HOW TO USE THIS GUIDE

This manual is an installation guide for the consultant. (Comprehensive reference material for the GBS/MVP is contained in the GBS/MVP System Manuals.)

This manual assumes a conceptual familiarity with KFAM-7, global partition architecture, and partition generation. For more information on these subjects, refer to the appropriate manuals:

- KFAM-7: ISS Release 5.0 User Manual
- BASIC-2: BASIC-2 Language Reference Manual
- Partition generation and MVP reconfiguration: 2200 MVP Introductory Manual

MINIMUM HARDWARE CONFIGURATION	
CPU	- 2200MVP with 32K*
Input	- 2236DE Interactive Terminal
Storage	- F/R Disk Drive (Hard Disk)
Printer	- any 132 character per line printer

Package Numbers	
Payroll	195-2109-3
B.O.M.	195-2110-3
AP/GL	195-2108-3
OE/Inv	195-2107-3
Inv. Mgt.	195-2111-3

* Allow 16K per bank for system overhead and global text and 16K for each terminal.

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

1.1 FEATURES

GBS/MVP is a system of programs and data files that perform a variety of business functions. Included in its repertoire are invoicing, inventory control, inventory management (forecasting)*, bill of materials**, sales analysis, order entry, accounts receivable, accounts payable, and payroll. Its disk capacity allows up to 200 independent companies to process simultaneously.

GBS was designed to allow easy modification thereby accommodating virtually any industry or business application requirement. A special initialization (or configuration) program provides the installing vendor with a high degree of flexibility in determining file locations and sizes, as well as the files' key field ID visual display and size. At any time during the initialization process, the vendor may examine input and change any parameters previously entered. With control over the location of files, the installing vendor is able to maximize disk space. Default values for data and key files' locations, along with input justification parameters, are embedded in the initialization program but can be overridden during the set-up procedure. A suggested procedure for installation is to place all master files on the fixed disk, and key (index) files, work files, transaction files, and programs on the removable disk.*** (Since only master files need be copied, this set-up simplifies back up procedures.) The only assumption made by the system is that all GBS programs reside on the same disk as the MVP operating system.

* Inventory Management is a separate system which is commonly integrated into the GBS Inventory Control System for its most efficient usage. However, Inventory Management System may be run independently of GBS. (See Appendix B for integration instructions).

** GBS may be purchased independently of the Bill of Materials System. However, the Bill of Materials System may not be purchased separately.

*** Suggested for the 2260BC disk drives or the 2280-1 disk drive. A different configuration may be determined for the 2280-2/-3 drive by the vendor. The initialization routine assumes only a fixed/removable configuration of programs and data files based around the 320-B20 device address scheme.

GBS/MVP supports a multiuser-terminal environment, in either a singlebank or multibank environment (singlebank = 64K, multibank = 2 or more 64K banks). In the singlebank environment, a single partition is referred to as a global partition by each terminal. In the multibank environment, a small partition is globally accessible by each bank, in addition to a global partition in each bank (to be discussed later). The use of global code on the MVP reduces the amount of memory necessary to run any program. The global partition in either the singlebank or multibank versions is resident at all times and acts as a library for KFAM-7, specialized PUT/GET subroutines and screen and date routines, all of which represent the GBS infrastructure.

The specialized subroutines, modified versions of ISS utilities, have special use in the GBS system. These subroutines include screen handling, data entry, and the date verification routines. The PUT/GET subroutines are unique program files that perform the file access routines. They read from and write to a specific KFAM or sequential file. KFAM-7 subroutines (Wang's Key File Access Method) coordinate random access disk activity to each file in the system. A KFAM-7 file is comprised of two separate files: a user file and a key (index) file. Each user file contains all of the information relative to a given product record customer record, etc., and has a corresponding unique key value in the key file used to randomly locate the record in the user file.

Due to the modularity of each program in the system, variable usage is unique to each program within the system. When a program is executed from the menu, separate variable identification program files containing the variables to be used during the execution of the program are overlayed (loaded) into memory from the menu. The actual subroutines executed during processing are located in either the universal global or local global in a multibank system or in local global in the singlebank version. (Global subroutines are covered in Chapter 5 which includes instructions on modifications and movement into and out of the global partition.)

GBS/MVP is now VP compatible with Release 2.0. All subroutines and utility programs that were previously reentrant (global) code will now be overlayed into memory at run time. Those routines affected by the operating environment of the VP are provided on a separate diskette. The minimum core requirements for the VP version of Release 2.0 is 32K. The VP version of GBS uses the multiplexed version of KFAM-7. Thus, two or more VP/MVP processors may be multiplexed to one disk drive.

GBS/MVP supports a variety of hardware configurations with varying numbers of systems and/or terminal printers. If a system printer is attached, it is reserved for exclusive use by the first terminal gaining access. If additional terminals request access to a busy printer, a message indicating in-use status and the partition in control are displayed.

CHAPTER 2
THE DATA FILES

This chapter provides information on the data files which are set up by the consultant. The following subjects are covered:

- Data file initialization
- Company file
- Device table
- Data file names
- Building A/R Open Item file
- Audit files
- File full condition
- Expanding KFAM files

2.1 DATA FILE INITIALIZATION

GBS/MVP, as delivered, is a set of source and compressed program diskettes. (Source programs are heavily remarked, with one instruction per line. Compressed programs have no remarks and have multi-statement lines. Both perform exactly the same function.) The diskettes must be copied to hard disk using the ISS Copy/Verify utility. (This is standard procedure and may be performed on T, VP, and MVP hardware.) During installation and customization of GBS/MVP, it is recommended that source programs be used (along with larger partitions). Once the system performs to the user's satisfaction the programs can be compressed with the ISS Compression utility. If no customization is to be done the system programs can be used as delivered.

The Initialization program is a standalone procedure; it can be run on VP and MVP hardware under configurations other than standard GBS. Once the GBS diskettes have been copied to disk, all files should be initialized. Initialization programs estimate file sizes based upon data in the GBS Configuration and Customer Survey, supplied to the consultant by the user, which is then supplied to the program by the consultant. To initialize all files, perform the following steps:

- Mount the diskettes which contain either source or system GBS programs (the system programs are the compressed version of the source programs).

- SELECT the disk that contains the "INIT" program using SELECT Disk XXX where XXX is the address.

- Enter LOAD RUN "INIT" and touch RETURN.

- Enter the address from where the programs are loaded.

- A company number can now be specified and any access to the files will be via their company file. (See Section 2.2).

- If the file was already established, previously entered parameters may be used. Otherwise, the system will enter into the question and answer mode.

- Select the system(s) for which files are to be initialized. If the system was previously initialized and direct access to the file initialization and creation section is desired, enter 0 for systems to install. Otherwise, enter the number of the systems to install.

- Respond to the prompts concerning numbers of customers, products, open items, orders, and so on. (These questions are grouped by system.)

- Study the hard copy produced to make sure that file sizes seem reasonable. To provide for file growth, enter a growth factor percentage.

- Enter the location of all data files (defaults are supplied).* If the standard GBS backup procedure provided with the system is to be used, all files that default to device 320 should remain with 320, since they are the files that must be backed up (if all data files reside on a different device address, the standard back-up can be used by changing the address in "BACK010A"). At this point, file sizes may still be changed. Estimates previously provided serve as defaults only. If file sizes call for zero sectors, the files are ignored in subsequent steps, making it possible to install the systems independently, simultaneously, or in stages.

Before the program initializes each KFAM file, the file parameters are displayed for review against the following chart. Note that only the number of sectors allocated for each file will be different.

File Name	Record Type	Record Length	Blocking Factor	Key Length	Start Position Of Key	Sectors Per Record
INVCF01X**	A	252	1	6	3	
INVTF01X	A	252	1	12	3	
ACCTF01X	A	50	5	12	3	
ORENF01X	A	252	1	4	3	
SALEF01X	A	252	1	3	3	
INVCF02X	A	252	1	6	3	

(continued)

* Lines 6000 through 7200 of the INIT program contain default information for every file in the system.

** X- indicates the company number. This will be unique to each company initialized on the disk.

File Name	Record Type	Record Length	Blocking Factor	Key Length	Start Position Of Key	Sectors Per Record
INVCF03X	A	252	1	6	3	
INVTF03X	A	18	13	17	3	
ACPAF01X	A	252	1	5	3	
ACPAF02X	A	63	4	15	3	
GENLF01X	A	252	1	8	3	
PYRLF01X	M			9	3	6
PYRLF02X	A	252	1	3	3	
FCSTF01X	A	252	1	10	3	
BOMS010A	N	252	1	15	3	1
BOMS020A	N	252	1	4	3	1

Next, the program initializes the sequential files calculating file sizes according to the following specifications:

File*	File Name	Number Of Records To Allocate
Control file	GBS1F01X	System allocates 370. (Used during A/R, A/P, and DE).
	GBS4F01X	System allocates 10. (Used during payroll).
A/R Sequential file	ACCTF02X	Should equal number of records specified for ACCTF01.
Inventory Sort Tags	INVTF02X	Calculated from the size of the Inventory file.
Backorder Keys file	ORENF02X	Calculated from the size of the Open Order file.
Lost Sales/Est. Shortage file	ORENF03X	Should be large enough to accommodate all line items that produce lost sales or estimated shortages per order entry run; 3 records blocked per sector.
Shipping Shortage file	ORENF04X	Should be large enough to accommodate all line items short-shipped per shipping confirmation run; 3 records blocked per sector.

* The GBS/MVP System Manuals have File-Program Cross Reference tables that indicate which programs make use of these files.

FILE*	FILE NAME	NUMBER OF RECORDS TO ALLOCATE
Sortwork file	SORTWORK	Calculated from the number of records in the largest file to be sorted or reorganized. The ISS Release 5.0 User Manual (SORT-4 chapters) documents the method by which the size of this file can be calculated.
Maintenance Audit file	AUDIF01X	Should be large enough to accommodate one day of maintenance activity; 3 records blocked per sector. Suggested file size is estimated.
Transaction Audit file	AUDIF02X	Should be large enough to accommodate one day of transaction activity. Suggested file size is estimated.
Order Analysis file	ORENF05X	Calculated from the size of the Open Order file.
J/E Sort Tags	GENLF05X	Calculated from the size of the Old Journal Entries File.
Product Structure file	BOMSF01X	Should be large enough to accommodate all finished goods that are in inventory. One sector is required for every eight components.
Where Used file	BOMSF01X	Size of where-used file is directly dependent upon size of product structure file.

Control File

When the A/R control file is initialized, the starting invoice number, the nonregular (credit/debit memo) invoice number, and all order numbers should also be established; the invoice number must be greater than the user's highest pre-GBS invoice number. If the starting invoice number for an operation is less than the highest pre-GBS invoice number and a general payment is issued to a customer with GBS invoices on the A/R Open Item file,

* The GBS/MVP System Manuals have File-Program Cross Reference tables that indicate which programs make use of these files.

the more recent invoices are wiped out instead of the oldest ones. In addition, all nonregular invoice numbers must be greater than regular invoice numbers. When a general payment is made via A/R Transaction Entry, the amount is applied to the lowest numbered invoice in the file, (if the invoice numbers are not assigned as just described, a payment is not necessarily applied to the oldest invoice.)

Wash accounts receive information entered against salesman or product file numbers not currently on file. They therefore should be established in the Inventory and Salesman Master files by executing the appropriate file maintenance routines. A wash account is identified by a 12-byte field of Zs for the product number and a 3-byte field of Zs for the salesman number. If these accounts are not established by file maintenance routines, the Invoice Transaction File Update program will establish them.

2.2 COMPANY FILE

During the initialization program that created GBS/MVP files, the company file FILEF01X (where X is a company code) is also created. The file stores the following information.

Sector 1

Q0$(9) 3	Record ID Parameters (see discussion of variable length record ID).
N2$60	Company Name - displayed at the top of each menu, and at the top of every GBS report.
P$16	Company password - required by the START Program.
S0$(15)1	Sp]ecialized company PUT/GET routine identifiers. 8th byte of the PUT/GET name.

Sectors 2-5 File Names & Addresses

A6$(90)11 (Redimensioned to A6$(8)124 prior to disk read) Each element in this array consists of a file name (8 bytes followed by an address (3 bytes)) for the file.

A6$(1) through A6$(24) contain the names of KFAM files GBS/MVP uses only the first 16, leaving space (slots 17-24) for additional files.

A6$(25) through A6$(48) contain the names of sequential files. GBS/MVP uses only the first 16, leaving space (slots 41-48) for additional files.

A6$(49) through A6$(72) contain the names of the primary key files associated with A6$(1-24). A6$(73) has the address of GBS programs. A6$(34) through A6$(90) are intended for secondary key file names and addresses, but are not currently used.

A7$(48)20 (Redimensioned to A7$(8)124 prior to disk read). It may be altered at installation time.

A7$() has the prose descriptions of GBS files, and is parallel to the first 48 entries in A6$().

Multiple Companies

The following provisions are made for multiple companies.

1. A unique company name for the company currently being processed is displayed on every menu and printed on every report.

2. A company file for each company (as described above), FILEF01X where X depends on the company number. (This is a BIN function of the number entered in order to keep the file name to 8 bytes.)

3. The company number is used to determine the last character of each data file name. During installation of the system, the last character of every file name defaults to the company number, but may be changed to some other character, prior to ending the INIT program.

For example, Q6$ = "FILEF01" & BIN(S+48) is always used to determine which company file to use. In this way, Company 0 gets FILEF010 Company 8 gets FILEF018, and Company 13 gets FILEF01=. (The "&" sign means concatenate what follows to what precedes this sign.)

Whenever the system opens a file, the file name is retrieved from the company file. Since the last character of the file name may be altered at installation time, two or more companies may share the same data files.

For example, suppose a manufacturer distributes products using two different company names. The same products are distributed, but two different customer bases are maintained.

When Company 0 is installed, the name of the customer file and the name of the inventory file can be the system defaults (INVCF010 and INVTF010 respectively). When Company 1 is installed, the name of the inventory file could be altered from the default of INVTF011 to INVTF010, while the default of INVCF011 could be used for the customer file name. Whenever the inventory file is needed, the file INVTF010 will be accessed.

All access to any data file goes through the menu. The menu always accesses the company file to determine data file names.

2.3 DEVICE TABLE USAGE

The 2200 MVP (and VP) operating systems maintain a local device table for each partition. This table keeps track of the devices selected for PRINT, LIST, CO, etc. (refer to VP language reference manual for more information) along with each file opened during the processing run.

There are 16 slots, known as device entries, to assist with file access, by storing device addresses, starting, current, and ending sector numbers. GBS uses device entry 0 for program overlays, since the address associated with this device defaults to the address from which the MVP operating system was loaded. The only GBS programs that perform a SELECT#0 are the Sort setups, since SORT-4 arbitrarily executes a SELECT DISK 310.

The other 15 slots in the device table are used by open data files, and to hog the printer. Device entry 15 is used to hog the printer.

Device entry 14 is used for all key files. KFAM-7 has been specifically modified for GBS to reduce the number of device numbers used. With the present modification to KFAM, each KFAM user file still requires a unique entry, but each key file will use slot 14. This capability is available since key file access is always in DA, not DC mode.

The menu passes file information into a variable, S$(), whenever a file is opened. This table parallels the device table, i.e., device number 1 corresponds to S$(1). (In addition, the device address is stored as a 3 byte literal in M$(), for use within some GBS programs. See the discussion of Sort setup modules.) At any time, a program can determine the name of the file open on a given device number by using S$(). This is useful for statements such as a Limits statement to determine the number of sectors used.

Not a single data file name is hard-coded into a GBS program. Instead, S$() is used. See Section 5.2 for a discussion of the manner in which data files are opened, and the way in which S$() is filled.

2.4 DATA FILE NAMES

The data files utilized by the GBS/MVP system appear below. They are created and catalogued during the Initialization procedure described in Section 2.1.

Accounts Receivable

Customer file (KFAM)
Inventory file (KFAM)**
Salesman file (KFAM)
A/R Open Item file (KFAM)
A/R Sequential Open Item file
Control file
Invoice Transaction files (KFAM)-2 files
Maintenance Audit file*
Transaction Audit file*
Sortwork file*

Payroll

Employee file (KFAM)
Bank Address file (KFAM)
Control file

* These files are common to all applications.

** This file is also common to the Bill of Materials and Inventory Management Systems

Accounts Receivable (Continued)
Order Entry file
Open Order file (KFAM)
P.O. Activity file (KFAM)
Lost Sales/Est. Shortage file
Shipping Shortage file
Order Analysis Work file
Backorder Keys file
Sort Tags for Inventory file

Accounts Payable & General Ledger

Vendor file (KFAM)
A/P Open Item file (KFAM)
General Ledger Chart of Accounts file (KFAM)
A/P Check file
Control file
Old Journal Entry file
New Journal Entry file
Standard Journal Entry file
Sort Tags For Old J/E file

Payroll

Bill of Materials

Product Structure file (KFAM)
Where Used file (KFAM)

Inventory Management

Profile file (KFAM)

2.5 A/R OPEN ITEM FILE BUILD PROGRAM

The A/R Open Item File Build program (INITA/R) adds records to the file ACCTF010 (A/R Open Items - KFAM), calculates a customer's A/R balance, and adds it to the current balance field in the Customer file. (It is assumed that these fields are equal to zero since no processing has taken place after Initialization.)

There are two prerequisites to running INITA/R: the entire Initialization procedure has been completed and all customers have been entered through Customer File Maintenance. INITA/R may be executed as many times as required to load all the open items. At any time an A/R Inquiry/List may be run using the All option to provide a balance for each customer, a total A/R balance, and all supporting details. At this time, any adjustments can be made through the A/R File Build program add, change, or delete functions. (End-of-month procedures are discussed in the GBS/MVP User Manual.) After creating the Open Item file, the INITA/R program should then be removed from the disk to prevent inadvertent use.

Follow the steps below to build the A/R Open Item file:

1. Load the A/R Build program by pressing SFK 27 from the System Menu.

2. Enter the default date in the specified form and touch RETURN. This date is automatically used as the transaction date for balance forward and service charge records. It may also be used in other types of transactions by touching RETURN when prompted for a transaction date.

3. Enter the maintenance type:

   A = Add
   C = Change
   D = Delete

4. Enter a customer number, or END and RETURN. The program verifies that the customer exists and the name is displayed; otherwise, go to Step 18.

5. Enter an invoice number, or END and RETURN. To add a record, enter a unique invoice number; to change or delete a record, enter an existing invoice number. (The invoice number must be numeric.) Balance forward and service charge transactions must be applied to an invoice number of zero. If END and RETURN are entered, go to Step 4.

6. Enter one of the transaction types listed below and touch RETURN.

   P = Payment
   CI = Credit Invoice
   DM = Debit Memo
   CM = Credit Memo
   BF = Balance Forward
   SC = Service Charge
   RETURN = Invoice

   The A/R file is checked for the existence of the invoice number record after the transaction type is entered. Only one balance forward and service charge transaction type may be applied per customer. Balance forward records may be created for balance forward customers only.

7. If a record is being added, go to Step 8. If a record is being changed or deleted, the selected record is displayed. To change a record, go to Step 8. When deleting a record, the prompt OK TO DELETE is displayed; enter Y or N and go to Step 5.

8. If the transaction type is BF or SC, go to Step 13; otherwise, go to Step 9.

9. Enter the original transaction date in the specified form or touch RETURN for the default date.

10. Enter the amount and touch RETURN. (It is not necessary to enter the sign; the transaction type determines it.)

11. If a payment is being entered, a check number prompt appears. Enter a check number.

12. Go to Step 17.

13. Enter the current amount and touch RETURN.

14. Enter the 31-to-60-day amount and touch RETURN.

15. Enter the 61-to-90-day amount and touch RETURN.

16. Enter the over-90-day amount and touch RETURN.

17. If the input data is acceptable, touch RETURN; the new or updated record is written to the file. Go to Step 5. If the data is unacceptable, enter N, RETURN, and go to Step 5.

18. Touch R to restart and go to Step 3, or touch E to end.

2.6 AUDIT FILES

Master file maintenance, file adjustment, and transaction activity produce audit trails (records) written to either the Maintenance Audit or the Transaction Audit files. They also update a particular data file record. Audit records generated by file maintenance, order adjustment, and shipping confirmation programs are written to the Maintenance Audit file (AUDIF010). Audit records generated by transaction entry or update programs are written to the Transaction Audit file (AUDIF020). Audit Report programs sort records in each respective file by category before printing. In order to reset (clear) these files, the audit report for that file must be generated for the entire file.

Both the Maintenance and transaction programs write to an audit file after a full sector of records has been accumulated. They also write to a file at the end of a processing run where a partial sector would be filled. (The entire disk is hogged during this operation.)

2.7 FILE FULL CONDITIONS

Because GBS/MVP and GBS/VP utilize fixed length files, it is possible for a File Full condition to be reached. At start-up time, the operator can determine which files in the system are approaching the File Full condition by selecting the File Status Report after entering the company number.

When a file becomes 80% full, a message is displayed informing the operator of this condition. Concerning sequential files, an additional message is displayed when the 95% full condition is reached. At this point, the current transactions processed are entered into the file and the file becomes closed. This procedure is most evident during maintenance processing. Concerning the audit files when a full condition is reached, the operator must run a report in the All mode so the file can be reset and cleared.

Concerning KFAM files, File Full conditions are controlled by the KFAM system. When a KFAM file approaches the File Full condition, it may be advantageous to run the REORGANIZE program against that file to release any space occupied by deleted records. The Reorganization Menu has a reorganization program for each KFAM file.

2.8 EXPANDING KFAM FILES

To increase the size of a KFAM file, the following steps may be used. (There are faster but less straightforward methods to do this and thus are not recommended.)

1. COPY all files to backup disk "A".

2. Scratch the user and key files (for the KFAM file to be enlarged) on the master disk.

3. MOVE the platters that contain the scratched files to backup disk "B" (KFAM-7 files may be moved without causing any problems). This will remove the space previously allocated to the scratched files by "squashing" the disk files.

4. COPY the files from backup "B" back to the master disk.

5. Use the DATASAVE DC OPEN command to open a new key and user file on the master platter. Then SCRATCH the key and the user file. (Both the key and user files must be expanded by the same amount)

6. MOVE the key file and user file from backup disk "A" to the master platter. (Use the MOVE T/xyz, "filename" to T/xyz ( ) command.)

7. Run the GBS/MVP Reset Access Tables utility to "fix" the MUX trailer or system control sector.

8. Run a program that opens and closes the file, to update the recovery information, and the KDR parameters.

> NOTE:
>
> Starting with the ISS Release 5.1, (which was used with GBS/MVP Release 2.0) the subroutine that opens a KFAM file was a Limits statement to determine the size of a file. The Reallocate KFAM-File Space utility no longer needs to be used to update the internal KDR parameters related to file size.

CHAPTER 3
GBS SYSTEM DESCRIPTION

This section gives a comprehensive view of the GBS system. The following subjects are covered:

- Overview of the system
- Features common to all systems
- System and program menus

3.1 OVERVIEW OF THE SYSTEMS

The General Business System is comprised of programs and files which relate to a variety of accounting applications. Descriptions of the systems which make up GBS follow:

Invoicing programs create postbilling invoices that contain ship-to, billing, and relevant product information through interaction with the Customer, Salesman, and Inventory files.

Accounts Receivable programs automatically interface with the invoice system through the Invoice Transaction file, thereby updating the Open Item and Control files. The Open Item file keeps track of accounts and generates statements.

Sales Analysis reports are single-line sales reports depicting sales and cost extension figures along with profit margins for current month and year-to-date figures. They are printed by the Customer, Inventory, and Salesman file display/print programs.

Order Entry programs process customer orders, update open order and appropriate master files, permit adjustment and confirmation of orders, and print shipping papers and invoices.

Inventory Control programs update the Inventory Master file with receipts or withdrawals not connected to invoicing. They also produce stock status, low stock, inactive items information, physical inventory sheets, and recommended purchase order reports.

Inventory Management programs provide accurate forecasts of sales demand and then incorporate these forecasts into reordering recommendations. Finally these programs generate sets of conditions that will insure any desired level of service on the most economical basis possible. (Optional Sub-system)

Bill of Materials programs provide and maintain a product structure file for manufacture. Raw materials inventory quantities and costs can be related to finished goods inventory. The system produces Bill of Materials explosions, and Where-Used listings. (Optional Sub-system)

Accounts Payable programs perform accounts payable posting, check writing, and automatic general ledger updating.

General Ledger programs perform general ledger posting and produce corporate financial reports.

Payroll programs calculate earnings and deductions based upon entered and/or stored data for hourly, hourly-exempt, and salaried employees. They support check, cash, and direct-deposit payment systems.

3.2 FEATURES COMMON TO ALL SYSTEMS

Descriptions of special features of the system follow:

Maintenance and Display/Print programs access data in individual files for viewing and/or altering.

Audit reports print information accumulated from maintenance or transaction entry.

Control files accumulate daily totals of all transactions (which may subsequently be printed by a display/print program). Each of these files contains 366 records; one for each day of the year. These files can serve as the basis for General Ledger Journal entries.

File Cleanup programs clear all totals from the Control files and clear month-to-date, quarter-to-date, and year-to-date totals from master files. Specific details appear below:

Cleanup Program Name	File Affected	Information Cleared
Clear File (Accounts Receivable Menu)	Customer	M-T-D and Y-T-D
	Inventory	M-T-D and Y-T-D
	Salesman	M-T-D and Y-T-D
	Control	Daily Totals

Clear File (General Ledger Menu)	Control	Daily Totals
	Vendor	Y-T-D
	General Ledger	M-T-D, Y-T-D for Income Expense
	All Journal Entry	Reset (see GBS/MVP Mod III System Manual)
Control File Maintenance	Control	Yearly Totals
End of Pay Cycle/ Period	Employee Master	Current Q-T-D, Y-T-D

3.3 SYSTEM and PROGRAM MENUS

The following displays are the menus for each subsystem within the GBS system. The Inventory Management and Bill of Materials versions of the Inventory Menu are optional menus not supplied with the standard GBS system.

GBS/MVP SYSTEM MENU
Release 2.0

SFK	PROGRAM NAME	SFK	PROGRAM NAME
(00)	INVOICING & RECEIVABLES MENU	(06)	UTILITIES MENU
(01)	INVENTORY MENU	(07)	FILE REORGANIZATION MENU
(02)	ORDER ENTRY MENU		
(03)	ACCOUNTS PAYABLE MENU		
(04)	GENERAL LEDGER MENU		
(05)	PAYROLL MENU	(31)	END OF PROCESSING

GBS/MVP INVOICING & ACCOUNTS RECEIVABLE SYSTEM
Release 2.0

SFK	PROGRAM NAME	SFK	PROGRAM NAME
(00)	ENTER INVOICES	(09)	MAINTAIN CUSTOMER FILE
(01)	PRINT INVOICE REGISTER	(10)	MAINTAIN SALESMAN FILE
(02)	PRINT INVOICES	(11)	ENTER CASH RECEIPTS
(03)	POST INVOICES	(12)	DISPLAY/PRINT CUSTOMERS
(04)	AGE A/R & CALCULATE SERVICE CHARGES	(13)	DISPLAY/PRINT SALESMAN
		(14)	DISPLAY/PRINT A/R OPEN ITEMS
(05)	PRINT A/R STATEMENTS	(15)	PRINT CREDIT REPORT
(06)	PRINT AGED TRIAL BALANCE	(16)	DISPLAY/PRINT CONTROL FILE
(07)	PURGE OPEN ITEM FILE		
(08)	CLEAR FILES	(31)	SYSTEM MENU

GBS/MVP INVENTORY SYSTEM
Release 2.0

SFK	PROGRAM NAME	SFK	PROGRAM NAME
(00)	MAINTAIN INVENTORY FILE	(05)	PRINT PHYSICAL INVENTORY SHEETS
(01)	ENTER INVENTORY TRANSACTIONS		
(02)	PRINT STOCK STATUS REPORT	(06)	ENTER PHYSICAL COUNT
(03)	PRINT INACTIVE/LOW STOCK REPORT	(07)	PRINT PHYSICAL INVENTORY REPORT
(04)	DISPLAY/PRINT INVENTORY	(08)	RECOMMEND PURCHASE ORDERS
		(31)	SYSTEM MENU

GBS/MVP INVENTORY SYSTEM WITH BILL OF MATERIALS
Release 2.0

SFK	PROGRAM NAME	SFK	PROGRAM NAME
(00)	MAINTAIN INVENTORY FILE	(08)	RECOMMEND PURCHASE ORDERS
(01)	ENTER INVENTORY TRANSACTIONS	(09)	MAINTAIN BILL OF MATERIALS FILE
(02)	PRINT STOCK STATUS REPORT	(10)	BILL OF MATERIALS PARTS EXPLOSION
(03)	PRINT INACTIVE/LOW STOCK REPORT	(11)	INDENTED COSTED BILL OF MATERIALS
(04)	DISPLAY/PRINT INVENTORY	(12)	DISPLAY/PRINT WHERE USED FILE
(05)	PRINT PHYSICAL INVENTORY SHEETS	(13)	DISPLAY/PRINT SINGLE LEVEL EXPLOSION
(06)	PRINT PHYSICAL INVENTORY REPORT	(14)	GROSS REQUIREMENTS EXPLOSION
(07)	ENTER PHYSICAL COUNT	(31)	SYSTEM MENU

GBS/MVP ORDER ENTRY SYSTEM
Release 2.0

SFK	PROGRAM NAME	SFK	PROGRAM NAME
(00)	ENTER ORDERS	(07)	PRINT LOST SALES/EST SHORTAGE REPORT
(01)	PRINT ORDER REGISTER	(08)	PRINT SHIPPING SHORTAGE REPORT
(02)	PRINT SHIPPING PAPERS	(09)	ADJUST OPEN ORDERS
(03)	CONFIRM SHIPMENTS	(10)	DISPLAY/PRINT OPEN ORDERS
(05)	POST INVOICES	(11)	ANALYZE OPEN ORDERS
(04)	PRINT SHIPPING REGISTER	(31)	SYSTEM MENU
(06)	PRINT BACKORDER SHIPPING PAPERS		

GBS/MVP ACCOUNTS PAYABLE SYSTEM
Release 2.0

SFK	PROGRAM NAME	SFK	PROGRAM NAME
(00)	ENTER TRANSACTIONS	(06)	MAINTAIN VENDOR FILE
(01)	PRINT CASH REQUIREMENTS	(07)	MAINTAIN OPEN ITEM FILE
(02)	SELECT ITEMS FOR PAYMENT	(08)	DISPLAY/PRINT VENDOR MASTER FILE
(03)	PRINT CHECKS	(09)	DISPLAY/PRINT OPEN ITEM FILE
(04)	PRINT CHECK REGISTER	(10)	REPORT ON DISTRIBUTION
(05)	PURGE PAID ITEMS	(31)	SYSTEM MENU

GBS/MVP GENERAL LEDGER SYSTEM
Release 2.0

SFK	PROGRAM NAME	SFK	PROGRAM NAME
(00)	JOURNAL ENTRY	(07)	CLEAR FILES (End of Month)
(01)	TRIAL BALANCE REPORT	(08)	MAINTAIN CHART OF ACCOUNTS
(02)	POST J/E TO CHART OF ACCOUNTS	(09)	MAINTAIN CONTROL FILE
(03)	PRINT INCOME STATEMENT	(10)	DISPLAY/PRINT CHART OF ACCOUNTS
(04)	PRINT BALANCE SHEET	(11)	DISPLAY/PRINT CONTROL FILE
(05)	PRINT BUDGET REPORT	(12)	DISPLAY/PRINT JOURNAL ENTRY
(06)	PRINT SCHEDULE REPORT	(31)	SYSTEM MENU

GBS/MVP PAYROLL SYSTEM
Release 2.0

SFK	PROGRAM NAME	SFK	PROGRAM NAME
(00)	START PAY CYCLE	(08)	EMPLOYEE MASTER FILE MAINTENANCE
(01)	ENTER HOURS & DEDUCTIONS	(09)	BANK ADDRESS FILE MAINTENANCE
(02)	CALCULATE GROSS AND NET PAY	(10)	CONTROL FILE MAINTENANCE
(03)	ENTER ADJUSTMENTS	(11)	EMPLOYEE MASTER FILE INQUIRY/LIST
(04)	PRINT EMPLOYEE REGISTERS	(12)	BANK ADDRESS FILE INQUIRY LIST
(05)	PRINT CHECKS, MEMOS, 941s, W-2s	(13)	CONTROL FILE/INQUIRY/LIST
(06)	PRINT RECONCILIATION LISTS	(31)	SYSTEM MENU
(07)	END OF PAY CYCLE/PERIOD		

GBS/MVP UTILITIES MENU
Release 2.0

SFK	PROGRAM NAME	SFK	PROGRAM NAME
(00)	PRINT MAINTENANCE AUDIT REPORT	(05)	COPY DATA TO BACKUP DISK
(01)	DISPLAY/PRINT TRANSACTIONS	(06)	DELETE INCOMPLETE INVOICES, ORDERS
(02)	BUILD ACCOUNTS RECEIVABLE FILE	(07)	RESET ALLOCATED (BACK-ORDERED) AMOUNTS
(03)	RESET ACCESS TIME	(31)	SYSTEM MENU
(04)	RESTORE DATA FROM BACKUP DISK		

GBS/MVP FILE REORGANIZATION MENU
Release 2.0

SFK	PROGRAM NAME	SFK	PROGRAM NAME
(00)	REORGANIZE CUSTOMER MASTER FILE	(04)	REORGANIZE P.O. ACTIVITY FILE
(01)	REORGANIZE INVENTORY MASTER FILE	(05)	REORGANIZE BILL OF MATERIALS FILES
(02)	REORGANIZE SALESMAN MASTER FILE	(31)	SYSTEM MENU
(03)	REORGANIZE OPEN ORDER FILE		

CHAPTER 4
OPERATING ENVIRONMENT

4.1 PARTITION GENERATION - MVP

The MVP operating system is stored in a data file named @@ on the GBS/MVP system diskettes. When the GBS diskettes are copied to hard disk, the operating system is copied as well, thereby eliminating a separate copy procedure for it. The operating system is loaded and executed by pressing the SF Key which corresponds to the disk holding the GBS programs.

Provided with GBS is a modified form of @GENPART, the MVP partition generation utility, which executes automatically and configures the system once the operating system has been loaded. The standard singlebank (64K or less) GBS configuration, which appears below, is supplied as an aid and can be employed when @GENPART requests configuration specifications. (It is not the mandatory configuration.) The standard GBS configurations for MVP (singlebank and multibank) follow:

GBS/MVP SINGLEBANK CONFIGURATION
Release 2.0

TERMINAL	PARTITION	PARTITION SIZE	PROGRAMMABLE	BOOTSTRAP
1	1	10K	Y	GBS (global code)
1	2	17K	Y	START (for Payroll)*
2	3	17K	Y	START
3	4	17K	Y	START
		61K**		

* If running Payroll, a partition must be 19K at the expense of the other terminals.

** This is a maximum (64K) type configuration for a singlebank MVP.

GBS
Loads @GBS
Loads @GBSKFAM
Loads @GBSFILE
Loads @GBSISS

GBS/MVP MULTIBANK CONFIGURATION
Release 2.0

No. of Banks = 2-3
No. of Terminals = 6-9
No. of Partitions = 9-14

BANK 1	TERMINAL	PARTITION	PARTITION SIZE	PROGRAMMABLE	BOOTSTRAP
	1	1	5K	Y	GBSM
	1	2	16K	Y	START
	2	3	16K	Y	START
	3	4	16K	Y	START
	1	5	8K	Y	@GBSM
			61K		

BANK 2	TERMINAL	PARTITION	PARTITION SIZE	PROGRAMMABLE	BOOTSTRAP
	1	1	16K	Y	START
	2	2	16K	Y	START
	3	3	16K	Y	START
	1	4	8K	Y	@GBSM
			56K		

Preceding is a suggested maximum multibank configuration.

GBSM Loads uGBS
uGBS* Loads Selected subroutines
@GBSM** Loads @GBSKFAM
@GBSFILE
@GBSISSM (dummy deffn')

*GBSM is the universal global code partition. It must be loaded into bank #1, partition #1, and the partiton-size must be 5K.

**GBSM is the local global partition. It must be loaded into every bank that is to execute GBS.

@GENPART automatically executes the program GBS or GBSM. (The date entered at terminal #1 is used merely as a header on reports and the default for certain data entry programs. When the date is important, a chance to change it is always offered.) Refer to Appendix A for @GENPART configuration procedures.

4.2 SYSTEM GENERATION - VP

Because the VP, a single terminal system, possesses a fixed memory size, it has no partition capabilities. Thus, there is no mechanism that would permit the execution of global routines. Therefore, the VP requires that all variables and marked (DEFFN') subroutines be resident in memory at execution time. Changes were necessary to the GBS/MVP to permit functional execution of MVP code on the VP. The changes needed included:

- SELECT @PART S$ (SO$) - Use of the variable global name S$ and SO$ are set prior to execution on the MVP to either uGBS or GBS and to " " on the VP. When the VP executes the SELECT @PART " ", no global partition is referenced.

- All subroutine modules, i.e., SUBUTIL, SUBDATE, DENTRY, etc., on the VP contain the actual executable marked subroutines and are overlayed into memory at run time.

- START was modified to permit the execution of the system date routine locally on the VP, whereas on the MVP it is executed in the global partition. START on the MVP also sets the values of S$ and SO$.

All VP required routines are supplied on a separate diskette. For VP installations, this diskette with VP compatibility subroutines should be used instead of the diskette with the MVP versions of the same subroutines.

The following is the minimum configuration for the VP version of MVP Release 2.0.

GBS/MVP VP VERSION
Release 2.0

No. of Terminals = 1
No. of Partitions = 1 -- VP only offers a single partition

TERMINAL	PARTITION	PARTITION SIZE
1	1	32K

A 32K VP is sufficient for GBS.

CHAPTER 5
PROGRAM STANDARDS AND TECHNIQUES

This chapter provides information concerning the GBS menus and programs. The following subjects are examined:

- Program Structure
- Menu Program Operation
- Multi Company Capabilities
- Naming of Variables
- Storage of Variables
- Length of Variables
- Variable Usage
- SORT-4 Utilities Program
- COMCLEAR Program
- KFAM-7 Access Modes
- KFAM-7 Access Tables
- File Protection (access conflicts)
- Record Protection
- Password Protection for Files
- Annotated Lists of the Global Subroutines
- Creation of Unique PUT/GET Routines

5.1 PROGRAM STRUCTURE

Each GBS program is composed of three parts: COM statements and the variable initialization routines (each program might need several sets of these) which are on disk as program module, the main body of the program which is in a program module as well, and the executable subroutine text which resides in global memory. (Subroutines called by the local program which are not global are in local memory.)

COM statements encompass lines 170--197; executable text in subroutine overlaps encompass lines 199--220. Once resolved, the COM statements are not retained. Instead, they are overlaid by COM lines for other subroutines or by COM/DIM lines for main line code.

The program's main body encompasses lines 4000--5999. Local program subroutines encompass lines 6000-8999. The variable description table encompasses lines 9000 and higher in the source programs.

The executable KFAM subroutine text (lines 580--3999) reside in the global partition and are associated with the COM and variable initialization statements in the program module.

In the event that the consultant wishes to reduce the size of global memory, a subroutine may be detached from global code and attached (copied) to the subroutine's correspondingly named COM and variable module. (Refer to Section 5.13 for the program names, and the BASIC-2 Language Reference Manual and the VP Disk Reference Manual for complete instructions for this procedure.)

All global subroutines can be modified, but local code, when appropriate, must be modified as well to make global and local codes consistent. Changes for PUT/GET subroutines, especially, must be consistent. For example, when changing the length of a variable, the field specification for $PACK/$UNPACK and the length in the COM statement must be changed as well; (no change in global code would be required). When adding to or deleting a variable from a GBS file, the global (executable) code must be also changed.

The GBS programs are run from menu programs written in BASIC. The menu programs originate from the same diskettes housing the various system programs.

Each time a GBS program is requested (from a menu) all the appropriate files associated with that program are opened. The required sets of COM statements and executable subroutine text are loaded by the menu into memory from the program modules on disk, thereby allocating space for the variables. The main program is then loaded into memory, overlaying the menu program. The menu program "destroys" itself. At this point, the local partition's symbol table has reserved space for all variables used by the main program and the subroutines. During execution the MVP operating system automatically searches for any subroutines listed in the main program. Local memory is searched first followed by a search of global memory. Subroutines placed in global memory are those that are executed frequently.

5.2 MENU PROGRAM OPERATIONS

Specifically, all GBS/MVP menu programs operate in the following manner.

The menu program passes the lengths of key file IDs (as numeric scalars in variable Q2) to the file definition subroutine overlays to dynamically dimension the length of some alphanumeric variables. For example, the length of the inventory key (product ID) is passed from Q0$(2), as stored in the company file,* to Q2 in the main menu immediately before overlaying, e.g., INVFILE. INVFILE has the following text lines that use Q2 (BASIC-2 requires the use of numeric scalar variables for dynamic allocation of variable space).

```
170 COM F1$Q2,F$12,....

199 L3$ = HEX(A0)&BIN(Q2)&HEX(A00C........
```

Thus, both the length of the product ID (F1$) and the image of the inventory record (L3$) for the $PACK and $UNPACK commands are controlled by parameters specified at installation time.

Each menu has a principal ID stored in Q0 which is used extensively in most programs called by that menu. The following list indicates the ID's used after each menu.

* See Section 5.5

1. Inventory menu uses the product ID, except for the Recommended Purchase Orders program, which uses the vendor ID.

2. Order Entry menu uses the product ID.

3. Accounts Receivable menu uses the customer ID, except for Enter Invoices, which uses the product ID, and the salesman's programs, which use the salesman ID.

4. Accounts Payable menu uses the vendor ID.

5. General Ledger menu uses the chart of accounts ID.

6. Payroll menu uses the employee ID.

The following blocking factors change when ID lengths change and are defined at menu level:

1. A/R Open Item file depends on customer ID length.
2. Invoice transactions and Open Order file (line items) depend on product ID length.
3. Bill of Materials file blocking factor depends on product ID length.

For example, the A/R menu performs the following calculations:

Q0 = VAL(Q0$(1)-1 provides the length of the customer ID (minus one byte for ship to code)

M0 = INT(249/(Q9+37)) provides the blocking factor for A/R Open Items.

The fixed length portion of the A/R record is 37 bytes long. As long as the customer ID is 12 bytes long, or less, the blocking factor will be 5; otherwise it will be 4. (In general, blocking factor calculations use 249 as the maximum number of bytes in a record, since this is the limit with ordinary DC mode operations. The transaction Audit file manages to squeeze 250 bytes into a sector).

5.3 SPECIAL MULTI-COMPANY CAPABILITIES

GBS/MVP will now support multiple company capabilities on a single system. Suppose GBS/MVP is used in a service bureau, where two or more companies are installed on the same system. The Start program could be modified to load a different system menu (SYS MENU) for each company. Every menu name is passed to a variable, M$. The system menu could be set up to load unique application menus for each company. Whenever a program returns to an application menu, M$ is used to load the menu.

Provision has also been made for unique programs for each company (also optional). The application menus place the program name in M5$. A subroutine within the menu has been provided (DEFFN '38) to alter M5$ immediately before the menu loads the mainline program. The seventh byte of M5$ could be used to specify a unique program name. If unique menus are used for different companies the Data statements with program names could be modified.

In either case, M5$ is always used for program overlays. The eighth byte of the program name always specifies which overlay to use. For example, suppose a company wants to control two separate divisions with slightly different record layouts for the customer master files for each company. The Customer Maintenance program could test S (company number) to perform some decisions. As an alternative it might be preferable to have two maintenance programs:

```
INVC020A for Company 0
INVC021A for Company 1
```

Because the company file maintains the names of the files used in the system and is accessed by each menu, the A/R menu could determine which file to use and which PUT/GET routine is necessary.

In addition, the invoicing programs might be different. The menu would set M5$ to either INVC030A or INVC031A. All segments of the invoicing sequence (INVOC30A to INVC030G) use M5$ to determine the next program overlay, for example:

```
STR(M5$,8)="B"
LOADTM5$4000
```

Unique PUT/GET routines may be defined for a file by initializing the appropriate S0$() element. S0$() is read from the company file; it is used by the menu to determine the eighth character of the PUT/GET routine to be loaded. The appropriate initialization module (INITC) should be modified to assure the setting of the S0$() variable.

Each menu as displayed in GBS is comprised of two separate program modules:

XXXXMENU -- Where XXXX refers to a specific menu in the system, namely ACCT, OREN, INVT, ACPA, GENL, PYRL, etc.

GBS MENU -- Overlayed into each XXXXMENU program.

The program XXXXMENU is an application menu that declares the menu variables, assigns values to variables unique to each menu and a set of DATA statements. The Data statements contain all of the information relative to the execution of an application menu. The application menu also contains two masked subroutines that are called by the execution module, GBS MENU, to perform tasks unique to each menu.

The Data statements in each XXXXMENU reside beyond line 7000 in the menu and contain 2 Data statements per program with a total of 5 data elements per program. The first Data statement has 3 elements. The first element is the program name, the second is the program description displayed by the menu and the third is a list of subroutines used by the program. The second Data statement contains the remaining two elements, the first of which indicates the number of KFAM files to be opened by the program and the last element indicates which files are to be opened. By manipulating the data in the last two elements, the menu will know whether or not it is preparing to Open KFAM or sequential files and whether or not it is about to execute a SORT-4 routine. (The specifics of the Data statements are explained in each XXXXMENU).

GBS MENU is the executable program for all of the GBS menus. This routine will allow the operator to select a program (task), check any control flags for the task, open any files required for the task, overlay the subroutine modules (DENTRY, KFAMCODE, etc.) and overlay the main program. Also, upon returning to the menu from a given main program, the menu will close all files previously opened and release all devices that were logged.

To accommodate the opening and closing of files, each GBS menu will overlay a subset of the KFAM routines specific to its needs. KFAMOpen will be loaded to open files and KFAMClos will be loaded to close the files.

5.4 NAMING VARIABLES

Specific naming conventions are employed for variables to increase understanding of the operation of various programs. Each variable used in GBS is named according to its specific use. Data file fields are assigned permanent variable names throughout the system. They start with a letter in the range A-P (with the exception of K) and end with either a blank space or a digit in the range 0-5 (A,A0-A5 through P,P0-P5). These variables appear in PUT/GET subroutines for the individual files. KFAM-7 specialized and KFAM-7 subroutines employ variables which start with a letter in the range Q-V. (There are exceptions: the Printer Availability routine uses the variables IO$ and @I4. The Page Eject routine uses the variables P1 for page count and L for line count.) The SORT-4 system variables are documented in the ISS Manuals and are of concern only when variables must remain in common throughout sorting. Variable names which change from program to program and which are described at the end of source program text, start with a letter in the range A-P and end with a digit in the range 6-9 and include all K variables. Variable names starting with a letter in the range X-Z are not used by GBS and, therefore, may be used by the installing vendor/consultant for any modifications. The Bill of Materials file variables start with the letter W.

5.5 STORAGE OF VARIABLES

The PUT/GET subroutines perform two functions: they $PACK/$UNPACK the file variables into or from a buffer and they read or write this buffer to or from a disk. Refer to the BASIC-2 Language Reference Manual for syntax specifications and examples. M$() is used as the disk I/O buffer, except when writing to the employee master file, and payroll control file in which case P$() is used.

5.6 LENGTH OF VARIABLES

The key field(s) for any KFAM file must be an alphanumeric variable, with a maximum length of 30 characters. If a GBS/MVP user already has an established numbering system for customers, products, or other records that belong in GBS files, GBS/MVP will accommodate numeric keys by treating them as alphanumerics. (This eliminates the need to change the numbering sequence.) To maintain the proper sequence for either alphanumeric or numeric keys, GBS/MVP provides for either left or right justification of the key after it is entered by the operator. A filler character for unused portions of the key may also be specified.

The length of the KFAM file key field(s) is specified at installation time. The size of these fields is automatically defaulted to the size of the standard GBS configuration but may be altered by the installing vendor. However, once the files have been created, the key lengths and right/left justification parameters should not be altered.

Once initialized, these parameters are stored in the company file in the variable, Q0$(). Each element in Q0$() is 3 bytes long. Byte 1 is the hexadecimal key length Q0$=BIN(Q2) where Q2 equals the length of the ID. Byte 2 is the fill character, and byte 3 is a switch for right or left justification (0=right justification, 1=left justification.)

Element Number	Type of ID
1	Customer
2	Inventory
3	Salesman
4	Vendor
5	Chart of Accounts
6	A/P Invoice Number
7	Profile ID
8	Employee ID
9	Bank Address

Whenever an operator enters a record ID, a GBS subroutine ('96) manipulates the data entered, and puts it into the appropriate form. For example, GOSUB '96(2) takes what is in Q6$, the input variable, uses Q0$(2) to manipulate the record ID entered and returns it to Q6$ in the correct Q0$(2) format.

The length of the ID is used throughout GBS to determine lengths of variables, to modify $PACK formats, and to determine appropriate blocking factors. Some examples are as follows.

- All the DISPLAY/PRINT programs have DIM statements that use Q0 to specify the length of starting, and ending product ID variables.

- Any STR (string) function that looks at specific sections of a record ID uses a variable to specify the position in the string variable. For example, the last character of the customer ID is a ship-to code controlled by GBS programs. The customer ID (byte # n+1) is stored in C$. Programs that look at the ship-to code might have the following statement.

```
BIN(STR(C$, Q0+1)) = INT((Q9+1)/2)+64
```

In this case, the ship-to number entered by the operator is Q9, and the last character of the customer ID (STR(C$,Q0+1)) is calculated from the ship-to number. Decimal 64 is added because GBS has always used the letter A as the ship-to code for the first two ship-to addresses, B for the next two, etc. (A is HEX(41), which is decimal 65).

- Arrays, or scalars, that serve as screen buffers use Q0, the ID length, to specify string lengths. For example, all the programs that work on the line items portions of orders and invoices store a screen full of product ID's in K3$().

  INVC030C (enter invoices, line items portion) has the following statement.

  ```
  DIM K3$(L9)Q0
  ```

  L9 is the number of lines/screen (defined in INVC030A as a common variable, and set to 15), and Q0 is the length of the product ID as determined by Q0S(1).

- Blocking factors for certain files depend on the length of a key. For example, the A/R Open Item file is a blocked file with 5 records per sector, which allows a maximum record length of 49 bytes.

In cases where product or customer lengths are made greater than the GBS default values, some screen and report layouts may have to be altered because of the change in the field length. In general, reports and screens show product descriptions right after product ID's. If the combined length of a product ID and description is too great, the columns that follow the two fields will be incorrectly placed. In cases such as this, GBS/MVP automatically truncates the description field and shifts it to the right. When the product ID is longer than 12 characters, a similar process is used for customer ID's and names. To simplify the task of modifying report layouts, programs such as DISPLAY/PRINT use variables to specify tab stops.

If oversized product ID's (other than the default as specified by GBS/MVP) and descriptions are necessary some information in audit files may be truncated by GBS/MVP programs. For example, the image ($PACK specification) for the Lost Sales file is defined as follows:

```
Q6$=HEX(A0) & BIN(Q0) & HEX(A0) & BIN(MIN(45-Q0,24)
```

This statement limits the amount of information in each lost sales record to 62 bytes. The fixed length portion of the record is 17 bytes long. This leaves 45 bytes for the product ID, and product description. The product description will always be 24 bytes, (as defined in the GBS system manual) unless the product ID length exceeds 21 bytes. In the unlikely event of product ID lengths greater than 21 bytes, the product description will be truncated. In cases such as these, truncating the field was deemed preferable to altering the blocking factor.

5.7 VARIABLE USAGE

The Start program declares and assigns values to all system variables. These variables are:

Variable Name	Description	Source
S2	Station number	#PART
Q1	Julian system date	@Q1
Q1$8	Gregorian system date	Q1
S	Company number	Operator
N2$60	Company name	Company file
M5$8	Program name	Operator & Data statements
M$8	Menu name	Menu programs
S0$8	Name of global partition	Test of $PSTAT
S0$8	Name of universal global partition	
Q0$(9)3	File ID parameters	Company file
S0$(15)1	Unique PUT/GET code	Company file
Q	Number of KFAM files to open	Menu programs
Q0	Principal file ID length	Menu programs
M2	Special function key chosen	Operator

In addition, a list of system work variables is declared. Most of these variables are used by the menus, and data entry subroutines. (M$(4)64, is worked extensively.)

Work Variable	Use
M$(4)64	Used of any DATALOAD BA. In particular, it is used by KFAM subroutines and replaces the @Q0$() array normally used by KFAM.
M$(4)62	Used whenever a buffer is needed to pass data from disk to memory or memory to disk.
M4(80)3	Used in the menu program when disk addresses are passed to the KFAMOpen (DEFFN '230) subroutine.
M$(2)124	Used when data is written to (or read from) the Transaction Audit file.
N	Used for device no. on Dataload statements.
I	All variables in common before I always remain in common.

5.8 SORTING

The SORT-4 setup module in GBS transfers data from GBS variables to the appropriate Sort variables. Small modifications were made to SORT-4 for GBS. Following all SORT-4 error messages STOP will not appear. Rather, SORT-4 will return to the appropriate GBS menu. In addition S is no longer used by SORT-4 as the ISS common variable to designate memory size. It is now used by GBS to designate the company number.

The M$() array remains in common throughout a Sort module. GBS uses M$() to pass information both to Sort and to programs that follow a setup preventing variable dimension conflicts between GBS and SORT. The six device addresses needed by the Sort setup module are placed in M$() by the GBS menu in the order required by Sort.

Following the loading of device addresses, Sort stores disk addresses in the variable M0$(). The Sort setup module passes M$() (first eighteen bytes) to the last eighten bytes of M0$() and sets the first 3 to the last 3 in M$(). Device numbers 0-6 are then selected to the disk addresses as specified in M0$(). The first 18 bytes of M$() are also passed to F$(), the variable used by Sort to store the disk addresses of the input file, key file, workfile, and output file. Finally, the Sort setup module uses S$() (the variable in which the menu places names of files) to pass the name of the input, output, and work file to N$(), the variable used by Sort to store file names. The Sort key positions and Sort key lengths take into account the variable length record ID's. All Sorts in GBS are tag Sorts with the output file usually being the Sortwork file.

Before a Sort, COMCLEAR M5$ clears out most of the common GBS variables. Following a sort, the subroutines needed by the program are overlaid and the KFAM routines to open and close files should be overlaid. Files involved in the sort should be opened after the sort. Before returning to the menu all files must be closed.

5.9 COMCLEAR PROGRAM

COMCLEAR, a dummy program, can be used at any time to discard unnecessary sections of a program resident in memory. For example, the menu overlays KFAMOpen, a subroutine containing a line of code that uses the variable Q to specify the number of elements in some array. Q depends on the number of KFAM files to be opened. The statement is coded as follows:

```
COM T5$ (Q)58
```

This statement needs only to be resolved once because Q may change with a successive overlay. Since a successive overlay may not have a line of code that overlays the COM statement, a dummy overlay of the COMCLEAR program clears the COM statement from memory. Furthermore, the same dummy overlay technique is used to clear the KFAMOpen logic from memory, after all the files have been opened and before the subroutines and main program are overlaid.

5.10 KFAM-7 ACCESS MODES

KFAM-7 offers four access modes: inquiry, read-only, shared, and exclusive. They only control access to the key files, never the data files. Their functions for GBS are more restricted than those for standard KFAM; specifications are as follows.

- Programs reading from a file but not altering data in the file can open the specified file only in the inquiry or read-only modes.

- Report programs accumulating totals can open the specified file in the read-only mode. For KFAM files, read only executes more quickly because record protection flags are not set or tested. See ISS Release 4.0 for further discussion.

- Data entry programs adding, changing, or deleting records can open the specified file in the shared mode.

- Sort programs must run one at a time since SORTWORK is opened in exclusive mode by the SORT-4 system.

The System Manuals include tables for each system which show the access mode used for each file in each program.

5.11 KFAM-7 ACCESS TABLES

Each GBS/MVP file contains an access table which contains file access information. Each table alerts the KFAM system as to the current processing mode for a specific file by each and every user in the system. The access table is always in the system control sector which is the last sector of the user file. The structure of the access table follows.

Byte Number	Content
1	A0 (HEX control byte indicating either the end of the file or the existence of the system control sector as defined by KFAM).
2-3	Number of sectors (in binary).
4	FD (Hex control byte indicating end of data).
5-8	"MUX" - This is stored in HEX.
9-16	Data file name.
17-32	Password. (Set to all 20 if not used).
33-80	Access mode per station: Spaces indicate a closed file; or access mode code for open file, one byte per station.

NOTE:

The DATASAVE DC END command should never be used on a KFAM file. It's execution rewrites the system control sector thereby destroying vital access information. The routine MUX END ('218) should be used instead; it simulates DATASAVE DC END.

5.12 FILE PROTECTION (ACCESS CONFLICTS)

Conflicts may occur among the access modes. The possible situations are detailed below.

Access Mode	Conflict Occurs When:
Inquiry	Another station has opened the same file in the exclusive mode.
Read-Only	Another station opened the same file in the shared or exclusive mode.
Shared	Another station opened the same file in the read-only or exclusive mode.
Exclusive	Another station opened file in any mode.

Access conflicts should only occur among stations, but they can occur for the same station when a file has been left open during a previous operation. (All programs normally end through the GBS escape routines, DEFFN'31, which close all files before returning to a menu. Operator initiated aborts are normally achieved by pressing SFK 31 which executes DEFFN'31.)

To discover the cause of an access conflict, the program Reset Access Table (UTIL010A) may run. If the conflict was a result of normal access mode activity by another station, a file status report (not related to the Start program) is displayed indicating the station and access mode in control. If the conflict was a result of an illegal program exit, no file status report is displayed and the access table is reset. In either case, a given station's access table can only be reset by that station.

The Reset Access Table program clears the disk access table for the station executing the program (see Section 5.7). If the file involved is a KFAM, the program does the following.

- Determines whether the global table of opened files (@T$()) contains an entry for this particular file and then removes the entry if the file is not presently open by another station.
- Checks the global table of protected sectors, removes entries for the particular station, and resets the appropriate access table.
- Removes entries remaining in the (internal) KFAM queue for that station.
- Rewrites the entire system control sector if the access table has been destroyed (see Section 5.7).

Only the portion of the access table corresponding to the station executing the Reset Access Table program is cleared. This program is a heavily modified version of the KFAM-7 utility and can be used at any time without interfering with other operations.

5.13 RECORD PROTECTION

In the shared or inquiry modes, records can be protected when the appropriate values are entered for the variable P in all KFAM subroutine calls. Protection lasts until a release is executed (GOSUB '238 (I)) or another KFAM subroutine is executed on the same KFAM file. Inquiry and data entry programs check the record protection value to determine whether or not another terminal is protecting the record. If data entry programs encounter a busy record, control returns to the prompt requesting the record. If a program updating records encounters a busy record, it enters a wait loop until the record is free. When an display/print or report program encounters a busy record during an All or Range inquiry, it enters a wait loop as well. The wait loop for display/print and report programs appears as follows.

```
5000 IF J0 = 7 THEN GOSUB '92
(This is executed after the first record busy signal is encountered)

5010 GOSUB '237 (1,0)
5020 IF Q$   "  " THEN GOSUB '91:ELSE J0 = 0

5030 ON J0 GO TO 6000, 7000, 7000, 7000, 7000 7000, 5000, 7000
5040 CONTINUE NORMAL PROCESSING
.
.
.

6000 END OF FILE PROCESSING
.
.
.
.

7000 KFAM ERROR PROCESSING
```

Subroutine '92 (in the first statement of the loop) checks for operator-initiated program abort and executes the Operator Wait routine (GOSUB '254). Subroutine '92 can be changed to transfer control to a $BREAK loop in order to eliminate operator intervention after a record busy signal.

5.14 PASSWORD PROTECTION FOR FILES

Besides specifying a company password, a password may be specified for every file when GBS is installed. Passwords should be used judiciously, since they will be required every time the file is opened. The KFAM and MUX Open subroutines have been modified to ask for a password. The password is requested only if specified in the Open subroutine call and if there was a password set-up when GBS was installed.

5.15 THE GLOBAL SUBROUTINES

Specialized Subroutines

Program Name	Subroutine Description
DENTRY	A specialized version of the ISS utilities Data Entry ('200) routine modified to use SFK 31 and FN key for program exit (after returning from '100, the calling program must check for a value of HEX(1F) in Q6$ for an indication that SFK 31 was pressed), SFK 15 (RECALL) to refresh the entry image, and SFK 4 (END) to enter the string END as a response for an alphanumeric entry or zero for a numeric entry.

GOSUB'100 (Q$(1), Q$(2), Q3, Q4, Q5$, Q5)

Q$(1) = Minimum value (validation list)*
Q$(2) = Maximum value (lowercase to uppercase translation table)*
Q3 = Integers left of decimal point
Q4 = Integers right of decimal point*
Q5$ = Prompt

Q5 = Input type (given to subroutine)

-1 = Numeric with default displayed; default must be in Q6$
0 = Numeric with default not displayed; default must be in Q9
1 = Numeric with default of 0
2 = Alphanumeric with default of spaces
3 = Alphanumeric with default displayed; default must be in Q6$

Q6$ = Returned to program by subroutine as response to numerics and alphanumerics

Q9 = Returned to program by subroutine either as response to numerics, or as the element number selected from the validation list

NOTE:

Some GBS programs INIT series, GBS and GBSM have some subroutines built in since global is not available when the programs are being executed.

* If alphanumeric input is selected when Q4 = 0, the subroutine performs minimum and maximum checking; if Q4 = 1, the subroutine performs a validation check against a list (Q$(1)) and translates the characters according to the translation table (Q$(2)). The validation list must consist of sets of valid responses exactly Q3 characters long separated by commas.

Program Name	Subroutine Description

SUBUTIL — A modified version of ISS utilities Operator Wait routine.

GOSUB'254

Q6$ = HEX(1F) returned by subroutine when operator presses SFK 31 or the FNKey to abort program execution

SUBDATE — A modified version of the ISS utilities Date routine.

GOSUB'121(U9$) U9 returned as Julian date
GOSUB'123(U9) U9$ returned as Gregorian date

PG EJECT — DEFFN '90, the page eject subroutine, is built into every GBS program that prints reports. It performs the following functions.

1. A KEYIN statement followed by a test for "P". At any time during the printing of reports, pressing the "P" will cause the program to pause, when '90 is executed. This happens immediately before each line, or block, of printed output. This is useful to straighten out paper jams, or insert new paper in the middle of long reports.

2. The same KEYIN statement accepts function keys. The subroutine tests for function key '31 to abort program execution.

3. Select the printer, with a SELECT PRINT<IO$> N statement.

4. Compare L, the actual line count, against LO, the maximum number of lines per page.

5. When L equals, or exceeds LO a PRINT HEX (OC) is issued. Following this, the company name is printed, as follows:

```
PRINT HEX(OE); TAB(3); N2$
```

Since the company name is 60 characters long, and since the Start program centers the company within the variable N2$, the company will be centered at the top of the page, in expanded print. This assumes 132 character-wide reports.

Program Name	Subroutine Description

6. Certain reports are only 80 characters wide and the company name is not expanded if it is more than 40 characters long. Even then, N2$ must be manipulated somewhat before printing.

7. Print report title and page headings.

GOSUB'90 This routine performs a page eject whenever L (line count) equals (or exceeds) LO (maximum number of lines/page).

PRNTAVAL A system subroutine, (DEFFN '93) used by every program that attempts to access the printer.

The subroutine performs the following functions.

1. Accept a printer address from the operator. A validation list of 204, 215, 216, 005, and " " determines which printer can be used. The consultant may wish to either add to this list of addresses, or eliminate the question when only one printer is available. Note that 005 dumps to the screen anything normally going to the printer without any pauses and is useful to bypass required print operations.

2. Select device 15, to be used in a $OPEN 15 to hog the printer. After this, device 15 is not used (see Section 2.3).

3. If the printer is available, place the printer address in IO$. If not, allow a different printer address to be entered. A blank printer address means no printer is available.

4. Display a paper-mount message passed to '93,) i.e., GOSUB '93 (MOUNT INVOICE FORMS). The default message is MOUNT PAPER INTO PRINTER, and is used if a blank message is passed to '93.

5. Test the printer selection with a $GIO sequence that sends data to the printer and waits for a ready condition. This is not performed with a terminal printer (204) since a terminal printer always appears to be ready. Instead, a prompt reminds the operator that the terminal printer must be selected.

6. Set L and LO to 56, the maximum number of lines/page. If shorter, or longer, paper is used, the only change required in the system is a change to LO.

   LO is the maximum number of lines per page, and L is the line count. To force a top-of-form, L is set equal to LO within this subroutine.

   After return from PRINTAVAL the user program can use IO$ to test printer availability, and to select the printer with a SELECT PRINT IO$ N| statement.

GOSUB'93 (Q6$) Printer Availability

```
Q6$ = Paper mount message; default is MOUNT PAPER INTO
      PRINTER if Q6$ is blank
IO$ = Returned by subroutine as printer type;
@IO$= One byte/printer; indicates which station is
      using the printer
LO  = Maximum number of lines/page.  Set = 56
```

KFAMCODE Translates the KFAM-7 return codes to numeric values (the GBS variable J0) and prints a corresponding screen error message based on J0 (J$(J01)). The messages are stored in the array (J$()) and are suppressed by setting the appropriate element to blank.

GOSUB'91

Return codes in Q$ for all KFAM subroutines:

```
   Q$ = J0  J$(J0)
BLANK = 0 - No Error
    E = 1 - End of File (FINDNEXT only)
    X = 2 - Improper Call (Improper KFAM call)
    N = 3 - Record Not Found (Null file or key not
            found)
    D = 4 - Duplicate Key
    S = 5 - No More Space
    A = 6 - Access Conflict
    B = 7 - Record Busy
    P = 8 - Invalid Password
```

After '91 informs the operator that a busy record has been encountered, GOSUB '92 enables the operator to exit from a display/print routine.

GOSUB'92 KFAM Busy Record Routine

The KFAM subroutines that open and close files are overlaid by the menus program (KFAMOpen and KFAMClos). The remaining KFAM subroutines are in the global partition. Any program that requires KFAM files must set Q to the number of KFAM files to be opened prior to overlaying the KFAMOpen routines.

The following subroutines are standard KFAM (refer to the ISS Release 5.0 User Manual for a detailed examination of each one). The COM and Variable program for all these subroutines is KFAMOpen.

GOSUB'217 (F$, F, C, S, A, Q7$, A1$, H) MULTIPLEX OPEN

F$ = File name
F = File device number (0-15)
C = CPU number (or station number)
S = Type of OPEN (0, 0, 0)
A = Access mode (1-4)
A1$ = Disk address
Q7$ = Description of file, if password entry is required
H = Hog mode (1 or 0)

GOSUB'218 (F$, F, A1$, H) END MULTIPLEX FILES

F$ = File name
A1$ = Disk address
F = File device number (0-15)
H = Hog mode (1 or 0)

GOSUB'219 (F$, F, C, A1$, H) CLOSE MULTIPLEX FILES

F$ = File name
F = File device number (0-15)
A1$ = Disk address
C = CPU number (or station number)
H = Hog mode (1 or 0)

GOSUB'230 (I, J, K, L, N$, A, Q7$, "YYY", "XXX") OPEN

I = Key file ID number used in KFAM subroutines (1-3)
J = Refer to SELECT statement key field device number*
K = User file device number
L = Key file number (multiple key files) (1-9)
N$ = Data file name
A = Access mode
Q7$ = Description of file, if password entry is desired.
"YYY" = Device address of Key file
"XXX" = Device address of User file

* A modified version of the KFAMOpen subroutine provided with GBS/MVP ignores the key file device number and allows it to use device number 14. (See Section 2.3.)

```
GOSUB'231 (I, P, A$) DELETE

       I = Refer to OPEN statement (GOSUB '230)
       P = Record protect option
      A$ = Record key

GOSUB'232 (I, P, A$) FINDOLD

       I = Refer to OPEN statement (GOSUB '230)
       P = Record protect option
      A$ = Record key

GOSUB'233 (I, P, A$, 0) FINDNEW

       I = Refer to OPEN statement (GOSUB '230)
       P = Record protect option
      A$ = Record key
       0 = Not used

GOSUB'234 (I, P, A$, 0) FINDNEW (HERE)

      I = Refer to OPEN statement (GOSUB '230)
     A$ = Record key
      P = Record protect option
      0 = Not used

GOSUB'235 (I, P) FINDFIRST

      I = Refer to OPEN statement (GOSUB '230)
      P = Record protect option

GOSUB'237 (I, P) FINDNEXT

      I = Refer to OPEN statement (GOSUB '230)
      P = Record protect option

GOSUB'238 (I) RELEASE

      I = Refer to OPEN statement (GOSUB '230)

GOSUB'239 (I) CLOSE

      I = Refer to OPEN statement (GOSUB '230)
```

PUT/GET Subroutines

For all subroutines, N = SELECT disk device number

Program Name	Subroutine Descrption	
CUSFILE	GOSUB'40(N, A$)	Customer Master File PUT
	GOSUB'50(N, A$)	Customer Master File GET
	A$ = Record type (Spaces = Master, 1 = Ship to)	
OPNFILE	GOSUB'41(N, Q)	A/R Open Item PUT (KFAM)
	GOSUB'51(N, Q)	A/R Open Item GET (KFAM)
	Q = Record number within data record	
INVFILE	GOSUB'42(N)	Inventory Master PUT
	GOSUB'52(N)	Inventory Master GET
FINFILE	GOSUB'68(N)	Inventory Master-PUT with Inventory Management Variables
	GOSUB'78	Inventory Master GET with Inventory Management Variables
SLSFILE	GOSUB'43(N)	Salesman Master PUT
	GOSUB'53(N)	Salesman Master GET
TRNFILE	GOSUB'44(N, Q9)	Invoice Transaction PUT
	GOSUB'54(N, Q9)	Invoice Transaction GET
	Q9 = Record type (1 = A1, 2 = B1, 3 = C1)	
SOPFILE	GOSUB'45(N)	A/R Open Item PUT (Sequential)
	GOSUB'55(N)	A/R Open Item GET (Sequential)
A/RCONT	GOSUB'46(N, Q9, F5)	Control PUT/GET
	F5 = Operation type Q9 = Julian date (1 = Read, 0 = Write)	

Program Name	Subroutine Description	
ORDFILE	GOSUB'47(N, Q9)	Open Order PUT
	GOSUB'57(N, Q9)	Open Order GET
	Q9 = Record type (1 = Header, 2 = Line items)	
TRAUFILE	GOSUB'48	Transaction Audit File
AUDIFILE	GOSUB'49	Maintenance Audit File
BOMFILE	GOSUB'60(N)	Product Structure PUT
BOMFILE	GOSUB'70(N)	Product Structure GET
VENFILE	GOSUB'60(N)	Vendor Master PUT
	GOSUB'70(N)	Vendor Master GET
A/PFILE	GOSUB'61(N, Q)	A/P Open Item PUT
	GOSUB'71(N, Q)	A/P Open Item GET
JENFILE	GOSUB'63(N, E6$)	Journal Entry PUT
	E6$ = Name of Journal Entry file	
CHKFILE	GOSUB'64(N)	A/P Check PUT
	GOSUB'74(N)	A/P Check GET
GENFILE	GOSUB'65(N)	Chart of Accounts PUT
	GOSUB'75(N)	Chart of Accounts GET
A/PCONT	GOSUB'66(N, Q9, F5)	Mod III Control PUT/GET
	Q9 = Julian date F5 = Operation type (1 = Read, 0 = Write)	
DAYFILE	GOSUB'62(N)	Employee Master PUT
	GOSUB'72(N)	Employee Master GET
BNKFILE	GOSUB'67(N)	Bank Address PUT
	GOSUB'77(N)	Bank Address GET

Program Name	Subroutine Description	
PAYCNTA	GOSUB'68(N, Q9, F5)	Payroll Control PUT/GET
	Q9 = Julian date F5 = Operation type (1 = Read, 0 = Write)	
PAYCNTB	GOSUB'78(N,Q9,F5)	Payroll Tax Switch PUT/GET
	Q9 = Julian date F5 = Operation type (1 = Read, 0 = Write)	
PROFILE	GOSUB'69(N)	Profile PUT
	GOSUB'79(N)	Profile GET

5.16 CREATION OF UNIQUE PUT/GET ROUTINES

For any one company a special PUT/GET routine may be created and accessed exclusively for that company by setting the appropriate SO$() variable in the company file (FILEF01X). By placing the PUT/GET routine code into the file definition module, the menu could load the unique PUT/GET routine based on the code in the SO$(). For example, if a unique customer record is required for Company 9, SO$() could be set to 9, concatenated to the PUT/GET routine "CUSFILE" and SO$() and accessed during the processing of Company 9. The following list displays the relationship between the SO$() elements and the PUT/GET routines. (Refer to Section 5-2 for further discussion.)

Element Number	Subroutine
1	CUSFILE
2	OPNFILE
3	INVFILE
4	SLSFILE
5	TRNFILE
6	SOPFILE
7	ORDFILE
8	A/PFILE
9	JENFILE
10	CHKFILE
11	GENFILE
12	PAYFILE
13	BNKFILE
14	PROFILE
15	BOMFILE

CHAPTER 6
CREATION OF BACKUP FILES

6.1 THE BACKUP PROCEDURE

The GBS/MVP system provides a mechanism for creating backups of both the data disks and the system disks, based on the assumption that the files are located on disk addresses 320/B20. To provide for proper control of the backup function the Release 2.0 version of GBS/MVP utilizes a label file to control the backup and restore function.

The label file (Diskname) is created by the vendor once the system has been initialized for use. Each platter to be used in the system should be properly labelled to reflect the contents and the status of the platter as well as the device address (the platters are defaulted to 320-B20 by the program). To intialize the label file the program "NAMEDISK" should be executed. Once initialized the system is ready for the backup procedure.

The backup (or restore data) procedure can only be executed from terminal number one and only after all terminals on the system have been logged off via End of Processing. Once this is completed the backup can take place for either the data disk, the system disk or both or the data can be restored from a backup copy.

The backup program (BACK010A) executes a new marked subroutine to control each step of the backup. It is as follows:

```
GOSUB'34(A6$,A7$,Q6$,B7$,B8$,B9$,Q7$,C7$,C8$,C9$)
```

where

```
A6$ =   Partial label for input platter
A7$ =   Platter type (Data or System)
Q6$ =   Master or backup platter
B7$ =   Label checking for input platter (Y/N)
B8$ =   Address of the input disk
B9$ =   Input platter verification prior to copy (Y/N)
Q7$ =   Output disk
C7$ =   Label checking for output (Y/N)
C8$ =   Address of the output disk
C9$ =   Verify output platter after copy (Y/N)
```

The functionality of this routine permits the vendor to change any of these parameters to reflect particular needs and requirements for backup. To establish the level of checking within any step of the backup the vendor must manipulate each occurrence of the GOSUB'34 to reflect particular requirements.

APPENDIX A
@GENPART and $GENPART

A.1 CONFIGURATION AND RECONFIGURATION PROCEDURES

@GENPART is an automatic configuration utility and therefore transparent to all users. The steps involved in its execution are enumerated below for the interested reader. Steps to change the standard configuration are provided as well.

- The operating system is loaded from disk when the appropriate SF Key is pressed (at terminal 1). The operating system looks for and executes @GENPART.

- @GENPART looks for @SYSFILE (where configurations are stored and which arrives on diskette with @GENPART) to configure the system.

- A list of configurations is displayed on the screen for selection by the operator.

- When a system configuration in @SYSFILE matches line 1050 of @GENPART, which sets A$ equal to a configuration name and B$ equal to the system reconfiguration password, the system is configured without operator intervention.

- If @SYSFILE is not on disk (it somehow has not been copied from diskette) or @SYSFILE does not match line 1050 of @GENPART, @GENPART links onto $GENPART (the original MVP configuration utility) which requests specifications from the operator for the creation of a configuration.

To change the standard configuration, $GENPART* must be executed. Observe the following steps:

- Touch RESET at terminal 1.

- Enter LOAD RUN "$GENPART" and touch RETURN.

- Modify the configuration desired, and save it.

To reconfigure the system without powering down, observe the following steps:

- Touch RESET at terminal 1.

- Enter $INIT "SYSTEM" (the reconfiguration password) and touch RETURN. The operating system can be loaded again.

* $GENPART is the original version of @GENPART as supplied by Wang Laboratories. It was deemed appropriate to modify @GENPART for the purpose of reducing the amount of operator intervention necessary to configure the system.

APPENDIX B
INTEGRATION OF THE INVENTORY MANAGEMENT SYSTEM WITH GBS

This appendix provides a brief overview of the steps necessary for integrating the Inventory Management System into the General Business System. A detailed description of the procedure may be found in the Inventory Management System User Manual.

B.1 PROGRAM CHANGES

The integration of the Inventory Management System with GBS involves a number of GBS program changes. The specific programs to be changed include: Maintain Inventory File, Display/Print Inventory, Inactive/Low Stock Report, Recommend Purchase Orders, and Clear Files. These modified programs are provided with the Inventory Management System. The modified programs are provided with a different name than the originals (the fifth character has been changed to a 2) to facilitate integration. In addition, a number of new programs have been written to support the functions unique to Inventory Management. Two menus (Accounts Receivable and Inventory) were modified to support the new and the modified programs.

Furthermore, the addition of Inventory Management to GBS requires one new data file, the Profile File along with the addition of new fields to the Inventory master record. Profile is supported by all of the standard GBS file handling and initialization routines. The additional fields are handled by a new pair of PUT/GET subroutines ('68 and '78) using a $PACK specification initialized in module FINFILE. FINFILE and subroutines '68 and '78 are designed to augment, rather than replace, INVFILE and subroutines '42 and '52. The subroutines '68 and '78 use the subroutines '42 and '52 to read and write the record and to $PACK and $UNPACK the leading portion of the record. The $PACK specification in FINFILE begins with a skip specification to skip over the leading portion of the record. Thus, changes may be made to the leading portion of the record without affecting FINFILE, '68 and '78, provided these changes don't carry the record into the portion used by the Inventory Management System variables.