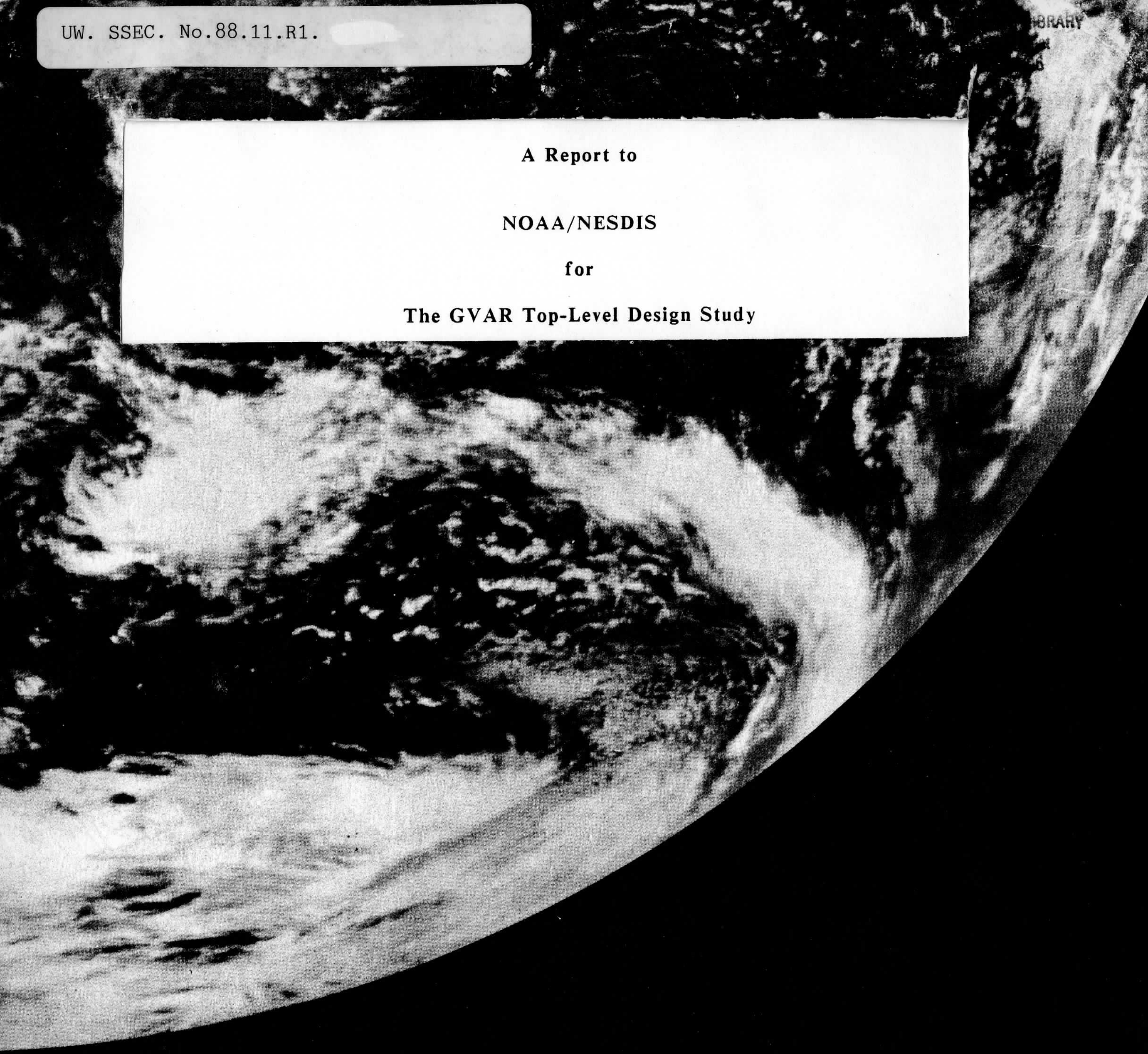


A Report to

NOAA/NESDIS

for

The GVAR Top-Level Design Study

A REPORT

from the space science and engineering center
the university of wisconsin-madison
madison, wisconsin

A Report to

NOAA/NESDIS

for

The GVAR Top-Level Design Study

Contract # 50-WCNE-8-06058

for the period of

1 February 1988 thru 31 October 1988

submitted by

Joseph P. Rueden

Space Science and Engineering Center
at the University of Wisconsin-Madison
1225 West Dayton Street
Madison, Wisconsin 53706
(608) 262-0544

November 1988

PREFACE

The purpose of this report is to document the findings of the GVAR Ingestor Top-Level Design Study. The proposed GVAR Ingestor design will be explained at a conceptual level and compared to the earlier SSEC AAA ingestor design to gauge its relative performance and functionality. Extensive use of graphics is made to provide clarity. Detailed supporting information is contained in the appendices.

ACKNOWLEDGMENTS

The technical contract monitor for this contract was Keith McKenzie (NOAA/NESDIS/OSD), and this study's existence was largely do to his efforts.

This study would not have been possible without the help of many people from SSEC, especially:

Bill Lagerroos - for extensive work on the AAA system performance,
Ted Barton - for design, consulting, and coordination,
Dave Jones - for the AAA hardware performance work,
Kevin Schlittler - for GVAR hardware design work,
John Benson - for design and consulting,
Paul Schnettler and Tony Wendricks - for the graphics work,
Jan Waite-Schuster - for the editing and formatting,

and Dan Adams, Ron Koch, Scott Mindock, Ric Suomi, Dee Wade in addition to all the others who participated in the GVAR ingestor design.

Also, thanks to Michael Goldberg of The MITRE Corporation for many good ideas, questions, and suggestions.

TABLE OF CONTENTS

List of Figures

List of Tables

1.0 INTRODUCTION

1.1 Purpose

The purpose of this report is to document the findings of the GVAR Ingestor Top-Level Design Study. The concepts, methodology, and assumptions of the design are described. Considerable effort has been expended to describe this design in such a way that intimate knowledge of the workings of McIDAS is not required.

The requirements for the GVAR ingestor have been taken primarily from the GVAR Ingestor Functional Description (reference 2). Reading the requirements sections will make this document more understandable. The GVAR Signal Format description (reference 3) was used extensively during this study, but is not really necessary reading if one understands the AAA format.

1.2 Assumptions

The basic assumptions upon which this study is founded are briefly described:

1.2.1 SSEC AAA-like Design

This assumption is specific in the study Statement-of-Work. The SSEC AAA design is a proven real-time design in use in several government and private facilities, both in the U.S. and abroad. It is designed to be used in an interactive, real-time system.

The AAA design philosophy has also been successfully used to build ingestors for other signal types; e.g., GMS, METEOSAT PDUS, and TIROS-N. This methodology, as evolved for the GVAR effort, is expected to be used for additional signal types as well.

1.2.2 Integrated with McIDAS

The AAA design is integrated with the McIDAS system. The two cannot be easily separated. This is because 'real-time' and 'interactive' are much more pervasive issues than just getting the data into the system. The data structures design, methods of initiating processes, and accessibility of the data by the users are inherent in the design of the entire system and provide the 'context' or 'environment' within which the ingestor is expected to function.

2.0 CONCEPTS OVERVIEW

The AAA design was based upon the earlier mode A and AA designs. In fact, the current AAA ingestor is also used to ingest A and AA signals from the SSEC GOES archive.

It is important to note that this design didn't just emerge full-blown but is based on 15 years of experience in dealing with satellite signals in a real-time way.

2.1 Design Goals

Before delving into the details, it is useful to describe the basic design goal and some basic ingestor concepts as viewed by SSEC.

2.1.1 <u>Real-time data access with low resource cost</u>

The goal of the AAA design was to provide image data in a real-time manner at as low a resource cost as was practical. Real-time manner here means in as close to real-time as possible, on a image line by image line basis. The users, primarily interactive forecasters, must be allowed access while the ingest is still occurring. Multiple users must be allowed access concurrently.

This goal led to the following ingestor consequences/concepts:

2.2 <u>Design Concepts</u>

2.2.1 <u>All byte and bit level functions are done in hardware</u>

System performance is improved most significantly by removing the need to manipulate each byte of data as it is received. This is accomplished by having the hardware format the data as it will appear in the database (at the image line level). The data is then moved from the ingestor to computer memory via DMA, and from computer memory to disc via DMA. Thus, the ingest handler task is a kind of 'traffic cop', merely directing the selection and flow of data. The approximate savings realized are documented in Barton, et al. (reference 1).

2.2.2 <u>Resolution reduction is done in hardware</u>

This follows from the first concept. Few applications require large quantities of the data at full resolution, or even all the data at *any* resolution. Reducing the resolution of the data before entering it into the database also reduces storage requirements, meshing nicely with the design goal.

The AAA design allows both sampling and pseudo-averaging (average elements, sample lines) for resolution reduction. These two methods were chosen as the best fit between what was economically possible, and what the scientist needed. Pseudo-averaged data performs as well as true averaged data for nearly all applications. However, for some applications sampling is more desirable than any form of averaging. Therefore, both pseudo-averaging and sampling were implemented.

Note that while the ingest hardware is responsible for the resolution reduction along each line, it is the software on the mainframe that is responsible for the reduction in the number of lines. This is easily handled because the software must request data from the hardware on a line-by-line basis anyway; it just omits requests for lines that should be skipped to make the resolution correct.

2.2.3 <u>Sectorization is done in the protocol</u>

Sectorization is the reduction of data volume by reducing geographic coverage. Since the software requests data from the hardware on a line-by-line basis, it is a simple matter to allow a byte offset and byte count for each line. This, coupled with the software only requesting needed lines, performs the sectorization in a simple and cost-effective way.

2.2.4 <u>Open database design</u>

McIDAS has to control the database at a low level in order to guarantee data structure integrity even while being written into. Reading while writing imposes new constraints on the database structure. Open database design here means to allow many to access data concurrently while still maintaining the integrity of the database.

It is necessary to have fairly simple structures, as the ingest handler must not hold locks if it is to maintain a high performance level. This basically eliminates any structures containing links or pointers.

A simple and direct I/O path must exist to the database. This is certainly true for the McIDAS AREA structure.

Note that while these constraints must apply to the image data structure, ancillary data structures do not necessarily have the same limitations. The LW structure in McIDAS is more complex, and has an I/O path with lower performance characteristics, than the AREA structure. The increased complexity of the structure is balanced by the ease with which McIDAS applications can be created. Overall system performance is not too adversely affected because imagery data is by far the highest volume.

2.2.5 <u>Navigation, calibration info needed at image start</u>

More than just the raw imagery data is needed in order to make the imagery useful to users and applications. Navigation (earth location for map drawing and collocation of other data types) and calibration (conversion of raw data values to radiance, temperature, or brightness) information is needed at the beginning of the image. This can be difficult when it may take minutes to receive the information from the signal itself.

McIDAS uses predictive navigation methods so a 'first guess' navigation is available until the transmitted navigation parameters can be filed. This temporary navigation is the information from the preceding image, and works because the spacecraft normally changes orbit and attitude slowly in a predictable manner.

Calibration information is usually available immediately. In the rare instances when it is not, again the calibration from the preceding image is used as a first guess. Calibration also normally changes slowly.

2.2.6 <u>'Automated' scheduling via signal tracking</u>

Automated scheduling here means requiring a low level of effort to cause data to be ingested continuously *even though the satellite schedule of images changes frequently and can not be predicted ahead of time.* This is more a practical limitation than a direct result of the primary goal.

This has been accomplished by the concept of signal tracking. Signal tracking is having a geography based set of requests, and letting the ingest handler match the incoming signal with the request set. The result should be image sectors placed into the database anytime the satellite sends them, without the operational impact of changing the schedule every time the satellite operational mode changes.

3.0 AAA INGESTOR DESIGN

3.1 Overview

The overall relationship of the various ingestor parts is illustrated in Figure 3.1. Note that this illustration shows both control and data flow.

3.2 Ingest Scheduler

The computer operator or operations manager is responsible for requesting what sectors of data will be ingested in what time interval. This is done by entering specific requests via the ingest schedule software. These requests are placed in a disc-based structure called the ingest schedule file. These requests are changed only as system loading or requirements change, typically a few times a year on an operational system. Figure 3.2 describes the relationship between the Ingest Scheduler and the Ingest Handler.

The first thing the requestor specifies is the satellite time window describing when the data is to be captured for this request. Both start day/time and stop day/time are specified. Within the window up to 15 entries (one entry for each sector) can be requested. Each entry can cause one sector to be created for each logical image sent, and will be repeated for each logical image sent within the window.

Entries are organized on a 'signal in, product out' basis; that is, for each discrete type of signal input (e.g., MSI or DS for modes AA or AAA) the requestor can select a particular product output (VIS (visible), IR (particular infrared band), MULT (collection of IR bands in a single image)). Because of this organization, and the satellite's multiple signal modes, it is unlikely that more than 10 sectors will be created in any one satellite transmission. In fact, the AAA ingestor has an internal limit of 10 simultaneous sectors.

Sectors can be requested either in earth coordinates (latitude/longitude), or directly in satellite coordinates (line/element). Scheduled processes convert earth based requests to satellite coordinates for the ingest handler at regular intervals, insuring that the most recent navigation parameters are being used for the transforms. The ingest handler never works from earth coordinates directly, primarily because navigation transforms are too slow to be run in the real-time critical path.

Space for the expected data is reserved at this time (in the area structure). This is necessary because the ingest handler does not have time to reserve space. Also, since the ingestor and user share the same database in McIDAS, it is necessary to reserve what space might be required at request time, rather than taking a chance on finding available space when the signal finally arrives.

3.3 Data Structures

Correct data structures design is critical to a successful ingestor design. The real-time requirement means that:

- data must be able to be inserted into the database instantly, with very low system overhead and,

- data must be accessible to applications *as it is being put into the database*, before the logical image has been completely received. This is necessary

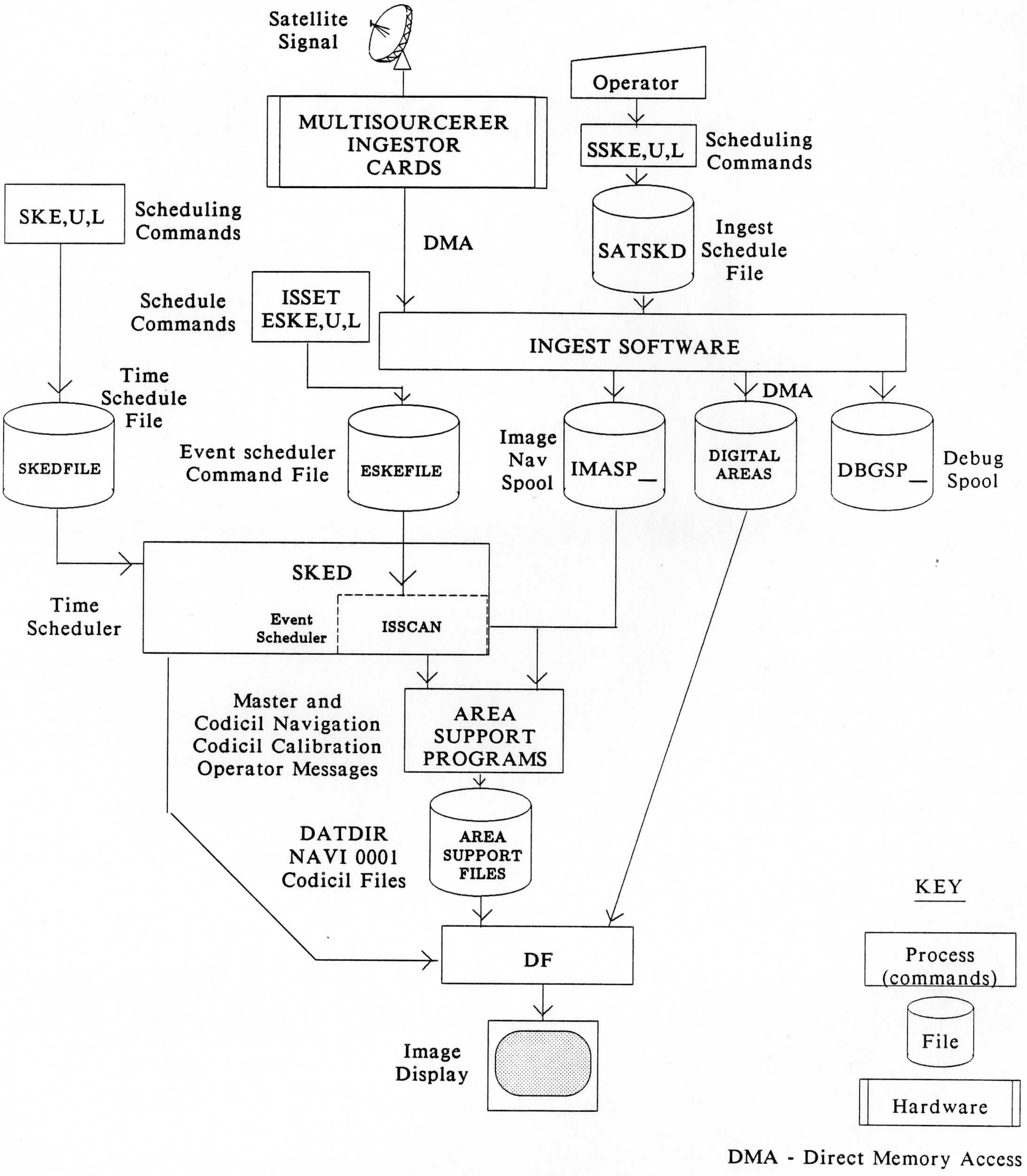


Figure 3.1: AAA Satellite Data Ingestion Path into the McIDAS Data Base

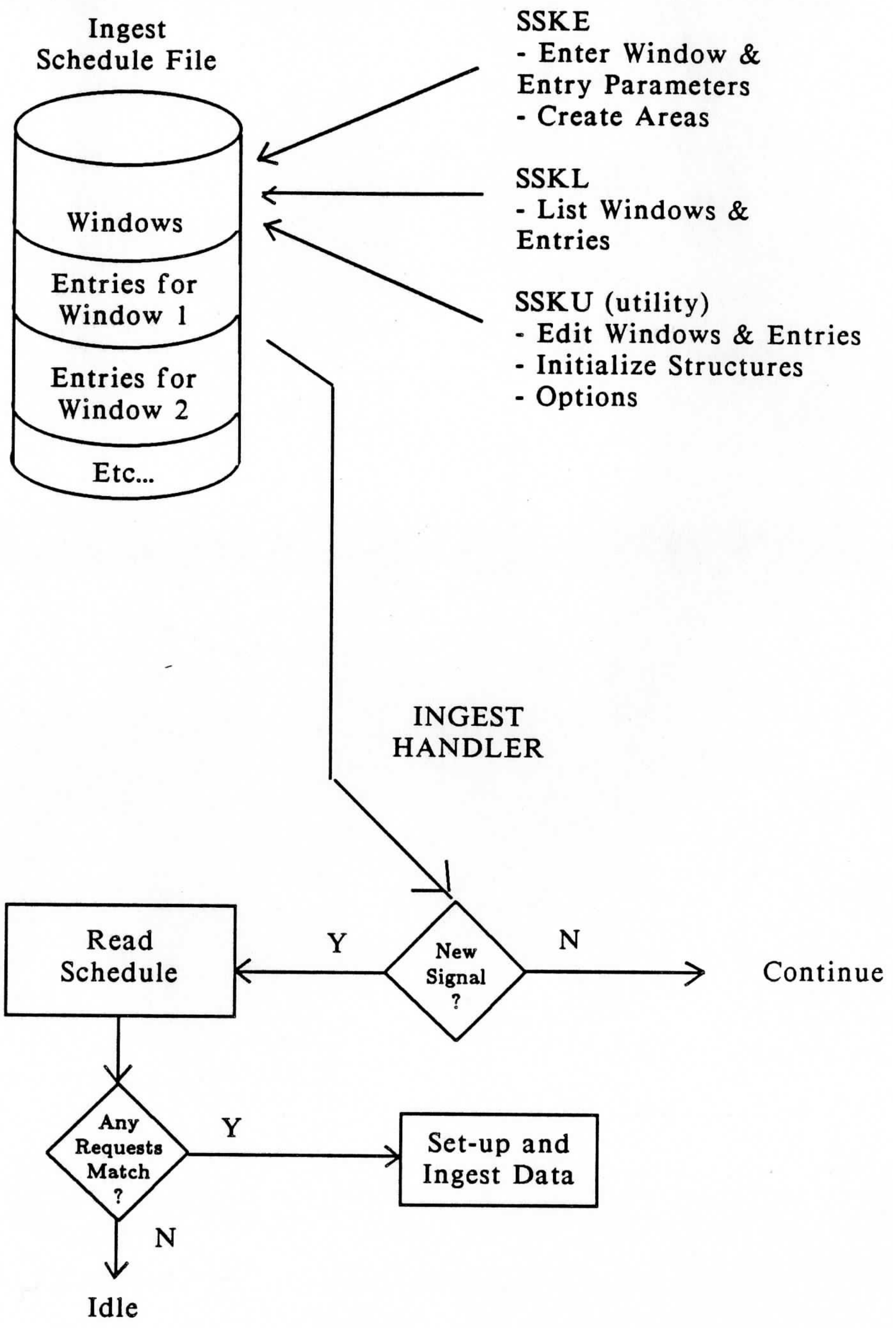


Figure 3.2: AAA Ingest Scheduler

because some applications require product delivery even before logical image completion (e.g., GOES-TAP creation, or image display to an interactive forecaster).

The interactive requirement means the interactive user must have a way to know the status of any image. In addition, operational environments require processes to be started automatically as the required portion of the data becomes available.

The following data structures are used to meet these requirements. For additional information, see The McIDAS Applications Programming Manual.

3.3.1 Areas

The area data structure contains image data. Either one band (as for visible imagery or single IR bands), or multiple bands (as for dwell sound IR imagery), can be contained in a single area. These structures are line oriented; that is, all data from a single geographic line is collocated. This organization meshes nicely with the way images are transmitted and normally accessed.

The data are stored on disc in large IBM files. Further organization is imposed on these IBM files by McIDAS. Data are accessed one logical track at a time, where a logical track is part of a physical track. Logical tracks are not allowed to cross physical track boundaries. This, plus giving the ingest handler the top priority of any application on the system, guarantees the best possible disc performance. The IBM access method is BDAM.

The logical organization of area files for AAA is described in Figure 3.3.1.

3.3.2 Area directory file

The area directory file contains the identification of the associated area data - the satellite sensor, day/time, bands, etc. Also, links to navigation and calibration information are kept here. A summary of the information contained is in Table 3.3.2.

3.3.3 Ingest schedule file

The ingest schedule file is an LW file where ingest requests are stored. The ingest handler picks them up from here and satisfies all that are possible, given the current signal day/time and type. A brief description of the contents is contained in Figure 3.3.3.

3.3.4 Image spool file

The image spool file is an LW file used by the ingest handler to communicate with the rest of the system. Blocks are written to this spool whenever a significant event occurs, or whenever certain data needed by the rest of the system is received. For example, Orbit and Attitude information from the AAA signal is collected and written to the spool. Also, the area directory entries are handled through the spool. The event scheduler is responsible for initiating processing of these blocks in a timely manner.

A summary of the kinds of blocks written to the Image Spool and their purposes are in Table 3.3.4.

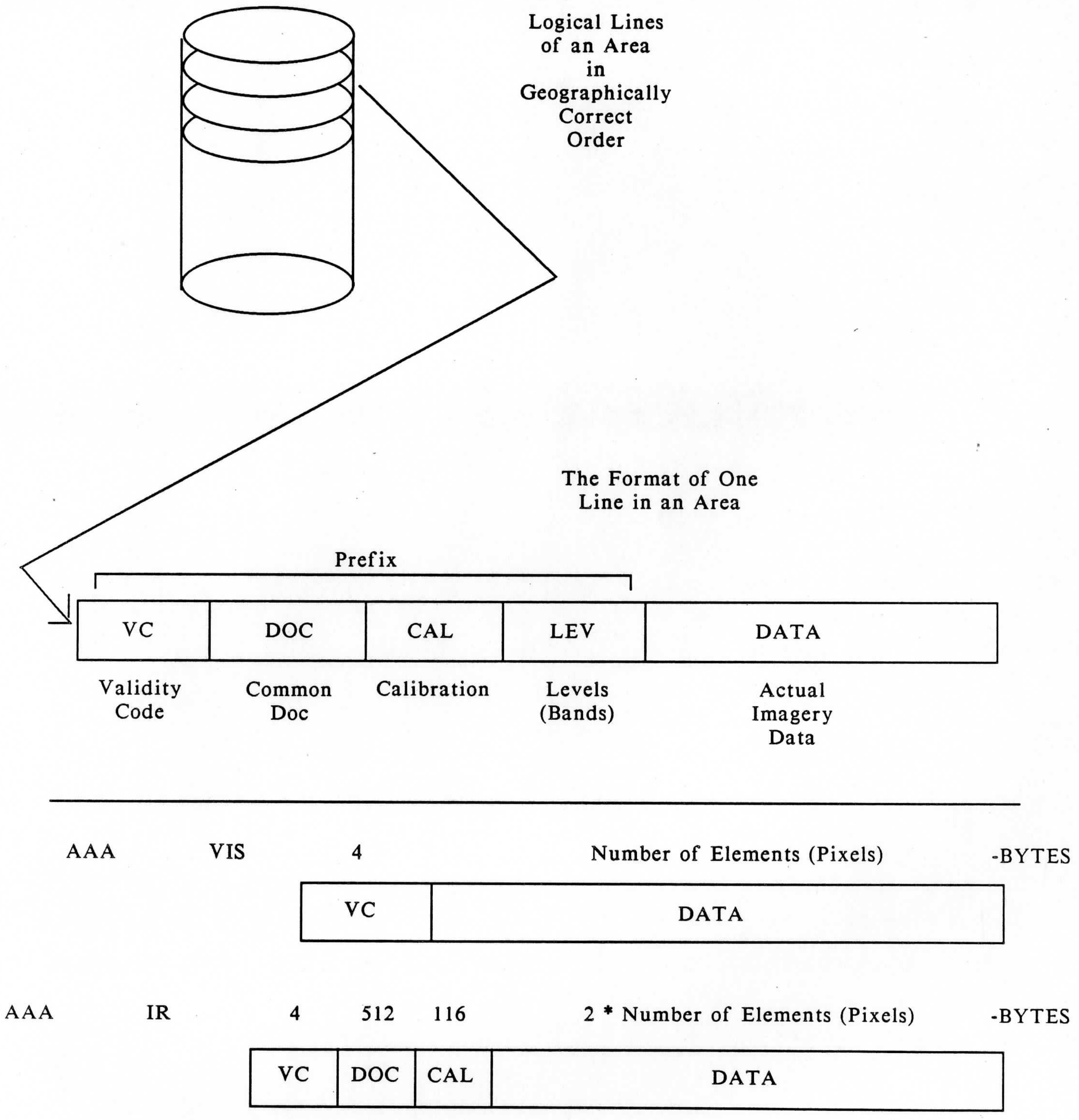

Logical Lines of an Area in Geographically Correct Order
The Format of One Line in an Area
Prefix
VC
DOC
CAL
LEV
DATA
Validity Code
Common Doc
Calibration
Levels (Bands)
Actual Imagery Data
AAA
VIS
4
Number of Elements (Pixels)
-BYTES
VC
DATA
AAA
IR
4
512
116
2 * Number of Elements (Pixels)
-BYTES
VC
DOC
CAL
DATA

Figure 3.3.1: AAA Area Format

Area identifiers:

- SSS	- Satellite identification number
- YYDDD	- Nominal year and Julian day
- HHMMSS	- Nominal time
- Y-COOR	- Upper left line number in satellite cooordinates
- X-COOR	- Upper left element number in satellite coordinates
- NLINES	- Number of lines
- NELES	- Number of elements
- NBANDS	- Number of bands (levels)
- NBYTES	- Number of bytes per pixel
- LRES	- Line resolution (sample or average down factor)
- ERES	- Element resolution (sample or average down factor)
- DOC	- Length of DOC section in bytes
- CAL	- Length of CAL section in bytes
- LEV	- Length of LEV section in bytes
- STYPE	- 4-character data source type identifier
- CTYPE	- 4-character data calibration type identifier
- MEMO	- 32-characters of comments
- CDATE	- Creation date
- CTIME	- Creation time
- FILT	- 32-bit filter map - one bit set for each band in area
- PKEYN	- Primary navigation codicil key
- SKEYN	- Secondary navigation codicil key
- PKEYC	- Primary calibration codicil key
- VALCOD	- Validity code value

Plus several other miscelllaneous fields. Sixty-four 32-bit words per entry.

Table 3.3.2: AAA AREA DIRECTORY DESCRIPTION

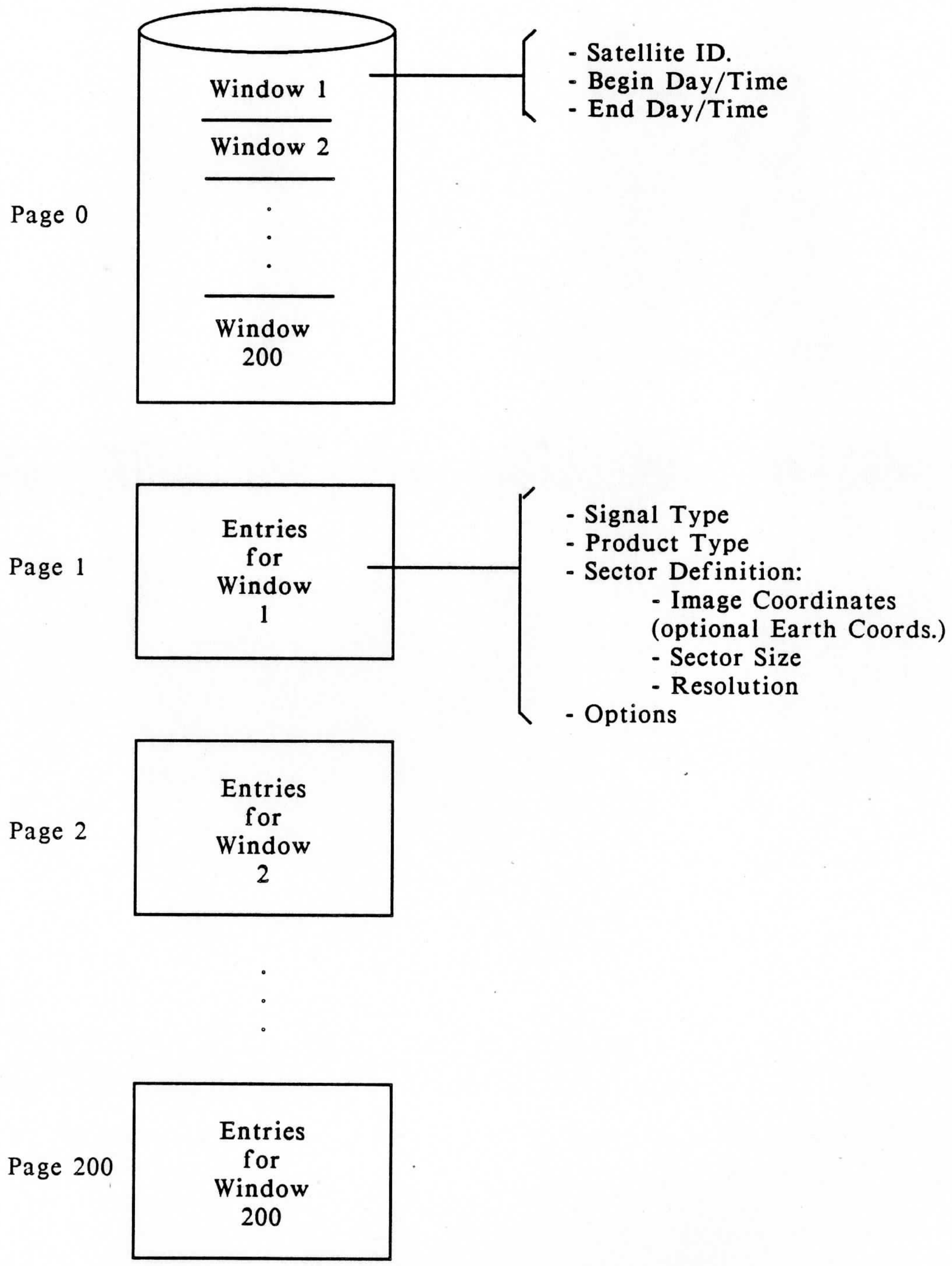


Figure 3.3.3: AAA Ingest Schedule File

Each block contains a header with the following information:

- message type code
- number of bytes in this message
- message created YYDDD
- message created HHMMSS
- scan line number
- Satellite SSS
- Satellite YYDDD
- Satellite HHMMSS
- miscellaneous

The message types currently defined are:

- frame start
- mode AA area numbers and Mode description
- Mode A documentation (message filed about every 200 scans)
- Mode AA documentaton (message filed about every 200 scans)
- frame end
- Orbit and Attitude
- ingestor active
- ingestor terminate
- AAA calibration
- beta block (filed 200 scans into the frame)

Most message types include the list of areas being written into for this frame, and the documentation, O&A, and frame end also include beta values.

Table 3.3.4: AAA IMAGE SPOOL DESCRIPTION

3.3.5 Debug spool file

The debug spool file is an LW file where the ingest handler places information appropriate to aid debugging any problems. This file is primarily used during development, but is also useful in tracking down signal related problems or bugs that surface later. During operations a minimum of information is placed here to minimize overhead. The basic format of this file is similar to the Image Spool file, but the data is always stored as text messages. A few user level tools are made available to selectively view this information, but this information is not intended to be useful to the average user.

3.3.6 Event schedule file

The event schedule file is actually a pair of LW files: the expression file and the flags file. The flags file contains integer and Boolean values which can be referenced by a Boolean language. The expressions in the expression file reference these flags and values. There is also a command line associated with every event expression, which is initiated when this expression evaluates as true. The contents of these files is described in Figure 3.3.6.

3.3.7 Time schedule file

The time schedule file is an LW file containing command lines to be executed and some associated parameters describing the time frequency and interval. The time scheduler accesses this file once per initiation to determine what commands to initiate at what times. The contents of this file are controlled by interactive users and are described in Figure 3.3.7.

3.3.8 Other Files

Other information regarding navigation and calibration parameters is kept in files call *codicil files*. Codicil files are keyed structures based upon arbitrary keys. These keys are defined by convention to identify the data type (e.g., NAV for navigation, CAL for calibration, etc.). Each assigned block is given a codicil number and this number is noted in all associated area directory entries. Each area has its own associated copy of calibration, and usually navigation (more than one area may point at a single codicil). See Figure 3.3.8A.

In addition, there is a file structure called the Master Navigation file structure. This file structure is used to archive a nominal orbit and attitude (plus betas and gammas) for each satellite/day. See Figure 3.3.8B.

3.4 Protocol

Protocol is the understanding between the ingest handler and the ingest hardware as to the meaning of the control fields (commands) passed back and forth. It also describes allowable sequences of commands, and the expected outcome.

The AAA protocol is quite complex because the AAA signal modes are complex. Only the MSI mode of AAA is comparable to the Imager in GVAR. For simplicity, the protocol described here will only deal with the MSI portion of the AAA protocol.

In order to describe the protocol it is useful to have a rudimentary understanding of the AAA signal format.

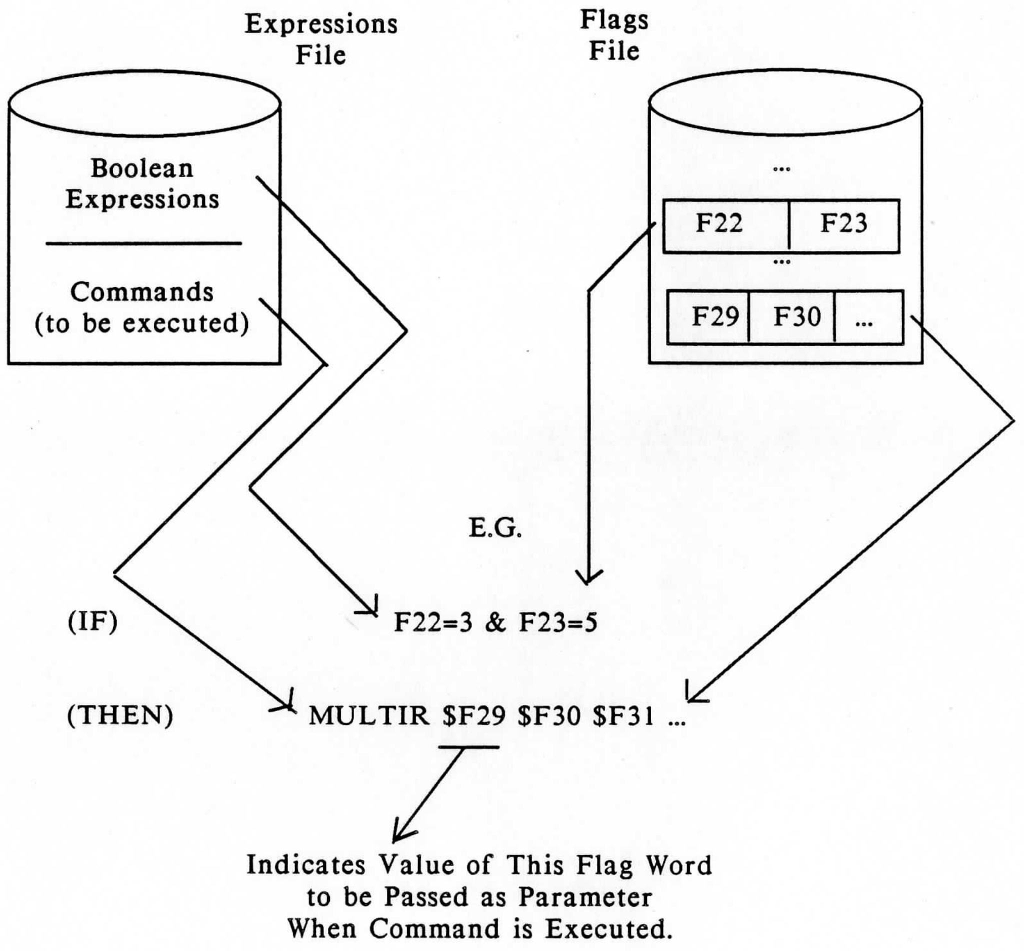

Expressions File
Flags File
Boolean Expressions
Commands (to be executed)
...
F22
F23
...
F29
F30
...
E.G.
(IF)
F22=3 & F23=5
(THEN)
MULTIR $F29 $F30 $F31 ...
Indicates Value of This Flag Word to be Passed as Parameter When Command is Executed.

Figure 3.3.6: Event Schedule File

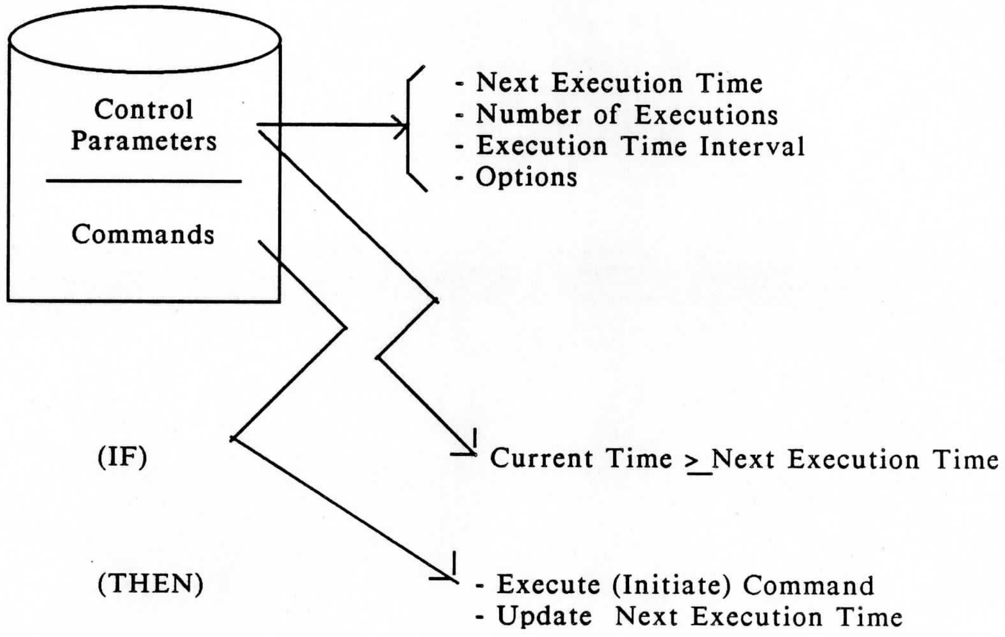

Control
Parameters
Commands
- Next Execution Time
- Number of Executions
- Execution Time Interval
- Options
(IF)
Current Time ≥ Next Execution Time
(THEN)
- Execute (Initiate) Command
- Update Next Execution Time

Figure 3.3.7: Time Schedule File

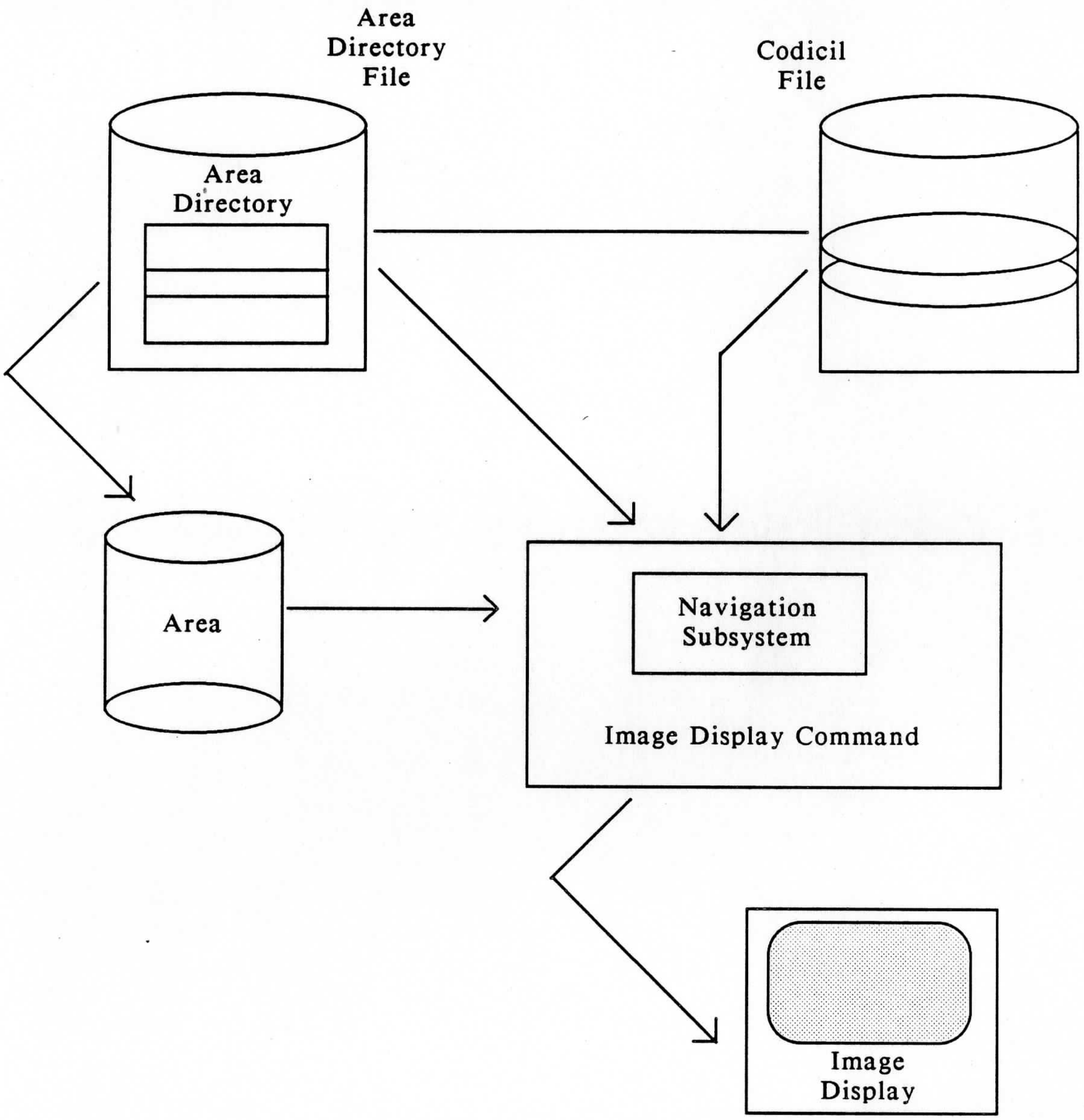


Figure 3.3.8A: Codicil Files

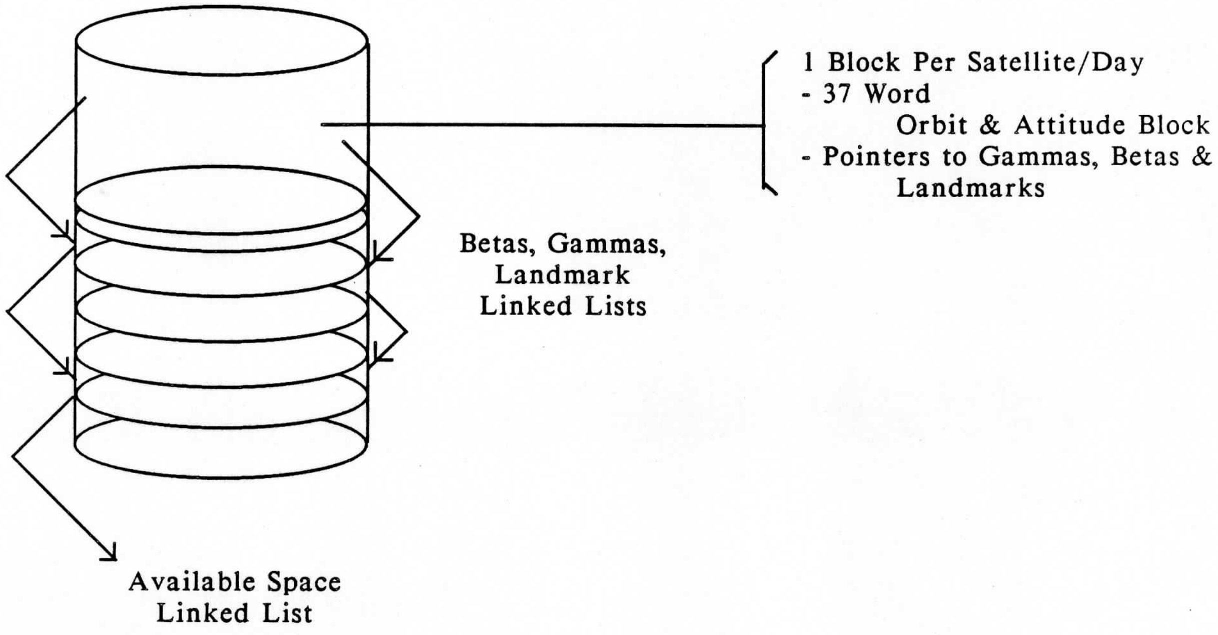


Figure 3.3.8B: Master Navigation

3.4.1 AAA signal overview

A AAA image transmission is composed of *scans*. Figure 3.4.1 is taken from Figures 1 and 2 from the 'Operational VAS Mode AAA Format'. It describes the 12 *signal sectors* which make up a single AAA scan.

IR sectors contain one line of one band of IR data
VIS sectors contain one line of Visible image data

Full resolution is:

VIS	- always 1 KM.
IR	- along a line - 4 KM.
	- from line to line - 8 or 16 KM, depending on the sensor.

3.4.2 AAA ingestor protocol description

It is important to note a couple of performance considerations the protocol must account for:

- There are 50 milliseconds per sector. We decided that this was adequate time to transfer as many sectors as we needed to the mainframe. Thus, only two sector buffers were provided in the hardware (one being filled, one being emptied).

- The response time of the mainframe from the end of the command chain to the *initiation* of the ingest handler task can be on the order of 100 to 150 milliseconds, depending upon system loading. This means there always must be two complete command chains in the queue, so that the data arriving while the ingest handler is processing a completed command chain is not lost. Double buffering of command chains in the ingest handler is also implied. This is essentially a 'ping-pong' kind of arrangement.

The first simplifying decision we made was to have each command chain correspond to exactly one satellite scan. 600 milliseconds is plenty of time for a single command chain, and it is simpler to think of command chains and scans one-to-one.

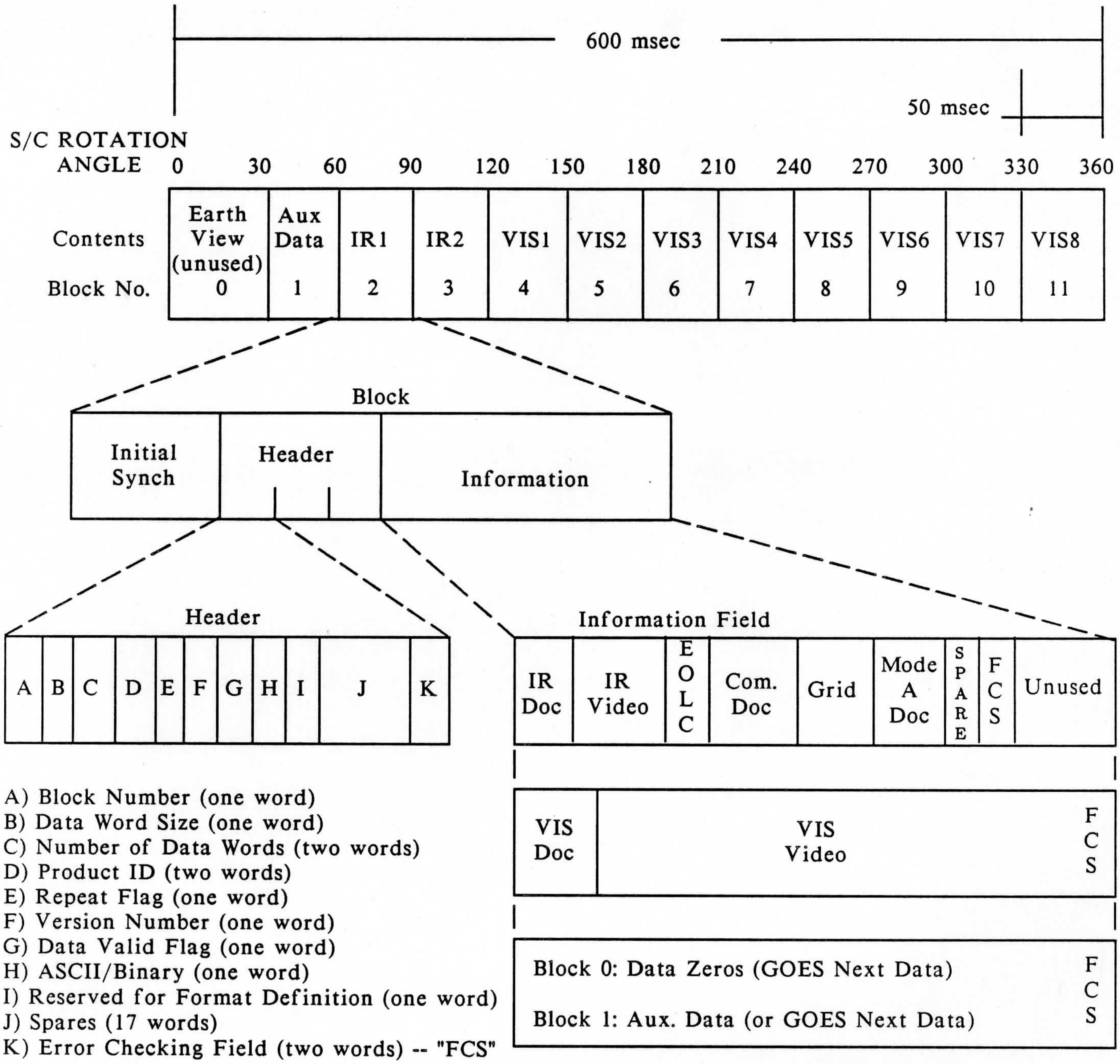
The way we chose to use the two command chains is for the ingest handler to predict what data the signal would contain two complete scans ahead of receipt. This sounds difficult, but there is always some separation between signal mode changes, allowing the handler to resynchronize with the spacecraft before again taking data. A scan or two may be lost while the ingest handler resynchronizations.

Given the desired database design for each line of image data and the signal format, the protocol really describes the transformation from signal to database format. The protocol is illustrated in Figure 3.4.2.

The commands in the illustration have the following functions:

- SYNC - wait for IR1, or scan timer to elapse, whichever is first.

- COMDOC - Send the ingest hardware/firmware the validity code value and miscellaneous other info. Retrieve the common documentation for this scan, plus a set of condition code flags (e.g., SYNC timed out,...).



A) Block Number (one word)
B) Data Word Size (one word)
C) Number of Data Words (two words)
D) Product ID (two words)
E) Repeat Flag (one word)
F) Version Number (one word)
G) Data Valid Flag (one word)
H) ASCII/Binary (one word)
I) Reserved for Format Definition (one word)
J) Spares (17 words)
K) Error Checking Field (two words) -- "FCS"

Figure 3.4.1: AAA Signal Format

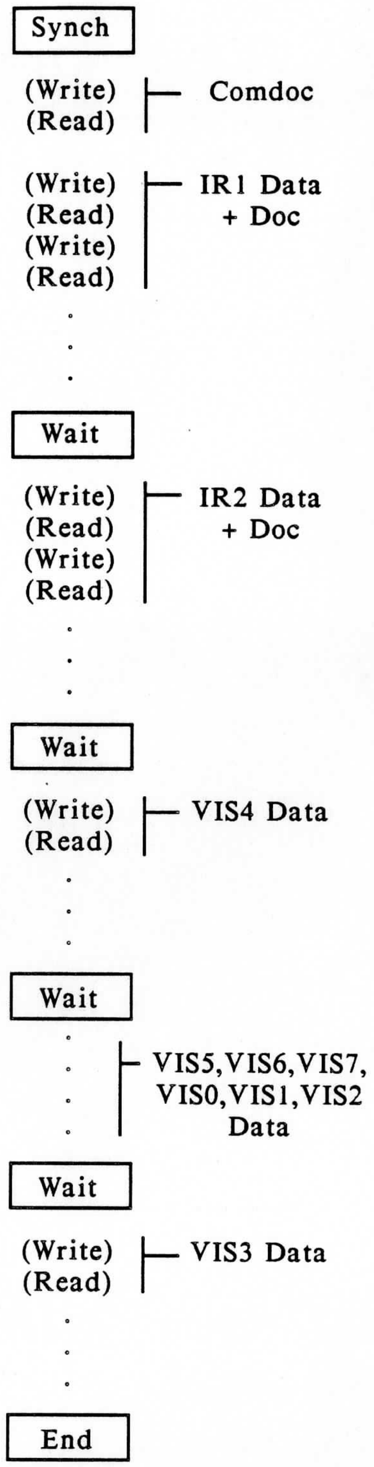


Figure 3.4.2: AAA Protocol

- WRITE/READ IR double pair - Write type of data read info, starting offset, length to transfer, whether to precede data with validity code, sample down factor, etc. Read back the IR data. Write the type of read and info for the common doc, READ the common doc as data.

- WRITE/READ VIS pair - Same purpose as read/write IR, but visible data uses a single pair; no common doc prefixed before it.

Note that as many WRITE/READ pairs may occur together as desired. For any data read resolution reductions of 1,2,4,8,16 by sampling or averaging, and for only a select portion of the total scan width, can be requested.

- WAIT - wait for start of next sector, or sector timer to elapse.

- IDLE - wait for IR1. If 90 seconds elapses without an IR1, then the hardware returns a Unit Exception status (command chain abnormal termination) to the ingest handler. This prevents the IBM system software from deciding the unit is ill because the unit has not responded to any outstanding command chain within 90 seconds.

Examples of command chains used for AAA can be found in Appendices A and B.

3.5 Ingest Hardware

3.5.1 Multisourcerer

The ingest preprocessor is a card, or card set, that plugs into a chassis called the Multisourcerer. The Multisourcerer essentially maps between the Multibus and the IBM channel. The Multisourcerer contains a driver card and a General Purpose Channel Interface (GPCI) card. The low level functions necessary to communicate with the IBM channel are handled in these cards. For example, the ingest hardware, or personality card, communicates to the GPCI by a status/control register and through a Multibus backplane. A Multisourcerer can hold up to eight personality cards. Multiple Multisourcerers can share a channel, within the loading limitations of the channel. The Multisourcerer with an example set of personality cards is illustrated in Figure 3.5.1.

3.5.2 AAA ingest (personality) card

Input to the AAA card is Stretched VISSR in GOES-BUS format. GOES-BUS format is the standard output format from GOES frame syncs.

The AAA ingest card data flow is illustrated in Figure 3.5.2.

Note that there are only two input buffers, each long enough to hold one sector of the satellite scan. All data desired from a sector must be transferred out of these buffers within one sector time - 50 ms.

The ingest cards are responsible for interfacing to the frame sync on one end, and the GPCI on the other end. Commands to the ingest hardware/firmware are passed along from the IBM mainframe by the GPCI card. All data formatting is performed here. The GPCI merely treats it as a string of data bytes to be sent out the IBM channel.

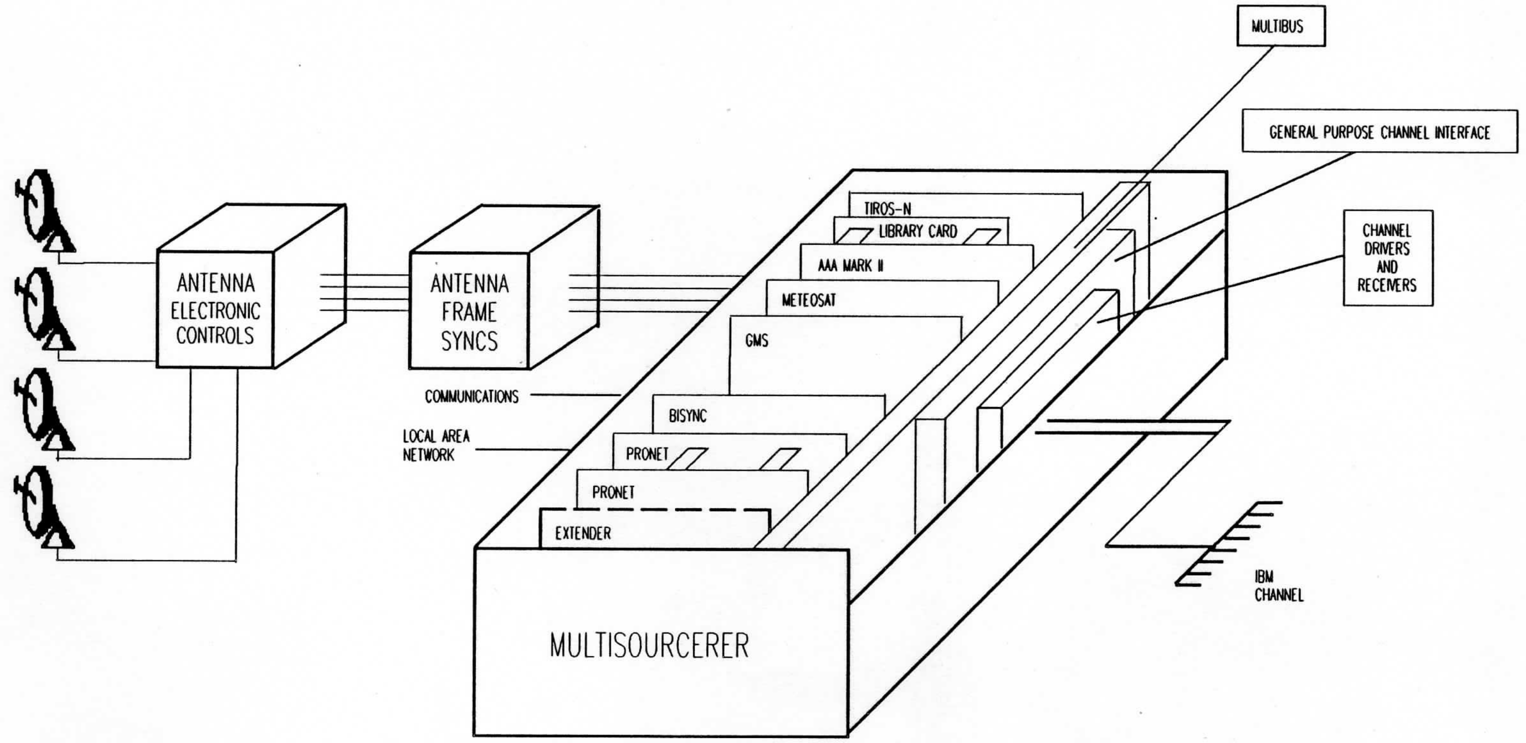


Figure 3.5.1: Multisourcerer Configuration

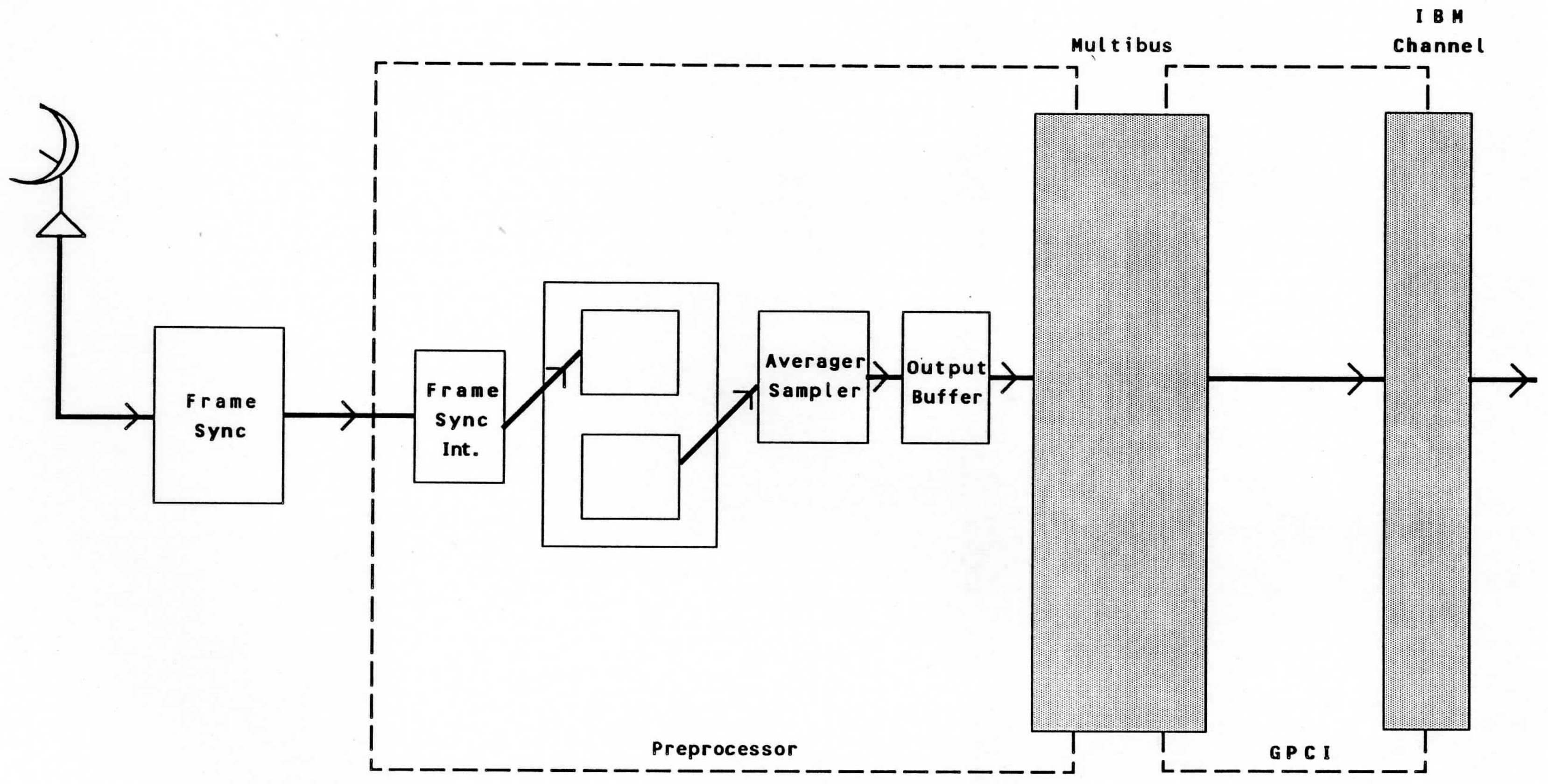


Figure 3.5.2: AAA DATA FLOW

Basically, the hardware/firmware is responsible for all decisions within any one scan (e.g., whether a signal sector was lost). The scan-to-scan decisions are made jointly with the Ingest handler (e.g., determination of the scan line number).

3.6 Ingest Handler

The ingest handler is the software on the mainframe directly responsible for communicating with the ingest personality card, and routing output from the ingest card to system disc. BDAM is used for disc access. The EXCP macro is used to send commands to the ingest card.

The ingest handler is responsible for all decisions regarding image start/stop and associating ingest scheduler requests with the current satellite signal. The scan-to-scan decisions are made jointly with the ingest card.

When associating requests with the satellite signal mode, extent of coverage and satellite time must also be taken into account. If any part of the request overlaps the expected satellite signal in extent and time, then the intersection of what the satellite is sending and the request is taken.

Each ingest handler is written almost entirely in Fortran. FORTRAN was chosen because the rest of McIDAS is written in FORTRAN, and is much easier to code than assembly language. Each ingest handler contains three concurrently executing control points. This makes writing the ingestor more complicated, but is necessary for performance. Initiation of any one control point is dependent upon completion of either an I/O or timer event.

3.7 Event Scheduler

The event scheduler is executed periodically (about every 48 seconds) off the time scheduler. It reads the ingest spools and evaluates all expressions in its expression list once for each entry in each spool. Every time an expression is evaluated as true by the event scheduler, a command line associated with that expression is executed. Thus, processes are initiated whenever certain combinations of events occur.

Filing of area directory entries, filing of navigation and calibration information, messages to the operator console regarding stop and start of ingests, and miscellaneous user requests are handled here.

3.8 Time Scheduler

The time scheduler is initiated off the master system clock about every 48 seconds. Any commands on its command list which are requested for this time are executed.

User commands to process the real-time data (e.g., TV displays, remaps, resectorize, etc.) are frequently placed on the time schedule. This works well when the schedules are relatively stable. For example, regardless of the AAA satellite mode, there is always a globe, or near globe, on the hour. Commands to process the hourly globe can then be reliably scheduled via the time schedule.

3.9 Database Access

McIDAS allows concurrent access by many users. Areas are not locked against reads. Only the area directories are locked against writes. Areas which are being written into by the ingest handler can be read concurrently. If the application gets ahead of the ingestor, it just picks up zeros for the data (which appear as black lines on the display).

In the AAA design, there really isn't any way for the application program or workstation user to know where in the area the ingestor currently is. Anyone with 3270 access can tell by logging onto the ingest status display screen.

Data structures other than areas have locks only to prevent write collisions. The locks are held at the subsystem level. Multiple read users are always permitted.

3.10 System Constraints

Because there are only 600 ms to get all data from any one scan from the hardware into the computer memory, there are, in general, only 600 ms to get those tracks written to disc. This implies performance constraints on the system configuration.

3.10.1 Ingest handler needs highest dispatch priority

The ingest handler must have the highest priority of any application. This is necessary to insure minimal time lag between the completion of any channel I/O and the initiation of the ingest handler. The intention of the DPRTY=(15,15) on the ingestor proc is to insure highest dispatch priority.

The ingest handler is non-swappable, but does not require contiguous core to be loaded (not V=R).

3.10.2 Priority queueing needed for disc I/O

The system generation must specify priority queueing for disc I/O. This, coupled with the previous requirement, insures that the disc response time is as good as possible by giving the ingestor top priority in the disc queue.

3.10.3 No long, chained I/Os on disc channel

Long, chained I/O's on the same channel, even to different address spaces, can cause excessive I/O wait times for the ingest handler. This is so because any chain started must finish before the ingest handler priority comes into play.

Although short unchained I/Os are required, we normally recommend a channel, controller, head of string, and at least one address space be dedicated to areas only. This not only insures adequate performance to allow the ingest task to work well, but also insures enough spare disc channel time to allow timely interactive response.

3.10.4 Other devices on same channel as ingestors

Bandwidth is an important consideration on the ingest channel. We normally recommend only ingestors be connected to the ingest channel. Even then, it is possible to overload a single channel with enough ingestors.

3.10.5 IOGEN views ingestor as device type DUMMY

The ingest hardware looks like device type DUMMY to MVS. This can be a limitation, since MVS/SP only allows 100 DUMMY type devices on a system.

4.0 GVAR INGESTOR DESIGN

4.1 Overview

The basic philosophy of the Imager/Block 11 and Sounder ingestors is for the software to ask the hardware what data it has in its buffers and, in the same channel command, ask for some older data it already knows about. Thus information about current data and some older data are transferred in every channel command. The most important implication is the amount of buffering needed in the ingest hardware: several complete scans worth.

Notice how this differs from the AAA method - predictive for AAA vs. deterministic for GVAR. There are several reasons for this change:

- The GVAR signal stream cannot be predicted. The possibility of a priority frame means there is no way to predict the characteristics of the next scan.

- The AAA method caused loss of data whenever the mode changed unexpectedly, or when noise began to affect the predict logic. This was deemed unacceptable. In the GVAR method there still may be loss of data under certain noisy signal conditions, but certainly much less, and less often, than in the AAA case.

- The error handling logic (which controls data loss) is exposed in the ingest handler rather than fixed into the overall design. In the AAA design the error handling was much more constrained because of the predictive nature of the protocol. When errors occurred, the data from the commands in the queue were lost. For GVAR, since identification is received before the data is requested, the error handling algorithms are totally contained within the ingest handler. This means the error handling algorithms can be more easily modified.

- The technology is just now to the point where we can get enough buffering on a plug-in card to allow this kind of logic.

A graphic representation of the GVAR Imager ingestor dataflow is presented in Figure 4.1. The Block 11 ingestor has an identical flow, only producing different data products.

4.2 Design Methodology

The GVAR Top-Level Design was worked out through a series of meetings, and numerous informal discussions. The starting assumption was to modify the AAA design to work for GVAR. In addition, since several changes were needed, it was decided that the modified approach should try to satisfy some of the problems with the AAA design. The gist of the approach is as follows:

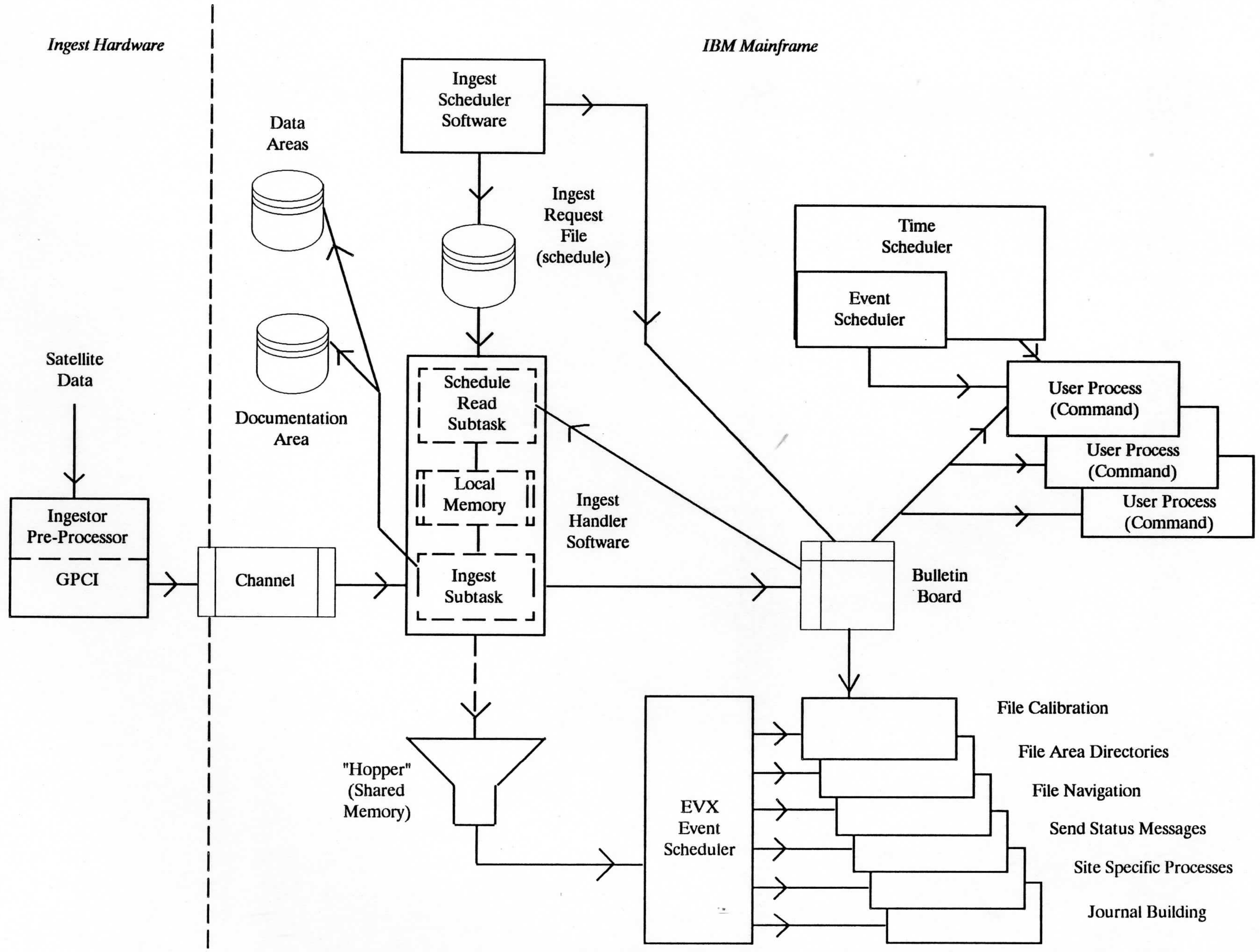


Figure 4.1: GVAR Imager Ingestor Data Flow

4.2.1 <u>What is different about the GVAR signal</u>?

- *Variable record length*

Maximum record length is about 25% larger for GVAR than for AAA. This implies a need for larger buffers throughout the system. Minimum size is only a few bytes. This implies low overhead per line is needed in the data structure.

- *VERY variable timing*

Not only does the record length vary, but the sectors in which the scan data are transmitted do also. This means the amount of time to handle an individual sector will vary depending upon the satellite scan mode. The variation is from about 119ms/scan to about 1045ms/scan, or nearly one order of magnitude!

This variation in the signal time used depending upon the amount of data being sent translates into less wasted signal time. Useful data is being sent a higher percentage of the time than on a comparable AAA transmission (on average). This means a higher duty cycle for the hardware and software receiving the signal - more resources (channel, disc, CPU) will be required to handle the data.

- *Multiple IR records in one IR sector*

In sector IR1 there are four logical IR records and in sector IR2 there are three. This must be taken into account but does not appear to have performance considerations.

- *Faster image switching (priority frame)*

The signal may switch from one logical image to another without warning; between scans. This means the ingest handler must be able to link up to user requests in less than one scan time, and in priority frame mode that will probably be quite a short scan time.

Repeat frames ('repeat last scan scenario') will also cause fast switching times, but not instantaneously as for priority frames.

- *Ability to resume an image after interruption (priority frame)*

An image can be begun, interrupted for a priority frame, and then resumed, multiple times. This will require extensive contingency buffering in the ingest handler to allow continuation of an image later. The hardware/firmware only needs to provide enough buffering to allow the transition; no information needs to be saved about the prior image.

- *Not predictable. Must react, not predict. (priority frame)*

This change has driven the majority of the ingest system changes from the frame sync through the ingest handler. Again, notice that priority frame is the driver here. However, the design changes for this reason will be beneficial in many ways besides enabling the handling of priority frame (see GVAR overview).

- Sounder data in block 11 records, not part of image data stream

All of the sounder data is packaged in the near-real-time portion of the signal: block 11s. This has its benefits and costs. The advantage is a simpler imager format. The main cost is either another piece of hardware to interpret the sounder blocks (development cost, channel, memory, cpu), or a set of decoders to reformat the sounder block 11's into the database format (considerably more cpu).

- Navigation

The decision to use a 3-axis stabilized spacecraft instead of a spinner may cause the navigation subsystem to have to be rebuilt. The body of knowledge acquired about the behavior of spinners will not be applicable, and a new body will have to be acquired through time for this breed of spacecraft. The main cost will be decreased reliability, and probably accuracy, until sufficient 'lore' is regained.

There is not sufficient information available at this time to make the decisions about what exactly needs to be done to produce accurate navigation.

- Calibration - simpler; more like mode A

The changes for the imager and sounder calibration should actually represent a simplification from the AAA format. This will save applications time resources (cpu). This calibration should be a simple table lookup, rather than a set of equations to be run on every data point.

4.2.2 <u>What new requirements are there?</u>

- Stricter application response requirements

The GVAR Ingestor Functional Requirements document lays out some stricter system response requirements than existed for AAA. This means additional work in interfacing the ingest handler to the rest of the system is required. This is viewed as a change for the better, since the older system had some inefficiencies which need addressing, particularly in the event scheduler.

- Better visibility of ingestor status from other processes

The GVAR implementation must support co-processes. That is, processes must be allowed to execute in parallel with the ingest handler on the same data. The co-processes would monitor the status of the ingestor; what image it is on, what area is being written, what line is now completely written into the area, etc. A memory resident structure called the Bulletin Board will be used to make this information available.

- Full resolution, full sector requirement

There was no requirement for AAA to ingest the full image. This has minor effects on the design, such as allowing adequate buffering in the ingest handler.

- ALL data in data stream must be available on system.

This includes the entire block 11 data set. Block 11 must be ingested to get the sounder imagery. Requiring all the data be made available impacts the design minimally (selection of interim data structure) , and the work to be done nominally (one command to view data needed).

- *Efficiency. Must plug in where AAA ingestors were*

The interim systems NESDIS is using with AAA will certainly need to handle GVAR before eventual consolidation on a much larger system. This means the GVAR ingestor should ideally fit into the same slot as the current AAA ingestors. However, the basic design of GVAR is to get more data into the same signal. In addition, GVAR represents some new capabilities (for example, additional bands in the imager), which the NESDIS users will undoubtedly want to utilize. The net effect is extreme resource shortages on the interim systems. The GVAR design will have to be inherently efficient, flexible, and the interim systems will have to be augmented in order to handle GVAR.

- *Compatibility with AAA, may run both concurrently.*

Since AAA and GVAR format signals are expected to be needed by the interim systems concurrently; it is necessary for the GVAR and AAA ingestors to be able to co-exist, preferably on the same channel.

4.2.3 Main design decisions made

- *Imager and Block 11 ingestors as separate units on the channel*

It was decided to implement these two basic ingestors as separate units on the same channel, and possibly the same card, to decrease the protocol complexity. This also made each of the two more like AAA and so reduced the risk.

The decision about a one or two card implementation was purposely left vague. It was felt this should be an implementation time decision, and was not important to the overall design scheme. The driving force on this decision will be the amount of space needed for memory.

- *Hardware buffers data by scans rather than by signal sectors*

This requirement is imposed by the need to request identification information about each scan before requesting the actual data. There is not enough time to request identification information, wait for mainframe to interpret the identification information, and then request the data sector-by-sector (or even scan-by-scan).

- *Make buffering flexible to handle variable scan lengths*

The amount of space on the ingest card to devote to memory is relatively small. Some methods of reducing the memory requirements are necessary. Since the signal transmission rate is constant, allowing the buffering to be flexible causes no more total memory to be used for short scans than for long ones. This provides equivalent time buffering, but unequal number-of-scan coverage. This is sufficient since the purpose of this buffering is to provide adequate time for the mainframe to receive id information, digest, and respond. It also introduces additional complexity to the firmware, but is felt to be worth it.

The exact amount of buffering is to be determined in the detailed design, but was estimated at about four megabytes (if used as 12-bit words). This was felt to be adequate time coverage even in noisy signal environments.

- *As for AAA, each unit is microcomputer controlled*

This point was not even debated, as we all agreed immediately. The decision as to which micro to use is left to the detailed design. The AAA design used an Intel 8085.

- *Simplify the protocol to optimize channel utilization*

Even before the performance measurement, we felt that the channel might be overloaded. We had bumped into this limitation when putting several AAA ingestors on the same channel. So, we decided to make some modest protocol changes to improve our utilization of the channel. These are described further in the protocol section.

- *Handle priority frame in the ingest handler software*

We decided there was no need to put constraints on the ingest hardware for the priority frame possibility. Since it is expected to be seldom (if ever) used, it was felt that the software should take on that responsibility. The cost to the ingest handler software is increased complexity, additional buffering requirements, and possibly minor increased CPU utilization. The performance costs downstream in the application software are much more severe, much more complex, and were left for later evaluation.

- *Handle fast image switching better in the ingest handler*

The hardware can switch from image to image quickly because it does not have decisions to make when this occurs. The software must match the ingest schedule to the signal and make decisions about sectorizing and where in the database to put the data. All necessary information must be resident in the ingest handlers memory constantly to allow this switch in minimum time. A scheme for doing this was developed and will be implemented (see Section 4.7).

- *Better ingest handler status accessibility*

AAA has a status display which is useful for humans, but not for processes on the system. It is necessary to remedy this situation in order to allow co-processes to track the ingestion of data. This way creation of products, for example GOES-TAP, can proceed in parallel with ingestion, reducing total time until product availability.

- *Ingest request scheduling changes needed*

There is only one set of software used for scheduling ingest requests for all satellite types. The original design and implementation of this software has become brittle, and not as useable as it should be. Changes are planned to recast the current design into a couple of separate ingest request schedulers based upon basic ingestor type. This also allows changes to the basic data structures used to hold the requests.

- Event Scheduler changes needed to improve timeliness

The AAA design allowed as much as a couple of minutes of lag between the receipt of data and the notification to a post-process to begin work on the data. It is necessary to reduce this to under one minute. This involves a more responsive design and implementation of the software responsible for starting these post-processes. The main change is to make the communication between ingest handler and event scheduler occur via shared memory rather than disc.

4.3 Ingest Scheduler

The AAA ingest scheduler also schedules requests for all real-time ingestors. This includes TIROS and others. After some experience using this schedule software, it is obvious that we need to divide the current scheduler into two or three separate schedulers; one scheduler for geosynchronous satellites, one for polar orbiting satellites, and quite possibly one for miscellaneous signals such as Auxiliary block and Block 11. The user interface and data structures can then be tailored more to the satellite type.

The GVAR ingest scheduler operation and organization of commands will be very much like the AAA version. The different types of real-time areas will be explicitly schedulable. This means that there will be at least one entry for documentation areas, several for IR and VIS areas, etc. Changes are planned to:

- Allow better scheduling by ingest scenario mode

For AAA the choices were limited to the 'instantaneous' signal mode: Multi-Spectral Imaging (MSI) or Dwell Sound (DS). No provision was made to discriminate by scenario: Normal, Rapid Image Scanning OPeration (RISOP), etc.

For GVAR the choices can be expanded because the common documentation is expected to contain a flag word describing the scenario: Normal, Watch, Warning. If this documentation word should not be present, there is a fall-back solution of having the ingest handler determine the scenario in which the spacecraft is being operated. One possible algorithm is based on image size, which is known at image start. Large images would be considered as Normal, small images would be considered as Warning, and the rest would be considered as Watch. These three scenarios are expected to produce images with those size distinctions.

Priority frames will also be determinable from the common documentation, and could be a fourth scenario to key on. The decision whether to add a special priority frame scenario mode to the ingest scheduler is delayed until a more detailed design is done.

- Allow more outstanding requests at once

The current limit is 15 requests per real-time satellite (the ingest handler only allows 10 to be serviced at any one time). For AAA this was adequate. For GVAR this is definitely not. Each of the IR bands will have to be individually scheduled in order to be individually stored. Several resolutions are typically needed. The current goal is to allow about 50 requests per real-time satellite.

- Easier to use scheduling commands

By breaking the schedule file and scheduling commands up by distinct type - geosynchronous, polar, and miscellaneous; the user interface should be able to be simplified.

4.4 Data Structures

The data structures constraints are the same for GVAR as for AAA, or for any other real-time satellite data source, for that matter. These constraints, as described in the AAA section, are because of the real-time and interactive nature of the system, not any particular data source.

4.4.1 Areas - Imager, Doc, Block 11, and Sounder

The rules for building areas are the same as for AAA. The format of the GVAR areas depends upon the type of data.

Although areas are primarily for imagery data, there is nothing to prevent them from being used as a convenient place to keep other line-oriented data. Areas are the only practical place to put moderate to high volume data from the ingest handler.

Several kinds of data are planned to be stored in the area file structure. All these types of data can be accessed by the standard area access routines, and for documentation and grid data special access routines will be built on top of the area access routines to make access easier for the applications.

- Documentation and Grid areas

The common documentation section of the GVAR signal is much larger than in AAA, and there are more IR bands which need to be stored separately. Keeping the documentation on the front of each IR area, as was done for AAA, would be prohibitively expensive in disc space. A good alternative is to keep this data in it's own area, linking it to the associated IR and VIS areas by convention or specific pointer (to be decided later).

The documentation area lines are linked to specific lines in the IR and VIS areas by the coordinate system (line and element in the directory), and by the same headers appearing in both data structures. The headers provide a useful verification that the two structures are consistent.

Note that the grid data is part of the documentation. No independent data structure is expected to be built. This structure is already adequate and has the advantage of simplicity.

Since separate access routines will be supplied for grid and documentation access, these data will appear as separate structures to the applications programs.

The structure of documentation areas are described in Figure 4.4.1A.

- Imager areas

The GVAR Imager data is most like the AAA MSI mode data. See Figure 4.4.1B.

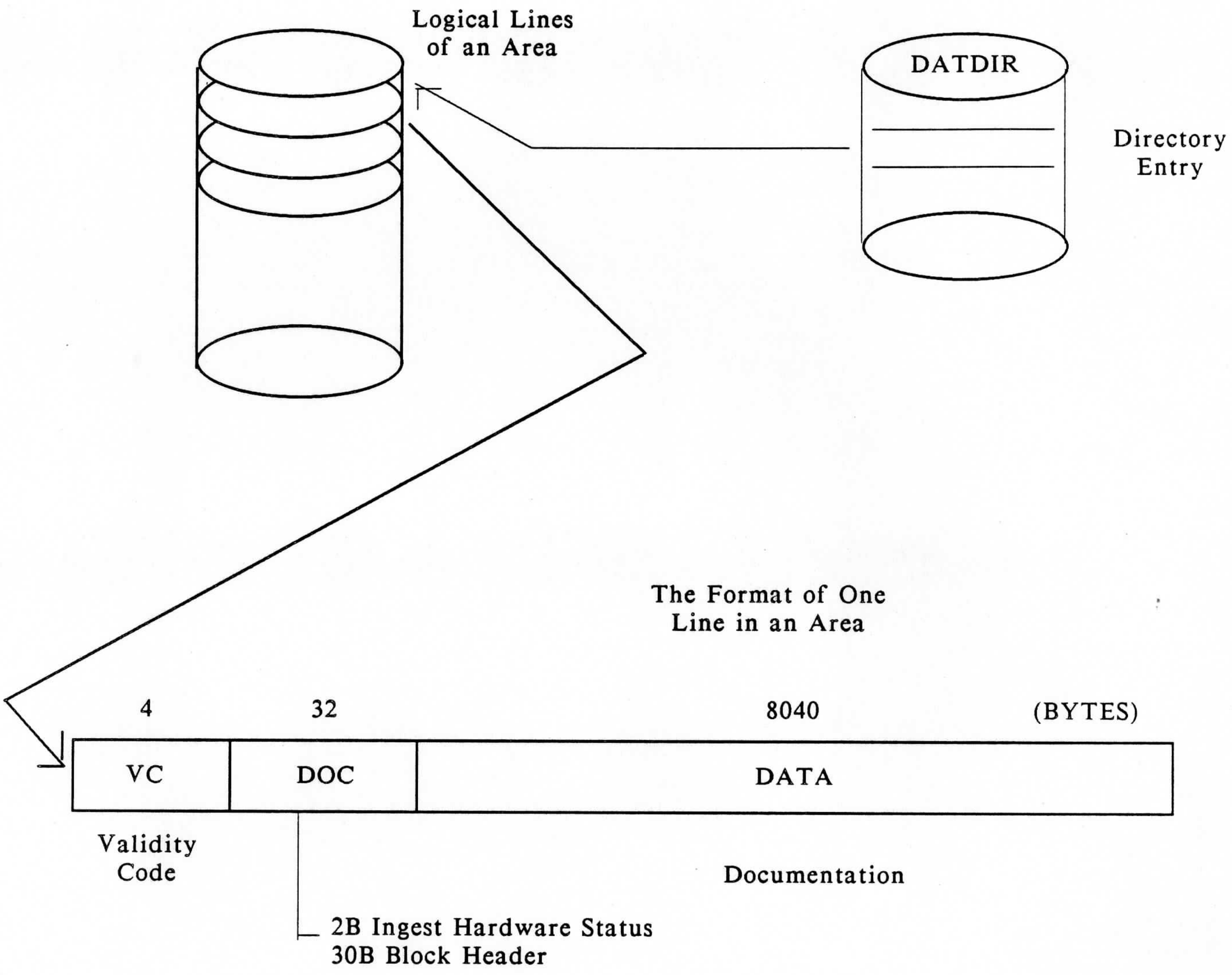


Figure 4.4.1A: GVAR Documentation Area Format

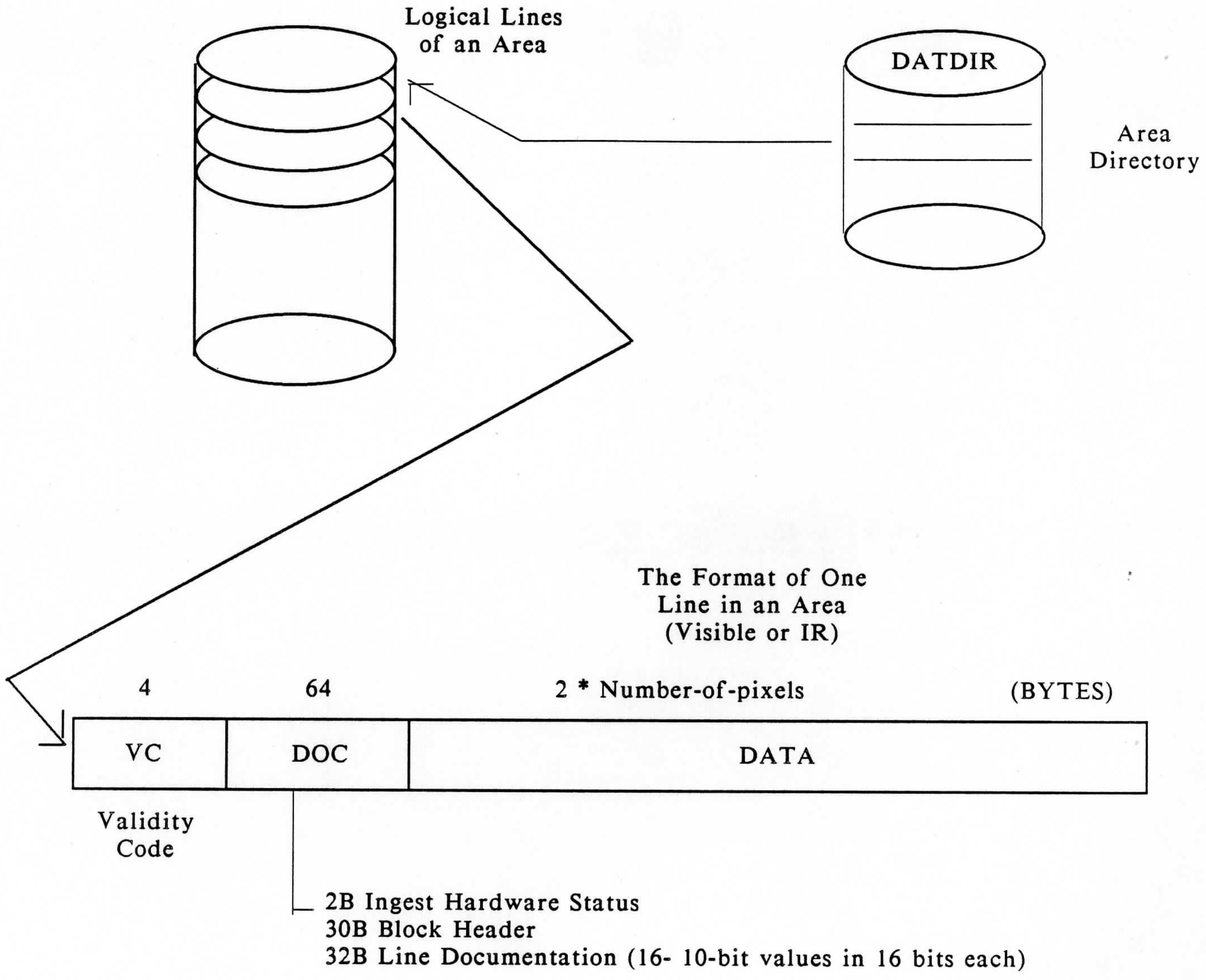


Figure 4.4.1B: GVAR Imager Area Format

- *Block 11 areas*

The block 11 areas are also not imagery data. This is just a convenient structure for the ingest handler to put this data. The requestor will be able to select the data to be put in this structure by block type code sets (from the block 11 header) through the ingest scheduler. In this way different data for different purposes can be directed to different areas. Up to 32 block types will be allowed per area, allowing for additional block types to be defined over the current 22.

Block 11 areas are expected to be the final resting place for some of the block 11 data. Other types (like the calibrated sounder data, sounder documentation, and text type blocks) will be taken from here by a post-process decoder, transformed into standard McIDAS data structures, and placed in the database.

The structure of the block 11 areas is described in Figure 4.4.1C.

- *Sounder areas*

The GVAR sounder data is most like the AAA sounder data. The main differences are: GVAR has more IR bands and an associated VIS band, and the data is placed in this structure by a decoder rather than an ingestor. It is possible special hardware will be built later to allow the sounder data to be placed here directly by an ingestor. If this is done, it will be strictly for performance. This structure will be finalized during the detailed design phase.

The calibrated sounder data is imagery, and so will be accessible through the standard area access subroutines. The uncalibrated data will be accessible from the Block 11 areas via the Block 11 access routine.

4.4.2 <u>Area directory</u>

The area directory file is expected to be used for GVAR as it was for AAA. The structure will be basically the same as for AAA. The only expected change is for a couple of additional words to be used (e.g., to indicate this is non-imagery data). This structure will be finalized during the detailed design phase.

4.4.3 <u>Ingest schedule file</u>

The ingest schedule file is expected to undergo some changes as described in sections 4.2.8 and 4.3. The basic contents of each entry will also be rearranged and made more efficient, both from the viewpoint of the ingest handler and disc space. The information contained will still be substantially the same. This is a reorganization, but not a redesign. The structure will be finalized during the detailed design phase.

4.4.4 <u>Image spool file</u>

The image spool file structure will not change. The use of this structure is expected to decrease as the new event scheduler takes on more of the functions previously performed through this structure. It is likely all navigation and area directory blocks will be deleted. The level to which this structure is obsoleted will be determined during implementation.

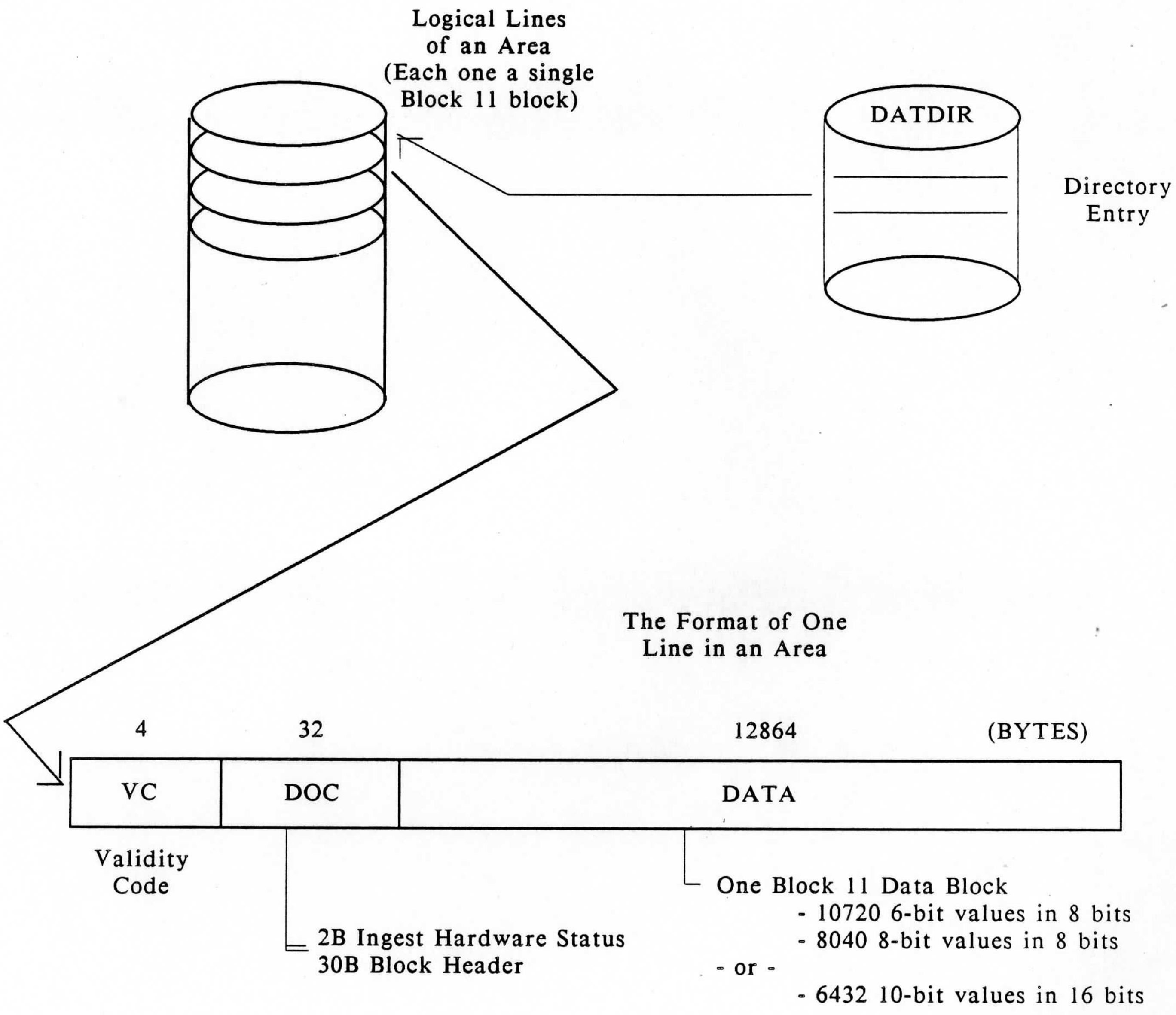

Logical Lines of an Area (Each one a single Block 11 block)
DATDIR
Directory Entry
The Format of One Line in an Area
4
32
12864
(BYTES)
VC
DOC
DATA
Validity Code
2B Ingest Hardware Status
30B Block Header
One Block 11 Data Block
- 10720 6-bit values in 8 bits
- 8040 8-bit values in 8 bits
- or -
- 6432 10-bit values in 16 bits

Figure 4.4.1C: GVAR Block 11 Area Format

4.4.5 <u>Debug spool file</u>

The debug spool file is expected to be used in the same manner as for AAA. It is necessary for debugging the ingest handler, but is generally not useful to the user.

4.4.6 <u>Event schedule file</u>

The event schedule file is used by the AAA event scheduler. It is expected to continue as before, but the commands which performed navigation filing, area directory filing, and probably some other tasks will be removed for GVAR. It will continue to be used for those tasks with lower performance requirements, or that require more flexibility than will initially exist in the new event scheduler (EVX).

A new event schedule/control file will be built to support a new event scheduler (EVX). More about this in Section 4.8.

4.4.7 <u>Time schedule file</u>

The time schedule file will not change for GVAR; however, an auxiliary file will be established to contain some tasks being routed to McIDAS from the new event scheduler. A disc file is useful because tasks can be queued even when the interactive portion of McIDAS is down. The time scheduler will remove entries from this queue and initiate their execution. The structure of this file will be determined during the detailed design.

4.4.8 <u>Journal file</u>

A journal file of all the signal activity for each ingestor will be kept through the new event scheduler (EVX). Events indicating changes in the signal mode or status will be passed to EVX, which will then add entries to the journal. A simple user command will be built to search and list the journal. The structure of this file will be determined during the detailed design.

4.4.9 <u>Other files</u>

A calibration file is not expected to be necessary for the Imager data, and is to be determined during the detailed design for the Sounder.

A navigation codicil will be necessary, but its form is yet to be determined. More information is needed to do this level of design.

It is not clear if the master navigation file will be necessary for GVAR. Probably, but in a different form. Again, more information is needed to do this level of design.

4.4.10 <u>Memory structures</u>

Two critical memory structures which exist in shared memory, and may outlive individual applications or the ingest handlers, are worth noting here.

- *The Bulletin Board*

The bulletin board is a region of shared memory (in the CSA region) used for communication between certain applications, such as the ingest scheduler, and the ingest handler. It will also be used for communication between some applications, such as the EVX control flag utility program, and EVX. Finally, it will provide instantaneous status information regarding the status of the ingest handler to any process that cares to look.

- *The EVX Event Hopper*

Pointers to the event input hopper are kept in shared memory, but the actual structure is inside EVX. This structure is a place for events to be held until EVX is able to process them. It can also be thought of as a FIFO.

4.5 Protocol

Protocol is the understanding between the ingest handler and the ingest hardware as to the meaning of the control fields (commands) passed back and forth. It also describes allowable sequences of commands, and the expected outcome.

The GVAR protocol is relatively simpler than the AAA protocol. The basic concept of buffering data extensively enough to allow requesting the identification of the data prior to requesting the data reduces the necessity of predicting the satellite behavior. It is still necessary to predict the satellite behavior on those occasions where the signal has become noisy enough to make the identification information suspect. Now it need not be built into the request protocol, but can be determined after the identification information has been received and scrutinized.

The basic differences between AAA and GVAR protocols have already been noted (see Section 4.2). One point mentioned for AAA not yet discussed, is how many scans are represented by each command chain. For GVAR this must be variable. Note that as the scan length decreases, the scan rate increases. The ingest handler must account for a minimum amount of time per channel command chain in order to guarantee the IBM can respond fast enough; 250 milliseconds is a typical goal.

4.5.1 GVAR signal overview

A very brief presentation is made here only to allow easier comparison to the AAA signal format. For more information see the GVAR Format Description (reference 5). See Figure 4.5.1.

4.5.2 GVAR Imager protocol description

Figure 4.5.2 illustrates a single command chain. Note that the Doc data being requested is for a different scan than the write/read pairs are requesting the data from. This reading of the new ID information while reading the old data allows the ingest handler to decide the fate of each scan's data before creating the command chain that will read the data. This is a change from AAA where the ingest handler predicted what data would arrive, and decide its fate before seeing and ID information.

The commands in the illustration have the following functions:

- SYNC - now more complicated than for AAA. If there is doc in the buffers that has not yet been sent to the host, continue immediately. If there is no doc, but there is data which has not yet been sent to the host, wait for the next sector zero, or short sector detector timeout (about 1.5 seconds),

DOC	IR1	IR2	VIS1	VIS2	VIS3	VIS4	VIS5	VIS6	VIS7	VIS8	SAD
0	1	2	3	4	5	6	7	8	9	10	11

- Each GVAR Block has 10032-bit Synchronization Code
  720-bit Header
  N-bit Information Field
  16-bit CRC

- Blocks 0 and 11 Have a Fixed Length Information Field of 64,320 Bits.

- Blocks 1 through 10 Have Variable Length Information Fields Directly Dependant on Width of Scan, with Minimum Information Field Length of 21440 Bits.

- A Single Imager Scan Generates Blocks 0-10 in Sequence.

- Blocks 0 through 10 May be Followed by Any Number of Block 11's (0-N) Depending on What is Available. In Priority Order, The Next Block(s) Transmitted will be:

1. Next Imager Scan- Blocks 0-10
2. Imager Compensation and Servo Errors- 1 Block 11
3. Sounder Compensation and Servo Errors- 1 Block 11
4. Imager Telemetry Statistics- 1 Block 11
5. Imager Spacelook Statistics and Data- 2 Block 11's
6. Imager Calibration Coefficients and Limits- 1 Block 11
7. Imager ECAL Statistics and Data- 2 Block 11's
8. Imager Blackbody Statistics and Data- 1 Block 11
9. Imager Visible NLUTS- 2 Block 11's
10. Imager Star Sence Data- 9 Block 11's
11. Sounder Scan Data- 2 to 400 Block 11's
12. Sounder Telemetry Statistics- 1 Block 11
13. Sounder Spacelook Statistics and Data- 5 Block 11's
14. Sounder Calibration Coefficients and Limits- 2 Block 11's
15. Sounder ECAL Statistics and Data- 3 Block 11's
16. Sounder Blackbody Statistics and Data- 5 Block 11's
17. Sounder Visible NLUTS- 9 Block 11's
18. Sounder Star Sense Data- 9 Block 11's
19. GIMTACS Text Messages- 1 to 2 Block 11's
20. SPS Text Messages- 1 Block 11
21. Auxiliary Data- 1 to N Block 11's
22. Fill Data- 1 Block 11

Figure 4.5.1: GVAR Signal Format

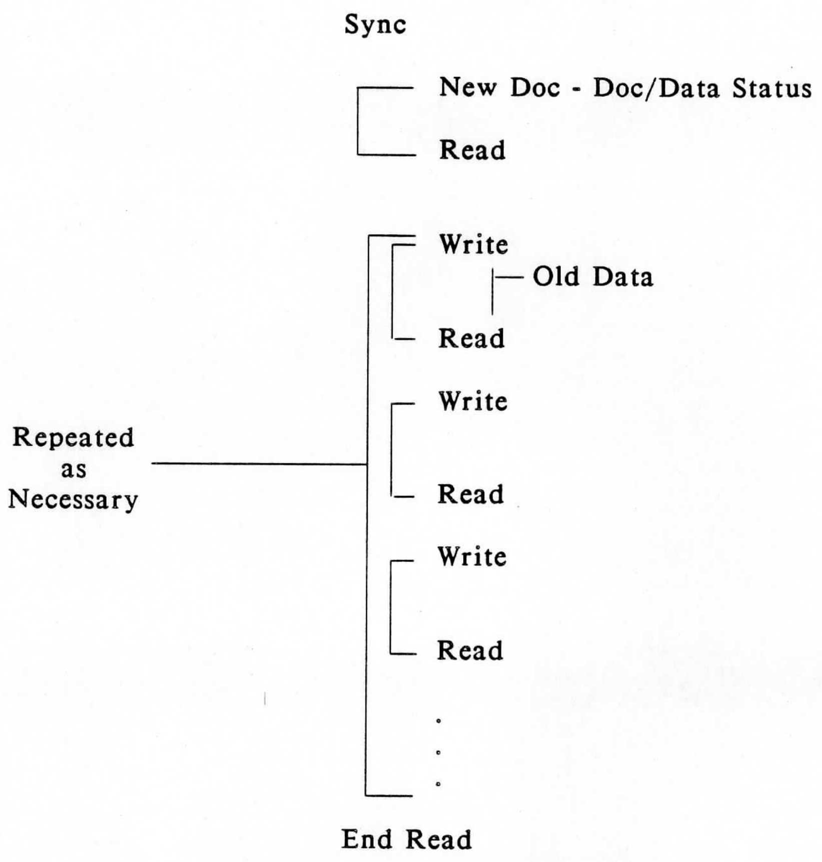


Figure 4.5.2: GVAR Imager Protocol

whichever comes first. If there is neither data nor doc, wait for the next sector zero, or a long sector timeout (about 29 seconds), whichever comes first.

- NEWDOC - a write/read pair that is used to send the ingest hardware the validity code and miscellaneous other info. Returned on the read is the identification of the next data scan not yet sent to the host, plus a set of status flags.

- WRITE/READ pairs - as many as are necessary to read one scan's worth of requests. Note that only data which is requested through the ingest scheduler is read from the ingest hardware. Resolution reductions of 1,2,3,4,6,8,12,16 by sampling or averaging, and for all or a select portion of the total scan width can be requested.
  These reduction factors were determined by user polls, and from input from various McIDAS sites around the world. They also turn out to be a relatively straightforward set to implement.

- ENDREAD - The endread command is used to note to the ingest handler the end of read requests for this scan. This is primarily to make the 'no requests' case unambiguous.

The entire NEWDOC, WRITE/READ, ENDREAD set may be repeated as many times as necessary to get the correct number of scans needed in one command chain for this particular satellite scan width. That is, for shorter scan widths, more sets must be included to encompass the same amount of time. As scan widths increase, fewer sets must be included.

4.5.3 Block 11 protocol

The block 11 protocol is very much like the Imager protocol. The main differences are:

- The NEWDOC command sends back up to 5 identification blocks, instead of always one as for the imager.

- The WRITE/READ pairs can specify whether to read as 6-in-8, 8-in-8, or 10-in-16 bit data. No resolution reduction is allowed for block 11's.

See Figure 4.5.3.

4.6 Ingest Hardware

4.6.1 The Multisourcerer

The AAA Multisourcerer is compatible with the GVAR ingest card. The only problem is throughput. The increased data density of the GVAR signal requires an increased bandwidth to get the data into the computer. The basic options are to add more CPU channels, or increase the efficiency of use of the existing channels. The latter option was selected because of compatibility with the existing systems. The following are two mechanisms recommended for increasing the efficiency on the IBM channel:

- A more channel efficient protocol. This has already been designed and is described in the protocol section.

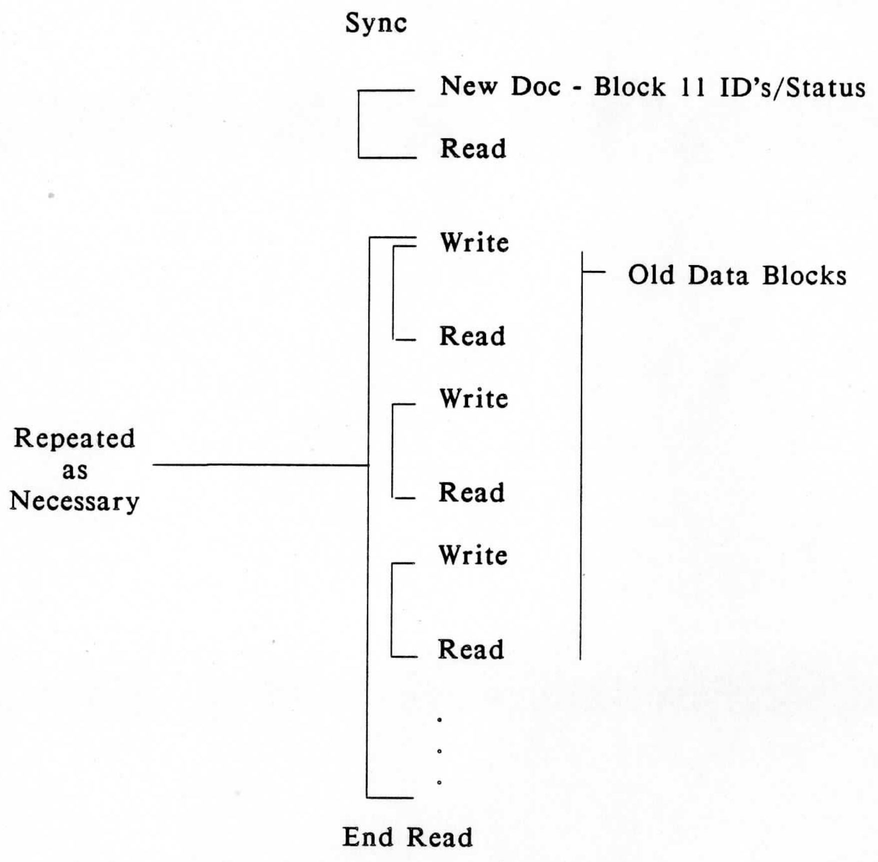


Figure 4.5.3: GVAR Block 11 Protocol

- Upgrade the Multisourcerer to handle Data Streaming mode. The Multisourcerer currently handles only Channel Interlock mode. This mode caps the bandwidth of a Block Multiplexer channel at about 1 MB/sec. If Data Streaming mode is implemented, the bandwidth of a Block Multiplexer channel is capable of about 3 MB/sec. The Multisourcerer has been designed with this data rate in mind, and just needs firmware, microcode, and possibly minor hardware changes to implement Datastreaming mode. On the other hand, Datastreaming is a very complex mode, and the learning curve is significant.

4.6.2 The Ingest (Personality) Cards

There are expected to be two logical cards in the GVAR ingestor. They may or may not be on one physical card - this is a detailed design time question. The concern with trying to fit both functions on one card is primarily space for the required memory. If both functions are implemented on the same card, they will be able to share the same frame sync interface, and, possibly, the same microprocessor. In any case, the two would have completely separate interfaces to the IBM host. The IBM host would not be able to tell the difference between a one card implementation and a two card implementation.

A two card implementation is simpler. That is what is described here.

4.6.3 Imager Card

Input to the Imager card is GVAR in GOES-BUS format. GOES-BUS format is the standard output format from GOES frame syncs.

The GVAR ingestor cards are described in Figure 4.6.3. The imager card and the block 11 card are expected to be the same hardware with different firmware and possibly a PAL change.

Note that there are multiple buffers. The exact number of logical buffers depends upon the satellite scan width, as the buffers are used efficiently regardless of the scan width, being dynamically assigned. Each buffer is one *scan* in size, rather than one signal sector as for AAA. The starting estimate for the amount of buffer space required is two megabytes (about 5 maximum width scans).

This card will do average or sample down at reduction 1,2,3,4,6,8,12,16 factors. These factors were determined over several years as the intersection of the set most needed by the users and the set whose implementation is straightforward.

The hardware/firmware is responsible for all decisions within any one scan. The scan-to-scan decisions are now entirely made by the ingest handler.

4.6.4 Block 11 Card

Input to the Block 11 card is GVAR in GOES-BUS format. GOES-BUS format is the standard output format from GOES frame syncs.

The block 11 sectors are also multiply buffered. This card is relatively simple, since no resolution reduction is necessary. The only complication is the need to

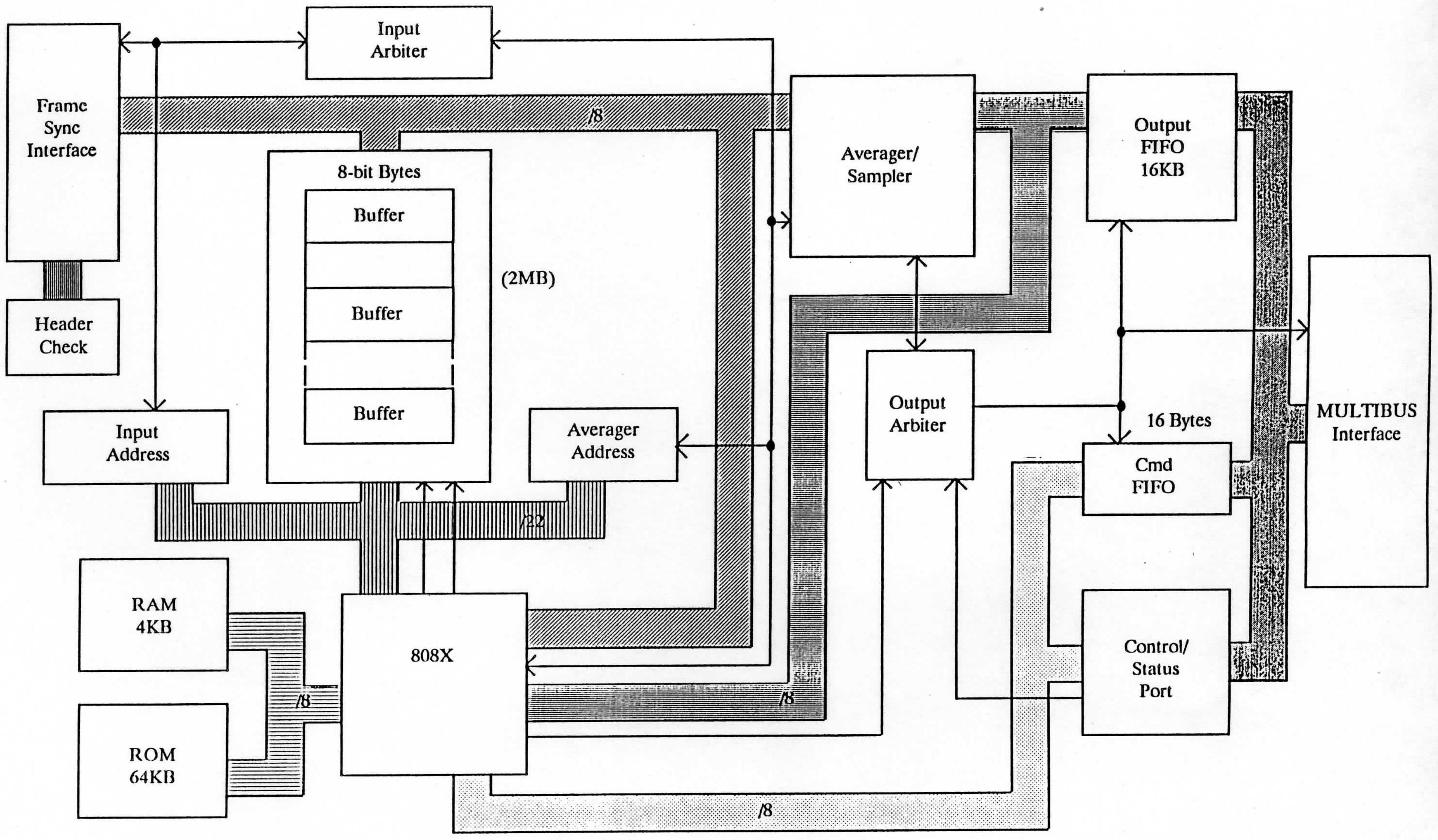


Figure 4.6.3: GVAR Block Diagram (Imager or Block 11)

convert 6,8,10 bit data to 8,8,16 bit words respectively. The estimate for an adequate amount of buffering is about 512KB.

The hardware/firmware makes very few decisions about the quality or usefulness of the data - the ingest handler has this responsibility.

4.7 Ingest Handler

The GVAR ingest handler is responsible for all decisions regarding image start/stop, associating ingest requests with the current satellite signal, and now all scan-to-scan decisions.

The organization of the GVAR ingest handler will be significantly different from the AAA ingest handler. More emphasis will be placed on performance, timely linkage of requests to satellite signal mode, timely linkages to the event scheduler, servicing more requests at once, and providing better status visibility to other system processes.

The mechanism used for fast linkages of requests to newly received signals is to have a separate control thread (sort of a subtask) keep the incore request table up-to-date (local to the ingest handler). This table will be updated whenever a semaphore in the bulletin board (set by the ingest scheduler commands) indicates a change in the ingest schedule. The ingest control thread will use the incore tables. The ingest hardware has adequate buffering to allow the ingest control thread to make the decision about applicability of the data, and then request the portion of the data desired.

Figures 4.7A and 4.7B illustrate the basic control flow of the GVAR ingest handler in active and start-up mode. Start-up mode describes linking requests to incoming signal, active mode describes the subsequent ingestion of the desired data.

4.8 Event Scheduler

The event scheduler will undergo major changes for GVAR. This will be necessary in order to reduce system resources usage and to improve timeliness of response by post-processes.

First, a memory-based event scheduler (called EVX) will be implemented. Then, the performance critical events will be moved to EVX one-at-a-time. As each is passed on to operations, the corresponding function will be 'turned off' in the old event scheduler. The fixed functions which are very time critical will not be controlled with flexible 'expressions', but rather controlled by a set of control flags. All critical information needed for decisions are kept in memory by EVX, thus providing a tremendous improvement in performance over the old event scheduler.

All events critical to the operation of the GVAR ingestor will be transferred by ingestor testing, but some of the flexibility of the current event scheduler will not.

Also, future plans call for EVX to be connected to a general 'blackboard evaluator' routine to provide enhanced flexibility, but this also will not be completed before delivering the GVAR ingestor. Thus, the GVAR ingestor will be delivered with both the old and new event schedulers. The new event scheduler (EVX) will perform all the performance critical functions, but the old event scheduler will still be

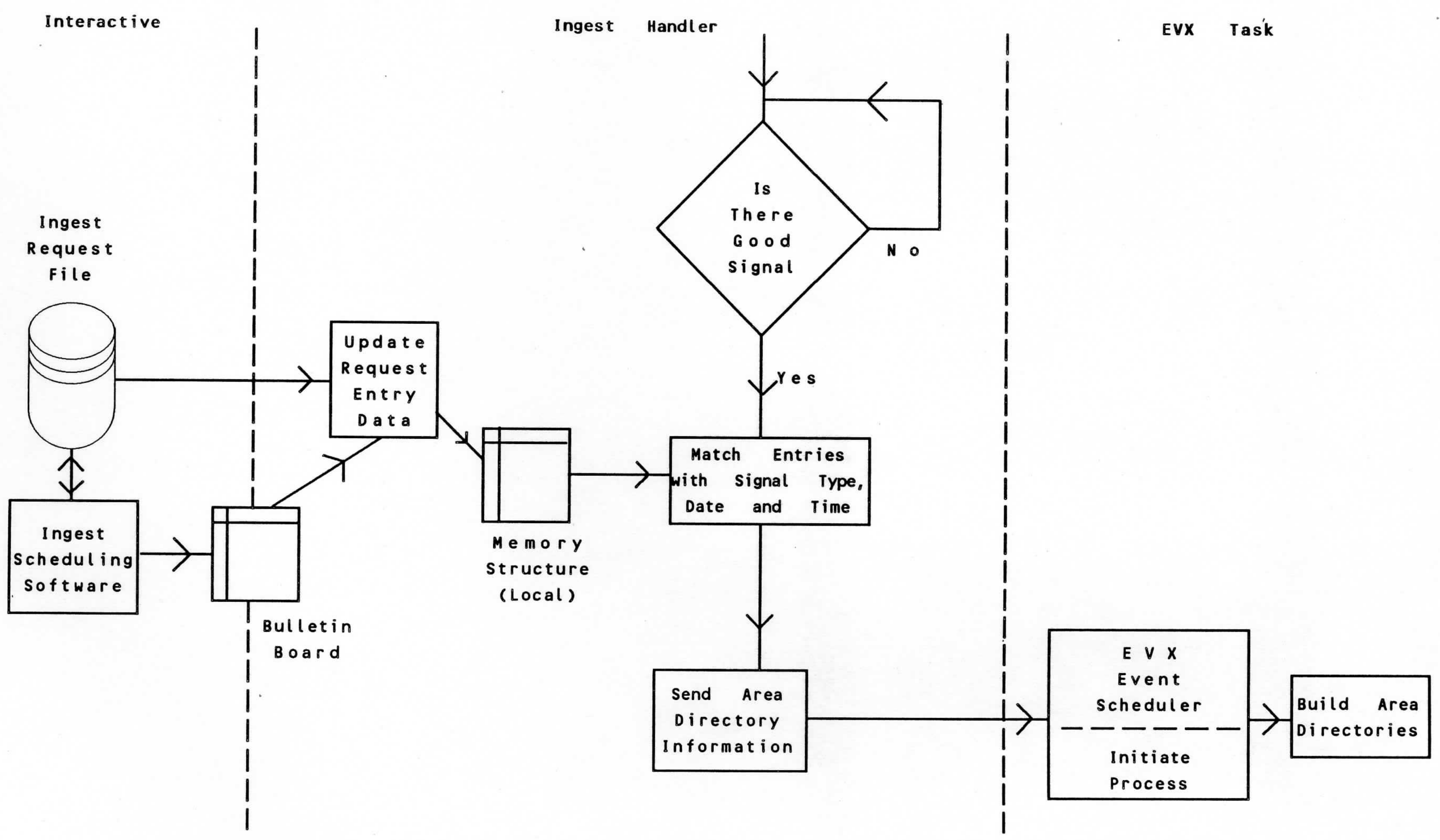
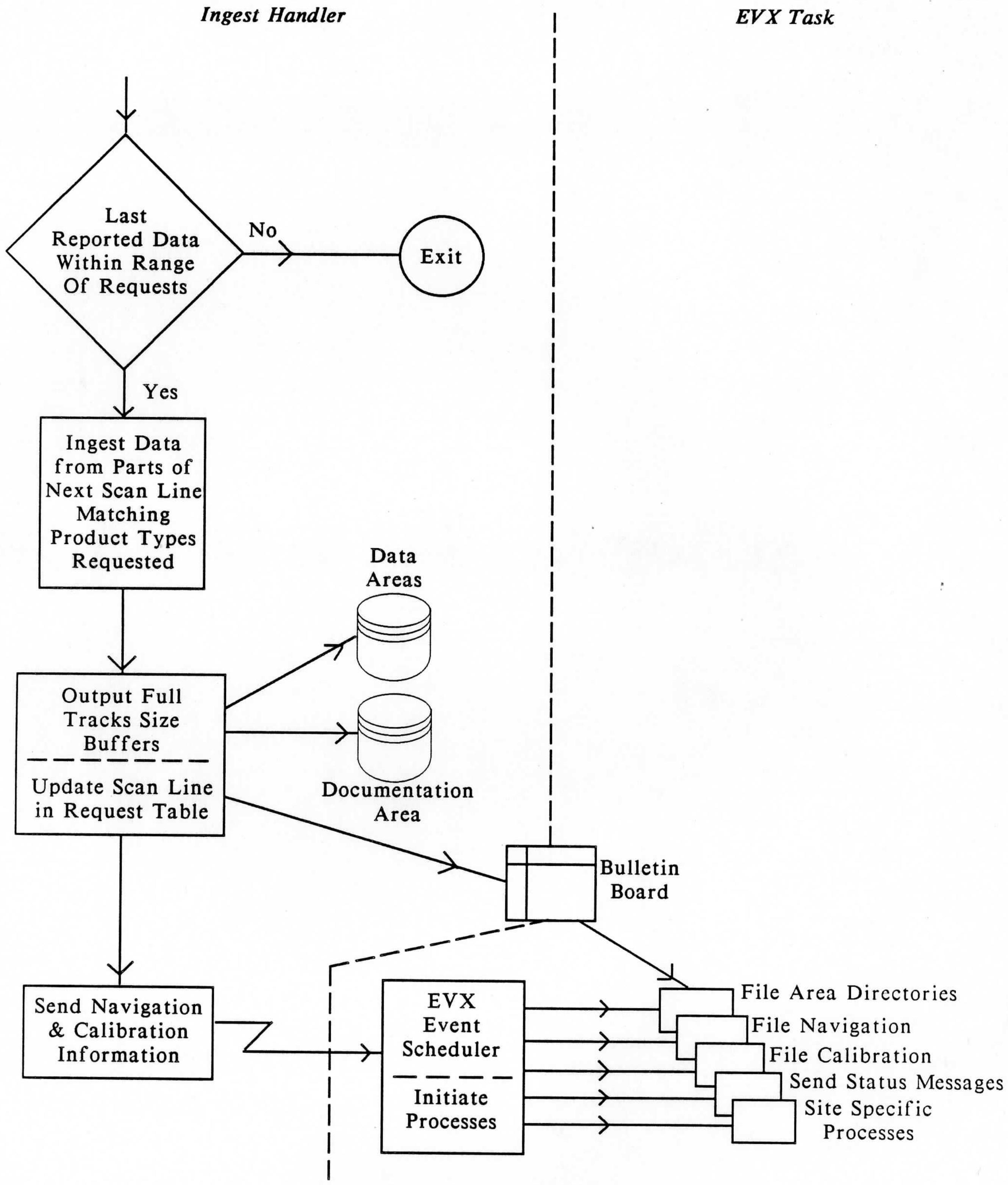


Figure 4.7A: Imager Control Flow [Image Start-up]



Ingest Handler
EVX Task
Last Reported Data Within Range Of Requests
No
Exit
Yes
Ingest Data from Parts of Next Scan Line Matching Product Types Requested
Data Areas
Output Full Tracks Size Buffers
Update Scan Line in Request Table
Documentation Area
Bulletin Board
Send Navigation & Calibration Information
EVX Event Scheduler
Initiate Processes
File Area Directories
File Navigation
File Calibration
Send Status Messages
Site Specific Processes

Figure 4.7B: Imager Control Flow [Ingesting Data]

useful for less performance critical functions, such as automatic image loading on user workstations.

4.9 Time Scheduler

The time scheduler will be used much as for AAA. The main difference is that the GVAR schedule is expected to be less predictable. Thus, it will be less useful to use this schedule to initiate post-processes. The event scheduler (old or new) is expected to be used more heavily for post-process initiation.

4.10 Database Access

GVAR data access is expected to be the same as for AAA. This is actually implicit in the requirements, since AAA and GVAR must be able to co-exist on the same interim system, and none of the interim systems can stand to have two complete sets of applications running - one for AAA and one for GVAR. Some extensions to ease the access for applications will be done, e.g., new access subroutines for Documentation and Grid bits. For a more detailed interface description see 'Applications Software Access to the GOES Real-time Database' (reference 6).

4.11 System Constraints

The system constraints for GVAR are the same as for AAA. The design relaxes the instantaneous response requirements on the ingestor channel, but the denser data in the GVAR signal means a higher percentage channel utilization (see Section 6.0). Total throughput on the channel is still critical. Datastreaming will allow two or more GVAR ingestors to comfortably coexist on the same channel. Tolerance of non-SSEC hardware on the same channel can not be guaranteed, but the likelihood of peaceful coexistence will be higher.

The GVAR ingestor will be able to coexist with the AAA ingestor, given sufficient system resources (see Section 6.0).

5.0 AAA INGESTOR PERFORMANCE MEASUREMENT

This study also funded a measurement of the AAA ingestor performance. See Appendix A for a more details.

The reason to measure AAA ingestor performance was to:

- *improve our understanding of AAA design strengths and weaknesses*

No comprehensive measurement of the AAA ingestor resource consumption had ever been done. A good understanding of the current AAA is necessary in order to extrapolate to the GVAR ingestor.

- *reasonably predict the following requirements for the GVAR ingestor*

  - CPU cycles consumed
  - dedicated mainframe memory
  - channel bandwidth
  - disc space

as compared to AAA. Since the GVAR ingestor is to be based on the AAA design philosophy, it should be possible to make some reasonable estimates of resource consumption.

A basic assumption in these measurements is that measuring the ingest hardware, ingest handler software, specific data structures for AAA and GVAR, and protocol is sufficient. The rest of the system is, from a performance standpoint, the same for AAA and GVAR, and therefore does not need to be measured.

A second assumption made in doing these measurements is that rather than trying to measure the resource consumption very precisely, calculating the upper bounds on resource consumption for a representative set of cases was sufficient. This could lead to resource estimates slightly on the high side, but would insure that no shortage of resources would result from use of these measurements.

5.1 System Resources

The system resources measured were:

- *CPU cycles consumed by the ingest handler*

These are stated both as CPU seconds on a 4381/P14 and as System Resource Units (SRUs). A standard IBM supplied System Resource Measurement (SRM) constant was used to allow intercomparisons between different IBM models. The SRM method of comparison was deemed superior to the other methods IBM uses because it represents an 'instantaneous' performance comparison, which is appropriate when comparing high-priority tasks at this level. The SRM method would not be appropriate if comparing system throughput. See Figure 5.1.

- *Mainframe memory required for the ingest handler*

The number of most concern is the amount of buffer space required in the ingest handler. The size of the code is of less concern, since it is expected to be essentially the same for the AAA and GVAR ingest handlers.

- *Disc space requirements for representative cases*

These are expressed in bytes, and do not account for any block size limitations in the disc allocation system. That is, space is allocated in groups of tracks call granules for the area structure in McIDAS. This means that more space is actually consumed on the disc than the number of bytes listed in the table. The size of the granules varies from system to system. There are a few tracks wasted per area on the average.

5.1.1 Measurement method used

Care was taken to get repeatable measurements. This meant measuring several cases until a model of resource use was put together, and then running several more until the model predicted the measured values. The SSEC archive/playback was used repeatedly over several data sets to keep data variability from clouding the measurements. (The SSEC archive/playback unit is capable of recording GOES data as output by the frame sync (GOES-BUS format) and later playing it back. The SSEC AAA ingestor can ingest data from either real-time or the archive/playback unit, as both appear to it as being in GOES-BUS format.)

Processor Model	Service Units Per Second of Task or SRB Execution Time	Seconds of Task or SRB Execution Time Per Service Unit	Number of Processors
4381-11	90.5	0.01105	1
4381-12	181.1	0.00552	1
4381-13	235.3	0.00425	1
4381-14	200.0	0.00500	2
4381-21	135.48	0.007381	1
4381-22	164.71	0.006071	1
4381-23	304.35	0.003286	1
4381-24	258.70	0.003866	2
3090-150	451.6	0.00221	1
3090-180	750.0	0.00133	1

Figure 5.1: Normalizing IBM Model Performance

The basic ground rules used when measuring were:

- *The ingest hardware being tested was isolated on the channel.*

To prevent multiple units on the same channel from clouding the measurements, the majority of the tests were done with only one active set of ingest hardware on the channel. Enough tests were later done to quantify the amount of interference between multiple units on the same channel.

- *The ingest handler being tested was isolated in its own task.*

It was not possible to remove the real-time ingestors while testing, but they were separated from measured values by isolating the test ingest handler in its own task.

- *SMF (System Measurement Facility) data was used to collect the basic measurements:*

  - CPU consumed
  - Area I/Os (to disc)
  - Channel I/Os (to ingest hardware).

- *One case run per ingest handler execution.*

In order for the SMF measurements to be unambiguous, the ingest handler was started before and stopped after each case.

- *Modified versions of the ingest handler used for special cases.*

Special cases were run to isolate CPU time for various parts of the ingestion process - disc I/Os, channel I/Os, etc. The ingest handler was modified to provide measurements of portions of the ingest process: CPU setup time for disc I/O or ingest I/O, etc.

- *The measurements were performed on a 4381/P14 at SSEC.* The system was operational during the measurements. There were up to 60 workstations and five real-time ingestors active. This is not likely to affect the measurements significantly, since the test ingestor was isolated from the real-time ingestors, and all activity from the workstations was at lower priority than the test ingestor. In fact, measuring on an active system should give a truer picture of actual performance.

5.1.2 <u>A representative case</u>

The results of the AAA System Resource study can best be illustrated by looking at a particular case. The case we chose is the AAA VDUC request case. This case was chosen because it is fairly demanding and is actually in use. See Table 5.1.2.

The column titled 'Sector' identifies the geographic coverage of this request.

The column titled 'Type' identifies the product to be created. 'VIS' is for visible sensor, 'IR' is for one particular IR band (usually band 8), and 'IR4' is for all IR bands sent (also called 'MULT').

Sector	Type	#pix/line	#lines	resolution	bytes/line	totbytes
USA	VIS	6720	3216	1	6724	21624384
N.HEMI	VIS	3800	1700	4	3804	6466800
GLOBE	VIS	1900	1621	8	1904	3086384
GLOBE	VIS	952	810	16	956	774360
N.HEMI	IR	3552	1700	4	7736	13151200
GLOBE	IR4	1900	1621	8	15836	25670156
GLOBE	IR4	952	810	16	8252	6684120
			TOTAL BYTES PER IMAGE			77,457,404

Table 5.1.2: AAA VDUC REQUEST CASE DESCRIPTION

The column titled '#pix/line' is the number of image pixels per line of data. VIS has 8 lines per scan, IR varies with the sensor and requested resolution.

The column titled '#lines' identifies the number of image lines of data to store on disc.

The column titled 'resolution' is the resolution of the data to be stored on disc. In the case of GOES this may be interpreted as a pixels coverage in square kilometers.

The column titled 'bytes/line' is the total number of bytes for each line of data. This includes overhead, such as the common documentation on the front of every line of IR. Note that the IR data is stored two bytes per pixel.

The column titled 'totbytes' just calculates the total byte storage requirements for one image of this size.

The 'total bytes per image' value indicates how much disc must be reserved for each image the satellite sends. The ingest scheduler allows the requestor to set up a different number of areas for each of the requests, in order to optimize the usage of the available disc space. It is generally true that the higher the resolution, the larger the resultant sector, and the shorter the time interval for which it is useful.

5.1.3 <u>The results</u>

The resultant values from running this case, and many subsets of this case, were consolidated into Figure 5.1.3.

The area writes section identifies the number of disc writes over the course of a 1620 scan test image, and the CPU cost broken out by product type.

The GPCI requests section identifies the number of I/O command chains to the ingest hardware, and the CPU cost broken out by product type.

The overhead section identifies a CPU cost which did not vary with the requests being serviced.

The Buffer Space section identifies the amount of memory needed by the ingest handler to process these requests concurrently. This figure does not include the ingest handler code size (GVAR and AAA should be basically the same size).

The total seconds of CPU item identifies the load on the IBM processor for this case over the entire image. Note that 1620 scans is about 16.2 minutes wall clock time. Therefore, 32.84 CPU seconds represents about 3.4% of one processor of a 4381/P14.

5.2 <u>Channel Utilization</u>

The other major measurement effort was channel utilization; how much of the bandwidth of the channel is consumed by one AAA ingestor.

AAA

Area Writes	Num. of Writes	Write Cost	Total Cost
VIS	2077	.0028	5.82
IR	855	.0028	2.39
MULT (IR4)	1600	.0028	4.48
DOC	-	.0028	-
Total	4582	.0028	12.69

GPCI Requests	Num. of Commands	Command Cost	Total Cost
VIS	7854	.00013	1.02
IR	1913	.00013	0.25
MULT (IR4)	3016	.00013	2.14
DOC	4125	.00013	0.54
Total	16912	.00013	3.95

	Num. of Scans	Cost per Scan	Total Cost
Overhead	1620	.01	16.20
Total Secs. CPU			32.84
Total SRU			6568

Buffer Space
15400 Bytes/Track — 23 Tracks (354200 Bytes)

Figure 5.1.3: AAA Performance - VDUC Case

5.2.1 Measurement method used

A command chain can be decomposed into basic subcommands:

- WAIT, IDLE, SYNCH subcommands

These subcommands all are variations of the 'wait until data arrives - and then continue' command. The channel time consumed is virtually the same for any of them. A few were measured and found to consume the same amount of channel bandwidth regardless of the case being tested.

- WRITE/READ pair.

These two need to be considered as coupled together, and the amount of channel bandwidth consumed varies with the case being used. Further decomposition is needed to provide accurate measurements.

Many of these measurements were made on the bench using an IBM channel emulator, and later verified on the IBM channel. Some of these measurements could only be made on the IBM channel. All of these measurements were made several times each over a range of test cases so accurate predictions would be possible.

5.2.2 Results

Figure 5.2.2 illustrates the findings for the WRITE/READ pairs. The other commands basically each consume about 15 microseconds (T1 in the illustration).

5.3 Conclusions

These measurements were used to create the following generalizations:

- WRITE/READ pairs consume the following amount of the channel bandwidth (see Figure 5.2.2):

  T2+T4+T5+T8

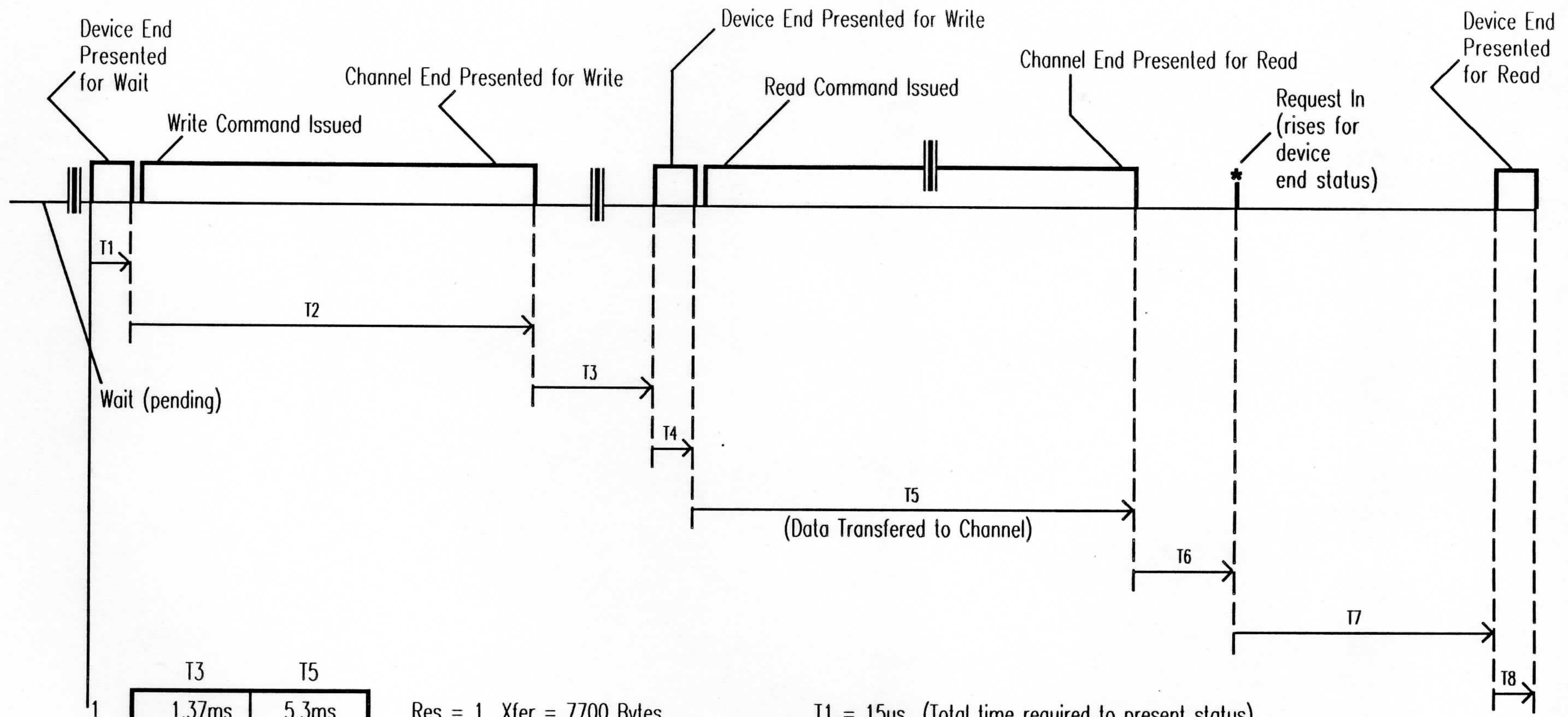
- WRITE/READ pairs consume the following amount of real-time, assuming data is available instantly:

  T2+T3+T4+T5+T6+T7+T8

- The equations to describe the three critical components of the system resource measurements are illustrated in Figure 5.3.

6.0 GVAR INGESTOR PERFORMANCE PROJECTIONS

The GVAR ingestor performance is predicted by projecting the equations derived for AAA to represent GVAR. Given the implementation scenario described in the earlier parts of this document, the AAA Channel I/O equations become those in Figure 6 for GVAR. The Buffer Space and Area Writes equations remain unchanged for GVAR.


Device End Presented for Wait
Write Command Issued
Channel End Presented for Write
Device End Presented for Write
Read Command Issued
Channel End Presented for Read
Request In (rises for device end status)
Device End Presented for Read
Wait (pending)
T1
T2
T3
T4
T5
(Data Transfered to Channel)
T6
T7
T8

	T3	T5	
1	1.37ms	5.3ms	Res = 1 Xfer = 7700 Bytes
2	0.91ms	2.1ms	Res = 1 Xfer = 3080 Bytes
3	0.68ms	1.1ms	Res = 1 Xfer = 1540 Bytes
4	0.68ms	0.7ms	Res = 1 Xfer = 768 Bytes
5	3.21ms	2.2ms	Res = 4 Xfer = 3080 Bytes
6	1.83ms	1.1ms	Res = 4 Xfer = 1540 Bytes
7	1.14ms	0.7ms	Res = 4 Xfer = 768 Bytes
8	2.29ms	1.1ms	Res = 8 Xfer = 1540 Bytes
9	1.37ms	0.7ms	Res = 8 Xfer = 768 Bytes

T1 = 15us (Total time required to present status)
T2 = 145us (Total time to issue write and pass parameters)
T4 = 15us
T6 = 35us (Total time required to finish up read)
T7 = 104us (Total time required for channel to respond to GPCI)
T8 = 15us

Figure 5.2.2: Write/Read Pair Measurement Results

Buffer Space, In Tracks, Per Request
AAA & GVAR

$$\text{Number of Tracks} = \frac{\dfrac{2 * \text{Length of Ingested Line} * 8}{\text{Resolution of Request}} + 2 * \text{Track Size} - 1}{\text{Track Size}}$$

Number of Area Writes, Per Request
AAA & GVAR

$$\text{Number of Writes} = \frac{\text{Number of Lines} * \text{Length of Ingested Line} + \text{Track Size} - 1}{\text{Track Size}}$$

Initiated I/O's (Write/Read Pairs) Per Request
AAA

$$\underset{\text{(number of lines ingested)}}{A} = \frac{\text{Number of Requested Scans} * 8}{\text{Resolution of Request}}$$

$$\underset{\text{(number of ingests/line)}}{B} = 1 + \frac{\text{Length of Ingested Line} + 16383}{16384} + \frac{\text{Length of Ingested Line}}{\text{Length of Request Buffer}}$$

$$\text{Number of AAA I/O Pairs} = A * B$$

Figure 5.3: AAA Critical Resource Equations

Initiated I/O's (Write/Read Pairs) Per Request
GVAR Data Pairs

$$\underset{\text{(number of lines ingested)}}{A} = \frac{\text{Number of Requested Scans} * 8}{\text{Resolution of Request}}$$

$$\underset{\text{(number of ingests/line)}}{B} = \text{Number of 16K Sub-buffers} + \frac{\text{Total Buffer Length}}{\text{Length of Request Buffer}}$$

$$\text{Number of GVAR I/O Pairs} = A * B$$

In addition, for each image, a documentation area is kept.

Initiated I/O's (Write/Read Pairs) Per Request
GVAR Doc

$$\text{Number of GVAR Doc Commands} = \text{Number of Scanlines} + \frac{\text{Number of Scanlines} + 4}{5}$$

Figure 6: GVAR Critical Resource Equations

6.1 System Resource Projections

6.1.1 VDUC AAA equivalent case

This case is the VDUC AAA case extrapolated for GVAR. The number of lines has gone from 1620 to 1418 (7/8 of AAA), and the number of pixels per line has been increased by adding 25% to the AAA length. These changes reflect the sensor size and scan pattern differences for GVAR, and assume equivalent geographic coverage. All 4 available IR bands are assumed to be desired. See Table 6.1.1.

Figure 6.1.1 illustrates the projected values in the same format as the earlier AAA example.

6.1.2 CDDF case

This case is the likely case for the CDDF system, should it be implemented. Again, all sizes have been adjusted for the GVAR sensors and scan patterns. See Table 6.1.2.

Figure 6.1.2 illustrates the projected values.

6.1.3 Full resolution, full sector case

This case is required by the GVAR Ingestor Functional Requirements document. See Table 6.1.3.

Figure 6.1.3 illustrates the projected values.

6.2 Channel resource projections

Figure 6.2 illustrates all three of the GVAR cases. The bar graph is meant to allow comparisons of performance across the various satellite scan widths.

6.3 Conclusions

The CPU consumption of the various cases across a range of IBM mainframes is illustrated in Figure 6.3. IBM supplied SRM constants were used to create this intercomparison.
Insert for section 6.3 of GVAR Top-Level Design Report 11-28-88

Notice how CPU consumption increases for GVAR. If the GVAR ingestor will be used in the same manner as the VDUC ingestor, the increases for the Imager data can be expected to be on the order of 110 to 115% of the AAA CPU consumption. The additional CPU consumption by the rest of the system because of the increase in data volume and increase in precision will add another few percentage points. Our best guess is to count on a 20% increase in CPU consumption to do about the same job as VDUC does now (plus or minus about 10% of AAA). Also notice that for the ingestor itself, this is still less than a 1% increase per ingestor. Result: plan on a 20% increase in CPU consumption to handle GVAR.

The disc space requirements for GVAR are a bigger factor. The GVAR disc space requirements are about 220% of the AAA requirements. Note that the scenario used for the GVAR VDUC case makes some assumptions about the resolutions of the

These values assume a normal full globe to be about 17.4 degrees wide by 17 degrees high. The actual values are rounded up to insure coverage. (19000 visible full res pixels by 1350 scan lines). It is assumed that all four IR bands are desired.

Sector	Type	#pix/line	#lines	resolution	bytes/line	totbytes
USA	VIS	8400	2814	1	16868	47466552
N.HEMI	VIS	4750	1488	4	9568	14237184
GLOBE	VIS	2376	1350	8	4820	6507000
GLOBE	VIS	1190	675	16	2448	1652400
N.HEMI	1 IR	4750	1488	4	9568	14237184
GLOBE	1 IR	2376	1350	8	4820	6507000
GLOBE	1 IR	1190	675	16	2448	1652400
N.HEMI	2 IR	4750	1488	4	9568	14237184
GLOBE	2 IR	2376	1350	8	4820	6507000
GLOBE	2 IR	1190	675	16	2448	1652400
N.HEMI	3 IR	4750	1488	4	9568	14237184
GLOBE	3 IR	2376	1350	8	4820	6507000
GLOBE	3 IR	1190	675	16	2448	1652400
N.HEMI	4 IR	4750	1488	4	9568	14237184
GLOBE	4 IR	2376	1350	8	4820	6507000
GLOBE	4 IR	1190	675	16	2448	1652400
DOC	DOC	----	1350	1	8076	10902600
			TOTAL BYTES PER IMAGE			170,352,072

Table 6.1.1: GVAR VDUC REQUEST CASE DESCRIPTION

GVAR

Area Writes	IMAGER Num. of Writes	IMAGER Write Cost	IMAGER Total Cost	BLOCK 11 Num. of Writes	BLOCK 11 Write Cost	BLOCK 11 Total Cost
VIS	4539	.002	9.08			
IR	6532	.002	13.06			
DOC	708	.002	1.42			
BLOCK 11	-	-	-	708	.0020	1.42
Total	11779	.002	23.56	708	.0020	1.42

GPCI Requests	Num. of Commands	Command Cost	Total Cost	Num. of Commands	Command Cost	Total Cost
VIS	9799	.00013	1.27			
IR	16036	.00013	2.08			
DOC	0	.00013	0			
BLOCK 11	-	-	-	1620	.00013	.21
Total	25835	.00013	3.35	1620	.00013	.21

	Num. of Scans	Cost per Scan	Total Cost	Num. of Scans	Cost per Scan	Total Cost
Overhead	1350	.01	13.50	1350	.007	9.45
Total Secs.			40.41			11.08
Total SRM			8082			2216
Buffer Space 15400 Bytes/Track	62 Tracks (954800 Bytes)			7 Tracks (107800 Bytes)		

Figure 6.1.1: GVAR Performance - VDUC Case

These values assume a normal full globe to be about 17.4 degrees wide by 17 degrees high. The actual values are rounded up to insure coverage. (19000 visible full res pixels by 1350 scan lines).

Sector	Type	#pix/line	#lines	resolution	bytes/line	totbytes
N.HEMI	VIS	19000	6000	1	38068	228408000
GLOBE	VIS	9500	5400	2	19068	102967200
GLOBE	VIS	4750	2700	4	9568	25833600
N.HEMI	1 IR	4750	2700	4	9568	25833600
GLOBE	1 IR	2376	1350	8	4820	6507000
N.HEMI	2 IR	4750	2700	4	9568	25833600
GLOBE	2 IR	2376	1350	8	4820	6507000
DOC	DOC	----	1350	1	8076	10902600
			TOTAL BYTES PER IMAGE			432,814,600

Table 6.1.2: GVAR CDDF REQUEST CASE DESCRIPTION

GVAR

	IMAGER			BLOCK 11		
Area Writes	Num. of Writes	Write Cost	Total Cost	Num. of Writes	Write Cost	Total Cost
VIS	23395	.002	46.79			
IR	4202	.002	8.40			
DOC	708	.002	1.42			
BLOCK 11	-	-	-	708	.0020	1.42
Total	28305	.002	56.61	708	.0020	1.42
GPCI Requests	Num. of Commands	Command Cost	Total Cost	Num. of Commands	Command Cost	Total Cost
VIS	32889	.00013	4.28			
IR	9361	.00013	1.22			
DOC	0	.00013	0			
BLOCK 11	-	-	-	1620	.00013	.21
Total	42250	.00013	5.50	1620	.00013	.21
	Num. of Scans	Cost per Scan	Total Cost	Num. of Scans	Cost per Scan	Total Cost
Overhead	1350	.01	13.50	1350	.007	9.45
Total Secs.			75.61			11.08
Total SRM			15122			2216
Buffer Space 15400 Bytes/Track	71 Tracks (1093400 Bytes)			7 Tracks (107800 Bytes)		

Figure 6.1.2: GVAR Performance - CDDF Case

These values assume a normal full globe to be about 17.4 degrees wide by 17 degrees high. The actual values are rounded up to insure coverage. (19000 visible full res pixels by 1350 scan lines).

Sector	Type	#pix/line	#lines	resolution	bytes/line	totbytes
GLOBE	VIS	19000	10800	1	38068	411134400
GLOBE	1 IR	4750	2700	4	9568	25833600
GLOBE	2 IR	4750	2700	4	9568	25833600
GLOBE	3 IR	4750	2700	4	9568	25833600
GLOBE	4 IR	4750	2700	4	9568	25833600
DOC	DOC	----	1350	1	8076	10902600
			TOTAL BYTES PER IMAGE			525,368,400

Table 6.1.3: GVAR FULL RESOLUTION/FULL SECTOR CASE DESCRIPTION

GVAR

Area Writes	IMAGER Num. of Writes	IMAGER Write Cost	IMAGER Total Cost	BLOCK 11 Num. of Writes	BLOCK 11 Write Cost	BLOCK 11 Total Cost
VIS	26797	.002	53.59			
IR	6712	.002	13.42			
DOC	708	.002	1.42			
BLOCK 11	-	-	-	708	.0020	1.42
Total	34217	.002	68.43	708	.0020	1.42

GPCI Requests	Num. of Commands	Command Cost	Total Cost	Num. of Commands	Command Cost	Total Cost
VIS	33051	.00013	4.30			
IR	12478	.00013	1.62			
DOC	0	.00013	0			
BLOCK 11	-	-	-	1620	.00013	.21
Total	45529	.00013	5.92	1620	.00013	.21

	Num. of Scans	Cost per Scan	Total Cost	Num. of Scans	Cost per Scan	Total Cost
Overhead	1350	.01	13.50	1350	.007	9.45
Total Secs.			87.85			11.08
Total SRM			17570			2216
Buffer Space 15400 Bytes/Track	60 Tracks (924000 Bytes)			7 Tracks (107800 Bytes)		

Figure 6.1.3: GVAR Performance - Full Res./Full Sector Case

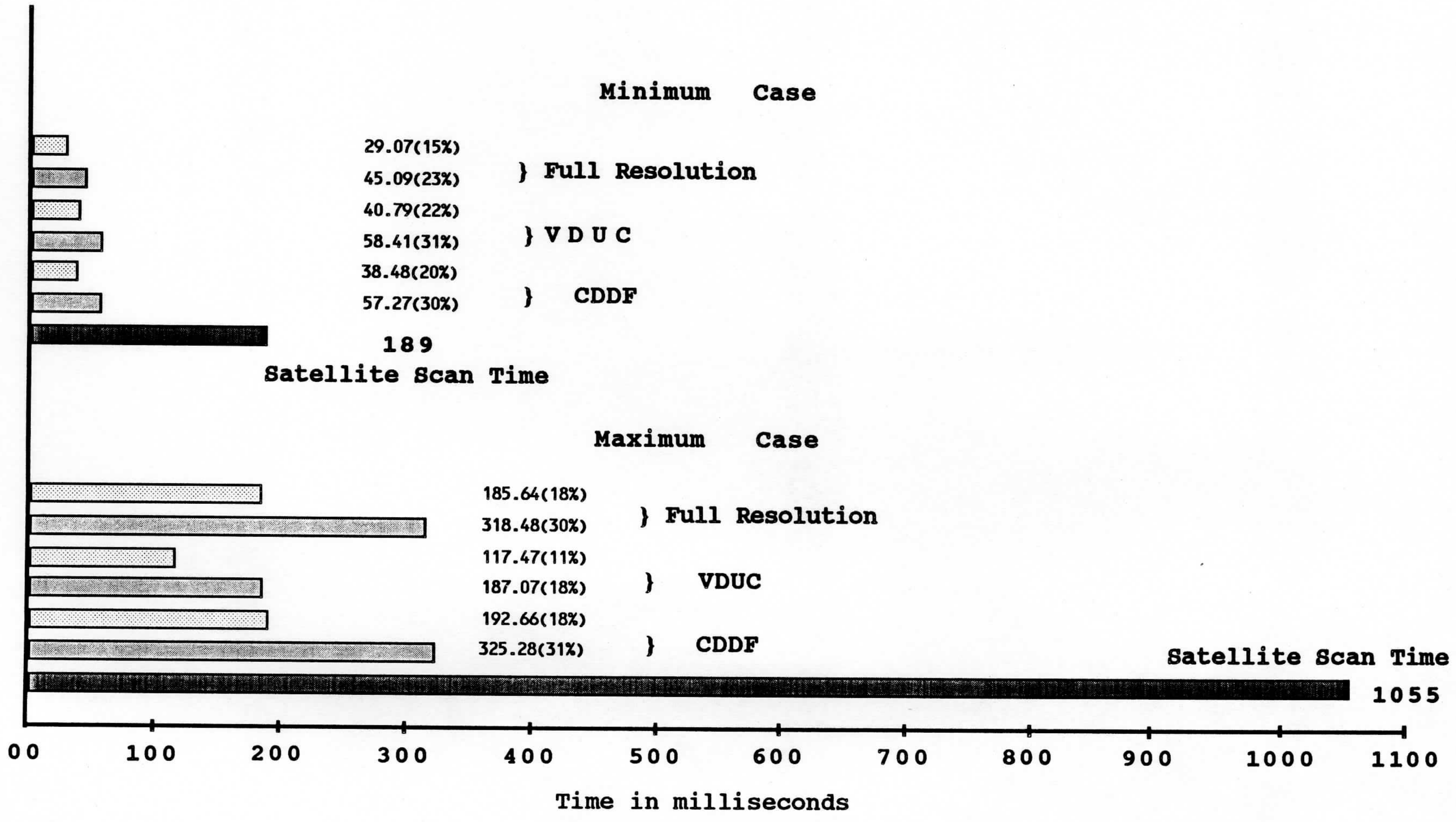


Figure 6.2: GVAR Channel Busy Projections

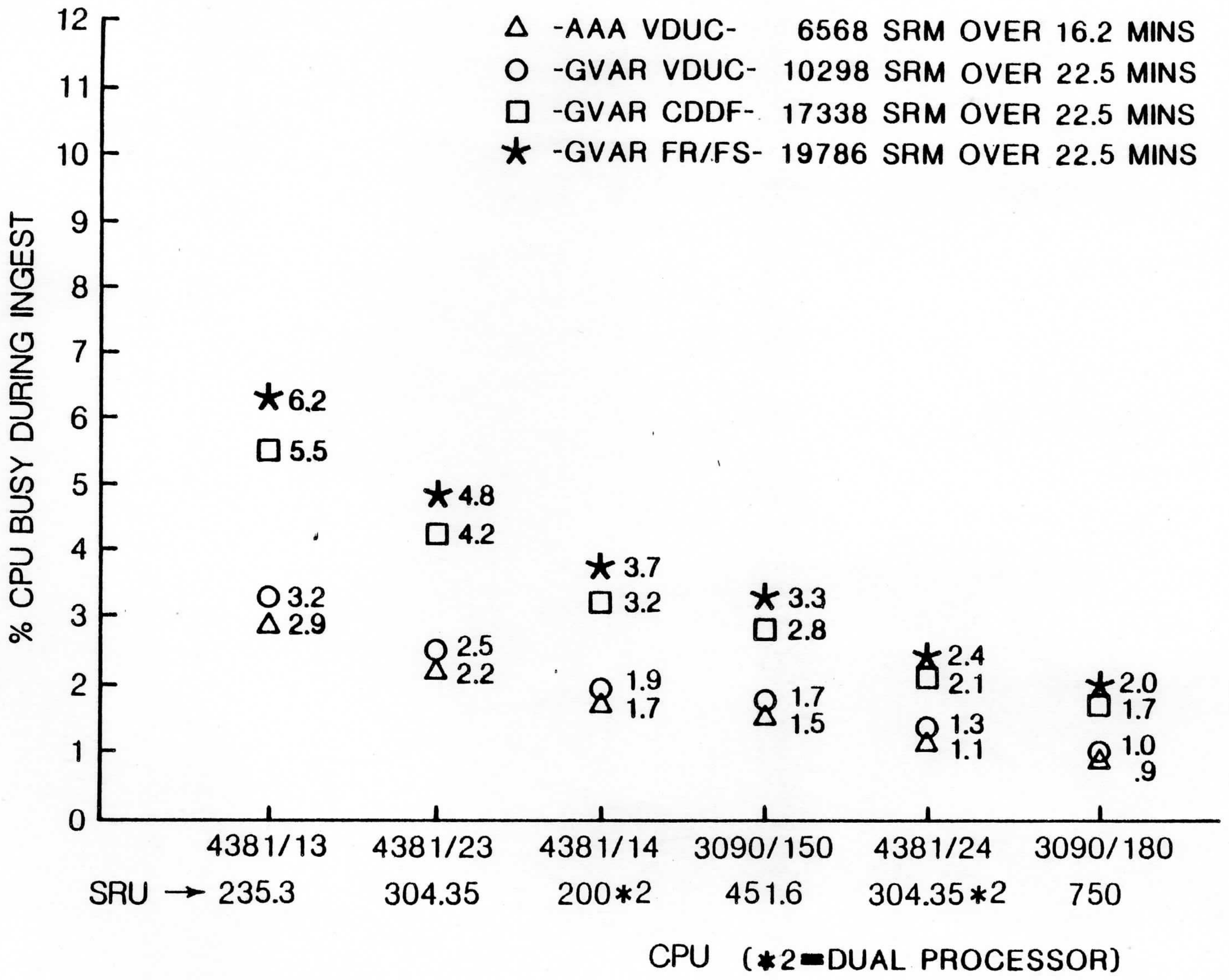


FIGURE 6.3: CROSS-SYSTEM CPU CONSUMPTION COMPARISON

IR bands which may not actually be what happens. However, no provision is made for setting aside space for the Watch and Warning modes either. The net effect is likely to be a draw. Result: plan on much more disc, at least twice what is being used for AAA now.

The memory requirements for the GVAR ingestor is significantly more than for the AAA ingestor. This is due to the increased number of pixels in a line and the increased number of IR bands to be decommutated. The GVAR ingestors, if no priority frame data is desired, will require about 300% of the AAA requirement, e.g., about 1.06MB for GVAR vs. .35MB for AAA.

If data from the priority frame is desired, the GVAR ingestors will require about 350% of the AAA requirement, e.g., about 1.22MB for GVAR vs. .35MB for AAA.

The channel bandwidth required for GVAR is substantially higher than for AAA. Currently, two AAA ingestors can share a channel with complete confidence there is adequate bandwidth. For GVAR, there may not be adequate bandwidth to support two ingestors on a single channel in channel interlock mode (now used by the AAA version of GPCI). Result: implementing Datastreaming will increase the bandwidth of the existing channels to the point where channel bandwidth need not be a concern.

7.0 GVAR INGESTOR IMPLEMENTATION RECOMMENDATIONS

The discussion to this point has concentrated on the technical aspects of how to build a GVAR ingestor, without making any recommendations. This section makes the recommendations.

7.1 Imager and Block 11 Ingest Cards

The Imager and Block 11 ingest cards are required to get the required information into the database.

The methods outlined in the earlier chapters are recommended to insure the best overall system performance. This includes the software enhancements discussed in the earlier chapters. These changes are necessary to achieve the performance and functionality objectives outlined in the GVAR Ingestor Functional Requirements document.

7.2 Datastreaming

It is likely there will be a shortage of channel bandwidth when running two GVAR ingestors on the same channel at the same time. This is shown in the GVAR performance projections. The alternatives to expand the channel resources are:

- *add more channels.*

This will certainly work, but will require a substantial cost for each system to utilize this GVAR ingestor. In addition, not all systems have the option of adding channels.

Adding more channels, as the only solution, is *not* recommended.

- *improve the performance of the existing channels.*

This is possible. The current Multisourcerer uses channel interlock mode for communication with the IBM host on the channel. This mode is limited to about 1.5 MB/sec transfer rate. The measured maximum transfer rate of the Multisourcerer on an IBM channel is 1.4 MB/sec.

There is a higher speed mode of communication call Data Streaming. This method can achieve 3 MB/sec transfer rates on the same channel. This communication method is compatible with the channel interlock mode; devices of both types can coexist on the same channel.

Upgrading the Multisourcerer to be capable of the Data Streaming mode of operation is recommended.

This recommendation is made primarily as a judgement of risk. The likelihood of channel bandwidth shortages are believed to be significant, but this conclusion is not definite. If the datastreaming mode is not implemented, and later channel bandwidth problems arise, the only possible timely solution is to buy additional IBM channels and Multisourcerers, and in the meantime the system will not function acceptably. The development cost for datastreaming is about the cost of upgrading one system to have additional channels plus another Multisourcerer, and insures the channel bandwidth will be adequate for *all* systems with GVAR ingestors.

7.3 Archive Unit

SSEC builds an archive/playback unit capable of recording GOES-BUS data and playing it back later into the same AAA ingestor. Upgrading this unit to allow record and playback of GVAR data would make troubleshooting the real-time GVAR signal and/or ingestor problems very much simpler. However, since the GVAR ingestor is to be implemented before actual GVAR signal becomes available, the archive/playback would be of limited value initially. The quality of the ingestor will very much depend upon the quality of the simulator used during it's development.

During satellite bringup, and all during the operational life of GOES, the archive/playback unit proved invaluable in allowing detailed looks at problems which occurred, but were not diagnosed, in real-time. Having this capability for a new satellite type as different as GOES I-M will be at least as valuable.

Upgrading the archive/playback unit is not recommended initially.

However, some sort of archive/playback unit will be necessary shortly after launch of GOES-I. An archive/playback unit should be built as a separate effort from the GVAR ingestor effort.

7.4 Sounder Card

This card would provide no new function. It merely moves the decoding of the sounder blocks from software to hardware/firmware. Our rough estimate of savings is *at least* 2 hours of CPU on a 4381/M03 per satellite per day. This rough estimate was derived from the SPIE paper by Barton, et al. (reference 1). In addition, there would also be a savings in I/O, because the sounder blocks would not have to be read, decoded, and rewritten by sounder decoder software.

Building a sounder card is *not* recommended at this time.

It provides no new functionality, and the resources used may not turn out to be important. This card could be built later, if necessary. If a card is later deemed necessary, the systems will continue to function, albeit at a less than desired performance level, until the sounder card is completed. The best strategy is to wait and see if it is necessary, then decide.

8.0 SUMMARY

The AAA ingestor design has been described as a starting point for the GVAR ingestor design. The GVAR extensions and modifications have been described. The AAA ingest hardware and handler performance have been measured and described. Projections of GVAR performance have been made based on the AAA performance measurements and the described extensions and modifications for GVAR. Finally, recommendations concerning the GVAR ingestor implementation have been made.

GLOSSARY

A	- Mode A - Oldest form of GOES satellite signal format. Until May, 1986.
AA	- Mode AA - Experimental form of GOES satellite signal format used while experimental VAS instrument being tested. 1981 until March, 1986.
AAA	- Mode AAA - Current form of GOES satellite signal format. Since March, 1986. Combines data of mode A and AA forms.
application	- any task whose role is limited to reading/writing data from/to the database and communicating with an interactive or batch user.
Application	- one of the six Product Generation and Development applications defined in the GVAR Ingestor Functional Requirements document as critical to be supported by this GVAR ingestor.
area	- in McIDAS, a section of an image stored on disc.
bandwidth	- in this document means the capacity of a channel, number of bytes per second that can be transferred across a channel.
BDAM	- Basic Data Access Method - one of the simplest data structure access methods supported by IBM. Underlies many of the more complicated access methods supported by McIDAS and IBM.
calibration	- in McIDAS, the process of converting raw measurements into scientifically meaningful temperatures, radiances, or brightnesses.
CDDF	- Central Data Distribution Facility. The GOES-TAP distribution system for GVAR data.
CPU	- Central Processing Unit - the 'brain' of a computer system.
database	- a centralized place where all data resides.
DMA	- Direct Memory Access. Used in this document as the method by which data is transferred to and from IBM main memory. This process does not consume CPU cycles.
EVX	- the proposed new Event Scheduler for McIDAS.
EXCP	- the IBM supported macro which allows command chains to be executed on a channel, used with the McIDAS ingestors.
FIFO	- First In, First Out. Queue.
GMS	- Geosynchronous Meteorological Satellite. The Japanese Weather satellite similar to GOES.
GOES	- Geosynchronous Orbiting Environmental Satellite. The United States' stationary weather satellites.
GOES-BUS	- the standard format for GOES (and now GVAR) data between the frame synchronizer and the ingestor or archive unit..
GVAR	- Goes VARiable signal format.
ingestor	- a combination of hardware and software which places satellite data into a database in such a way that it is concurrently accessible to applications.
I/O	- Inputs/Outputs.
IOGEN	- The process of describing the configuration of an IBM mainframe system to MVS.
KB	- Kilobytes - thousands of bytes.
LW	- Long Word - the basic access method underlying all non-real-time data structures in McIDAS.
McIDAS	- Man computer Interactive Data Access Method.
MB	- Millibytes - millions of bytes.
METEOSAT	- The European Meteorological Satellite. Similar to GOES.

MVS	-	Multiple Virtual Systems. The IBM operating system McIDAS is based upon.
navigation	-	the process of locating each pixel precisely in earth coordinates (latitude/longitude), or predicting which pixel will cover any given latitude/longitude.
NESDIS	-	National Environmental Satellite Data Information Service (?). The branch of NOAA that chartered this study.
NOAA	-	National Oceanic and Atmospheric Administration (?)
OSD	-	Office of Systems Development - The department of NESDIS that chartered this study.
process	-	a task or subtask. Any independent program that performs a specific set of functions.
Pseudo-averaging	-	the process of averaging elements, but sampling lines, that the McIDAS ingestors use instead of tru averaging. True averaging is averaging elements and lines.
real-time	-	used here to mean as close to actual sensed time as possible (within the span of human tolerance). Refers to getting the data to the database, and then the user, as it is received.
Resolution	-	here used as reduction factor from full resolution, or the end coverage of a pixel of data after a reduction factor has been applied.
Sampling	-	the process of skipping or selecting data to reduce its resolution. As opposed to averaging.
Sector	-	a portion of an image. Resides in an area after it has been ingested.
Sectorization	-	the process of 'cutting out' just the pieces of the image desired, one or more sectors.
Signal sector	-	one physical block of the AAA or GVAR signal stream.
SMF	-	System Maintenance Facility. Used on IBM systems to measure resource consumption by processes.
SSEC	-	Space Science and Engineering Center. At UW.
TIROS-N	-	The U.S. polar orbiting weather satellite.
UW	-	University of Wisconsin.
VAS	-	Vertical Atmospheric Sounder - the instrument on GOES which sensed 12 IR channels and was experimentally tested using the AA signal format. It is now used operationally via signal mode AAA. Also sometimes used to refer to the program supporting research and operations of the VAS instrument.
VDUC	-	VAS Data Utilization System. Currently in the World Weather Building in Washington, D.C. Where the operational VAS retrievals are performed.

REFERENCES

1. Barton, Ted A., Eric W. Suomi, Joseph P. Rueden, 1987: Preprocessing: A Key to the Distributed Processing of Digital Meteorological Images. Reprint from SPIE Vol. 846--Digital Image Processing and Visual Communications Technologies in Meteorology, October 1987.

2. "GVAR Ingestor Functional Requirements," The MITRE Corporation, Civil Systems Division, 7525 Colshire Drive, McLean, VA 22102-3481, December 22, 1987, amended 7/88).

3. Ford Aerospace, Space Systems Division, Operations Ground Equipment (OGE), Interface Specification, DRL 504-02, WDL-_R110718, 6 October 1988. Chapter 3: GVAR Transmission Format. ISI-SP-36-002 OGE External Interface Control Document.

4. Space Science and Engineering Center, 1225 W. Dayton, University of Wisconsin, Madison, WI 53706: McIDAS Applications Programming Manual, February 1988.

5. NOAA/NESDIS, U.S. Dept. of Commerce: Operational VAS Mode AAA Format, SFP-002, October 1986, Version 2.3.

6 "Applications Software Access to the GOES Real-time Database," The MITRE Corporation, Civil Systems Division, 7525 Colshire Drive, McLean, VA 22102-3481, November 1988.

APPENDIX A

USING AND MEASURING AAA PERFORMANCE AS A MEANS OF ESTIMATING GVAR PERFORMANCE

The AAA ingestor (both hardware and software handler) can be used as a predictor of GVAR performance. Data flow and the sequence of operations are parallel, if not always the same in both ingestors. Distribution of CPU use, over the major activities (GPCI I/O and area I/O) can be analyzed in each case and the amount of I/O to GPCI and disk can be quantified. Once it is known what the various activities cost in terms of computer resources for AAA, we can calculate the resource cost for GVAR.

GVAR will differ in that it will know in advance what data is being sent, but this will not affect the number or average rate of I/O operations. Block 11 data will be sent to buffers that are handled and managed as the buffers in AAA are handled.

The fact sheet at the end of this appendix (Fig. A-1) contains measurements and results used in this study.

1.0 METHOD AND PROCEDURE

1.1 <u>Ingest Handler CPU Time is the Sum of the Following</u>:

- *Image Recognition and Event Setup:*

Time it takes to recognize that data is from a legitimate signal, desired by a user and to set up tables to parameterize the ingest.

- *Overhead:*

Time it takes a handler to set up commands to spin through an image, ingesting control information but no data.

- *Command time:*

Time the handler spends placing data commands to the channel and to the ingest hardware.

- *Disc time:*

Time it take to set up commands sending ingested data to disc areas.

1.2 <u>Overhead Time</u>

To get overhead time the handler was directed to receive signal but to ingest no data, all the way through an image.

1.3 <u>Command and Disc Time Calculated Indirectly</u>

To get command time and disc time, the handler was directed first to receive data and send it to disc, and then to receive data but not to send it to disk. When disk time is zero (no data to disk) the command time is the difference between the total time and the overhead. Once the command time is known, the disk time can be calculated by comparing the time of the command and area run with the time of the run with no disk I/O.

1.4 Multiband Requests Differ from Single-band Requests

It is assumed that channel commands are of two kinds: those that request multiband data (data all of whose elements for a given geographic location on a scan line are sent together) and those that request non-multiband data such as Vis, single band IR, or documentation. Most ingesting schemes do not interleave the bands; however, AAA does and it is important to remove these ingests from the calculations because GVAR will only ingest in the single band mode. The distinction is based on the amount of CPU effort it takes to place each read request in the command chain. More calculations are made to insert a multiband request, and correspondingly, there are more lines of code doing calculations for the multiband than for the single.

1.5 Single Band Time Calculated First

The Vis setup time was found first, by comparing the times for tests 63, 64, and 53, and then choosing a value. This value was used for all the doc and single band requests to calculate the amount of time not used for multiband requests. The multiband setup time was calculated by taking the time used by the multi-commands (total setup time - single band time) and dividing it by the number of multiband requests.

1.6 Multiband Times Not Needed

It should be noted that this study found results for the multiband setup time that differed, but were also within the experimental error. GVAR, however, will not use multiband requests, and it was decided not to design a test that would reduce the size of the error.

1.7 Image Recognition Time is Negligible

Several tests were run using 400, 600, 1200, 1500 and 1600 scan lines. The overhead in each test showed little variation from a ratio of 100 lines per second. If the image recognition time, which involves accessing at least three records from disk, files were non-trivial the overhead would have varied more than it did. The image recognition time was deemed negligible.

1.8 Requests were Varied, but Similar to VDUC's for the East Satellite

Test were run on a group of entries that included a variety of single band, multiband, and documentation requests. The entries were similar to the group used by VDUC for the East satellite. The entire group was tested, all the multi entries were tested as a group and all the Vis entries were tested as a group. Tests were also run on an individual set of Vis entries as a consistency check against the Vis in the VDUC group.

1.9 Performance Tool is SMP

Performance results were obtained from SMP, an IBM accounting tool.

The direct results included CPU time, in secs of complete run of each image, the number of command groups sent to the hardware, the number of disk area outputs and the number of seconds the handler was running.

1.10 Starting and Stopping the Tests

Tests began at the beginning of a test image and stopped immediately when the last line of the image was reached.

The tests for the individual Vis entries were run over 1200 scans and not 1600 as were the tests for the VDUC group entries. This does not seem to have affected the consistency of the results. (See the Fact Sheet.)

2.0 DIFFICULTIES

2.1 Round-off

SMP returned results in seconds. Results calculated by subtracting one result from another may be as small as one second and therefore they may have a large relative error.

2.2 Full Image Tests Necessary Before We Saw any Consistency in CPU Time

Initially, results were meager. In early tests not enough data was ingested to see a difference between overhead time and total time. It was necessary to ingest data from at least 1200 scans with at least 3600 elements per read before results became meaningful.

2.3 Area Track Placement Caused CPU Times to Vary

Total CPU time for each window varied considerably from test to test, sometimes by more than 50%. It was determined eventually that these differences depended on the location of each track (track index) in an area with respect to the beginning of the volume in which the track resides. More CPU time was necessary to write to areas with higher indices than to those with lower indices.

2.4 Optimal Track Placement Reduces Variability

The final results were obtained by forcing the placement of the area tracks as close to the beginnings of their volumes as possible thereby reducing their index numbers.

In GVAR performance will be enhanced by modifying the algorithm used to locate successive area tracks. Disk I/O set up time will be no greater than those indicated in the results.

The column marked initial CPU set up time in the Fact Sheet contains results with no forcing of track placement. The column marked optimized CPU time contains results after the tracks were forced.

2.5 Load Conditions Account for Remaining Variability

Some variability remained. Several successive runs might all take the same amount of time, but a subsequent one might vary by as much as 15%. When this occurred the time of the most frequent measurement value was used, or the result most consistent with other results was used.

Test number 64 in the Fact Sheet illustrates the two different times. The cause of this variability, we believe, is related to variations in system loading. The amount of time consumed by system calls originating from the handler will be affected by loading.

For example when setting up a disk write, if the channel is currently busy, the request must be placed on a list. The handler is charged cycles while it is being queued.

Secondly, because the host used for testing has a dual processor, the handler may be delayed (but charged for CPU cycles) while waiting to get at system data structures held by the CPU not attached to the handler.

It should be noted that the system was usually busy but the load varied throughout testing. McIDAS users were on the system, people were using the editor, and other ingest handlers were active. Variations is this loading affected the wait times for systems service requests.

3.0 RESULTS

3.1 Image Recognition Time

This time was declared to be zero in Section 1.7 of this appendix. Whatever time was used by image recognition, it did not affect the overhead time.

3.2 Overhead

Overhead remained consistent throughout the tests, about 100 lines per second.

3.3 CPU Time for Disk I/O

The CPU time for each disk set up varied initially between .0024 to .0037 seconds. The second run of test number 64 was eliminated from consideration because the calculated value for disk I/O is inconsistent (30-50%) when compared to the other values. The range then became .0024 to .0029. The number chosen for predicting GVAR performance was .0028. It is within the range, not at the end of the range, and is also close to the number calculated from the results for test 52, the test with the largest number of seconds devoted to disk I/O time.

Note that in predicting disk set-up time .0028 was reduced to .0020. This is because we feel we can reduce the algorithm which locates area tracks on disk.

3.4 Request Set-up Time

All the Vis set-up times were based on one second of CPU time. They were consistent to 30 microseconds. The value chosen, .00013, is the value calculated in the VDUC window.

If .00013 seconds was used as the channel command set-up time for non-multiband commands, the indirectly calculated multiband set-up time would be .00073 seconds if test number 54 were used or .00105 if test 52 were used. This variation depends on the fact that the total set-up time for test 52 is not equal to the total set-up time of test 53 and 54 together, even though the total number of respective GPCI commands agree. The channel command set-up time in each case is indirectly calculated, and based on the difference between two measured times, each of which is accurate to a second. Thus, a four-second range is possible between the set-up time sum of tests 53 and 54 and the set-up time of test number 52.

Fact Sheet for Determining CPU Effort Necessary
for Setting up Line Read and Disk I/O Requests

Test Label	Description	CPU Initial	CPU-Track Placement Optimized	Tests Rerun Disk I/O Off	Overhead	Channel Command Set Up Time	GPCI Line Read Request Count	CPU Setup Time Per Line Read	Seconds for Disk I/O	Number of Disk I/Os	CPU Setup Time Per I/O
63	Vis 3600 elem. 1200 lines	29	19	13	12	1	10049 Vis	.00010	6	2244	.0026
64	Vis 4800 elem. 1200 lines	48	21	13	12	1	10099 Vis	.00010	8	2991	.0027
64	Vis 4800 elem. 1200 lines		24	13	12	1	10099 Vis	.00010	11	2991	.0037
52	VDUC elem. 1621 lines		35	22	16	5	7854 Vis 1913 IR 4129 Doc 3016 Mult 16912 tot.	.00013 .00013 .00013 .00105 .00029	13	4532	.0029
53	Vis VDUC elem. 1621 lines		22	17	16	1	7854 Vis	.00013	5	2077	.0024
54	Mult VDUC elem. 1621 lines		25	19	16	3	3016 Mult 6042 Doc+IR 9058	.00073 .00013 .00033	6	2455	.0024

Figure A.1: Fact Sheet

APPENDIX B

AAA CHANNEL COMMANDS

1.0 MOVING DATA FROM THE HARDWARE TO THE HANDLER

Data from the GOES satellites flows through a frame sync, a bit sync, and an ingestor card (or cards) which holds it and sometimes formats it before sending it to the host computer. The software in the host, the ingestor handler, controls the flow of data from the ingestor cards to itself. This appendix discusses the interaction between the host and the ingestor cards, and describes the structure and chaining of commands built in the host.

2.0 CHANNEL COMMANDS

2.1 Instruction Format

IBM I/O commands are eight bytes long. They must start at a double word boundary. They are formatted as follows:

```
ccaaaaaa bbllllll
```

- The byte cc is the command.

- The three byte field, aaaaaa, is the address field. This field contains a pointer to a buffer in the host.

- The byte bb contains bits used to modify the I/O operation. The ingestors use the 4 bit and the 2 bit of the first nibble. When the 2 bit is set, the IBM hardware will chain to the next command word. The ingestor handlers always turn this bit off in the last command word in the chain to indicate chain termination. The 4 bit is used to indicate (to the channel) that the length field may be incorrect, not consistent with the number of bytes the device is expecting to send or receive.

- The 3 byte field, llllll, contains the number of bytes being received or sent.

```
01333333 60444444   WRITE-READ Pair
02333333 60555555
03000000 60000001   NOP
07000000 60000001   SYNCH
0B000000 60000001   WAIT FOR SECTOR
0F000000 60000001   IDLE
```

2.2 Telling the Ingestor Hardware What to Send to the Host

Ordinarily, to transfer data to or from a device, the ingestor must send information to the device telling it what kind of buffer to expect, or to prepare. This information is sent in the write portion of a write-read pair. Usually this information is sent in a short data field no more than 12 bytes long. The address field in the write command of the write-read pair will point to the data field, and the length field will contain a byte count of 12 or less.

In the read portion of the write-read pair, the address field will contain the location of the first word in memory of the buffer receiving the data. The length field will contain the number of bytes to send to memory.

2.3 NOP, SYNCH, WAIT, IDLE Instructions

These instructions are, by convention, commands that need no buffer location or length fields.

NOP is a null instruction, set up at the beginning of a chain.

The last three are control instructions which tell the ingestor hardware to respond when various events occur.

The hardware responds to an IDLE when it discovers that some signal data has been received.

The hardware responds to a SYNC when it discovers that it has received information positioned at the beginning of the next geographic scan line.

The hardware responds to a WAIT when it discovers that it has received information for the next sector in the sequence of data sectors, whether or not the sector is for the same geographic scan line.

2.4 Control Instruction Time-out

If a control instruction times out, control works its way back to the handler so it can start contingency activity. The amount of time before time-out for each of these instructions varies from ingestor to ingestor. The time is usually 90 seconds for an IDLE. For WAIT and SYNC the time will vary from satellite to satellite. These times also vary with tape speed if signal is coming from archival storage.

3.0 COMMAND CHAINS

3.1 Ingestor Waiting for Signal

Below is a simple chain typical of those used when an ingestor is waiting for a signal (IDLing). First in the chain is a No-Operation instruction. The next tells the ingestor card to wait for signal. When signal is sensed, a write-read pair tells the hardware to send a copy of scan line documentation to the mainframe.

Note that there is not any length field telling the hardware how many common doc bytes to send. This length is preset in the hardware for each ingestor.

```
1000      03000000      60000001      NOP
1008      0F000000      60000001      IDLE
1010      011225E8      60000005      WRITE 5 bytes
1018      02133A60      20000072      READ 114 bytes
                                      beginning at byte
                                      address 133a60
                            .
                            .
                            .
1225E8    A0000000      00000000      Command string
                                      telling hardware to
                                      send common doc
                            .
                            .
                            .
133A60    00000000      00000000      Beginning of buffer
                                      that receives
                                      documentation
```

3.2 Ingestor Receiving Data

The chain below sets up data reads for one scan line, and is typical of a chain created when the ingestor is taking data. Note that a non-zero valcode is sent during the doc request, and that a sync is used instead of an idle.

```
1000      03000000     60000001     NOP
1008      07000000     60000001     SYNC
1010      011225E8     60000005     WRITE 5 bytes
1018      02133A60     20000072     READ DOC
1020      011225F4     60000005     READ data
1028      0216001C     600003E8
1030      01122600     60000005     READ DOC as data
1038      02160000     6000001C
                           .
                           .
                           .
1225E8    A0060780     BB000000     Command string

                                    telling hardware to
                                    send common doc
1225F4    400000E8     03000000     Command string for
                                    data read
122600    80000018     00000000     Command string for
                                    doc as data
                           .
                           .
                           .
133A60    00000000     00000000     Beginning of buffer
                                    that receives
                                    documentation
                           .
                           .
                           .
160000    00000000                  Beginning of PFX in
                                    data buffer
                           .
                           .
                           .
16001C    00000000                  Beginning of Data in
                                    data buffer
```

3.3 The Operating System and the Command Chains

Once a chain is constructed, it is sent as a group to an operating system routine in the host which initiates it. Except for setting status bits indicating that a chain timed out, or completed normally or abnormally, the operating system exerts little control over it once it starts. Each command executes when the previous one completes.

When the main driver of an ingestor is re-entered, it checks for timeout, abnormal completion, normal completion, or none of these conditions. (The last alternative leads to an error state.) The abnormal condition usually means one of the commands was not satisfied and a new attempt will be necessary after analysis. If a timeout occurs, this means no signal was sensed and the write-read pair was never issued. If the full chain is completed, handlers usually proceed to analyze common doc.

3.4 AAA Ingestor Command Structure

Channel CMD	Parameters	Code
NOP	NONE	03
SENSE	NONE	04
READ	NONE	02
IDLE	NONE	0F
WAIT	NONE	0B
SYNC	NONE	07
WRITE	FOLLOW:	01

DOC CODE #PARAM,VC(4),FLAGS,BANDS(2),VIS SCAN(2),RAVSCAN(2)

DATA CODE #PARAM,OFFSET(2),LENGTH(2),CLIP(1)

#PARAM	The number of parameter bytes which follow.
VC	Validity Code
FLAGS	Bit 0=0 - Real time =1 - Archive Bit 1=0 - Step only (non-dwell sub_1 and sub_3 only) =1 - Raw Bit 2=0 - Not raveled =1 - Raveled if card #2 present
BANDS	Filter bands, low byte first, high byte second. Each bit specifies the desire for a specific band. Bit 0=1 indicates the desire for band 1 and so on. Only one raveled request possible per scan line.
SCAN	Scan line desired, low byte first, high second.
RAVSCAN	Scan line desired if data is sent raveled.
OFFSET	Offset from the start of data buffer, low byte first.
LENGTH	Length of data transfer, low byte first.
CLIP	Clip byte count for library data requests.

DATA/DOC CODES

(UPPER NIBBLE)

Bits	128	64	32	16
Fields	7	6	5	4
	VC 1=Yes	Prefix 1=Yes	Comdoc 1=Yes (off first card)	Card count 1-2 cards

The prefix bit specifies whether the prefix or data part of a mode AA/AAA IR sector is to be sent.

0XH = Data w/o Validity Code, no prefix, 1 card
8XH = Data with Validity Code, no prefix, 1 card

20H = Comdoc w/o Validity Code
A0H = Comdoc with Validity Code

1XH = Data w/o Validity Code, no prefix, 2 cards
5XH = Data w/o Validity Code, prefix, 2 cards
9XH = Data with Validity Code, no prefix, 2 cards
DXH = Data with Validity Code, prefix, 2 cards

X = OUTPUT RESOLUTION CODE (LOWER NIBBLE)

VAL	NIBBLE	FUNCTION
0	0000	FULL RES BYTE
1	0001	FULL RES 2 BYTE
2	0010	÷ 2 AVERAGE
3	0011	÷ 3 AVERAGE
4	0100	÷ 4 AVERAGE
5	0101	÷ 6 AVERAGE
6	0110	÷ 8 AVERAGE
7	0111	÷ 12 AVERAGE
8	1000	÷ 16 AVERAGE
9	1001	÷ 2 SAMPLE
A	1010	÷ 3 SAMPLE
B	1011	÷ 4 SAMPLE
C	1100	÷ 6 SAMPLE
D	1101	÷ 8 SAMPLE
E	1110	÷ 12 SAMPLE
F	1111	÷ 16 SAMPLE

3.5 <u>Raw Data Formats</u>

3.6 Output Area Data Format

VIS	Valcod (4)	data (4-15800)					
A-IR	Valcod (4)	Comdoc (128)	data (4-15800)				
AAA-IR	Valcod (4)	Prefix (632-644)	data 16 bit vals (4-99372 bytes)				
PREFIX OF AAA-IR	Comdoc (512)	Cal (116)	Levels (4, 8, 12, or 16) Levels is one byte for each band in this line of data in the order they occur.				
CAL of PREFIX	day (4)	time (4)	scan (4)	Chan (2)	nspin (2)	DeltaF (2)	Yz (2)
						Repeats 13 times (also in order)	
COMDOC and STATUS	Valcod (4)	Status (8)	Comdoc (128,512)	Gridbits (0-??)			
STATUS ACCOMPANY-ING COMDOC	Mode (1)	Scan (2)	flags (1)	dcount (1)	spares (3)		
	Mode is A, AA, or AAA Scan is micro scan no.			dcount is no. empty DMAs in previous spin			
FLAGS	unused	Sig	SD	Chk			
	crds	- # cards present bit; 0-1 card; 1-2 cards					
	Sig	- signal present bit (frame code)					
	SD	- Sector Detector bit (no comdoc)					
	Chk	- checksum bits (one for each common doc parity result)					