BUFFER)
           - or -
     CALL REDARA(AREA,LINE,Ø,1ØØØØØ,'ALL ',BUFFER)

     CALL CLSARA(AREA)

2.   CALL OPNARA(AREA)    <or   IS=OPNARA(AREA) >
     CALL ARAOPT(AREA,1,'SPAC',4)
     DO 1ØØ LINE=1,1ØØ

     CALL READA(AREA,LINE,BUF2)
           - or -
     CALL REDARA(AREA,LINE,Ø,1ØØØØØ,'ALL ',BUF2)
   1ØØ     CONTINUE
     CALL CLSARA(AREA)
```

REDARA is element-based, not byte-based. When requesting one band, specify the data element offset (not byte offset) and the number of elements to read. When requesting all bands, specify 'ALL ' (don't calculate which byte because REDARA does the calculations) and the number of groups of elements to read, as follows:

```
CALL REDARA(AREA,LINE,ELEM,NELE,'ALL ',BUFFER)
     where:
     NELE=NUMBYT/NBANDS*(BYTES/ELEMENT)
     ELEM=BYTE/NBANDS*(BYTES/ELEMENT)
```

REDARA Examples:

1. Area contains 1Ø bands with 2 bytes/pixel and we want 1ØØ elements from band=5, beginning at element 2Ø.

```
ELEM=2Ø
NELE=1ØØ
IBAND=5
CALL REDARA(AREA,LINE,ELEM,NELE,IBAND,BUFFER)
```

2. Area contains 1Ø bands with 2 bytes/pixel and we want 1ØØ elements from all bands, beginning at element 2Ø.

```
ELEM=2Ø
NELE=1ØØ
IBAND=LIT('ALL ')
CALL REDARA(AREA,LINE,ELEM,NELE,IBAND,BUFFER)
```

Typical Flow for Reading an Area:

Routine	Description of Tasks
OPNARA	Fill the common block "groups" that contain the following entries: NAREA - number of areas open (maximum of 5) NGRP(5) - number of track groups reserved for this area IA(5) - area numbers open LISTS(2ØØ,5) - actual sequence of track group numbers NEL(5) - total bytes per line NOFF(5) - number of bytes in the prefix REANTS(64,5) - area directories for open areas The common is shared by all area read/write routines.
ARAOPT	Set options buffer for type of calibration, packed/cracked, etc.
REDARA	Read a given line of data from an area, calibrate it, crack it (if desired), and return the buffer. REDARA checks the prefix common for the proper area line combination. If not the same then REDARA calls REDPFX internally to load the prefix common.

WRITING TO AREAS

Example:

```
CALL MAKARA(AREA,DIR)
CALL OPNARA(AREA)     <or   IS=OPNARA(AREA) >
CALL WRTARA(AREA,LINE,BUFFER)
CALL CLSARA(AREA)
```

Typical Flow for Writing an Area:

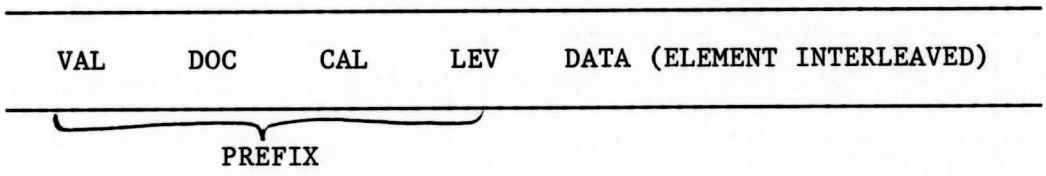
Routine	Description of Tasks
MAKARA	Calculate number of bytes/line, determine the available space on the volume needed by the area, and reserve the tracks for the area. Also writes the initial area directory entry. Space is not allocated until the area quit protection (AQP) is checked.
OPNARA	same as reading areas
WRTARA	Write data elements to disk (assumes data is in its interlaced form).

ASSUMPTIONS

- NAV and CAL are dynamically linked and loaded and are both bidirectional.
- All data in area is at LINERES and ELERES or has been brought up to that point in the area directory.
- All lines are the same length.
- All elements are the same length.
- All levels (e.g., bands in soundings) may appear in one line.

WORD ALLOCATION

All areas have the following format:



VAL → validity code. If validity code in area directory is not zero or null, then VAL exists; otherwise, it does not. When VAL exists, it is always four bytes long. VAL is primarily intended to be used by real-time ingestors and/or tape read programs.

DOC → documentation section. The contents of this section are dependent upon the SSS. DOC's length (in bytes) is in the area directory (zero indicates not present).

CAL → calibration section. The contents of CAL are dependent upon the SSS. CAL's length (in bytes) is in the area directory (zero indicates not present).

LEV → byte array that lists which levels have data in this line and the order in which the levels appear. The length of this array equals the number of levels in the area, but is always a multiple of 4. Zeros appear where there is no level. The length of LEV is also kept in the area directory.

PREFIX → VAL through LEV are collectively called the prefix. It is assumed that each section of the prefix is a multiple of four bytes. This restriction is imposed by the GET/PUT tape format.

DATA → element interleaved data in the order specified by the levels section. The total length of the DATA section is always a multiple of four bytes.

RELATED SUBPROGRAMS

Name	Member	Function
AQUARA	MRAQUARA	acquire disk tracks for an area
ARABOX	MRARABOX	get digital data for a box within an area
ARALOC	MRARALOC	find area of input frame
ARAOPT	MRARASUB	set area read/write options
ARASIZ	MRARASIZ	create file for image of 'LINSIZ' x 'ELESIZ'
CLSARA	MRARASUB	remove area from lists
GETCAL	MRGETCAL	get calibration section from a prefix
MAKARA	MRMAKARA	make initial area directory
MAKPFX	MRARASUB	make prefix entry for a line
OPNARA	MRARASUB	produce list of track groups assigned to an area
RAVLER	MARARSUB	create interlaced line of data
READA	MRREADA	read a line from 'AREA,' without the prefix
REDARA	MRARASUB	read line from an area
REDPFX	MRARASUB	read prefix from an area line
WRTARA	MRARASUB	write a line to an area

CALLING SEQUENCES

Arguments beginning with 'C' are character; all others are integer.

AQUARA SUBROUTINE AQUARA(AREA, NBYTS)
Acquire disk tracks for a new area.
input: AREA = area number
NBYTS = number of bytes required

ARABOX SUBROUTINE ARABOX(AREA,FSTLN,LSTLN,FSTEL,LSTEL,BAND,CAL,SPAC,IBUF)
Get digital data for a section within an area.
input: AREA = area number
FSTLN = first line to read
LSTLN = last line to read
FSTEL = first element to read

```
                LSTEL = last element to read
                BAND  = spectral band (Ø for noninterlaced data)
                CAL   = calibration type ('RAW','RAD','TEMP', etc.)
          output: SPAC  = precision (1, 2 or 4 byte)
                IBUF  = output array

ARALOC    SUBROUTINE ARALOC (FRAME,AREA,TIME,SS,DAY)
          Find area of input frame.
            input:  FRAME = frame number
            output: AREA  = area number from which frame was loaded
                    TIME  = time of frame image, HHMMSS
                    SS    = satellite ID number of frame image
                    DAY   = date of frame image, YYDDD

ARAOPT    SUBROUTINE ARAOPT(AREA,NOPT,COPT,VAL)
          Set area read/write options.
            input:  AREA = area number to set options
                    NOPT = number of options in options buffer
                    COPT = buffer containing names of options to set
                    VAL  = value(s) for options:
                           'SPAC' - spacing of output data (1, 2 or 4 bytes)
                           'PREC' - precision of stored data (1, 2 or 4 bytes)
                           'UNIT' - output units (e.g.,'TEMP','RAD','BRIT')
                           'SCAL' - scale factor to apply to data

ARASIZ    SUBROUTINE ARASIZ (AREA,LINSIZ,ELESIZ)
          Create a file for an image of 'LINSIZ' x 'ELESIZ'.
            input:  AREA   = area number
                    LINSIZ = number of lines requested
                    ELESIZ = number of elements requested, plus prefix

CLSARA    SUBROUTINE CLSARA(AREA)
          Remove area from lists.
            input:  IAREA = area number

GETCAL    SUBROUTINE GETCAL(AREA,LINE,IBUF)
          Fetch calibration section from prefix.
            input:  AREA = area number to read
                    LINE = line number to read
            output: IBUF = array holding the prefix

MAKPFX    SUBROUTINE MAKPFX(AREA,LINE,DOC)
          Make the prefix entry for a given line.
            input:  AREA = area number
                    LINE = line number
                    DOC  = array holding documentation

MAKARA    SUBROUTINE MAKARA(AREA,ARADIR)
          Acquire disk space for an image and make initial area directory.
            input:  AREA   = area number
                    ARADIR = new area directory entry for this area
```

OPNARA FUNCTION OPNARA(AREA)
Collect a list of track groups assigned to an area and place them in the common block.
input: AREA = area number
output: function value = pointer number into area common of area just opened

RAVLER SUBROUTINE RAVLER(AREA,LINE,SLOT,BAND,CAL,IARRAY,JARRAY)
Make an interlaced line of data.
input: AREA = area number
LINE = line number
SLOT = position of data in interlace sequence
BAND = band number to interlace
CAL = buffer holding calibration information for this band
IARRAY = array of data to ravel
output: JARRAY = raveled array

READA SUBROUTINE READA (AREA,LINOFF,BUF)
Read a line from 'AREA' without the prefix.
input: AREA = area number
LINOFF = line offset (Ø-based)
output: BUF = data returned in this array

REDARA SUBROUTINE REDARA(AREA,LINE,ELEM,NELE,BAND,IARRAY)
Read a line from an area, returning a specified subset of the line.
input: AREA = area number
LINE = line to read
ELEM = starting element offset. Ø returns first element of the data.
NELE = number of elements. Will not return any past the line end.
BAND = spectral band of desired data, or 'ALL' for all bands
output: IARRAY = array to hold returned data

REDPFX SUBROUTINE REDPFX(AREA,LINE,IBUF)
Read the prefix of an area.
input: AREA = area number to read
LINE = line number to read
output: IBUF = array holding the prefix

WRTARA SUBROUTINE WRTARA(AREA,LINE,IARRAY)
Write a line to an area.
input: AREA = area number
LINE = line number
IARRAY = source of data, length dependent on area size

BYTE PACKING/CRACKING

RELATED SUBPROGRAMS

Name	Member	Function
MOVH	SRMOVH	move halfwords
MOVPIX	MRMOVPIX	pixel sampling with offsets
MOVWRD	SRMOVWRD	move words
MPIXEL	SRMPIXEL	in-place pixel cracker/packer
MPIXTB	SRMPIXTB	in-place pixel cracker/packer with table lookup
MVASTB	SRMVASTB	in place pixel VAS cracker/packer with table lookup and YSUBZ

CALLING SEQUENCES

MPIXEL SUBROUTINE MPIXEL(NUM,SLEN,DLEN,BUF)

In-place pixel cracker/packer.

```
  input:  NUM   = number of pixels to move
          SLEN, DLEN = source and destination pixel length in
                       bytes (1,2,4)
  input/output:  BUF = array of pixels
```

MPIXTB SUBROUTINE MPIXTB(NUM,SLEN,DLEN,BUF,TABLE)

In-place pixel cracker/packer with table lookup.

```
  input:  NUM   = number of pixels to move
          SLEN, DLEN = source and destination pixel length in
                       bytes (1,2,4)
          TABLE = 4-byte table array
  input/output: BUF   = array of pixels
```

MVASTB SUBROUTINE MVASTB(NUM,SLEN,DLEN,BUF,TABLE,YSUBZ)

In-place pixel VAS cracker/packer with table lookup & YSUBZ.

```
  input:  NUM   = number of pixels to move
          SLEN  = source pixel length in bytes (1, 2, 4)
          DLEN  = destination pixel length in bytes (1, 2, 4)
          TABLE = 4-byte table array
          YSUBZ = most significant bits, not including the sign
                  bit, of the space view value
  input/output: BUF = array of pixels
```

MOVPIX SUBROUTINE MOVPIX(NUM,SOURCE,SOFF,SINC,DEST,DOFF,DINC)

Pixel sampling with offsets.

```
  input:  NUM    = number of bytes to move
          SOURCE = array of bytes
          SOFF   = start byte offset
          SINC   = source increments in bytes (positive,
                   negative or Ø)
          DINC   = source increments in bytes (positive,
                   negative or Ø)
          DOFF   = start byte offset
  output: DEST   = array of bytes
```

```
MOVWRD  SUBROUTINE MOVWRD(NUMB,ISORCE,IINC,JDEST,JINC)
        Move words.
          input: NUMB   = number of words to move
                 ISORCE = source
                 IINC, JINC = increments measured in words
                 JDEST  = destination

MOVH    SUBROUTINE MOVH(NUMB,ISORCE,IINC,JDEST,JINC)
        Move half-words.
          input:  NUMB   = number of half-words to move
                  ISORCE = source
                  IINC, JINC = increments measured in words
                  JDEST  = destination
```

DYNAMICALLY LOADED CALIBRATION MODULES (GENERIC FORM)

Example:

```
//KBXV7ØØØ JOB CLASS=B,MSGLEVEL=(Ø,Ø)
//* MLKBXVIS DAS  Ø6/Ø4/85: MEMBER UPDATED
// EXEC MCPRG,OPT=Ø
//FORT.SYSIN DD *
      FUNCTION KBXINI(CIN,COUT,IOPT)
      CHARACTER*(*) CIN,COUT
      DIMENSION IOPT(*)
      COMMON/VISRXX/JTYPE,ISOU,IDES,JOPT(2Ø)
      CALL MOVW(2Ø,IOPT,JOPT)
      JTYPE=Ø
      ISOU=IOPT(1)
      IDES=IOPT(2)
      IF(CIN.EQ.'BRIT'.AND.COUT.EQ.'TEMP') JTYPE=1
      IF(JTYPE.EQ.Ø) GO TO 9ØØ
      KBXINI=Ø
      RETURN
9ØØ   CONTINUE
      KBXINI=-1
      RETURN
      END
      FUNCTION KBXCAL(CALB,IDIR,NVAL,IBAND,IBUF)
      COMMON/VISRXX/JTYPE,ISOU,IDES,JOPT(2Ø)
      INTEGER IBUF(*),IDIR(*)
      DIMENSION ITAB(256)
      DATA LASDAY/-1/
      IDAY=IDIR(4)
      IF(IDAY.NE.LASDAY) CALL TEMP(IDAY,ITAB)
      LASDAY=IDAY
      CALL MPIXTB(NVAL,ISOU,IDES,IBUF,ITAB)
      KBXCAL=Ø
      RETURN
      END
      FUNCTION KBXOPT(CFUNC,IIN,IOUT)
      COMMON/VISRXX/JTYPE,ISOU,IDES,JOPT(2Ø)
      INTEGER IIN(*),IOUT(*)
      KBXOPT=Ø
```

```
      RETURN
      END
/*
//LKED.SYSIN DD *
 INCLUDE SYSLIB(KBHEAD,UTPACK,LWPACK,SCPACK)
 ENTRY KBHEAD
 ALIAS KB2VISR,KB3VISR
 NAME KB1VISR(R)
/*
//
```

RELATED SUBPROGRAMS

Name	Member	Function
KBPREP	MRKBPREP	set up for dynamic load of calibration
KB_CAL	MRNVPACK	calibrate a buffer of data
KB_INI	MRNVPACK	initialize calibration routines for data type
KB_OPT	MRNVPACK	special calibration options

CALLING SEQUENCES

```
KBPREP  SUBROUTINE KBPREP(SLOT,ITYPE)
        Set up for dynamic load of calibration.
          input: SLOT  = dynamic load module slot (1 or 2)
                 ITYPE = calibration type (from area directory)

KBnCAL  FUNCTION KBnCAL(PFX,IDIR,NVAL,IBAND,IBUF)
        Calibrate a buffer of data; n indicates a number 1 to 3.
          input: PFX   = line prefix
                 IDIR  = area directory
                 NVAL  = number of data values to calibrate
                 IBAND = band
                 IBUF  = I/O buffer

KBnINI  FUNCTION KBnINI(CIN,COUT,IOPT)
        Initialize the calibration routines for this data type; n
        indicates a number 1 to 3.
          input: CIN   = input units ('RAW','RAD', etc.)
                 COUT  = output units ('RAW','RAD', etc.)
                 IOPT  = options buffer from ARAOPT

KBnOPT  SUBROUTINE KBnOPT(CFUNC,INBUF,OUTBUF)
        Special calibration options; n indicates a number 1 to 3.
          input: CFUNC  = optional function
                 INBUF  = input buffer
                 OUTBUF = output buffer
```

VOLUME DIRECTORIES

(blank holds volume number Ø-6)
Type: LW file
Size: 1Ø24 Words (2Ø48 integer*2)
Initialization: None
Documented by: R. Dengel

WORD ALLOCATION

Each area volume (address space) has an associated small McIDAS file named VOLnDIR (where n holds the number of the volume to which the file applies, e.g., VOL1DIR). These files contain the assignments of the track groups (1 track group = 8 tracks) on the volume. The directory consists of a half-word (16 bits) for each track group; there are 2Ø48 track groups on a volume. The track group entry is n (>Ø) to indicate that the group is presently assigned to an area number n, or it is Ø to indicate that the group is available for assignment to an area, or it is negative to signify the end of the table for searching (thus a volume can be configured with fewer than 2Ø48 track groups).

Word	Track Group	Tracks
1	1	1-8
2	2	9-16
.	.	.
.	.	.
.	.	.
2Ø48	2Ø48	16377-16384

The number of active area volumes on the system is found at location 8 of the global user common and must be set in the WAKEUP program.

The first volume directory (VOLØDIR) has the additional function of holding volume statistics for all the active area volumes on the system. The statistics are found on the 1Øth page (word 9216) of the VOLØDIR file. These statistics include the following values:

Word	Contents
FSTARA	number of first area on this volume
LSTARA	number of last area on this volume
NBPT	number of bytes per track (default=19Ø68)
NTPG	number of tracks per track group (default=8)
NGPV	number of track groups per volume (default=2Ø48)
MXTPA	maximum number of tracks per area (default=2ØØ)

For each active area volume on the system there is a statistics group in VOLØDIR. The location of the appropriate statistics for a given volume is determined with the following calculation:

POS = (VOLUME#-1)*6 + 9216

These statistics are used by area read/write routines to determine which areas are contained on which volumes, the maximum amount of space available for each area, etc. Initialization of these constants is performed by the McIDAS foreground job AVU. The initialization of the track tables is performed by the INIDSK job.

RELATED PROGRAM AND SUBPROGRAMS

Name	Member	Function
ARASUB	MRARASUB	area read/write routines
AVU	MLAVU	area volume utility
INIDSK	MLINIDSK	set up track tables
WAKEUP	MLWAKEUP	set global user common, other system initialization

CALLING SEQUENCES

ARASUB See "Areas" for the various routines and their calling sequences.

INTERNAL SUBPROGRAMS FOR AREAS

INTRODUCTION

The following area access subroutines are used internally by the area access package. Generally, they are not used by applications programmers. Internal subprograms for area access are included here for programmers who are interested in a deeper layer of subprograms.

RELATED SUBPROGRAMS

Name	Member	Function
GETTRK	MRARASUB	give track number in volume for area relative track
GETVOL	MRARASUB	find volume number containing 'area'
HOWBIG	MRHOWBIG	return area dimensions
ISBAND	MRARASUB	check for existence of 'level'
RDTRKA	SRRDTRKA	area access method for BDAM read
VOLUME	MRVOLUME	get volume dimensions
VOLX	MRVOLX	return area volume on which 'area' resides
WRTRKA	SRRDTRKA	area access method for BDAM write

CALLING SEQUENCES

GETTRK SUBROUTINE GETTRK(AREA,TRACK)
Give the track number in the volume for the area-relative track.
input: AREA = area number
TRACK = area-relative track

GETVOL SUBROUTINE GETVOL(AREA,SLOT)
Find volume number containing specified area. Load common 'VOLCOM'.
input: AREA = area number
SLOT = 'VOLCOM' slot number

HOWBIG SUBROUTINE HOWBIG(AREA,NLIN,NELE)
Return area dimensions.
input: AREA = area number
output: NLIN = number of lines in area
NELE = number of elements in area

ISBAND FUNCTION ISBAND(AREA,LINE,BAND,OFFSET,INTVAL)
Check for the existence of a specified level.
input: AREA = area number
LINE = line number
BAND = spectral band, or 'ALL'
output: OFFSET = offset into data to desired element (byte)
INTVAL = interval between each element (bytes)
FNCVAL = Ø if band is found; <Ø if band not found

RDTRKA SUBROUTINE RDTRKA(INDEX,TRACK,ARRAY,LENGTH)
Area access method for BDAM (Basic Data Access Method) read.
input: INDEX = specify which DCB is used, range Ø to 7
TRACK = specify track number, range Ø to 16383
ARRAY = I/O buffer address
LENGTH = buffer length, in bytes

VOLUME SUBROUTINE VOLUME(AREA,NVOL,VOLNAM,FSTARA,LSTARA,NBPT,NTPG,
NGPV,MXTPA)
Get volume dimensions.
input: AREA = area number
output: NVOL = volume number containing 'AREA'
VOLNAM = volume name (DIMENSION 3)
FSTARA = first area in volume
LSTARA = last area in volume
NBPT = number of bytes per track
NTPG = number of tracks per group
NGPV = number of groups per volume
MXTPA = maximum number of tracks per area

VOLX FUNCTION VOLX(AREA,FSTARA,LSTARA,NBPT,NTPG,NGPV,MXTPA)
FUNCTION VOLX(AREA)
VOLX returns the area volume on which 'AREA' resides.
input: AREA = area number
output: FSTARA = first area in volume
LSTARA = last area in volume
NBPT = number of bytes per track
NTPG = number of tracks per group
NGPV = number of groups per volume
MXTPA = maximum number of tracks per area
FUNCTION VALUE = volume number

WRTRKA CALL WRTRKA(INDEX,TRACK,ARRAY, LENGTH)
Area access method for BDAM write.
input: INDEX = specify which DCB is used, range Ø to 7
TRACK = specify track number, range Ø to 16383
ARRAY = I/O buffer address
LENGTH = buffer length

CODICILS

INTRODUCTION

The Codicil Structure is a generalized key-based data storage and retrieval system. Originally designed to provide an efficient means of producing variable-length addenda to existing files, it may be used in any case where named data items of arbitrary size are required.

Examples include the navigation and area codicils. The former contains all the information necessary to navigate a satellite image and is attached to the area directory, thus making the area self-contained. Area codicils, also associated with area directory entries, provide audit trail information. If the data in the area have been manipulated in any way, the codicil information will reflect all modifications to the original data.

RELATED SUBPROGRAMS

Name	Member	Function
NXD	MRNXO	general codicil file deletion
NXDLOK	MRNXO	codicil file deletion with locks
NXI	MRNXI	general codicil file read
NXKEYS	MRNXKEYS	general codicil search
NXLOCK	MRNXO	general codicil file lock
NXO	MRNXO	general codicil file write
NXOLOK	MRNXO	codicil file write with locks
NXUNLK	MRNXO	general codicil file unlock

CALLING SEQUENCES

```
NXD     SUBROUTINE NXD(CSNAME,KEY1,KEY2)
          input:  CSNAME = name of indexed structure
                  KEY1   = primary key
                  KEY2   = secondary key

NXDLOK  SUBROUTINE NXDLOK(CSNAME,KEY1,KEY2)
          input:  CSNAME = name of indexed structure
                  KEY1   = primary key
                  KEY2   = secondary key

NXI   FUNCTION NXI(CSNAME,KEY1,KEY2,IARR)
          input:  CSNAME = name of indexed structure
                  KEY1   = primary key, e.g., area number
                  KEY2   = secondary key
          output: IARR   = array to contain data
                  function value =  Ø if successful
                                   -1 or -2 if not found
                                   -3 if bad primary key (KEY1)

NXLOCK  SUBROUTINE NXLOCK(CSNAME)
          input:  CSNAME = name of indexed structure
```

```
NXO      FUNCTION NXO(CSNAME,KEY1,KEY2,IARR)
           input:  CSNAME = name of indexed structure
                   KEY1   = primary key
                   KEY2   = secondary key
                   IARR   = array containing data
           output: function value =  Ø if successful
                                    -2 if file full
                                    -3 if bad primary key (KEY1)

NXOLOK   FUNCTION NXOLOK(CSNAME,KEY1,KEY2,IARR)
           input:  CSNAME = name of indexed structure
                   KEY1   = primary key
                   KEY2   = secondary key
                   IARR   = array containing data
           output: function value =  Ø if successful
                                    -2 if file full
                                    -3 if bad primary key (KEY1)

NXUNLK   SUBROUTINE NXUNLK(CSNAME)
           input:  CSNAME = name of indexed structure
```

AREA CODICILS

(blank holds codicil file number)
Type: Codicil
Size: Variable
Initialization: Standard codicil read/write routines
Documented by: R. Dengel

WORD ALLOCATION

Area codicils contain accounting information in the form of 8Ø-character (2Ø-word) entries. These entries list all operations which in any way affect the digital data or its supporting information. A total of 5 entries are contained in each codicil. Codicils are allocated as they are needed by the supporting software. The contents are listed through the LA [area#] FORM=AUDIT command.

RELATED SUBPROGRAM

Name	Member	Function
STAMP	MRSTAMP	stamp day, time, and audit information into area codicil

CALLING SEQUENCE

STAMP SUBROUTINE STAMP(ANUM)
Stamp day, time, and audit information into area codicil.
input: ANUM = area number

GRAPHICS DIRECTORIES

(blank holds graphics directory file number)
Type: Codicil file
Size: 128 words per entry
Initialization: Codicil write routines
Documented by: R. Dengel

WORD ALLOCATION

The structure of the virtual graphics and physical graphics directories are similar; the distinction is that virtual graphics exist within save files and may be regenerated via a set of specific McIDAS commands. The physical graphics directory refers to the window currently displayed on the graphics frame of a video terminal.

Words	Contents
1	graphics codicil type ('GRAF'-physical, 'GRAV'-virtual)
2	physical primary - terminal + graphics frame, TTFF
3	physical secondary - graphics number (1-based)
4	USER - logon user ID (1-4 characters)
5	PROJ - project number of user
6	creation day, YYDDD
7	creation time, HHMMSS
8	TV line containing minimum graphics pixel (TV coor)
9	TV element containing minimum graphics pixel (TV coor)
10	TV line containing maximum graphics pixel (TV coor)
11	TV element containing maximum graphics pixel (TV coor)
12	graphics type (Ø=WRRM, >Ø=area number)
13	(reserved)
14	virtual primary - virtual graphics file number
15	virtual secondary - pen motion file pointer (1-1Ø24)
16	name of program which generated graphic (first 4 characters)
17	name of program which generated graphic (second 4 characters)
18	(reserved)
19	(reserved)
2Ø	(reserved)

Reserved for graphics navigation parameters:

Words	Contents
21	projection, may be 'MERC', 'CONF' or 'SAT '
22	minimum latitude, deg * 1ØØØØ
23	maximum latitude, deg * 1ØØØØ
24	minimum longitude, deg * 1ØØØØ
25	maximum longitude, deg * 1ØØØØ
26	TV line containing 22
27	TV line containing 23
28	TV element containing 24

Words	Contents
29	TV element containing 25
3Ø	standard latitude ('CONF')
31	standard latitude (same as 3Ø for polar stereo)
32	normal longitude ('CONF')
33	scale or number of lines in frame ('CONF')
34-4Ø	reserved for navigation
41-48	memo (total of 32 characters)
49-64	(reserved)
65-128	frame directory for satellite navigation

NAVIGATION CODICILS

(blank holds navigation codicil file number)
Type: LW File (Codicil)
Size: Variable
Initialization: None
Documented by: D. Santek

WORD ALLOCATION (128 word blocks)

Words	Contents
1	navigation type (4-byte EBCDIC). Can be one of the following: 'GOES' for GOES satellites 'NOAA' for NOAA polar orbiters 'PS ' for polar stereographic projection 'LAMB' for Lambert conformal projection 'MERC' for Mercator projection 'MET ' for Meteosat ' ' (or binary Ø) if there is no navigation for this image 'RADR' for radar projection 'RECT' for rectilinear projection (pseudo-Mercator)

If the type is 'GOES', the structure of the succeeding words is:

Words	Contents
2	satellite ID, year, and Julian day, SSSYYDDD
3	nominal image start time, HHMMSS
	orbit parameters:
4	orbit type (always 1)
5	epoch date (ETIMY), YYMMDD
6	epoch time (ETIMH), HHMMSS
7	semimajor axis (SEMIMA), KM * 1ØØ
8	orbital eccentricity (ECCEN), * 1ØØØØØ
9	orbital inclination (ORBINC), deg * 1ØØØ
1Ø	mean anomaly (MEANA), deg * 1ØØØ
11	argument of perigee (PERIGEE), deg * 1ØØØ
12	right ascension of ascending node (ASNODE), deg * 1ØØØ
	attitude parameters:
13	declination of satellite axis (DECLIN), DDDMMSS
14	right ascension of satellite axis (RASCEN), DDDMMSS
15	picture center line number (PICLIN)
	spin:
16	spin period (SPINP). Either the period of a satellite on a given day, in microseconds, or the spin rate in revolutions/minute.

Words	Contents
	frame geometry:
17	total sweep angle, line direction (DEGLIN), DDDMMSS
18	number of scan lines (LINTOT), NNLLLLL. NN is number of sensors, LLLLL is number of scans; total number of actual lines is NN x LLLLL.
19	total sweep angle, element direction (DEGELE), DDDMMSS
2Ø	number of elements in a scan line (ELETOT)
	camera geometry:
21	forward-leaning (PITCH), DDDMMSS
22	sideways-leaning (YAW), DDDMMSS
23	rotation (ROLL), DDDMMSS
	other parameters:
24	(reserved)
25	east-west adjustment value (IAJUST), in visible elements (+ or -)
26	time that IAJUST computed from first valid landmark of the day (IAJTIM), HHMMSS
27	(reserved)
28	angle between the VISSR and sun sensor (ISEANG), DDDMMSS
29	(reserved for later implementation of *SKEW*. Must be Ø.)
3Ø	(reserved)
31	scan line 1
32	time of scan line 1 (beginning), HHMMSS
33	time of scan line 1 (continued), milliseconds * 1Ø
34	beta count 1
35	scan line 2
36	time of scan line 2 (beginning), HHMMSS
37	time of scan line 2 (continued), milliseconds * 1Ø
38	beta count 2
	gamma for this image:
39	gamma, element offset * 1ØØ. This is the nominal offset at time Ø of this day.
4Ø	gamma dot, element drift per hour * 1ØØ
41-12Ø	(reserved)
121-128	memo entry: Up to 32 characters of comments

If the type is 'NOAA', the following is the structure of the succeeding words:

Words	Contents
2	satellite ID, year, and Julian day, SSSYYDDD (SSS: TIROS-N=41, NOAA-6=42, NOAA-7=4Ø)
3	nominal image start time, HHMMSS

Words	Contents
	orbit parameters:
4	orbit type (always 1)
5	epoch date (ETIMY), YYMMDD
6	epoch time (ETIMH), HHMMSS
7	semimajor axis (SEMIMA), KM * 1ØØ
8	orbital eccentricity (ECCEN), * 1ØØØØØ
9	mean anomaly (MEANA), deg * 1ØØØ
10	mean anomaly (MEANA), deg * 1ØØØ
11	argument of perigee (PERIGEE), deg * 1ØØØ
12	right ascension of ascending node (ASNODE), deg * 1ØØØ
13	number of samples/line
14	angular increment between sample, deg * 1ØØØØØØ (left-to-right is positive)
15-45	(reserved)
46	direction of satellite pass (-1=descending, +1=ascending)
47	line number of earliest line to navigate (image coordinates)
48	time at start of earliest line, milliseconds from start of day
49	time interval between lines, milliseconds
50	flag for inverted image (Ø=normal, 1=flipped)
51	number of lines in area (for flipped)
52	number of elements in area (for flipped)
53-120	(reserved)
121-128	memo entry: up to 32 characters of comments

If the type is 'PS ', the following is the structure of the succeeding words:

Words	Contents
2	image line of North Pole
3	image element of North Pole
4	standard latitude, DDDMMSS
5	spacing at standard latitude, M
6	normal longitude, DDDMMSS
7	radius of planet, M
8	eccentricity of planet, * 1ØØØØØØ
9	coordinate type ≥Ø planetodetic <Ø planetocentric
1Ø	longitude convention ≥Ø west positive <Ø west negative
11-12Ø	(reserved)
121-128	memo entry: up to 32 characters of comments

If the type is 'LAMB', the following is the structure of the succeeding words:

Words	Contents
2	image line of North Pole
3	image element of North Pole
4	standard latitude #1, DDDMMSS
5	standard latitude #2, DDDMMSS

Words	Contents
6	spacing at standard latitude, M
7	normal longitude (DDDMMSS)
8	radius of planet, M
9	eccentricity of planet, * 1ØØØØØØ
1Ø	coordinate type ≥Ø planetodetic <Ø planetocentric
11	longitude convention ≥Ø west positive <Ø west negative
12-12Ø	(reserved)
121-128	memo entry: up to 32 characters of comments

If the type is 'RADR', the following is the structure of the succeeding words:

Words	Contents
2	row (image coordinates) of radar site
3	col (image coordinates) of radar site
4	latitude of radar site (DDDMMSS)
5	longitude of radar site (DDDMMSS)
6	pixel resolution (meters)
7	rotation of north from vertical (deg * 1ØØØ)
8	if present, same as 6, but only for longitude direction

If the type is 'RECT', the following is the structure of the succeeding words:

Words	Contents
2	a particular image row number
3	latitude corresponding to 2 (deg * 1ØØØØ)
4	a particular image column number
5	longitude corresponding to 4 (deg * 1ØØØØ)
6	latitude degrees/image line (deg * 1ØØØØ)
7	longitude degrees/image line (deg * 1ØØØØ)
8	radius of planet (meters)
9	eccentricity of planet (* 1ØØØØØØ)
1Ø	coordinate type ≥Ø planetodetic <Ø planetocentric
11	longitude convention ≥Ø west positive <Ø west negative

If the type is 'MERC', the following is the structure of the succeeding words:

Words	Contents
2	image line of equator
3	image element of equator
4	standard latitude
5	spacing at standard latitude, M
6	normal longitude, DDDMMSS
7	radius of planet, M

Words	Contents
8	eccentricity of planet, * 1ØØØØØ
9	coordinate type ≥Ø planetodetic <Ø planetocentric
1Ø	longitude convention ≥Ø west positive <Ø west negative
11-12Ø	(reserved)
121-128	memo entry: up to 32 characters of comments

If the type is 'MET ', the following is the structure of the succeeding words:

Words	Contents
2	satellite ID, year, and Julian day, SSSYYDDD
3	nominal image start time, HHMMSS
4	reference position of telescope (RPR)
5	line number corresponding to RPR (LRE)
6	reference radiometer position (IRP)
7-9	(reserved)
1Ø-3Ø	PDL (polynomial direct line) coefficients (real * 4)
31-51	PCL (polynomial direct element) coefficients (real * 4)
52-72	PIL (polynomial indirect line) coefficients (real * 4)
73-93	PIC (polynomial indirect element) coefficients (real * 4)

coefficient order:

AØØ	AØ1	AØ2	AØ3	AØ4	AØ5
A1Ø	A11	A12	A13	A14	
A2Ø	A21	A22	A23		
A3Ø	A31	A32			
A4Ø	A41				
A5Ø					

$I + J <= 5$

$$P = \text{summation}_{I,J = Ø} A_{IJ} * X^{I} * Y^{J}$$

RELATED PROGRAMS

A navigation load module must be prepared for each new satellite type, projection, or other navigation type. Only one module must be coded; it will serve for navigations in all three slots. (To accomplish this, alias statements are supplied to the Linkage Editor to give the NAV Load Module three different names. In this way we insure that the operating system loads a fresh copy of the same load module for each slot.)

You must code four subroutines:

NVXINI, which is invoked by NV1INI, NV2INI, or NV3INI
NVXSAE, which is invoked by NV1SAE, NV2SAE, or NV3SAE
NVXEAS, which is invoked by NV1EAS, NV2EAS, or NV3EAS
NVXOPT, which is invoked by NV1OPT, NV2OPT, or NV3OPT

NVXINI receives navigation parameters from the calling application program. It should check these for validity and move them into a common block (NVXINI also handles a request for a change of earth-based coordinates to (X,Y,Z) or back to (LAT,LON)). NVXSAE implements the transformation from full-resolution image coordinates to earth coordinates. NVXEAS implements the reverse transformation. NVXOPT handles transformations which will be defined in the future (and which may depend on the navigation type).

To generate codicil navigation use the following programs, depending on type:

Type	Name	Member
GOES	NK	MLNK
NOAA	NKTI	MLNKTI
PS,LAMB,MERC, RADR,RECT	MAKNAV	MLMAKNAV

NOTE: See DNGENNAV for more information.

RELATED SUBPROGRAM

Name	Member	Function
NVSET	MRNVSET	codicil navigation set-up

CALLING SEQUENCE

```
NVSET  FUNCTION NVSET(COPT,NUM)
         input:  COPT = option, can be 'area', 'frame', or 'graf'.
                 NUM  = area or frame number
         output: function value =  Ø if successful
                                =  -1 if no area or frame
                                =  -2 if no navigation
                                =  -3 if invalid option (COPT)
```

STOCKROOM
(SSEC site-specific generalized structure)

(blank holds file number)
Type: LW File (codicil)
Size: Variable
Initialization: None
Documented by: D. Santek

WORD ALLOCATION (128 word blocks)

Inventory and Bins

The primary key is the bin number (CLASS*1ØØØØ+ITEM) or project number (For 1ØØØ-NN projects, use (NN*1ØØØØØ)).

Primary key: BIN#
Secondary key: 'INFO' (stockroom) 'MAYN' (Maynard)

Record Format:

Words	Contents
1-2	requisition number (char)
3	invoice number
4	date, YYMMDD
5	quantity
6	unit price, cents
7-1Ø	(reserved)
11-1ØØ	repeat (1-1Ø) 9 times
1Ø1	number of entries
1Ø2	number of words/entry
1Ø3	number of entries/record
1Ø4	total quantity
1Ø5	restock point
1Ø6-112	28-character item description
113-128	64-character note

Maximum Number of Items in Class

Primary key: 1ØØØØ
Secondary key: 'DIRC' (stockroom) 'DIRM' (Maynard)

Words	Contents
1-14	maximum number of items for classes 1-14
15-99	(reserved)
1ØØ-1Ø2	special function
1Ø3-128	(reserved)

Project File Format

Primary key: Project number
Secondary key: 'INFO' (stockroom) 'MAYN' (Maynard)

Words	Contents
1	number of project entries
2	number of words/entry
3	number of entries/record
4-128	(reserved)

Project List

Primary key: 1ØØØØ
Secondary key: 'LIS1', 'LIS2'.... List of project numbers for Stockroom
Secondary key: 'MAY1', 'MAY2'.... List of project numbers for Maynard

Words	Contents
1	number of projects in block
2-128	project numbers

Billing Info

Primary key: Project number
Secondary key: Ø,1,2... (Stockroom) 5ØØ,5Ø1,5Ø2,...(Maynard)

Words	Contents
1-2	requisition number, char
3	invoice number
4	date, YYMMDD
5	quantity
6	unit price, cents
7	bin number
8	user, char

Repeat (1-8) 16 times

Voucher Subfile Format

Primary key: 2ØØØØØ
Secondary key: 'INFO' (Stockroom) 'MAYN' (Maynard)

Words	Contents
1	number of entries
2	number of words/entry
3	number of entries/record
4-128	(reserved)

Voucher Subfile

Primary key: 2ØØØØ
Secondary key: Ø,1,2... (Stockroom) 5ØØ,5Ø1,5Ø2,... (Maynard)

Words	Contents
1-2	requisition number, char
3	invoice number
4	quantity
5	unit price, cents
6	bin number

Repeat (1-6) 2Ø times

RELATED PROGRAMS

Name	Member	Function
STKP	ALSKTP	stockroom project utility
STKI	ALSKTI	stockroom inventory utility
STKC	ALSTKC	stockroom charge-out
STKB	ALSTKB	stockroom bin utility

GEOGRAPHICAL MATRIX DATA

Type: Grids
Documented by: D. Santek

INTRODUCTION

Grids are the McIDAS data structures which represent geographically distributed data of low resolution or volume (as compared to soundings or areas). They are n x m matrices, limited to a total size of n x m = 1Ø24Ø. Each number in the grid is an observation or a calculated value of a meteorological quantity, whose geographical coordinates are inferred from the matrix coordinates and information about the projection that is kept in the header associated with the grid. Grids are typically made by some process of interpolation from an MD (Meteorological Data) file containing the raw observations.

Grids are grouped into "grid files," up to 159 grids per file. There is no necessary relationship among the grids in a grid file; the files just provide a convenient way of handling large numbers of grids. McIDAS provides for up to 9999 grid files.

GRID TYPES

Currently, four types of grids are defined, based on geographical projection. The types are pseudo-Mercator (lat/lon), polar stereographic, Lambert conformal, and equidistant (grid points equally spaced in distance). The constants describing these four projections are given in words 33-4Ø of the grid header. There is also a grid type for which there is no navigation, where the grid points are equally spaced in TV space.

Lat/lon grids are two-dimensional matrices with 1-based (FORTRAN-type) subscripts. The northwest (upper left) corner corresponds to the (1,1) coordinate. Latitude decreases as the first coordinate increases; longitude decreases as the second coordinate increases. The matrix is stored internally by column.

You should follow these conventions when creating grids to ensure proper handling by the grid utilities.

WORD ALLOCATION

Grid files, like most other McIDAS files, are designed to be accessed by the LW disk I/O subroutines. That is, they are thought of as a large, continuous array of words, with the first word numbered Ø.

A grid has 2 components: a 64-word header that describes the grid projection, gridded variable, level parameter, time of the data, and so forth; and the actual grid, an n x m-word matrix in which n x m cannot exceed 1Ø24Ø.

A grid file can contain up to 159 grids, numbered 1 to 159. There is actually space for a grid "Ø", but this space is taken up by the collection of 64-word grid headers. These headers are also numbered Ø to 159 (header Ø contains an arbitrary 8-word ID for the whole file). An empty grid slot is represented by a negative value in the first word of the header, which otherwise gives the total size of the grid.

The formulas for locating grid headers and matrices in LW terms (that is, treating the file as a giant array whose first word is numbered Ø) are the following:

Header n begins at word 64 x n
Matrix n begins at word 1Ø24Ø x n

The value 1Ø24Ø was chosen as big enough for practical applications, and also as a multiple of the 1Ø24-word disk page size on IBM/McIDAS. Grid matrices can be smaller than 1Ø24Ø; the actual size is recorded in the grid header. But because the McIDAS disk I/O package allocates pages only as needed, small grids result in the consumption of only a small amount of disk space, even though 1Ø24Ø "virtual words" are reserved for each grid matrix.

The word numbers of the various components of a grid file are as follows:

file identification:

Words	Contents
Ø-7	32 characters of label information (EBCDIC)
8	project number under which created
9	file creation date, YYDDD

header information:

Words	Contents
64-127	header (64 words) for grid 1

Words	Contents
1	total size (rows x columns)
2	number of rows
3	number of columns
4	Julian date of data, YYDDD
5	time of data, HHMMSS
6	valid time for grid (if applicable)
7	name of gridded variable, MD file terms
8	scale of gridded variable, MD file terms
9	units of gridded variable, MD file terms
10	value of vertical level. Can be: 1Ø13 = 'MSL' 999 = ' ' Ø = 'TRO' 1ØØ1 = 'SFC' Otherwise, displayed as entered.

Words	Contents

Words	Contents
11	scale of vertical level
12	unit of vertical level
13	gridded variable type 1 = time difference 2 = time average 4 = level difference 8 = level average Or any sum of 1, 2, 4, and 8.
14	used if grid parameter is a vertical (level) difference or average, must be the same scale as word 1Ø
15	used if grid parameter is a time difference or time average, value should be in HHMMSS
16-32	(reserved)
33	grid origin, identifies type of program that generated the grid data
34	grid type: 1 = Pseudo-Mercator 2 = Polar Stereographic or Lambert Conformal 3 = equidistant 4 = Pseudo-Mercator (more general) 5 = no navigation

If type is 'Pseudo-Mercator', the following is the structure of the succeeding words:

35	maximum latitude of grid, deg * 1ØØØØ
36	maximum longitude of grid, deg * 1ØØØØ
37	minimum latitude of grid, deg * 1ØØØØ
38	minimum longitude of grid, deg * 1ØØØØ
If TYPE=1	
39	increment between grid points, same in x and y directions inc *10000
4Ø	(reserved)
If TYPE=4	
39	increment between grid points (latitude)
4Ø	increment between grid points (longitude)

If type is 'Polar Stereographic' or 'Lambert Conformal' the following is the structure of the succeeding words:

35	row number of North Pole, * 1ØØØØ
36	column number of North Pole, * 1ØØØØ
37	column spacing at standard latitude, meters
38	longitude parallel to columns, deg * 1ØØØØ
39-4Ø	standard latitudes, deg * 1ØØØØ. Set these two equal for Polar Stereographic.

Words	Contents
	If type is 'Equidistant' the structure of the succeeding words is:

Words	Contents
35	latitude of (1,1), deg * 10000
36	longitude of (1,1), deg * 10000
37	clockwise rotation of column 1 relative to north, deg * 10000
38	column spacing, meters
39	row spacing, meters

If type is 'No Navigation' words 35-40 are not used.

41	initials of user
42	project number under which grid created
43	arbitrary character ID supplied by program

128-191	header for grid 2
.	.
.	.
.	.
10176-10239	header for grid 159

grid information:

Words	Contents
1024Ø-2Ø479	grid 1
2Ø48Ø-3Ø759	grid 2
.	.
.	.
.	.

RELATED PROGRAMS AND SUBPROGRAMS

Name	Member	Function
GRDIMG	MLGRDIMG	convert grid to TV image
IGG	MLIGG	grid utility
IGTV	MLIGTV	generate contours and streamlines from grids
IGU	MLIGU	grid file utility
SCLG	VLSCLG	smooth grid with selected filter
GRDDEF	MRGRDDEF	set up grid transformation
IGCURF	MRIGMAKE	set current grid file number
IGGET	MRIGMAKE	read a grid file
IGMAKE	MRIGMAKE	create a grid file
IGMOP	MRIGMAKE	close all open grid files
IGNAME	MRIGMAKE	construct a grid file name
IGOPEN	MRIGMAKE	open grid file

Name	Member	Function
IGPUT	MRIGMAKE	write to a grid file
IGQUIT	MRIGMAKE	delete a grid file
IJLL	MRMAPDEF	convert grid row/column to lat/lon
LLIJ	MRMAPDEF	convert lat/lon to grid row/column

CALLING SEQUENCES

GRDDEF SUBROUTINE GRDDEF(IGHD)
Set up grid transformation.
input: IGHD = 64-word grid directory
Note: Use IJLL and LLIJ for transformations

IGCURF FUNCTION IGCURF(N)
Set current grid file number.
input: N = grid file number
output: function value = current grid file number

IGGET FUNCTION IGGET(GFNO,GNO,MAXWDS,GRID,NR,NC,TABLE)
Read a grid file.
input: GFNO = grid file number (1 to 9999)
GNO = grid number within grid file (1 to 159)
MAXWDS = maximum size of grid allowed to read
output: GRID = array to contain grid
NR = number of rows in grid
NC = number of columns in grid
TABLE = array to contain grid header
function value = Ø if successful
-1 if grid doesn't exist or is too big
-2 if no such file

IGMAKE FUNCTION IGMAKE(GFNO,IDENT)
Create a grid file.
input: GFNO = grid file number (1 to 9999)
IDENT = optional 8-words of EBCDIC text (ignored if GFNO is Ø)
output: function value = Ø if successful
1 if already exists
-1 if can't create

IGMOP SUBROUTINE IGMOP
Close all open grid files.
no arguments

IGNAME SUBROUTINE IGNAME(GFNO,FILNAM)
Construct a grid file name.
input: GFNO = grid file number (1 to 9999)
output: FILNAM = file name, in form 'GRIDNNNN'

IGOPEN FUNCTION IGOPEN(GFNO,FILNAM)
Open a grid file.
input: GFNO = grid file number (1 to 9999)
output: FILNAM = file name (8 characters)
function value = Ø if successful
-1 if can't open

IGPUT FUNCTION IGPUT(GFNO, IGNO, GRID, NR, NC, TABLE, GNO)
Write to a grid file.
input: GFNO = grid file number
IGNO = grid number. If > 0, grid is written in next empty slot after IGNO. IF < Ø, grid is written in ABS(IGNO), overwriting any grid that is there.
GRID = array containing the grid
NR = number of rows in grid. If < Ø, grid is deleted.
NC = number of columns in grid. If < Ø, grid is deleted.
TABLE = array containing grid header
output: GNO = actual grid number stored
function value = Ø if successful
-1 if no room
-2 if no such grid file

IGQUIT SUBROUTINE IGQUIT(GFNO)
Delete a grid file.
input: GFNO = grid file number (1 to 9999)

IJLL SUBROUTINE IJLL(XROW,XCOL,XLAT,XLON)
Convert grid row and column to latitude and longitude.
input: XROW, XCOL = grid row and column
output: XLAT, XLON = latitude and longitude
Note: Must be set up with a call to GRDDEF

LLIJ SUBROUTINE LLIJ(XLAT,XLON,XROW,XCOL)
Convert latitude and longitude to grid row and column.
input: XLAT, XLON = latitude and longitude
output: XROW, XCOL = grid row and column
Note: Must be set up with a call to GRDDEF

METEOROLOGICAL DATA FILES

INTRODUCTION

The Meteorological Data file system (MD) is designed to accommodate many different types of data under one generalized structure. A common set of routines can be employed to access all MD files, regardless of data types.

Conceptually, an MD file looks like a large two-dimensional matrix of data records. These records are addressed by 2-D coordinates; the upper left record is (1,1). Attached to each row is an additional record of several words known as the row header. Similarly, attached to each column is a column header. Logically speaking, the whole file thus looks like a large table with multiple items of information in each box and labels attached to each row and column. The actual useful information associated with each box consists of the information in the box plus the appropriate row and column headers, or:

RECORD = row header + column header + data portion

In matrix coordinates, the row headers are in column Ø, and the column headers in row Ø. Thus:

RECORD(M,N) = (M,Ø) + (Ø,N) + (M,N)

The record defined above is simply a group of words (32-bit entities) in the order just given. Each record can be up to 4ØØ words long.

FILE SCHEMA

The fields in a record are given names called "KEYS" which are 1 to 4 characters long. These names are recorded in a list called a file "SCHEMA". The SCHEMA name defines the data type. For example, the SCHEMA defining MD files for radiosonde observations is called "RAOB", and consists of KEYS such as "P" (pressure), "T" (temperature), "Z" (height). Once defined, a SCHEMA provides a blueprint for all MD files of that type.

A useful convention which should be followed whenever practicable is to design SCHEMAs so that all data for a particular time period are grouped in a single row, with the row header describing the period (including such KEYS as day and time). All data observed at a particular geographic location should be grouped in a single column, with the column header describing the location (e.g., station identifier, latitude, longitude, elevation). The time and place of an observation thus determine unique row and column coordinates at which to locate the data.

SCHEMAs are prepared in editor members and registered by the program 'SCHE'. This program converts the SCHEMA into an internal form in an LW file called 'SCHEMA'.

WORD ALLOCATION

MD files are implemented as LW level files. The locations of the various MD file components in the contiguous word space of the LW file follow. All word numbers are Ø-based.

Words	Contents
Ø-63	MD file header (label). Can be: Ø = SNAME SCHEMA name 1 = SVSN SCHEMA version number 2 = SDATE SCHEMA registration date 3 = NR number of rows 4 = NC number of columns 5 = NKEYS total number of KEYS in record 6 = NRKEYS number of KEYS in row header 7 = NCKEYS number of KEYS in column header 8 = NDKEYS number of KEYS in data portion 9 = CPOS 1-based position in record of column header 1Ø = DPOS 1-based position in record of data portion 11 = NREPS number of repeat groups 12 = REPSIZ size of repeat group 13 = REPPOS starting position in record of repeat group 14 = MDATA missing data code 15 = ID integer ID of file 16-23 = TEXTID text ID of file 24 = CPROJ creator's project number 25 = CDATE creation date 26 = COWNER creator's initials (logon ID) 27 = FRHOF Ø-based offset to row header 28 = FCHOF Ø-based offset to column header 29 = FDATOF Ø-based offset to data portion 3Ø = FENDOF first unused word in the file 31 = FUSROF start of user record 32 = FNAMOF start of key names 33 = FSCAOF start of scale factors 34 = FUNIOF start of units 35-6Ø = (reserved) 61-62 = file name 63 = MD file number
64-463	user record (MD coordinates Ø,Ø). This block of words is not described by any part of the SCHEMA and is available for storage of arbitrary information.
464-863	names of the file KEYS
864-1263	scale factors for the KEYS
1264-1663	units of the KEYS
1664-4Ø95	(reserved)

Beginning at word number 4Ø96:

Words	Contents
first NR*NRKEYS	row headers
next NC*NCKEYS	column headers
next NR*NC*NDKEYS	data portion
any remaining words	available for user purposes

MD FILE ACCESS SUBPROGRAMS

Name	Member	Function
MDCLOS	MRMDMAKE	close (optional - performed by the system)
MDCODE	MRMDMAKE	get missing data code
MDCRNT	MRMDMAKE	set or retrieve current MD file number
MDDEL	MRMDMAKE	delete a record
MDGET	MRMDMAKE	read a record from an MD file
MDI	MRMDI	read from an MD file
MDINFO	MRMDINFO	return MD header
MDKEYS	MRMDKEYS	return names,scales,units and locations of KEYS
MDMAKE	MRMDMAKE	create
MDNAME	MRMDMAKE	make 8-character MD file name
MDO	MRMDI	write to an MD file
MDOPEN	MRMDMAKE	open
MDPUT	MRMDMAKE	write a record to an MD file
MDQUIT	MRMDMAKE	delete an MD file
MDSCH	MRMDSIN	search MD file for general matching conditions
MDSIN	MRMDSIN	set up search conditions for MDSCH
MDSVC	MRMDSVC	get number of the current real-time data file
MDTAB	MRMDTAB	return a KEY's table of values from row or column headers

Calling Sequences

NOTE: Arguments beginning with 'C' are CHARACTER*12. All others are integer.

MDCLOS
```
SUBROUTINE MDCLOS(MDNO)
  input:   MDNO = MD file number
```

MDCODE
```
FUNCTION MDCODE(MDNO)
  input:   MDNO = MD file number
  output:  function value = missing data code
                            Ø if file is not open
```

MDCRNT
```
FUNCTION MDCRNT(MDNO)
  input:   MDNO = MD file number
  output:  function value = current MD file number
```

MDDEL
```
SUBROUTINE MDDEL(MDNO,M,N)
  input:   MDNO = MD file number
           M    = row number (=Ø for column header)
           N    = column number (=Ø for row header)
```

MDGET
```
FUNCTION MDGET(MDNO,M,N,ARRAY)
  input:   MDNO = MD file number
           M    = row number (=Ø for column header)
           N    = column number (=Ø for row header)
  output:  ARRAY= array to receive data
           function value = Ø if successful
                            -1 if unsuccessful
```

```
MDI      FUNCTION MDI(MDNO,M,N,NPTRS,PTRS,ARRAY)
           input:   MDNO  = MD file number
                    M     = row number
                    N     = column number
                    NPTRS = number of values to retrieve
                    PTRS  = array of locations (as returned by MDKEYS or
                            MDSCH)
           output:  ARRAY = array to receive data
                    function value = Ø if successful
                                    -1 if unsuccessful

MDINFO   FUNCTION MDINFO(MDNO,IHDR)
           input:   MDNO = MD file number
           output:  IHDR = 64-word array for file header
                    function value = Ø if file is open
                                    -1 if file is not open

MDKEYS   FUNCTION MDKEYS(MDNO,NLIST,LIST,SCALES, UNITS,LOCS)
           input:   MDNO  = MD file number
                    NLIST = number of KEYS in list
                          = -1 to retrieve all KEYS
                    LIST  = array of KEY names
           output:  SCALES= array of scale factors
                    UNITS = array of units
                    LOCS  = array of locations
                    function value = number of KEYS returned
                                   = -1 if unsuccessful

MDMAKE   FUNCTION MDMAKE(MDNO,PSNAME,PSVSN,PNR,PNC,PINTGR,PTEXT)
           input:   MDNO   = MD file number
                    PSNAME = SCHEMA name
                    PSVSN  = SCHEMA version number
                    PNR    = number of rows
                    PNC    = number of columns
                    PINTGR = integer ID
                    PTEXT  = 8-word array containing text id
           output:  function value = Ø if file was made
                                     1 if file already exists
                                    -1 if file can't be made

MDNAME   SUBROUTINE MDNAME(MDNO,FILNAME)
           input:   MDNO    = MD file number
           output:  FILNAME = 2-word integer array for MD file name

MDO      FUNCTION MDO(MDNO,M,N,NPTRS,PTRS,ARRAY)
           input:   MDNO  = MD file number
                    M     = row number
                    N     = column number
                    NPTRS = number of values to write
                    PTRS  = array of locations (as returned by MDKEYS or
                            MDSCH)
           output:  ARRAY = array of data to write out
                    function value = Ø always
```

```
MDOPEN  FUNCTION MDOPEN(MDNO,ACCESS)
          input:   MDNO  = MD file number
                   ACCESS= access code
                         = 1 for read only
                         = 2 for read or write
          output:  function value = Ø if file is open
                                    -1 if unable to open

MDPUT   FUNCTION MDPUT(MDNO,M,N,ARRAY)
          input:  MDNO = MD file number
                  M    = row number (=Ø for column header)
                  N    = column number (=Ø for row header)
                  ARRAY= array of data to write out
          output:  function value = Ø if successful
                                    -1 if unsuccessful

MDQUIT  SUBROUTINE MDQUIT(MDNO)
          input:   MDNO = MD file number

MDSCH   FUNCTION MDSCH(OROW,OCOL,NKO,OPTRS,OVAL)
          input:   NKO  = number of values in IOVAL
          output:  OROW = row where match occurred
                        = -1 if EOF detected
                   OCOL = column where match occurred
                        = -1 if EOF detected
                   OPTRS= array of locations
                   OVAL = array of data
                   function value = Ø always

MDSIN   FUNCTION MDSIN(MDNO,OPT,NKI,KIN,LOVAL,HIVAL,NKO,KOUT)
          input:   MDNO = MD file number
                   OPT  = no longer used
                   NKI  = number of search KEYS in KIN
                   KIN  = list of search KEYS
                   LOVAL= array of lower bounds on search conditions
                   HIVAL= array of upper bounds on search conditions
                   NKO  = number of KEYS in KOUT
                   KOUT = array of KEY names of values to be output
          output:  function value = Ø if successful
                                    -1 if file doesn't exist
                                    -2 argument error

MDSVC   FUNCTION MDSVC(CTYP,YYDDD)
          input:   CTYPE = real-time data type
                         = 'SVCA','RAOB','RSIG','SHIP' or 'FOUS'
                   YYDDD = Julian day
          output:  function value = MD file number

MDTAB   FUNCTION MDTAB(MDNO,CKEY,MAX,ITAB)
          input:   MDNO = MD file number
                   CKEY = KEY name
                   MAX  = maximum number of values to return in ITAB
          output:  ITAB = array containing table of values
                   function value = number of values returned in ITAB
                                  = -1 if unable to access MDNO
```

Searching in an MD File

It is often necessary to scan an MD file for records that meet certain conditions. The sequence of calls to set up and execute a search is as follows:

1) CALL MDOPEN — open the file
2) CALL MDSIN — set up search conditions
3) CALL MDSCH — find record meeting search conditions
4) MDI/MDO — perform some function upon the record
5) To find the next record meeting the search conditions, go back to number 3.

BLUEPRINTS OF ALL REGISTERED SCHEMAS

Type: LW file
Size: 2Ø32 pages
Initialization: Created and modified by the command 'SCHE'
Documented by: G. Dengel

WORD ALLOCATION

Words	Contents
Ø-31999	up to 5ØØ skeleton MD file headers. Each is 64 words long and is laid out like an MD file header.

Words	Contents
1	SNAME = SCHEMA name
2	SVSN = SCHEMA version number
3	SDATE = registration date
4	NR = number of rows
5	NC = number of columns
6	NKEYS = total number of KEYS
7	NRKEYS = number of KEYS in row header
8	NCKEYS = number of KEYS in column header
9	NDKEYS = number of KEYS in data record
1Ø	CPOS = 1-based position in record of column header
11	DPOS = 1-based position in record of data
12	NREPS = number of repetitions of repeat group
13	REPSIZ = size of repeat group
14	REPPOS = starting position of repeat group
15	MDATA = missing data code
16	ID = integer ID of file
17-24	TEXTID = text ID of file
25	CPROJ = creator's project number
26	CDATE = creation date
27	COWNER = creator's initials (logon ID)
28	FRHOF = Ø-based offset to row header
29	FCHOF = Ø-based offset to column header
3Ø	FDATOF = Ø-based offset to data portion
31	FENDOF = position of first unused word in file
32	FUSROF = start of user record
33	FNAMOF = start of KEY names
34	FSCAOF = start of scale factors
35	FUNIOF = start of units
35-6Ø	(reserved)
61-62	file name
64	MD file number

32ØØØ-32767	(reserved)
32768-...	4Ø96 word per SCHEMA.

Words	Contents
1-64	repeat of the file header
65-464	(reserved)
465-864	KEY names (1 per word)
865-1264	scale factors (1 per word)
1265-1664	units (1 per word)
1665-4Ø96	(reserved)

RELATED PROGRAMS

Name	Member	Function
SCHE	MLSCHE	register new SCHEMAS
LSCHE	MLLSCHE	list SCHEMAS

INDIVIDUAL FILE STRUCTURES

INDIVIDUAL FILE STRUCTURES

INTERFACE BETWEEN RD6Ø4 AND SVCT

Type: LW file
Size: 262144 words (256 1Ø24-word pages)
Initialization: FAA6Ø4 ingestor
Documented by: G. Dengel

WORD ALLOCATION

Page	Words	Contents
Ø	1	pointer to last page written by 6Ø4 ingestor
	2-1Ø24	(reserved)
1	1	pointer to last page processed by traffic handler
	2	pointer to last record processed by traffic handler
	3-1Ø24	(reserved)
2-255	1-1ØØØ	8Ø-character lines of text
	1ØØ1-1Ø24	(reserved)

RELATED PROGRAM

Name	Member	Function
SVCT	SLSVCT	FAA6Ø4 traffic handler

AREA STATISTICS RESULTS FILE

Type: MD file
Size: 5ØØ rows by 3Ø columns
Initialization: AX (area statistics option selector)
Documented by: R. Dengel

WORD ALLOCATION

1. Area statistics option list (stored in record (0,0) of each DAASTA file)

Words	Unit	Contents
1	CHAR	outline type
2	CHAR	color of outline (for "GO" command)
3	CHAR	color to fill interior of outline
4	CHAR	outline point sampling mode
5	PIXL	outline sampling distance
6	MSEC	outline sampling time interval (for "GO" command)
7	CHAR	measurement brightness units
8		tail percentage
9		function used for weighted sum
10	CHAR	outline output coordinates
11	CHAR	histogram output device
12		number of threshold/limit pairs
13-44		alternating threshold/limit values for 16 measurements
45		low measurement cut-off value
46		high measurement cut-off value
47		total number of outlines in file
48		pointer to last outline measured
49		number of vertices in last outline
50		maximum outline line (image coord*2)
51		minimum outline line (image coord*2)
52		maximum outline element (image coord*2)
53		minimum outline element (image coord*2)
54-57		(reserved)
58		current frame pointer
59		current area pointer
6Ø-4ØØ		(reserved)

2. Outline Point Storage

Area statistics outlines are stored in the MD file region unused by the row+column+data section. A word in the MD file directory (FENDOF) defines this word address. Outlines are stored in 300 word blocks. A total of 15Ø separate vertices (2 words per point) are possible per outline. The first 15Ø words hold line coordinates and the second 150 words hold element coordinates. In order to access this region, the area statistics package uses the basic LW READ/WRITE routines.

WINDS ALOFT FORECAST (FD) STATION LIST

Type: Editor member
Size: Variable
Initialization: Entered through the editor
Documented by: G. Dengel

WORD ALLOCATION

Each station occupies one 80-character line in the following format:

Characters	Contents
1-4	station call letters
5	blank
6-12	latitude, deg * 1ØØØØ, north>Ø
13	blank
14-21	longitude, deg * 1ØØØØ, west>Ø
22	blank
23-26	elevation, ft
27-28	numeric state or regional code
29-8Ø	blank

RELATED PROGRAM

Name	Member	Function
*	MBMKFD	initialize FEDEXFD file

* background job

TERMINAL FORECASTS (FT) STATION LIST

Type: Editor member
Size: Variable
Initialization: Entered through the editor
Documented by: G. Dengel

WORD ALLOCATION

Each station occupies one 80-character line in the following format:

Characters	Contents
1-4	station call letters
5	blank
6-12	latitude, deg * 1ØØØØ, north>Ø
13	blank
14-21	longitude, deg * 1ØØØØ, west>Ø
22	blank
23-26	elevation, ft
27-28	numeric state or regional code
29-8Ø	blank

RELATED PROGRAM

Name	Member	Function
MKFT	MLMKFT	initialize FEDEXFT file

TAF CODE TERMINAL FORECASTS STATION LIST

Type: Editor member
Size: Variable
Initialization: Entered through the editor
Documented by: G. Dengel

WORD ALLOCATION

Each station occupies one 80-character line in the following format:

Characters	Contents
1-4	station call letters
5	blank
6-7	state or regional code
8	blank
9-1Ø	country code
11	blank
12-15	latitude, DDMM
16	N or S
17	blank
18-22	longitude, DDMM
23	E or W
24	blank
25-28	elevation, m

RELATED PROGRAMS

Name	Member	Function
MKFT	MLMKFT	initialize FEDEXTAF file
*	MBMAKTAF	build DAIDTAF from master station list

* Background job

TABLE OF CONTENTS FOR MCIDAS.LINKLIB

Type: Editor file
Size: Variable
Initialization: Rewritten daily by running 'SULMAP'
Documented by: G. Dengel

WORD ALLOCATION

The first two lines are comments. Beginning with line 3, each line contains up to six names of library subprograms beginning in columns 5, 17, 29, 41, 53, and 65. Names are up to 6-characters (EBCDIC left-adjusted) in length.

RELATED PROGRAM

Name	Member	Function
*	SULMAP	write file 'DALINK'

* background job

TABLE OF CONTENTS FOR MCIDAS.LOADLIB

Type: Editor file
Size: Variable
Initialization: Rewritten daily by running 'SUXMAP'
Documented by: G. Dengel

WORD ALLOCATION

The first two lines are comments. Beginning with line 3, each line contains up to six names of McIDAS load modules beginning in columns 5, 17, 29, 41, 53, and 65. Names are up to 6-characters (EBCDIC left-adjusted) in length.

RELATED PROGRAM

Name	Member	Function
*	SUXMAP	write file 'DALOAD'

* background job

LIST OF NON-COMMAND LOADLIB MEMBERS THAT SHOULD NOT BE "ORPHANED"

Type: Editor file
Size: Variable
Initialization: Entered and modified through the editor
Documented by: G. Dengel

WORD ALLOCATION

The first five lines are comments and headings. Beginning with line six, each line contains an entry for one LOADLIB module. Entries consist of the module name, the date the entry was made, the programmer's initials, and a brief description of the module's function.

RELATED PROGRAM

Name	Member	Function
ORPHAN	SUORPHAN	set up program to delete all load modules not listed in 'DNKEYINS,' 'DAMISC,' or 'DNTOVS'

RAOB STATION LIST

Type: Editor file
Size: 262 lines
Initialization: Entered through the editor
Documented by: G. Dengel

WORD ALLOCATION

Each station occupies one 8Ø-character line in the format:

Characters	Contents
1-5	5-digit station identifier
6	blank
7-11	latitude, DDMM, north>Ø
12	blank
13-17	longitude, DDMM, west>Ø
18	blank
19-22	elevation, M
23	blank
24-25	numeric state or regional code
26-8Ø	blank

RELATED PROGRAM AND SUBPROGRAM

Name	Member	Function
RAB	MLRAB	RAOB decoder
IDRAOB	MRIDRAOB	initialize 'IDRAOB' file

CALLING SEQUENCE

```
IDRAOB  FUNCTION IDRAOB(STAT)
          output:  STAT =  Ø if file was made or already exists
                           -1 if unable to access 'DARAOBID'
                   function value =  Ø if successful
                                    -1 if unsuccessful
```

TABLE OF CONTENTS FOR MCIDAS.SOURCE

Type: Editor file
Size: Variable
Initialization: Rewritten daily by 'SUEMAP'
Documented by: G. Dengel

WORD ALLOCATION

The first two lines are comments. Beginning with line 3, each line contains up to six editor file names beginning in columns 5, 17, 29, 41, 53, and 65. Names are up to six characters (EBCDIC left-adjusted) in length.

RELATED PROGRAM

Name	Member	Function
*	SUEMAP	write file 'DATABLE'

* background job

SURFACE HOURLY DATA (SAO) STATION LIST

Type: Editor member
Size: Variable
Initialization: Entered through the editor
Documented by: G. Dengel

WORD ALLOCATION

Each station occupies one 8Ø-character line in the following format:

Characters	Contents
1-4	station call letters
5	blank
6-12	latitudes, deg * 1ØØØØ, north>Ø
13	blank
14-21	longitudes, deg * 1ØØØØ, west>Ø
22	blank
23-26	elevations, ft
27-28	numeric state or regional code
29-8Ø	blank

RELATED PROGRAM AND SUBPROGRAM

Name	Member	Function
TRAFARZ	MLARZ	surface hourly data decoder
IDSVCA	MRIDSVCA	initialize 'IDSVCA' file

CALLING SEQUENCE

IDSVCA FUNCTION IDSVCA(STAT)
output: STAT = Ø if file was made or already exists
-1 if unable to access 'DCIDFILE'
function value = Ø if successful
-1 if unsuccessful

FOUS STATION LIST

Type: Editor member
Size: Variable
Initialization: Entered through the editor
Documented by: G. Dengel

WORD ALLOCATION

Each station occupies one 8Ø-character line in the following format:

Characters	Contents
1-3	station call letters
4	blank
5-11	latitudes, deg * 1ØØØØ, north>Ø
12	blank
13-2Ø	longitude, deg * 1ØØØØ, west>Ø
21	blank
22-26	elevations, ft
27	blank
28-29	state or regional codes
3Ø-8Ø	blank

RELATED PROGRAM AND SUBPROGRAM

Name	Member	Function
TRAFFOUS	MLRFOUS	FOUS decoder
IDFOUS	MRIDFOUS	initialize 'IDFOUS' file

CALLING SEQUENCE

IDFOUS FUNCTION IDFOUS(NSTA)
output: NSTA = number of stations in the file
function value = Ø if successful
-1 if unsuccessful

RAOB STATION LIST

Type: Editor file
Size: 262 lines
Initialization: Entered through the editor
Documented by: G. Dengel

WORD ALLOCATION

Each station occupies one 8Ø-character line in the format:

Characters	Contents
1-5	5-digit station identifier
6	blank
7-11	latitude, DDMM, north>Ø
12	blank
13-17	longitude, DDMM, west>Ø
18	blank
19-22	elevation, M
23	blank
24-25	numeric state or regional code
26-8Ø	blank

RELATED PROGRAM AND SUBPROGRAM

Name	Member	Function
RAB	MLRAB	RAOB decoder
IDRAOB	MRIDRAOB	initialize 'IDRAOB' file

CALLING SEQUENCE

```
IDRAOB  FUNCTION IDRAOB(STAT)
          output:  STAT =  Ø if file was made or already exists
                           -1 if unable to access 'DCRAOBID'
                   function value =  Ø if successful
                                     -1 if unsuccessful
```

Type: Editor file
Size: Variable
Initialization: Entered and modified through the editor
Documented by: G. Dengel

WORD ALLOCATION

The first three lines are comments and headings. Beginning with line four, each line contains an entry for one command. Entries consist of the command name, the date the entry was made, the editor file containing the source code, and a brief description of the function of the command.

RELATED PROGRAM

Name	Member	Function
ORPHAN	SUORPHAN	set up program to delete all load modules not listed in 'DNKEYINS', 'DAMISC', or 'DNTOVS'

LIST OF ALL KNOWN MD FILE KEYS, UNITS, AND MAXIMUM FIELD WIDTHS

Type: Editor file
Size: Variable
Initialization: Entered and modified through the editor
Documented by: G. Dengel

WORD ALLOCATION

Lines beginning with ' * ' are comments. The first 14 lines are comments followed by two lines of heading information. KEYS begin at line 2Ø, one line per KEY (possibly followed by comment lines) in the format:

Characters	Contents
1-4	KEY name (EBCDIC left-adjusted)
5-9	blank
1Ø-11	maximum field width
12	blank
13-16	units (EBCDIC left-adjusted)
17-23	blank
24-72	description
73-8Ø	blank

RELATED PROGRAMS AND SUBPROGRAM

Name	Member	Function
MDO	MLMDO	list data from MD files
MKFWDT	MLMKFWDT	create 'FLDWDTH' file from list in DNKEYS
PMSIZE	MRPMSIZE	get maximum field widths of MD file KEYS

CALLING SEQUENCE

NOTE: All arguments are integer.

PMSIZE
```
SUBROUTINE PMSIZE(NUM,NKEYS,UNITS,SCALES,NUNITS,KEYS,SIZE,
NSCALE)
  input:  NUM    = number of KEYS input to MDO
          NKEY   = total number of KEYS
          UNITS  = array of units for KEYS
          SCALES = array of scales for KEYS
          NUNITS = array of modified units
          KEYS   = array of KEYS
  output: SIZE   = array of field widths
          NSCALE = array of modified scales
```

Type: Editor file
Size: 9Ø lines
Initialization: Entered and modified through the editor
Documented by: H. Woolf

WORD ALLOCATION

The first six lines are comments. Beginning with line seven each line contains an entry for one command. Entries consist of the command name, three periods, and a brief description of the command.

RELATED PROGRAM

Name	Member	Function
ORPHAN	SUORPHAN	set up program to delete all load modules not listed in 'DNKEYINS', 'DAMISC', or 'DNTOVS'

EVENT FLAGS

Type: LW file
Size: 1 page
Initialization: Standard LW file routines
Documented by: J. Ide

WORD ALLOCATION

There are integer and Boolean event flags, partitioned as follows:

Flags	Contents
1-799	integer flags, 4-byte integers
8ØØ-6999	Boolean flags, packed 1 bit/flag, least to most significant within each word

See DNEFLAGS and the McIDAS Reference Manual for information about individual flags.

RELATED PROGRAMS AND SUBPROGRAMS

Name	Member	Function
ESKE	MLESKC	enter request into event scheduler
GETF	MLGETF	display value of event flag
SETF	MLSETF	set value of event flag
DOFLGS	MRDOFLGS	set event flags for message received in image spool
GETFLG	MRGETFLG	read value of event flag
SETFLG	MRSETFLG	write value of event flag

CALLING SEQUENCES

DOFLGS
```
SUBROUTINE DOFLGS
     no arguments
```

GETFLG
```
FUNCTION GETFLG(FLGNUM)
     input:  FLGNUM = event flag number
     output: function value = current value of event flag
```

SETFLG
```
SUBROUTINE SETFLG(FLGNUM,NEWVAL)
     input:  FLGNUM = event flag number
             NEWVAL = value to which flag is set
```

COLOR ENHANCEMENT STORAGE FILE

Type: LW file
Size: Variable, maximum 3,4Ø6,624 words
Initialization: Function of program
Documented by: A. Weickmann

WORD ALLOCATION

Words	Contents
Ø	word location of next available data block
1-2ØØ	starting word locations of the data for each of the 2ØØ possible terminals
1Ø24-n	enhancement tables

Each terminal has its own block allocated upon initial usage of the enhancement feature, thereby conserving space.

The starting word of the next available data block is computed as:

Old Starting Word + (256*66 + 132)

where there are 256 tables per terminal, 66 words per table plus two 66-word tables for the current enhancement data. Each table consists of 64 words of enhancement data and two words for a name.

RELATED PROGRAMS AND SUBPROGRAM

Name	Member	Function
ENHIO	MRENHIO	I/O enhancement tables to and from ENHANCES
EB	MLEB	black and white contrast stretching
EC	MLEC	clear enhancement table
EF	MLEF	plot current enhancement table
EL	MLEL	create an enhancement table
ER	MLER	restore an enhancement table
ES	MLES	save an enhancement table
ET	MLET	pseudo coloring for frame overlays

CALLING SEQUENCE

```
ENHIO  SUBROUTINE ENHIO(CKEY,IFRM,TABLE,ITERM)
         input:  CKEY = I/O request.  Can be 'READ', 'WRIT', 'REFR', or
                        'WRFR'.
                 IFRM = terminal frame number
       in/output:  TABLE = enhancement table array
                   ITERM = terminal number if CKEY is 'REFR'; output
                           flag for read errors.
```

EVENT-DRIVEN SCHEDULED COMMANDS

Type: LW file
Size: 1Ø23 pages (first extent)
Initialization: MUESINIT
Documented by: J. Ide

WORD ALLOCATION

Single entry (4ØØ possible)

Words	Contents	Description
1	SKSDAY	day that scheduling request becomes active
2	SKSTIM	time that scheduling request becomes active
3	SKEDAY	day that request is deleted
4	SKETIM	time that request is deleted
5	SKTERM	terminal at which command executes
6	SKID	ID number or name assigned when command was entered into schedule
7	SKNAME	initials of person who entered command into schedule
8	SKPROJ	project number under which command executes
9-32	SKETXT	text of expression to be evaluated
33-36	SKDUMY	not used
37-96	SKCTXT	text of valid McIDAS command

RELATED PROGRAMS AND SUBPROGRAMS

Name	Member	Function
ESKE	MLESKE	enter event scheduler requests
ESKL	MLESKL	list entries in event schedule
ESKU	MLESKU	event scheduler utility
ESDEL	MRESKIO	delete schedule file entry
ESKED	MRESKED	makes pass through scheduled command
ESKIO	MRESKIO	file I/O for event scheduler
ESLIS1	MRESLIS1	list one schedule file entry
ESLOCK	MRESKIO	lock schedule file entry
ESUNLK	MRESKIO	unlock schedule file entry
EXPR	MREXPR	evaluate expression field

CALLING SEQUENCES

ESDEL SUBROUTINE ESDEL(KENT)
[illegible] KENT = entry ID number

ESKED SUBROUTINE ESKED
no arguments

```
ESKIO     SUBROUTINE ESKIO(FUNC,KENT)
               input:  FUNC = I/O function.  Can be:
                              1 = read first part of directory
                              2 = read last part of directory and
                                  command text
                              4 = write first part
                              5 = write last part
                       KENT = entry ID number

ESLIS1    SUBROUTINE ESLIS1
               no arguments

ESLOCK    SUBROUTINE ESLOCK
               no arguments

ESUNLK    SUBROUTINE ESUNK
               no arguments

EXPR      FUNCTION(CSTR,VALUE)
               input:  CSTR = string containing expression field of ESKE
                              keyin
               output: VALUE= value to which EXPR evaluates
                       function value = -1 if syntax error
```

FORECAST FILE FOR WINDS AND TEMPERATURES ALOFT
(written exclusively for Federal Express)

Type: LW file
Size: 2ØØ1 pages
Initialization: by background job in 'MBMKFD'
Documented by: G. Dengel

WORD ALLOCATION

Pages	Words	Contents
Ø	1	number of stations
	2-1Ø24	station IDs
1	1-7	catalog 261 header save area

Word	Contents
1	base day (YYDDD)
2	base time (HHMMSS)
3	valid day (YYDDD)
4	valid time (HHMMSS)
5	start of forecast period (HH)
6	end of forecast period (HH)
7	level: 1 = 3ØØØ - 39,ØØØ 2 = 3ØØØ - 18,ØØØ 3 = 24ØØØ - 53,ØØØ

	8-14	catalog 262
	15-21	catalog 263
	22-28	catalog 264
	29-35	catalog 265
	36-42	catalog 266
	43-49	catalog 267
	5Ø-56	catalog 268
	57-63	catalog 269
	64-7Ø	catalog 27Ø
	71-77	catalog 271
	78-84	catalog 272
	85-91	catalog 295
	92-98	catalog 299
	99-1Ø24	reserved
2	1	number of regions
	2-342	regional identifiers
	343-683	number of stations in each region
	684-1Ø24	pointer to ID list for each region
3-11	1-1Ø24	ID lists
12-523	1-1Ø24	station pointer block (512 words per station)

Words	Contents
1-21	forecasts valid for ØØZ

Words	Contents
1-7	3ØØØ to 39,ØØØ ft

Word	Contents
1	pointer to text
2	base day (YYDDD)
3	base time (HHMMSS)
4	valid day (YYDDD)
5	valid time (HHMMSS)
6	start of forecast period (HH)
7	end of forecast period (HH)

Words	Contents
8-14	3ØØØ to 18,ØØØ ft
15-21	24,ØØØ to 53,ØØØ ft

Words	Contents
22-42	forecasts valid for Ø1Z
43-63	forecasts valid for Ø2Z
64-84	forecasts valid for Ø3Z
85-1Ø5	forecasts valid for Ø4Z
1Ø6-126	forecasts valid for Ø5Z
127-147	forecasts valid for Ø6Z
148-168	forecasts valid for Ø7Z
169-189	forecasts valid for Ø8Z
19Ø-21Ø	forecasts valid for Ø9Z
211-231	forecasts valid for 1ØZ
232-252	forecasts valid for 11Z
253-273	forecasts valid for 12Z
274-294	forecasts valid for 13Z
295-315	forecasts valid for 14Z
316-336	forecasts valid for 15Z
337-357	forecasts valid for 16Z
358-378	forecasts valid for 17Z
379-399	forecasts valid for 18Z
4ØØ-42Ø	forecasts valid for 19Z
421-441	forecasts valid for 2ØZ
442-462	forecasts valid for 21Z
463-483	forecasts valid for 22Z
484-5Ø4	forecasts valid for 23Z
5Ø5-512	reserved

Pages	Words	Contents
524-2ØØØ	1-1Ø24	text, 8Ø characters per line, 51 lines per page

RELATED PROGRAMS

Name	Member	Function
FD	MLFD	list FD forecasts
TRAFFXFD	MLFXFD	decode and file FD forecasts
*	MBMKFD	initialize file

* background job

FORECAST FILE
(written exclusively for Federal Express)

Type: LW file
Size: 8184 pages
Initialization: By background job in 'MBMKFECF'
Documented by: G. Dengel

WORD ALLOCATION

Pages	Words	Contents
Ø	1	number of stations
	2-1Ø24	station IDs
1	1-1Ø24	station IDs
2	1	number of regions
	2-342	regional identifiers
	343-683	number of stations in each region
	684-1Ø24	ID lists
3-11	1-1Ø24	ID lists
12-2Ø59	1-1Ø24	station pointer blocks (128 words per station) as follows:

Word	Contents
1	pointer to most recent block
2	deleted flag
3-17	three blocks of pointers (5 words each), each block as follows:

Word	Contents
1	pointer to forecast text
2	day
3	time
4	reserved
5	number of lines of text

	18-128	reserved
2Ø6Ø-8183	1-1Ø24	text (8Ø characters per line, 51 lines per page)

RELATED PROGRAMS

Name	Member	Function
FOR	MLFOR	list forecasts from FOR file
FFOR	MLFFOR	list forecasts from FOR, FT, and TAF files
FILFOR	MLFILFOR	file forecasts
*	MBMKFECF	initialize file

* background job

TERMINAL FORECASTS FILE
(written exclusively for Federal Express)

Type: LW file
Size: 8184 pages
Initialization: By command MKFT
Documented by: G. Dengel

WORD ALLOCATION

Pages	Words	Contents
Ø	1	number of stations
	2-1Ø24	station IDs
1	1-1Ø24	station IDs
2	1	number of regions
	2-342	regional identifiers
	343-683	number of stations in each region
	684-1Ø24	pointer to ID list for each region
3-11	1-1Ø24	ID lists
12-139	1-1Ø24	station pointer blocks (128 words per station) in the following format:

Word	Contents
1	pointer to most recent block
2	deleted flag
3-17	three blocks of pointers (5 words each), in the following format:

Word	Contents
1	pointer to forecast text
2	day
3	time
4	flag: 1=ceiling ≧2ØØØ ft and/or visibility ≧3Ø mi 2=ceiling <2ØØØ ft and/or visibility <3Ø mi 3=ceiling <8ØØ ft and/or visibility <2Ø mi 4=ceiling <2ØØ ft and/or visibility <5 mi
5	number of lines of text

18	pointer to most recent decoded forecast
19	day
2Ø	time
21-128	36-hour blocks (three words each) in the following format:

Word	Contents
1	pointer to decoded forecast text
2	flag (same as above)
3	number of lines of text

14Ø-8183	1-1Ø24	text, 8Ø characters per line, 51 lines per page

RELATED PROGRAMS

Name	Member	Function
FTL	MLFTL	list terminal forecasts
TRAFFXFT	MLFXFT	decode and file terminal forecasts
MKFT	MLMKFT	initialize file

SURFACE HOURLY DATA (SAO) TEXT FILE
(written exclusively for Federal Express)

Type: LW file
Size: 8184 pages
Initialization: MBMAKFWX
Documented by: G. Dengel

WORD ALLOCATION

Pages	Words	Contents
Ø	1	number of surface hourly data stations
	2-1Ø24	station IDs
1	1-1Ø24	station IDs
2	1	number of regions
	2-1Ø24	regional identifiers
3	1	number of regions
	2-1Ø24	number of stations in each region
4	1	number of regions
	2-1Ø24	pointer to ID list for each region
5-5Ø	1-1Ø24	ID lists
51-2Ø99	1-1Ø24	station pointer blocks (1Ø24 words per station)

Words	Contents
1	pointer to most recent observation (hourly or special)
2	pointer to most recent hourly block (see below)
3	day of most recent observation
4	time of most recent observation
5	pointer to text of most recent hourly
6	day of most recent hourly
7	time of most recent hourly
8	pointer to most recent decoded line
9	day of most recent decoded line
10	time of most recent decoded line
11-16	reserved
17-52	pointer to the text for the last 36 hours
53-1Ø24	hourly blocks (27 words per hour)

Hourly blocks (27 words per hour):

Words	Contents
1	number of observations (maximum of 8)
2-17	pointer/length pairs, pointer to text, number of lines
18	number of lines of decoded observations
19-27	pointers to decoded observations

Page		
21ØØ-8183	1-1ØØØ	text (8Ø characters per line, 51 lines per page)
	1ØØ1-1Ø24	unused

RELATED PROGRAM AND SUBPROGRAM

Name	Member	Function
SA	MLSA	list surface hourly data text
GETIDS	MRGETIDS	return all IDs for a given region

CALLING SEQUENCE

```
GETIDS  INTEGER FUNCTION GETIDS(CTYPE,REG,MAXIDS,IDS)
          input:  CTYPE  = 'SA' or 'FT'
                  REG    = regional ID
                  MAXIDS = size of output array IDs
          output: IDS    = array of station IDs
                  function value = number of stations IDs returned in
                                   'IDS'
```

TERMINAL FORECASTS IN TAF FORMAT FILE

Type: LW file
Size: 8184 pages
Initialization: By command MKFT
Documented by: G. Dengel

WORD ALLOCATION

Pages	Words	Contents
Ø	1	number of stations
	2-1Ø24	station IDs
1	1-1Ø24	station IDs
2	1	number of regions
	2-1Ø24	regional identifiers
3	1	number of regions
	2-1Ø24	number of stations in each region
4	1	number of regions
	2-1Ø24	pointer to ID list for each region
12-139	1-1Ø24	station pointer blocks (128 words per station) in the following format:

Word	Contents
1	pointer to most recent block
2	reserved
3-17	three blocks of pointers (5 words each), as follows:

Word	Contents
1	pointer to forecast text
2	day
3	time
4	reserved
5	number of lines of text

18-128	reserved

14Ø-8183	1-1Ø24	text, 8Ø characters per line, 51 lines per page

RELATED PROGRAMS

Name	Member	Function
TAFL	MLTAFL	list TAF forecasts
TRAFTAF	MLTAF	decode and file TAF forecasts
MKFT	MLMKFT	initialize file

MAXIMUM FIELD WIDTHS OF ALL KNOWN KEYS IN MD FILES

Type: LW file
Size: Variable
Initialization: 'MKFWDT', using the list of KEY names in 'DNKEYS'
Documented by: G. Dengel

FLDWDTH is used for formatting by Command MDO.

WORD ALLOCATION

Two words are allocated for each KEY. The first contains the KEY name (4-character EBCDIC, left-adjusted), and the second contains the maximum field width.

The last word in the file contains a stop code of 999999.

RELATED PROGRAMS AND SUBPROGRAM

Name	Member	Function
MDO	MLMDO	list data from MD files
MKFWDT	MLMKFWDT	create 'FLDWDTH' file
PMSIZE	MRPMSIZE	get maximum field widths of MD file KEYS

CALLING SEQUENCE

NOTE: All arguments are integer.

```
PMSIZE  SUBROUTINE PMSIZE(NUM,NKEYS,UNITS,SCALES,NUNITS,KEYS,SIZE,
                          NSCALE)
          input:   NUM    = number of KEYS input to MDO
                   NKEYS  = total number of KEYS
                   UNITS  = array of units for KEYS
                   SCALES = array of scales for KEYS
                   NUNITS = array of modified units
                   KEYS   = array of KEYS
          output:  SIZE   = array of field widths
                   NSCALE = array of modified scales
```

VIDEO FRAME DIRECTORY

Type: LW file
Size: First extent
Initialization: SUFRAMED
Documented by: R. Dengel

WORD ALLOCATION

The file containing the directory entries for frames is called 'FRAMED'. There is space for a maximum of 512 entries per terminal, numbered Ø to 511. The Ø entry is reserved by the system to contain information about the terminal configuration. (Terminals are numbered from Ø, so that the first group of 512 entries pertains to terminal Ø, the operator terminal.) The structure of the Ø entry is as follows:

Words	Contents	Description
1	MAXFRM	maximum number of frames available on this terminal
2	MAXGRA	maximum number of graphics available on this terminal
3	NLIN	maximum number of lines in frame
4	NELE	maximum number of elements in frame
5	L/R	identifies terminal as local or remote
6	NORTAL	NORTAL switch
7-64		(reserved)

Each directory entry is 64 words long (integer) with the following structure:

Words	Contents	Description
1	SS	satellite ID number
2	YYDDD	year and Julian day of image
3	HHMMSS	time of image
4	BAND	spectral band
5		image line
6		image element
7		image Z (1 for areas)
8		TV line containing image line (5)
9		TV element containing image element (6)
1Ø		line blowup (see also word 33)
11		line blowdown
12		element blowdown
13	YYDDD	creation date of area
14	HHMMSS	creation time of area
15	YYDDD	creation date of frame
16	HHMMSS	creation time of frame
17	AREA	number of area that frame was loaded from
18-32		identification (reserved for expansion)
33		element blowup (if positive; if nonpositive, same as line blowup)

Words	Contents	Description
34		primary key of associated navigation codicil
35		secondary key of associated navigation codicil

(If primary key ≤ Ø, use (SSSYYDDD,HHMMSS) to fetch system navigation.)

RELATED SUB-PROGRAMS

Name	Member	Function
GETFRM	MRGETFRM	read entry from frame directory
PUTFRM	MRPUTFRM	write entry into frame directory

CALLING SEQUENCES

```
GETFRM  SUBROUTINE GETFRM(FRAME,ENTRY)
          input:  FRAME = frame number
          output: ENTRY = 64-word array to contain directory entry

PUTFRM  SUBROUTINE PUTFRM(FRAME,ENTRY)
          input:  FRAME = frame number
                  ENTRY = 64-word array to contain directory entry
```

TERMINAL FORECASTS (FT) NEW STATION IDENTIFIER STATISTICS FILE

Type: LW File
Size: Variable
Initialization: By command MKFT
Documented by: G. Dengel

WORD ALLOCATION

Words	Contents
Ø	n, number of stations in the file
1	ID 1
2	count 1
3	ID 2
4	count 2
:	:
:	:
2n-1	ID n

RELATED PROGRAMS AND SUBPROGRAMS

Name	Member	Function
FTSTAT	MLFTSTAT	list statistics on FTIDs
TRAFFXFT	MLFXFT	FT decoder
MKFT	MLMKFT	initialize file

TERMINAL FORECASTS (FT) OLD STATION IDENTIFIER STATISTICS FILE

Type: LW File
Size: Variable
Initialization: By command MKFT
Documented by: G. Dengel

WORD ALLOCATION

Words	Contents
Ø	n, number of stations in the file
1-n	station call letters
n+1-2ØØØ	Ø
2ØØ1-2ØØn	counts

RELATED PROGRAMS AND SUBPROGRAMS

Name	Member	Function
FTSTAT	MLFTSTAT	list statistics on FTIDs
TRAFFXFT	MLFXFT	FT decoder
MKFT	MLMKFT	initialize file

GOES INGESTOR SCHEDULE FILES

Type: LW file
Size: 8Ø pages, up to 256 pages
Initialization: SCHD
Documented By: J. Rueden

WORD ALLOCATION

GOESCH contains an index followed by schedule records. There can be 1ØØ records in the file at any one time. Each record can have up to 31 entries.

A single page contains all index entries. The index entries consist of eight words for each record:

Words	Contents	Description
1	SSS	satellite identification number
2	YYDDD	year and Julian day of record start time
3	HHMMSS	hours, minutes and seconds of record start time
4	YYDDD	year and julian day of record end time
5	HHMMSS	hours, minutes and seconds of record end time
6		(reserved)
7		terminal number at which this file was made
8		user initials that made this file

Each set of entries (record) is on one page. Each record has an eight word header:

Words	Contents	Description
1	STATUS	record status (not currently used)
2	NENTS	number of entries
3	COMMON DOC	dump frequency (default=2ØØ)
4		debug level option (default=Ø)
5-8		(reserved)

The entries themselves have the following structure:

Words	Contents	Description
1	IMAGE	type of image this entry is for. May be any combination of: 1 = VISSR 2 = Multispectral imaging 4 = Dwell sounding
2	FILTYP	type of file to produce. May be: 1 = Visible area 2 = IR area 3 = Sounding
3	Y-COOR	upper left line in satellite coordinates

Words	Contents	Description
4	X-COOR	upper left element in satellite coordinates
5	Z-COOR	upper left Z-COOR--not used for areas
6	Y-SIZE	number of lines in image
7	X-SIZE	number of elements in image
8	Z-SIZE	number of bytes (1 if area)
9	Y-RES	line resolution
1Ø	X-RES	element resolution
11	Z-RES	Z resolution (not used for areas)
12	Y-Z PRE	number of bytes--must be multiple of four
13	BANDS	bitmap of bands desired in this area or sounding. The rightmost bit is for band 1.
14	FIRST	first area/sounding to be used
15	LAST	last area/sounding to be used
16	INTRVL	HHMMSS of start time, e.g. ØØ3ØØØ means 1 frame per half hour
17	EARLY	HHMMSS of how early image is allowed to start, e.g., ØØØ1ØØ means take any frame up to 1 minute early
18	LATE	tolerance for late starting frames
19	KEY	key entered in add (by user)
20	YORGIN	Y coordinate as entered by user
21	XORGIN	X coordinate as entered by user
22	Y-SIZE	Y size as entered by user
23	X-SIZE	X size as entered by user
24		(reserved)
25		BAND8-STEP/DWELL option. May be: 'S' = step 'D' = dwell ' ' = auto-select (default)
26-32		(reserved)

Words 19-23 are used for recalculation of navigation transformations.

Pages 1Ø1-11Ø are reserved for IR offset settings. The first 8Ø words are directory information. Each satellite ID has a 4-word entry:

Words	Contents
1	SS identifier
2	starting page of offset information
3	starting word of offset information
4	number of 1-word entries

Beginning at the location given in the directory a block of information exists for each satellite. The first word is the SS identifier, followed by 1Ø-word entries containing:

Words	Contents
1	beginning day, YYDDD
2	beginning time, HHMMSS
3	ending day
4	ending time
5	sensor number
6	line offset
7	element offset
8	day entry was made
9	time entry was made
1Ø	(reserved)

Last 17 pages (111-127) are reserved for scratch use by active ingestors. They will be allocated backwards from 127 as needed (based on ingest numbers).

RELATED PROGRAM

Name	Member	Function
SCHD	MLSCHD	GOES schedule record

Also, the ingestor subsystem found in members with the prefix IN.

FOUS STATION LOCATOR FILE

Type: LW file
Size: Variable
Initialization: By decoder TRAFFOUS with list of stations in 'DCIDFOUS'
Documented by: G. Dengel

WORD ALLOCATION

Words	Contents
Ø	n, number of stations in the file
1-n	station call letters
n+1-2n	latitudes, deg * 1ØØØØ, north>Ø
2n+1-3n	longitude, deg * 1ØØØØ, west>Ø
3n+1-4n	elevations, ft
4n+1-5n	state or regional codes

RELATED SUBPROGRAM

Name	Member	Function
IDFOUS	MRIDFOUS	initialize 'IDFOUS' file

CALLING SEQUENCE

```
IDFOUS  FUNCTION IDFOUS(NSTA)
          output:  NSTA =  number of stations in the file
                   function value = Ø if successful
                                    -1 if unsuccessful
```

RAOB STATION LOCATOR FILE

Type: LW file
Size: Variable
Initialization: 'IDRAOB', using list of stations in 'DCRAOBID'
Documented by: G. Dengel

WORD ALLOCATION

Words	Contents
Ø	n, number of stations in the file
1-n	5-digit station identifier
n+1-2n	latitudes, deg * 1ØØØØ
2n+1-3n	longitudes, deg * 1ØØØØ
3n+1-4n	elevations, M
4n+1-5n	state codes (post office abbreviations)

RELATED SUBPROGRAM

Name	Member	Function
IDRAOB	MRIDRAOB	initialize 'IDRAOB' file

CALLING SEQUENCE

IDRAOB FUNCTION IDRAOB(STAT)
output: STAT = Ø if file was made or already exists
-1 if unable to access 'DCRAOBID'
function value = Ø if successful
-1 if unsuccessful

SURFACE AIRWAYS OBSERVATION STATION LOCATOR FILE

Type: LW file
Size: Variable
Initialization: 'IDSVCA' using list of stations in 'DCIDFILE'
Documented by: G. Dengel

WORD ALLOCATION

Words	Contents
Ø	n, number of stations in the file
1-n	4-character (EBCDIC left-adjusted) station call letters
n+1-2n	latitudes, deg * 1ØØØØ
2n+1-3n	longitudes, deg * 1ØØØØ
3n+1-4n	elevations, M
4n+1-5n	state codes (post office abbreviations)

RELATED SUBPROGRAM

Name	Member	Function
IDSVCA	MRIDSVCA	initialize file

CALLING SEQUENCE

IDSVCA FUNCTION IDSVCA(STAT)
output: STAT = Ø if file was made or already exists
-1 if unable to access 'DCIDFILE'
function value = Ø if successful
-1 if unsuccessful

SVC-A STATION LOCATOR FILE

Type: LW file
Size: 6161 words
Initialization: 'IDSVCA' using list of stations in 'DAIDFILE'
Documented by: G. Dengel

WORD ALLOCATION

Words	Contents
Ø	N = number of stations in the file
1-N	4-character (EBCDIC left-adjusted) station call letters
N+1-2N	latitudes, Deg * 1ØØØØ
2N+1-3N	longitudes, Deg * 1ØØØØ
3N+1-4N	elevations, M
4N+1-5N	state codes (post office abbreviations)

RELATED SUBPROGRAM

Name	Member	Function
IDSVCA	MRIDSVCA	initialize file

CALLING SEQUENCE

```
IDSVCA  FUNCTION IDSVCA(STAT)
          output:  STAT = Ø if file was made or already exists
                          -1 if unable to access 'DAIDFILE'
                   function value = Ø if successful
                                    -1 if unsuccessful
```

SYNOPTIC SURFACE STATION LOCATOR FILE

Type: LW file
Size: Variable
Initialization: 'SYNFL' from master weather station text file 'IDSYNOP'
Documented by: G. Dengel

WORD ALLOCATION

Words	Contents
Ø	n, number of stations in the file
1-n	5-digit station identifiers
n+1-2n	4-character (EBCDIC left-adjusted) station call letters
2n+1-3n	latitudes, deg * 1ØØØØ
3n+1-4n	longitudes, deg * 1ØØØØ
4n+1-5n	elevations, M
5n+1-6n	2-character (EBCDIC) state or regional codes
6n+1-7n	2-character (EBCDIC) national codes

RELATED PROGRAMS

Name	Member	Function
SYNFL	MLSYNFL	build 'IDSYN' from text file 'IDSYNOP'
SYNFLR	MLSYNFLR	rearrange 'IDSYN' for USA bias

MASTER WEATHER STATION TEXT FILE

Type: LW file (EBCDIC 4-characters per word)
Size: Variable
Initialization: Read from UNIVAC tape by background job in MLIDSYNO
Documented by: G. Dengel

WORD ALLOCATION

Each station occupies one 12Ø-character record in the format:

Characters	Contents
1-6	numeric station identifier
7	blank
8-11	station call letters
12-13	blank
14	call letter status blank = inactive * = active
16-34	station name
35	blank
	data types:
36	I = 3-hourly synoptic
37	M = 6-hourly synoptic
38	N = off-hourly synoptic
39	O = AERO
4Ø	W = airways
41	T = METAR
42	R = RAOB
43	P = PIBAL
44	A = forecast - TAF
45	L = forecast - PLATF
46	D = radar
47	B = coastal/SMARS
48	C = short range terminal forecast (12-hours or less)
49	blank
	numeric reporting indicator (# = numeric identifier, 2 = numeric identifier, call letters or both):
5Ø	hourly (SA)
51	forecast (FT)
52	radar (SD)
53	extended hourly data base = *
54-55	state or regional code
56	blank
57-58	country code
59	blank
6Ø-63	latitude (DDMM)

Characters	Contents
61	latitude (N or S)
65	blank
66-7Ø	longitude (DDMM)
71	longitude (E or W)
72-76	elevation (M)
77	blank
78	pressure reporting level: Ø = not used 1 = sea level 2 = station 3 = 85Ø mb 4 = 7ØØ mb 5 = 5ØØ mb 6 = unknown A = 5ØØ gpm B = 1ØØØ gpm C = 2ØØØ gpm D = 2517 gpm E = 33Ø8 gpm
79	surface wind and temperature units: Ø = knots and deg F 1 = knots and deg C (max and min temps in deg F) 2 = knots and deg C 3 = m/s and deg C 4 = knots and unknown 5 = m/s and unknown 7 = unknown
8Ø	responsible agency: 1 = Air Force or Army 2 = Navy, Marines and Coast Guard blank = other
81	WMO region: 1 = Africa 2 = Asia 3 = South America 4 = North and Central America 5 = Southwest Pacific 6 = Europe 7 = Antarctic Ø = permanent ships
82-83	runway direction (deg/1Ø)
85-88	U/A latitude (DDMM)
89	U/A latitude (N or S)
91-95	U/A longitude (DDMM)
96	U/A longitude (E or W)
97-1Ø1	U/A elevation (M)
1Ø2	blank
1Ø3	pibal units: Ø = no pibals 1 = knots and feet 3 = m/s and meters 7 = knots and meters 9 = unknown

1Ø4	RAOB units: Ø = no RAOBS 3 = deg C, m/s and meters 7 = deg C, knots and meters 9 = unknown
1Ø5	upper air variations from WMO code: Ø = unknown 1 = pibals reported in feet 3 = may report 3ØØ, 6ØØ and 9ØØ m AGL winds in addition to MSL winds 4 = Chinese u/a type using 'C' series codes, also includes variation 3 U/A 7 = no variation
1Ø6	upper air instrument type indicator
1Ø7	blank
1Ø8	AFGWC met watch indicator: blank = non met watch station M = met watch station
1Ø9	AFGWC coastal station indicator: blank = WMO station not on coast C = located on coast and/or reports sea data
11Ø	AFGWC planning forecast station indicator: blank = non planning forecast station 1 = planning forecast station
111	supplementary aviation reporting station: blank = non aviation station 1 = supplementary aviation station
112-116	blank
117-118	type of last change
119	multiple change indicator: blank = single change * = type of change indicated represents highest priority change
12Ø	blank

RELATED PROGRAM

Name	Member	Function
*	MLIDSYNO	read UNIVAC tape and write 'IDSYNOP' file

* Background job

IMAGE SPOOLS

(blank holds the spool number, 1 to 9)
Type: LW file
Size: 257 pages
Initialization: ISU
Documented by: J. Rueden

The image spools are the GOES ingestors' method of communication to McIDAS space. Messages regarding the health of the ingestor, common documentation, picture start stop, and debug messages (optional) are placed here. Actually, there is one spool for each ingestor. The spool naming convention is 'IMASP'//INGNUM where INGNUM is from 1 to 9. The ingestor number is the INGESAT number.

All times are in HHMMSS.

WORD ALLOCATION

Page Ø contains the last processed pointers and current options and values--4 bytes each.

Words	Contents
1	block
2	MSG Day
3	MSG Time
4	starting byte number
5	SAT Day
6	SAT Time
7	FILEOA (FILEOA=1 → file O&A in master nav)
8	DOGAMM (DOGAMM=1 → calculate gammas for nav codicil)
9	DOCHEB (DOCHEB=1 → file O&A in Chebyshev codicil)
1Ø	DONAV (DONAV=Ø → shut off all nav steps for this spool)
11	DOCODI (DOCODI=1 → file master nav codicil)
12	DOFLAG (DOFLAG=1 → set event flags for messages)
13	DOSKED (DOSKED=1 → do event-driven scheduling for spool)

Pages 1 thru 255 contain the information in variable blocked (but not block spanned) form, as follows:

Each message consists of a header and the indicated number of bytes of information. The next message starts immediately after the last. Messages do not span pages so types Ø and Z8Ø are reserved for end of information in this page.

Header:

Size	Description
1	type code ... message type
3	number of bytes in message
4	YYDDD message created
4	time message created
4	satellite ID
4	satellite YYDDD
4	satellite time
4	scan number
4	checksum flag from common DOC
4	frame flag (IN/OUT of picture)
4	mode flag: 1 = mode A 2 = MSI 4 = DS

Data Type Codes:

Code	Description
1	4 dummy bytes (zeroes)...frame start message
2	area numbers, and mode AA PDL
3	area numbers, and mode A DOC
4	area numbers, and mode AA DOC
5	4B each - (SCAN1,DAY1,TIME1,MS1,BETA1,BETADOT1,SCAN2,DAY2, TIME2,MS2,BETA2,BETADOT2,AREA NUMBERS) --- occurs at frame end
6	4B each - (SCAN1,DAY1,TIME1,MS1,BETA1,BETADOT1,AREA NUMBERS, 4 unused), Orbit and Attitude Block
7	INGE active block-area numbers
8	INGE term block-area numbers
9	FLYWEL debug block (at common DOC frequency)
1Ø	scan 2ØØ Beta block...same format as code 5

Note:

Area numbers now consist of: NAREAS(4B,ZERO if rest of values to be ingnored), AREANUMBERS (1Ø-4B values), nominal YYDDD, NOMINAL HHMMSS, and RTFLAG (Ø if archive retrieval, 1 if realtime).

RELATED PROGRAMS

Name	Member	Function
ISU	MLISU	image spool utility
ESKE	MLESKE	enter event scheduler requests

Also, the ingestor subsystem found in members with the prefix IN.

STORAGE FILE FOR IR ENHANCEMENT TABLES

Type: LW file
Size: 46Ø,8ØØ words
Initialization: Program function
Documented by: A. Weickmann

WORD ALLOCATION

The file allows data for terminals numbered 1 thru 99. Each terminal has 4,6Ø8 words, allocated as 9 tables of 512 words each. Tables are numbered Ø through 8. Tables are accessed by terminal number:

```
Terminal data = (term * 512 * 9) - (512 * 9)
Table data    = (terminal data) + (table * 512)
```

where term equals terminal ID number (LUC(Ø)) and table equals table number requested (Ø-8). The tables contain:

Words	Contents
Ø-255	interpolated data, pixel brightness values (Ø-63)
256-512	endpoints for data interpolation, pixel brightness values (Ø-255)

RELATED PROGRAM

Name	Member	Function
IS	MLIS	create and store IR enhancement tables for DF

KAVOURAS RADAR BILLING FILE

Type: LW file
Size: Variable
Initialization: Self-initializing
Documented by: J. Rueden

INTRODUCTION

Kavouras radar billing files are journals of activity from the program KAVRAD. The files are named KAVBLx, where x is the box number (1 through 5). A file is composed of 2Ø,ØØØ eight-byte entries, the first of which is reserved as a header. The program OLKAVRAD (the process KAVRADx, where x is blank or 2 through 5) writes the records into this file. The program KAVBIL creates output summaries for SSEC billing.

WORD ALLOCATION

Header Definition:

Words	Contents
1	pointer to last updated entry in file (maximum of 2Ø,ØØØ)
2	day pointer was last updated, YYDDD
3	time pointer was last updated, HHMMSS
4-8	not used

Entry Definition:

1	code indicating entry type: =1 - entry active - W8 = entry number =2 - active entry expired - W8 = entry number =3 - inactive entry expired - W8 = entry number =4 - entry deleted - W8 = entry number =5 - image complete block - W8 = bits 16-31 entry number = bits Ø-15 bad count =6 - error block - W8 = error code =7 - ingestor active - W8 = unused =8 - ingestor term - W8 = termination code
2	station ID (4-character EBCDIC)
3	project number
4	initials
5	day, YYDDD
6	time, HHMMSS
7	packed value bits Ø-15 = repeat factor of request bits 16-27 = interval of request, in minutes bits 28-31 = range requested - bit map 28 = 6Ø nm 29 = 12Ø nm 3Ø = 18Ø nm 31 = 24Ø nm
8	type-dependent parameter (see above)

CONTAINS MAIL DIRECTORY AND MAIL MESSAGES

Type: LW file
Size: Maximum 26,ØØ1 words
Initialization: Standard LW file read/write routines
Documented by: A. Weickmann

WORD ALLOCATION

Directory - 6 words in length and a maximum of 1ØØØ directory records (6ØØØ word directory)

Word	Contents
Ø	number of messages in the file
	directory record:
1	message status
2	recipient (either initials, terminal number, or project number)
3	date message was issued, YYDDD
4	time message was issued, HH:MM
5	initials of message sender
6	expiration date of message

Message - 8Ø character (2Ø word) text

Words	Contents
6ØØ1-6Ø2Ø	first message
6Ø21-6Ø4Ø	second message text
6Ø41-...	third - Nth messages at 2Ø word intervals

RELATED PROGRAMS

Name	Member	Function
MAIL	MLMAIL	McIDAS Mailbox
LOGON	MLLOGON	McIDAS Logon Procedure

NAVIGATION FILES

(blank holds a four digit number, ØØØ1 to 9999)
Type: LW file
Size: 8188 - 256 word blocks
Initialization: NVU_MAKE
Documented by: W. Lagerroos

Navigation files are used to store navigation information for a particular satellite, year, date and, where appropriate, time. NAVIØØØ1 is the master navigation file. The other navigation files (ØØØ2 to 9999) are available to users.

WORD ALLOCATION

The first block, block zero, contains a description of the file structure and a pointer to the available space list. The next 366 blocks are day blocks. They are used to store orbit and attitude parameters and pointers to blocks of beta, gamma, Earth edge and landmark information for satellites on a particular day of the year. As more navigation information is made available for a particular satellite on a particular day it is placed in the remaining blocks, which are linked to data chains emanating from the pointers in the day blocks.

Block Ø:

Words	Contents
Ø	points to first word in AVS chain
1	last block in file
2	size of each block in file
3	size of each day block
4	endmark
5-255	endmarks (Z8Ø8Ø8Ø8Ø)

Available Space Chain:

Block Ø, word 1 points to first block of AVSL chain.

Word 1 of each block in the chain points to the next block. If the word is an endmark the block is the first in the Available Space Pool. If the word is -1, there are no blocks of available space in the file.

Block Chains:

Block chains are groups of either day blocks (37 words long) or BLEG (Beta, Landmark, Earth edge and Beta) information (6 words long). These groups are packed into blocks, and the blocks are chained.

Day Blocks:

Day blocks contain orbit and attitude information for a satellite on a particular day and year. They also contain pointers to Earth edge, landmark, beta and gamma blocks elsewhere in the navigation file.

The blocks are 37 words long and they are hashed, by day, into the first 366 records of the navigation file. This means that a block may contain day-block information for the same satellite for more than one year and information for several satellites for the same day-year but all the day blocks in a block are for the same day of the year.

Words	Contents
1	pointer to the next block for this date if the first day block, otherwise, an endmark.
2	SSYYDDD of this day block, or an endmark if the previous day block was the last one in the chain.
3	(reserved)
4	orbit type
5	epoch day
6	epoch time
7	semi-major axis
8	eccentricity
9	inclination
10	mean anomaly
11	argument of perifocus
12	right ascension of ascending node
13	attitude declination (spin-axis)
14	attitude right ascension (spin-axis)
15	attitude - picture center line (spin-axis)
16	spin rate
17-20	camera constants for satellite. Stored in file NAVIGEOM.
21	camera geometry - pitch
22	camera geometry - yaw
23	camera geometry - roll
24-27	(reserved)
28	sun sensor mounting angle
29	skew
30	precession rate
31	precession direction
32	pointer to first landmark block
33	pointer to first Earth edge block
34	pointer to first beta block
35	pointer to first gamma block

BLEG Data:

Beta, Landmark, Earth edge and Gamma (BLEG) data for a particular SSYYDDD combination are stored in 6 word blocks packed into block chains. Pointers to the beginning of these chains are stored in words 32 to 35 of the day block.

Words	Contents
1	if >Ø a pointer to next block in this block chain, if -1 last block in this block chain
2-253	groups of 6 word BLEG data blocks.

An endmark will appear in what would ordinarily be word 1 of the next 6 word blocks in the last block of a block chain.

Gamma Blocks:

Words	Contents
1	time
2	Ø
3	gamma
4	gamma-dot
5-6	(reserved)

Earth Edge Data:

Words	Contents
1	time
2	code for Earth edge
3	line
4	element
5-6	(reserved)

Landmark Data:

Words	Contents
1	time
2	code for landmark
3	line
4	element
5	latitude
6	longitude

Beta Blocks:

Words	Contents
1	scheduled time of beginning of scan
2	code: 1 = early scan 2 = later scan
3	scan number for beta
4	time beta was recorded, HHMMSS
5	fraction of second after time that beta was recorded, milliseconds/1Ø
6	beta

RELATED PROGRAMS

Name	Member	Function
BETAS	MLBETAS	file betas into navigation file
MAKNAV	MLMAKNAV	create navigation entry
ND	MLND	delete navigation data
NL	MLNL	list navigation data
NM	MLNM	move navigation data
NVU	MLNVU	navigation file utility
OANDA	MLOANDA	file orbit and attitude parameters and gammas
XPORTN	MLXPORTN	send navigation to remote CPUs
BETGET	SRBETGET	get betas from navigation file
DGTNAV	MRDGTNAV	fill NAVCOM and BETCOM from navigation file
GETGAM	MRGETGAM	get gamma and gammadot from navigation file

CALLING SEQUENCES

BETGET

```
SUBROUTINE BETGET(IDAY,ITIME,ISCAN1,ISCAN2,ISTIM1,ISTIM2,
 *ISMIL1,ISMIL2,IBET1,IBET2)
Get betas from navigation file.
  output:  IDAY   = date of the scan
           ITIME  = time scan is scheduled to start
           ISCAN1 = first scan line
           ISCAN2 = last scan line
           ISTIM1 = time of first scan line, HHMMSS
           ISTIM2 = time of last scan line, HHMMSS
           ISMIL1 = time of first scan line, milliseconds*1Ø
           ISMIL2 = time of last scan line, milliseconds*1Ø
           IBET1  = beta corresponding to first scan line
           IBET2  = beta corresponding to second scan line
```

DGTNAV

```
SUBROUTINE DGTNAV(IDAY,DSEMIM,DOECCE,DORBIN,DPERHE,DASNOD)
Fill NAVCOM and GETCOM from navigation file.
  input:   IDAY   = Julian day
  output:  DSEMIM = orbit semi-major axis
           DOECCE = orbit eccentricity
           DORBIN = orbit inclination
           DPERHE = orbit perihelion
           DASNOD = orbit ascending node
```

GETGAM

```
SUBROUTINE GETGAM(DAY,TIME,GAMMA,GAMDOT)
Get gamma and gammadot from navigation file.
  input:   DAY    = Julian day
           TIME   = time
  output:  GAMMA  = gamma
           GAMDOT = gamma dot
```

SATELLITE CAMERA CONSTANTS

Type: LW file
Size:
Initialization:
Documented by: W. Lagerroos

WORD ALLOCATION

NAVIGEOM contains camera constants for the various satellites. The words are stored in groups of 12; the first word for a satellite is located by calculating SATNO * 12, where SATNO is the satellite number.

Words	Contents
1	frame geometry - degrees of sweep in line direction (DEGLIN)
2	frame geometry - total number of lines (LINTOT)
3	frame geometry - degrees of sweep in elements (DEGELE)
4	frame geometry - total number of elements (ELETOT)
5	camera geometry - pitch
6	camera geometry - yaw
7	camera geometry - roll
8	(reserved)
9	beta adjustments (IAJUST)
1Ø	time for IAJUST computed (IAJTIM)
11	(reserved)
12	sun sensor mounting angle (ISEANG)

RELATED PROGRAM

Name	Member	Function
NGE	MLNGE	Enter new satellite data into NAVIGEOM file

NSSL LIGHTNING FILE

Type: LW file
Size: Variable
Initialization: Created and modified by the program 'LITN'
Documented by: G. Dengel

WORD ALLOCATION

Words	Contents
Ø	status flag (non-zero if lightning box is unavailable)
1	Julian day of last request, YYDDD
2	GMT time of last request, HHMMSS
3	time increment, seconds, covered by last request
4	number of lightning strokes received as a result of last request
5-2Ø4	text of job to be submitted to request lightning data
2Ø5-...	lightning data (8Ø characters per report)

RELATED PROGRAMS

Name	Member	Function
GETLIT	MLGETLIT	submit request for lightning data
LINT	SLLITN	obtain, decode and file lightning data

BASE MAP FILES

(blank for OUTL is SUPU,SUPW,USAL,CONC,CONE,COPL,COSC,COSE,COW)
(blank for OUTSH is LAK,LAN,LAU,ROA)
Type: LW File
Size: Variable
Initialization: From tape
Documented by: D. Santek

WORD ALLOCATION

Words	Contents
Ø	number of blocks (line segments)
1-6ØØØ	directory for line segments

Words	Contents
1	minimum latitude (REAL*4)
2	maximum latitude (REAL*4)
3	minimum longitude (REAL*4) West positive
4	maximum longitude (REAL*4) West positive
5	beginning word of data start (INTEGER)
6	number of words to read (INTEGER)
7-12	directory for line segment 2
:	:
:	:

5995-6ØØØ	directory for line segment 1ØØØ
6ØØ1-...	data: alternating latitude, longitude (REAL*4)

ADDITIONAL DESCRIPTION

OUTLSUPU - High-resolution USA
OUTLUSAM - Medium-resolution USA
OUTLSUPW - World coastal
OUTLCO(XX) - County outlines
NC - Northcentral
NE - Northeast
PL - Plains
SC - Southcentral
SE - Southeast
W - West
OUTLHPOL - Political boundaries
OUTLHRES - High-resolution world coastal

Cape Canaveral area maps:
OUTSHLAK - lakes
OUTSHLAN - land
OUTSHLAU - launch pads and runways
OUTSHROA - roads

RELATED PROGRAM AND SUBPROGRAMS

Name	Member	Function
IC	MLIC	draw base maps

CALLING SEQUENCE

SUPMAP SUBROUTINE SUPMAP(CNAME,COLOR)
input: CNAME = map file name
COLOR = graphics color level

BASE MAP FILES

(blank is AUST,EURO,POLT,USA)
Type: LW file
Size: Variable
Initialization: from tape
Documented by: D. Santek

WORD ALLOCATION

Words	Contents
Ø	longitude, deg * 1Ø, Ø to 36Ø West positive
1	latitude, deg * 2Ø, Ø 18Ø North positive, odd=pen down; even=pen up
2 ...	alternating longitude, latitude
N-1	to signify end of file

ADDITIONAL DESCRIPTION

OUTLUSA - Medium-resolution USA and North America
OUTLUSAL - Low-resolution USA and North America
OUTLWRLD - World coastal
OUTLEURO - Europe
OUTLPOLI - World political
OUTLAUST - Australia

RELATED PROGRAM AND SUBPROGRAM

Name	Member	Function
MGETPT	MRMGETPT	Retrieve next lat/lon pair

CALLING SEQUENCE

MGETPT FUNCTION MGETPT(FILNAM,LAT,LON,PENFLG)
input: FILNAM = map file name
output: LAT = latitude, deg * 1ØØØØ
LON = longitude, deg * 1ØØØØ
PENFLG = if pen is up, = Ø; if pen is down, = 1
function value = Ø if OK to continue
-1 if EOF

HOLDS BOX DEFINITION TABLES AND LABEL FOR OVERLAY SOFTWARE

Type: LW File
Size: Variable
Initialization: Overlay Save Utility (DTU)
Documented by: R. Dengel

WORD ALLOCATION

The file consists of three parts:

1. Directory (3 pages)
2. Box Definition Table Save Space (256 pages)
3. Label Save Space (512 pages)

Part 1: Directory

The overlay directory occupies the first 3 pages of the file. The first page contains save names and user IDs for overlays saved using the "DTU" command. The first 256 words store user ID (4 characters each), and the next 768 words store the save names (12 characters each). The second page contains accounting information used to identify the user, project number, date created, and date used. Each category occupies 256 words in the above order. The third page is blank.

Part 2: Box Definition Table (BDT) Save Space

The BDT Save Space holds the box definition tables for all defined overlays. Each BDT occupies 1 page (1Ø24 words), starting at page 4. There is a maximum of 256 tables.

An active overlay box definition table for a terminal resides in the McIDAS communications disk region (contiguous disk pages within the Write-Protected file 'MCTOC'). The BDT occupies the second page of the 0-based file, and may be accessed only with subroutines designed to Read/Write this file. A total of 25Ø possible boxes may be defined from each overlay using the 'DTB MAKE' function.

BDT Directory (2Ø words)

Words	Contents
1	modification flag (Ø = in use, 1 = being built)
2-4	overlay name (1-12 characters)
5	author's ID (1-4 characters -- usually user's initials)
6	number of boxes defined in this overlay
7	number of points for current box (Ø or 1)
8	device on which the overlay was defined (1 = WRRM, 2 = tablet)
9	graphic frame number on which the overlay is drawn
1Ø	maximum line on current input device
11	maximum element on current input device
12	used for tablet alignment
13	ID of user who displayed the overlay -- used to lock graphics
14	label flag (Ø = no labels defined, >Ø = label defined)
15-2Ø	memo (1-24 characters)

BDT Border (4 words)

Words	Contents
1	upper left Y position
2	upper left X position
3	lower right Y position
4	lower right X position

BDT Box Coordinate Buffers (4*25Ø words)

Words	Contents
1	ULY = upper left Y positions
2	ULX = upper left X positions
3	LRY = lower right Y positions
4	LRX = lower right X positions

Part 3: Label Save Space

The Label Save Space holds the definitions of any graphic which has been assigned the boxes using the'DTE' command. Each overlay is assigned a 2-page label space which is divided into 8 256-word blocks. The first of these blocks holds the graphic type code (integer) where 1 = Alpha Text, 2 = Map, and 3 = Subdivided. The following 7 blocks contain information which is used to re-create the graphic. Entries are defined by data type as follows:

Block1	Block2	Block3	Block4	Block5	Block6	Block7	Block8
1	Text1-4	Text5-8	Text9-12	TV Line	TV Elem	Siz/Col	#Chars
2	Map-Typ	Lat1	Lat2	Lon1	Lon2	Col	Empty
3	Box	NROWS	NCOLS	Value	Incre	Size	Color

RELATED PROGRAMS AND SUBPROGRAMS

Name	Member	Function
DTBOX	MLDTBOX	define an overlay box (not for general use)
DTB	MLDTB	overlay building function
DTE	MLDTE	overlay string definition/labeling function
DTU	MLDTU	overlay save utility
TABERA	MLTABERA	erase the interior of specified box(es)
TABCOL	MLTABCOL	set color of overlay box(es)
TABWRM	MLTABWRM	WRRM/string table harness (not for general use)
TABLOC	MLTABLOC	determine location of cursor within a divided or map box
TABUP	MLTABUP	tablet update of specified parameters
TABENT	MLTABENT	enter a label into an overlay
TABSCL	MLTABSCL	divide interior of an overlay box into regions
CNAME	MRCNAME	convert integer number to equivalent box name
CONVRT	MRCONVRT	convert HHMMSS/DDMMSS to "HH:MM"/"DD:MM"
ERABOX	MRERABOX	erase the contents of a box
INAME	MRINAME	convert character type box name to integer
TABTAB	SRTABTAB	determine box name from input position
TABTV	MRTABTV	convert from one scaled device to another

CALLING SEQUENCES

NOTE: Arguments beginning with 'C' are CHARACTER*12. All others are integer.

CNAME CHARACTER FUNCTION CNAME(I)
- input: I = integer value
- output: function value = box name

CONVRT CHARACTER FUNCTION CONVRT(X)
- input: X = integer form of time or lat/lol
- output: function value = character form of time or lat/lon (HH:MM or DD:MM)

ERABOX SUBROUTINE ERABOX(I,CBOX)
- input: I = box number
  CBOX = box name

INAME INTEGER FUNCTION INAME(CBOX)
- input: CBOX = box name
- output: function value = integer value of box name

TABTAB INTEGER FUNCTION TABTAB(IDEV,X,Y,CNAME,NBOX,ULIN,UELE,LLIN,LELE)
- input: IDEV = input device:
  1=WRRM
  2=TAB1, tablet measuring 24ØØ x 24ØØ
  3=TAB2, tablet measuring 3ØØØ x 3ØØØ
  X,Y = element/line position of input device
- output: CNAME = string name to be executed
  NBOX = overlay box number
  ULIN,UELE = upper left line/element position
  LLIN,LELE = lower right line/element position

TABTV SUBROUTINE TABTV(IDEV,IULY,IULX,ILRY,ILRX)
- input: IDEV = input device:
  1=WRRM
  2=TAB1, tablet measuring 24ØØ x 24ØØ
  3=TAB2, tablet measuring 3ØØØ x 3ØØØ
- output: IULY = upper left line coordinate
  IULX = upper left element coordinate
  ILRY = lower right line coordinate
  ILRX = lower right element coordinate

PATH DATA FILE

Type: LW file
Size: 1Ø24 word blocks
Initialization: Standard LW file routines
Documented by: A. Weickmann

The LW file, PATHFIND, contains all the paths defined by the PATH command. A path consists of a series of latitude, longitude pairs with an optional label for each pair. The paths are accessible from all terminals and by all individuals. A path is constructed using a linked list. An available links list and a data directory are also constructed using a linked list technique. File space is allocated in 1Ø24 word blocks.

WORD ALLOCATION

Words	Contents
Ø	starting word location of next available 1Ø24 word block
1ØØ1	location of next available data record
1ØØ2	starting word of the directory linked list
1ØØ3	location of next available directory record

data record:

Words	Contents
1	latitude
2	longitude
3-7	label
8	back-link (tail)
9	fore-link (head)

directory record:

Words	Contents
1	CNAME(1)
2	CNAME(2)
3	CNAME(3)
4	path creation date
5	user initials (created the path)
6	minimum latitude
7	maximum latitude
8	minimum longitude
9	maximum longitude
1Ø	starting word of path
11	back-link (tail)
12	fore-link (head)

Available Links Block

Data Record: The next available 1Ø24-word block is determined from word Ø. The 1Ø24-word block has a pointer to every ninth word. When all data records are filled within the block, another new block is selected and linked by every ninth word.

Directory Record: The next available 1Ø24-word block is determined from word Ø. The 1Ø24-word block has a pointer to every twelfth word. When all data records are filled within the block, another new block is selected and linked by every twelfth word.

Upon deletion of a data record, the word location of the record is linked to the top of the available data record linked list; that word location is stored in word 1ØØ1.

Upon deletion of a directory record, the word location of the record is linked to the top of the available directory linked list; that word location is stored in word 1ØØ3.

RELATED PROGRAM AND SUBPROGRAMS

Name	Member	Function
PATH	MLPATH	create and store lists of latitudes, longitudes, and labels
PTHDIR	MRPTHDIR	add, delete, and list the directory
PTHADD	MRPTHADD	add data records to an existing list

CALLING SEQUENCES

PTHDIR
```
SUBROUTINE PTHDIR(COPT,CNAME,FLAG)
  input:  COPT  = directory option ('MAKE', 'QUIT', or 'DIR')
          CNAME = path name
  output: FLAG  = status of directory option
```

PTHADD
```
SUBROUTINE PTHADD(CNAME,INDEX,NPTS,ALAT,ALONG,CLAB,FLAG)
  input:  CNAME = path name
          INDEX = position in list of the point to be added
          NPTS  = number of points to be added
          ALAT,ALON = latitude and longitude arrays
          CLAB  = point label, 2Ø characters
  output: FLAG  = status of the add
```

KAVOURAS RADAR SCHEDULE FILE

Type: LW file
Size: Variable
Initialization: RSKU_INIT
Documented by: R. Dengel

WORD ALLOCATION

Words 1-23 are the master directory.

Page	Words	Contents
Ø	Ø	scheduler on/off (Ø=off, 1=on)
	1	scheduler functions (Ø=none, 1=reset, 2=restart, 3=end)
	2	master MOD flag (Ø=no modification, 1=modified)
	3	maximum number of entries in file (currently=25Ø)
	4	word offset to first entry (currently=2Ø48)
	5	word offset to radar station list (currently=1,Ø24,ØØØ)
	6	message device (CHARACTER*4; N=none, P=hardcopy)
	7	YYDDD of most recent state table update
	8	HHMMSS of most recent state table update
	9	YYDDD of most recent CRAB schedule update
	10	HHMMSS of most recent CRAB schedule update
	11-14	reserved
	15	ID of locally connected radar site for box 1
	16	ID of locally connected radar site for box 2
	17	ID of locally connected radar site for box 3
	18	ID of locally connected radar site for box 4
	19	ID of locally connected radar site for box 5
	2Ø-23	reserved
State table section:		25Ø entries, each containing 4 words.
	24	scheduler box number for entry number 1 (=<Ø if not yet assigned)
	25	MOD flag for entry number 1 (-1=not exist, Ø=no MOD, 1=suspended, 2=added/released, 3=edited, 4=deleted)
	26	YYDDD of next image for entry number 1
	27	HHMMSS of next image for entry number 1
	.	.
	.	.
	.	.
	1Ø2Ø	scheduler box number for entry number 25Ø
	1Ø21	MOD flag for entry number 25Ø
	1Ø22	YYDDD of next image for entry number 25Ø
	1Ø23	HHMMSS of next image for entry number 25Ø
1	Ø1Ø23	reserved

Pages 2 through 999 are reserved to hold the actual scheduler entries. Each entry is 64 words long. The word position of the first entry in this region is found in absolute word 3 of the master directory. The contents of each entry are listed below.

Page	Words	Contents
2-999	1	reserved
	2	station ID (4-character EBCDIC)
	3	latitude (DDDMMSS)
	4	longitude (DDDMMSS)
	5-1Ø	telephone number (CHARACTER*24)
	11	begin day (YYDDD)
	12	begin time (HH:MM:SS)
	13	polling interval (MMM)
	14	polling repeat factor (9999 → many)
	15	range flag (bit map LSB ON → range1...)
	16	Kavouras map flag (Ø=no map, 1=map on)
	17	begin area for range number 1
	18	end area for range number 1
	19	begin area for range number 2
	2Ø	end area for range number 2
	21	begin area for range number 3
	22	end area for range number 3
	23	begin area for range number 4
	24	end area for range number 4
	25-3Ø	station text (CHARACTER*24)
	31	project number
	32	expiration YYDDD (includes picture lag time)
	33	expiration HHMMSS (includes picture lag time)
	34	tolerance, in minutes (maximum of 255)
	35	initials of person who last changed or released this entry
	36-64	reserved

The radar station ID table begins at page 1ØØØ. It contains information for each valid site that may be polled. The total number of stations in the table is given by the first word of the table. To insert a new station into the file, the edit member DAIDRADR must first be updated with the new information. Following that, the command IDRADR is run to place the updated table in the scheduler file RADRSCH.

Let k = offset to start of station ID file (word 4 of the directory). The position of the start of individual blocks in the ID table is listed below.

Words	Contents
k	n, number of stations in the file
k+1	4-character (EBCDIC) station call letters
n+k	4-character (EBCDIC) state ID (P.O. abbreviation)
2n+k	latitudes (DDDMMSS)
3n+k	longitudes (DDDMMSS)
4n+k	elevation
5n+k	4-character (EBCDIC) radar class (A or C)
6n+k	telephone number (CHARACTER*24)
12n+k	city text (CHARACTER*24)

RELATED PROGRAMS AND SUBPROGRAMS

Name	Member	Function
READR	ORREADR	read radar scheduler entry
WRITR	ORREADR	write radar scheduler entry
LOKRAD	ORREADR	lock radar scheduler file
UNLKRD	ORREADR	unlock radar scheduler file

CALLING SEQUENCE

READR SUBROUTINE READR (ENTRY,ARRAY)
Read a radar scheduler entry.
input: ENTRY = entry number
output: ARRAY = array to hold entry

WRITR SUBROUTINE WRITR (ENTRY,ARRAY)
Write a radar scheduler entry.
input: ENTRY = entry number
output: ARRAY = array to hold entry (64 words)

MCGILL RADAR PRODUCT FILES

Type: LW file
Size: Variable
Initialization: MGU
Documented by: R. Dengel

WORD ALLOCATION

Product file(s): file RAPIDPRx (where x = handler number 1, 2...)
Page 1 is the control section:

Word	Contents
1	product handler status (Ø=off, 1=on)
2-1Ø24	unused

Page 2 is the directory:

Word	Contents
1	number of the current packet
2	current product code number
3	packet size, in words (currently=64)
4	maximum packet number, maxpac (currently=2Ø48)
5	word offset to assignment table (currently=1Ø24)
6	word offset to product table (currently=3Ø72)
7	word offset to packet cylinder
8-1Ø24	unused

Page 3 is the assignment table:

A product number is assigned to each complete product in the file. It ranges from 1 to the maximum packet number (maxpac). The table consists of half-word entries (1 for each packet) that hold the assigned number for the product. Numbers are assigned in sequence from 1 to maxpac. When the product number reaches maxpac, it recycles to 1. All product slots in the table that are not assigned are set to a value of zero.

If a new product uses packet slots assigned to an old product, the old product is terminated by setting all its product slots to zero. The current value of the product number is stored in word 3 of the file directory and is modified by packet write and deletion subroutines. The start of this table is listed in word 6 of the file directory.

Word	Use
1	product assignment number for packets 1 and 2
2	product assignment number for packets 3 and 4
3	product assignment number for packets 5 and 6
.	.
.	.
.	.
1Ø24	product assignment number for packets 2Ø47 and 2Ø48

Page 4 is the product table; it holds information describing products that currently exist in the cylinder. Each entry consists of the following three words:

(1) product: type number/altitude/resolution/request number
(2) YYDDD of the product
(3) HHMMØØ of the product

The number of entries in the file depends on the size of the cylinder. Because each packet in the cylinder is potentially a product, room is allotted for one entry per packet.

For example: For a cylinder length of 2Ø48 packets, there are 2Ø48 entries in the product table, each with a length of 3 words.

The start of the product table is in word 7 of the file directory.

Word	Contents
1	product: type number/altitude/resolution/request number
2	YYDDD for assignment number 1
3	HHMMØØ for assignment number 1

The starting address of the packet cylinder depends on the length of the cylinder. Because room must be made for the product table, the position of the start of the table varies as the length of the product table varies. Each entry in the cylinder has the same length, defined in word 5 of the file directory. The total number of packets in the file is listed in word 6.

MCGILL RADAR REQUEST FILE

Type: LW file
Size: Variable
Initialization: MGU
Documented by: R. Dengel

WORD ALLOCATION

Word	Contents
1-1Ø24	request file control block
1Ø25-2Ø48	product destination defaults for scheduled/requested products
2Ø49-2112	request entry number 1 (64 words)
2113-2176	request entry number 2 (64 words)

Page 1, words 1-1Ø24, is the request file control block:

Word	Contents
1	request file status (Ø=off, 1=on)
2	maximum number of requests in file (currently 223)
3	word offset to start of entries (currently 2Ø48)
4	number of requests waiting to be decoded
5-24	unused
25-524	status table (Ø=none, 1=pending, 2=active)
525-1Ø24	decoder queue

Page 2 is the product destination defaults for scheduled/requested products:

Word	Contents
1-1Ø	product default slot number 1
11-2Ø	product default slot number 2
:	
1Ø11-1Ø2Ø	product default slot number 1Ø2

Page 3 is the destination default parameters, 1 set per product:

Word	Contents
1	product type (1=CAPPI, 8=ET, 10=VS, 11=PF)
2	product altitude for CAPPI and ET; = Ø for VS and PF
3	product resolution for CAPPI and ET; = Ø for VS and PF

Page 4 is request entry documentation:

Word	Contents
1	request status (same as status table)
2	priority of request (1=low, 2=high)
3	request type (1=add, 2=delete, 3=command)
4	active period of request (units of 5 minutes)
5	frequency of request (1,2,3,4,6,12 - 5,1Ø,15,2Ø,3Ø,6Ø)
6	product type (1=CAPPI, 8=ET, 1Ø=VS, 11=PF)
7	day request was sent (YYDDD)
8	time request was sent (HHMMSS)
9	reserved
1Ø	reserved
11-2Ø	destination-dependent parameters (request type=add)
21-47	product-dependent parameters
48-64	packed request (66 bytes ready for transmission)

Destination-dependent parameters, words 11-2Ø, are broken down as follows:

Word	Contents			
11-13	action type (decoder name if different from default)			
14	destination type (area, MD, grid, LW...)			
	area	grid	MD	LW
15	cyl-beg	cyl-beg	cyl-beg	name (1:4)
16	cyl-end	cyl-end	cyl-end	name (5:8)
17	cyl-ptr	cyl-ptr	cyl-ptr	name (9:12)
18				index-ptr
19				
2Ø				

Request to generate a product may contain destination other than default for this product (see command MGE).

Product-dependent parameters, words 21-47, are broken down as follows:

CAPPI parameters:

Word	Contents
21	altitude of CAPPI (1-16)
22	precipitation algorithm (1=average, 2=maximum)
23	resolution of CAPPI (1=1km, 2=2km, 3=4km)
24-47	unused

ET parameters:

Word	Contents
21	altitude threshold (1-16)
22	intensity threshold (mm/hr*1Ø, limit Ø to 2ØØØ)
23	resolution (1=1km, 2=2km)
24-47	unused

VS parameters:

Word	Contents
21	beginning range (units of km * 1Ø)
22	beginning bearing (degrees * 1Ø)
23	ending range (units of km * 1Ø)
24	ending bearing (degrees * 1Ø)
25	corridor width (km * 1Ø)
26	horizontal resolution (km * 1Ø)
27-47	unused

PF parameters:

Word	Contents
21-47	unused

FAA6Ø4 LINE TRAFFIC ROUTING TABLE

Type: LW file
Size: 932 words
Initialization: Created and modified by the command 'ROOT'
Documented by: G. Dengel

WORD ALLOCATION

For each catalog number (2ØØ - 432), there are 4 words reserved for the names of decoders to which all text received under that catalog header should be routed. Decoder names are 4-characters (EBCDIC left-adjusted) long, one per word. Catalog 2ØØ begins at word Ø.

RELATED PROGRAMS

Name	Member	Function
ROOT	MLROOT	create, maintain and list FAA6Ø4 traffic routing table
SVCT	SLSVCT	FAA6Ø4 traffic handler

SURFACE AIRWAYS OBSERVATION (SAO) NEW STATION IDENTIFIER STATISTICS FILE

Type: LW File
Size: Variable
Initialization: by decoder TRAARZ
Documented by: G. Dengel

WORD ALLOCATION

Words	Contents
Ø	n, number of stations in the file
1	ID 1
2	count 1
3	ID 2
4	count 2
:	:
:	:
2n-1	ID n
2n	count n

RELATED PROGRAMS AND SUBPROGRAMS

Name	Member	Function
STATS	MLSTATS	list statistics on SAO IDs
TRAARZ	MRTRAARZ	SAO decoder

SURFACE AIRWAYS OBSERVATION (SAO) OLD STATION IDENTIFIER STATISTICS FILE

Type: LW File
Size: Variable
Initialization: by decoder TRAARZ
Documented by: G. Dengel

WORD ALLOCATION

Words	Contents
Ø	n, number of stations in the file
1-n	station call letters
n+1-2ØØØ	Ø
2ØØ1-2ØØn	counts

RELATED PROGRAMS AND SUBPROGRAMS

Name	Member	Function
STATS	MLSTATS	list statistics on SAO IDs
TRAARZ	MRTRAARZ	SAO decoder

SATELLITE INGESTOR SCHEDULE FILES

Type: LW File
Size: 28Ø pages
Initialization: None
Documented by: J. Hagens

SATSKD contains satellite schedule information. The first page, the SSS/YYDDD/HHMMSS windows, describes the satellites and times requested. The next pages contain entries that describe input signal types, output product types, and geography of each sector requested. A maximum of 2ØØ windows is set, with a maximum of 15 entries per window.

Note: This schedule file is intended to contain imagery ingestion information. 'TIP ' and 'AUX ' (block 1) data are not included. Pages are numbered from Ø; words are numbered from 1.

WORD ALLOCATION

Page Ø is the windows page, structured as follows:

Words	Contents
1	≤ 0 → schedule off; = 1 → schedule on
2	wake-up bits, one per ingestor LSB=INGE1, etc. BIT ON means wake up; BIT OFF means stay asleep
3-24	reserved
25-1Ø24	windows, numbered 1 to 2ØØ

Each window descriptor is five words long as follows:

Words	Contents
1	SSS, satellite identification number; <Ø means available
2	YYDDD, year and Julian day of window start time
3	HHMMSS, hours, minutes, and seconds of window start time
4	YYDDD, year and Julian day of window end time
5	HHMMSS, hours, minutes, and seconds of window end time

Pages 1-2ØØ are the entries pages, one set of entries per page, as follows:

Pages	Words	Contents
1-2ØØ	1-64	header

Words	Contents
1	≤ Ø → window suspended, =1 → window active (released)
2	nents, number of entries
3	common doc dump frequency (default=2ØØ
4	debug level option (default=Ø)
5	last YYDDD an entry in page changed
6	last HHMMSS an entry in page changed
7	terminal number where file was made
8	users initials of file's creator
9-64	reserved

65-1Ø24 64-word entries, each structured as follows:

Words	Contents
1	ACTIVE, ≤ Ø → window suspended, =1 → active (released)
2	LDAY, last YYDDD entry was modified
3	LTIME, last HHMMSS entry was modified
4	INGE, ingestor number (Ø=autoselect)
5	TRACK, 'AUTO' if ingestor should track satellite; 'INT ' if ingestor should operate only during times specified
6	INTRVL, HHMMSS of start time, e.g., ØØ3ØØØ is 1 frame per half hour
7	EARLY, HHMMSS early tolerance for image start; e.g., ØØØ1ØØ means take any frame up to one minute early
8	LATE, HHMMSS late tolerance for image start
9	FOLLOW: YES → ignore y-coor (start at image start) NO → start at y-coor (default)
1Ø	SIGNAL, 4-character string describing signal, listed below
11	SIGCODE, code for input signal type, stored as a bit map. For example: 1 'A ' - GOES Mode A 2 'AADS' - GOES Mode AA DS 4 'AAMS' - GOES Mode AA MSI 8 'DS ' - GOES Mode AAA DS 16 'MSI ' - GOES Mode AAA MSI 'LAC ' - TIROSN (AVHRR) local area coverage 'GAC ' - TIROSN (AVHRR) global area coverage 'FLYO' - TIROSN (AVHRR) flyover (HRPT) data

Words	Contents
12	PROTYP, product type, describes area type to produce: 'RAW ' - raw ingest (COMDOC and VIS) 'RAWI' - raw ingest (IR only) 'VIS ' - visible 'IR ' - single band IR 'MULT' - multibanded image
13	Y-COOR, upper-left line in satellite coordinates
14	X-COOR, upper-left element in satellite coordinates
15	reserved
16	Y-SIZE, number of lines in image
17	X-SIZE, number of elements in image
18	Z-SIZE, number of bytes per pixel
19	Y-RES, line resolution
2Ø	X-RES, element resolution
21	Z-RES, number of bands (size area created)
22	FIRST, first area to be used
23	LAST, last area to be used
24	BANDMAP, bit maps for bands requested
25	reserved
26	OPTION, sample/average 'S'=sample,'A'=average
27	DOC, length of DOC section
28	CAL, length of CAL section
29	LEV, length of LEV section
3Ø-39	reserved
4Ø	KEY, coordinate transformation key
41	Y-ORIGIN, y-coordinate as entered by user
42	X-ORIGIN, x-coordinate as entered by user
43	Y-SIZE, y-size as entered by user
44	X-SIZE, x-size as entered by user
45-64	reserved

Pages 2Ø1-229 are reserved for IR offset settings. The first 8Ø words are directory information. Each satellite ID has a 4-word entry, as follows:

Word	Contents
1	SS identifier
2	starting page of offset information
3	starting word of offset information
4	number of 1Ø-word entries

A block of information is included for each satellite, beginning at the location specified in the directory. The first word, the SS identifier, is followed by 1Ø-word entries structured as follows:

Word	Contents
1	beginning day (YYDDD)
2	beginning time (HHMMSS)
3	ending day
4	ending time
5	sensor number
6	line offset
7	element offset
8	day entry was made
9	time entry was made
1Ø	reserved

Pages 23Ø-255 are reserved for scratch use by active ingestors. They are allocated backward from 255 as needed, based on INGESAT numbers.

Pages 256-274 are reserved for scheduling control tables, to specify restrictions on parameters entered via SSKE and SSKU depending upon signal type. This section is updated by background program SBSATTBL. The general format of the file is card images in the following three forms:

```
*  SS SS SS BETAOPT CHKPGMNAME
-  SIG CODE SIG CODE SIG CODE
+  SIGNAL PRODUCT FOLLOWDEFAULT NBANDSDEFAULT
```

Quoted fields are comments.

Page 275 is reserved for the 'NAME's of particular sectors. This page is structured as 1ØØ ten-word entries with a 24-word header, as follows:

Words	Contents
1	last updated YYDDD
2	last updated HHMMSS
3-24	reserved

Each name is structured as follows:

Words	Contents
1-3	sector name
4-9	six parameters in I * 4
1Ø	reserved

For example:

```
GLO     IU_8Ø1_1_34ØØ_38ØØ_4
```

RELATED PROGRAMS

Name	Member	Function
SSKE	MLSSKE	enter schedule requests
SSKL	MLSSKL	list schedule entries
SSKU	MLSSKU	change/update schedule

STRING TABLE SAVE FILE

Type: LW File
Size: 2Ø47 pages
Initialization: LWU MAKE SAVESTR
Documented by: R. Dengel

WORD ALLOCATION

The String Table Save File is divided into 2 separate regions. The directory region occupies the first 4 pages of the file and contains the following information:

Words	Contents
1-512	author code - user ID of string table author
513-2Ø48	save name - 12 character name given table when saved
2Ø49-4Ø96	accounting information (4 words per saved table)

Words	Contents
1	project number
2	user terminal number
3	date created
4	date of last restore

The next region contains the text of each saved table arranged in sets of 15 pages each. The position of a saved table is determined by its directory entry location. (Directory entry 1 refers to the first 15 pages of the save table space.) Within the save space there is room for up to 512 saved string tables. The organization of words within a save space entry is identical to the form described by the "STRTABLE" file documentation.

Words	Contents
4Ø97-19457	saved string table 1
19458-34818	saved string table 2
.	.
.	.
.	.
7853Ø56-7868416	saved string table 512

RELATED PROGRAMS

Name	Member	Description
TU	MLTU	string table save utility

SCHEDULED MCIDAS COMMANDS

Type: LW File
Size: 1Ø23 pages (first extent)
Initialization: SUSKINIT
Documented by: R. Dengel

WORD ALLOCATION

Single entry (4ØØ possible)

Words	Contents	Description
1	SKNEXT	time of next scheduled execution (all times are kept internally in seconds since Jan 1, 1972)
2	SKNREM	number of executions remaining
3	SKINTV	interval between executions
4	SKTOL	late tolerance (should scheduler be delayed)
5	SKTERM	terminal at which the command executes
6	SKID	ID number or name assigned when command was entered into schedule
7	SKNAME	initials of person who entered command into schedule
8	SKPROJ	project number under which command executes
9	SKSTRT	time of first execution of command
1Ø	SKNTIM	total number of times command is to execute
11-71	SKTEXT	text of legal McIDAS command

SKEDFILE is an LW file; that is, it is accessed via subroutines LWGET/LWPUT, which enables it to be treated as a single large stream of words. Physically, it is divided into 2 regions. First is a region which contains the first part of the directory (everything up to SKPROJ). This is separated out so that the scheduler itself ('SKED') can quickly scan all the entries to decide which are due for execution. Next is a region which contains the rest of the directory and the text (SKTEXT). This section of the file need only be read when listings are made or when a command is ready for actual execution. This file division is invisible to all parts of the schedule system which access the file via 'SKID'.

RELATED PROGRAMS

Name	Member	Function
SKE	MLSKE	make entries into schedule file
SKED	MLSKED	real-time schedule program; cause automatic execution of McIDAS commands
SKL	MLSKL	list all or part of entries in the file
SKU	MLSKU	schedule utility functions

STRING TABLE FILE

Type: LW file
Size: 1485 pages (15 pages/terminal)
Initialization: Standard LW file creation
Documented by: R. Dengel

WORD ALLOCATION

The McIDAS string table file holds all the tables currently active on the system. The file is partitioned into 15 page blocks which contain the following information:

Words	Contents
1-256	length of each string (in bytes)
257-1Ø24	string names (character*12)
1Ø25-4Ø96	(reserved)
4Ø97-1Ø47553	256 strings (16Ø bytes/string)

These blocks are arranged in the file by increasing terminal number. All other string tables not currently active reside in the string table save file SAVESTR.

RELATED PROGRAMS AND SUBPROGRAMS

Name	Member	Function
TD	MLTD	delete strings
TE	MLTE	enter and edit strings
TL	MLTL	list strings
LBGET	MRLBGET	write to table
LBPUT	MRLBPUT	read from table

CALLING SEQUENCES

LBGET
```
FUNCTION LBGET(CNAME,COUT)
  input:  CNAME = string table name
  output: COUT  = to contain string value
```

LBPUT
```
FUNCTION LBPUT(CNAME,COUT)
  input:  CNAME = string table name
          COUT  = to contain string value
```

SYSTEM OPTIONS TABLE

Type: LW file
Size:
Initialization: background job SBSYSKEY
Documented by: T. Whittaker

INTRODUCTION

The contents of SBSYSKEY are a system-wide set of parameters; SYSKEY contains values that are unique to an individual system, values that are different for each system. SYSKEY is a data file containing global UC values, the first 3ØØ words have meanings that correspond exactly with global UC. The LW file 'SYSKEYTABLE6' is initialized by background job SBSYSKEY. The meaning of each word in SYSKEYTABLE6 is pre-assigned. SYSKEY is accessed through FORTRAN-callable integer function KSYS.

WORD ALLOCATION

Word	Contents
1-299	identical to global UC (SYSCOM)
3ØØ-1ØØØ	unassigned
1ØØ1-2999	automatic logon of terminals during initialization
3ØØØ-3Ø99	area numbers for real-time satellite input
31ØØ	first reserved grid file number for NMC products
31Ø1	number of reserved grid files for NMC products

TAF CODE TERMINAL FORECASTS NEW STATION IDENTIFIER STATISTICS FILE

Type: LW File
Size: Variable
Initialization: by command MKFT
Documented by: G. Dengel

WORD ALLOCATION

Words	Contents
Ø	n, number of stations in the file
1	ID 1
2	count 1
3	ID 2
4	count 2
:	:
:	:
2n-1	ID n
2n	count n

RELATED PROGRAMS AND SUBPROGRAMS

Name	Member	Function
FTSTAT	MLFTSTAT	list statistics on TAF IDs
TRAFTAF	MLTAF	TAF decoder
MKFT	MLMKFT	initialize file

TAF CODE TERMINAL FORECASTS OLD STATION IDENTIFIER STATISTICS FILE

Type: LW File
Size: Variable
Initialization: By command MKFT
Documented by: G. Dengel

WORD ALLOCATION

Words	Contents
Ø	n, number of stations in the file
1-n	station call letters
n+1-2ØØØ	Ø
2ØØ1-2ØØn	counts

RELATED PROGRAMS AND SUBPROGRAMS

Name	Member	Function
FTSTAT	MLFTSTAT	list statistics on TAF IDs
TRAFTAF	MLTAF	TAF decoder
MKFT	MLMKFT	initialize file
MKFT	MLMKFT	initialize file

HIGH-RESOLUTION TOPOGRAPHY MAP

Type: LW file
Size: 1142 pages
Initialization: none
Documented by: J. Rueden

INTRODUCTION

Resolution of the file TOPOHRES is 1/6 degree (horizontal) and 3Ø meters (vertical).

WORD ALLOCATION

There are 144 INTEGER*2 elements per record. (24° of LAT at same LON.) There are 15 records per latitude. Longitudes range Ø to 359 5/6; east is positive. Latitudes range Ø to 9Ø, -1/6 to -9Ø. Each element is FLAG*512 + height.
FLAG = Ø for water
> Ø for land
Height is in 1ØØs of feet.

RELATED SUBPROGRAM

Name	Member	Function
HRTOPO	URHRTOPO	get high resolution elevation and surface characteristics, given 'LAT' and 'LON'

LOW-RESOLUTION TOPOGRAPHY MAP

Type: LW file
Size: 35 pages
Initialization: none
Documented by: J. Rueden

INTRODUCTION

Resolution of the file TOPOLRES is 1 degree (horizontal) and 1 meter (vertical). TOPOLRES is averaged from TOPOHRES (high resolution).

WORD ALLOCATION

There are 192 INTEGER*4 words per record. (One full latitude). The first 12 are bit map flags (Ø=sea, 1=land). There are 36Ø INTEGER*2 elements that list height, in meters. Longitudes range Ø to 359; east is positive. Latitudes range 9Ø to -87.

RELATED SUBPROGRAM

Name	Member	Function
LANSEA	URLANSEA	get elevation and surface characteristics from 'LAT' and 'LON'

TERMINAL REQUEST BLOCK

Type: Temporary memory, not on disk
Size: 256 bytes
Initialization: McIDAS executive, timer routine, and scheduler
Documented by: J. Benson

The terminal request block, or TRB, is a transaction-oriented block which is created for each line entered from a terminal device; the TBR lives as long as the request defined by the text of that line is being processed. The block contains typed text and a snapshot of the terminal state at the moment the user strikes the Enter key, as well as processing information used in status reports and accounting.

Because the typical typed line contains one or more McIDAS commands, the primary purpose of the TRB is to carry the command text while the command is being scanned (parsed) and processed. In addition, TRBs are created to contain the responses to WTRR (write to terminal and await response) subroutine calls.

WORD ALLOCATION

TRB format (FORTRAN subscripts refer to an INTEGER*2 array):

Byte Offset	FORTRAN Subscript	Contents
Ø-3	-3,-2	time-stamp-1; time at which command processing is completed
4-7	-1,Ø	time-stamp-2; elapsed time of command (First two entries are for accounting only, and are placed in the TRB by the McIDAS executive. Both are INTEGER*4 (fullword) quantities.)
8-87	1-4Ø	command text (first part)
88-89	41	current image frame
9Ø-91	42	loop upper bound (image frames)
92-93	43	loop lower bound (image frames)
94-95	44	image frame control word
96-97	45	current graphics frame
98-99	46	loop upper bound (graphics frames)
1ØØ-1Ø1	47	loop lower bound (graphics frames)
1Ø2-1Ø3	48	graphics frame control word
1Ø4-1Ø5	49	cursor size in lines (vertical size)
1Ø6-1Ø7	5Ø	cursor size in elements (horizontal size)
1Ø8-1Ø9	51	cursor position (line coordinate)
11Ø-111	52	cursor position (element coordinate)
112-113	53	cursor control word
114-115	54	(reserved)

116-117	55	TRB type: 0 = unsolicited command 1 = unsolicited tablet pen-down event 2 = solicited response (r... Type) 3 = solicited delay/status response
118-119	56	project number for which the TRB created
		Next 3 entries are periodically maintained by the McIDAS executive's timer routine throughout the life of the TRB.
12Ø-121	57	# of disk file I/O's performed by the commands(s)
122-123	58	# of terminal I/O's performed by the commands(s)
124-125	59	CPU time (secs x 10) used by the command(s)
126-127	6Ø	terminal number for which the TRB created
128-131	61-62	ID of user (4 chars) for which TRB created (present only when the TRB is created by the scheduler)
132-133	63	data tablet -- pen x-coordinate; (0,0) is lower-left
134-135	64	data tablet -- pen y-coordinate
136-137	65	data tablet -- mode: 2 or 3 = pen down 1 = pen up but proximate (x and y coordinates are valid) 0 = pen up and far (x and y coordinates are not valid)
138-171	66-82	(reserved)
172-251	83-122	command text (second part; present only when TRB created by the scheduler; (binary) Ø otherwise.)
252-255	123-124	address of word in TRB-address list which corresponds to this TRB (plugged when TRB allocated from TRB pool; used to return this TRB to pool when processing completed.)

NOTES:

1) TRB's are created a) by the McIDAS executive when text is received from a terminal, b) by the timer routine to start up the 6Ø4-line processor (SVCT), and c) by the real-time scheduler to initiate a scheduled command.

2) The McIDAS executive always copies over a TRB before passing it to an initiator for command execution or back to a calling WTRR subroutine.

RELATED PROGRAM AND SUBPROGRAM

Name	Member	Function
SCANNE	MLSCANNE	McIDAS command scanner
WTRR	SRWTRR	write to terminal and wait for response

CALLING SEQUENCE

```
WTRR   SUBROUTINE WTRR(MSG,RSPONS)
          input:  MSG = array of message sent to terminal
          output: RSPONS = array of response
```

USER COMMON

Type: Temporary Memory
Size: Variable
Initialization: MLWAKEUP, MRB2MAIN, MRT2MAIN
Documented by: J. Benson

INTRODUCTION

The User Common is accessed by the subroutines LUC (look at) and PUC (put into).

UC variables reside in the system communications region (SYSCOM). They are not lost when your initiator aborts, but are lost when McIDAS aborts. UC variables with negative subscripts pertain to the initiator and reside in the initiator region of SYSCOM (this includes UC(Ø)); UC variables with positive subscripts pertain to your terminal and reside in the (separate) terminal region of SYSCOM. This division is transparent to any program that calls LUC or PUC.

Preceding the UC variables in SYSCOM is the global SYSCOM area that contains variables accessible to all initiators and at all terminals. These variables are inspected by function ND and are written by subroutine WD. ND(Ø) returns the first word of this area.

WORD ALLOCATION

Global SYSCOM Region:

Indices are those used to access SYSCOM via subprograms ND and WD.

Note that there are three places where global UC is initialized - MLWAKEUP (see SYSKEY, p. 3-44.1), MRB2MAIN, and MRT2MAIN.

Word	Contents
1-2	system version
3	YYDDD when WAKEUP was executed
4	HHMMSS when WAKEUP was executed
5	largest foreground terminal number defined in system (71)
6	maximum permitted area number (7999)
7	maximum permitted number of areas per volume (1ØØØ)
8	number of active area volumes (8)
9	offset from GMT
1Ø	default system printer number (at logon)
11	GOES ingestor flag word
12	active Kavouras radar boxes
13-299	(reserved)

UC:

Words	Contents
-1Ø1	pass/fail code from the scheduler. Ø implies safely scheduled.

The following words give the names of navigation coreloads which have been dynamically loaded into core. These are referenced by subroutine NVLOPE, which releases the core used by these coreloads at the conclusion of each SQX'ed module.

-94,-95	reserved for future navigation coreloads
-93	suffix of navigation-3 coreload (name='NV3' + suffix, e.g., 'GOES')
-92	suffix of navigation-2 coreload (name='NV2' + suffix)
-91	suffix of navigation-1 coreload (name='NV1' + suffix)
-9Ø	suffix of calibration-3 coreload (name='KB3' + suffix, e.g., 'VISR')
-89	suffix of calibration-2 coreload (name='KB2' + suffix)
-88	suffix of calibration-1 coreload (name='KB1' + suffix)

Data Tablet Group. These words contain the information returned from the terminal on the tablet state, as well as the tablet box number, name and position, as determined by the scanner.

-79	highest y coordinate of box (top)
-78	lowest y coordinate of box (bottom)
-77	highest x coordinate of box (right)
-76	lowest x coordinate of box (left)
-75	box stringname (A4 EBCDIC)
-74	box number (box 1 is at upper left)
-73	pen mode: 2 or 3 = pen down 1 = proximate Ø = pen up
-72	pen position - y coordinate (Ø,Ø) is bottom-left
-71	pen position - x coordinate

Motion Vector (WINDCO) UC words used (see also UC 98,1ØØ):

-62	joystick control value save
-61	mode control for joystick 2
-6Ø	mode control for joystick 1
-58	ending column number in current MD WIND file
-57	ending row number in current MD WIND file
-56	starting row number in current MD WIND file
-55	current MD WIND file used for core output
-54	current MD WIND file used IMV, selector and core input
-53	element size of target cursor (TV coord)
-52	line size of target cursor (TV coord)
-51	space bar toggle
-5Ø	error status word

Words	Contents
-44	2nd cursor state control word bits Ø-2 not used bits 3-5 are cursor color (DT command) bits 6-8 are cursor type (DT command)
-43	2nd cursor element position
-42	2nd cursor line position
-41	2nd cursor element size
-4Ø	2nd cursor line size
-38	next record pointer for DEV=F file
-37 - -35	LW filename for DEV=F
-34	current graphics virtual frame (if <0, don't plot on WRRRM)
-33	current DMES device: 1 = CRT 2 = local printer 3 = system printer Ø = black hole
-32	current EMES device: 1 = CRT 2 = local printer 3 = system printer Ø = black hole
-31	current LTQ device: 1 = CRT 2 = local printer 3 = system printer Ø = black hole
-3Ø	auto-context-table-search. When 1, parameter fetching subroutines (e.g., CKWP) resort to system string table for missing keyword parameters.
-29	(reserved for future use by system subroutine SQX)
-28	name (characters 5-8) of program currently running
-27	name (first 4 characters) of program currently running
-26	type of program currently running: 1 = macro Ø = non-macro
-25	type of program currently running: 1 = foreground Ø = background -1 = background with FORTRAN main, can do FORTRAN I/O 2 = console started task
-24	current SQX level (scanner is Ø, next program is 1, etc.)
-23	number of this initiator
-22	command was started by scheduler (=1), not by scheduler (=Ø)
-18 - -21	(reserved)
-17	user initials under which this command runs; may be different from logged-on initials in UC(2) if command is scheduled
-16	project number under which this command runs; may be different from logged-on project number in UC(1)
-1 - -13	snapshot of the terminal state taken just before the command begins. This is in a format close to the terminal protocol.

Words	Contents
-13	cursor state control word bits Ø-2 control joystick assignments: Ø = cursor is host-controlled 1 = j1 (joystick 1) is position vernier and j2 is coarse position 2 = j1 is cursor size and j2 is disconnected 3 = j1 is cursor size and j2 is coarse position 4 = j1 is velocity and j2 is disconnected 5 = j1 is velocity and j2 is coarse position 6 = dual cursor mode; host controlled 7 = dual cursor mode; j1 is vernier position of 2nd cursor, while j2 is coarse position of both cursors bits 3-5 are cursor color (DT command) bits 6-8 are cursor type (DT command)
-12	cursor element position
-11	cursor line position
-1Ø	cursor element size
-9	cursor line size
-8	graphics state control word bit Ø = 1 means graphics connected to loop control (J key) bit 1 = 1 means graphics frame is looping (L key) bit 2 = Ø means graphics frame is blanked (W key)
-7	graphics lower bound
-6	graphics upper bound
-5	current graphics frame
-4	frame state control word bit Ø = 1 means frame connected to loop control (Y key) bit 1 = 1 means frame is looping (L key) bit 2 = Ø means frame is blanked (K key)
-3	image frames lower bound
-2	image frames upper bound
-1	current image frame
Ø	user's terminal number
1	project number under which current user is logged on; may be different from UC(-16)
2	user's initials
4	current navigation file number
5	current MD file number
6	current grid file number
9	second printer number for this user; system printer = Ø
1Ø	first printer number, i.e., local printer for this user; system printer = Ø
11	number of lines on TV screen
12	number of elements on TV screen
13	number of image frames
14	number of graphics frames
15	terminal is remote (=1), local (=Ø)
16	terminal is video (=1), nonvideo (=Ø)
17	flag for E key: Ø = lat/lon are displayed in DDDMMSS 1 = lat/lon are displayed in decimal
18	SVCT flag on the terminal wanting 6Ø4 over comm

Words	Contents
19	McIDAS disk file I/O trace: 1 = do trace anything else = do not do trace
2Ø	set to 1 by G key
21	set to 1 by Q key
22	import pointer for others
23-24	terminal identifier (name or location) - 8 characters
25	name of CRT component of terminal - 4 characters: 'TLVD' = Televideo '3278' = IBM 3278 'TEK4' = Tektronics 'CRT ' = others
26	data tablet horizontal extent (max x coordinate + 1)
27	data tablet vertical extent (max y coordinate + 1)
28	non zero if terminal down
29	non-zero if may quit areas even if not the owning project (operator's terminal only)
38	1 = draw graphics on WRRRM Ø = do not draw graphics on WRRRM
39	Ø = do not write virtual graphics non zero = virtual frame to write
4Ø	3-D graphics menu pointer
41	planetary programs navigation pointer for current area
42	PLTPAK right/left frame inversion flag for two screen stereo
43	3-D graphics and planetary programs status file ID
44	reserved for 3-D graphics and planetary programs
45	3-D graphics message waiting flag
46	default graphics line width
47	graphics dash pattern - dash length, in pixels
48	graphics dash pattern - gap length, in pixels
49	graphics dash pattern - gap color
5Ø-79	reserved for keeping host's copy of the terminal state. This is in a format easy to manipulate and related to the functions of the terminal as an abstract device. Subroutine STATEU composes this state into a valid terminal order (protocol resembles snapshot in UC -1 through -13).

Words	Contents
5Ø	loop control: 1 = system is looping Ø = system is not looping

Loop control system consists of LB command and A, B, J, Y, O and L keys. Images and graphics are independently connected to and disconnected from the loop system via UC 54 and 59.

51	current image frame
52	image frame loop - upper bound
53	image frame loop - lower bound
54	1 = image frames connected to loop control Ø = images frames not connected

Words	Contents
55	1 = image frames visible Ø = image frames blanked
56	current graphics frame
57	graphics frame loop - upper bound
58	graphics frame loop - lower bound
59	1 = graphics frames connected to loop control Ø = graphics frames not connected
6Ø	1 = graphics frames visible Ø = graphics frames blanked
61	cursor size - vertical
62	cursor size - horizontal
63	cursor position - line number
64	cursor position - element number
65	cursor type: 1 = box 2 = crosshair 3 = box and crosshair 4 = solid box 5 = Star Wars
66	cursor color, Ø through 7
67	joystick 1: 0 = disconnected 1 = controls cursor position 3 = controls cursor size 4 = velocity cursor
68	joystick 2, same values as joystick 1 Currently the only legal states for joystick1/ joystick2 are: Ø/Ø , 2/1 , 3/1 , 3/Ø , 4/1 , 4/Ø in single cursor mode Ø/Ø , 2/1 in dual cursor mode
69	2nd cursor size - vertical
7Ø	2nd cursor size - horizontal
71	2nd cursor position - line number
72	2nd cursor position - element number
73	2nd cursor type; as for primary cursor, word 65
74	2nd cursor color; as for primary cursor, word 66
75	cursor mode switch: Ø = single cursor mode 1 = dual cursor mode
76-79	(reserved for terminal state expansion)

8Ø	current TOVS orbit/sounder/retrieval file number
81	current VAS area/sounder file number
83	terminal type, for PC at logon
84-85	software release version for PC AT
88	McBASI auto-line-number line number
89	McBASI run abort flag

Data Tablet related setup options:

91	(reserved)
92	upper-left line coordinate of current box
93	upper-left element coordinate of current box
94	lower-right line coordinate of current box
95	lower-right element coordinate of current box

Words	Contents
96	box name (ASCII 4-char)
97	box number
98	current MD core output file for motion vectors (WINDCO)
99	nortle toggle (line interlace)
1ØØ	current MD selector file for motion vectors (WINDCO)
1Ø1-12Ø	copy of raw text from current command or last command entered from keyboard
121	single letter command (=non-zero), otherwise (=Ø)
122	used by scheduler (SKED) - last time through all entries flag

RELATED SUBPROGRAMS

Name	Member	Function
LUC	MRLUC	look at User Common
NVLOPE	MRNVLOPE	entry point of every McIDAS load module
PUC	MRLUC	poke word into User Common

CALLING SEQUENCES

LUC

```
FUNCTION LUC(INDEX)
LUC looks at User Common (peek).
  input:  INDEX = where, in User Common, to do transfer
  output:  function value = word extracted from User Common
```

NVLOPE

```
SUBROUTINE NVLOPE(INAME, ITVEC, JPAR, JTOK)
NVLOPE is the entry point of every McIDAS load module.
  input: INAME = program name
         ITVEC = transfer vector; provides addresses for
                 access to routines in scanner and initiator
          JPAR = array containing parameters to be passed to
                 the load module; for ISQXed load modules,
                 contains an analysis of sequence.
          JTOK = array containing parameters to be passed to
                 the load module.  For ISQXed load modules, JTOK
                 is a sequence of parameter tokens.
```

PUC

```
FUNCTION PUC(FROM,INDEX)
PUC pokes word into User Common.
  input:  FROM = full word data value to transfer to User Common
         INDEX = where, in User Common, to do transfer
```

VIRTUAL GRAPHICS STORAGE FILES

(blank holds terminal number)
Type: LW files
Size: Variable
Initialization: Automatic
Documented by: T. Whittaker

WORD ALLOCATION

Words 1-512 — Treated as 1Ø24 integer*2 words, one for each virtual graphic. A negative value indicates that this virtual graphic has not been written. A positive value indicates the graphic is in use and, further, the value points to the initial entry in the Space Allocation Table (SAT).

Words 513-2562 — Treated as 4Ø96 integer*2 words for the Space Allocation Table (SAT). A negative value means the associated space is available. A positive value means the space is in use. Bits 11 through Ø point to the next link used by the current frame. If this field is zero the space is in use but there are no more links in the frame. If bit 12 <> Ø the space begins data for the next consecutive scene in the current frame.

Words 2563-... — Treated as blocks of 4 x 512 integer*2 words which are pointed to by the SAT. These words contain the actual plotting information for the associated graphic. Each group of 4 words is the data for one graphic point: PEN, X, Y, and Z coordinates respectively. Only the least significant 8 bits of the "pen" should be used. The most significant 8 bits are used as flag bits. Bit 8 <> Ø means this is the end of this scene. Bit 9 <> Ø means that dash mode is on.

RELATED SUBPROGRAM

Name	Member	Function
VIRTUA	MRVIRTUA	handle virtual graphics data filing
SVG	MLSVG	replot virtual graphics, frames or scenes
SVS	MLSVS	replot virtual graphics

CALLING SEQUENCE

```
VIRTUA  SUBROUTINE VIRTUA (PEN,X,Y,Z,TYPE)
          input:  PEN   = pen value for plotting
                  X,Y,Z = coordinates
                  TYPE  = Ø for initialization
                         -1 for erase
                          2 for endplt bindoff
                          3 for normal plotting
```

STATE AND TOWN NAMES

Type: LW file
Size: 1Ø word records
Initialization: Restructured from tape
Documented by: J. Benson

WORD ALLOCATION

This is the LW file containing the primary "SELS" database. It consists of 1Ø-word records, sorted by state and town name.

Within each record, the data is as follows:

Words	Contents
Ø-5	alpha Town Name
6	1ØØØ*(State-Number) + (County-Number)
7	pointer to next town in box (-1 IMP END)
8	latitude
9	longitude

The "box" to which word 7 refers is a 25 x 64 grid of 1° squares superimposed on the US.

RELATED PROGRAM

Name	Member	Function
WHODER	MLWHODER	list 2Ø towns closest to cursor position

FILE CONTAINING COUNTY AND STATE NAMES

Type: LW file
Size: 16ØØ word records
Initialization: Restructured from tape
Documented by: J. Benson

WORD ALLOCATION

This LW file is auxiliary data used in conjunction with WHODERØ1. The first 16ØØ words of the file are the string heads for the 16ØØ boxes (25 x 64).

The rest of the file is 5-word records containing a county and state name. The index to this data is 5* (Word6 of WHODERØ1 record). Since the first state-county index is 1ØØ1, the first of these records is found at LW Word 5ØØ5.

RELATED PROGRAM

Name	Member	Function
WHODER	MLWHODER	list 2Ø towns closest to cursor position

TEXT SAVED FROM THE FAA6Ø4 LINE

Type: LW file
Size: 4291Ø24 words
Initialization: Created by background program in 'MLMAKWXT'
Documented by: G. Dengel

WORD ALLOCATION

Text is EBCDIC, 4 characters per word.

Directory - words Ø-1Ø23

Words	Contents
Ø	'ON' or 'OFF'
1	file size in words
2-31	reserved
32-1Ø23	up to 32 directory entries (31 words each)

Words	Contents
1	data type
2	starting word for text
3	ending word for text
4	time cutoff in hours
5	pointer to first word of the last block written
6	search grid cutoff
7	latest sequence number
8-31	catalog numbers

1Ø24-415743 8Ø-character text records

Words	Contents
1Ø24-11Ø1Ø23	11ØØK records for SAs
11Ø1Ø24-13Ø1Ø23	2ØØK records for FTs
13Ø1Ø24-17Ø1Ø23	ØØØK records for TAFs
17Ø1Ø24-2ØØ1Ø23	ØØØK records for SDs
2ØØ1Ø24-2Ø51Ø23	5ØK records for WWs
2Ø51Ø24-2Ø91Ø23	4ØK records for FDs
2Ø91Ø24-2331Ø23	24ØK records for UAs
2331Ø24-2371Ø23	4ØK records for ABs
2371Ø24-2971Ø23	6ØØK records for SMs
2971Ø24-3171Ø23	2ØØK records for FOs
3171Ø24-3221Ø23	5ØK records for FEs
3221Ø24-3461Ø23	24ØK records for FPs
3461Ø24-3661Ø23	2ØØK records for NOs
3661Ø24-3861Ø23	2ØØK records for WAs
3861Ø24-4161Ø23	3ØØK records for UJs
4161Ø24-4241Ø23	8ØK records for MISCs
4241Ø24-4291Ø23	5ØK records for USERs

Each block of text is prefaced by a flag record in the following format:

Words	Contents
1	'$' indicates this is a flag record
2	catalog number
3	sequence number (wraps around at 999999)
4	number of text records in this block
5	day text was filed, YYDDD
6	time text was filed, HHMMSS
7-2Ø	blank

RELATED PROGRAMS AND SUBPROGRAMS

Name	Member	Function
ROOT	MLROOT	create, maintain and list FAA6Ø4 traffic routing table
SVCT	SLSVCT	FAA6Ø4 traffic handler
WXTRDI	MRWXTRDI	read WXTEXT directory entry
WXTWDI	MRWXTWDI	write WXTEXT directory entry
INTEXT	MRINTEXT	retrieve a record from WXTEXT

CALLING SEQUENCES

NOTE: Arguments beginning with 'C' are CHARACTER*12. All others are INTEGER.

WXTRDI FUNCTION WXTRDI(CFILE,TYPE,ICAT,DIRENT,ADDRES)

```
input:  CFILE  = weather text filename
        TYPE   = data type
               = -1 if ICAT is specified
        ICAT   = catalog number
               = -1 if TYPE is specified
output: DIRENT = 31-word array containing directory entry
        ADDRES = address in WXTEXT of directory entry
        function value =  Ø if successful
                       -1 if unsuccessful
```

WXTWDI FUNCTION WXTWDI(CFILE,ADDRES,DIRENT)

```
input:   CFILE  = weather text filename
         ADDRES = address in WXTEXT of directory entry
         DIRENT = 3Ø-word array containing directory entry
output:  function value = Ø if successful
                        -1 if unsuccessful
```

INTEXT FUNCTION INTEXT(ITYPE,ICAT,BUF)

```
input:   ITYPE= data type
         ICAT = catalog number
output:  BUF  = 2Ø-word array containing one record
         function value = 1 if BUF contains a flag record
                          Ø if BUF contains a text record
                         -1 if EOF detected
```

TAPE FILES

TAPE BLOCK DESCRIPTIONS

An important feature of the SSEC tape formats is the division of the information into fixed-length physical blocks, with logical records of unrelated length crossing block boundaries. Tapes formats are designed to be readily transportable to a wide variety of machines, and particularly easily to machines with 32-bit words.

The format is characterized by fixed-length tape blocks which are self-identifying; that is, each block contains words specifying the block type and the position of the block's data within the whole data set.

Type	Definition
'LABL'	Label block; must appear as the first block on the tape, and may appear nowhere else; 'LABL' identifies the tape as an SSEC save tape. DESC1 is the number of 8Ø-byte EBCDIC text images contained in IDATA. The number is between Ø and 96 (therefore NDATA is 8Ø*DESC1.) As of April 1984, one card image is present in IDATA, and it says: 'UW/SSEC MCIDAS DATASET UTILITY V3.ØØ 4/84'.
'COM '	Comment block; a block of EBCDIC card images of arbitrary content. These are intended to provide further documentation about the data on the tape. 'COM ' blocks may occur anywhere on the tape, but if they occur between a 'HDR ' block and an 'END ' block, they provide information about the particular image in which the block is embedded. DESC1 is a number between Ø and 96 defining the number of 2Ø-word comments contained in IDATA (therefore NDATA=2Ø*DESC1)
'HDR '	Header block; first block of a satellite image. DESC1 says 'AREA' (4-byte EBCDIC). DESC2 gives the origin of the image within the McIDAS system; it is the number (1 to 9999) or the digital area that was transferred to tape.

NOTE: 'DIR ', 'NAV ', 'CHEB', AND 'AUX ' blocks, when present, may occur in any order, but always appear after the 'HDR ' block and before the first 'WORD' block.

'DIR '	Directory block; the block containing the 64-word (256-byte) area directory entry pertaining to the image saved on tape. Its internal structure is documented below. NDATA is 256.
'NAV '	Navigation block; a block of information which associates the data saved with the underlying geography of the earth. 'NAV ' blocks are 128 words (512 bytes) long. DESC1 is either Ø (the navigation was drawn from the system navigation files), or 1 (the navigation was manually produced, or was generated by a remap program, or originated on another SAVE tape, or was obtained in some manner other than from the system navigation files). In either case, the internal format of the block in IDATA is the same and is documented below.

'CHEB' Chebyshev block; a block of special navigation information for GOES satellite images, containing information from the GOES IR documentation with Chebyshev coefficients for the spacecraft orbit and attitude. While 'CHEB' blocks may be transcribed from the IR documentation to SAVE tapes, the information within them is not supported by SSEC. 'CHEB' blocks are 128 words (512 bytes) long. DESC1 is either Ø (the navigation was drawn from the system navigation files) or 1 (the navigation was manually produced, or was generated by a remap program, or originated on another SAVE tape, or was obtained in some manner other than from the system navigation files). In either case, the internal format of the block in IDATA is the same, and is documented elsewhere.

'AUX ' Auxiliary block; a block of (typically) descriptive data auxiliary to the given data structure. DESC1 defines the 'AUX ' block type. This may be integer or EBCDIC. The internal structure of the 128-word (512-byte) block in IDATA depends on this type and will be documented elsewhere.

'WORD' Word block; a block containing some of the data from the actual satellite image. DESC1 defines the position of the data in this block within the total digital area (see the section on internal format of digital areas). DESC1 gives the starting byte number of the data within the area, where Ø is the first byte. DESC2 and NDATA both give the number of bytes of data in this block. DESC1, DESC2, and NDATA are always multiples of 4; i.e., data is parcelled out into 'WORD' blocks in integral numbers of words. (For all 'WORD' blocks except the last, DESC2 and NDATA are 768Ø.)

'END' End block is the last block for this satellite image. The next block is a 'COM ', or an 'HDR ' for the next satellite image, or a tapemark indicating the end of the tape.

ORDER OF BLOCKS ON THE TAPE

All character fields (indicated within quotes) are EBCDIC.

Length (Bytes)	Description
77Ø4	'LABL' block
-	tapemark (always present)
77Ø4	Ø or more 'COM ' blocks
77Ø4	'HDR ' block for first image on tape
77Ø4	'DIR ' block for first image on tape
77Ø4	'NAV ' block for first image on tape
77Ø4	Ø or more possible 'CHEB' and 'AUX ' blocks
77Ø4	1 or more 'WORD' blocks containing image data
77Ø4	'END ' block marking end of first image
-	tapemark for first image

```
77Ø4          'HDR ' block for 2nd image (if present)
               ...
77Ø4          'END ' block for 2nd image (if present)
  -           tapemark for second image


               ...
77Ø4          'END ' block for last image (if more than 2)
  -           tapemark for last image


  -           tapemark signifying end-of-tape (always present)
```

NOTES:

1. 'COM ' blocks may appear anywhere after the 'LABL' block and before the terminating tapemark. They may be ignored.

2. The number of 'WORD' blocks for an image may be computed from the area directory entry ('DIR' block) by the following formula:

   NBLOCKS= ((PRESIZ + NELES*ELESIZ)*NLINES + 7679) / 768Ø

   The descriptor quantity NDATA is 768Ø for all blocks except the last in an image. The last NDATA is as follows:

   NDATA = MOD(((PRESIZ + NELES*ELESIZ)*NLINES) / 4 * 4, 768Ø)

   or NDATA=768Ø if the above formula produces Ø.

GARS TAPE FORMAT

Type: LW file
Size: Variable
Initialization: DATOLW
Documented by: R. Dengel

WORD ALLOCATION

GARS tapes contain no IBM standard volume or file labels. Instead they appear as one or more files (each terminated by a tapemark), each containing one or more satellite images. Unless a special user request is made, each file contains exactly one image. The end of the tape is signified by two tapemarks following the last file (instead of one).

The block size is 77Ø4 bytes. The first 24 bytes provide identifying information for the block. The remaining 768Ø bytes contain data or may be empty, depending on the block type. (768Ø is the least common multiple of 512, 64Ø, and 3. If the lines in the area are 512 bytes long or 64Ø bytes long (both common lengths), and if IR documentation is not present, then the lines do not cross block boundaries.)

The blocks all have the following internal structure:

Bytes	Contents	Description
Ø-3	TYPE	block type (EBCDIC)
4-7	DESC1	. further descriptive information
8-11	DESC2	. (depends on the block type)
12-15	DESC3	.
16-19	DESC4	.
2Ø-23	NDATA	number of bytes of data in the block values Ø to 768Ø
24-77Ø4	DATA	data (if NDATA is not Ø)

Each satellite image begins with block type 'HDR ' and ends with block type 'END '. All blocks between the 'HDR ' and the 'END ' pertain to the satellite image being saved. 'DIR ', 'AUX ', 'NAV ', 'CHEB', and 'COM ' blocks supply descriptive data associated with the image; the image is stored in one or more 'WORD' blocks.

DESC1...NDATA are all 4-byte fields that should be interpreted as two's complement binary integers, unless otherwise stated. When they are not explicitly mentioned in a block description, they are binary Ø.

RELATED PROGRAMS

Name	Member	Function
PUTGAR	SLPUTGAR	GARS McIDAS tape save module
GETGAR	SLGETGAR	GARS McIDAS tape restore module

PUT TAPE FORMAT

Type: LW file
Size: Variable
Initialization: DATOLW
Documented by: R. Dengel

WORD ALLOCATION

PUT tapes contain no IBM standard volume or file labels. Instead they appear as one file (terminated by two tapemarks). The block size is 412Ø bytes. The first 24 bytes provide identifying information for the block. The remaining 4Ø96 bytes contain data or may be empty, depending on the block type. The blocks all have the following internal structure:

Bytes	Contents	Description
Ø-3	TYPE	block type (EBCDIC)
4-7	DESC1	. further descriptive information
8-11	DESC2	. (depends on the block type)
12-15	DESC3	.
16-19	DESC4	.
2Ø-23	NDATA	number of bytes of data in the block values Ø to 768Ø
24-412Ø	DATA	data (if NDATA is not Ø)

Each satellite image begins with block type 'HDR ' and ends with block type 'END '. All blocks between the 'HDR ' and the 'END ' pertain to the satellite image being saved. 'DIR ', 'AUX ', 'NAV ', 'CHEB', and 'COM ' blocks supply descriptive data associated with the image; the image is stored in one or more 'WORD' blocks.

DESC1...NDATA are all 4-byte fields which should be interpreted as two's complement binary integers, unless otherwise stated. When they are not explicitly mentioned in a block description, they are binary Ø.

RELATED PROGRAMS AND SUBPROGRAMS

Name	Member	Function
PUT	SLPUT	SSEC McIDAS tape save module
GET	SLGET	SSEC McIDAS tape restore module
PUTNAV	SRPUTNAV	write navigation codicils to "PUT" tape
PUTAUX	SRPUTAUX	write auxiliary codicils to "PUT" tape
PUTCOM	SRPUTCOM	write McIDAS documentation file to tape

CALLING SEQUENCES

PUTAUX SUBROUTINE PUTAUX(KIND,KEY1)

```
  input:  KIND   = type of auxiliary codicils
          KEY1   = primary key for codicil structure
```

PUTCOM SUBROUTINE PUTCOM(SET,CNAME)

```
  input:  SET    = character set used; 0=EBCDIC, 1=ASCII
          CNAME  = file name
```

PUTNAV FUNCTION PUTNAV(NAVTYP,KEY1,KEY2)

```
  input:  NAVTYP = navigation codicil type ("NAV" or "CHEB")
          KEY1   = primary key for codicil type
          KEY2   = secondary key for codicil type
```

LOGON FILES

LOGON PROCEDURES

The logon procedure is defined by nine steps executed in sequence. They are:

1. Check that terminal is not currently logged on.
2. Check for valid logon order (initials project number).
3. Check initials against valid initials in BLALOC LW file
4. Retrieve terminal configuration from CONFIG LW file.
5. Put terminal configuration data and user information into User Common.
6. Initialize terminal according to command.
7. Display the terminal configuration on one line at the top of the Output Device.
8. Display Message-of-the-Day, found in LOGON LW file.
9. Display pending mail messages, found in MAILBOX LW file.

If steps 1 through 3 are not successfully completed, the logon procedure is aborted and explanatory messages are given to the user. Following is the file structure for each of the four LW files.

VALID USER INITIALS FOR THE SYSTEM

Type: LW file
Size: Variable
Initialization: Entered through the editor
Documented by: A. Weickmann

WORD ALLOCATION

Each record is 8Ø characters in length.

Characters	Contents
1-4	user initials
5-8Ø	full user name and additional description

RELATED PROGRAMS

Name	Member	Function
MAIL	MLMAIL	McIDAS Mailbox
LOGON	MLLOGON	McIDAS Logon Procedure

TERMINAL INFORMATION FOR EACH TERMINAL

Type: LW file
Size: Variable
Initialization: background job SULOGON
Documented by: A. Weickmann

WORD ALLOCATION

Each record is 8Ø characters in length; some of the data is formatted as keywords.

Characters	Contents
1-4	T = Terminal number
5-12	physical terminal location
13-28	TYP= (VID,NVD) (LCL,REM) (3278,3178,NEITHER)
29	blank
If TYP is NVD:	
30-33	blank
34-36	node number
37-39	blank
4Ø-8Ø	associated user of the terminal
If TYP is 'VID':	
3Ø-41	NLIN= number of lines and elements on video
43-54	NFRM= number of image and graphics frames
If TYP is 'NVD REM':	
3Ø-38	blank
39-5Ø	line and port of remote terminal
If TYP is 'NVD' and T= 95, 96, 97, 98, or 99 (BKGO or Tape partition):	
5-12	blank

RELATED PROGRAMS

Name	Member	Function
Logon	MLLOGON	McIDAS logon procedure

THE MESSAGE-OF-THE-DAY

Type: LW file
Size: Variable
Initialization: McBASI editor
Documented by: D. Wade

This LW file is read by the LOGON process. It contains a "MESSAGE OF THE DAY" that is copied out to the terminal at LOGON time.

WORD ALLOCATION

Each record is 8Ø characters in length.

Characters	Content
1 - 8Ø	message text

GOES ARCHIVE RETRIEVAL FILES

GOES CATALOG FILE

Type: LW file
Size: Variable
Initialization: GDOC_INIT and GSCD_INIT
Documented by: W. Hibbard

WORD ALLOCATION

The structure of the GOESCAT file is defined in terms of constants and arrays found in the ODCATDEC element. This element should be included in the declarations of any routine which will access the GOESCAT file. The GOESCAT file is divided into sections for directory, documentation and catalog elements. Because the catalog can hold 3ØØ elements of 1ØØ,ØØØ words each while an LW file can only hold a few more than 8,ØØØ,ØØØ words, GOESCAT is extended by files GOESCAT1, GOESCAT2 and GOESCAT3.

Directory:

Words	Contents
Ø-299	satellite number of element for 3ØØ elements
3ØØ-599	year of element for 3ØØ elements
6ØØ-899	actual length of element in words for 3ØØ elements

Documentation:

Words	Contents
9ØØ-1799	12 byte names for 3ØØ activities
18ØØ-2699	12 byte class names for 3ØØ activities
27ØØ-2999	indices of formats for 3ØØ activities
3ØØØ-3899	12 byte names for 3ØØ comments
39ØØ-4799	12 byte class names for 3ØØ comments
48ØØ-5Ø99	indices of formats for 3ØØ comments
51ØØ-5399	12 byte names for 1ØØ formats
54ØØ-9899	12 byte names for 15 fields for each of 1ØØ formats
99ØØ-11399	lengths for 15 fields for each of 1ØØ formats
114ØØ-12899	repeats for 15 fields for each of 1ØØ formats
129ØØ-3Ø899	24Ø byte explanations for 3ØØ comments

Elements:

1ØØ,ØØØ words are allocated for each element, although the actual number of words of data in each element is given in the directory. The file name and starting location of element number IELM is given by the following table.

IELM Range	File Name	Starting Word Number
1 to 8Ø	GOESCAT	3Ø9ØØ+1ØØØØØ*(IELM-1)
81 to 16Ø	GOESCAT1	1ØØØØØ*(IELM-81)
161 to 24Ø	GOESCAT2	1ØØØØØ*(IELM-161)
241 to 3ØØ	GOESCAT3	1ØØØØØ*(IELM-241)

Within each element the data are organized sequentially with pointers for quick access inside the element. Data are inserted in an element by moving part of the element up to create a gap, then inserting the new data in the gap, and adjusting pointers for the part of the element which moved. Many pointers are relative and need no adjustment if the head and tail of the pointer are on the same side of the insert point. Data are deleted by the inverse operation of an insert. Some structures in an element may be repeated an arbitrary number of times, with the first word of the structure being a relative pointer to the next structure. The end of such a chain is marked by a zero word where the next pointer would be expected.

Most of the data items of an element are one word long. The exceptions are the schedule names, which are 12 bytes long (3 words), and the data in fields of activities and comments, whose lengths (in bytes) and repeat factors are given in the documentation section. The field data of an activity or comment are padded with extra bytes so that the activity or comment ends on a word boundary.

The structure of an element is specified below as a sequence of data items, some of which are grouped into repeating structures. Structures may also be nested. All time fields are in minutes since midnight.

```
satellite number (1 word)
year (1 word)
first day - always equal to 1 (1 word)
number of words in day array - always 365 or 366 (1 word)
day array
   absolute pointer to day structure (1 word)
end of day array
schedules
   relative pointer to next schedule or zero (1 word)
   schedule name - the same as scan mode (3 words=12 characters)
   number of word pairs in the image array (1 word)
   image array - ordered by image start time
      absolute pointer to activity (1 word)
      image start time (1 word)
   end of image array
   activities
      relative pointer to next activity or zero (1 word)
      index to activity name in documentation section (1 word)
      activity number (1 word)
      field data (length depends on activity's format)
   end of activities
end of schedules
days - ordered by day
   intervals - ordered by time range low with no overlaps in time
      relative pointer to next interval or zero (1 word)
      absolute pointer to schedule (1 word)
```

```
      time range low (1 word)
      time range high (1 word)
      comments - ordered by time range low
         relative pointer to next comment or zero (1 word)
         index to comment name in documentation section (1 word)
         time range low (1 word)
         time range high (1 word)
         field data (length depends on comment's format)
      end of comments
   end of intervals
   tapes - ordered by start time with no overlaps in time
      relative pointer to next tape or zero (1 word)
      tape ID (1 word)
      start time (1 word)
      stop time (1 word)
      mount operator's ID (1 word=4 characters)
      dismount operator's ID (1 word=4 charcaters)
   end of tapes
end of days
```

RELATED PROGRAMS AND SUBPROGRAMS

Name	Member	Function
GASK	MLGASK	search the GOES catalog
GCAT	MLGCAT	make GOES catalog entries
GDOC	MLGDOC	set up documentation section of GOES catalog
GSCD	MLGSCD	set up schedules for satellite-year element
ACTFLD	MRRDELM	get values for field of current activity
BUMP	MRRDELM	create a gap in a catalog element
CCOM	MRRDELM	insert comment into catalog file
CHKIMG	MRRDELM	check if image with overlap of line range exists
COMFLD	MRRDELM	get values for field of current comment
NXTIMG	MRRDELM	get time of next image in current element
PUTFLD	MRRDELM	insert values for field in current comment
RDELM	MRRDELM	read element of GOESCAT
SETACT	MRRDELM	get position of activity in current schedule
SETDAY	MRRDELM	get position of day in current element
SETIMG	MRRDELM	get position of image in current schedule
SETINT	MRRDELM	get position of interval in current day
SETSCD	MRRDELM	get position of schedule in current element
SETTAP	MRRDELM	get position of tape in current day
WRTELM	MRRDELM	write currently open element of GOESCAT

CALLING SEQUENCES

ACTFLD

```
FUNCTION ACTFLD(CNAME,VALS)
  Get values for field of current activity.
  input:  CNAME = field name
  output: VALS  = array of field values
```

```
BUMP    SUBROUTINE BUMP(J,N)
        Create a gap in a catalog element.
          input:  J = position of gap
                  N = size of gap

CCOM    FUNCTION CCOM(COMNA,TIMLOW,TIMHI)
        Insert comment into catalog file.
          input:  COMNA  = comment name
                  TIMLOW = comment low time
                  TIMHI  = comment high time

CHKIMG  FUNCTION CHKIMG(TIME,SM1,SM2,SM3,SM4,LLINE,HLINE,NL,LINA,LINB)
        Check if image with overlap of line range exists.
          input:  TIME  = time of image
                  SM1   = allowed schedule name for image
                  SM2   = allowed schedule name for image
                  SM3   = allowed schedule name for image
                  SM4   = allowed schedule name for image
                  LLINE = low line of image
                  HLINE = high line of image
          output: NL    = number of lines of overlap
                  LINE  = low line of overlap
                  LINB  = high line of overlap

COMFLD  FUNCTION COMFLD(CNAME,VALS)
        Get values for field of current comment.
          input:  CNAME = field name
          output: VALS  = array of field values

NXTIMG  FUNCTION NXTIMG(TIMLOW,TIMHI,SM1,SM2,SM3,SM4,LLINE,HLINE)
        Get time of next image in current element.
          input:  TIMLOW = start of time interval
                  TIMHI  = end of time interval
                  SM1    = allowed schedule name for image
                  SM2    = allowed schedule name for image
                  SM3    = allowed schedule name for image
                  SM4    = allowed schedule name for image
          input/output:  LLINE = low line of image
                         HLINE = high line of image
          output: function value = time of next image in given time
                                   interval with given schedule names, in
                                   current element

PUTFLD  FUNCTION PUTFLD(CNAME,VALS)
        Insert values for field in current comment.
          input:  CNAME = field name
          output: VALS  = array of field values

RDELM   FUNCTION RDELM(SAT,YEAR)
        Read element of GOESCAT.
          input:  SAT   = satellite number of element to read
                  YEAR  = year of element to read
```

SETACT FUNCTION SETACT(ACTNA,ACTNUM)
Get position of activity in current schedule.
input: ACTNA = activity name
ACTNUM = activity number
output: function value = position of activity of given name and number in current schedule

SETDAY FUNCTION SETDAY(DAY)
Get position of day in current element.
input: DAY = Julian day of year
output: function value = position of day at given day in current element

SETIMG FUNCTION SETIMG(TIME)
Get position of image in current schedule.
input: TIME = time of image
output: function value = position of image at given time in current schedule

SETINT FUNCTION SETINT(TIME)
Get position of interval in current day.
input: TIME = time of interval
output: function value = position of interval at given time in current day

SETSCD FUNCTION SETSCT(SCDNAM)
Get position of schedule in current element.
input: SCDNAM = schedule name
output: function value = position of schedule of given name in current element

SETTAP FUNCTION SETTAP(TIME)
Get position of tape in current day.
input: TIME = time of tape
output: function value = position of tape at given time in current day

WRTELM SUBROUTINE WRTELM
Write currently open element of GOESCAT.
no arguments

GOES CATALOG SATELLITE FILE

Type: LW file
Size: Variable
Initialization: SDOC_INIT
Documented by: W. Hibbard

WORD ALLOCATION

The structure of the GOESSATS file is defined in terms of constants and arrays found in the ODSATDEC element. This element should be included in the declarations of any routine which will access the GOESSATS file. The GOESSATS file is divided into sections for directory, documentation and satellite elements.

Directory:

Words	Contents
Ø-39	satellite number of element for 4Ø elements

Documentation:

Words	Contents
4Ø-639	12 byte names for 2ØØ activities
64Ø-1239	12 byte class names for 2ØØ activities
124Ø-1439	indices of formats for 2ØØ activities
144Ø-1589	12 byte names for 5Ø formats
159Ø-3839	12 byte names for 15 fields for each of 5Ø formats
384Ø-4589	lengths for 15 fields for each of 5Ø formats
459Ø-5339	repeats for 15 fields for each of 5Ø formats

Elements:

53ØØ words are allocated for each element. The starting location of element number IELM is given by 534Ø+53ØØ*(IELM-1).

An element consists of a set of three arrays giving satellite center longitude as a function of date, followed by a sequence of variable length activities. The field data of an activity are padded with extra bytes so that the activity ends on a word boundary. The structure of an element is specified below. Refer to the section of this manual on the GOESCAT file for more detailed information on element structure.

```
array of 1ØØ center longitudes
   center longitude (1 word)
end
array of 1ØØ start days
   start day (1 word)
end
array of 1ØØ end days
   end day (1 word)
end
activities - not ordered, no overlap in day for same activity name
   relative pointer to next activity or zero (1 word)
   index to activity name in documentation section (1 word)
   day range low (1 word)
   day range high (1 word)
   field data (length depends on activity's format)
end of activities
```

RELATED PROGRAMS AND SUBPROGRAMS

Name	Member	Function
SDOC	MLSDOC	set up documentation section of GOES catalog satellite file
GSAT	MLGSAT	make GOES catalog satellite file entries
GASK	MLGASK	search the GOES catalog
EZNAV	MRRDSAT	map from lat/lon to line/element
RDSAT	MRRDSAT	read element of GOESSATS file
WRTSAT	MRRDSAT	write element of GOESSATS file

CALLING SEQUENCES

EZNAV FUNCTION EZNAV(SAT,DAY,ALAT,ALON,SLIN,SELE)

Map from latitude and longitude to line and element.

```
  input:  SAT  = satellite number
          DAY  = day
          ALAT = latitude
          ALON = longitude
  output: SLIN = line number
          SELE = element number
```

RDSAT FUNCTION RDSAT(SATN)

Read element of GOESSATS file.

```
  input:  SATN = satellite number of element to read
```

WRTSAT SUBROUTINE WRTSAT

Write element of GOESSATS file.

no arguments

GOES CATALOG WEATHER FILE

Type: LW file
Size: Variable
Initialization: WDOC_INIT and GWX_INIT
Documented by: W. Hibbard

WORD ALLOCATION

The structure of the GOESWX file is defined in terms of constants and arrays found in the ODWXDEC element. This element should be included in the declarations of any routine which will access the GOESWX file. The GOESWX file is divided into sections for directory, documentation and weather elements.

Directory:

Words	Contents
Ø-39	year of element for 4Ø elements

Documentation:

Words	Contents
4Ø-639	12 byte names for 2ØØ activities
64Ø-1239	12 byte class names for 2ØØ activities
124Ø-1439	indices of formats for 2ØØ activities
144Ø-1589	12 byte names for 5Ø formats
159Ø-3839	12 byte names for 15 fields for each of 5Ø formats
384Ø-4589	lengths for 15 fields for each of 5Ø formats
459Ø-5339	repeats for 15 fields for each of 5Ø formats

Elements:

5ØØØ words are allocated for each element. The starting location of element number IELM is given by 534Ø+5ØØØ*(IELM-1).

An element consists of a sequence of variable length activities. The field data of an activity are padded with extra bytes so that the activity ends on a word boundary. The structure of an element is specified below. Refer to the section of this manual on the GOESCAT file for more detailed information on element structure.

```
activitites - not ordered, activity/event name pair unique
   relative pointer to next activity or zero (1 word)
   index to activity name in documentation section (1 word)
   event name (3 words=12 characters)
   day range low (1 word)
   day range high (1 word)
   field data (length depends on activity's format)
end of activities
```

RELATED PROGRAMS AND SUBPROGRAMS

Name	Member	Function
GASK	MLGASK	search the GOES catalog
GWX	MLGWX	make GOES catalog weather file entries
WDOC	MLWDOC	set up documentation section of GOES catalog weather file
RDWX	MRRDWX	read element of GOESWX for year
WRTWX	MRRDWX	write currently open element of GOESWX

CALLING SEQUENCES

RDWX FUNCTION RDWX(YEAR)
Read element of GOESWX for year.
input: YEAR = year of element to read

WRTWX SUBROUTINE WRTWX
Write currently open element of GOESWX.
no arguments

GOES CATALOG REQUEST FILE

Type: LW file
Size: Variable
Initialization: GREQ_INIT
Documented by: W. Hibbard

WORD ALLOCATION

The structure of the GOESREQS file is defined in terms of constants and arrays found in the ODREQDEC element. This element should be included in the declarations of any routine which will access the GOESREQS file. The GOESREQS file is divided into sections for status, directory and user requests.

Status:

Words	Contents
Ø-199	request number of open request, or -1 for 2ØØ terminals
2ØØ-399	order number of open request for 2ØØ terminals

Directory:

Words	Contents
4ØØ-1399	order number, or Ø for available, or -1 for open for 1ØØØ requests
14ØØ-2399	place in queue (<Ø for review, >Ø for process) for 1ØØØ requests
24ØØ-12399	4Ø byte requestor name for 1ØØØ requests

Requests:

1ØØØ words are allocated for each request. The starting location of request number IREQ is given by 124ØØ+1ØØØ*(IREQ-1).

A request consists of a header portion and a sequence of area specifications. In this context 'area' does not mean image file. An area specifies a satellite number, a sector of data, and a set of dates and times. The header portion of a request contains information about the requestor and the desired format of the data, while the areas specify what data is being requested.

Header:

Words	Contents
Ø	order number
1	project number
2	header status - 1's bit set if date present, 2's bit for name, 4's bit for address, 8's bit for phone number, and 16's bit for device
3	(reserved)

Words	Contents
4	current ingest area
5	current ingest date
6	current ingest time
7-26	current ingestor parameters
27	date request received
28	desired data delivery date
29-38	requestor name (4Ø characters)
39-68	address (12Ø characters)
69-78	phone number (4Ø characters)
79-81	output device (12 characters)
82	tape density
83-85	enhancement name (12 characters)
86	grid/no grid
87	dwell/dwell+step
88-147	comment (24Ø characters)

The header is followed by a sequence of variable length area structures, linked by relative pointers. The first 21 words of an area structure have a fixed format, followed by variable length date and time arrays.

Area:

Words	Contents
Ø	relative pointer to next area, or zero if end of areas, or <Ø if area open. If open 2's bit =Ø for dates present and 4's bit =Ø for times present.
1	satellite number
2	area format (4 characters)
3	center latitude, or center line, or north latitude
4	number of lines, or south latitude
5	center longitude, or center element, or west longitude
6	number of elements, or east longitude
7	visible resolution
8	IR resolution
9-11	select scan mode 1 (4 characters)
12-14	select scan mode 2 (4 characters)
15-17	select scan mode 3 (4 characters)
18-2Ø	select scan mode 4 (4 characters)
21	number of word pairs in date array (=NDATES)
22	day array date or date range low (1 word) Ø or date range high (1 word) end of day array
22+2*NDATES	number of word triples in time array (=NTIMES)
23+2*NDATES	time array (times in minutes since midnight) time or time range low (1 word) Ø or time range high (1 word) Ø or time step (1 word) end of time array

The next area starts at 23+2*NDATES+3*NTIMES.

RELATED PROGRAMS AND SUBPROGRAMS

GANA	MLGANA	analyze GOES archive data request against catalog
GING	MLGING	manage processing of archive data requests
GQUE	MLGQUE	manage queues of archive data requests
GREQ	MLGREQ	set up GOES archive data requests
RDREQ	MRRDREQ	read GOES archive request from GOESREQS
WRTREQ	MRRDREQ	write currently open archive request to GOESREQS

CALLING SEQUENCES

RDREQ FUNCTION RDREQ(ONUM)
Read GOES archive request from GOESREQS.
input: ONUM = order number

WRTREQ SUBROUTINE WRTREQ
Write currently open archive request to GOESREQS.
no arguments

GOES CATALOG MESSAGE FILE

Type: LW file
Size: Variable
Initialization: GMES_INIT
Documented by: W. Hibbard

WORD ALLOCATION

The structure of the GOESMES file is defined in terms of constants and arrays found in the ODMESDEC element. This element should be included in the declarations of any routine which will access the GOESMES file. The GOESMES file is divided into sections for directory and message elements.

Directory:

Words	Contents
Ø-99	satellite number of element for 1ØØ elements
1ØØ-199	date of element for 1ØØ elements

Elements:

38Ø1 words are allocated for each element. The starting location of element number IELM is given by 2ØØ+38Ø1*(IELM-1).

An element consists of 1 word which gives the index of the currently open message in the element, followed by 2ØØ messages of 19 words each. The starting location of message number IMES is given by 1+19*(IMES-1). The format of a message is given below.

Message:

Words	Contents
Ø	message types - the sum of some subset of the numbers 1, 2 and 5 to indicate the types of spooler messages combined in this GOES catalog message
1	satellite number
2	satellite date
3	satellite time
4	image mode: 1 for Mode A 2 for MSI 4 for DS
5	image start scan number
6	image start date
7	image start time
8	image end scan number
9	image end date
1Ø	image end time
11-18	PDL for MSI or DS mode

RELATED PROGRAMS AND SUBPROGRAMS

Name	Member	Function
GMES	MLGMES	manage GOES catalog message file
GMSG	MLGMSG	put ingest messages into GOES catalog message file
CATMES	MRCATMES	get message from spool, combine for same image and put in GOESMES
DAYCON	MRRDMES	convert YYDDD to days since January 1, 19Ø1
RDMES	MRRDMES	read element for satellite and day from GOESMES
TIMCON	MRRDMES	convert HHMMSS to seconds and HHMM to minutes
WRTMES	MRRDMES	write currently open element to GOESMES

CALLING SEQUENCES

CATMES SUBROUTINE CATMES
Get message from spool, combine for same image and put in GOESMES.
no arguments

DAYCON FUNCITON DAYCON(YYDDD)
Convert YYDDD to days since January 1, 19Ø1.
input: YYDDD = Julian day
output: function value = days since January 1, 19Ø1

RDMES FUNCTION RDMES(SAT,DAY)
Read element for satellite and day from GOESMES.
input: SAT = satellite
DAY = Julian day (YYDDD)

TIMCON FUNCTION TIMCON(HHMMSS)
Convert HHMMSS to seconds and HHMM to minutes.
input: HHMMSS = time
output: function value = number of seconds or minutes

WRTMES SUBROUTINE WRTMES
Write currently open element to GOESMES.
no arguments

GLOSSARY OF McIDAS TERMS

Terms	Definition
extent	For the purpose of holding the page tables to exactly one sector (1Ø24 words) in length, a file is divided into extents of maximum length 1Ø23*1Ø24=1Ø47552 words. A file can have up to eight extents to achieve its maximum size. It will usually have fewer, because an extent has physical existence only if it contains data. Each file extent has one associated page table and one entry in the filename tables.
garbage collector	A McIDAS program which runs periodically to gather up unused sectors so they can be allocated as pages to McIDAS files.
LW	Stands for 'Large Word (array)'. LW is used as a prefix to the names of all the basic level disk file routines.
McIDAS file	A named collection of data in McIDAS space. McIDAS files are viewed by the software as ordered arrays of up to 838Ø416 words (almost 32m bytes), numbered from Ø.
McIDAS space	A collection of sectors residing on one or more disk volumes. These sectors are treated abstractly as a single entity out of which are carved large numbers of McIDAS files. Sectors in McIDAS space have full word addresses. The leftmost byte gives the volume number (Ø-3), and the rightmost three bytes give the relative sector number (Ø-65535) within the volume.
MCTOC	A single-extent McIDAS file containing the disk Table of Contents. The first 2 pages are the 'disk free space table' and the next 11 pages are the 'filename tables'.
page	A unit of 1Ø24 words (4Ø96 bytes) within the virtual address space of a McIDAS file. A page can contain data, in which case it is mapped onto a sector in McIDAS space by means of the file's 'page tables', or it can be empty, in which case it does not reside anywhere on disk but appears to be full of missing data codes (Z8Ø8Ø8Ø8Ø).
page table	The mechanism by which virtual address space for files is achieved. Virtual address space is divided into 1Ø24 word pages. The status of each page is recorded in a page table. If null the page is empty. Otherwise the page table will contain the actual sector address of the data in the page. A file has one page table for each extent which contains data.

sector	A physical disk record in McIDAS space, 1Ø24 words (4Ø96 bytes) in length.
virtual address space	Another name for the ordered array of 838Ø416 words that constitutes a McIDAS file. It is called virtual because it is possible that at any given time not all of the words will contain data and occupy actual space on the disk. Instead, there can be large gaps which are empty (appear to be filled with the missing data code Z8Ø8Ø8Ø8Ø) and which do not take up disk space.