MEMMAN

Voyager CCS Memory Management Software

Functional User's Guide

Revised May 15, 1990

Jet Propulsion Laboratory

Gregory F. Welch

Table of Contents

1. Statement of Purpose 1

2. Introduction 2

3. System Environment 3
3.1 Hardware Required 3
3.2 Directory Structure 3

4. Program Overview 4
4.1 Automatic Management 4
4.2 Manual Management 5
4.3 Management via the Force File 6

5. Input 8
5.1 Keyboard Input 8
5.2 File Input 8

6. Output 9
6.1 Printouts 9
6.2 File Output 9

7. Use Instructions 10
7.1 Starting the Program 10
7.2 Using Automatic Management 10
7.2.1 Auto Screen 10
7.2.1.1 User Name 11
7.2.1.2 Input SRF Version 11
7.2.1.3 Category Status 12
7.2.1.4 Load Boundaries 12
7.2.1.5 Master Processor 13
7.2.1.6 Automatic Goal 13
7.2.1.7 Output File/Format 14
7.2.1.8 Comment Status 14
7.2.2 Automatic Attempt 15
7.2.2.1 Terminate an Endless Loop 15
7.2.2.2 Where Now? 15
7.2.2.3 Status Prompt 16
7.2.2.4 Bypass Automatic Management 16
7.3 Using Manual Management 16
7.3.1 Manual Screen 17
7.3.1.1 Line Number 17
7.3.1.2 Line Color Significance 18
7.3.1.3 Processor ID Column 18
7.3.1.4 Moved Command Markers 18
7.3.1.5 Present Word Count 18

7.3.1.6 CHKPNT SET 0 Status 19
7.3.1.7 Select Status 20
7.3.2 Manual Attempt 20
7.3.2.1 Move a Single Command 20
7.3.2.2 Move a Cyclic 20
7.3.2.3 Move a Group of Commands 21
7.3.2.4 Retract a Single Command 21
7.3.2.5 Retract a Cyclic 21
7.3.2.6 Retract a Group of Commands 22
7.3.2.8 Print Annotated SRF 22
7.3.2.9 String Search 22
7.3.2.10 View Command Cost 23
7.3.2.11 Go To a Line 23
7.3.2.12 Toggle CHKPNT SET 0 23
7.3.2.13 Re-attempt Automatic Management 24
7.3.2.14 Quit & Generate Output 24

8. Output SRF 26

9. Output Runstream 27

10. Force File 29
10.1 Force File Format 29
10.2 Force File Commands 29
10.3 Force File Maintenance 30
10.4 Re-using the Force File 30

11. Seqtran Macro Description File (SMDF) 31
11.1 SMDF Format 31
11.2 SMDF Commands 31
11.3 SMDF Maintenance 33

12. Syspar File 34
12.1 Syspar File Format 34
12.2 Syspar File Commands/Variables 34

13. Category File 36
13.1 Category File Format 36
13.2 Category Commands 36
13.3 Category File Maintenance 37

14. Log File 38
14.1 Log File Format 38
14.2 Log File Maintenance 38

15. Configuration File 39
15.1 Re-using the Configuration File 39

Appendix A: Sample ASCII Files 40

A.1 Sample System Parameter File 40
A.2 Sample Category File 41
A.3 Sample SMD File 42
A.4 Sample Force File 43
A.5 Sample Log File 44
A.6 Sample Output Runstream 45

Appendix B: Glossary 46

1. Statement of Purpose

The purpose of this document is to provide the user with instructions for using the CCS memory management program, MEMMAN. In addition, this document is intended to serve as a reference manual while using (or preparing to use) MEMMAN.

It is assumed that the reader has a an understanding of what memory management is, with respect to the processing of an SRF by SEQTRAN. In addition, it is assumed that the user understands the topics of CCS time-event regions, master tables, current & overlap regions, checkpoints and breakpoints.

This document will describe the MEMMAN memory management software, and explain its use.

2. Introduction

Although each Voyager spacecraft is controlled by several computers, the main supervisor of the spacecraft is the Computer Command Subsystem (CCS). Each Voyager actually has two CCSs, each of which has its own memory for storing events (commands) to be executed. A list of commands which are to be executed over a period of time make up what is known as a *sequence* for the spacecraft.

In general, it is desired that each sequence would have its instructions somewhat equally divided between each of the two CCSs, which operate simultaneously. The existing sequence translator (SEQTRAN) uses a Sequence Request File (SRF) as input, and produces a spacecraft Desired Memory Words File (DMWF) as output. Therefore, it is SEQTRAN which assembles commands into either processor (CCS), depending on initial default settings, or pre-planned redirection via user input (CONFIG macros inserted in the SRF).

MEMMAN uses the same input as the SEQTRAN operator has in the past; an SRF and a set of rules by which he/she would decide what could be moved. However, in a fraction of the time needed to run SEQTRAN, the system will (with interaction from the user) develop either a new SRF or a Univac SRF editor runstream, either of which should provide a successfully managed sequence, with the first SEQTRAN run. The system also provides the user with a pre-SEQTRAN word count for the sequence, and a graphical picture of how that sequence will be organized in CCS memory.

Before reading this document, it is important that the reader understands the phrases *moving* a command and *retracting* a command as they apply here.

Moving a command should be thought of as moving a command from the master processor, to the slave processor. *Moving* a command actually refers to the act of placing the appropriate CONFIG statements in the SRF, directing SEQTRAN to assemble the command in the slave processor.

Retracting a command should be thought of as moving a command from the slave processor, where the command had previously been moved by the system, back to the master processor. *Retracting* a command refers symbolically to the act of removing the appropriate CONFIG statements in the SRF, which previously directed SEQTRAN to assemble the command in the slave processor.

Keeping in mind the fact that the action of *moving* and *retracting* commands (or events) is really the action of moving them back and forth between processors, it is important to remember that because all commands initially default to the master, a command can only be *retracted* by MEMMAN if it was previously *moved* by MEMMAN.

Finally, the operator should be aware that MEMMAN's ability to successfully manage the memory is tightly governed by the amount of freedom the operator gives it, and the accuracy of the input provided by the user. By carefully checking the memory limits given to the system, and by knowingly choosing events which are allowed to move, the operator should be able to create either a new SRF or a working runstream which when used on the original SRF will result in a successfully managed sequence.

3. System Environment

In order to use MEMMAN, certain hardware is needed, as well as a specific directory structure. The following two sections (3.1 and 3.2) will address these topics. The hardware requirements are specified as a minimum configuration, but can be upgraded as desired (for speed, etc...). The directory structure described below will be the assumed directory structure throughout the rest of this document. If the user is not satisfied with the directory structure shown below, they can change it to whatever they wish. They must however update the system parameter file (Syspar) to reflect any new directory structure (see 12 Syspar File).

3.1 Hardware Required

The following list describes the minimum hardware requirements in order to run MEMMAN.

1) IBM PC-AT
2) 640 KByte RAM
3) 20 MByte fixed disk drive (assumed C: below)
4) CGA graphics adapter (color)
5) CGA monitor (color)
6) Internal clock/calendar
7) Math coprocessor chip
8) Dot matrix printer

3.2 Directory Structure

The following list describes a suggested directory structure to be used with MEMMAN. This directory structure can be altered, but the SysPath and DataPath variables in the Syspar file must be updated to reflect the new DOS paths (see 12 Syspar File).

C:\SEQ-TOOL
C:\SEQ-TOOL\MEMMAN

MEMMAN.EXE	MEMMAN executable (program
MEMMAN.SYS	System parameter file (Syspar)
MEMMAN.CAT	Category file
MEMMAN.SMD	SEQTRAN Macro Description File (SMDF)
MEMMAN.LOG	Run history file (log file)

C:\SEQ
C:\SEQ*SequenceID*

SRFversion.SRF	Input SRF
SRFversion.FRC	Force file for above version of SRF
SRFversion.CON	Configuration file for above version of SRF
SRFversion.CFG	Output runstream (optional method of output)
OutSRFversion.SRF	Output SRF (optional method of output)

4. Program Overview

The MEMMAN program provides basically three methods of managing the CCS sequence memory; *automatic* management, *manual* management and management *via the force file*. Both *automatic* and *manual* management have screens (displays) which generally belong to them, while the *force file* is simply an input file (optional and unique to each sequence) which can be edited and used to affect a run for a particular sequence.

Generally the user would allow MEMMAN to attempt to automatically manage the sequence memory, with the force file and the manual management options being reserved for special cases which [for one reason or another] can't be managed automatically. However, even under normal circumstances the user is eventually brought to the manual management screen after the automatic attempt is made by the program (whether it fails or not).

The automatic management system makes repeated attempts at moving commands back and forth between CCS processors, while all the while maintaining the current word count to see if the commands are yet distributed according to the specified *automatic goal*. The *automatic goal* can be either to move enough words from the master processor to the slave so that the master is loaded just under its limit (referred to as *just under*), or (more commonly) the goal can be simply to balance the command distribution between the two processors (referred to as *balanced*). The manual management system provides the user with a more in depth look at the result of the automatic attempt, and a chance to make minor modifications for any particular reason. Here, the user can *check the results of the automatic attempt, and then manually* move individual commands or groups of commands back and forth between the two processors as needed.

No matter what method of management the user chooses, the aim of the program is always to produce a new output file which is either a new *managed* SRF, or a Univac runstream which can then be used to place CONFIG macros in (to *manage*) the original input SRF. When the user is satisfied with the results of the management attempts (either manual or automatic) they can press a single key to begin the process which will generate the new output file, and then exit the program.

The program begins by bringing the user to the Auto screen (automatic management input screen). The user is then required to input some basic sequence-specific information such as the SRF version for the current sequence, memory load boundaries, etc.... It then allows the user to further control the management path taken by instructing the program to either go with the automatic management attempt, or to bypass the automatic attempt and proceed directly to manual management.

4.1 Automatic Management

When attempting to automatically manage the memory, the system begins by reading (and parsing) the SRF into the computer's memory. Next, based on certain rules governing which commands are allowed to move, the program will attempt to move commands back and forth between processors, until a satisfactory state is reached. This attempt at memory management is referred to as the *automatic attempt*.

The desired state of memory required to satisfy the Auto management, is determined by the selected *automatic goal* (from the Auto screen). If the goal selected is *balanced*, then the Auto system will repeatedly move (and retract if needed) commands until the amount of free memory in each processor is as close to the opposite processor as possible, (e.g. 50 free words in CCSA and 51 free words in CCSB).

If the goal selected is *just under*, then the Auto system will repeatedly move (and retract if needed) commands until the master is loaded just under its limit. At that point, the user is offered three choices; first they can accept the *just under* status. Second, they can continue to manage towards a balanced goal. Or third, they can complete the attempt by choosing to move a specified number of words from the master to the slave (must be less than or equal to the number of free words in the slave).

Whether the automatic goal is *balanced* or *just under*, eventually the attempt will succeed, or come as close as possible. In either case, the user will be notified of the results, and will be brought to the manual management screen. *In a very rare case during an automatic attempt the Auto system will begin to loop endlessly (due to circumstances which do not allow the balanced or just under goal to be reached) and must be interrupted by pressing a key on the keyboard (see 7.2.2.1 Terminate an Endless Loop).*

If a force file (*SRFversion*.FRC) is available for the sequence being managed, then that file is processed (parsed and read into memory) prior to reading the SRF into memory. In this case, any macro commands encountered in the SRF which are referred to in the force file will be forced into the specified processor as they are read from the SRF.

In general, the steps taken by the Auto system are similar to those taken by the CCS programmer when managing the memory without the aid of MEMMAN. There are repeated attempts (by the program) at finding the best command(s) to move, moving the command(s), and then checking the word count to see if the appropriate goal has been achieved.

If the Auto system fails to achieve the *automatic goal* specified in the Auto screen (either *balanced* or *just under*) then the user will be presented with a choice. They have the opportunity to either return to the Auto screen to re-enter parameters, re-select moveable events, etc..., or if there is no desire to re-attempt the AUTO management, the user will be presented with the Manual screen so that they can continue management manually, (see 4.2 Manual Management).

If the operator wishes to entirely skip the automatic portion of the system, they can go directly to the manual portion, which will display a word count for the SRF as it appears in its un-managed state. See 7.2 Using Automatic Management for further instructions on using the automatic management.

4.2 Manual Management

The manual management system provides the user with a screen that offers a visual representation of the SRF, and the present command allocations in the CCS memory. The main purpose of this screen is to give the user a chance to manually move a command (or a block of commands) and then easily retract it (or them) if the results weren't satisfactory. In addition, the user has many helpful options available, most of which are discussed briefly in the following paragraphs.

At any time, the operator can view the results of manual command movement by pressing a special key, which causes the calculation of the latest word count and updates the word count totals on the screen header. Also, the user is provided with many methods of moving throughout the SRF. They can use standard cursor control keys, or they can choose to go directly to a specific line (specified by number). They also have the ability to search through the SRF for any string of characters (a command name, a time, a parameter, ...any string of characters). Once pointing at a command, the user can then press a single key to display its cost, which may assist in making a decision about whether a command should be manually moved or not.

At any time during the manual management the user may also choose to print an *annotated SRF*. The result would be the printing of a complete SRF, with the addition of all of the information available on the manual management screen such as each command's processor, its move status and the total word cost. It may often be helpful for the user to view the sequence on a single printed listing, with the necessary memory management related information.

Finally, the operator is always free to switch back and forth between both the automatic portion of the program, and the manual portion, allowing them to obtain results which are truly satisfactory. For example, if the automatic management attempt fails, the operator can continue by manually moving commands or they can switch back to the Auto screen and retry the automatic management attempt, with possibly new rules governing which categories can be moved. Keep in mind however that if commands have been selected to move manually, and the automatic management is then re-attempted, those items selected to move manually will remain tagged as being moved and will (most likely) affect the automatic management attempt.

Because the user is always brought to the manual management screen eventually in the program, it is from there where the user issues the final command to *quit* and *generate the output*. Even if the user does not wish to make any manual adjustments to the management, they must (will) always visit the manual management screen in order to generate the MEMMAN output (see also 7.3.2.14 Quit & Generate Output). See 7.3 Using Manual Management for further instructions on using the manual management screen.

4.3 Management via the Force File

A final addition to the management system allows the use of a force file to specify a processor for any occurrence of a command, (determined by a string of characters), which will have precedence over either of the two above-mentioned methods of management.

Generally, if the user chooses to bypass the automatic management attempt and to go directly to the manual management screen, the effect is simply to display the word cost that would result from running SEQTRAN with the un-managed input SRF (no management being done). It is possible however to prepare a force file for the sequence which is so complete that it removes the necessity of using the automatic management. Such was the case during the Neptune encounter (August, 1989) when virtually every command was routinely forced into either processor, in order to prepare for possible power-on-resets (PORs). In this case, a force file was created once, copied to a new file for each sequence where PORs were a concern, and simply re-named every time there was a new version of each of those respective SRFs.

To force events into either processor, the user first creates an ASCII text file which contains information about commands to appear in a particular processor. Any alpha/numeric string of characters can be used to force events into a processor. The string of characters is used as a mask for records read from the SRF. If a match for the mask is discovered, then that command is forced into the appropriate processor.

For example, placing the instruction `FORCE CC3A:A` in the force file will cause the character mask "CC3A" to be checked against every record read from the SRF. If a record is read which contains the four characters "CC3A" (together in a row) it will be automatically forced into the A processor.

The result of forcing a command depends on the current config status at the time when the command to force was read. If the current config status is to the slave processor, and a force to the slave processor is required, then the command is simply marked as unmovable, and left alone when the output is generated. On the other hand, if the current config status is to the master processor, and a force to the slave processor is required, then

the command is not only marked as unmovable but additional commands will also be placed in the output which will direct SEQTRAN to place the command in the slave processor.

It is possible that a Force instruction could be ignored under certain circumstances. For example, a conflicting Force statement in the force file will result in only the first one (first match in the force file) being honored. Also, an attempt to force an event into a processor which disagrees with that command's *processor* parameter (if such a parameter is a part of the command) will be ignored. In both cases, the user will be notified of the problem.

A Force instruction can have known exceptions handled by using the Except command in the force file. A virtually unlimited (limited by memory) number of exceptions can be specified for any single Force command. Each exception contains simply the Except command with an associated exception mask (string). Exceptions apply only to the Force command immediately preceding them. See ? Force File for more information on implementing the force file.

5. Input

Input to MEMMAN is provided in only two fashions; in the form of keyboard input (by the user) or in the form of file input. The following two sections will list the various types of input to MEMMAN.

5.1 Keyboard Input

Because MEMMAN is somewhat interactive, throughout the program operation there are many instances where the user is asked to respond (with keyboard input) to a message or a decision. The following list of keyboard input items is only intended to cover what is considered to be significant (some minor input has been omitted).

a) Input SRF version
b) CCS memory load boundaries
c) Master processor
d) Management goal
e) Command category status
f) Output filename/format
g) Comment display status
h) Manually selected commands to move

5.2 File Input

Almost every input file is exclusively an input file, with the one exception being the *configuration file*. This file is *created* by MEMMAN (output), and then later re-read as input (see 15. Configuration File). The following is a complete list of all of the MEMMAN input files.

a) Input SRF (*InSrfVersion*.SRF)
b) Seqtran Macro Description File (MEMMAN.SMD)
c) Force file (*InSrfVersion*.FRC)
d) Configuration file (*InSrfVersion*.CON)
e) Category file (MEMMAN.CAT)
f) System parameter file (MEMMAN.SYS)

6. Output

Output from MEMMAN is generally provided in three forms; printouts, files and the screen. The screen output will not be discussed here. The following two sections will list the other two MEMMAN output forms.

6.1 Printouts

MEMMAN provides several printed listings via the parallel printer port (LPT1:). At the moment, MEMMAN is set-up to use an Epson LQ1500 printer. The following is a complete list of all of the MEMMAN printouts.

a) Annotated SRF
b) Management run history worksheet.
c) Management runstream listing for the Univac @ED editor.

6.2 File Output

Almost every output file is exclusively an output file, with the one exception (again) being the *configuration file*. This file is created by MEMMAN (output), and then later re-read as input (see 15. Configuration File). In addition, the log file is not re-created with every run, it is simply appended to. The following is a complete list of all of the MEMMAN output files.

a) Configuration file (*InSrfVersion*.CON).
b) Univac @ED editor runstream file (*InSrfVersion*.CFG)
c) Output SRF (*OutSrfVersion*.SRF)
d) Log file (MEMMAN.LOG)

7. Use Instructions

7.1 Starting the Program

To begin the MEMMAN program, the user must first assure that the directory structure complies with that discussed in 3.2 Directory Structure. If it doesn't, then the Syspar file should be modified to reflect the new directory structure (see 12. Syspar File). To begin the program, use the following steps.

1) Type `CD \SEQ-TOOL\MEMMAN`
2) Type **MEMMAN** and press <Enter>

Both the automatic and manual management systems have steps which should normally be followed in a certain order. The following sections (7.2 Using Automatic Management and 7.3 Using Manual Management) will describe the steps to be taken in order to use the respective parts of the program.

7.2 Using Automatic Management

Use of the automatic management system is fairly straightforward, or automatic. To begin the management however, the program needs some basic information from the user, which is gathered at the Auto screen. In addition, the user must be aware of input that might possibly be required during the automatic attempt. The following two sections (7.2 Auto Screen and 7.2.2 Automatic Attempt) will provide the user with instructions relating to these two areas of concern.

Table I: Valid Auto Screen Keys

Key(s)	Description
<F1>	Help screen
<F3>	Go to manual management
<Esc>	Abandon MEMMAN
Arrows	Point to categories
<SP>	Toggle category status
<U>	Enter user name
<S>	Enter SRF version
<B>	Enter load boundaries
<M>	Toggle master processor
<A>	Toggle automatic goal
<O>	Toggle output file/format
<C>	Toggle comment status
<G>	Go - automatic management

7.2.1 Auto Screen

The automatic management is really the nicest feature of the program, and is therefore the default management method when the user begins. The program always starts by displaying what is called the Auto screen.

The Auto screen is an input screen which requires the operator to provide the following information:

1) The five character SRF version (e.g. B951A).
2) A user name (e.g. the user's last name)
3) The upper and lower memory limits to be used by the program, for each processor.
4) The master processor, (defaults to B for S/C 32, A for S/C 31).
5) The management goal for the automatic attempt, (*balanced* or *just under*).
6) The output file/format (SRF or runstream)
7) The comment display status (on or off)
8) Selection of command categories which are allowed to move.

This information must be provided to the management program before any form of management can begin. However once the information has been gathered and the user chooses to begin (either the automatic or the manual management) the information will be written to the configuration file (see 15. Configuration File). This is done so that subsequent runs [with the same SRF] will only require the user to input the User ID and the SRF version, and then all of the other information will be retrieved from the file.

Table I displays a complete list of all of the valid keys for the Auto screen. Instructions for each input are discussed in greater detail in the following sections.

7.2.1.1 User Name

The user should always begin the program by supplying a name to be used in the log file. This name could be a first name, last name, or any name which would uniquely identify the user.

To enter the user name, use the following steps:

1) Press the <U> key
2) Type the user name and press <Enter>

Note that the user name will not be retained in the configuration file (it must be re-entered with each MEMMAN run).

7.2.1.2 Input SRF Version

The input SRF version (as opposed to the output SRF discussed in 8. Output SRF) should always be entered as either the first or second Auto screen parameter (second only if the user name is input first). This is because if a configuration file exists for this particular SRF version, then all of the rest of the input information will be retrieved from the file, possibly eliminating any more input for the Auto screen.

To enter the SRF version, use the following steps:

1) Press the <S> key
2) Type the five character SRF version and press <Enter>

Note that the first character of the SRF version should normally be either an "A" or a "B" to identify which spacecraft is being managed, so that the master processor can be inferred. If the first character used is not an "A" or a "B" then check the master processor parameter (see 7.2.1.5 Master Processor).

In addition, the last character should (normally) be a letter ("A" through "Z"). If the fifth character is not a letter, and the user toggles the output file/format to SRF, then the results of the derived output SRF version are unpredictable (see 7.2.1.7 Output File/Format).

Finally, the first four character of the SRF version (the *sequence ID*) are appended to the DataPath variable and the result is used as the *DOS path* for all of the sequence's data files (see the explanation of the DataPath variable on page 34, under 12.2 Syspar File Commands/Variables). When the SRF version is entered, MEMMAN will look for the *DOS path,* using the *sequence ID* appended to the DataPath variable. If the path is not found, the user will be notified. If this occurs, the user should exit the program, check the data path, and re-run. If necessary, the user may have to create a new subdirectory under the directory specified by the DataPath variable (in the Syspar file) with a name of the *sequence ID.*

Of course the SRF version will not be retained in the configuration file, as it is actually used as the *file name* for each sequence's configuration file (the configuration file is the *SRFversion* with the extension ".CON").

7.2.1.3 Category Status

When the program starts, it attempts to open the MEMMAN category file (MEMMAN.CAT) in order to read all of the predefined categories and their respective category masks (see 13. Category File). If (as usual) categories do exist, they will be listed on the right hand side of the Auto screen. The user can then point to a category and alter its movable status (the default comes from the category file). Each category's status can be toggled between moveable and non-moveable, instructing MEMMAN to either allow or not allow commands which match a category to move.

To change a category's status between moveable and non-moveable use the following steps:

1) Use the arrow keys to point to a command category
2) Press the <Space> key to toggle the category status

Note that in the moveable state, the category name will appear in green. In the non-moveable state, it will appear in red. The category settings will be retained in the configuration file, in case the same SRF is reused at a later date.

7.2.1.4 Load Boundaries

The user must always specify the upper and lower memory load boundaries for the sequence before MEMMAN can begin any form of management. This is necessary so that the number of unused words in each processor may be determined (and used) during the management.

To enter the memory load boundaries, use the following steps:

1) Press the <B> key
2) Type the lower boundary for CCSA and (or) press <Enter>
3) Type the upper boundary for CCSA and (or) press <Enter>
4) Type the lower boundary for CCSB and (or) press <Enter>
5) Type the upper boundary for CCSB and (or) press <Enter>

Note that all four boundaries must be *stepped through* each time the user presses the <B> key. However, if the user does not wish to change any of the four boundaries, they can simply press <Enter> at that particular boundary input field, and the old value will be retained. These values will be retained in the configuration file, in case the same SRF is reused at a later date.

If a routine is encountered (while reading the SRF) which has a load address which appears to be in a region of memory outside that which is defined by the upper and lower load boundaries, then the user will be notified with both a warning, and a suggestion that the user inspect the limits for errors[1].

7.2.1.5 Master Processor

The master processor parameter specifies which CCS will be used as the default (initial) load processor. When a CCS is chosen as the master processor, all command movement will be *from* that processor, *to* the opposite. Likewise, all command retraction will be *from* the opposite back *to* the chosen master.

When the user enters an SRF version, if no configuration file was found then the first character of the SRF version is used to determine the default setting for the processor. An "A" as the first character will result in CCSA being the default setting. Likewise a "B" will result in the default being set to CCSB. If the first character of the SRF version is neither an "A" or a "B" then the default setting of the master processor is unpredictable.

To toggle the master processor setting, use the following single step:

1) Press the <M> key

Notice that each time the key is pressed, the setting toggles between CCSA and CCSB. The master processor setting will be retained in the configuration file, in case the same SRF is reused at a later date.

7.2.1.6 Automatic Goal

This parameter, which is only relevant if automatic management is to be attempted, is used to tell MEMMAN what goal should be sought after when attempting to automatically manage the sequence memory (when should the automatic attempt stop).

This parameter has two settings; *balanced* and *just under*. The *balanced* option will instruct MEMMAN to attempt to manage such that an equal number of CCS words are left available in both processors. The *just under* option will instruct MEMMAN to move commands from the master processor to the slave, only until the instance when the master processor is no longer overflowing. This would then (nominally) leave the maximum number of free CCS words available in the slave processor.

[1] Unlike SEQTRAN, the memory limits must include (in the range they define) any routines such as CCSTIM, which are loaded in the SRF being used. For example, if CCSTIM is to be loaded at the bottom of memory in processor A, then the lower limit for processor A should be set at the load address of CCSTIM. Here, the routine will be contained in the range defined by the upper and lower limits. This requirement frees the user from having to calculate the routine cost later.

To toggle the automatic goal parameter setting, use the following single step:

1) Press the <A> key

Notice that each time the key is pressed, the setting toggles between "Balanced" and "Just Under". In one particular case, it is possible that this parameter may possibly be modified during the automatic management attempt. This would occur when the automatic goal has been toggled to the *just under* setting, the automatic goal was reached. At the "Where now?" prompt, the user can choose to continue managing and *balance* the memory. In only this case, the parameter would be (automatically) toggled from *just under* to *balanced* (see also 7.2.2.2 Where Now?). The automatic goal setting will be retained in the configuration file, in case the same SRF is reused at a later date.

7.2.1.7 Output File/Format

The toggling of the output file/format parameter serves two purposes (as implied by the name). It not only selects the output format (either SRF or runstream) but it simultaneously (and automatically) determines the output file name. As the user toggles back and forth between the two options, the output file name will be displayed, followed by the format in parenthesis.

The output file name will be determined as follows. If the output file/format setting is toggled to "SRF", then the output file will be an SRF, and its file name will be almost the same as the input SRF file name but the fifth character will be "bumped up" one letter. For example, if the input SRF file name was B005A.SRF (SRF version B005A) and the user toggled the output file/format to "SRF" then MEMMAN would (derive) display the output file name as B005B.SRF, followed by *SRF* in parenthesis. See also 8. Output SRF.

If the output file/format setting is toggled to "Runstream", then the output file will be a runstream, and its file name will simply be the input SRF version with the extension ".CFG". For example, if the input SRF file name was B005A.SRF (SRF version B005A) and the user toggled the output file/format to "Runstream" then MEMMAN would (derive) display the output file name as B005A.CFG, followed by *Runstream* in parenthesis. See also 9. Output Runstream.

To toggle the output file/format parameter setting use the following single step:

1) Press the <O> key

The output file/format setting will be retained in the configuration file, in case the same SRF is reused at a later date.

7.2.1.8 Comment Status

The comment parameter setting controls whether or not comments in the input SRF are to be ignored. This setting can be toggled between either "On" or "Off". If toggled to "On", any comments appearing in the input SRF will appear in the manual management screen. If toggled to "Off", all comments in the input SRF will be ignored (they will not appear in the manual management screen).

To toggle the comment status setting use the following single step:

1) Press the <C> key

The comment status setting will be retained in the configuration file, in case the same SRF is reused at a later date.

7.2.2 Automatic Attempt

Automatic memory management is attempted by the system after the user presses the <G> key from the Auto screen (see Table I). When the user presses the <G> key, MEMMAN immediately proceeds to read the input SRF from the appropriate directory, parsing each command and storing it in memory (if needed).

Basically once the automatic management has started it does not require any input from the user until it has completed (see 7.2.2.3 Status Prompt below). There are however two instances where the user will need to provide some input for the program. In the first case, the automatic attempt might result in an endless series of moves and retractions of the same group of commands. If this occurs the user will have to interrupt the iteration. In the second case, MEMMAN will prompt the user for optional further instructions when the automatic goal was *just under* and that goal has apparently been achieved. The following two sections (7.2.2.1 Terminate an Endless Loop and 7.2.2.2 Where Now?) will discuss these two unusual cases in greater detail. Then 7.2.2.3 Status Prompt (below) will discuss the normal termination of the automatic attempt.

7.2.2.1 Terminate an Endless Loop

On a very rare occasion, the CCS memory may be so full that the automatic management attempt is left working with very small margins of CCS words. When this occurs it is possible that MEMMAN may enter a continuous loop where it moves a certain series of commands, retracts the same series of commands (in a different order) and then moves them all back again. For example, it may move a cyclic call, a PROG command and then an SPROG command. Next, after calculating the new word count, it may notice that it has too many words in the slave. MEMMAN may then retract the same cyclic definition, the same PROG and then the same SPROG (retracted in a different order). Next, after calculating the new word count, it may notice that it has too many words in the master, and so forth (in a continuous loop). This situation is *almost* impossible to prevent.

In fact, there is usually no obvious evidence that MEMMAN has entered such a loop. Typically the way in which a user discovers the problem is by simply watching the status messages go by as the automatic attempt moves commands, retracts commands, and re-calculates the word count. This will occur under normal circumstances, except that there should be no repeating patterns of commands moved and retracted. If a pattern is detected (same commands keep getting moved and retracted) then the loop must be interrupted.

To interrupt the automatic management attempt, press any key on the keyboard. After the key is pressed, a prompt window will pop-up, asking the user if they wish to interrupt the management attempt. If the user presses the <N> key (for No) the attempt will continue. If the user presses the <Y> key (for Yes) then the attempt will end after getting a final word count. After that final word count, the user will be advised of the resulting CCS memory allocation (see 7.2.2.3 Status Prompt).

7.2.2.2 Where Now?

As discussed in section 7.2.1.6 Automatic Goal, the user can select an automatic goal which is *balanced* or *just under*. In the case of *balanced* memory, the goal is either achieved or it isn't. However, the *just under* case is slightly different.

If the *just under* goal is not achieved, the user is simply presented with the results of the attempt, and prompted to either re-attempt the automatic attempt or to accept and continue (see 7.2.2.3 Status Prompt). However if the *just under* goal is achieved, the user has several options available to them. First of all, they can simply accept the management as it is, with the just under word count. Secondly, they can chose to resume automatic management with the automatic goal changed to *balanced*. Thirdly, they can input a final number of words to move from the master to the slave (must be less than the free memory in the slave).

If the user selects the second option, the automatic goal parameter will actually be changed to *balanced*, and the automatic attempt will continue where it left off. As a result, when the user returns to the Auto screen they will see this effect (the automatic goal setting will be toggled to Balanced). The use of this option is fairly rare, but none the less it is available.

If the user selects the third option, they will be prompted to input a number of CCS words between zero and the number of free words in the secondary. MEMMAN will then continue (after the just under goal was reached) and it will move the specified number of CCS words from the primary to the secondary. For example, the user may desire exactly ten free CCS words in the master processor, no more, and no less. To accomplish this, they could chose the *just under* automatic goal, and allow automatic management to get them to the *just under* state. When the goal was achieved, they could chose to move ten words to the secondary. This would normally result in exactly ten free CCS words in the master processor.

7.2.2.3 Status Prompt

Under normal circumstances, after the automatic attempt has completed the user would be presented with a special prompt called the *status prompt*. This prompt will display the results of the automatic attempt, and then prompt the user to either accept the results or re-attempt the automatic management.

There are actually two situations where the prompt will appear. The first is when an attempt to balance the unused memory has succeeded (automatic management with an automatic goal of *balanced*). In this case, the user is simply presented with the final word count, and asked to "press any key" to continue on to the manual management. The second case is where either an attempt at automatic management with an automatic goal of *balanced* or just under has failed. In this case the user is presented with the final results, and then asked if they wish to re-attempt the automatic management, or accept those [unsuccessful] results and continue on to the manual management.

7.2.2.4 Bypass Automatic Management

After all of the needed information has been gathered at the Auto screen, the user may desire to completely bypass the automatic management attempt, and proceed directly to the Manual screen. This could be for various reasons, including the situation where a force file satisfactorily manages the sequence without any help. In order to bypass the automatic management attempt, use the following steps.

1) Press the <F3> key to bypass the automatic management attempt.
2) Press <Enter> to confirm the choice.

7.3 Using Manual Management

Steps taken in the manual management system can be very simple. For instance, they could be as simple as just choosing to quit the program and generate the output. On the other hand, the steps could be somewhat

involved. For instance, they could involve moving several items, printing an annotated SRF, re-attempting the automatic management, and then choosing to quit the program and generate the output.

7.3.1 Manual Screen

Unlike the Auto screen which only serves as an input screen in preparation for the actual automatic management attempt, the Manual screen provides the user with a very detailed look at the sequence and the command distribution.

The screen provides the user with a line by line view of the SRF, with each record marked and color coded to signify any particular attributes related to memory management. Each line (record) typically has a character placed in a column near the right side of the screen which identifies whether a command is presently in CCSA, CCSB or both. At the top of the screen (the *header*) the user will find various management status information.

7.3.1.1 Line Number

The Manual screen *header* provides the user with a line number, based on numbering which begins with the first *valid* command in the SRF. The first *valid* command is the first command following the SRF header information, and any cyclic definitions.

Table II: Valid Manual Screen Keys

Key(s)	Description
<F1>	Help screen
<F2>	Select/Unselect
<F3>	Go to the Auto screen
<F4>	Print annotated SRF
<F5>	Go to a specific line
<F6>	Toggle CHKPNT SET 0 option
<F7>	Move command(s)
<F8>	Retract command(s)
<F9>	Re-calculate word count
<F10>	Exit and generate output
<Alt><S>	Character search
<Alt><R>	Repeat previous search
<Tab>	View command cost
<Esc>	Abandon MEMMAN
Up Arrow	Move up (back) a line
Dn Arrow	Move down (forward) a line
<PgUp>	Move up (back) a page
<PgDn>	Move down (forward) a page
<Home>	Move to start of SRF
<End>	Move to end of SRF

7.3.1.2 Line Color Significance

Each line from the SRF is displayed on the Manual screen in a particular foreground and background color combination. This color combination is intended to provide some intuitive information about a command's characteristics.

Every command which appears in red, is not moveable. Every command which appears in green, is moveable. In addition, any commands shown with a grey background will be assembled in the overlap region of the CCS memory. All other commands will be assembled in the current region of memory.

Incidently, if a command is marked moveable, it only means that there is nothing which "physically" constrains the command movement such as a command parameter which specifies the processor. It is left up to the operator to decide whether a command which is marked moveable (green) should really be moved (e.g. does it make sense, does it violate unwritten load strategies?).

7.3.1.3 Processor ID Column

On the right side hand of the Manual screen header are two column titles marked [A] and [B]. All commands that appear in the SRF and will be assembled (not comments, etc.) have a character which appears under one of those column titles, denoting the processor where the command will be assembled. Normally this character is either an A or a B, providing sort of redundant information (e.g. an A in the [A] column or a B in the [B] column).

This character however is also used to display information about any CCS overhead costs that are detected when the word count is calculated. Such overhead costs would include new time event regions or hours mode switches (caused by an excessive time gap). When such overhead occurs, the command associated with (causing) the overhead will have a special character printed in the processor column, rather an A or a B. For example, if a command in CCSA begins a new time event region, then as is shown in Figure 1 (at 90-120/12:41:00.945) a paragraph marker (¶) will be placed in the A column (to represent the start of a new time event region) rather than an A (likewise for the B processor, or both processors). If (following a checkpoint) a command in CCSA does not occur sooner than approximately 36 hours after the last command in the same processor, then SEQTRAN will assemble some overhead to switch to hours mode, wait for an hour, back to seconds mode, and wait for a certain amount of seconds. In this case, as is shown in Figure 1 at 90-115/12:40:00.945, a double arrow character (↕) will be placed in the A column (to represent the hours switch) rather than an A (likewise for the B processor, or both processors).

7.3.1.4 Moved Command Markers

As commands or groups of commands are moved (see 7.3.2.1 Move a Single Command and 7.3.2.3 Move a Group of Commands) they are re-displayed in red (non-moveable) and markers are placed next to them on the screen in order to remind the user that they were moved manually. These markers are placed immediately to the right of the right-most processor ID column (the B processor column), and appear as brackets which include every command moved manually. For example, Figure 1 shows both a group of commands moved to CCSB and an individual command moved to CCSB (the salve processor in this example).

7.3.1.5 Present Word Count

The Manual screen header always displays the most recent CCS word count. This count is obtained either as a result of the automatic management, or as a result of a user request to re-calculate (see 7.3.2.7 Update Word

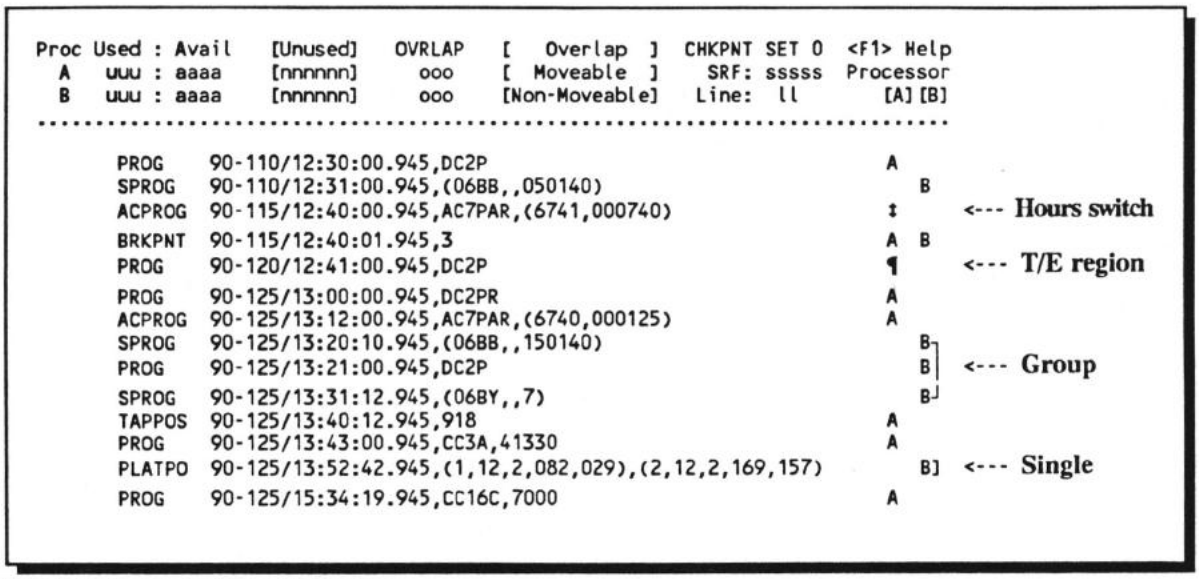

Figure 1: Sample Manual Screen

Count). The displayed word count is actually broken down into three separate components; the total used memory (including both current and overlap), the available memory and the amount of memory used by the overlap region.

Figure 1 shows where in the header each of the word count components appears. The total amount of memory used (including both current and overlap) is displayed under the title *Used*, shown in Figure 1 as *uuu*. The total amount of memory available (based on the load boundaries entered at the Auto screen) is displayed under the title *Avail*, shown in Figure 1 as *aaaa*. Similarly the total amount of unused memory (based on the load boundaries entered at the Auto screen) is displayed under the title *[Unused]*, shown in Figure 1 as *nnnnnn*. Finally, the size of the overlap region for each CCS is displayed under the title *OVRLAP*, shown in Figure 1 as *ooo*.

Notice that the difference between the *available* and *used* CCS words determines the number of *unused* words. The number of *overlap* words however can't be algebraically associated with any of the other values on the screen. It is a special component of the total *used* memory, but there are other components of the current region which are not displayed.

7.3.1.6 CHKPNT SET 0 Status

Near the top right-hand side of the Manual screen header shown in Figure 1, is the string CHKPNT SET 0. This string is displayed any time when the option [by the same name] is turned on. If the option is turned off, then this area of the header will simply be blank (unless the user is selecting commands to move as discussed in 7.3.1.7 Select Status). See also 7.3.2.12 Toggle CHKPNT SET 0 for more information about turning the CHKPNT SET 0 option on and off.

7.3.1.7 Select Status

In the same location where the CHKPNT SET 0 status is displayed, the string SELECT will be flashed whenever the user is selecting commands to move. As soon as the user has either completed the move or turned the selection off again, any text that was behind the flashing status message will be restored (such as the CHKPNT SET 0 status message). For more information about selecting commands, see 7.3.2.3 Move a Group of Commands.

7.3.2 Manual Attempt

The manual attempt at memory management is a method of moving commands between each processor which requires the user to select commands to be moved and retracted. As stated earlier, events can only be retracted if they have been moved, (see 2. Introduction). This rule then applies to the Manual screen in that only events which have been manually moved can be manually retracted.

Although there are many features available to the user while at the Manual screen, most of the features are provided as support for the two main functions of the manual management system; manual command movement and retraction. While at the Manual screen, there are options for moving commands:

1) Move a single command
2) Move a block of commands

Any events which are selected to move manually, are always stored in the configuration file (*SRFversion*.CON) for future re-runs of the same SRF. Because the instructions [placed in the configuration file] to manually move commands are tied directly to specific commands, MEMMAN clears all instructions for manually moved commands whenever it notices that the user is using a newer version of the SRF (newer than the one where the commands were manually moved originally). The user is notified whenever this is done.

The following sections will provide the user with instructions for using all of the available features at the Manual screen. For a more general explanation of manual management see 4.2 Manual Management.

7.3.2.1 Move a Single Command

In order to move a command, the command must be colored green - moveable, and it must be in the master processor (any command in the slave will always be colored red - unmovable). In order to move a single command from the master processor to the slave, use the following steps.

1) Using the cursor control keys, move the command pointer to the desired command.
2) Press the <F7> key to move the command to the slave processor.
3) Press the <F9> key to re-calculate the CCS word count.

7.3.2.2 Move a Cyclic

Using the same steps as above (7.3.2.1 Move a Single Command) the user can move a cyclic definition and all of its calls to the slave processor. To move a cyclic, simply point to one of the calls for that cyclic (in step 1 above).

The manual movement of cyclics does however require the use of caution. MEMMAN will allow you to move a parallel cyclic, if the particular call you are pointing to is not parallel. The user should study a cyclic and all of its calls closely before choosing to move it.

7.3.2.3 Move a Group of Commands

To move a group of commands the user must first select the desired group using the *select* key, and then instruct MEMMAN to move the entire group. The commands must all be colored green - moveable, and they must all be in the master processor (any command in the slave will always be colored red - unmovable). In addition, the group of commands can only include "true" SEQTRAN macro calls (not comments, etc...). In addition, cyclics can not be moved in a group (see 7.3.2.2 Move a Cyclic). If the user is selecting the commands and attempts to select a non-moveable command (or a comment, etc...) then they will be notified that this is illegal. In order to move a single command from the master processor to the slave, use the following steps.

1) Press the <F2> key to turn on the selection option (the string SELECT should start flashing in the screen header).
2) Press the down arrow key until the entire [desired] group is selected. Note that the down arrow is the only cursor movement allowed while selecting commands (can not select backwards).
3) Press the <F7> key to move the entire group of commands to the slave processor.
4) Press the <F9> key to re-calculate the CCS word count.

If during command selection the user decides that they no longer wish to move continue (with the command selection) they can simply press the <F2> key a second time to turn off the selection option.

7.3.2.4 Retract a Single Command

In order to retract a command manually (or group of commands for that matter) it must presently be in the slave processor, and it must be there because it was moved there manually. In other words, to retract a command it must have a *moved command marker* appearing next to the processor ID column for that command (see 7.3.1.4 Moved Command Markers).

In addition, it is <u>not</u> possible to retract a single command from within a group of previously manually moved commands. In this case, the entire group of commands must be retracted and the desired commands must then be moved once again (see 7.3.2.6 Retract a Group of Commands). To retract a single command, use the following steps.

1) Using the cursor control keys, move the command pointer to the desired command.
2) Press the <F8> key to retract the command back to the master processor.
3) Press the <F9> key to re-calculate the CCS word count.

7.3.2.5 Retract a Cyclic

Using the same steps as above (7.3.2.4 Retract a Single Command) the user can retract a cyclic definition and all of its calls back to the master processor. To retract a cyclic, simply point to one of the calls for that cyclic (in step 1 above).

Once again, MEMMAN does not check cyclic movement or retraction very closely, so the user should be confident that what they are attempting to do is legal.

7.3.2.6 Retract a Group of Commands

In order to retract a group of commands manually the entire group must presently be in the slave processor, and they must all be there because they were moved there manually. In other words, to retract a group of commands the group must have *moved command markers* appearing next to their processor ID columns for that group of commands (see 7.3.1.4 Moved Command Markers). To retract a group of commands, use the following steps.

1) Using the cursor control keys, move the command pointer to any one of the commands within the desired group of commands.
2) Press the <F8> key to retract the entire group of commands back to the master processor.
3) Press the <F9> key to re-calculate the CCS word count.

If the user desires to retract a single command from within a group of commands that are tagged to be moved together, then they must retract the entire group, and then move only the desired commands.

7.3.2.7 Update Word Count

Whenever commands are moved or retracted manually, the user should update the word count to see the results of the command movement. *Note that the word count values in the screen header are only updated when the user chooses to update them, therefore it is up to the user to keep them up to date.* In order to update (re-calculate) the CCS word count, use the following single step.

1) Press the <F9> to re-calculate the word count.

MEMMAN will take several seconds to look through the entire SRF (with management), and then it will update the word count totals in the Manual screen header.

7.3.2.8 Print Annotated SRF

In order to provide the user with a more complete (continuous) picture of the CCS command distribution, MEMMAN provides the user with the ability to print out the entire SRF, annotated with all of the information available normally on the Manual screen. This printout will include such information as each command's processor, command overhead cost markers (for a new time event region or an hours switch), and manually moved command markers (single commands or groups). In order to print an annotated SRF, use the following steps.

1) Check to see that the printer is on-line (and otherwise ready).
2) Press the <F4> key to begin printing the annotated SRF.

7.3.2.9 String Search

Often times the user may wish to check a certain command (or string of characters) to see something about it. Rather than manually stepping through the entire SRF, looking for the character string, the user can use the search feature. Using this feature, the user can search forward in the SRF for any string of characters

(such as a command name, a time tag, a parameter, etc...). In order to search for a string of characters, use the following steps.

1) Press <Alt><S> (both keys together) to pop-up the search input window.
2) Type the string to be searched for and press <Enter>.

If the string is not unique (in the SRF) it is possible that the search may find the wrong occurrence of the string. If this is the case, the search can be repeated (several times if needed) until the correct occurrence is found. To repeat a search for a string, use the following single step.

1) Press <Alt><R> (both keys together) to repeat a search for a string.

Note that because the *repeat search* looks for the last string entered with the <Alt><S> keys, an initial <Alt><S> search must have been initiated prior to using the repeat search function.

7.3.2.10 View Command Cost

Because each command's CCS word cost is not continually displayed on the screen with the command, a method for viewing a command cost is provided to assist the user in making a decision about command movement (or retraction). This feature provides the user with only the command's inherent cost; it does not include any costs associated with assembly overhead (such as new time event regions starts, etc...). To view the CCS word cost for a command, use the following steps.

1) Using the cursor control keys, move the command pointer to the desired command.
2) Press the <Tab> key to view the command's CCS word cost.

7.3.2.11 Go To a Line

Besides searching for a command (see 7.3.2.9 String Search), the user can proceed directly to a specific command in the manual system, if they know the line number of that command. This is particularly useful if the user has been working with a certain group of commands, but needs to temporarily look somewhere else in the SRF. In this case, they can look at the present line number in the header (see 7.3.1.1 Line Number), go wherever they want in the SRF, and then return to that same line number. To go to a specific line, use the following steps.

1) Press the <F5> key. The old line number (in the header) will be erased, and the cursor will wait there for the new line number.
2) Type the new line number and press <Enter>.

7.3.2.12 Toggle CHKPNT SET 0

A special trick often used by the SEQTRAN/COMSIM analyst to save CCS words is to override the SEQTRAN symbol called CHKPNT with a value of zero, immediately after a BRKPNT macro (which normally follows a CHKPNT macro). This symbol is automatically set (by SEQTRAN) to a value of one whenever a CHKPNT macro is encountered in the SRF. As a result, future occurrences of an excessive time gap will result in SEQTRAN's assembly of an hours switch at a cost of four CCS words, as opposed to a new time event region at a cost of only three CCS words.

It is usually preferable (and possible) to have a new time event region started for the time gap, as it costs one less word. If the word count is very high and there are a few checkpoint regions in the sequence, then the difference between the cost of an hours switch and the cost of a new time event region could mean the difference between being able to fit the sequence into the available CCS memory and not being able to.

MEMMAN has the ability to automatically insert the needed command to reset the CHKPNT symbol after every occurrence of a BRKPNT macro. As a default, MEMMAN will not do this. If however the user sees a need for this trick, they can toggle MEMMAN's CHKPNT SET 0 option on and off, and view the separate word costs with and without the option turned on. If the option is turned on when the output is finally generated, the commands to implement the option will be inserted into the output file in the appropriate places. To toggle the CHKPNT SET 0 option, use the following steps.

1) Press the <F6> key to toggle the CHKPNT SET 0 option (see 7.3.1.6 CHKPNT SET 0 Status).
2) Press the <F9> key to re-calculate the CCS word count.

7.3.2.13 Re-attempt Automatic Management

While at the Manual screen, the user may desire to return to the Auto screen in order to re-attempt the automatic management. This could be for various reasons, including an unsatisfactory prior attempt. In order to return to the Auto screen, use the following steps.

1) Press the <F3> key to return to the Auto screen.
2) Press <Enter> to confirm the choice.

7.3.2.14 Quit & Generate Output

When the user is finally satisfied with the results of the management, they can then exit the program and in the process generate the output file specified by the *output file/format* setting (see 7.2.1.7 Output File/Format). In either case, the only way to generate the output is by leaving the program. Once the user chooses to this option, MEMMAN will follow three (or four) separate steps.

1) The runstream will be printed (only printed if the output format is toggled to runstream)
2) The MEMMAN run history will be printed.
3) The output file will be generated.
4) The program will terminate.

The run history printout contains information about the results of the management attempt as well as a description of all of the settings used during the attempt. The sheet should be used to document the management attempt, as well as to re-create the conditions required to repeat similar management results on a later version of the SRF (the same sequence). The run history contains the following information:

1) User name
2) Input SRF version
3) Output file/format
4) Master processor
5) Automatic management goal
6) Memory load boundaries
7) Total memory available

8) Used & unused word count totals
9) Category settings
10) CHKPNT SET 0 status
11) A list of all items moved automatically
12) A list of all items moved manually
13) A list of all items moved via the force file

In order to quit the program and generate the output, use the following steps.

1) Check to see that the printer is on-line (and otherwise ready).
2) Press the <F10> key to generate the output and leave the program.
3) Press <Enter> to confirm the choice.

8. Output SRF

If the output file/format setting is toggled to SRF when the user chooses to quit the program and generate output, then the output file will be another SRF with the last letter of the version ID bumped up by one character. This SRF will be identical to the input SRF except for the following differences:

1) The output SRF will have CONFIG macro calls placed throughout the file in order to accomplish the desired memory management.
2) The *PREP field in the SRF header will be modified to reflect the fact that the output file was created by MEMMAN.

Due to the addition of the normally many CONFIG macro calls, the output SRF will typically be much larger than the input SRF.

Once the output SRF has been written, it would typically then be uploaded to the Univac, and used as the input to a SEQTRAN run for that particular sequence. The result of the SEQTRAN run should match (*very* closely if not perfectly) the word count predicted by MEMMAN. If this is not the case, and the word counts are off by several words (three, four or more) then the user should closely examine their work, including (but not limited to) the MEMMAN run to be sure it appears to be correct. A typical error is to supply incorrect load boundaries for MEMMAN.

9. Output Runstream

If the output file/format setting is toggled to Runstream when the user chooses to quit the program and generate output, then the output file will be an editor runstream designed for the JPL editor (the @ED editor) on the Univac 1100.

The runstream, which is a series of editor instructions, is uploaded to the Univac and then executed from within the original SRF. The resulting SRF will be identical (with respect to the macro commands) to the one which would be created by MEMMAN using the SRF option of the output file/format (see 8. Output SRF). A complete sample runstream can bee seen on page 45 (A.6 Sample Output Runstream), however the individual parts of the runstream are described below. The sample runstream is for a sequence where CCSB is the master, and commands to be "moved" are moved to CCSA.

```
DP 2IB          CONFIG  0.0,2,0
DP 3I          CONFIG  0.0,0,2
```

This first two lines of the sample runstream (shown above) define two editor procedures. The first will insert a CONFIG macro prior to the one pointed to by the editor. This CONFIG macro instructs SEQTRAN to begin assembling in CCSA (would be CCSB if CCSA were the master). The second will insert a CONFIG macro after the one pointed to by the editor. This CONFIG macro instructs SEQTRAN to begin assembling in CCSB (would be CCSA if CCSB were the master).

```
0
F,8 CCSTIM  90-150/16:30:01.000
IB          CONFIG  0.0,0,2
```

The next three lines of the sample runstream (above) serve to establish the master (default) processor for the SEQTRAN assembly. The line with the 0 simply takes the editor to the top of the file. The next line will look for the string "CCSTIM 90-150/16:30:01.000" (in column 8) and will insert a CONFIG to CCSB immediately prior to it (the master processor in this example). The result of this is that all commands will now be assembled in CCSB unless specified otherwise (by a subsequent CONFIG macro, a command parameter, etc...).

```
0
F,8 CCSTIM  90-150/16:30:01.000
DP 1F,36 CC3A
MC 8*(1,2,3)
```

In the sample runstream, the next four lines (shown above) will take the first eight occurrences of SRF records with "CC3A" in column 36, and add CONFIG macros before and after them. These CONFIG macros will then cause SEQTRAN to assemble the command in the slave processor.

```
0
F,8 CCSTIM  90-150/16:30:01.000
DP 1F,36 (06BB
MC 8*(1,2,3)
MC 1*(1)
MC 7*(1,2,3)
```

The above 6 lines will take the first eight occurrences of SRF records with "(06BB" in column 36 and add CONFIG macros before and after them, skip the next occurrence (one), and finally add CONFIG macros before and after the next seven occurrences. This (in a sense) moves almost all of the first 16 occurrences of "(06BB" to the slave process, all 16 except for the ninth occurrence. This will occur in a runstream when a command(s) needs to be skipped, while the rest of the same type can be moved (a command may need to be skipped for various reasons, such as when it already has a CONFIG - to the slave - just prior to it).

10. Force File

The force file is an ASCII text file which contains certain commands in a specified format, which will force certain events into either processor. A record in the SRF is "forced" to either processor when that record has a valid match with any mask specified with a Force statement in the force file. In addition, each Force command can be followed by an unlimited number of Except commands to handle exceptions to that particular Force statement, in that SRF.

If a command is forced to the slave processor, then MEMMAN will added the necessary commands to tell SEQTRAN to assemble the command in the slave processor. If a command is forced to the master processor, then MEMMAN will simply mark it as un-moveable.

The force file can be created by the user for each new SRF version, or an old one can be re-used (see 10.3 Force File Maintenance and 10.4 Re-using the Force File). The force file is read (and processed) automatically by MEMMAN when the SRF is read into memory.

10.1 Force File Format

The force file is comprised of the following elements (some optional):

$$FORCE	Must appear as the first record in the file.
Force Command	Force instruction(s) (up to 30 Force commands).
Exception	Optional exception to the preceding Force command (unlimited number of Except commands for each Force command).
$$EOF	Must appear as the last record in the file.

The force file must not contain any blank lines. In addition, there are no means available to add comments to the file. See A.4 Sample Force File for an example of a force file.

10.2 Force File Commands

The above commands appear as follows in the force file, or MEMMAN will notify the user with a fatal error message.

The Force command (shown above) takes on the following form:

FORCE *"mask":processor*

Force command parameter descriptions:

"mask"	The force mask can be up to 40 characters long, surrounded by quotes. Notice the single space between the Force command and the mask, and no space between the mask and the colon.
processor	Can be either A or B (any other characters will result in an error). Notice there are no spaces between the colon and the processor specification.

The Except command takes on the following form:

EXCEPT *"mask"*

Except command parameter descriptions:

"mask"	The exception mask can be up to 40 characters long, surrounded by quotes. Notice the single space between the Except command and the mask.

10.3 Force File Maintenance

The force file must be named *SRFversion*.FRC, and placed in the appropriate sequence directory (see the DataPath command under 12.2 Syspar File Commands/Variables). This is so that each force file will be associated with an SRF version, (e.g. B951P.FRC for SRF B951P).

To create or edit the force file, simply use any ASCII text editor, or a word processor which is capable of save a file in *DOS text format*. For example, use the DOS EDLIN editor, the TOPDOS editor, the Norton Commander editor or Wordstar in non-document mode.

Once a force file has been created for a sequence, then each time a new SRF is released for that sequence (a new SRF version) the same force file can be re-used. In fact, a force file for one sequence can actually be re-used for another sequence (see 10.4 Re-using the Force File).

10.4 Re-using the Force File

Once a force file has been created (for any sequence) it can be re-used for virtually any other sequence. In fact, this is usually what is done for each new sequence where a force file is needed (the old force file is used as a template for the new one). To re-use a force file, use the following steps.

1) Copy the old force file to a new one.
2) Make any necessary edits to the file for the new sequence.

For example, if this is the first run of a sequence (e.g. B952, SRF version B952A) and there is a desire to use a force file from a previous sequence (e.g. B951, SRF version B951E) use the following steps.

1) Type `MD C:\SEQ\B952` (if the directory does not already exist)
2) Type `CD C:\SEQ\B952`
3) Type `COPY C:\SEQ\B951\B951E.FRC B952A.FRC`
4) Edit the file B952A.FRC as needed

If this is not the first run of a sequence, but a new version of a previous SRF (e.g. B952, SRF version B952C), and there is a desire to use a force file from the previous SRF version (e.g. SRF version B952A) use the following steps.

1) Type `CD C:\SEQ\B952`
2) Type `COPY B951A.FRC B952C.FRC`
3) Edit the file B952C.FRC as needed

11. Seqtran Macro Description File (SMDF)

In order to be able to successfully manage the sequence command placement, MEMMAN must have access to an accurate description of the command characteristics of every command that it might possibly encounter in the SRF. In this way, the program can make decisions based upon rules that the SEQTRAN/COMSIM analyst would normally use.

For instance, MEMMAN must know how many CCS memory words a particular command costs (a cost which includes the initial time word). In addition, it must know if a command belongs in a certain processor. If so, then it must know if there is a parameter associated with the command which specifies which processor it is to be assembled in. MEMMAN must decide whether or not a command can be moved automatically. Some commands can be moved [to the slave CCS] freely, while others might be constrained to remain in one particular processor. Other commands may fall under both categories, under different circumstances.

To solve this often complicated problem, MEMMAN retains two sources of SEQTRAN macro descriptions. First it uses internal descriptions. These descriptions are hard-coded in the MEMMAN program in a procedure called ProcessCommand (see MEMMAN Software Description Document). This method of command description is reserved for the more complicated command descriptions, commands which require complicated processing or run-time decision making.

The second source of command descriptions is the SEQTRAN Macro Description File (SMDF). This file (MEMMAN.SMD) is an external ASCII input file which is read at the start of MEMMAN, and contains descriptions of SEQTRAN macros that generally do not require complicated processing or run-time decision making. As new commands appear in the MSS, they can be added to the SMDF as long as they do not require any special decision making on the part of MEMMAN. Otherwise they must be added to the MEMMAN program internally.

It is important that the user does not confuse this file with the force file. The SMDF can not be used to force a command into a specific processor. This file is only used to describe a command's characteristics to MEMMAN, such as a command's word cost, and which processor the command will be assembled in by SEQTRAN (inherently, without any memory management).

11.1 SMDF Format

The SMD file is comprised of the following elements:

$$SMDF	Must appear as the first record in the file.
;	Comment(s) (unlimited number of comments).
SMD(s)	Macro description(s), see below (unlimited number of macro descriptions)
$$EOF	Must appear as the last record in the file.

The SMD file must not contain any blank lines. The existing SMD file also contains format and syntax instructions in the form of comments. See A.3 Sample SMD File for an example of the SMD file.

11.2 SMDF Commands

The macro description commands must appear as follows in the SMD file, or MEMMAN will notify the user with a fatal error message.

The SMD command (shown above) takes on the following form:

"macro_name", word_cost, processor, auto_move, manual_move

SMD command parameter descriptions:

"macro_name"

This parameter is a string of characters enclosed in double quote signs ("). This string reflects the actual SEQTRAN macro name, found in column 8 of the SRF. This is a required parameter.

e.g. `"BR01GO"`

word_cost

This parameter reflects the CCS word cost for this command description. The amount should be for one processor only, with the ability to specify the processor as "BOTH" using the *processor* parameter. If you know that the macro is to be assembled in both processors (parallel), and that the cost for the two CCS processors is different, then this command can not be added to the SMDF, it must be specially added to the ProcessCommand procedure MAN_PCMD.PAS (see the MEMMAN Software Description Document and the MEMMAN Software Maintenance Document).

e.g. `"BR01GO",2`

Use 0 if there is no word cost.

processor

The CCS processor where MEMMAN will cost this command. Note that this parameter does not control where SEQTRAN actually assembles the command, but only where MEMMAN "thinks" SEQTRAN will assemble it. If the parameter is EITHER, MEMMAN will cost the command in the current processor (based on CONFIGs in the SRF, or the master processor if there haven't been any CONFIGs.

Valid PROCESSOR values:

`PROCA`	CCS Processor A
`PROCB`	CCS Processor B
`BOTH`	Parallel
`MASTER`	Master CCS (Chosen from a MEMMAN input screen)
`SLAVE`	Slave CCS (Opposite processor)
`UNDEFINED`	Neither processor (command has no MEMMAN impact)
`EITHER`	Follow current assembly path (processor)

e.g. `"BR01GO",2,PROCB`

auto_move

This is a boolean parameter which governs whether this command will be allowed to move during the automatic memory management. The parameter can be either TRUE (allowed to move automatically) or FALSE (not allowed). However, if the *processor* parameter is BOTH or SLAVE, then *auto_move* will automatically be FALSE, regardless of the value specified in the file.

e.g. `"BR01GO",2,PROCB,FALSE`

manual_move

This parameter is similar to the *auto_move* parameter, except that it controls movement during the manual management of a sequence. The same constraints regarding the *processor* parameter with the *auto_move* parameter apply here.

e.g. `"BR01GO",2,PROCB,FALSE,FALSE`

11.3 SMDF Maintenance

The SMD file must be named *MEMMAN.SMD*, and placed in the directory which contains the program executable (see 3.2 Directory Structure).

Every time new SEQTRAN macros are added to the MSS, they will either have to be added to the MAN_PCMD.PAS subroutine of the MEMMAN source code (see MEMMAN Software Description Document and MEMMAN Software Maintenance Document) or to the SMDF. The preferable method, if appropriate, is to add the new macros to the SMDF as this does not require a re-delivery of MEMMAN. For very simple macro commands which are always the same cost and always in the same processor, the SMDF method should be able to be used. More complex macros will have to be added to the MEMMAN source code and MEMMAN will have to be re-delivered.

To edit the SMD file, simply use any ASCII text editor, or a word processor which is capable of save a file in *DOS text format*. For example, use the DOS EDLIN editor, the TOPDOS editor, the Norton Commander editor or Wordstar in non-document mode. Any updates made to the SMD file will be automatically reflected in future MEMMAN runs.

12. Syspar File

The System Parameter File (Syspar) is an ASCII text file which contains several modifiable MEMMAN parameters. The file contains variable names, followed by their default settings. Each of these settings can be modified by the user, in order to change one of the defaults. The Syspar variables are listed under 12.2 Syspar File Commands/Variables.

12.1 Syspar File Format

The Syspar file is comprised of the following elements:

$$SYSPAR	Must appear as the first record in the file.
variable_name "setting"	Syspar variable and setting.
$$EOF	Must appear as the last record in the file.

The Syspar file must not contain any blank lines, and there is no existing method to add comments to the file. See A.1 Sample Syspar File for an example of the Syspar file.

12.2 Syspar File Commands/Variables

The following text describes all valid values of *variable_name* that are presently available to the user, and lists all of the valid *setting* values.

`DATAPATH` *"pathname"*

Specifies the DOS path to use while looking for the data files (SRFs, force files, configuration files, etc...). As each sequence should have its own directory, the *pathname* should specify the path to the parent directory of all of the sequence directories.

e.g. `DATAPATH "C:\SEQ\"`

The above example would assume that each sequence will have its own subdirectory underneath the \SEQ directory named for the sequence ID, (notice the back-slash character after the directory name). For instance, with the above example, use of the SRF version B951A would cause MEMMAN to search the path C:\SEQ\B951 for all of the data file for that SRF.

`SYSPATH` *"pathname"*

Specifies the DOS path to use while looking for the MEMMAN system files (the category file, the SMDF, the log file).

e.g. `SYSPATH "C:\SEQ-TOOL\MEMMAN\"`

The above example will cause MEMMAN to look in the MEMMAN subdirectory of the SEQ-TOOL directory for all of the MEMMAN system files, (notice the back-slash character after the directory name).

LOWBOUNDA *"boundary"*
LOWBOUNDB *"boundary"*
UPBOUNDA *"boundary"*
UPBOUNDB *"boundary"*

The absolute highest/lowest memory location to be used during memory management. These values are used to constraint check the limits entered at the Auto screen.

OUTPUTFMT *"format"*

Specifies the output file/format default at the Auto screen. Valid values for *format* are SRF and RUNSTREAM.

e.g. OUTPUTFMT "RUNSTREAM"

SHOWCOMMENTS *"status"*

Specifies the default state for the Show Comments parameter at the Auto screen. Valid values for *status* are TRUE and FALSE.

e.g. SHOWCOMMENTS "FALSE"

12.3 Syspar File Maintenance

The Syspar file must be named *MEMMAN.SYS*, and placed in the directory which contains the program executable (see 3.2 Directory Structure).

To edit the Syspar file, simply use any ASCII text editor, or a word processor which is capable of save a file in *DOS text format*. For example, use the DOS EDLIN editor, the TOPDOS editor, the Norton Commander editor or Wordstar in non-document mode. Any updates made to the Syspar file will be automatically reflected in future MEMMAN runs.

13. Category File

The category file is an ASCII text file which is used to group commands together into categories which can be marked as moveable or non-moveable. Each category has a name, and some associated mask strings which describe all SRF records which should belong to that category.

When using MEMMAN, a complete list of all of the categories appears on the Auto screen so that the user may toggle any of them between a moveable and non-moveable state. Any SRF records which match a mask associated with a category name will follow the state of that category. It is possible to have more than one category who's masks match an SRF record. In this case, the user will be notified of the conflict and informed of the state used.

Like the SMD file, the category file is not to be confused with the force file. The categories can only control whether a type (category) of command(s) can be moved, not what processor they belong in.

13.1 Category File Format

The category file is comprised of the following elements:

$$CATEGORY	Must appear as the first record in the file.
;	Comment(s) (unlimited number of comments).
Category(s)	Category name(s) and default state(s) (up to 15 categories)
Mask(s)	Category mask(s) for preceding category (unlimited number of masks per category)
$$EOF	Must appear as the last record in the file.

The category file must not contain any blank lines. See A.2 Sample Category File for an example of a category file.

13.2 Category Commands

The Category command (shown above) takes on the following form:

CATEGORY *"category_name":default_state*

Category command parameter descriptions:

category_name — This specifies the name of the category as it will appear on the Auto screen. The category name can be up to 15 characters long. Note the single space between the Category command and the left quote of the category name. Also note that there are no spaces between the category name and the colon.

default_state — This specifies the default state for the category (named above). Valid values for the *default_state* are TRUE or FALSE. Notice that there are no spaces between the default state and the colon.

The Mask command (shown above) takes on the following form:

MASK *"mask"*

Mask command parameter descriptions:

mask	This specifies a string of characters to be used as a mask for matching SRF records as they are read from the file. If an SRF record contains the specified mask, then the command will belong to the category specified by the preceding Category command (in the category file).

13.3 Category File Maintenance

The category file must be named *MEMMAN.CAT*, and placed in the directory which contains the program executable (see 3.2 Directory Structure).

The category file will only need to be updated when the user determines that they would like a new level of control over the movement of a certain type (category) of commands.

To edit the category file, simply use any ASCII text editor, or a word processor which is capable of save a file in *DOS text format*. For example, use the DOS EDLIN editor, the TOPDOS editor, the Norton Commander editor or Wordstar in non-document mode. Any updates made to the category file will be automatically reflected in future MEMMAN runs.

14. Log File

The log file is an ASCII text *output* file which is updated by MEMMAN every time the user chooses to quit and generate output (generate an SRF or a runstream). The file contains an ongoing history of each instance where MEMMAN was used to generate an output file. The log file can be viewed by the user at any time in order to see who has recently used MEMMAN to generate output, and what the output was they generated.

14.1 Log File Format

The log file begins with a header record which will automatically be created for the user if the file is missing (deleted). Otherwise, every entry (record) written to the log file contains the following information:

1) The input SRF filename
2) The output filename/format (the file extension indicates the format; .SRF is an SRF output while .CFG is a runstream output.
3) The date of the output file generation
4) The time of the output file generation
5) The user who generated the output (user name entered at the Auto screen)

See A.5 Sample Log File for an example of a log file.

14.2 Log File Maintenance

The log file is named *MEMMAN.LOG*, and appears in the directory which contains the program executable (see 3.2 Directory Structure). Because the log file is an ASCII text file, it can be viewed at any time. In fact, it can even be edited at any time, although there should never be a need to do so.

If there is ever such a time when the log file becomes large and cumbersome, or even at predefined intervals (such as end of year, etc...), the log file can be renamed to something else and MEMMAN will automatically create a new one.

For example, at the DOS prompt, the user could type:

```
RENAME MEMMAN.LOG 1990.LOG
```

The next time MEMMAN was used to generate some output, it would then re-create a new log file, header and all.

15. Configuration File

The configuration files are named as *SRFversion*.CON (i.e. B951E.CON) and are located in the individual sequence directories, under the main data directory specified by the DataPath variable in the Syspar file (see 12. Syspar File). As implied above, there is a separate configuration file created by MEMMAN for each individual SRF.

The configuration file is a binary file which contains all of the MEMMAN parameters and values from one particular run that are needed to repeat a subsequent run for the same input SRF. Specifically, the file contains all of the parameters discussed in 7.2 Auto Screen, along with a complete list of any events tagged to be moved manually (see 7.3 Using Manual Management). With all of this information retained, the user should only have to enter the information once for each sequence.

As stated above, there is a separate configuration file created by MEMMAN for each individual SRF. However, configuration files can be re-used for different SRFs with no harm being done. This is in fact common practice (see 15.1 Re-using the Configuration File).

Because the file is a binary file, the user can not view or edit the file with a standard text editor or word processor. The only way to view the contents of the file is to start the MEMMAN program, and at the Auto screen input the SRF version corresponding to the configuration file. This should automatically open the file, and display the contents. All of the parameters at the Auto screen will be updated with the values from the file, if it exists.

15.1 Re-using the Configuration File

As mentioned above, the same configuration file can be re-used for subsequent versions of the same SRF. To re-use the configuration file, the user simply needs to copy the file to a new file with the new SRF version as the name. For example, if the first SRF version for the sequence B951 to be run through MEMMAN was B951F, and a new version such as B951H was now available, use the following steps.

1) Type `CD \SEQ\B951`
2) Type `COPY B951F.CON B951H.CON`

Then, simply start the MEMMAN program, and enter B951H as the SRF version, an the new configuration file will automatically be processed and its data will be displayed in the remaining spaces at the Auto screen.

When this copying of the files is done however, any manually tagged events in the original file will be disabled when the new (copy) file is used. This is so that [MEMMAN's] internal pointers for manually tagged events will not get confused because the SRF is now different. The user will be notified if the manually tagged events are ever disabled in this fashion.