To the Graduate Council:

I am submitting herewith a thesis written by John H. Jones entitled "Software Techniques for the Acquisition of Optical Data from a Minicomputer-Based Image-Intensifier-Vidicon System." I recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Computer Science.

Fletcher W. Donaldson

Fletcher W. Donaldson, Major Professor

We have read this thesis
and recommend its acceptance:

R. A. B.

C. C. Limbaugh

Accepted for the Council:

L. Evans Lott

Vice Chancellor
Graduate Studies and Research

SOFTWARE TECHNIQUES FOR THE ACQUISITION OF OPTICAL DATA FROM A MINICOMPUTER-BASED IMAGE-INTENSIFIER-VIDICON SYSTEM

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

John H. Jones

June 1979

DEDICATION

This thesis is dedicated to the author's wife, Carol T. Jones, and his children--Marla, Ben, and Evan.

ACKNOWLEDGMENTS

The author wishes to express his appreciation to Dr. Homer Powell, Richard McGuire, and Newt Wright for their assistance and contributions to this effort. The author also wishes to express thanks to Dr. F. W. Donaldson, Dr. Ron Belz, Dr. Chad Limbaugh, Dr. J. W. L. Lewis, and the author's wife, Carol.

Appreciation is also expressed to the officials of the Air Force and ARO, Inc., Arnold Air Force Station, Tennessee, for providing the resources required in this effort and for the opportunity to attend The University of Tennessee Space Institute.

The research reported herein was sponsored by Headquarters, Arnold Engineering Development Center, Arnold Air Force Station, Tennessee, within the terms of Contract F40600-77-C-0003 with ARO, Inc., a Sverdrup Corporation company. Further reproduction is authorized to satisfy the needs of the United States Government.

ABSTRACT

The software required to support a new data acquisition technique which can acquire large amounts of optical data in a short time period has been developed. The data acquisition system incorporates an image intensifier and vidicon tube in combination and is controlled by a minicomputer utilizing the described software. The software handles all control of the image-intensifier-vidicon system, acquires the data from the system, and stores the data for subsequent retrieval and reduction. The software incorporates a disk operating system for a single floppy disk drive. This disk operating system utilizes a specially designed technique that optimizes the rate of writing data onto the disk.

The system was implemented to acquire temperature and density data from the flow of a rocket engine exhaust plume. This application coupled the system with an electron beam mechanism to acquire the data using an electron beam fluorescence technique and was used to verify that the system meets the requirements set forth for it.

TABLE OF CONTENTS

CHAPTER PAGE

LIST OF TABLES

LIST OF FIGURES

FIGURE PAGE

INTRODUCTION

Many fields of endeavor in the physical sciences involve the study of some phenomenon by observation of the electromagnetic radiation (light) it produces. The observer can indirectly quantify the phenomenon through its emitted light. Characteristics of the light give insight into the nature of the phenomenon and may be interpreted to give the underlying physical insights which the observer is seeking.

Measurement of low-level light intensities requires highly specialized and complex instrumentation. Obtaining a complete set of descriptive data using these instruments can require an enormous amount of time. Consequently, conventional instrumentation is not practical in many applications. An instrument which can acquire these data in large quantities in a short time period is required.

In response to this requirement, an image-intensifier-vidicon data acquisition system (vidicon system) was designed, built, and tested. Low-level light intensities over a wide range of wavelengths can be measured, digitized, and recorded.

The vidicon system consists of the following:

1. An image intensifier with gain-selection power supply.

2. A silicon-target vidicon equipped with video-processing instrumentation.
3. A computer interface.
4. An oscilloscope display.
5. A minicomputer with associated line printer, Teletype(R), input/output (I/O) ports, and single floppy disk drive.

The image intensifier controls the admittance of light to the vidicon and in the process amplifies the light to a detectable level. The vidicon converts the light striking its rectangular target into charge proportional to the incident intensity and stores the charge on the target at the location of incidence. The computer interface controls the video-processing instrumentation to convert this charge into digital equivalents which are sent to the computer upon request. The computer directs the operation of the interface by indicating to it which data to convert and when. The computer stores the data on floppy disk for later retrieval and display. An oscilloscope is used for (a) monitoring the status of the computer interface, (b) displaying data in analog form, and (c) adjusting and setting up the vidicon system.

The vidicon components are integrated into a system flexible enough to handle a wide variety of applications. The vidicon system's flexibility arises from such features as:

1. The ability to handle a wide range of intensities.
2. The ability of the vidicon target to integrate the incoming light.
3. The ability of the image intensifier to allow time-resolved or time-integrated measurements.
4. The adaptability of the computer software to different test situations.

The minicomputer is the hub of the vidicon system; all activities are monitored and controlled by it. This requires that the software allow flexibility and adaptability. The adaptability of the computer to any particular application requires a nucleus of systems software routines that allow the user to avoid the details of the operation of the vidicon system [1].[1] Such a set of routines was designed for the system and is called the Minicomputer-Based Image-Intensifier-Vidicon System Software (MIIVSS). MIIVSS was designed to control all vidicon operations and the acquisition, storage, and retrieval of data with strict attention being paid to the optimization of throughput. The user software calls MIIVSS routines to perform the necessary vidicon operations and the order of such calls, under certain restrictions, is left to the

---

[1]Numbers in brackets refer to similarly numbered references in the Bibliography.

user for definition thereby allowing flexibility to accommodate different situations.

The primary functions of MIIVSS are as follows:

1. Inputting data from the vidicon system, storing the data in files on disk, and retrieving the data for display and data reduction.
2. Opening and closing files.
3. Retaining the locations of files on disk.
4. Deleting files from storage.
5. Defining the format of files.
6. Detecting and tracing errors.
7. Controlling test mode operation of the vidicon system.

MIIVSS creates and maintains files on disk with a disk management system specially designed by the author for the vidicon system. The disk management system utilizes a unique disk writing technique that provides throughput an order of magnitude greater than simple disk writing techniques.

Descriptions of the MIIVSS software, the vidicon system hardware, and an application of the vidicon system are presented in the following material. Chapter I introduces the major components of the vidicon system. Chapter II describes the functions provided by MIIVSS. Chapter III presents an application of the vidicon system and illustrates how MIIVSS is used in this application.

The author's contribution to the overall vidicon system project was the design, programming, and implementation of MIIVSS and the software required for the application presented in Chapter III.

CHAPTER I

VIDICON SYSTEM HARDWARE

1.0 INTRODUCTION

Any discussion of MIIVSS would be incomplete without a description of the vidicon system's hardware components. The components integrated into a unified system are shown in block diagram in Fig. 1.

This chapter describes (a) the image intensifier, (b) the vidicon, and (c) the computer interface. The vidicon system's components are listed in Appendix A. Details of the floppy disk system are given in Appendix B.

1.1 THE IMAGE INTENSIFIER

The image intensifier (intensifier) controls the intensity and duration of the light reaching the vidicon target. The intensifier can completely block the light or admit it and, at the same time, amplify it by a preset gain factor. An illustration of the intensifier is shown in Fig. 2.

Light striking the photocathode causes electrons to be displaced from it in quantities directly proportional to the incident light intensity. These electrons are drawn into an "electron multiplication zone" by an electric field between this zone and the photocathode.

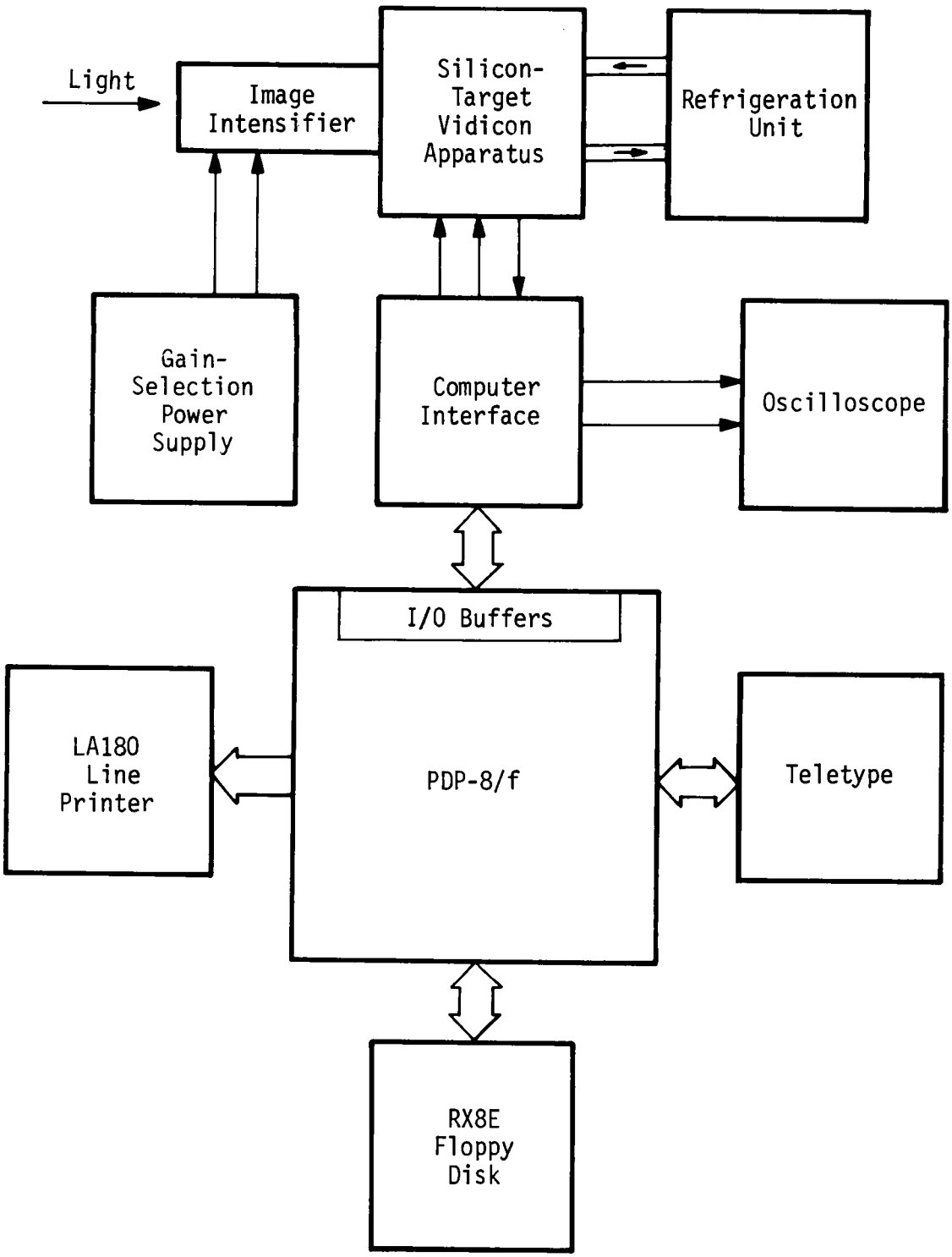


Figure 1. Diagram of the vidicon system.

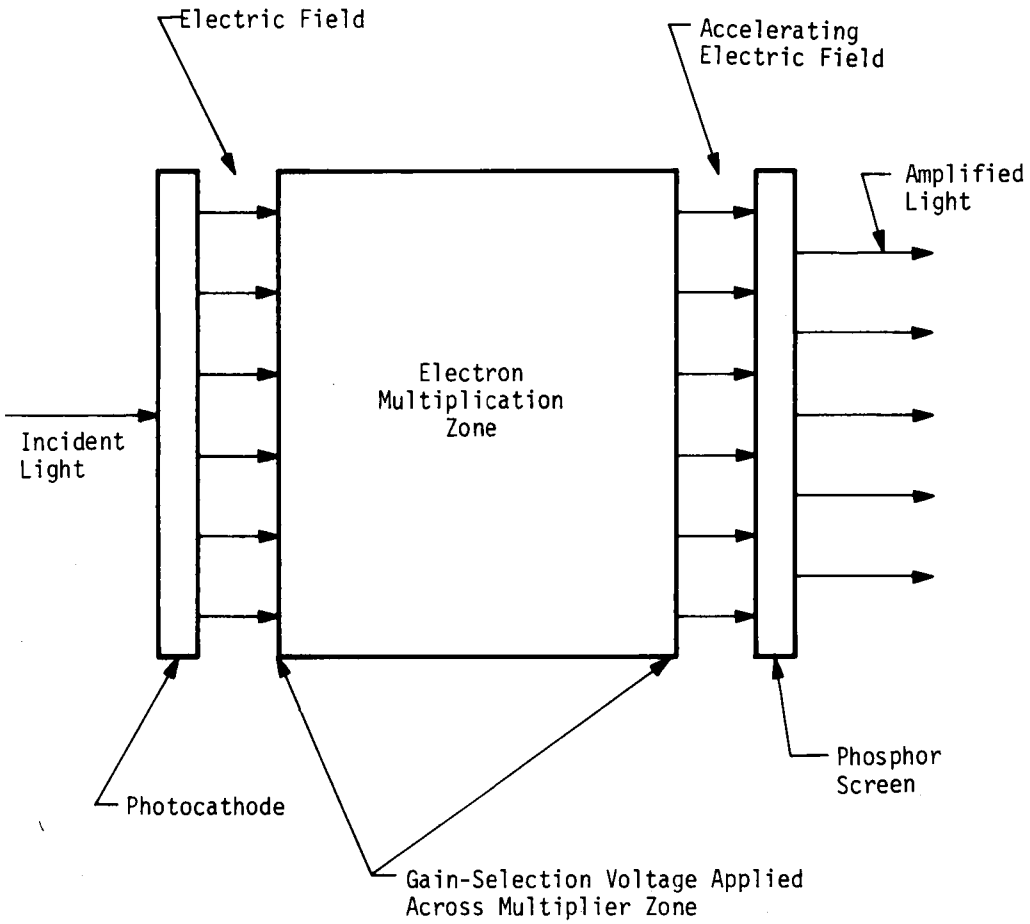


Figure 2. Cross section of the image intensifier.

Each electron entering the multiplication zone causes a number of other electrons to be displaced and move with it. The electron cloud produced in the multiplication zone is then accelerated to a higher energy by an electric field between the zone and a phosphor screen. The electron cloud strikes the phosphor screen causing it to emit light proportional in intensity to the number and energy of the electrons colliding with it. Since the electrons striking the phosphor screen are identical in spatial distribution, but greater in number and energy than those produced by the photocathode, the intensifier effectively produces an output spatially comparable to, but greater in intensity than, the incident light.

The number of electrons produced in the multiplication zone by each incoming electron is directly proportional to the voltage applied across the zone. This means that this voltage controls the amplification, or gain, of the intensifier.

The intensifier can block the transmission of the light by replacing the electric field between the photocathode and the multiplication zone with a slight repulsive field. This blocks all electrons from passing from the photocathode, thereby preventing electrons from reaching the multiplication zone and the phosphor screen.

Operation of the intensifier is not handled by MIIVSS; it is left to the operator to set the gain and to determine the duration of the incoming radiation. The

repulsive and accelerating electric fields are produced by conventional circuitry.

1.2 THE VIDICON

The vidicon target is constructed within a vidicon tube which also contains an electron beam mechanism used for obtaining the data from the target. An illustration of the vidicon tube is shown in Fig. 3. Light striking the silicon target causes charge to be built up and stored on the target. The electron beam is then turned on and strikes the target causing the output of a video signal which is amplified and sent to the computer interface for digitization. The devices which produce the video signal are called the video-processing instrumentation.

The vidicon target is divided by the vidicon system into an array containing 256 rows and 256 columns of storage elements called pixels [2]. Each pixel has unique row and column addresses. The computer interface converts the pixel's row and column addresses into voltages applied to coils to position the electron beam. The resultant video signal amplitude is proportional to the incident light intensity at this location. The pixel-reading process destroys the charge in the pixel.

The vidicon target is susceptible to noise buildup over a period of time. Noise is extraneous charge caused by thermal effects in the silicon target and by light emitted by the electron beam mechanism. Noise builds up

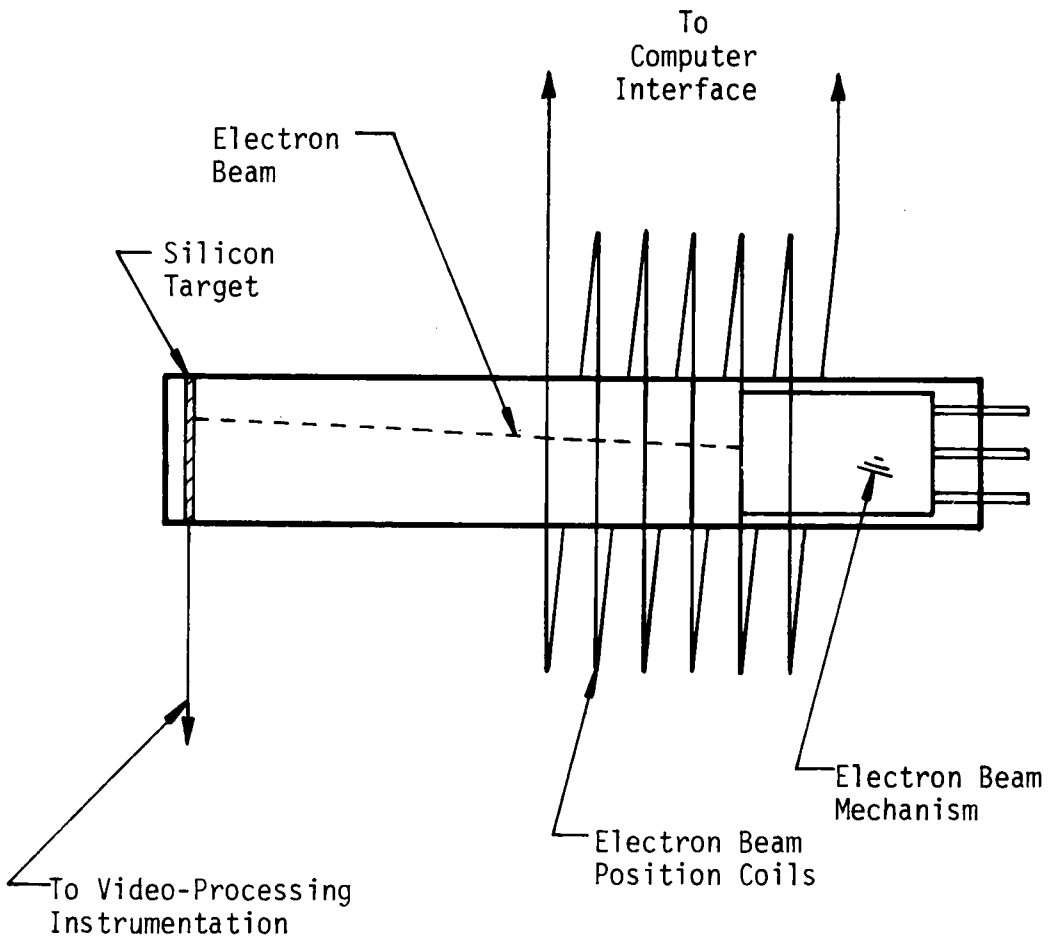


Figure 3. An illustration of the vidicon tube.

at a rate of approximately 1 percent of full scale, or saturation, per second. The noise problem can be minimized by operating the vidicon tube at reduced temperatures to minimize thermal effects. The vidicon system is equipped with a refrigeration unit that cools the vidicon tube to a range of -40 to -50°C.

But even at reduced temperatures, noise buildup still occurs and is a problem since the noise can reach levels that obscure the data. It is imperative, therefore, that the data be read from the target as quickly as possible.

1.3 THE COMPUTER INTERFACE

The computer interface was specially designed for the vidicon system. A diagram of the interface is shown in Fig. 4. Under computer direction, the interface addresses individual pixels, converts the resultant video signals into digital equivalents, and sends these data to the computer.

The interface contains an analog-to-digital (A/D) converter and two digital-to-analog (D/A) converters. The A/D converter converts the video signal coming from the video-processing instrumentation into 8-bit bytes of digital data in the range 0 to 255. The D/A converters provide voltage to the beam-positioning coils surrounding the vidicon tube; one converter selects the row deflection

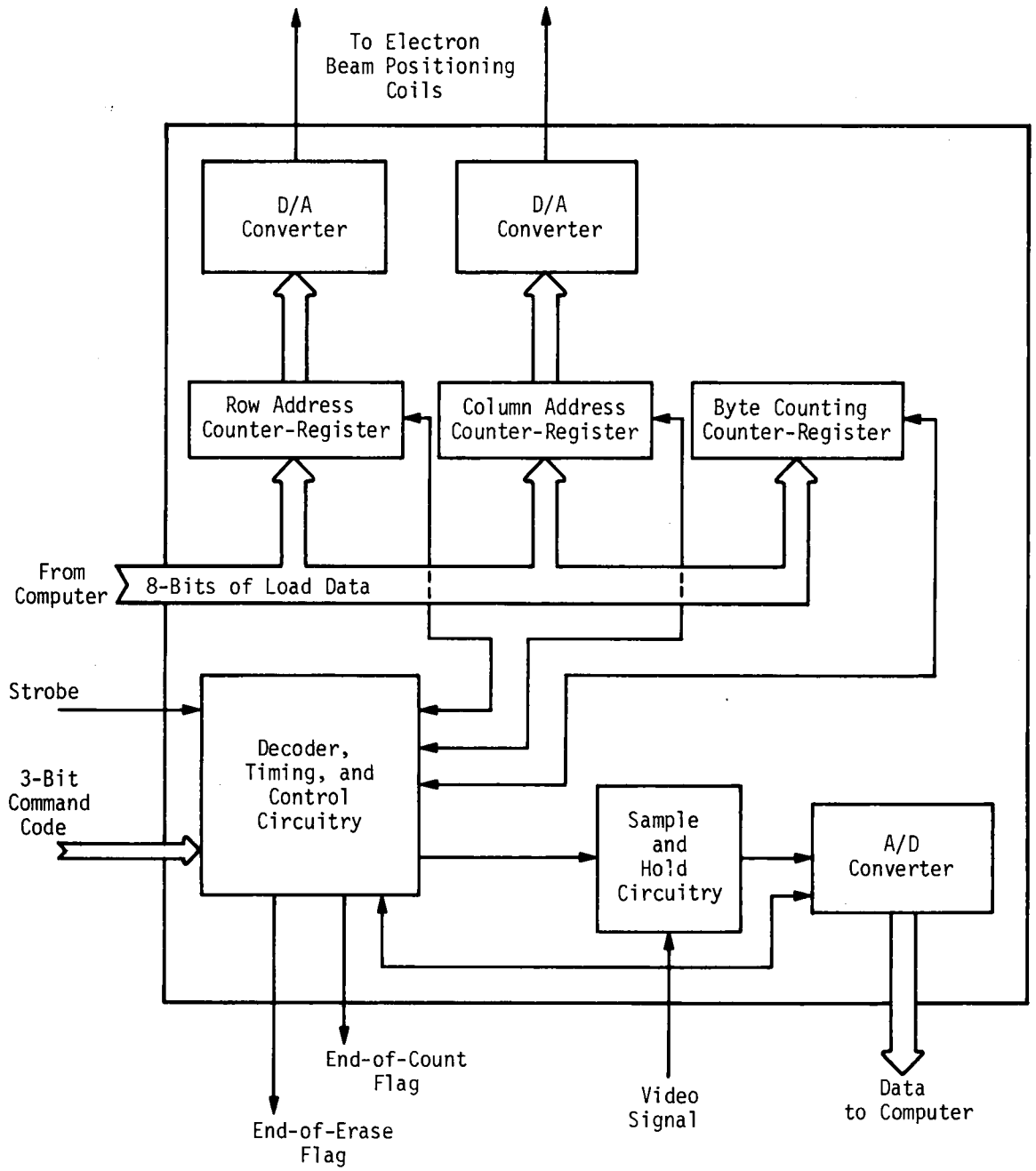


Figure 4. Diagram of computer interface.

while the other selects the column deflection. The inputs to the converters are 8-bit bytes in the octal range -200 to +177.

Other devices in the interface include three 8-bit counter-registers, a sample-and-hold circuit, timing and control circuits, and a 3-bit command decoder. Two of the counter-registers are used to hold the row and column addresses for the D/A converters. In register mode, these are loaded by the computer to address specific pixels; in counter mode these registers are successively incremented to step through a series of pixel addresses. The third counter-register is used to count each byte of data as it is produced and sent to the computer. The sample-and-hold circuit samples the video signal and holds the resultant voltage for conversion by the A/D converter. Timing and control circuitry provides the synchronization required for the operation of the interface. The 3-bit command decoder chooses the proper selection line to enable each operation.

Several functions besides data conversion can be performed by the interface as commanded by the computer. These are (a) load the row-address register, (b) load the column-address register, (c) load the data-count register, (d) perform a single-cycle erase, and (e) perform a multi-cycle erase. The multicycle erase command causes the interface to repeatedly cycle through all pixels on the target removing all charge present in them by performing

data conversion and discarding the results. The single-cycle erase is performed for one cycle and stops at the end of that cycle. The multicycle erase mode is terminated by entering the single-cycle mode.

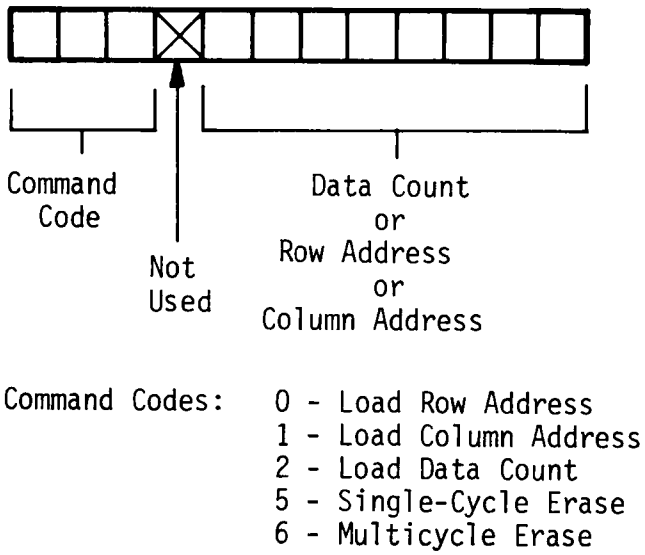
These functions are controlled by the computer with a 12-bit command word. The command codes and the format of the command word are shown in Fig. 5a. The three most significant bits of the word contain the code of the function to be performed. The eight least significant bits contain the row address, column address, or data byte count depending on the function being executed. The remaining bit of the command word is not used.

Several flags are output by the interface to indicate to the computer when certain operations have been completed; these are end-of-erase, end-of-conversion, and end-of-count.

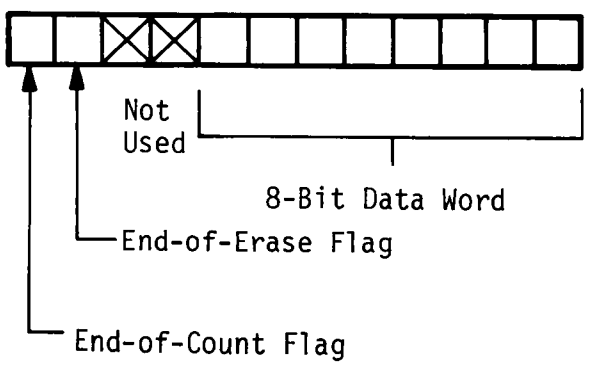
The 8-bit data byte and two of the flags are packed into a 12-bit data word which is sent to the computer. The data word's format is shown in Fig. 5b. The eight least significant bits of this word contain the data produced by the A/D converter. The two most significant bits contain the end-of-count and end-of-erase flags. Handling of the end-of-conversion flag is discussed in Section 1.4.

1.4 COMPUTER/INTERFACE COMMUNICATIONS

Information transfer between the computer and the vidicon system interface is accomplished by passing the



a. Command word format



b. Data word format

Figure 5. Formats of the command and data words.

command words to the interface or the data words to the computer.

MIIVSS commands a vidicon operation by sending a command word to the interface via a 12-bit parallel output buffer. The interface reads the word, decodes it, and performs the requested operation.

Transfer of a data word requires synchronization between the computer and the vidicon system interface. When a byte of data is requested, MIIVSS must wait before inputting the data word until the interface converts the data and signals the end of the conversion process. Computer/interface synchronization is accomplished by a flag mechanism which is implemented by connecting the end-of-conversion flag to a flip-flop in the minicomputer. MIIVSS periodically checks the status of the end-of-conversion flip-flop and inputs the data word via a 12-bit parallel input buffer when the flip-flop becomes set. This technique allows asynchronous operation between the computer and the interface.

During the process of acquiring data from the interface, MIIVSS inputs a series of data words and checks each for the end-of-count flag. When this flag becomes set, MIIVSS detects that the number of data bytes requested has been reached and that it may then move to the next function.

During the single-cycle erase mode, MIIVSS inputs a series of data words and checks the end-of-erase flag for an indication of when the erase cycle has completed.

The computer instruction which inputs the data word also outputs a signal (strobe) that triggers the vidicon system interface to move to the next pixel in the currently addressed row. The column address is incremented by one and the conversion process is initiated. This technique allows a series of pixels in a row to be read without MIIVSS having to select each pixel by its address. Initialization of this process requires loading the row and column addresses of the first pixel and the number of pixels, or data bytes, to be read. An input instruction is then executed to initiate the reading process.

CHAPTER II

THE MINICOMPUTER-BASED IMAGE-INTENSIFIER-VIDICON SYSTEM SOFTWARE

2.0 INTRODUCTION

Maximized data storage throughput is a necessity for MIIVSS because of the quantity of data which can be acquired by the vidicon system and the problem of noise buildup on the vidicon target.

The ability to achieve the required speed is impeded by the slowness of the floppy disk system. The data acquisition and storage technique used must take advantage of the speed of the computer and the interface, yet take into account the lack of speed of the floppy disk.

A disk writing technique (described below) was developed for MIIVSS to allow the computer to acquire and store data on the disk at high transfer rates. MIIVSS drives the floppy disk at its fastest rate and performs data acquisition while the disk system simultaneously records data onto the disk.

This disk writing technique is part of the disk management software used by MIIVSS to acquire and store data. The disk management software arranges the data into files, writes the files onto the disk, maintains a directory to keep track of the data, and retrieves data

from the disk for data reduction or display. The disk management software also has an error tracing feature which facilitates determining the causes of any errors that might occur. These features are described in this chapter.

2.1 FIELDS

Data are read from the vidicon target in rectangular patterns called fields. Fields are rectangular because of the simplicity, speed, and flexibility which they afford. The simplicity arises from the fact that any rectangular geometry which may be needed can be specified by five parameters. These consist of the following:

1. The address of the first row which is to be read.
2. The column address of the first pixel which is to be read from each row.
3. The number of pixels which are to be read from each row.
4. The number of rows which are to be read.
5. The spacing between rows.

Note that the spacing between pixels is controlled by the computer interface which only allows stepping between adjacent pixels in the direction of increasing address.

More complex geometries could be used but these would slow down the data acquisition process. These geometries might require the addresses of each pixel to be calculated, thereby requiring computer time on the order

of several hundred microseconds to perform the address calculation. This is opposed to a few microseconds to acquire the data from each pixel as required by the rectangular pattern geometry. Another technique for describing the complex geometries is to calculate the row addresses, the initial column addresses, and the number of pixels per row and then let the computer interface perform the stepping between pixels. Here again the rectangular pattern is faster; each initial column address and the number of pixels per row are already given and each row address can be calculated by a simple addition of the row spacing increment to the previous row address.

The simplicity of the definition of rectangular fields allows simplicity in operating the vidicon system. The five definition parameters may be entered directly through the Teletype keyboard and maintained in a table. More complex geometries would require that a defining function be input; this could require reprogramming of the definition algorithm for each application.

The use of rectangular fields does not preclude the reading of other geometries since such patterns can be enclosed within fields and, during data reduction, all data not in the pattern of interest can be ignored. This allows flexibility for use in a wide variety of applications.

Since there may be several fields defined on the target, a number between 1 and 255 is arbitrarily assigned

to each field by the operator and is used for identification during data acquisition and retrieval.

The five defining parameters and the associated field number are herein referred to as the field specification.

2.1.1 The Field Map Table

The field specifications are maintained in a list herein called the field map table. The table's format is shown in Table I.

The field map table has two counters--an entry counter and an entry-used counter. The entry counter indicates the number of entries, or field specifications, in the table and the entry-used counter indicates the number of entries that have been used by MIIVSS. These counters allow the table to be used as a first-in-first-out (FIFO) list [1]. The FIFO list capability afforded by the two counters allows MIIVSS to copy entries in order from the table without removing the entries from the table. The table can be re-used by setting the entry-used counter to zero.

The table is limited to 32 entries as a compromise between limiting the table's size and allowing as many fields as possible to be maintained. The selection of 32 fields was directed by the size of the file directory which is discussed in Section 2.2.1, page 33.

Table I. Format of the Field Map Table

Word Number	Content
1	Entry-Used Counter
2	Entry Counter (N)
3	Field Identification (1)
4	Row Address (1)
5.	Column Address (1)
6	Number of Rows (1)
7	Number of Pixels per Row (1)
8	Row Spacing Increment (1)
9	Field Identification (2)
10	Row Address (2)
11	Column Address (2)
.	.
.	.
.	.
6N - 3	Field Identification (N)
6N - 2	Row Address (N)
6N - 1	Column Address (N)
6N	Number of Rows (N)
6N + 1	Number of Pixels per Row (N)
6N + 2	Row Spacing Increment (N)

2.1.2 Field Map Table Input

A flowchart of the field map table input routine is shown in Fig. 6. This routine sets up the table via inputs from the Teletype keyboard.

The table is first cleared by setting the entry counter to zero. Each field specification parameter is then input as a decimal number in the range 1 to 256. The input number is terminated by a single character that is used to identify which parameter the input number represents. A list of these terminating characters and their meanings is given in Table II.

Row and column addresses on the vidicon target lie in the octal range -200 to +177. These "physical" addresses are converted for input as "logical" addresses in the decimal range 1 to 256. The logical addresses are converted into the equivalent physical addresses for storage in the table. The correspondence between physical and logical addresses is shown in Fig. 7.

Each parameter entered is saved in a predetermined location and a flag is set to indicate that this parameter has been input. These flags are checked before the field specification is added to the table as an indication of field specification completeness.

Also, the range of each field is computed and checked for overlap of other fields. If overlap is detected, the message "OVERLAP" and the number of the field are printed on the Teletype printer. An indication of

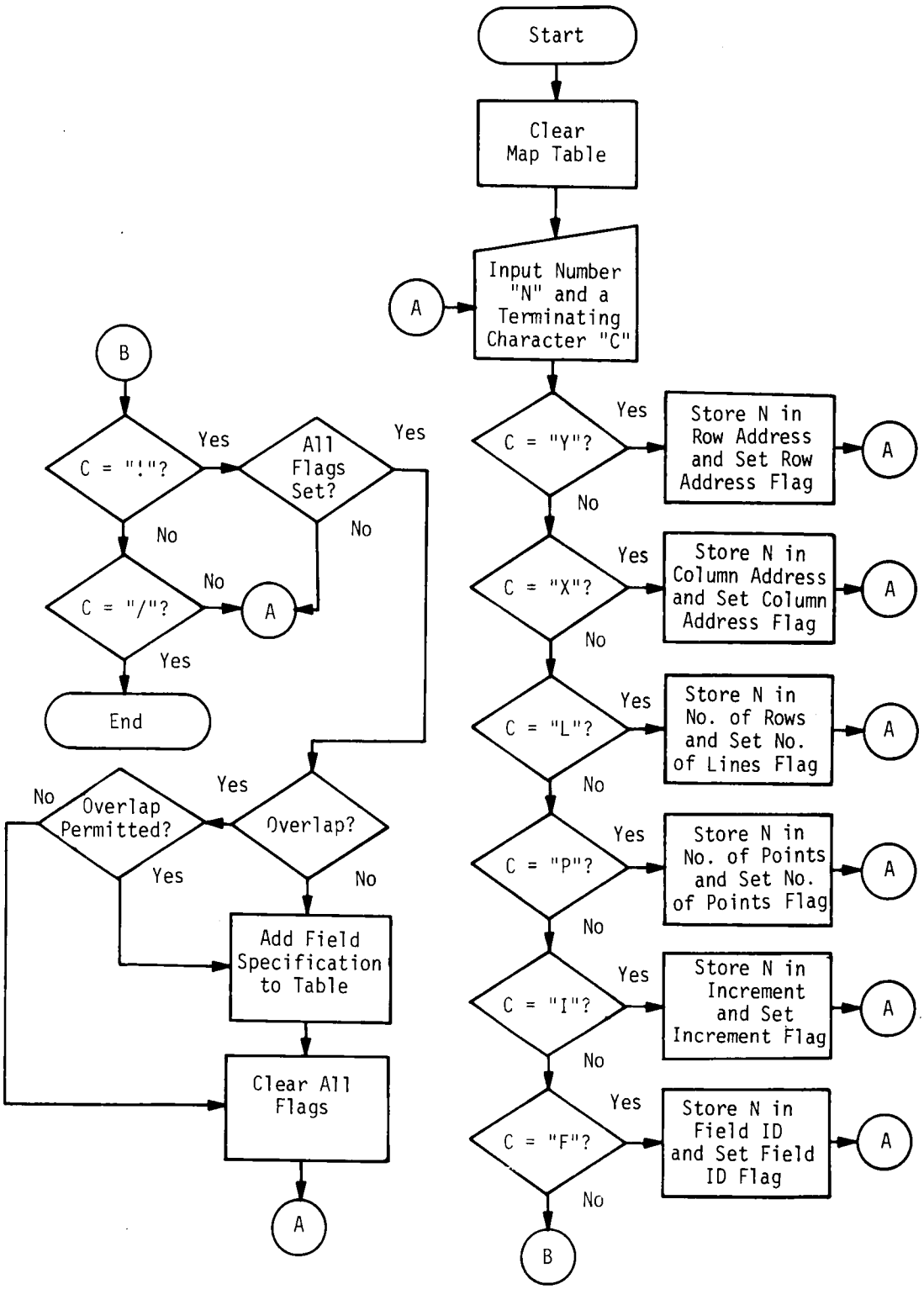


Figure 6. Flowchart of field map table input.

Table II. List of Valid Terminating Characters for Field Map Table Input

Terminating Character	Meaning of Character
Y	The input number is the initial row address of the field
X	The input number is the column address of the first pixel read from each row
L	The input number is the number of rows to be read
P	The input number is the number of pixels to be read from each row
I	The input number is the spacing between rows
F	The input number is the field number
!	Add the current field specification to the table
/	Terminate table input

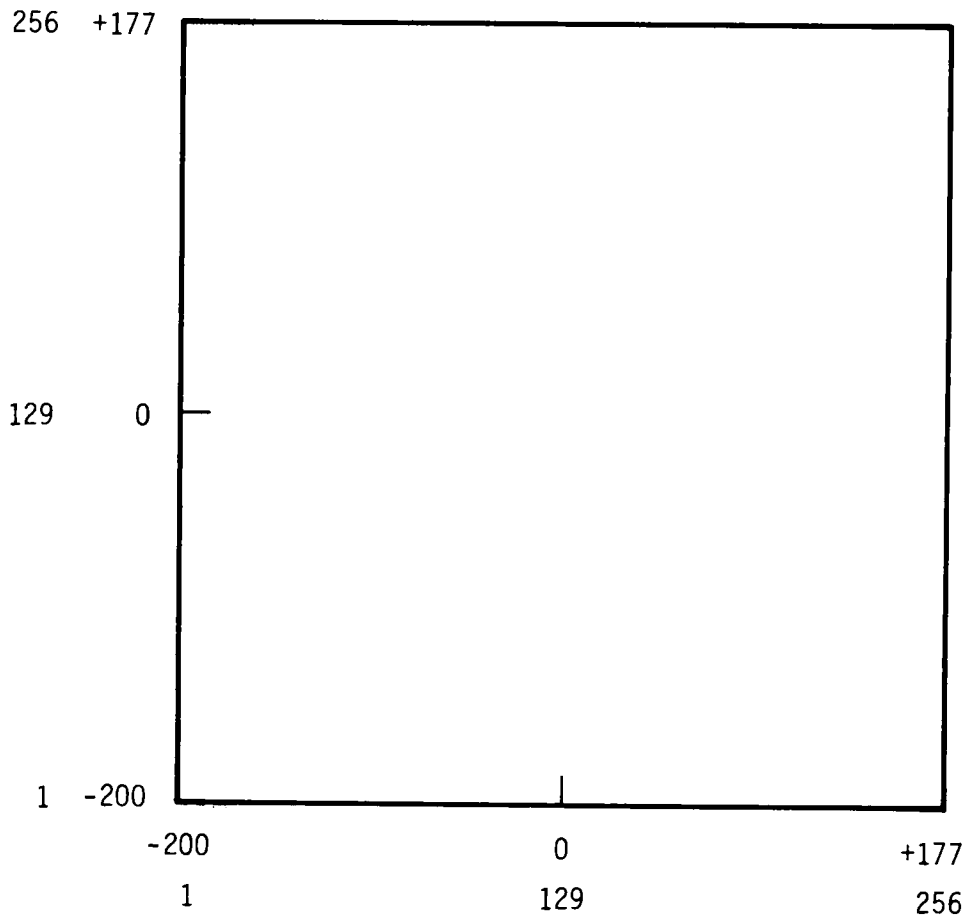


Figure 7. The vidicon target with physical and logical addressing.

overlap permissibility is then expected as a single character input via the keyboard. Permissibility is indicated by input of the slash character; all other characters cancel the current field specification.

2.1.3 Field Map Table Output

A flowchart of the field map table output routine is shown in Fig. 8. This routine prints the field map table in tabular form on the line printer. An example of table output is shown in Table III.

At the end of the table, the number of sectors required for storage of the file defined by this field map table is printed to give the operator an indication of the size of each file.

2.1.4 Disk Storage and Retrieval of Field Map Table

The MIIVSS software allows a permanent record of the field map table to be maintained on disk. Flowcharts of the storage and retrieval routines are shown in Figs. 9 and 10.

The table is written onto disk in the first two sectors of disk storage--sectors zero and three of track zero. These two sectors are permanently allocated for storage of the map.

The map is written onto the disk in 8-bit bytes. Whenever the field specification parameters representing the number of rows and the number of pixels per row are equal to 256, they are written onto disk as 8-bit zeroes

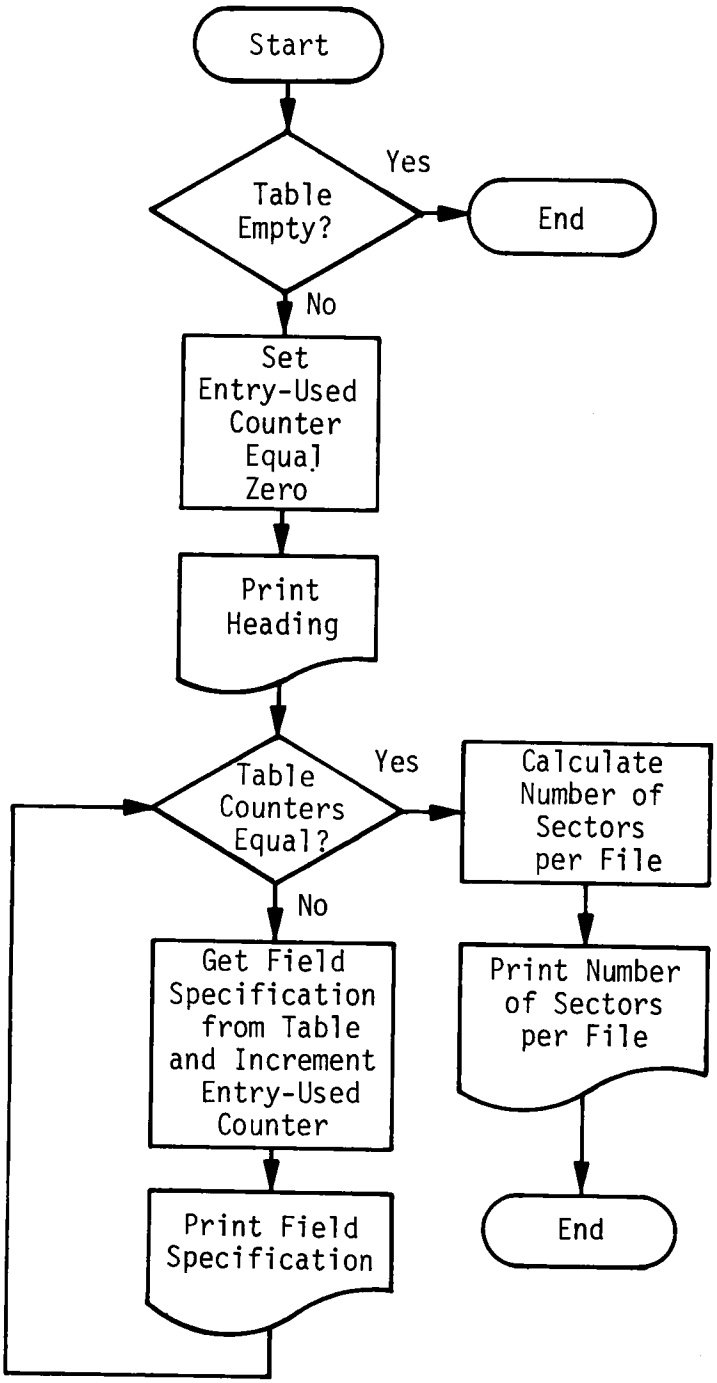


Figure 8. Flowchart of field map table output.

Table III. An Example of Field Map Table Output

ID	Y	X	#Y	#X	YINC
0001	0007	0008	0004	0248	0001
0002	0027	0008	0004	0248	0001
0003	0047	0008	0004	0248	0001
0004	0067	0008	0004	0248	0001
0005	0087	0008	0004	0248	0001
0006	0107	0008	0004	0248	0001
0007	0127	0008	0004	0248	0001
0008	0147	0008	0004	0248	0001
0009	0167	0008	0004	0248	0001
0010	0187	0008	0004	0248	0001
0011	0207	0008	0004	0248	0001
0012	0227	0008	0004	0248	0001
0013	0247	0008	0004	0248	0001
#SEC=0103					

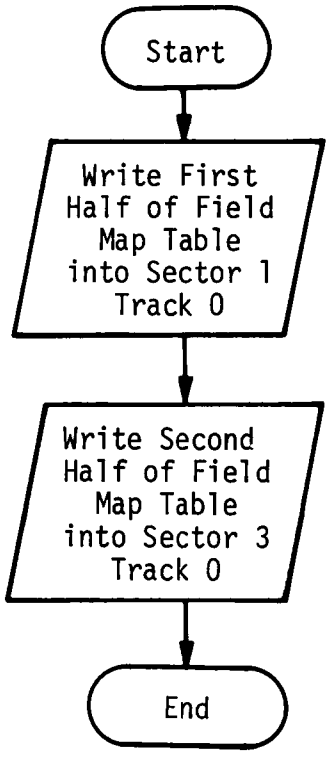


Figure 9. Flowchart of disk storage of field map table.

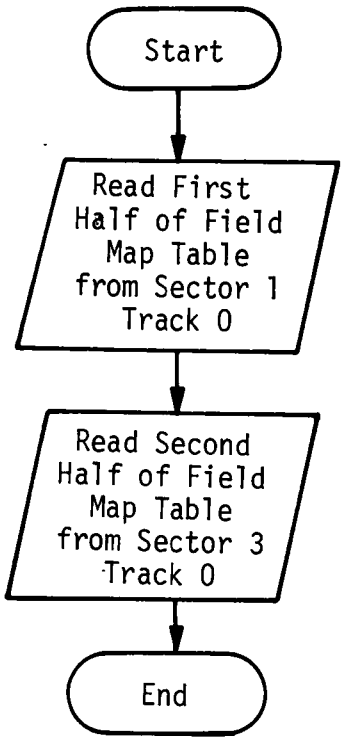


Figure 10. Flowchart of disk retrieval of field map table.

since 256 cannot be represented by an 8-bit number. The 8-bit zero is recognized as 256 because zero is not valid for these two parameters.

2.2 FILES

Data acquired from the vidicon target are maintained on the disk in files [1]. Each time the vidicon target is read, a new file is created. Since the field map table defines the order in which pixels are read, it also defines the format of the file.

2.2.1 Byte String, Record, and File Format

A byte string is the series of 8-bit bytes obtained by reading the pixels from a row on the vidicon target. Each byte string is identified by appending three bytes of unique identifying information called a key. A byte string and its appended key are collectively called a keyed record (record) [1]. The format of a record is shown in Table IV.

The key contains:

1. The address of the row from which the data were obtained.
2. The column address of the pixel within the row corresponding to the first byte in the string.
3. The number of bytes in the string.

Records are packed within a 128-byte memory buffer, herein called the data buffer, prior to being transferred to disk. The data buffer contents are written onto disk

Table IV. Record Format

Byte Number	Content	
1	Row Address	Key
2	Column Address	
3	Byte Count (N)	
4	Byte #1	
5	Byte #2	
6	Byte #3	
.	.	
.	.	
.	.	
N + 3	Byte #N	

into one sector. The process of filling the buffer and writing its contents onto disk is called sector writing. A flowchart of the sector writing operation is shown in Fig. 11.

Each sector might be packed with several records or only a portion of a record depending upon the length of the byte string. Any portion of a record remaining after a sector has been filled is written onto disk starting at the beginning of the next sector.

Because records are written sequentially into a series of sectors, the locations of the records corresponding to each field are specified by the location of the first record in the sequence. Subsequent records are located with reference to the first. The location of the first record is specified by a sector and track number collectively called the physical block address (PBA) and an offset pointer called the physical block offset (PBO) [1]. The PBA specifies the sector in which the record begins and the PBO points to the location within this sector of the first byte of the record.

The PBA's and PBO's and the corresponding field identification numbers are maintained in a file map table (FMT) [1], the format of which is shown in Table V. The FMT is an integral part of each file and is written onto disk into the next sequential sector following the file.

The end of the FMT is marked by a null, or zero, field identification number. The absence of the null field

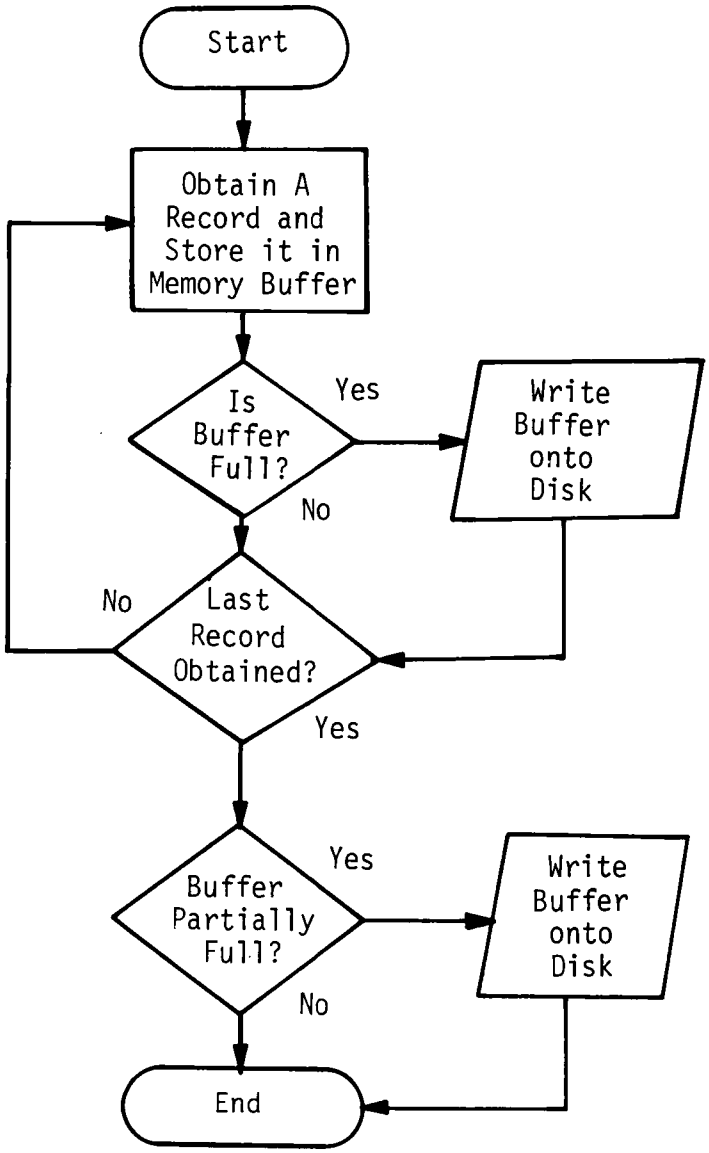


Figure 11. Flowchart of sector writing operation.

Table V. File Map Table Format

Byte Number	Content
1	Field Identification (1)
2	Sector (1)
3	Track (1)
4	Offset (1)
5	Field Identification (2)
6	Sector (2)
7	Track (2)
8	Offset (2)
.	.
.	.
.	.
4N - 3	Field Identification (N)
4N - 2	Sector (N)
4N - 1	Track (N)
4N	Offset (N)
4N + 1	Null (marks end of directory)

number indicates that the FMT contains its maximum of 32 entries. Limiting the size of the FMT to 32 entries allows the FMT to be written into one sector.

2.2.3 File Writing Technique

The process of writing a file onto disk by means of a series of sector writing operations is called file writing. A unique file writing technique was developed for MIIVSS to maximize the rate of data acquisition and storage.

The transfer of the data buffer's contents to disk consists of two parts--filling the disk's storage buffer and writing this buffer's contents onto the disk. A flow-chart of this process is shown in Fig. 12. MIIVSS transfers the data buffer's contents to the disk buffer and issues a command to write the disk buffer's contents onto the disk in a preselected sector. The disk's read/write mechanism moves the read/write head to the desired track and sector and writes the data onto the disk. The operation can require from 9.2 to 175.9 milliseconds for those cases in which the read/write head is positioned over the desired track. The timing is determined by noting:

1. Filling the disk buffer requires approximately 3.2 milliseconds.
2. Writing the data into a sector requires approximately 6.0 milliseconds once the sector is located.

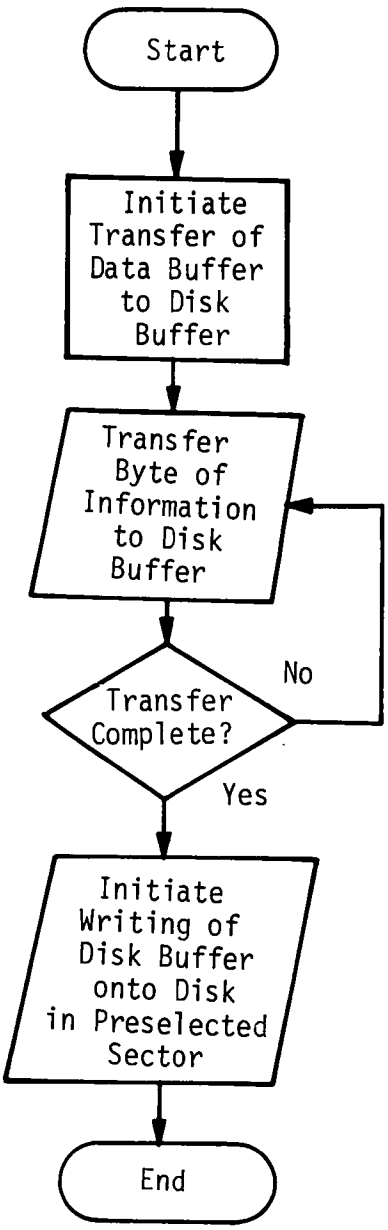


Figure 12. Flowchart of transfer of data buffer to disk.

3. One disk revolution requires approximately 166.7 milliseconds.

Simultaneous filling of the disk buffer and writing of its contents onto disk is not possible with this floppy disk. Therefore, adjacent sectors cannot be filled during the same disk revolution. MIIVSS fills every second sector during one revolution of the disk and the remaining sectors during the next revolution. While the skipped sector passes the read/write head, the disk buffer is filled and its contents are written into the next sector. Sector writing cannot be performed at a rate faster than this because one sector is the minimum number which can be skipped. Therefore, this file writing technique provides the fastest possible data storage rate and saves approximately 160.7 milliseconds per sector over the simple method of filling adjacent sectors.

Once a track has been filled, MIIVSS moves the read/write head to the beginning of the next sequential track. The starting sector for this track is determined from a list of starting sectors. This list, shown in Table VI, was obtained by allowing for skipping six sectors between the end of one track and the beginning of the next. Skipping sectors in this manner allows enough time for the read/write head to move to the next track and avoids taking one full revolution to begin at sector one of each track. Therefore, this technique allows one track to be filled in approximately 2.2 disk revolutions instead of 3.0

Table VI. List of Starting Sectors

Track Number	Starting Sector
0, 17, 34, 51, 68	1
1, 18, 35, 52, 69	4
2, 19, 36, 53, 70	9
3, 20, 37, 54, 71	14
4, 21, 38, 55, 72	19
5, 22, 39, 56, 73	24
6, 23, 40, 57, 74	2
7, 24, 41, 58, 75	5
8, 25, 42, 59, 76	10
9, 26, 43, 60	15
10, 27, 44, 61	20
11, 28, 45, 62	25
12, 29, 46, 63	3
13, 30, 47, 64	8
14, 31, 48, 65	13
15, 32, 49, 66	18
16, 33, 50, 67	23

revolutions as required by starting with sector one. This gives a time savings of approximately 131 milliseconds per track.

The starting sector list is circular [1] and has 17 entries. After the first 17 tracks have been filled, the list is repeated for the next 17 tracks and so forth, for all 77 tracks. The first and last entries in the list are of such a nature that the six-sector skip is accounted for in this transition.

The technique of skipping sectors dictates a minimum time of 12 milliseconds for writing into one sector on the disk. This is necessary because two sectors must pass the read/write head for each sector written. Within this time period, MIIVSS must also pack the data buffer with vidicon data. To allow a maximum amount of time for this, MIIVSS begins acquiring the data immediately after the command is issued to the disk read/write mechanism to write the contents of the disk buffer onto disk. Since transferring the contents of the data buffer to the disk buffer requires only 3.2 milliseconds, this procedure allows approximately 8.8 milliseconds for filling the data buffer. MIIVSS can obtain these data in approximately 3.5 milliseconds, thus allowing time to spare. It is therefore obvious that the acquisition of data from the vidicon system incurs no overhead because it occurs concurrently with the disk writing operation.

Prior to starting the file writing procedure, MIIVSS performs several operations including the following:

1. Check the field map table to assure that it is defined properly.
2. Check disk space to assure that there is enough room for another file to be added to the disk.
3. Pick up the file identifier and determine whether it is a legal identifier.
4. Initialize the file map table and the data buffer.

The file identifier consists of four 12-bit words and is used to identify the file on disk. The file identifier can have any format the user desires but cannot be the null, or zero, identifier which is treated as a special case by MIIVSS.

A flowchart of the file writing technique sequence is shown in Fig. 13. At the beginning of this sequence, MIIVSS performs a "home-up" operation which consists of reading the sector which is four preceding the first sector of the file. This brings the read/write head "close" to the first sector of the file and assures that the timing of the file writing sequence is as nearly as possible the same for every file.

Then, to properly start the data acquisition and storage sequence, the disk read/write mechanism is initiated to read the sector which is two preceding the first. While this sector is being read into the disk

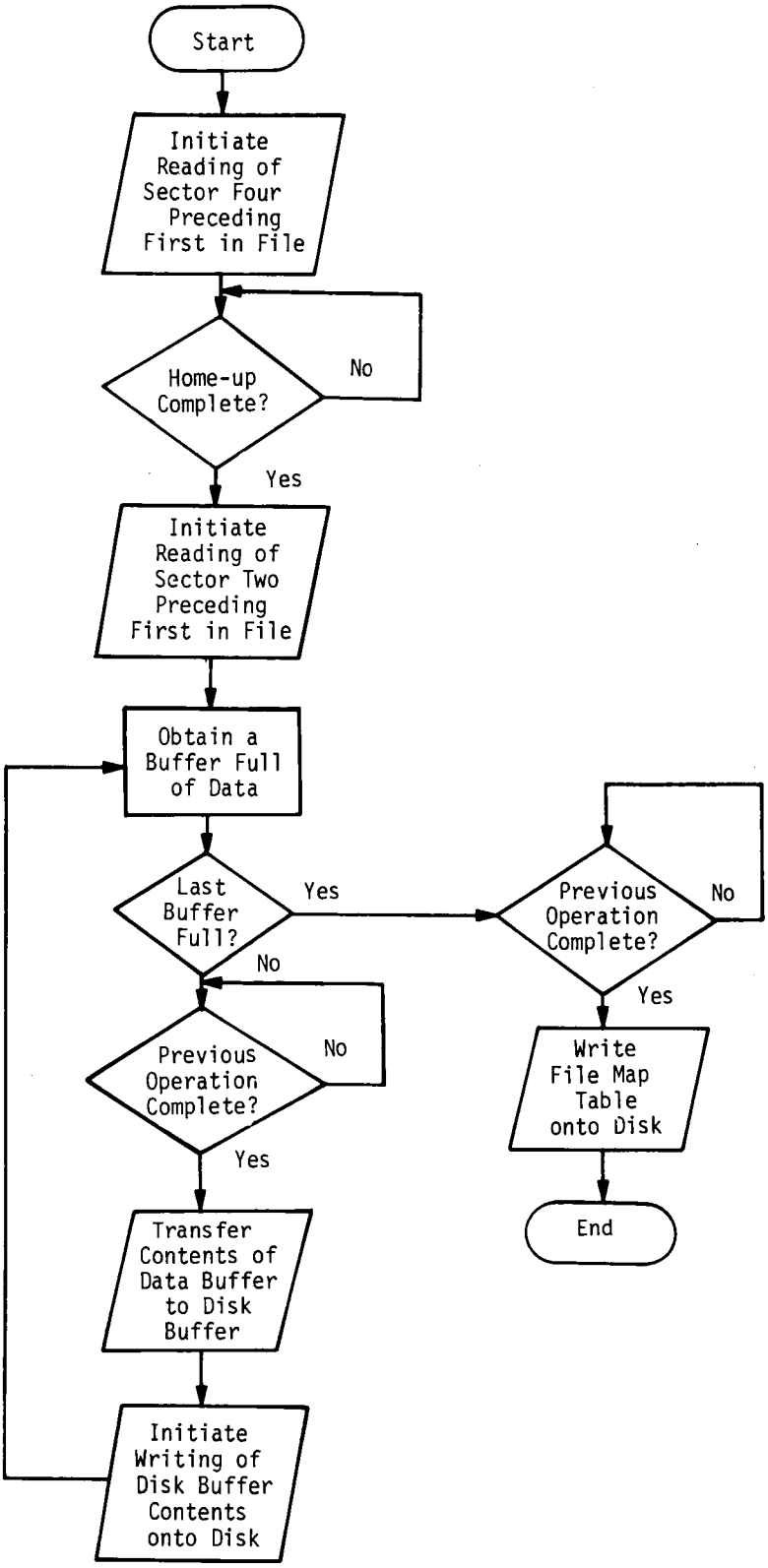


Figure 13. Flowchart of file writing technique sequence.

buffer, MIIVSS obtains the first data from the vidicon target and fills the data buffer. After the disk read operation is finished, MIIVSS transfers the first data to the disk buffer and commands the disk's read/write mechanism to write the data onto the disk when the first sector of the file passes the read/write head. While the writing operation is occurring, MIIVSS obtains the next portion of vidicon data and fills the data buffer. This sequence continues until all vidicon data have been acquired.

Whenever a field is read, the field's location is added to the FMT. At the end of the data acquisition sequence, the FMT is written onto the disk in like manner and marks the end of the file.

With this file writing technique, the data from the entire vidicon target may be read and stored onto disk in approximately 7.0 seconds. With the simple technique of writing sequentially into adjacent sectors and starting each track with sector one, the same amount of data can be read and stored in approximately 80.0 seconds. Therefore, the specially designed file writing technique has more than an order of magnitude higher throughput rate than the simple technique.

2.2.4 Field Reading

Data are read from the vidicon target one field at a time and are stored in the data buffer to be written onto

the disk. The field reading operation is invoked by the file writing operation for filling the data buffer. A flowchart of the field reading operation is shown in Fig. 14.

At the beginning of the field reading operation, the field map table entry-used counter is cleared to initialize the table for use. The first field specification is then retrieved from the table to define which pixels are contained in this field. The first record's key is then obtained from the field specification and is stored in the data buffer to mark the beginning of the first record.

The row address, column address, and byte count defining the first row are then output to the computer interface and the first data conversion is initiated. The end-of-conversion flag is then monitored, and the data word is input and stored in the data buffer when the flag becomes set. The instruction which inputs the data word also initiates the next conversion in sequence, thereby eliminating the need to specify each pixel's address. Each pixel after the first in the row is automatically read and the resultant data word is input when the end-of-conversion flag becomes set.

The end-of-count flag contained in each data word is checked upon input to determine when the entire row has been read. When the end of the row has been reached, MIIVSS checks to see if the end of the field has been

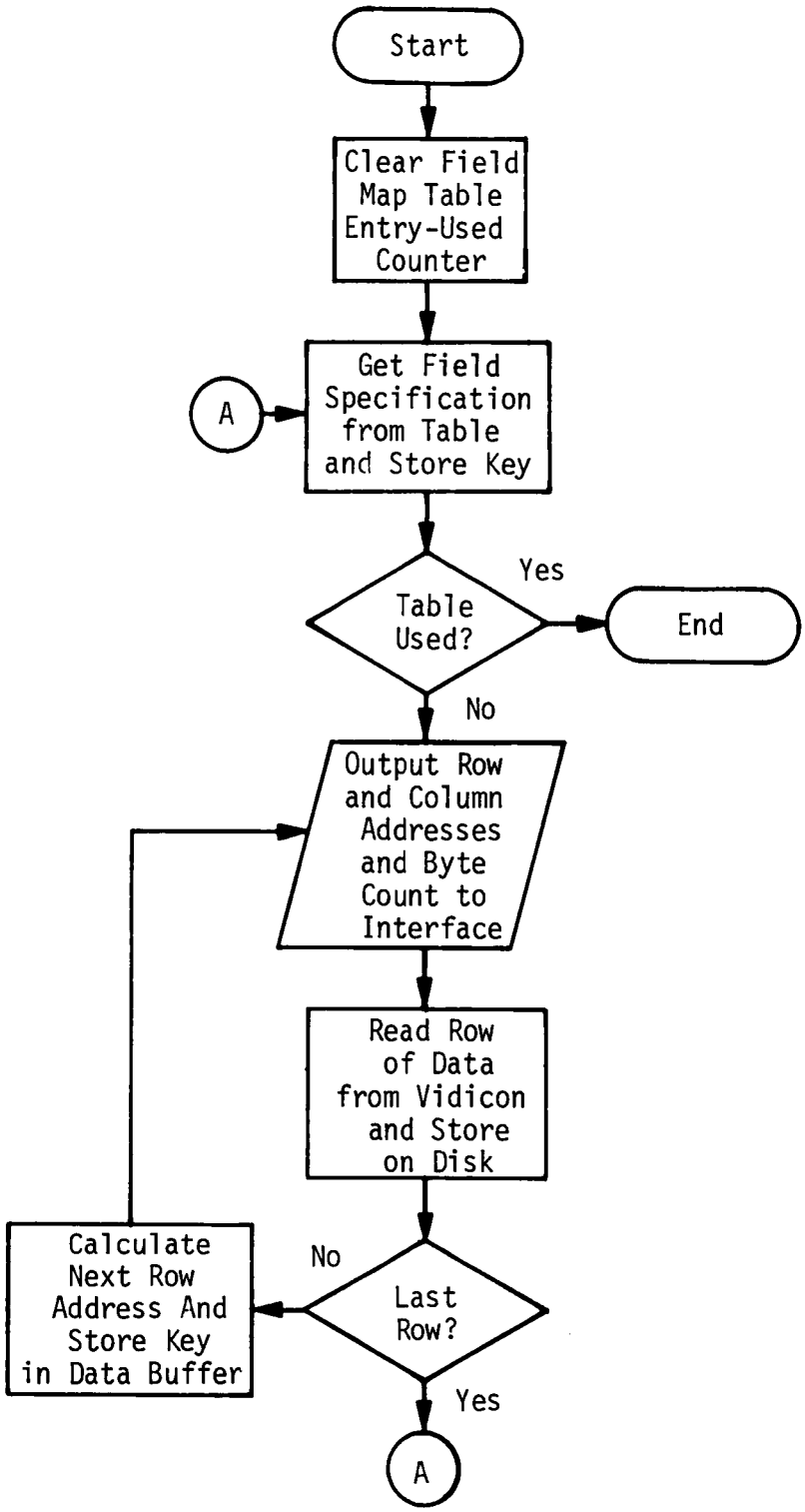


Figure 14. Flowchart of field reading operation.

reached. If not, the row spacing increment is added to the current row address to determine the address of the next row to be read. The old row address in the key is then replaced by the new address and the resultant key is stored in the data buffer. This row is then read as before. When the end of a field has been reached, MIIVSS moves to the next field and this continues until all fields have been read.

Each time a field specification is obtained from the field map table the entry-used counter is incremented. Whenever the entry-used counter becomes equal to the entry counter, all entries have been used from the table and, therefore, every field has been read.

Whenever the data buffer becomes full, the field reading software returns control to the file writing software which then writes the contents of the data buffer onto disk. Control is then returned to the field reading software for filling the data buffer again. This alternating of control continues until all of the data have been input from the vidicon and stored on disk.

2.3 FILE DIRECTORY

MIIVSS maintains the location of each file on the disk in a file directory [1] which is also maintained on disk. Each file's location is specified by a pointer which points to the file's FMT.

The file directory is divided into chained blocks [1], each of which fits into one sector on the disk. The format of a directory block is shown in Table VII. The first word of the block contains an entry counter. The next 60 words are used for storage of up to 10 entries. Words 62 and 63 contain chain pointers [1], herein referred to as linkages, which are used to connect the directory blocks. Word 64 indicates the number of empty sectors remaining on the disk for data storage.

A directory entry requires six words of storage--four words for the file identification and two words for the track and sector of the FMT. Each entry is added at the end of the block and the entry counter is incremented to reflect this addition.

The end of the file directory is marked by a null entry which contains the null file identifier and the track and sector of the beginning of the empty space on the disk. Each entry added to the directory takes the place of the null entry, and a new null entry is added at the end of the directory to point to the new beginning of the empty space. The amount of disk space used by the newly created file is deducted from word 64 of the last directory block to reflect the decrease in empty disk space. A flowchart of file directory updating is shown in Fig. 15.

Each directory block is kept in memory until it is full and then it is written onto the disk. Sector 26 on each track is reserved for storing the directory blocks.

Table VII. File Directory Block Format

Word Number	Content
1	Entry Counter (N)
2	File Identification (1)
3	
4	
5	
6	Sector (1)
7	Track (1)
.	.
.	.
.	.
6N - 4	File Identification (N)
6N - 3	
6N - 2	
6N - 1	
6N	Sector (N)
6N + 1	Track (N)
.	.
.	.
.	.
62	Reverse Linkage
63	Forward Linkage
64	Number of Empty Sectors

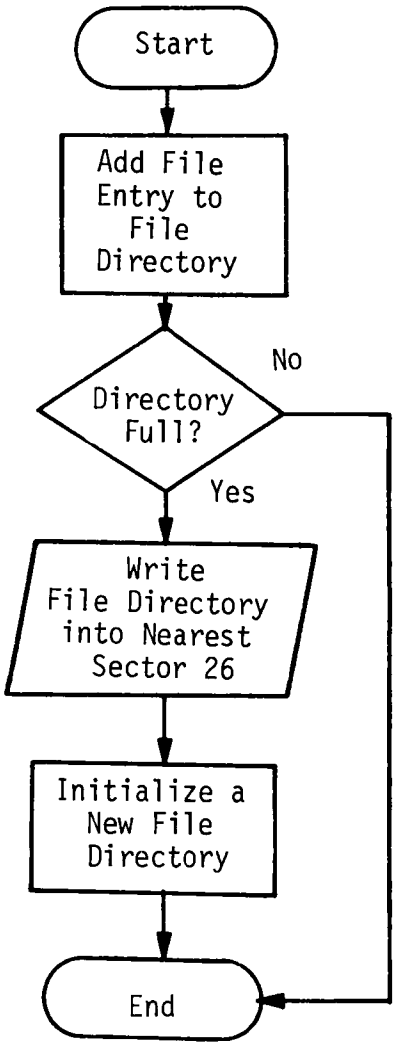


Figure 15. Flowchart of file directory updating.

When a directory block becomes full, it is written into sector 26 of the track over which the disk's read/write head is positioned. If this sector has been used previously, the block is written into the next empty sector 26. Writing the directory blocks into sectors 26 eliminates interference with the sequential file writing technique and keeps the read/write head positioned near the beginning of the empty data storage space.

The chaining [1] of these blocks together is accomplished by two linkages. An illustration of block linkage is shown in Fig. 16. The "reverse" linkage points to the track containing the preceding block and the "forward" linkage points to the track of the succeeding block. Before a directory block is written onto disk, the reverse linkage is set to point to the preceding block and zero is put into the forward linkage to indicate that it is unknown. The forward linkages are set in the "close-file-for-write" operation.

2.3.1 The Close-File-for-Write Operation

A flowchart of the close-file-for-write operation is shown in Fig. 17. The forward linkage in the last directory block is set to zero to indicate that this is the end of the directory, and the block is written onto the disk. The number of the track into which this block is written is saved and the preceding directory block pointed to by the reverse linkage is read into memory. The forward

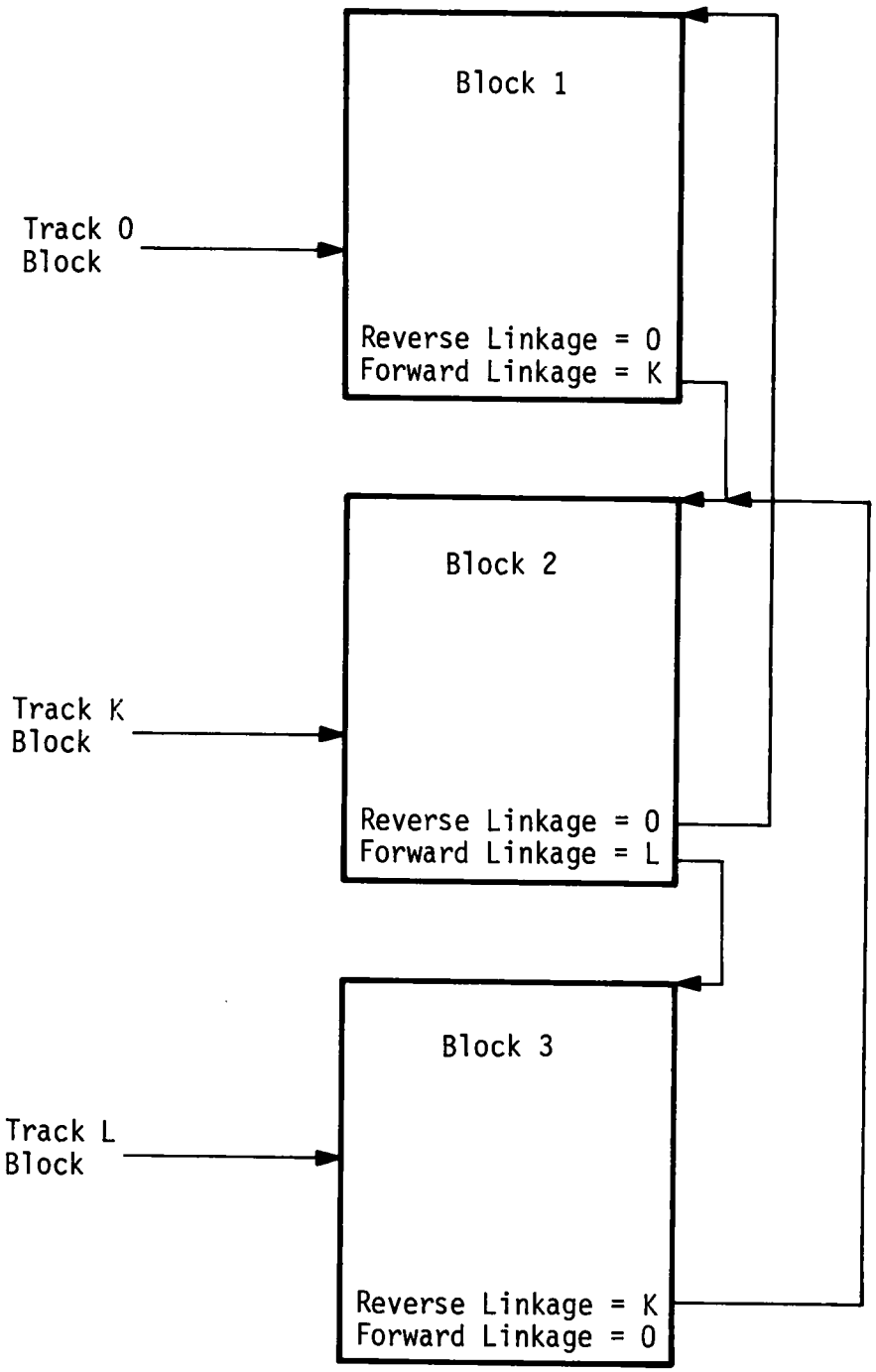


Figure 16. Directory block linkage for a three-block directory.

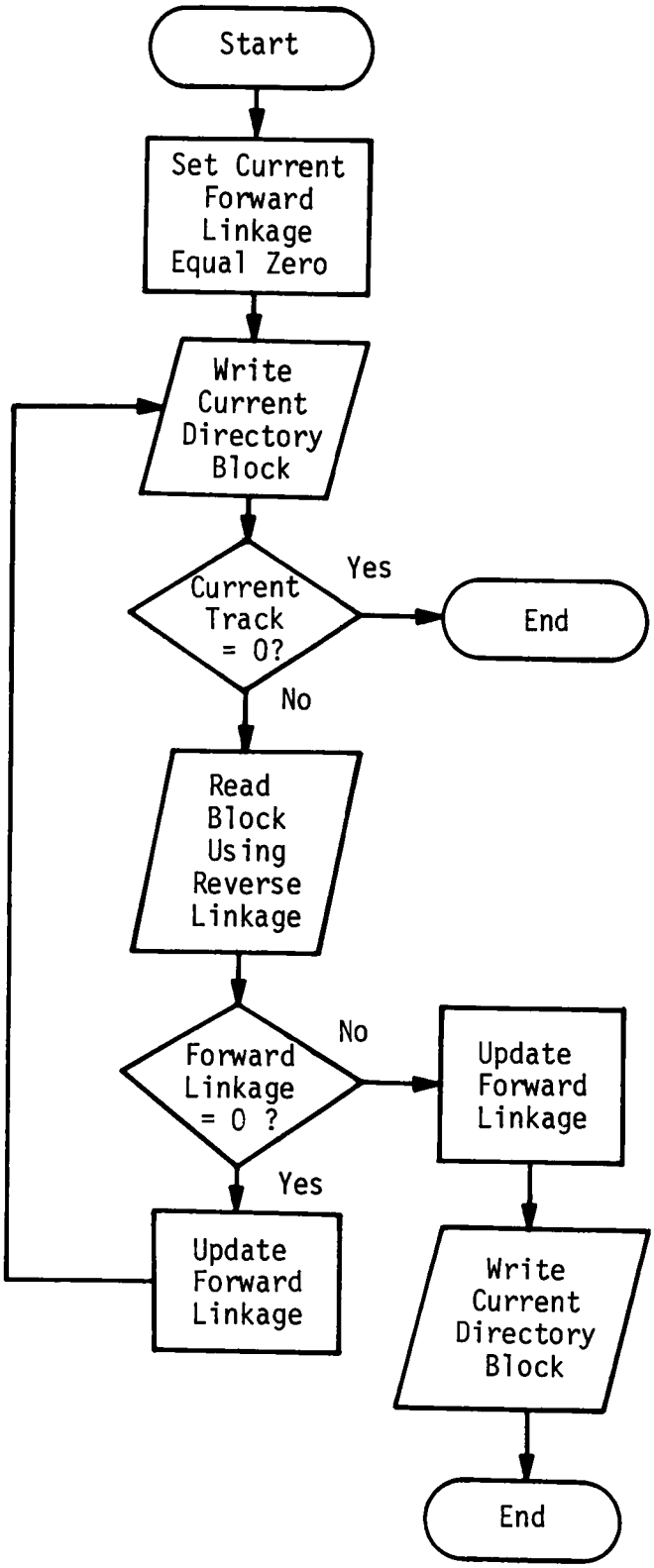


Figure 17. Flowchart of close-file-for-write operation.

linkage in this block is then filled with the track number of the last directory block and the block is rewritten into its original location. This operation is repeated until the block in track zero has been updated. The block in track zero might not always contain directory entries, but is always used to mark the beginning of the directory.

For those cases in which the directory has been closed previously and reopened using the "open-file-for-write" operation, not all of the forward linkages must be updated. Linkage updating is performed only until a non-zero forward linkage is detected and then this linkage is changed to reflect the new location of its succeeding block.

2.3.2 The Open-File-for-Write Operation

The open-file-for-write operation is used for reading the last directory block into memory. A flowchart of the open-file-for-write operation is shown in Fig. 18. MIIVSS performs the open-file-for-write operation by first reading the directory block from track zero to get the location of the next block. The next block is read and its forward linkage is used in reading the next succeeding block. This continues until the forward linkage is zero, indicating that the last directory block has been read.

MIIVSS maintains a memory word as an indicator of whether a file is open. This word is set equal to -1 when the file is opened for the write operation and is cleared when the file is closed. This word is called a key [1] and

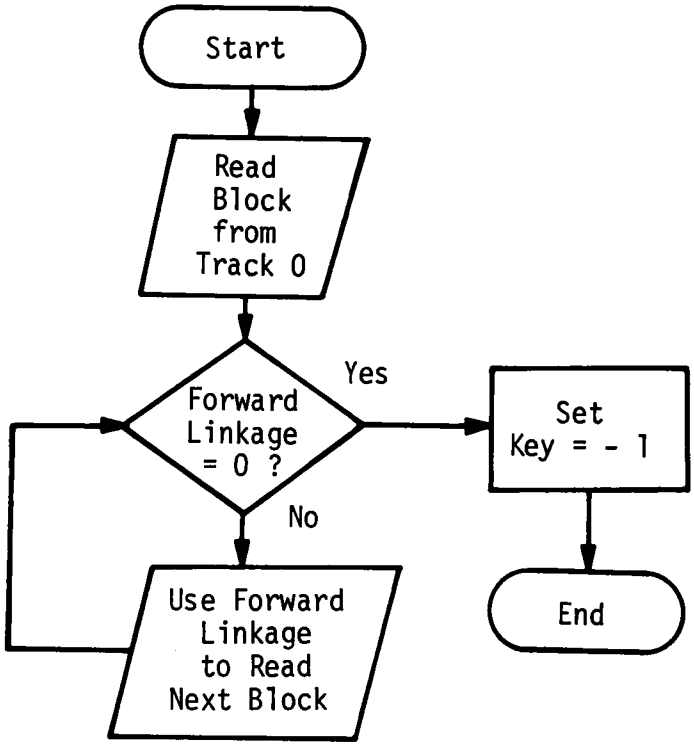


Figure 18. Flowchart of open-file-for-write operation.

is used to prevent the software from performing file addition and directory updating should the file directory not be opened properly.

2.3.3 Opening and Closing Files for Data Retrieval

Opening a file for data retrieval is called the open-file-for-read operation. A flowchart of this operation is shown in Fig. 19. The objective of the open-file-for-read operation is to read into computer memory the FMT of some specific file for use in locating the data in that file. The file identification number is sought in the file directory and the location of the FMT is thereby obtained. The FMT is read into memory and the status of the key is set to +1 to indicate that the open-file-for-read operation has been performed.

The key may be 0, +1, or -1 to indicate that all files are closed, a file has been opened for the read operation, or a file has been opened for the write operation. The key has these different statuses because of the very different requirements of the two file-opening operations.

The close-file-for-read operation is very simple because there is no need to rewrite the FMT onto the disk. MIIVSS performs the close-file-for-read operation by merely setting the key status to zero. A flowchart of this operation is shown in Fig. 20. MIIVSS allows the user to avoid selecting which close-file operation to perform. The

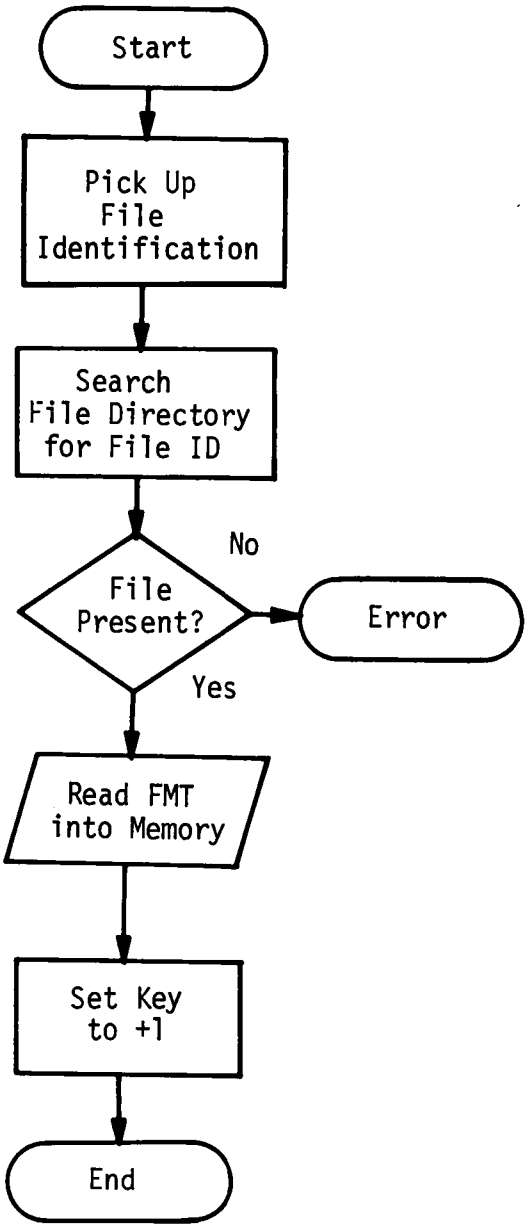


Figure 19. Flowchart of open-file-for-read operation.

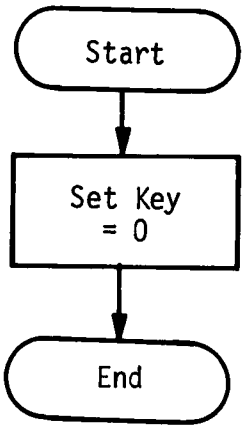


Figure 20. Flowchart of close-file-for-read operation.

user can merely request a close-file operation and MIIVSS performs the proper operation as directed by the key. A flowchart of the close-file operation is shown in Fig. 21.

2.3.4 Directory Initialization

Each unused diskette is devoid of information and must be initialized to contain a directory. A flowchart of directory initialization is shown in Fig. 22. Directory initialization is accomplished by writing a predefined directory block into sector 26 of track zero. This block is defined to contain only a null entry that points to sector five of track zero, which is the location of the beginning of the data storage area. The forward and reverse linkages are both set equal to zero to indicate that this is the only block in the directory. Word 64 is set equal to 1923 to indicate that there are that many unused sectors in the data storage area.

2.3.5 File Manipulation

Files are written onto the disk in a sequential manner. Therefore, the only file manipulations allowed are the addition of new files or the deletion of the last file written onto the disk. A flowchart of file deletion is shown in Fig. 23. File deletion involves removing the last entry in the directory and reclaiming the space used by the file. This is accomplished by replacing the last directory entry with a null entry that points to the beginning of the deleted file. Word 64 of this directory block is then

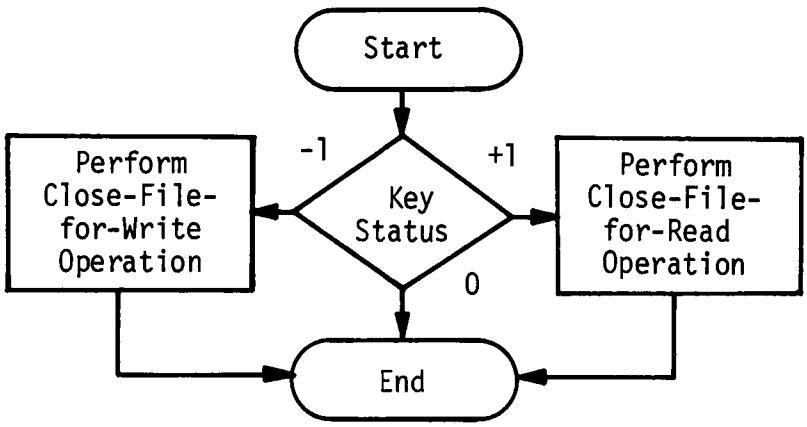


Figure 21. Flowchart of close-file operation.

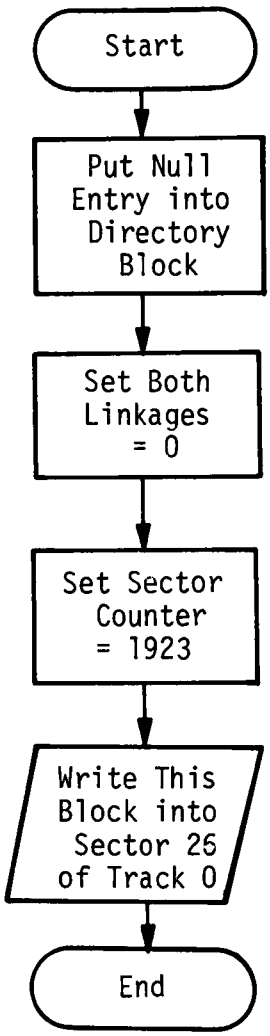


Figure 22. Flowchart of file directory initialization.

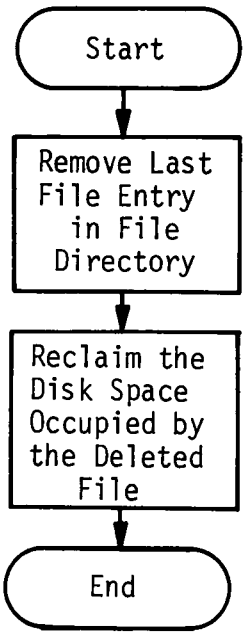


Figure 23. Flowchart of file deletion.

updated to reflect that the space formerly used for this file is available for reuse.

2.4 DATA RETRIEVAL

MIIVSS retrieves the data from the disk one byte string at a time and any method of working with the data is allowed if it can be implemented by the recall of individual byte strings. A flowchart of the data retrieval procedure is shown in Fig. 24.

To specify a byte string, the user must identify the string with the following parameters:

1. The file identification number.
2. The field identification number.
3. The key of the record which contains the desired byte string.
4. A pointer to a string buffer in which the byte string is to be returned to the user.

The location of the record containing the byte string is determined by means of the file identification number, the field identification, and the record key. The file identification number is used in the open-file-for-read operation to read into memory the correct FMT. The field identification number is used to retrieve the PBA and PBO of the desired field from the FMT. A new PBA and PBO which point to the desired record are then calculated using the old PBA and PBO, the record key, and the field specification. The key of the accessed record is then retrieved

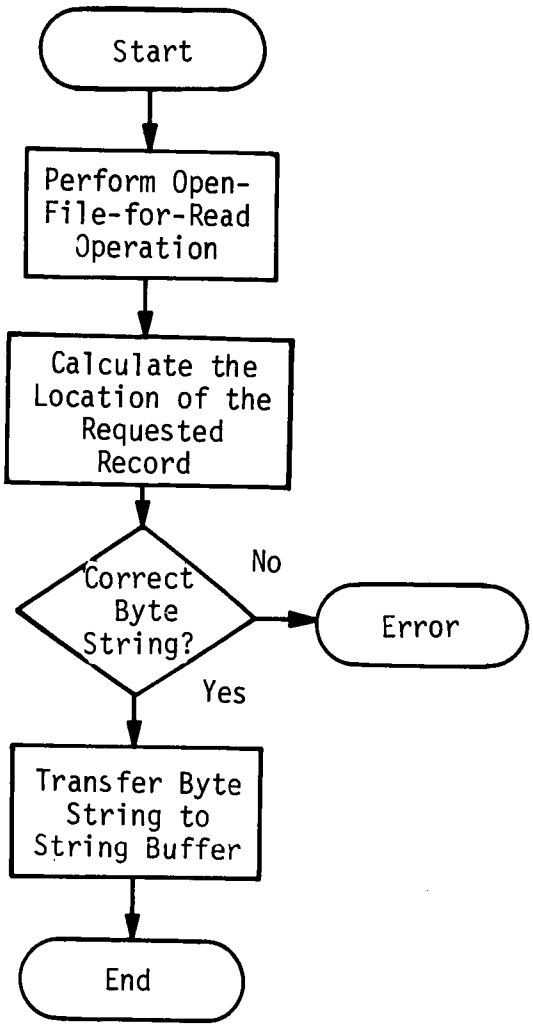


Figure 24. Flowchart of the data retrieval procedure.

and compared to the specified key to assure positive identification. If the two keys match, the byte string is sequentially read from disk and placed into the string buffer for return to the user. This method of record retrieval is called direct access [1].

2.4.1 Worklist

While performing an extended data retrieval operation, MIIVSS may have to open each file many times. Each time the file is opened, MIIVSS must determine the location of its FMT. The entire operation can, therefore, require many file directory searches. MIIVSS helps the user avoid repetitious directory searching by allowing him to set up a list of FMT pointers. This list is called the worklist. A flowchart of worklist setup is shown in Fig. 25, and the worklist's format is shown in Table VIII. Prior to data retrieval, the user can specify the files with which he intends to work. MIIVSS obtains the FMT pointers from the file directory and sets up the worklist. Then, during the open-file-for-read operation, MIIVSS checks the worklist to obtain the FMT pointer of the desired file. If the pointer is not in the worklist, it is obtained by a search through the file directory.

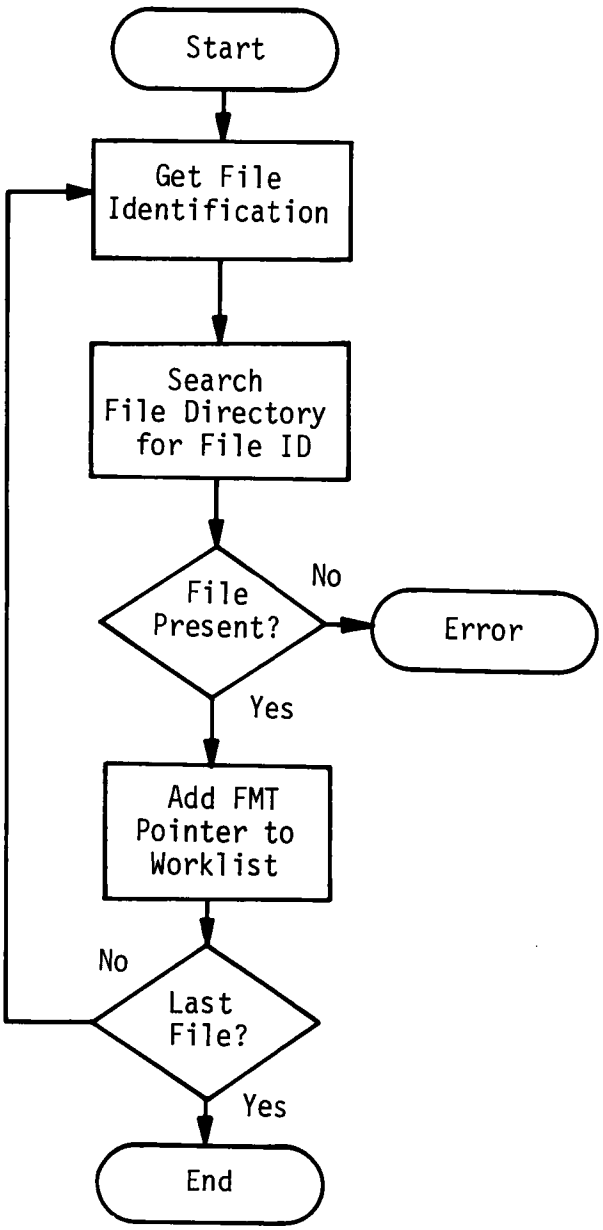


Figure 25. Flowchart of worklist setup.

Table VIII. Worklist Format

Word Number	
1	Entry Counter (N $\leq$ 4)
2	File Identification (1)
3	
4	
5	
6	Sector (1)
7	Track (1)
8	File Identification (2)
9	
10	
11	
.	.
.	.
.	.
6N - 4	File Identification (N)
6N - 3	
6N - 2	
6N - 1	
6N	Sector (N)
6N + 1	Track (N)

2.5 ERROR TRACING

MIIVSS performs error tracing whenever an error is detected. The error tracing facility helps the user determine the nature of any error that should occur.

Because each MIIVSS routine may be called by other MIIVSS routines, the error tracing technique provides the user a means to trace through each level of subroutine nesting. MIIVSS does this by printing a series of error messages. The routine in which an error is first detected prints an error message on the line printer to indicate what happened. This routine indicates to its calling routine that an error occurred. The calling routine likewise prints an error message and indicates to its calling routine that an error has occurred. This continues through all levels of nesting. Each error message consists of the text "ERROR #" followed by the error number. The user must look for this number in a list which contains possible error numbers and their meanings. The series of error messages thus obtained gives an indication of what the error was and where it occurred. The list of possible error numbers is given in Appendix C.

2.6 TEST OPERATIONS

Setup and adjustment of the vidicon system requires certain modes of operation which allow the user to monitor

the system's status. MIIVSS provides three test modes which are as follows:

1. Reading one row on the target continually and displaying the results on the oscilloscope (see Fig. 26).
2. Reading all rows stepping through them one at a time and displaying the results on the oscilloscope (see Fig. 27).
3. Erasing the vidicon target under Teletype control (see Fig. 28).

MIIVSS also allows target erasure under total software control without input from the Teletype. MIIVSS allows the user to initiate a multicycle erase and to terminate erasure by entering the single-cycle erase mode.

2.7 DATA PLOTTING

MIIVSS provides a limited data-plotting capability by displaying the data in analog form on the oscilloscope. A flowchart of the data display operation is shown in Fig. 29. The user merely loads the data into a buffer and gives MIIVSS the location of the buffer and the number of data bytes in the buffer. MIIVSS picks up the data and plots the data via the computer interface.

Each data byte, which must be in the decimal range 0 to 255, is picked up and converted to its physical address equivalent. The physical address is then loaded into the row address counter-register which is used for

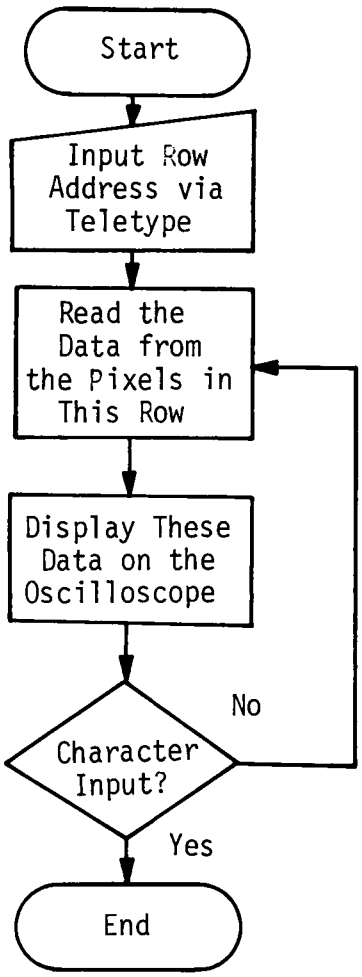


Figure 26. Flowchart of row-reading test operation.

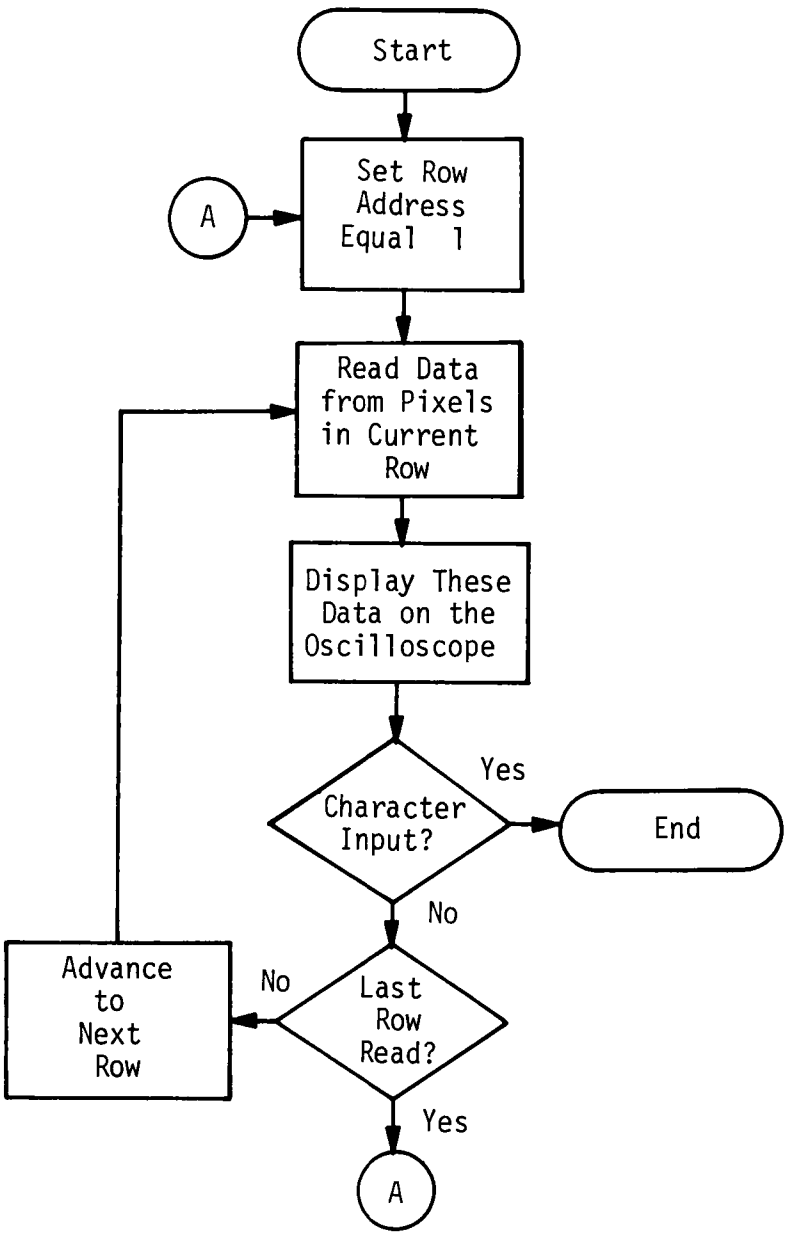


Figure 27. Flowchart of target-reading test operation.

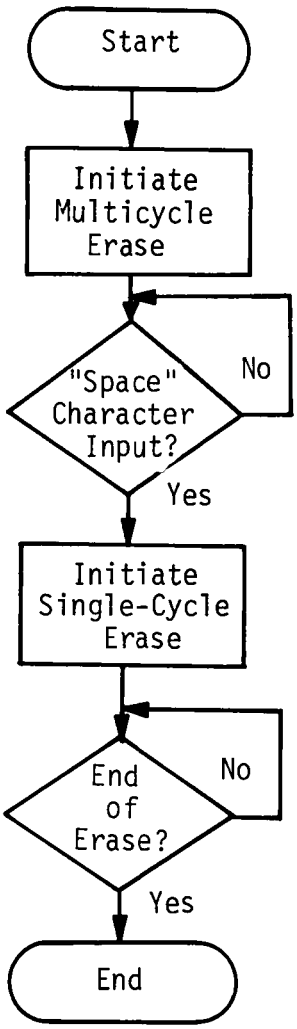


Figure 28. Flowchart of erasure test operation.

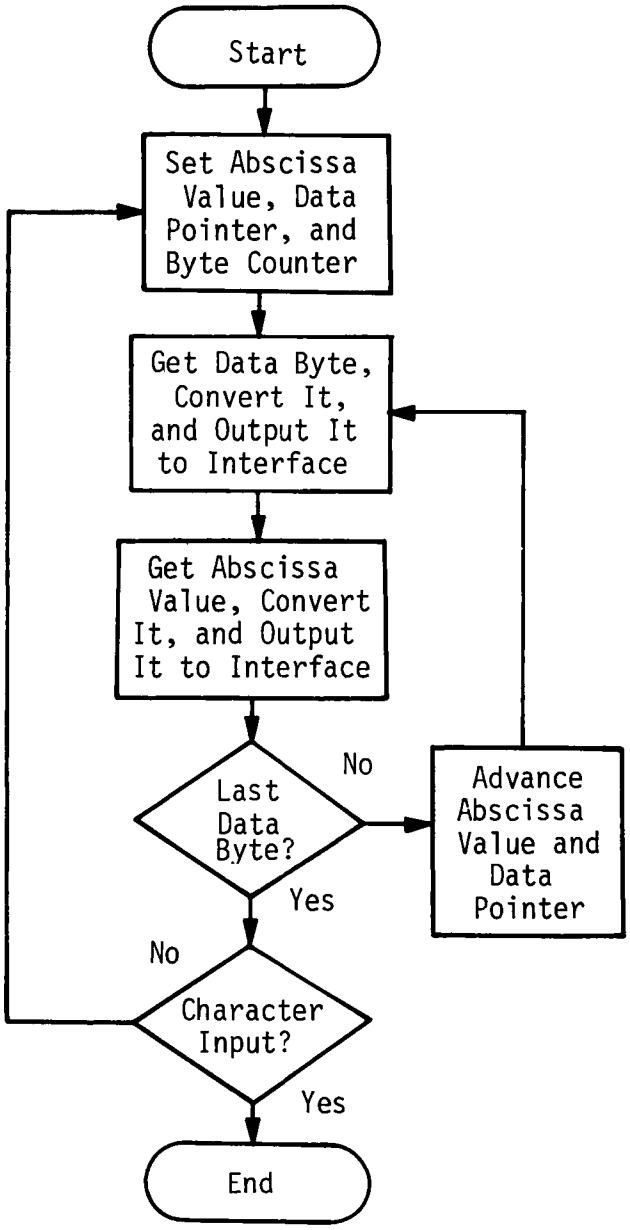


Figure 29. Flowchart of data display on oscilloscope.

plotting the ordinate values of the display. The corresponding abscissa value is converted to its physical address equivalent and is loaded into the column address counter-register. The net effect of this is that one point is plotted on the oscilloscope. The plotting of points continues until all data values in the buffer have been plotted and then the plot is repeated. Repetition of the plot continues until any key on the Teletype keyboard is typed. Repetition of the plot causes the points, which are not stored on the oscilloscope screen, to become visible.

2.8 MEMORY REQUIREMENTS

MIIVSS requires 4571 words of memory. This includes 731 words for storage buffers and 3840 words for routines.

CHAPTER III

AN APPLICATION OF THE VIDICON SYSTEM

3.0 INTRODUCTION

An important application of the vidicon system is rocket plume diagnostics, which requires the determination of the temperatures and densities of gases in a rocket nozzle exhaust plume at various radial and longitudinal positions.

The temperatures and densities of the gases are obtained using the electron beam fluorescence technique. This technique deals with the interpretation of the light emitted when the molecules in a gas are "excited" by a narrow beam of high-energy electrons. This electron beam passes through the gas exciting the molecules by collision, causing them to reach higher energy states. When the molecules spontaneously decay to their normal states, they emit light characteristic of each kind of gas and of the temperature of the gas. The intensities of the light emitted at various wavelengths by the excited molecules are characteristic of the density of the gases. The distribution of the radiant intensities over a range of wavelengths is called a spectrum and each kind of molecule has its own characteristic spectrum. The various spectra

of emitted light can be interpreted to give the temperatures and densities of the gases in the rocket plume.

3.1 TEST CHAMBER AND ROCKET

The vidicon system was first used to acquire electron beam fluorescence data from the exhaust plume of a rocket engine mounted in the Aerospace Chamber 10V at the Arnold Engineering Development Center. An illustration of the 10V chamber is shown in Fig. 30. This chamber is capable of simulating high altitude and deep space conditions. The rocket engine was a pulsed, bipropellant rocket mounted on a movable carriage which allowed the rocket to be moved so that different plume positions could be viewed by the stationary optics system. The chamber and rocket had their own control and monitoring instrumentation which was used to fire the engine, monitor firing, and monitor the status of the chamber. The instrumentation also provided timing and synchronization signals which synchronized the vidicon-based data acquisition system to the firing of the engine.

3.2 HARDWARE SETUP

The vidicon system was used in conjunction with a double monochromator spectrometer to obtain the spectra of the light emitted from the rocket gases. An illustration of the spectrometer/vidicon setup is shown in Fig. 31. The spectrometer grating spread the light out on the vidicon

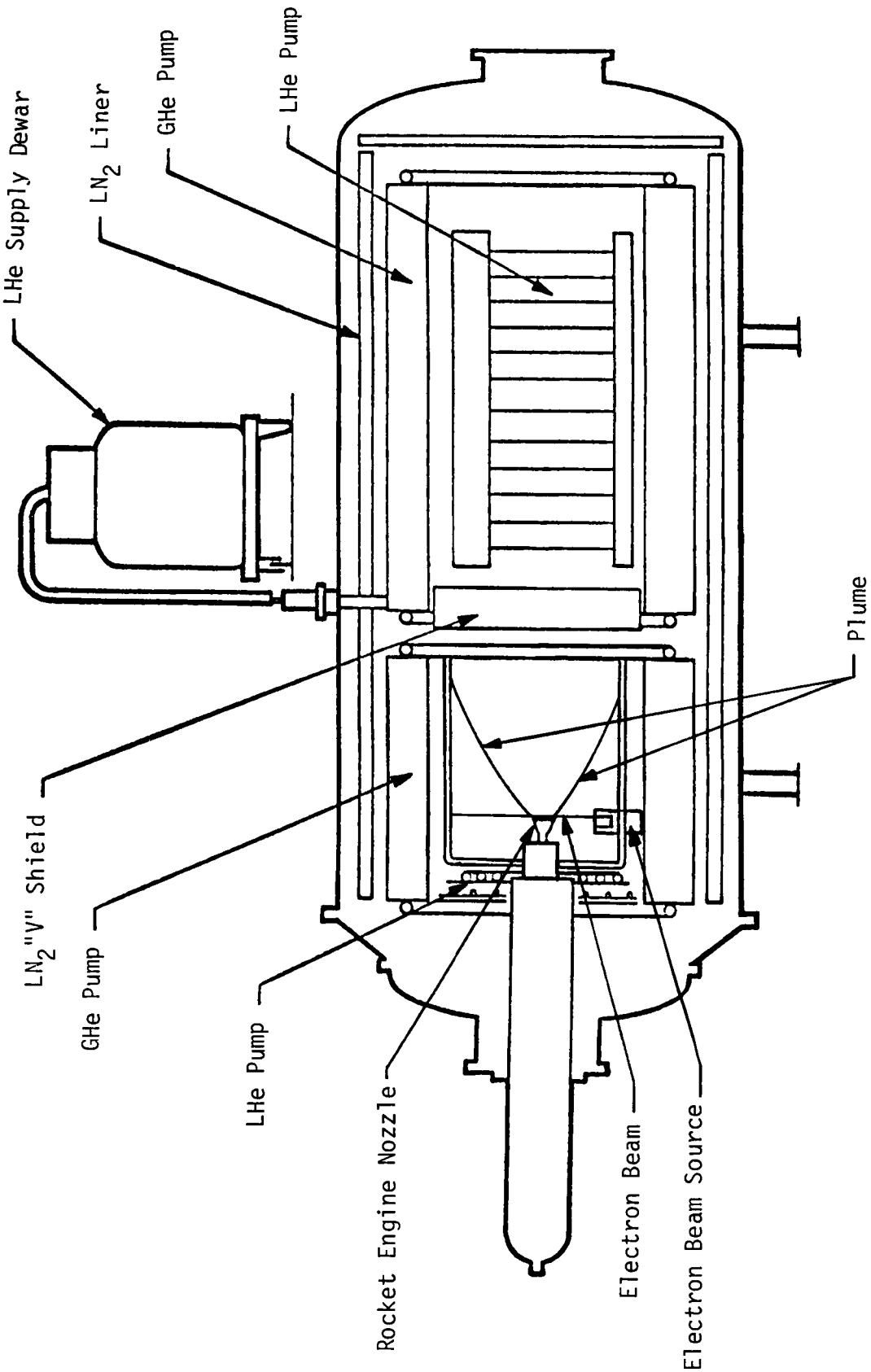


Figure 30. Aerospace Chamber 10V.

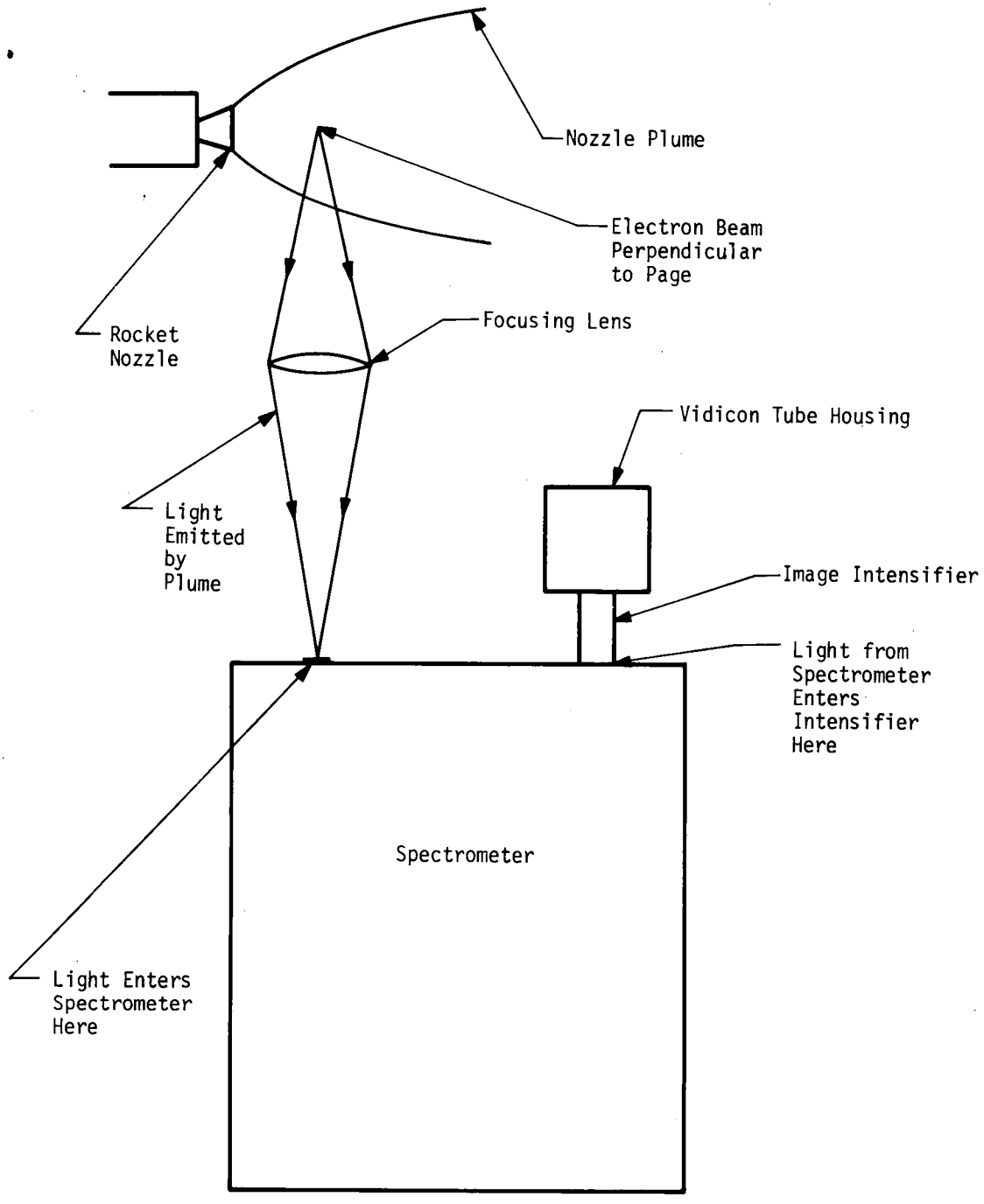


Figure 31. Setup of the spectrometer and vidicon system.

target so that the light was distributed across the target according to wavelength. Spectral variation was along each row of pixels. The amount of charge in each pixel corresponded to the intensity of the light at one wavelength.

An optics system was designed to transmit the light to the vidicon target in such a manner that each row on the target corresponded to a unique radial position in the rocket plume. The optics system focused on a small portion of the electron beam passing through the plume; its area of focus was represented by a short line segment parallel to and coincident with the electron beam.

Therefore, a variation along a row of pixels resulted in a variation in wavelength and a variation from row to row resulted in a variation in spatial position.

The vidicon was placed at the exit slit of the spectrometer with the slit removed. This allowed spectral intensities to be measured over a small range of wavelengths giving the vidicon system the ability to scan those wavelengths which could be imaged onto the target. This was sufficient for obtaining spectra of the light within predetermined regions of interest. Different regions of the spectrum were selected depending on the particular gas which was to be monitored and were positioned on the vidicon using the spectrometer wavelength selection mechanism. A typical region of interest spanned a range of approximately 40 Å, giving a system resolution of better than 0.5 Å.

The data acquisition system also included:

1. A variable power supply which was used for selecting the gain of the image intensifier.
2. A gating device which turned on the image intensifier at the appropriate times.
3. Timing and gating circuitry that synchronized the data acquisition system to the firing of the engine.
4. Electron beam instrumentation that produced and controlled the electron beam.
5. Auxiliary A/D converters and digital counters for acquiring data parameters other than those of the vidicon system.

3.3 APPLICATION SOFTWARE FUNCTIONS

The software written for this application is called the application software. The purpose of the application software was to tie together all of the system components into a functional, efficient system.

The application software was controlled by a simple monitor routine by which functional commands were input via the Teletype keyboard. The functions provided by monitor commands were:

1. Perform the data acquisition sequence consisting of synchronizing the computer and the firing of the rocket engine and acquiring data at appropriate times.

2. Control the setup, output, and disk storage and retrieval of the field map table.
3. Call MIIVSS to initialize a new diskette.
4. Call MIIVSS to perform file deletion.
5. Close files via calls to MIIVSS.
6. Perform data retrieval and display via calls to MIIVSS.
7. Invoke test operations via Teletype inputs.
8. Adjust the spectrometer wavelength setting.

3.4 FIELD MAP TABLE SETUP

Prior to starting the data acquisition sequence, the field map table was set up to define thirteen fields on the vidicon target. Due to the optical setup, these fields corresponded to thirteen positions in the plume and covered a length of approximately one and one-half inches to match the nozzle exit diameter, as illustrated in Fig. 32.

Each field used in this application contained four adjacent rows each consisting of 248 pixels. Each row started with the eighth pixel. The fields were placed symmetrically about one field which was in the middle of the target. The fields were spread across the target with a spacing of sixteen rows. The fields were numbered sequentially with the field having the lowest row address being identified as field one.

The maximum amount of data which can be stored on a diskette was important when determining the definition of

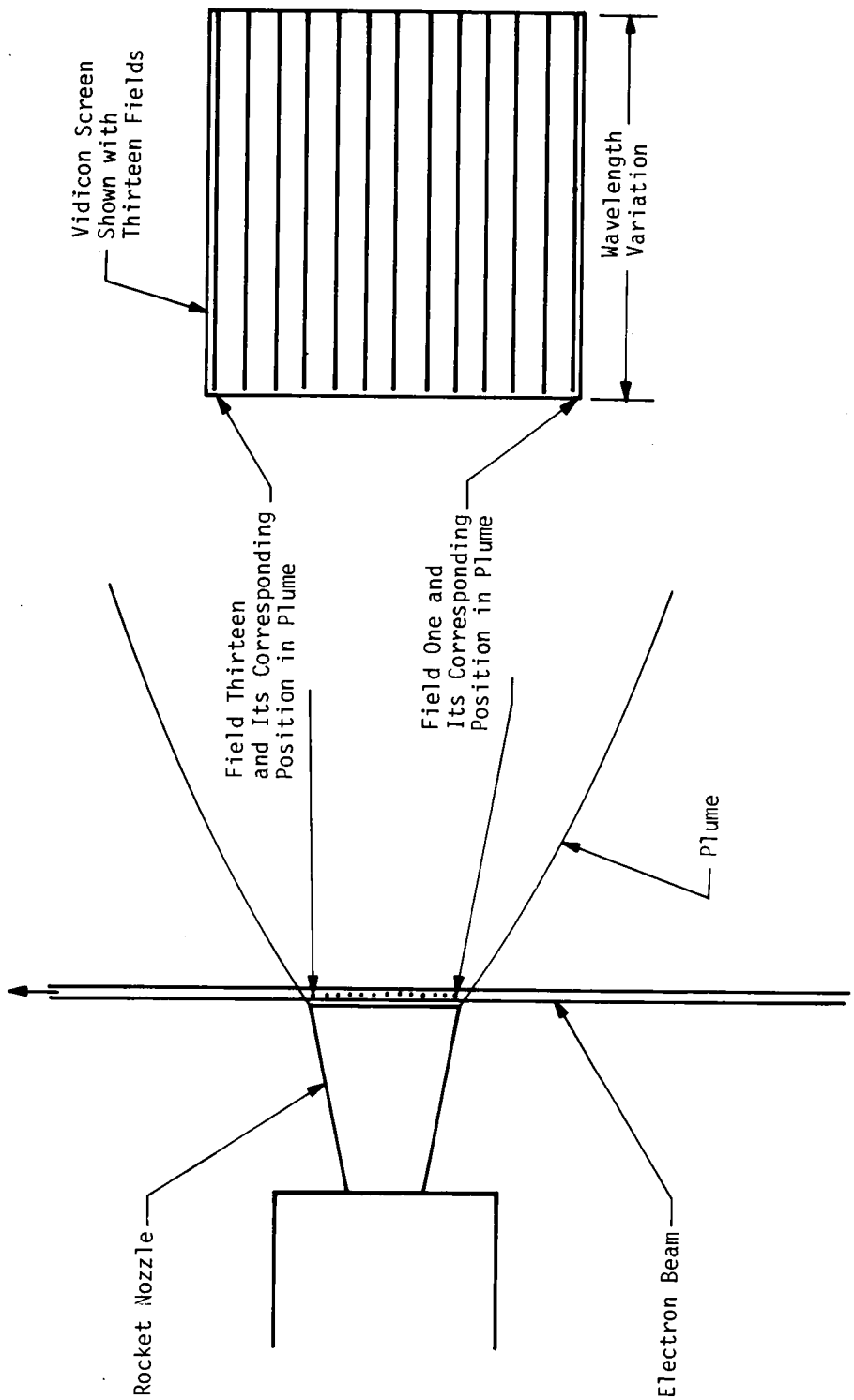


Figure 32. Illustration of correspondence between plume positions and fields.

the field map. Thirteen fields with the current configuration represented a compromise between monitoring as many positions in the plume as possible and conserving disk space by limiting the number and size of fields.

3.5 DATA ACQUISITION

Data acquisition consisted of taking data with the electron beam on and then with it off. The latter condition produced background data representative of the light which was not directly attributable to the gas molecules excited by the electron beam; the background also contained some noise due to charge buildup on the vidicon target. During data reduction, the background data were subtracted from the electron beam data to give a spectrum representative of the light emitted by the excited gas molecules.

A flowchart of the data acquisition sequence is shown in Fig. 33. The first step in the sequence was a setup procedure which included the following:

1. Setting the spectrometer's wavelength selector.
2. Setting the image intensifier's gain.
3. Inputting the run number and number of engine pulses.
4. Turning on the electron beam.
5. Initiating multicycle erasure.

The application software then waited for a switch on the computer console to be set. At that point, the erasure was terminated and engine firing began. The

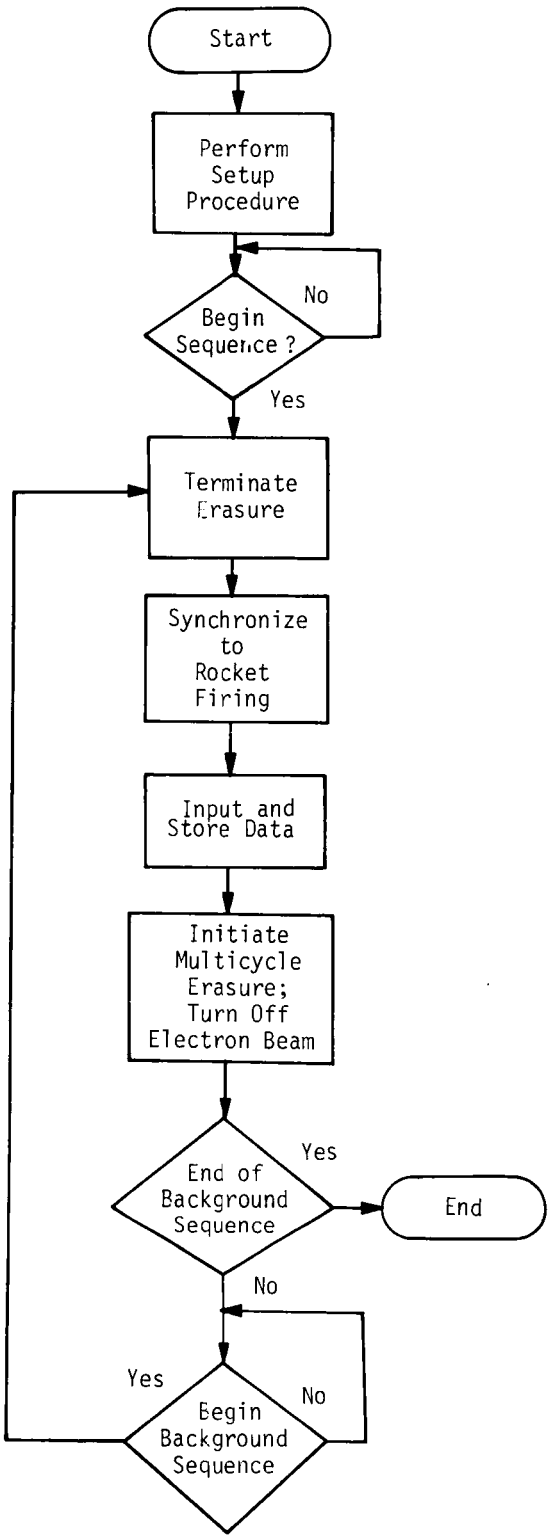


Figure 33. Flowchart of data acquisition sequence.

firings continued until the preset number of pulses had been reached. The vidicon data and the auxiliary data parameters were then read and stored. Auxiliary parameters included (a) electron beam voltage, (b) engine temperature, (c) chamber pressure, and (d) elapsed time. These were stored on magnetic tape.

The multicycle erasure was then restarted and the electron beam was turned off by the operator in preparation for the background sequence. Erasure of the target was maintained at least 30 seconds to prepare the target for the background data.

The background sequence was then performed in like manner by hitting the keyboard space bar to restart the sequence with the termination of erasure. The end of the background sequence marked the end of the data acquisition sequence.

During the data acquisition sequence, the application software relied upon MIIVSS to perform the following operations:

1. Vidicon erasure.
2. Opening and closing of files.
3. Reading the vidicon and storing the data on disk.

The two files of data thus obtained were identified by using the run number and the ASCII character "D" for the electron beam data file and ASCII "B" for the background file.

During each engine firing pulse, the image intensifier was turned on to admit light to the vidicon target. The number of times the engine was fired was based on the amount of light produced in the rocket plume. When the intensities were low, a large number of pulses were required. The light levels observed during this particular application were large enough that a single one-second firing was sufficient.

The amount of time required in this application to read the data from all defined fields on the target and store the data on disk was slightly less than two seconds. This held the noise buildup to a worst-case value of approximately 2 percent of saturation.

3.6 DATA REDUCTION

The data reduction routines in the application software relied upon MIIVSS for retrieving the data. Since MIIVSS can retrieve only one row of data at a time from the disk, the application software was designed to do the same.

During data reduction, the application software retrieved one row of data from the data file and stored it in a memory buffer. Then the same row was requested from the corresponding background file and it was stored in another memory buffer. The contents of these two buffers were then subtracted from one another and the results were added to the contents of a third memory buffer; the third

buffer maintained a running total of the data corrected for background.

Addition of background-corrected data was performed on all four rows of one field or on all rows of one field from several different runs. For example, the totals for the four rows of field 13 of run 100 were added to the totals from field 13 of run 101. The total was then divided by the total number of rows to produce an average spectrum that was more representative of the actual spectrum than that produced by one row. Within each individual row, the effects of random noise could obscure the spectrum making it inaccurate. The spectrum produced by the averaging process was less obscure because the random noise averaged to a value close to zero over several rows.

The retrieval/reduction process was quite fast. As an illustration, the data from nine runs were averaged in approximately 60 seconds. In this case, 18 files were involved and one field at a time was averaged. Each of the four rows in the desired field of the first run were background corrected and the results were added to the running total which had previously been cleared. Then the four rows of the same field of the next run were background corrected and added to the total. This continued for all nine runs. Within the 60-second time period, the following operations were performed:

1. The file directory was searched once for each file for a total of 18 directory searches.
2. Each file was opened four times for a total of 72 open-file-for-read operations.
3. Each row was background corrected giving a total of 36 background corrections.
4. Each background-corrected row was added to the running total giving 36 addition operations.
5. Each file was closed four times giving a total of 72 close-file-for-read operations.
6. A total of 72 records was retrieved from the disk.

The application software allowed the reduced spectrum to be printed on the line printer or displayed on the oscilloscope. The application software handled the printing of the data in its entirety, but MIIVSS was called to provide the display function. An illustration of a spectrum display generated by MIIVSS on the oscilloscope is shown in Fig. 34.

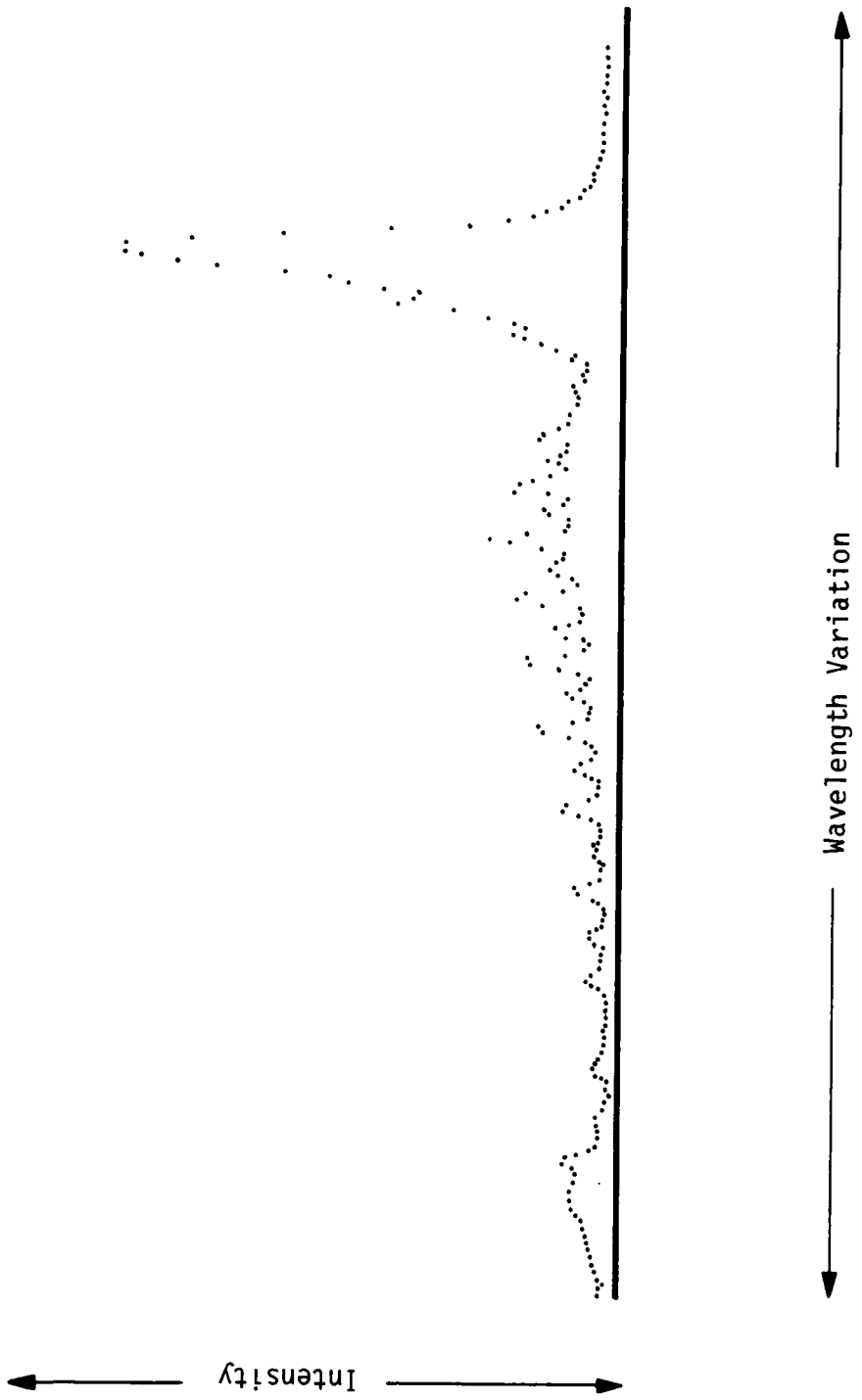


Figure 34. An example of a typical reduced-spectrum display generated by MIIVSS.

CHAPTER IV

SUMMARY AND CONCLUSIONS

Operating system software has been designed, coded, and implemented for a minicomputer-based image-intensifier-vidicon data acquisition system (vidicon system). The vidicon system was designed and used to acquire large quantities of data in a short period of time. High-speed data acquisition was required in using the system because of the amount of data produced and because noise that can obscure the data builds up on the vidicon target. The vidicon system was designed to be adaptable and flexible so that it can be used in a variety of applications. These same attributes were prime considerations in the design of the minicomputer software.

Adaptability was achieved by a nucleus of systems routines called the Minicomputer-Based Image-Intensifier-Vidicon System Software (MIIVSS) that allows the user to avoid the details of the operation of the vidicon system. MIIVSS controls and monitors all operations of the vidicon system and acquires and stores the vidicon data on floppy disk for later retrieval. MIIVSS was designed to be independent of any particular application and of any specific data acquisition and reduction technique.

Flexibility of data acquisition was achieved by reading the data from the vidicon target in rectangular

fields. Patterns with more complex geometries can be enclosed within fields for data acquisition purposes and the superfluous data can be discarded during data reduction. The use of rectangular fields allows easy setup of the location algorithm used during the reading of data from the vidicon target.

Flexibility was also allowed in the naming of files and fields and also in the order of calls to MIIVSS routines. This flexibility permits the user to identify his data by whatever means required and to acquire his data by whatever scheme best suits his needs.

High-speed data storage was attained by developing a special disk writing technique. The floppy disk slows down the storage of data, but this problem was minimized by writing into alternating sectors and assigning each track its own starting sector. This technique achieved a data transfer speed that was fast enough to minimize the problems of noise buildup. The use of rectangular fields was a contributing factor in the success of this technique.

The utility of the vidicon system was illustrated by an actual application in which temperature and density measurements in a rocket engine exhaust were obtained by means of an electron beam fluorescence technique. The vidicon system was used in conjunction with a spectrometer to get spectra of the light emitted by the molecules of the exhaust gases excited by an electron beam. These spectra

were subsequently used to calculate the temperatures and densities of the exhaust gases.

The vidicon system software was designed using both standard and nonstandard software techniques. Many of the techniques implemented in MIIVSS have general applicability and others are peculiar to the vidicon system. The choice of techniques was based on the most appropriate solutions to the problems encountered in the vidicon system. The final set of systems routines met the requirements set forth for the software.

BIBLIOGRAPHY

BIBLIOGRAPHY

1. Madnick, Stuart E., and John J. Donovan. Operating Systems. New York: McGraw-Hill Book Company, 1974.

2. Carruthers, George R. "Electronic Imaging for Space Science and Applications," Astronautics and Aeronautics, XV:56-58, October, 1977.

3. Introduction to Programming. Fourth edition. Maynard, Massachusetts: Digital Equipment Corporation, 1973.

4. PDP8/E, PDP8/M, and PDP8/F Small Computer Handbook. Maynard, Massachusetts: Digital Equipment Corporation, 1973.

5. Programming Languages. Second edition. Maynard, Massachusetts: Digital Equipment Corporation, 1972.

6. LA180 DECprinter I User's Manual. First edition. Maynard, Massachusetts: Digital Equipment Corporation, 1975.

7. RX8/RX11 Floppy Disk System Maintenance Manual. Second edition. Maynard, Massachusetts: Digital Equipment Corporation, 1975.

8. M1705 OMNIBUS Output Interface. Maynard, Massachusetts: Digital Equipment Corporation, 1973.

9. M1703 OMNIBUS Input Interface. Maynard, Massachusetts: Digital Equipment Corporation, 1972.

APPENDIXES

APPENDIX A

LIST OF VIDICON SYSTEM COMPONENTS

1. An image intensifier with a gain-selection amplifier.
2. A General Electric Z7975A silicon-target vidicon tube with video-processing instrumentation and a refrigeration unit.
3. A specially designed computer interface.
4. An oscilloscope display.
5. A Digital Equipment Corporation PDP-8/f minicomputer equipped with an LA180 line printer, a Teletype, an RX8E Floppy Disk System, an M1703 input interface module, and an M1705 output interface module [3 through 9].

APPENDIX B

THE FLOPPY DISK SYSTEM

The mass storage device used in this system is a Digital Equipment Corporation RX8 Floppy Disk System consisting of an RX01 drive and an RX8E interface for the PDP-8/f minicomputer. The RX8 is a random-access, mass-memory device that stores data on flexible diskettes as opposed to the hard-surface disks normally used.

Each diskette is logically divided into 77 concentric circles called tracks; these are numbered from 0 to 76. Each track is subdivided into 26 fixed-length blocks called sectors; sectors are numbered from 1 to 26. Each sector holds one hundred and twenty-eight 8-bit bytes or sixty-four 12-bit words. Between sectors 1 and 26 on each track is a space approximately two sectors in length called the "pre-index gap." This gap coincides with a "hard-index mark" generated when a hole in the diskette passes over a detector mechanism; this technique is used by the system for checking the diskette's position.

The RX8 system allows data storage in two formats--8-bit bytes or 12-bit words. In the 12-bit word mode, 64 words are written into each sector giving a total capacity of 128,128 words per diskette. In the 8-bit byte mode, one hundred and twenty-eight 8-bit bytes are written into each sector giving a total capacity of 256,256 bytes.

In the 12-bit word mode, zeroes are appended to each word to give a total of 16 bits for each word written onto the diskette; this 16-bit word is split into two 8-bit bytes for storage purposes. The two modes of operation may be mixed as needed.

The time required for the disk system to service a data transfer request is 23 microseconds for the 12-bit word mode or 18 microseconds for the 8-bit byte mode. There is no maximum time in which the host computer must service a data request, thereby allowing asynchronous operation between the computer and the disk system.

The floppy disk system is composed of two primary components--a disk read/write mechanism and a buffer memory used to temporarily store data being transferred to or from the disk. To write onto the disk, the computer must fill the buffer memory with data and issue a "write-on-disk" command to the disk read/write mechanism. The read/write mechanism then writes the data onto the disk from the buffer. To read data from the disk, the computer must issue a "read-from-disk" command; this loads the requested data into the buffer memory. The computer may then transfer the data from the buffer memory to its internal memory. Filling or emptying the buffer cannot occur concurrently with the write-on-disk or read-from-disk operations.

Given a track number and a sector number, the RX8 system moves its read/write head to the requested track and

performs an operation on the sector. When an error occurs during disk operation, the RX8 system determines the cause and signals the computer that an error has occurred. The computer may then interrogate the system to get explicit error information [7].

The specifications of the floppy disk system are listed in Table IX.

Table IX. Specifications of the RX8 Floppy Disk System

System Reliability	
Minimum Number of Revolutions per Track	1 million/media (head loaded)
Seek Error Rate	1 in 10^6 seeks
Soft Read Error Rate	1 in 10^9 bits read
Hard Read Error Rate	1 in 10^{12} bits read
Drive Performance	
Capacity	256,256 bytes or 128,128 words per diskette
	3,328 bytes or 1,664 words per track
	128 bytes or 64 words per sector
Data Transfer Rate	
Diskette to Controller Buffer	4 microseconds/data bit
Buffer to CPU Interface	2 microseconds/bit
CPU Interface to I/O Bus	18 microseconds/byte
	23 microseconds/word
Track-to-Track Move	10 msec/track minimum
Head Settle Time	20 msec maximum
Rotational Speed	360 rpm ± 2.5 percent;
	166 ms/rev nominal
Recording Surfaces per Disk	1
Tracks per Diskette	77
Sectors per Track	26
Recording Technique	Double frequency
Bit Density	3,200 bpi at inner track
Track Density	48 tracks/inch
Average Access Time	488 msec

APPENDIX C

LISTING OF ERROR CODES AND THEIR MEANINGS

Error No.	Explanation
0001	An error occurred while performing a disk-drive initialization operation.
0002	A parity error occurred while reading an error message from the disk controller.
0003	An error occurred while transferring data from the disk buffer to the computer memory.
0004	An error occurred while transferring data from the computer memory to the disk buffer.
0005	An error occurred in reading the "home-up" sector.
0006	An error occurred while reading a file directory block from the disk into the disk buffer.
0007	An error occurred while transferring a file directory block from the disk buffer to the computer memory.
0008	Drive 0 failed to find its home position during the initialization operation.
0009	An error occurred in reading a file map table from the disk into the disk buffer.
0010	An error occurred while transferring a file map table from the disk buffer to the computer memory.

Error No.	Explanation
0011	An error occurred while reading data from the disk into the disk buffer.
0012	An error occurred while transferring data from the disk buffer to the computer memory.
0013	An error occurred while transferring a file directory block from the computer memory to the disk buffer.
0014	An error occurred while writing a file directory block onto the disk from the disk buffer.
0015	The end of the disk space allocated for file directory storage has been reached.
0017	An error occurred while trying to write a file directory block onto the disk.
0018	Disk space allocated for data storage has been completely filled.
0019	An error occurred while writing data onto the disk from the disk buffer.
0020	The end of disk space allocated for data storage was reached while writing data onto the disk.
0021	An error occurred while transferring data from the computer memory to the disk buffer.
0022	An error occurred while writing data onto the disk from the disk buffer.
0023	An error occurred while transferring a file map table from the computer memory to the disk buffer.

Error No.	Explanation
0024	The disk controller found the home position while stepping out ten tracks during the initialization operation.
0025	An error occurred while writing a file map table onto the disk from the disk buffer.
0026	The end of disk space allocated for data storage was detected after a file map table was written onto the disk.
0027	The field map table was determined to be full while attempting to add more field specifications to it.
0028	The specified field identification was not found in the field map table while attempting to transfer that field's specification to the control block.
0029	An error occurred while transferring the field map table from the computer memory to the disk buffer.
0030	An error occurred while writing the field map table onto the disk from the disk buffer.
0031	An error occurred while reading the field map table from the disk into the disk buffer.
0032	An attempt was made to access a track numbered greater than 76.
0033	An error occurred while transferring the field map table from the disk buffer to the computer memory.

Error No.	Explanation
0034	The end of disk space allocated for data storage was reached while attempting the retrieval of a byte of data from the disk.
0035	An error occurred while reading the next sector of data from the disk during the data byte retrieval operation.
0036	The specified file identification was not found in the file directory.
0037	An error occurred while attempting to retrieve a file directory block from the disk.
0038	The entry counter for the file directory block just read from the disk is zero.
0039	A file was already open when an attempt was made to perform the open-file-for-write operation.
0040	The home position was detected by the disk controller before the desired track was reached.
0041	An error occurred during the disk drive initialization operation preceding the open-file-for-write operation.
0042	During the open-file-for-write operation, a zero directory block counter was encountered.
0043	An error occurred while attempting to read a file directory block from the disk during an open-file-for-write operation.

Error No.	Explanation
0044	An attempt was made to open the null file during the open-file-for-read operation. The null file identification is not allowed for general use but is reserved for use by the systems software.
0045	An error occurred in the file directory search during the open-file-for-read operation.
0046	The specified file identification was not found in the file directory during the open-file-for-read operation.
0047	An error occurred while attempting to read a file map table from disk during the open-file-for-read operation.
0048	A self-diagnostic error occurred within the disk controller.
0049	A file was already open when an attempt was made to perform the open-file-for-read operation.
0051	During the open-file-for-write operation, the last entry in the file directory was determined to be other than the null entry. While adding the null entry, it was determined that the end of disk space allocated for data storage had been reached.
0052	During the close-file-for-write operation, the last file directory block was lost due to an error that occurred while attempting to write this directory block onto the disk.

Error No.	Explanation
0053	An error occurred when an attempt was made to read a directory block from the disk during the close-file-for-write operation.
0054	An error occurred when an attempt was made to write a directory block onto the disk during the close-file-for-write operation.
0055	An error occurred while writing data onto the disk during the file writing operation.
0056	The disk controller could not find the requested sector after looking at 52 headers on the disk.
0057	The open-file-for-write operation had not been performed when an attempt was made to create a file of data.
0058	An empty field map table was encountered when an attempt was made to create a file of data.
0059	It was determined that not enough disk space remained for adding another file of data.
0060	An attempt was made to identify a file using the null identification. This is not allowed.
0061	An error occurred while performing the "home-up" operation.
0062	An error occurred while attempting to write data onto the disk during the file writing operation.

Error No.	Explanation
0063	An error occurred while attempting to write the file map table onto the disk during the file writing operation.
0065	An error occurred while attempting to update the file directory following the file writing operation.
0066	The file map table was full when an attempt was made to add another entry to it during the file writing operation.
0067	An error occurred while attempting to write the file map table onto the disk during the file writing operation.
0068	An error occurred while attempting to write the file directory onto the disk during the file writing operation.
0069	The open-file-for-read operation had not been performed for the specified file prior to attempting to recall data from the file.
0070	The null field identification was specified during the recall of data from the specified file. The null identification is not allowed for general use but is reserved for use by MIIVSS.
0071	The specified field identification was not found in the field map table during the recall of data from the disk.

Error No.	Explanation
0072	The disk controller waited more than 40 micro-seconds and did not see a SEP clock.
0073	An error occurred while retrieving a string of data bytes from the disk.
0074	An invalid row address was sought while attempting to recall a string of data bytes from the disk.
0075	The end of disk space allocated for storage of data was reached while retrieving a string of data bytes from the disk.
0076	An error occurred while attempting to retrieve a string of data bytes from the data buffer.
0077	An error occurred while attempting to read a sector of data into the data buffer.
0078	An error occurred while attempting to retrieve a byte of data from the data buffer.
0079	The row address specified did not match the row address stored on disk for the requested byte string.
0080	The disk controller could not find a preamble.
0081	The column address specified did not match the column address stored on disk for the requested byte string.
0082	The number of bytes in the byte string requested did not match the number of bytes stored on the disk for the specified byte string.

Error No.	Explanation
0083	The specified field identification was not found in the field map table during the retrieval of field specifications from the table.
0084	A larger than allowable number of files was requested during the definition of the worklist table.
0085	A file was already open before an attempt was made to define the worklist.
0086	An error occurred while reading a directory block during the definition of the worklist.
0087	A file identification requested during the definition of the worklist was not present in the file directory.
0088	The disk controller found a preamble but no I/O mark was found within the allowable time span.
0089	An error occurred when an attempt was made to perform the disk drive initialization operation during the file deletion operation.
0090	An error occurred while attempting to read a file directory block during the file deletion operation.
0091	An error occurred while attempting to read a file directory block during the file deletion operation.
0092	An error occurred while attempting to perform the open-file-for-write operation during the file deletion operation.

Error No.	Explanation
0093	The end of disk space allocated for data storage was reached while adding the null entry to the file directory during the file deletion operation.
0094	An error occurred while attempting to perform the close-file-for-write operation during the file deletion operation.
0096	A CRC error occurred while looking at a header.
0104	The header track address of a good header does not compare with the requested track.
0112	There were too many tries to find an IDAM (header identifier).
0120	A data AM was not found in the allotted time.
0128	A CRC error occurred while reading a sector from the disk.
0136	A parity error occurred during operation of the disk.

VITA

John H. Jones was born on May 22, 1948, in Shelbyville, Tennessee. He attended schools in Bedford and Rutherford Counties and was graduated from Bedford County Central High School in 1966. He was graduated with honors from Middle Tennessee State University in 1970 with Bachelor of Science degrees in Physics and Mathematics. At Middle Tennessee State University he served as vice-president and later president of the Physics Club, was a member of the Phi Sigma Beta honor society, won the Physics Award for 1970, and was listed in "Who's Who Among Students in American Universities and Colleges."

He was married in 1972 to the former Carol Troxler. They have three children--Marla, Ben, and Evan.

In 1973 he became a math data aide with ARO, Inc., and was promoted to scientific computer programmer in 1975. His work beginning in 1973 involved programming a variety of computers for the acquisition and reduction of data from aerospace chambers and wind tunnels.

He began his graduate work in 1974 at The University of Tennessee Space Institute and received the Master of Science degree with a major in Computer Science in June 1979.

He and his family reside in Shelbyville, Tennessee, where he is an active member and officer of the Jaycees.