Synertek

Systems

Corporation

Resident Assembler Editor

RAE-1
REFERENCE MANUAL

RAE-1
REFERENCE
MANUAL

SCC PUB MAN-A-260027-C

Fourth Printing: February 1981

The contents of this manual were placed in the public domain when SCC ceased SYM-1 operations.

Synertek Systems Corporation
150 South Wolfe Road * Sunnyvale, CA 94068 * (408) 988 - 5600 TWX: 910-338-0135

TABLE OF CONTENTS

LIST OF TABLES

APPENDICES

SECTION 1.0

INTRODUCTION TO RAE

1.0 <u>INTRODUCTION</u>

This 6502 resident relocating macro assembler and text editor reside simultaneously in 8K bytes of ROM memory. Sufficient memory must be provided for a Source (text) file and Label file (symbol table). Approximately 2K is sufficient memory for the source file for small programs or larger programs if assembled from tape. A good rule of thumb is one byte of memory for the label file for each byte of object code. If an executable object code file is to be stored in memory during assembly, sufficient memory must be provided for it also. On cold start entry, the RAE will set the file boundaries as follows:

- Source file = 0200-0BFC **1000-7FFC in GoertzWorks! Ram Model**
- Object file = Not Specified **0200-0FFF in GoertsWorks! Ram Model**
- Label file = 0C00-0EFC **D000-DFFC in GoertzWorks! Ram Model**
- Relocatable object buffer = 0F00 **N/A in GoertzWorks! Ram Model**

The label file and text file that RAE generates are position independent and may be located practically anywhere in RAM memory. The object code file location is dependent on the beginning of assembly (.BA) and the move code (.MC) pseudo ops.

RAE was designed such that records in the label file and text file are variable in length and directly dependent on the number of characters to be stored. This results in efficient utilization of memory.

Some major features of RAE are:

- Macro and conditional assembly support.
- Labels up to 10 characters in length.
- Auto line numbering for ease of text entry.
- Creates either executable code in memory or relocatable object code on tape.
- Manuscript feature for composing letters and other text.
- Loading and storing of text on tape.
- Supports up to two tape decks, terminal with keyboard, and printer.
- String search and replace capability, plus other powerful editing commands.
- Upper and lower case accepted.

Throughout this document, output generated by RAE is underlined if necessary to distinguish it from user input.

Initial entry (cold start) to RAE is at address B000. Warm start is at address B003. If the break command (≥BR) is executed, one may return to the address following the break. Initial entry provides the following default parameters:

FOR TED

* Format - set
* Manuscript - clear
* Auto line numbering - 0 or clear
* Text file - clear
* Tape units - off
* Hardcopy - clear

FOR ASSEMBLER

* Assumes assembling from memory (otherwise use .CT).
* Does not store object code in memory (otherwise use .OS).
* Begins assembly at $0200 (otherwise use .BA).
* Output listing clear (otherwise use .LS or >ASSEMBLE LIST).
* Stops assembly on errors (otherwise use .CE).
* Stores object code beginning at $0200 unless a .BA or .MC is encountered and if .OS is present.
* Generates relocatable addresses.
* Macro object code is not output (otherwise use .ES).

The RAE is designed to operate with a cassette record unit and a play unit. A single record/play unit may be used but one will not be able to create relocatable object files when assembling from tape.

When inputting to RAE the following control codes are useful:

CONTROL H (hex 08)
Rubout or Delete (hex 7F) Backspaces over previous character. More than one of these may be entered to delete a number of characters. A backslash is echoed if rubout is depressed.

CONTROL X (hex 18) Deletes the entire line.

Break Key Halts outputting, and waits for input of appropriate control code.

For a more detailed list see **Section 8**.

SECTION 2.0

GETTING STARTED WITH RAE

2.1 GENERAL

An assembler is a program which allows the user to compose and enter programs at the machine language level in a form that is much more convenient than actual machine code. The assembler accepts mnemonic names for individual instructions, allows symbolic names to be assigned to memory locations and data, provides for address arithmetic in terms of symbolic names, and certain other features, depending on the sophistication of the assembler in question.

The Synertek System's Resident Assembler/Editor (RAE) is a full features assembler. Other major features include: macros with nesting capability, conditional assembly, creation of relocatable object code supported by a relocating loader, string search/replace and line editing, automatic control of two I/O tape units, and assemble directly from tape.

It is commonly thought that the primary feature offered by an assembler is that of writing machine instructions in a more convenient form. However, this is only one aspect of the advantage of an assembler, and perhaps not even the most significant. The use of symbolic names to represent numbers makes variables of what most likely would have been considered constants. The very presence of symbols bestows a generality and flexibility to a program which otherwise might have seemed quite rigid. This encourages the programmer to abstract the immediate problem and perhaps develop a more adaptable program. Also, since the actual calculation or assignment of a value to a symbol can be deferred, the development of logically separate modules can proceed freely. Programs so organized become much more readable and manageable, both in their maintenance and amenability to revision.

The purpose of an assembler is to translate a program written in assembly language into machine language. Machine language refers to that representation of instructions which are immediately interpretable by the machine being considered. For all intents and purposes, the machine language of the 6502 consists of hexadecimal opcodes and data. An assembly language is a symbolic representation of machine language instructions; e.g., LDA # is used to represent the instruction **A9**, **L**oa**D** the **A**ccumulator with the value following the **#** sign (immediate).

The program written in assembly language is called the source code, the machine language program produced by the assembler is called the object code.

With or without an assembler, it should be realized that programs are usually written in assembly language. The assembler simply saves us the tedious and error-prone task of translating our program into machine code.

The assembler accomplishes the conversion of the source code to machine code in two passes; that is, the source program is scanned twice. During the first pass all symbols and their associated values are collected into a label file (also called a symbol table). During the second pass the assembler converts the program to machine language (also called object code), using the definitions collected in the first pass.

One important feature that all assemblers share is that of assembler directives, or pseudo ops. These are special orders to the assembler itself about the way it is to deal with the source program, or for the definition and manipulation of symbols and allocation of storage. The distinction between operations (machine instructions) and pseudo operations is similar to that between a manuscript to be typed and the author's marginal notes to the typist. For example, directives are used to tell the assembler to set aside 100 memory locations to be used for an array, or to tell it where the object code is to be stored in memory.

2.2 HARDWARE PREPARATION

2.2.1 RAE ROM ADDRESSING

Your RAE is contained in one (RAE-1) or two (RAE-1/2) ROMs. These ROMs are designed to run under the **SYM-1 SUPERMON** monitor or the **MDT-1000** system monitor.

Before you install the ROMs into your system, refer to your system reference manual to locate or "strap" the desired ROM socket at the correct memory address as shown below.

RAE-1

* Single 8K byte chip (2364)
* P/N 02-0053A
* Chip select pin 20
* Address: 1 Prom
  B000-BFFF
  E000-EFFF

RAE-1/2

* Two 4K byte chips (2332's)
* P/N's 02-0023A and 02-0024A
* Chip select pin 20
* Address: 2 Prom's
  B000-BFFF (02-0023A)
  E000-EFFF (02-0024A)

For detailed discussion of jumper configurations for SYM-1, see **Section 10.0, paragraph 4.**

2.2.2 AUDIO CASSETTE I/O

RAE is designed to work with a dual audio cassette system using Synertek Systems' high speed recording format. Cassette unit #0 is designated the record unit and unit #1 is designated the play unit. A single cassette player may be used for most operations except where the user wishes to assemble source from tape and store object code back onto tape.

Refer to your particular system's reference manual for details on I/O addressing, remote control, and adjustments. The following is a summary of each of these:

ADDRESSING (IN, OUT, REMOTE CONTROL)

SYSTEM	AUDIO IN	AUDIO OUT	REMOTE CONTROL # 0 RECORD	REMOTE CONTROL # 1 PLAY
SYM-1	A000-BIT 6	A400 (SYM ref. manual)	A00C-CB2	A000-BIT 7
MDTI000	9600-BIT 7	9600-BIT 6	9703-CB2	9701-CA2

AUDIO CASSETTE RECORDER ADJUSTMENTS

TONE High or treble.

VOLUME* 2V peak-to-peak or saturation (max volume) from recorder (suggested for most recorders).

TAPE Data tape or high quality, low noise audio tape. Short lengths (30 mm or less) works best

SUGGESTED RECORDER Sanyo M2544A or equivalent

*** Each recorder type will require a volume adjustment in order to obtain maximum reliability, 2vp-p or saturation (max volume) works well on most recorders.**

2.3 STEP-BY-STEP EXAMPLE

To access the Resident Assembler/Editor (RAE), power up your system and log-on to your terminal, then type "G B000"; this is the cold start entry point. (The warm start entry point is B003).

RAE will respond with:

```
RAE V1 .0
COPYRIGHT 1979 SYNERTEK SYSTEMS CORP.


0200-0BFC  0C00-0EFC  0F00
0200  0C00
```

```
NOTE:  TEXT FILE 0200-0BFC
       LABEL FILE 0C00-0EFC
       RELOCATABLE OBJECT BUFFER 0F00
       CURRENT END OF TEXT BUFFER 0200
       CURRENT END OF LABEL BUFFER 0C00
```

If you inadvertently stop RAE's log-on printout before the first prompt character (>) is displayed, RAE will double echo each character typed and also ignore any commands. To exit this mode, type CONTROL O

The ">" is the prompt symbol from RAE, indicating it is ready to accept commands. In the following procedures the ">" is not shown. Only the most commonly used commands and the major features of RAE will be discussed in the following section. Several examples will be used to illustrate their use and action.

NOTE

ALL COMMANDS MUST BE ENDED WITH A CARRIAGE RETURN. If you make a typing error, enter a CONTROL H or a RUBOUT to delete the last character. Several CONTROL H's can be entered to remove more than one character. A CONTROL X will eliminate the entire line. Processing can be suspended by pressing the BREAK key and resumed with a CONTROL Q.

We will begin by entering a program segment which fills page 3 (0300-03FF) of memory with zeros. Each line of text must be preceded by a line number, so that RAE can order them properly, as well as process any changes we may wish to make as we go along.

Type in the following lines exactly as they appear, immediately following the prompt symbol:

```
10 LDX #0
20 TXA
30LOOP STA $300,X
40 DEX
50 BNE LOOP
```

Note that the instruction mnemonics and addressing mode formats are those defined and described in the SY 6500 Programming Manual (MNA-2).

Now type in:

```
PRINT
```

RAE will respond with:

```
0010               LDX #0
0020               TXA
0030  LOOP         STA $300,X
0040               DEX
0050               BNE LOOP
//
```

Notice that RAE automatically lines up the label, instruction, and operand fields, and that if the first character is a blank the label field is skipped. To examine lines 20 through 40 only, type in:

```
PRINT 20 40
```

RAE will reply with:

```
0020               TXA
0030  LOOP         STA $300,X
0040               DEX
//
```

Notice that the line numbers in the PRINT command are separated by blanks, not commas. This is the convention used by RAE in specifying all command parameters. Let us now try to assemble our program. Type in:

```
ASSEMBLE LIST
```

RAE will print:

```
                0050            BNE LOOP
!07 AT LINE 0050/44
```

This is an error message, telling us that the .EN (end of program) pseudo op is missing. It is required to indicate to RAE the end of the source program. Let us put it in and try again.

Type in the following:

```
60 .EN
ASSEMBLE LIST
```

RAE will respond with:

```
0200- A2 00     0010                LDX #0
0202- BA        0020                TXA
0203- 9D 00 03  0030 LOOP           STA $300,X
0206- CA        0040                DEX
0207- D0 FA     0050                BNE LOOP
                0060                .EN

LABEL FILE: [ / = EXTERNAL ]

LOOP=0203
//0000 0209,0209
```

This time assembly of our program was successful. The listing produced shows us the object code as well as the source code. The leftmost column contains the address of the first byte of each instruction. As can be seen, the default beginning address is $200. The .BA (Begin Assembly) pseudo op is used when we wish RAE to assemble beginning at some other address, say $500.

Type in:

```
5 .BA $500
ASSEMBLE LIST
```

RAE will respond with:

```
                0005                .BA $500
0500- A2 00     0010                LDX #0
0502- 8A        0020                TXA
0503- 9D 00 03  0030 LOOP           STA $300,X
0506- CA        0040                DEX
0507- D0 FA     0050                BNE LOOP
                0060                .EN

LABEL FILE: [ / = EXTERNAL ]

LOOP=0503
//0000,0509,0509
```

Up to this point everything RAE has done has been "on paper". If we want the object code generated by RAE to be actually stored in memory at the address specified, we need to include the .OS (object store) pseudo op. Type in the following:

```
6 .OS
ASSEMBLE
```

Notice that the LIST option was omitted from the ASSEMBLE command. This time RAE will simply print:

```
//0000,0509,0509
```

Let us exit RAE momentarily to examine some memory locations. To exit to the system monitor type:

```
BREAK (or CONTROL C)
```

The system monitor will print:

```
B0AC,0
.
```

Now type:

```
V 0500
```

The system monitor will reply:

```
0500 A2 00 8A 9D 00 03 CA D0,66
 0366
.
```

This is the object code of our program, stored by RAE. To continue where we left off type either:

```
G B003 or G
```

B003 is the warm start entry point to RAE. If the cold entry point were used our text file would be lost.

RAE will print:

```
0200-0BFC  0C00-0EFC  0F00
0246  0C06
```

In order to execute our program without exiting RAE, we need to make the last executable instruction an RTS so that control will be returned to RAE.

Type in:

```
55 RTS
ASSEMBLE
```

RAE will print:

```
//0000,050A,050A
```

Now enter:

```
RUN $500
```

RAE will come back with the prompt sign, "≥". Let us exit RAE again to verify that the program ran.

Type:

```
BREAK
```

System monitor will print:

```
B0AC,0
.
```

Now type:

```
V 0300
```

System monitor will print:

```
0300 00 00 00 00 00 00 00 00,00
 0000
.
```

Apparently our program worked as intended. Get back into RAE. Recall that the warm entry point is B003.

Let us begin a new example. This time we will change the starting boundary of the text file to allow room for object code to be stored in memory at the RAE default origin. Type the following:

```
SET $300
```

RAE will respond with:

```
0300-0BFC  0C00-0EFC  0F00
024E  0C06
```

We must now clear the text file because its starting boundary has been changed. Failure to do so is catastrophic. To do this type:

```
CLEAR
```

If you now type PRINT, RAE will simply print //, which is the end-of-text indicator. The following code is for a pseudo-random number generator. To make the entering of the text easier, first type in:

```
AUTO 10
```

This command enables the automatic line numbering option. The 10 will be used as the line number increment. AUTO goes into effect after a line is referenced.

Type in:

```
100RND SEC
```

RAE will now respond with:

```
0110>
```

which is the next line number. Now enter the following lines after each line number, remembering to leave a space if there is no label:

```
 LDA TABLE+1
 ADC TABLE+4
 ADC TABLE+5
 STA TABLE
 LDX #4
MOVE LDA TABLE,X
 STA TABLE+1,X
 DEX
 BPL MOVE
 RTS
```

To exit AUTO, type:

```
//
```

We must be sure to include the .EN (end of program) pseudo op, so enter:

```
999 .EN
```

RAE will respond with:

```
1009>
```

This is because the AUTO mode is still enabled. Type // to exit AUTO, then, to turn off the AUTO option, type:

```
AUTO 0
```

Let's now try to assemble our code. Enter:

```
ASSEMBLE LIST
```

RAE will reply:

```
0200- 38         0100 RND          SEC
                 0110              LDA TABLE+1
!08 AT LINE 0110/00
```

This error message tells us that there is an undefined label in line 110. The problem is, of course, that RAE has no way of knowing what the symbol TABLE represents. TABLE is meant to be the name of an array of six elements. The pseudo op .DS (**D**efine **S**torage) is used to tell RAE to set aside a specified number of memory locations.

Type in:

```
90TABLE .DS 6
ASSEMBLE LIST
```

RAE will print:

```
0200-            0090 TABLE        .DS 6
0206- 38         0100 RND          SEC
0207- AD 01 02   0110              LDA TABLE+1
020A- 6D 04 02   0120              ADC TABLE+4
020D- 6D 05 02   0130              ADC TABLE+5
0210- 8D 00 02   0140              STA TABLE
0213- A2 04      0150              LDX #4
0215- BD 00 02   0160 MOVE         LDA TABLE,X
0218- 9D 01 02   0170              STA TABLE+1,X
```

```
021B- CA        0180             DEX
021C- 10 F7     0190             BPL MOVE
021E- 60        0200             RTS
                0999             .EN

LABEL FILE:  [ / = EXTERNAL ]

TABLE=0200             RND=0206              MOVE=0215

//0000,021F,021F
```

Notice that TABLE has been assigned the address 200 (hex), and that the first byte of code is at location 206. Thus locations 200 - 205 have been reserved; TABLE+1 is memory location 201, TABLE+2 is 202, etc.

To test this routine we will add some code which will call RND as a subroutine and print out the pseudo random numbers generated. To aid us in the output we will call on two subroutines in the system monitor: OUTBYT and CRLF. OUTBYT outputs the contents of the accumulator as two hex digits, and CRLF outputs a carriage return and a line feed.

In order to use them, we must tell RAE where they are located. This is done using the .DE (**D**efine **E**xternal) pseudo op, which tells RAE that the addresses specified are external to our program. Type in the following lines:

```
40 .OS
500OUTBYT .DE $82FA
60CRLF .DE $834D
300START LDY #8
310NEXT JSR RND
320 LDA TABLE
330 JSR OUTBYT
340 JSR CRLF
350 DEY
360 BNE NEXT
370 RTS
```

Assemble, and check that your output looks exactly as follows:

```
                0040             .OS
                0050 OUTBYT      .DE $82FA
                0060 CRLF        .DE $834D
0200-           0090 TABLE       .DS 6
0206- 38        0100 RND         SEC
0207- AD 01 02  0110             LDA TABLE+1
020A- 6D 04 02  0120             ADC TABLE+4
020D- 6D 05 02  0130             ADC TABLE+5
0210- BD 00 02  0140             STA TABLE
0213- A2 04     0150             LDX #4
0215- BD 00 02  0160 MOVE        LDA TABLE,X
0218- 9D 01 02  0170             STA TABLE+1,X
021B- CA        0180             DEX
021C- 10 F7     0190             BPL MOVE
021E- 60        0200             RTS
021F- A0 08     0300 START       LDY #8
0221- 20 06 02  0310 NEXT        JSR RND
0224- AD 00 02  0320             LDA TABLE
```

```
0227- 20 FA 82  0330              JSR OUTBYT
022A- 20 4D 63  0340              JSR CRLF
022D- 88        0350              DEY
022E- D0 F1     0360              BNE NEXT
0230- 60        0370              RTS
                0999              .EN

LABEL FILE:  [ / = EXTERNAL ]

/OUTBYT=82FA    /CRLF=834D        TABLE=0200
RND=0206        MOVE=0215         START=021F
NEXT=0221

//0000,0231 ,0231
```

Since the .OS (Object Store) pseudo op was present, the object code was stored in memory, so we can now run the program. Type in:

```
RUN START
```

The output you get will depend on what values happened to be in memory at locations 200-205. With 20 (hex) in each location, the output will be:

```
61
A2
E3
25
A7
AC
33
3C
```

It is common practice to place all subroutines after the main body of the program. Thus, in the above example, we would like to place lines 100 through 200 after line 370. The MOVE command allows this to be done very easily.

Type in:

```
MOVE 370 100 200
```

To see what has been done, enter:

```
PRINT 360 999
```

RAE will print:

```
0360            BNE NEXT
0370            RTS
0370  RND       SEC
0370            LDA TABLE+1
0370            ADC TABLE+4
0370            ADC TABLE+5
0370            STA TABLE
0370            LDX #4
0370  MOVE      LDA TABLE,X
0370            STA TABLE+1,X
0370            DEX
```

```
0370          BPL MOVE
0370          RTS
0999          .EN
```

If you type PRINT 100 200 you will see that lines 100 through 200 no longer exist. Since all the moved lines have been given the same number, we would like to renumber the text file. That is the purpose of the NUMBER command.

Type in:

```
NUMBER 90 10
```

The 90 specifies the line to begin the renumbering, and the 10 specifies the increment to use. If you now PRINT out the entire file you will see that each line number is again unique.

NOTE

The following example will utilize the audio cassette storage unit. If your cassette unit is not connected or adjusted refer to your system reference manual.

The next example is a routine which multiplies the contents of memory location MLTPLR times the contents of location MLTPND. The product will be two bytes long; the high part will be in the accumulator and the low part in location RESLO. OUTBYT will again be used to output the result. Type in:

```
CLEAR
AUTO 10
100MULT LDA #0
```

RAE will respond with:

```
0110>
```

Now enter the following lines after each line number:

```
 STA RESLO
 LDX #8
LOOP LSR MLTPLR
 BCC NOADD
 CLC
 ADC MLTPND
NOADD LSR A
 ROR RESLO
 DEX
 BNE LOOP
;LINE 210
 JSR OUTBYT
 LDA RESLO
 JSR OUTBYT
 RTS
 .EN
 //
AUTO 0
```

Line 210 is a comment line. A comment line begins with a semicolon and may contain any characters after that, as comment lines are ignored by RAE. In this case, it is used to separate the multiplication routine from the output section for better readability. Comments may also appear on any text line by simply separating the text and comment by at least one space. As an example, retype lines 100 and 110 as follows:

```
100MULT LDA #0  ZERO RESULT HI
110 STA RESLO  ZERO RESULT LOW
```

Before this routine will assemble we need to define the symbols OUTBYT, RESLO, MLTPLR and MLTPND. Type in:

```
40OUTBYT .DE $82FA
50RESLO .DS 1
60MLTPLR .BY 2
70MLTPND .BY 3
```

The .BY (store bytes of data) pseudo op directs RAE to store the following value in the next memory location. MLTPLR and MLTPND will thus contain the numbers 2 and 3, respectively.

Finally, we need to add the .OS (object store) pseudo op, and let us also put in the .LS (print source listing on pass 2) pseudo op which enables the list option on assembly. Enter:

```
10 .OS
20 .LS
ASSEMBLE
```

RAE will print:

```
                 0010 .OS
                 0020 .LS
                 0040 OUTBYT      .DE $82FA
0200-            0050 RESLO       .DS 1
0201- 02         0060 MLTFLR      .BY 2
0202- 03         0070 MLTPND      .BY 3
0203- A9 00      0100 MULT        LDA #0 ZERO RESULT HI
0205- BD 00 02   0110             STA RESLO ZERO RESULT LOW
0208- A2 08      0120             LDX #8
020A- 4E 01 02   0130 LOOP        LSR MLTPLR
020D- 90 04      0140             BCC NOADD
020F- 18         0150             CLC
0210- 6D 02      0160             ADC MLTPND
0213- 4A         0170 NOADD       LSR A
0214- 6E 00 02   0180             ROR RESLO
0217- CA         0190             DEX
0218- D0 F0      0200             BNE LOOP
                 0210 ;LINE 210
021A- 20 FA 82   0220             JSR OUTBYT
0210- AD 00 02   0230             LDA RESLO
0220- 20 FA 82   0240             JSR OUTBYT
0223- 60         0250             RTS
                 0260             .EN
```

```
LABEL FILE: [ / = EXTERNAL ]

/OUTBYT=82FA          RESLO=0200        MLTPLR=0201
MLTPND=0202           MULT=0203         LOOP=020A
NOADD=0213
//0000,0224,0224
```

If your output looks exactly as the above, the program is ready to be run.

Type in:

```
RUN MULT
```

The output will be:

```
0006
```

Now change the values in lines 60 and 70, assemble the new program and run it. For example the product of 4 and 9 is 0024 (hex), and that of 45 and 68 is 0BF4 (hex).

One of the most important and fundamental features of RAE is the ability to read and write to the cassette unit. We will save on tape and then retrieve the current program. Place a blank tape in your recorder, advance tape beyond blank leader and put the recorder in record mode.

Now type:

```
PUT F1
```

After the file has been recorded RAE will return with the prompt. Repeat this procedure twice more to ensure a good recording. We will now read in the text file just recorded. Rewind the tape.

Now put the tape unit in the play mode, and type in:

```
GET F1
```

When the file has been read in successfully, RAE will print:

```
F01  011F  0200-031F
```

If you now type PRINT, you can verify that the file was read in correctly. If an error occurs, retype GET F1 and start the tape again.

Now that you are acquainted with the basic features offered by RAE, you are encouraged to read **Sections 3 and 4** in order to become familiar with the many other commands, pseudo ops, and editing features available to you. By far the most effective, efficient and enjoyable way to do this is to construct examples to try out each feature. Learning by doing will show you exactly how each feature works, and will enable you to utilize the full potential of the Synertek System's Resident Assembler/Editor.

SECTION 3.0

TEXT EDITOR (TED)

3.1 TEXT EDITOR COMMANDS

The TED provides 27 command functions. When entered, a command is not executed until a carriage return is given. Although a command mnemonic such as >PR may be several non-space characters in length, the ASM/TED considers only the first two. For example, >PR, >PRI, >PRINT, and >PRETTY will be interpreted as the print command.

Some commands can be entered with various parameters. For example, >PRINT 10 200 will print out the text in the text file with line numbers between 10 and 200. One must separate the mnemonic and the parameters from one another by at least one space. Do not use commas. For alphabetic parameters, only the first character is considered. For example "FORMAT CLEAR" is the same as “FO C.”

NAME	EXAMPLE	PURPOSE/USE
>ASSEMBLE w x	>AS LI >AS N >AS L 200	Clear the label file and then assemble source in the text file starting at line number x or 0 if x is not entered. If w = LIST then a listing will be generated. If w = NOLIST or not entered then an errors only output will be generated.
>AUTO x	>AU 10 >AU >AUTO 20	Automatic line numbering occurs when an x value not equal to zero is entered. x specifies the increment to be added to each line number. Auto line numbering starts after entering the first line. To prevent auto line numbering from reoccurring enter >AU or >AU 0, after first exiting with //.
>BREAK	>BR >BRK	Break to system monitor (executes BRK instruction). A return to the TED can be performed at the address immediately after the break instruction, has the same effect as CONTROL C.
>CLEAR	>CL	Clear text file and turn off tape units.
>COPY x y z	>CO 110 10 40 >CO 300 100 200	Copy lines y thru z in the text file to just after line number x. The copied lines will all have line numbers equal x. At completion, there will be two copies of this data - one at x and the original at y.

>DELETE x y	>DE 40 >DE 100 301	Delete entries in text file between line numbers x and y inclusive. If only x is entered, only the first occurrences of that line is deleted.
>DUPLICATE Fw	>DUP >DUP F10 >DU F	Duplicate files from tape unit 1 to tape unit 0 until file w. This command starts by reading the next file on tape 1 and if that file is file w or an end of file mark then it stops. If not, the file just read will be written to tape 0 and then tape 1 is read again. This continues until file w or an end of file record is encountered.
>FORMAT w	>FO S >FO C >FO SET >FORMAT S	Format the text file (where w = SET) or clear the format feature (where w = CLEAR). Format set tabulates the text file when outputted. This lines up the various source statement fields. This feature, set or clear, does not require extra memory. Assembly output is dependent on the state of the format feature.
>GET Fx y	>GE >GET F13 100 >GET APPEND >GET F2 A	Get text file with data associated with file number x from tape. The data will be loaded at line number y, or will be appended to end of the text file if the key-word APPEND is entered for y. Defaults are x = 00 and y = 0.
>HARD w x	>HA S 1 >HARD C >HA P	Control output to hard COPY output device (printer). Turn on outputting (w = SET) or turn off (w = CLEAR). The starting page number is x. This command is designed to leave a small margin at top and bottom, and provide a page number heading at the top of each page. It is designed to work with 66 line pages. An entry of >HA PAGE results in the printer advancing to the top of the next page. >HA set will cause output to go through the printer vector in addition to OUTVEC.
>LABELS	>LA >LAB	Print out the label file generated by the previous ASSEMBLE.

>**MANUSCRIPT w**	>MA S >MA C	If w = SET, line numbers are not outputted when executing the >PR command. If w = CLEAR, line numbers are outputted when the >PR command is executed. Assembly output ignores the >MA command. If manuscript is to be generated with RAE, manuscript should be set and format clear (>MA SET, >FO CLEAR). Since the TED considers a blank line a deletion, one must enter a non-printable control character to "trick" the TED into inserting a blank line, e.g., 'TAB' (CONTROL I).
>**MOVE x y z**	>MO 110 10 40 >MO 300 100 200	Move lines y thru z in the text file to just after line number x. The moved lines will all have line numbers equal to x. The original lines y thru z are deleted.
>**n**	>10 >100	Any entry beginning with one or more decimal digits is considered an entry/deletion of text. See **Section 3.4.**
>**nnnn//**	>2000//	Used to exit temporarily from auto line number mode so that commands may be entered. Entry of a line number rather than a command will cause return to auto line number mode.
>**NUMBER x y**	>NU 0 10 >NU 100 10	Renumber the text file starting at line x in text file and expanding by constant y. For example to renumber the entire text file by 10, enter >NU 0 10.
>**OFF n**	>OF 0 >OF 1 >OFF	Turn off tape unit n, where n is 0 (record unit), or 1 (play unit). If an n is not entered, 0 is assumed.
>**ON n**	>ON 0 >ON 1 >ON	Turn on tape unit n, where n is 0 (record unit), or 1 (play unit). If an n is not entered, 0 is assumed.
>**OUTPUT Fw**	>OU F >OU F14 >OUT	Create a relocatable object file on tape unit 0 and assign file number w to the recorded data. If w is not entered 00 will be assumed. This command uses the 256 byte relocatable buffer that can be relocated via the >SET command.

>PASS

```
>PA
>PASS
```

Execute the second pass of assembly. Not required if source is all in internal memory and the .CT pseudo op is not encountered.

>PRINT x y

```
>PR
>PRINT 10
>PRINT 100 301
```

Print the text file data between line number x and y on the terminal. If only x is entered, only that line is printed. If no x and y, the entire file is outputted.

>PUT Fw x y

```
>PU F13
>PU F13 200 300
>PUT F
>PUT
```

Put text file between lines x and y inclusive to tape, and assign the recorded data file number w. If w is not entered, 00 will be assumed. If x and y are not entered, the entire text file is recorded. If the letter X is entered as the parameter such as >PU X an end of file mark is recorded.

>RUN label expression

```
>RU START
>RU $1000
>RUN TEST+5
```

Run (execute) a previously assembled program. If a symbolic label is entered, the label file is searched for its value. The called program should contain a JMP warm start (4C03B0) as the last executable instruction.

>SET ts te ls le bs

```
>SE
>SE $1000 $2000 $200 $3FF $400
>SET
```

If no parameters are given, the text file, label file, and relocatable buffer boundaries (addresses indicating text file start, end, label file start, end, and relocatable buffer start) will be output on the first line, then on the second line the output consists of the present end of data in the text file followed with the present end of data in the label file. If parameters are entered, the first two are text file start (ts) and end (te) addresses, then the label file start (ls) and end (le) addresses, and finally the relocatable buffer start address (bs). Parameters may be entered either in decimal form, or if preceded by a $, in hex form.

>**USER**	>US >USR	User defined command. The RAE will transfer control to location $0003. The user routine can re-enter RAE via a JMP warm start (4C03B0).
>**LOAD f x**	>LOAD DUMP 0 >LO RX320c 4	***GoertzWorks! Ram Model only.*** Load file name f from floppy device x. If source file is not clear loaded source will be appended at the end of the current source file. Requires **SYMDOS** to function.
>**ENTER f x**	>ENTER DUMP 0 >EN RX320c 4	***GoertzWorks! Ram Model only.*** Save file name f to floppy device x. Requires **SYMDOS** to function.

Floppy I/O functions are attached to RAE by issuing a call to **RAEENTRY** in **SYMDOS** once RAE is running. Floppy device [0,4], [1,5], [2,6], and [3,7], are the same device except device 4, 5, 6, and 7 perform a write verify when saving.

3.2 EDIT AND FIND COMMANDS

STRING SEARCH AND REPLACE (EDIT) COMMAND

```
>EDIT string or >EDIT n
```

A powerful string search and replace, and line edit capability are provided via the >EDIT command to easily make changes in the text file. Use Form 1 to string search and replace, and Form 2 to edit a particular line.

```
FORM 1           #
                 *
>EDIT tS1tS2t %d ~ x y
```

where:

t is any non numeric terminator, e.g., ".", "/".

S1 is the string to search for.

S2 is the string to replace S1.

d is the "don't care" character. Precede with % character to change the don't care; this character used within S1 indicates which position to ignore for a search "match" condition.

***** indicates to interact with user via subcommands before replacing S1 (see below).

~ (a space character) indicates to alter and print all lines altered.

indicates to alter but provide no printout.

x line number start in text file.

y line number end in text file.

Asterisk * prompted subcommands:

A alter field accordingly.

D delete entire line.

M move to next field - don't alter

S skip this line - don't alter

X exit >ED command

CONTROL F enter form 2

Form 1 Defaults:

```
          d =  %
          x =  0
          y =  9999
          ~ =  (space) print all lines altered
```

For example, to replace all occurrences of the label LOOP with the label START between lines 100 and 600, enter:

```
>EDIT /LOOP/START/ 100 600
```

To simply delete all occurrences of LOOP, enter:

```
>EDIT /LOOP// 100 600
```

Use the * and # as described.

The slash was used in the above examples as the terminator but any non-numeric character may be used.

At the end of the >EDIT operation, the number of occurrences of the string will be output as //xxxx where xxxx is a decimal quantity.

FORM 2

```
>EDIT n
```

where:

n is line number (0-9999) of line to be edited.

Subcommands:

CONTROL F Find user specified character.

CR carriage return. Retain remaining part of line.

CONTROL D Delete any remaining part of line.

CONTROL H Delete a character.

For example, to change LDA to LDY in the following line,

```
LOOP1 LDA #L,CRTBUFFER ;LOAD FROM BUFFER
```

type CONTROL F followed with A, then CONTROL H, then Y, and then terminate line with a carriage return.

The corrected line will be outputted and entered in the text file.

FIND STRING S1 COMMAND

Used to find certain occurrences of a particular string. It's form is:

```
              #
              *
>FIND tS1t %d ~ x y
```

where:

t, S1, %, d, x, y are as defined in the EDIT command, FORM 1.
***, ~** indicates print all lines containing occurrences of S1.
indicates no printout.

At the end of the >FIND operation, the number of occurrences of the string will be output as //xxxx where xxxx is a decimal quantity.

A unique use of this command is to count the number of characters in the text file (excluding line numbers). The form for this is:

```
>FIND /%/#
```

3.3 HOW TO USE EDIT AND FIND

We will show with a simple example, how to use some of the EDIT features of RAE. Other features, such as the use of a "don't care" character in string searching, and the control of the degree of user interaction, are described elsewhere in this manual. FIND is used to search for, but not alter, strings. It is particularly useful in finding cross-references in a source code; its use is like that of the form of EDIT which does not use a line number.

Let the text to be edited be manuscript, rather than source code. SET FORMAT CLEAR, AUTO 10, and enter the manuscript. After entry, print and examine, and make the desired corrections.

For example, let the manuscript read:

"10 Now is the time for all good men"

and let it be corrected to read

"10 Now is the best time for most good women"

The procedure is as follows

```
>pr
0010 Now is the time for all good men
//

>ed 10
Now is the time for all good men
^F>eNow is the best^F>l time for al^F>1l\\\most
  0010  Now is the best time for most good men
>ed /mem/women/*
23  0010  Now is the best time for most good men *a
  0010  Now is the best time for most good women
//0001
>pr
 0010  Now is the best time for most good women
 //

>
```

All underlined characters and symbols are RAE outputs.

For insertions, find the starting point and enter new material, ending with RETURN.

For deletions, find the end of the string, and delete with either DELETE, RUBOUT or CONTROL H, depending on the type of terminal. New material may then be added if desired; if not hit RETURN.

The CONTROL Fe was entered to find the "e" in "The".

The CONTROL Fl was entered twice to find the second "l" in "all".

The "*" was used to permit interaction in case the string being searched for had multiple occurrences, and replacement was to be on a selective basis. The "23" is the count (in hex) to the start of the string /man/ in line 0010. The "a" is user approval to alter; entry of "s" would skip the alteration.

When editing is completed, enter MANUSCRIPT SET, to inhibit line number printing, and print the final copy. The process is less complicated than it would appear from the example, and will soon become almost automatic; the user will see, almost at once, simpler, though less illustrative, means for accomplishing the editing above.

It is good operating procedure to have a backup copy of the material which is being edited on tape, in case of operator errors with the MO, CO, DE, etc. commands.

3.4 ENTRY/DELETION OF TEXT

Source is entered in the text file by entering a line number (0-9999) followed by the text to be entered. The line number string can be one to n digits in length. If the string is greater than 4 digits in length, only the right-most 4 are considered. Text may be entered in any order but will be inserted in the text file in numerical order. This provides for assembling, printing, and recording in numerical order. Any entry consisting of a line number with no text or just spaces results in a deletion of any entry in the text file with the same number. If text is entered and a corresponding line number already exists in the text file, the text with the corresponding number is deleted and the entered text is inserted.

TO DELETE THE ENTIRE FILE, use the >CL command.

TO DELETE A RANGE OF LINES, use the >DE command.

TO EDIT AN EXISTING LINE or lines having similar characteristics, use the >ED command.

TO FIND A STRING, use the >FI command.

TO MOVE OR COPY LINES use the >MO or >CO commands.

TO COPY FROM INPUT TAPE TO OUTPUT TAPE until a specific file, use the >DU command.

The terminal input buffer is 80 characters in length. There are 9 tab points preset at 8 character intervals. Thus, the first tab point is at the 8th column, the second at the 16th column, etc. Entry of TAB or CONTROL I will result in a movement to the next tab point. When inputting, the cursor may not position exactly at the tab point but will position properly when the text file is outputted via the >PR command.

Text may be entered more easily by use of the auto line numbering feature (>AU command). Any >AU x where x does not equal 0 puts the TED in the auto line number mode. To exit from this mode, type >//.

When entering source for the assembler, one need not space over to line up the various fields. Labels are entered immediately after the line number or > when in auto line numbering. Separate each source field with one or more spaces. If the format feature is set (see >FO command), the TED will automatically line up the fields. Note: If a space is entered before the label, the TED will line up the label in the next field. This should result in an assembler error when assembled. If a control I (tab) is entered, a tab to the 8th column is formed. These tabs are preset and can not be changed. Commands, mnemonics, and pseudo ops may be entered as upper case or lower case characters. Assembly labels may also be entered in upper or lower case but a label entered as upper case will be different from the same label entered as lower case.

SECTION 4.0

ASSEMBLER (ASM)

4.1 ASSEMBLER FEATURES

The ASM scans the source program in the text file. This requires at least two passes (or scans). On the first pass, the ASM generates a label file (or symbol table) and outputs any errors that may occur. On the second pass the ASM creates a listing and/or object file using the label file and various other internal labels.

A third pass (via >OU) may be performed in order to generate a relocatable object file of the program in the text file. This file is recorded on tape unit 0 and may be reloaded into the memory using the relocating loader at practically any location.

4.2 SOURCE STATEMENT SYNTAX

Each source statement consists of five fields as described below:

line number   label   mnemonic   operand   comment

Line number

The line number is any number between 0 and 9999. If more than 4 numbers are inputted, only the last 4 digits are recognized.

Label

The first character of a label may be formed from the following characters:

```
@ A thru Z [\]^
```

while the remaining characters which form the label may be constructed from the above set plus the following characters:

```
. / 0 thru 9 : ; < > ?
```

The label is entered immediately after the line number or prompt (>) if in the auto line numbering mode.

Mnemonic or Pseudo Op

Separated from the label by one or more spaces and consists of a standard 6502 mnemonic from Table A or pseudo op from Table B.

Operand

Separated from mnemonic or pseudo op by one or more spaces and may consist of a label expression from Table C and symbols which indicate the desired addressing mode from Table D.

Comment

Separated from operand field by one or more spaces or tabs and is free format. A comment field begins one or more spaces past the mnemonic or pseudo op if the nature of such does not require an operand field. A free format comment may be entered if a semicolon (;) follows the line number or > if in auto line numbering mode.

For converting 6502 assembly language programs written on the System 65 or on MOS Technology Timesharing Cross Assembler, refer to Appendix C.

TABLE A - 6502 MNEMONICS

For a description of each mnemonic, consult the MNA-2 SY6500 Programming Manual.

ADC
AND
ASL
BCC
BCS
BEQ
BIT
BMI
BNE
BPL
BRK
BVC
BVS
CLC
CLD
CLI
CMP
CLV
CPX
CPY
DEC
DEX
DEY
EOR
INC
INX
INY
JMP
JSR
LDA
LDX
LDY
LSR
NOP
ORA
PHA
PHP
PLA
PLP
ROL
ROR
RTI
RTS
SBC
SEC
SED
SEI
STA
STX
STY
TAX
TAY
TSX
TXA
TXS
TYA

TABLE B - PSEUDO OPS

NAME	**EXAMPLE**	**PURPOSE/USE**
.BA expression	.BA $200	Begin assembly at the address calculated from the label expression. This address must be defined on the first pass or an error will result and the assembly will halt.
.BY	.BY 00 'ABCD' 47 69 'Z' $FC %1101	Store bytes of data. Each hex, decimal, or binary byte must be separated by at least one space. An ASCII string may entered by beginning and ending with apostrophes (').
.CE	.CE	Continue assembly if errors other than !07, !04, or !17 occur. All error messages will be printed.
.CT	.CT	Indicates that the source program continues to tape.

label .DE expression	IN .DE INDEV	Assign the address calculated from the expression to the label. Designate as external and put in label file. An error will result if the label is omitted.
label .DI expression	ASCII .DI TABLE	Assign the address calculated from the expression to the label. Designate as internal and put in label file. An error will result if the label is omitted.
.DS expression	.DS 20 .DS $00F0	Define a block of storage. For example, if expression equated to 4, then ASM will skip over 4 bytes. Note: The initial contents of the block of storage are undefined.
.EC	.EC	Suppress output of macro generated object code on source listing. See **Section 4.7.** This is the default condition.
.EJ	.EJ	Eject to top of next page if >HA SET was previously entered.
.EN	.EN	Indicates the end of the source program.
.ES	.ES	Output macro generated object code on source listing. See **Section 4.7.**
.LC	.LC	Clear the list option so that the assembly terminates printing the source listing after the .LC on pass 2.
.LS	.LS	Set the list option so that the assembly begins printing out the source listing after the .LS on pass 2.
!!!label .MD (p1 p2 p3...)		Macro definition. See **Section 4.7.**
.MC expression	.MC $700 .MC CAT .MC ORIGIN+$1000	When storing object code, move code to the address calculated from the expression but assemble in relation to that specified by the .BA pseudo op. An undefined address results in an immediate assembly halt.
.ME	.ME	Macro end. See **Section 4.7.**

.OC	.OC	Clear the object store option so that object code after the .OC is not stored in memory. This is the default option.
.OS	.OS	Set the object store option so that object code after the .OS is stored in memory on pass 2.
.RC	.RC	Provide directive to relocating loader to resolve address information in the object code per relocation requirements but store code at the pre-relocated address. This condition remains in effect until a .RS pseudo op is encountered. The purpose of the .RC op is to provide the capability to store an address at a fixed location (via .SI pseudo op) which links the relocatable object code module to a fixed module.
.RS	.RS	Provide directive to relocating loader to resolve address information in the object code per relocation, and store the code at the proper relocated address. This is the default condition.
.SE expression	.SE BASIC .SE $C000	Store the address calculated from the expression in the next two memory locations. Consider this address as being an external address. Note: If a label is assigned to the .SE, it will be considered as internal.
.SI expression	.SI START .SI TABLE .SI =+4	Store the address calculated from the expression in the next two memory locations. Consider this address as being an internal address.

NOTE

Labels may be entered with any of the pseudo ops, but are mandatory where indicated.

TABLE C - EXPRESSIONS

An expression must not contain embedded spaces and is constructed from the following:

Symbolic Labels:

One to ten characters consisting of the ASCII characters as previously defined.

Constants:

Decimal, hex, or binary values may be entered. If no special symbol precedes the numerals then the RAE assumes decimal (example: 147). If $ precedes then hex is assumed (example: $F3). Only the last four hex digits are used. If % precedes then binary is assumed (example: % 11001). Leading zeros do not have to be entered. All numbers greater than 65,536 are reduced modulo 2^{16}.

Program Counter:

To indicate the current location of the program counter use the symbol = .

Arithmetic Operators:

Used to separate the above label representations:

    + addition, - subtraction

Examples of some valid expressions follow:

```
LDA  #%1101         load immediate 00001101
STA  *TEMP+$01      store at byte following TEMP; Zero page
LDA  $471E36        load from $1E36; 47 is ignored
JMP  LOOP+C-$461
BNE  =+8            branch to current PC plus 8 bytes; current PC is
                    first byte of next instruction
```

One reserved symbol is A, as in ASL A. The letter A followed with a space in the operand field indicates accumulator addressing mode.

ASL A+$00 does not result in accumulator addressing but instead references a memory location.

TABLE D - ADDRESSING MODE FORMAT

Immediate

```
LDA #%1101          binary 00001101, the pound sign (#) indicates
                    immediate addressing
LDA #$F3            hex F3
LDA #F3             load value of label F3
LDA #'A             ASCII A
LDA #H,expression   hi part of the value of the expression
LDA #L,expression   lo part of the value of the expression
```

Absolute

```
LDA expression
```

Zero Page

```
LDA *expression     the asterisk (*) indicates zero page
                    addressing
```

Absolute Indexed

```
LDA expression,X
LDA expression,Y
```

Zero Page Indexed

```
LDA *expression,X
LDX *expression,Y
```

Indexed Indirect

```
LDA (expression,X)
```

Indirect Indexed

```
LDA (expression),Y
```

Indirect

```
JMP (expression)
```

Accumulator

```
ASL A               letter A indicates accumulator addressing mode
```

Implied

```
TAX                 operand field ignored
CLC
```

Relative

```
BEQ expression
```

4.3 LABEL FILE (OR SYMBOL TABLE)

A label file is constructed by the assembler and may be outputted at the end of assembly (if an .LC pseudo op was not encountered) or via the >LA command. The output consists of each label encountered in the assembly and its hex address. A label in the label file which begins with a slash (/) indicates that it was defined as an external label. All others are considered as being internal labels. When a relocatable object file is generated (via >OU command), any instruction which referenced an internal label or a label expression which consisted of at least one internal label will be tagged with special information within the relocatable object file. The relocating loader uses this information to determine if an address needs to be resolved when the program is moved to another part of memory.

Conversely, instructions which referenced an external label or a label expression consisting of all external references will not be altered by the relocating loader.

At the end of the label file the number of errors which occurred in the assembly will be outputted in the following format:

```
//xxxx,yyyy,zzzz
```

where xxxx is the number of errors found in decimal representation, yyyyy is last address in relation to .BA, and zzzz is last address in relation to .MC.

4.4 ASSEMBLING FROM MEMORY

With the source program in the text file area, simply type >AS x. Assembly will begin starting at line number x. If a .CT pseudo is not encountered, both passes will be accomplished automatically. If a .CT pseudo op is encountered, the >PA command would have to be executed to perform the second pass.

4.5 ASSEMBLING FROM TAPE

Source for a large program may be divided into modules, entered into the text file one at a time and recorded (>PU) on tape.

At assembly, the assembler can load and assemble each module until the entire program has been assembled. This would require two passes for a complete assembly. When assembling from tape, the file identification numbers assigned to the modules are ignored. NOTE: SYM users should refer to **Section 10.0, paragraph 4**, before assembling from tape.

Source statements within a module will be assembled in numerical order but the modules will be assembled in the order in which they are encountered. Source statement numbering is restarted for each module. If a line number is specified in the >AS command indicating the start of assembly, it applies for all modules.

The ASM assumes that if an end of file condition is encountered before the .EN pseudo op and a .CT pseudo op had not been encountered, an error is present (!07 AT LINE xxxx).

When assembling from tape, the assembler should encounter a .CT pseudo op before the end of the first module. Two ways to accomplish this are:

1. a) Load the first module via the >GE command.
   b) This module should contain a .CT pseudo op.

or

2. a) Clear the text file via the >CL command.
   b) Enter >9999 .CT. 9999 is entered since one may have requested any assembly beginning with a line number. This insures that the .CT gets executed.

Next ready the play unit and type >AS x. Either way the ASM will start and stop tape unit 1 until the .EN pseudo op is encountered. At that point tape unit 1 is turned off, and the message RDY. FOR PASS 2 is outputted.

RAE is now in the TED mode. Rewind the tape unit (>ON 1 and >OFF 1 accordingly). Perform 1 or 2 as described above and type >PASS to perform the second pass. Again tape unit 1 will be turned on and off accordingly under control of the ASM software.

4.6 CREATING A RELOCATABLE OBJECT FILE

In order to create a relocatable object file, the programmer should define those labels whose address should not be altered by the relocating loader. This is done via the .DE pseudo op. Constants (example: $0169) are also considered as being external. All other labels (including those defined via the .DI pseudo op) are considered as internal. Addresses associated with internal labels are altered by an offset when the program is loaded via the relocating loader. Also .SE stores a two byte external address and .SI stores a two byte internal address. Similarly the relocating loader will alter the internal address and not the external address.

An example of an external address would be the calls to the system monitor or any location whose address remains the same no matter where the program is located. Locations in zero page are usually defined as external addresses. Expressions consisting of internal and external labels will be combined and considered an internal address. A label expression consisting entirely of external labels will be combined and considered as external.

To record a relocatable object file, insert a blank tape in tape unit 0 and ready. If the entire source program is in memory, simply type >OU.

If the source program is on tape type >OU, the ASM will turn both tape units on and off until the end of assembly. The relocatable object file will be recorded on the tape in unit 0.

After the relocatable object file has been recorded, record an end of file mark via the >PU X command.

4.7 MACROS

RAE provides macro capability. A macro is essentially a facility in which one line of source code can represent a function consisting of many instruction sequences. For example, the 6502 instruction set does not have an instruction to increment a double byte memory location. A macro could be written to perform this operation and represented as INCD (VALUE.1). This macro would appear in your assembly language listing in the mnemonic field similar to the following:

```
BNE SKIP
NOP
.
.
.
.
INCD  (VALUE.1) ;INCREMENT DOUBLE
LDA   TEMP
.
.
.
.
```

Before a macro can be used, it must be defined in order for ASM to process it. A macro is defined via the .MD (macro definition) pseudo op. Its form is:

```
!!!label .MD (l1 l2 . . . ln)
```

Where label is the name of the macro (!!! must precede the label), and l1, l2,..., ln are dummy variables used for replacement with the expansion variables. These variables should be separated using spaces, do not use commas.

To terminate the definition of a macro, use the .ME (macro end) pseudo op.

For example, the definition of the INCD (increment double byte) macro could be as follows:

```
!!!INCD      .MD    (LOC)         ;INCREMENT DOUBLE
             INC    LOC
             BNE    SKIP
             INC    LOC+1
SKIP         .ME
```

This is a possible definition for INCD. The assembler will not produce object code until there is a call for expansion. Note that a call for expansion occurs when you enter the macro name along with its parameters in the mnemonic field as:

```
INCD (TEMP) or INCD (COUNT) or INCD (COUN+2)
```

or any other labels or expressions you may choose.

NOTE

> **In the expansion of INCD the code to increment the variable LOC is not being generated; instead the code to increment the associated variable in the call for expansion. Also parentheses must be used with the parameter labels both in the definition and in the call.**

If you tried to expand INCD as described above more than once, you will get a !06 error message. This is a duplicate label error and it would result because of the label SKIP occurring in the first expansion and again in the second expansion.

There is a way to get around this and it has to do with making the label SKIP appear unique with each expansion. This is accomplished by rewriting the INCD macro as follows:

```
!!!INCD      .MD    (LOC)         ;INCREMENT DOUBLE
             INC    LOC
             BND    ...SKIP
             INC    LOC+1
...SKIP      .ME
```

The only difference is ...SKIP is substituted for SKIP. What the ASM does is to assign each macro expansion a unique macro sequence number (2^{16} maximum macros in each file). If the label begins with ... the ASM will assign the macro sequence number to the label. Thus, since each expansion of this macro gets a unique sequence number, the labels will be unique and the !06 error will not occur.

If the label ...SKIP also occurred in another macro definition, no !06 error will occur in its expansion if they are not nested. If you nest macros (i.e., one macro expands another), you may get a !06 error if each definition uses the ...SKIP label.

The reason this may occur is that as one macro expands another in a nest, they are each sequentially assigned macro sequence numbers. As the macros work out of the nest, the macro sequence numbers are decremented until the top of the nest. Then as further macros are expanded, the sequence numbers are again incremented. The end result is that it is possible for a nested macro to have the same sequence number as one not nested. Therefore if you nest macros, it is suggested that you use different labels in each macro definition.

Some further notes on macros are:

1. The macro definition must occur before the expansion.

2. The macro definition must occur in each file that references it. Each file is assigned a unique file sequence number (2^{16} maximum files in each assembly) which is assigned to each macro name. Thus the same macro definition can appear in more than one file without causing a !06 error. If a macro with the same name is defined twice in the same file, then the !06 error will occur.

3. Macros may be nested up to 32 levels. This is a limitation because there is only so much memory left for use in the stack.

4. If a macro has more than one parameter, the parameters should be separated using spaces - do not use commas.

5. The number of dummy parameters in the macro definition must match exactly the number of parameters in the call for expansion.

6. The dummy parameters in the macro definition must be symbolic labels. The parameters in the expansion may be symbolic or non-symbolic label expressions.

7. If the .ES pseudo op is entered, object code generated by the macro expansion will be output in the source listing. Also, comment lines within the macro definition will be output as blank lines during expansion. If .EC was entered, only the line which contained the macro call will be output in the source listing.

4.8 CONDITIONAL ASSEMBLY

ASM also provides a conditional assembly facility to conditionally direct the assembler to assemble certain portions of your program and not other portions. For example, assume you have written a CRT controller program which can provide either 40, 64 or 80 characters per line. Instead of having to keep 3 different copies of the program you could use the ASM conditional assembly feature to assemble code concerned with one of the character densities.

Before we continue with this example, let us describe the conditional assembly operators:

IFE expression If the expression equates to a zero quantity, then assemble to end of control block.

IFN expression If the expression equates to a non zero quantity then assemble to end of control block.

IFP expression If the expression equates to a positive quantity (or 0000), then assemble to end of control block.

IFM expression If the expression equates to a negative (minus) quantity, then assemble to end of control block.

******* Three asterisks in the mnemonic field indicates the end of the control block.

SET symbol = expression Set the previously defined symbol to the quantity calculated from the expression.

NOTE

All expressions are evaluated using 16 bit precision arithmetic.

Going back to the CRT controller software example, a possible arrangement of the program is as follows:

```
CHAR.LINE    .DE 40
             .
             .
             .
             IFE CHAR.LINE-40
;CODE FOR 40 CHAR./LINE
             .
             .
             .
             ***
             IFE CHAR.LINE-64
;CODE FOR 64 CHAR./LINE
             .
             .
             .
             ***
             IFE CHAR.LINE-80
;CODE FOR 80 CHAR.ILINE
             .
             .
             .
             ***
;COMMON CODE
```

Shown is the arrangement which would assemble code associated with 40 characters per line since CHAR.LINE is defined as equal 40. If you wanted to assemble for 80 characters, simply define CHAR.LINE as equal 80, with SET CHAR.LINE = 80.

Conditional assembly can also be incorporated within macro definitions. A very powerful use is with a macro you don't want completely expanded each time it is referenced. For example, assume you wrote a macro to do a sort on some data. It could be defined as follows:

```
EXPAND    .DE  0
!!!SORT   .MD
          IFN  EXPAND
          JSR  SORT.CALL ;CALL SORT
          ***
          IFE  EXPAND
          JSR  SORT.CALL
          JMP  ...SKIP
;SORT CODE FOLLOWS
SORT.CALL
          .
          .
          .
          RTS
...SKIP   SET EXPAND = 1
          ***
          .ME
```

In this example, EXPAND is initially set to 0. When the macro is expanded for the first time, EXPAND equals 0 and the code at SORT.CALL will be assembled. Also the first expansion sets EXPAND to I. On each succeeding expansion, only a JSR instruction will be assembled since EXPAND equals 1. Using conditional assembly in this example resulted in more efficient memory utilization over an equivalent macro expansion without conditional assembly.

4.9 ASSEMBLER DEFAULT PARAMETERS

* Assumes assembling from memory (otherwise use .CT).
* Does not store object code in memory (otherwise use .OS).
* Begins assembly at $0200 (otherwise use .BA).
* Output listing clear (otherwise use .LS or >ASSEMBLE LIST).
* Stops assembly on errors (otherwise use .CE).
* Stores object code beginning at $0200 unless a .BA or .MC is encountered and if .OS is present.
* Generates relocatable addresses.
* Macro object code is not output (otherwise use .ES).

SECTION 5.0

RELOCATING LOADER

A source listing of the relocating loader (Appendix D) is provided. The relocating loader is not part of the RAE program body, and the user will have to enter it via the listing.

If you prefer to have the loader reside in some other part of memory, you should enter the source into the text file, assemble, and then create a relocatable object file on tape.

To record a program in relocatable format, first assemble (without an .OS pseudo op) the program at location 0000 (.BA $0). Next create a relocatable object file via the >OU command. Terminate the relocatable object file with an end of file mark via the >PU x command. To reload a program in relocatable format, first enter the address where you want the program to reside in memory locations $00E0 (lo) and $00E1 (hi), the object file number into $0110, the relocatable buffer address in 00C8 (lo) and 00C9 (hi) and then start execution at $0200.

When executing the relocating loader, if an error or an end of file mark is detected, a break (BRK) instruction will be executed so as to return to the system monitor. The contents of register A indicates the following:

00 good load
EE error in loading

All programs to be created in relocatable format should be assembled at $0000. This is because the offset put in $00E0 and $00E1 before execution is added to each internal address by the loader in order to resolve addresses while relocating the program. If the program was originated at say $1000, then one would have to enter F200 as the offset in order to relocate to $0200 (i.e., F200+1000 = 0200). This is somewhat more confusing than an assembly beginning $0000.

In addition to the program memory space, the relocating loader uses the following memory locations:

00C8-00C9, 00DC-00E1
0110, 011E-0121, 017A-0184

plus other stack area for subroutine control.

SECTION 6.0

FILE NUMBERS

Information to be recorded on or read from tape via the >PU, >GE, and >OU commands may be assigned a file identification number to distinguish between files. A file number is a decimal number between 0 and 99. To enter a file number as a parameter in the >PU, >OU, or >GE commands, begin with the letter 'F' followed by the file number. Examples are F0, F17, F6, etc. If no file number is entered with the >PU >GE, and >OU commands, file number 0 will be assigned by default.

When loading, all files encountered will result in the outputting of their associated file numbers and file length in bytes. The loaded file has, in addition, the memory range of the location of the loaded data.

Example:

```
>GET F17
F00 01A3
F67 0847
F17 0F93  0200-1193
>
```

An end of file mark may be recorded via the >PU X command to indicate the end of a group of files. If an end of file mark is encountered when loading, FEE will be outputted and a return to the command mode will be performed.

SECTION 7.0

ERROR CODES

An error message of the form !**xx AT LINE YYYY/ZZ** where **xx** is the error code, **YYYY** is the line number, and **zz** is the file number, will be outputted if an error occurs. Sometimes an error message will output an invalid line number. This occurs when the error is on a non-existent line such as an illegal command input.

The following is a list of error codes not specifically related to macros:

17 Checksum error on tape load.
16 Illegal tape unit number.
15 Syntax error in >ED command.
14 Cannot generate relocatable object tape with errors or no previous assembly.
11 Missing parameter in >NU command.
10 Overflow in line # renumbering
CAUTION: You must properly renumber the text file or part of the file may be deleted by subsequent operations.
0F Overflow in text file - line not inserted.
0E Overflow in label file - label not inserted.
0D Expected hex characters, found none.
0C Illegal character in label.
0B Unimplemented addressing mode.
0A Error in or no operand.
09 Found illegal character in decimal string.
08 Undefined label (may be illegal label).
07 .EN pseudo op missing.
06 Duplicate label.
05 Label missing in .DE or .DI pseudo op.
04 .BA or .MC operand undefined.
03 Illegal pseudo op.
02 Illegal mnemonic.
01 Branch out of range.
00 Not a zero page address.
ED Error in command input.

The following is a list of error codes that are specifically related to macros:

2F Overflow in file sequence count; 2^{16} maximum.
2E Overflow in number of macros; 2^{16} maximum.
2B .ME without associated .MD.
2A Non-symbolic label in SET.
29 Illegal nested definition.
27 Macro definition overlaps file boundary.
26 Duplicate macro definition.
25 Number of macro reference parameters is different from the number of macro dummy parameters or illegal characters.
24 Too many nested macros; 32 maximum.
23 Macro definition not complete at .EN.
22 Conditional suppress set at .EN.
21 Macro in expand state at .EN.
20 Attempted expansion before definition.

SECTION 8.0

CONTROL CODES

ASCII characters whose hex values are between hex 00 and 20 are normally nonprinting characters. With a few exceptions, these characters will be output in the following manner; ^C where C is the associated printable character if hex 40 was added to its value. For example, ASCII 03 will be output as ^C, 18 as ^X, etc.

In addition, some of these control codes have special functions in RAE.

Control codes which have special functions are:

CONTROL B		go to BASIC.
CONTROL C		go to System Monitor (executes BRK instruction).
CONTROL D		delete - used by >EDIT form 2.
CONTROL F		find - used by >EDIT form 1 and form 2.
CONTROL C	*	bell.
CONTROL H	*	backspace (delete previous character).
CONTROL L	*	horizontal tab.
CONTROL J	*	line feed.
CONTROL M	*	carriage return.
CONTROL O		continue processing after break key but suppress output to CRT.
CONTROL Q	*	continue after break key.
CONTROL Tn		(as CONTROL Tn) toggle motor control on tape unit n.
CONTROL X		delete entire line entered.
CONTROL Y		jump to location $0000. Return via JMP to B0B1. NOTE: location $0000 must first be initialized by the user.
CONTROL Z		terminate processing and go to ">" mode after break key
CONTROL [	*	escape character.

***** - Non-printing control character.

SECTION 9.0

SPECIAL NOTES

* In addition to the program memory space the RAE uses the following memory locations:

0100 - up	depending on type of function
00B6 - 00FF	reserved for RAE and system monitor

plus other stack area for subroutine control. **The terminal input buffer is in locations 0135 - 0185.**

* Keep the cover closed on the tape unit as this keeps the cassette cartridge stable.

* When entering source modules (without .EN) you can perform a short test on the module by assembling the module while in the text file and looking for the !07 error. If other error messages occur, you have errors in the module. This short test is not a complete test but does check to make sure you have lined up the fields properly, not entered duplicate labels within the module, or entered illegal mnemonics or addressing modes.

* A 64 character/line (or greater) output device should be used with this program when outputting an assembly listing in order to provide a neat printout.

* Any keyboard input greater than 80 characters in length will be automatically inserted in the text file without the user having to enter a carriage return.

* Locations $00D5 (lo) and $00D6 (hi) contain the address of the present end of the label file. These locations contain invalid data until after the first assembly. This address +2 should contain a zero (a forward pointer).

* Locations $00D3 (lo) and $00D4 (hi) contain the address of the present end of the text file. This address +2 should contain a zero (a forward pointer).

* To find the address of an entry in the text file, output the line via the PR command, issue the BR command, and then get the contents of memory location 00DD, 00DE. This is an address which points to the end of the outputted line.

SECTION 10.0

SPECIFIC APPLICATION NOTES

1. The default file boundaries for RAE are: text file = 0200-0BFC, label file = 0C00-0EFC, and relocatable buffer = 0F00. When entering the file boundary via the SET command, enter the end address minus 3.

   Example: If the end = 0BFF, then enter 0BFC.

2. RAE provides software for controlling two tape motors. RAE assumes the record unit (unit 0) is connected to the SYM motor control. If the user implements motor control hardware for the play unit (unit 1), RAE can control it via APB7, pin A-15 ("1" = off, "0" = on).

3. MDT1000 has both unit 0 and unit 1 remote motor control hardware as standard hardware.

4. The following must exist for installation of RAE-1/2 (two 4K ROMs) into SYM.

   RAE P/N 02-0023 inserted into socket U22 and
   RAE P/N 02-0024 inserted into socket U23

   The jumpers must be configured as follows:

   C-1
   D-1
   G-2
   H-3
   L-46, 46*
   M-15, 16

5. The following must exist for installation of RAE-1 (one 8K ROM) into SYM.

   RAE-1 (P/N 02-0053A) in socket U23

   The jumpers must be configured as follows:

   D-1
   M-15, 16, 46, 47*

   Add the following:

   Jumper 4 to U2-1
   Jumper H to U2-2

   (U2 is an inverter located to the right of logo)

   RAE-1 (P/N 02-0053B) in socket U23

   The jumpers must be configured as follows:

   D-1
   H-4
   M-15, 16, 46, 47*

   For both versions, remove jumper from D to 3 and also jumper from H to 6.

6. A manually-entered patch is required for RAE-1 V1.0 when used on SYM for assembly from cassette tape. The user must enter a flag and a vector into zero page. The patch may be stored any place in RAM which does not conflict with RAE-1, SYM-1, or application software. Since RAE-1 cold start entry clears the flag to zero, the patch must be entered after first transferring control to RAE-1 and then exiting RAE-1.

* In early versions of SYM-1, jumper points 46 and 47 are not labeled. For these boards, jumper points 46 and 47 are identical to U10-7 and U10-9, respectively.

The patch shown below is placed at the end of the default label file.

LOCATION	**CONTENT**	**COMMENT**
EE	01	Enter flag
F6	F5	Enter vector to
F7	0E	. . .patch
EF5	AD	Patch is 3
EF6	11	. . .instructions
EF7	01	
EF8	D0	
EF9	03	
EFA	8D	Store 0 into
EFB	10	location $110
EFC	01	
EFD	4C	Jump back into
EFE	68	RAE-1
EFF	EF	

To install the patch, perform the following:

1. Enter RAE-1 Type: G B000 <CR>

2. Exit RAE-1 Type: BR <CR>

3. Use M command three times to modify EE, F6-F7, and EF8-EFF

4. Return to RAE warm entry Type: G <CR>

Appendices

APPENDIX A

ASCII CHARACTER CODES

DECIMAL	HEX	CHAR	DECIMAL	HEX	CHAR	DECIMAL	HEX	CHAR
000	000	NUL	043	02B	+	086	056	V
001	001	SOII	044	02C	,	087	057	W
002	002	STX	045	02D	-	088	058	X
003	003	ETX	046	02E	.	089	059	Y
004	004	EOT	047	02F	/	090	05A	Z
005	005	ENQ	048	030	0	091	05B	[
006	006	ACK	049	031	1	092	05C	\
007	007	BEL	050	032	2	093	05D	]
008	008	BS	051	033	3	094	05E	^
009	009	HT	052	034	4	095	05F	
010	00A	LF	053	035	5	096	060	\
011	00B	VT	054	036	6	097	061	a
012	00C	FF	055	037	7	098	062	b
013	00D	CR	056	038	8	099	063	c
014	00E	SO	057	039	9	100	064	d
015	00F	SI	058	03A	:	101	065	e
016	010	DLE	059	03B	;	102	066	f
017	011	DC1	060	03C	<	103	067	g
018	012	DC2	061	03D	=	104	068	h
019	013	DC3	062	03E	>	105	069	i
020	014	DC4	063	03F	?	106	06A	j
021	015	NAK	064	040	@	107	06B	k
022	016	SYN	065	041	A	108	06C	l
023	017	ETB	066	042	B	109	06D	m
024	018	CAN	067	043	C	110	06E	n
025	019	EM	068	044	D	111	06F	o
026	01A	SUB	069	045	E	112	070	p
027	01B	ESCAPE	070	046	F	113	071	q
028	01C	FS	071	047	G	114	072	r
029	01D	GS	072	048	H	115	073	s
030	01E	RS	073	049	I	116	074	t
031	01F	US	074	04A	J	117	075	u
032	020	SPACE	075	04B	K	118	076	v
033	021	!	076	04C	L	119	077	w
034	022	"	077	04D	M	120	078	x
035	023	#	078	04E	N	121	079	y
036	024	$	079	04F	O	122	07A	z
037	025	%	080	050	P	123	07B	{
038	026	&	081	051	Q	124	07C	|
039	027	`	082	052	R	125	07D	}
040	028	(	083	053	S	126	07E	-
041	029	)	084	054	T	127	07F	DEL
042	02A	*	085	055	U			

LF= Line Feed

FF=Form Feed

CR=Carriage Return

DEL=Rubout

^=Control Key

APPENDIX B

RAE I/O LINKAGES

The following describes user I/O linkages and page 0 (zero) vectors. Functions described include CRT, keyboard, break key, printer, CONTROL Y, and user. Page 0 (zero) locations $EC - $F7 are reserved for future RAE extensions.

BREAK KEY RAE vectors thru INSVEC ($A666) in system RAM for testing for the break key being depressed. This 3-byte location contains a JMP instruction. If you wish to substitute another routine to detect if the break key is depressed, change the 2-byte address part of the JMP instruction to point to the alternate break key processing routine.

CONTROL Y When a CONTROL Y (^Y) is entered, RAE “JUMPS” to location $0000 for execution of user supplied instructions. RAE does not enter any default code at this location. The user supplied routine can reenter RAE via a JMP to the warm start address ($B003). None of the registers need be preserved.

CRT RAE vectors thru OUTVEC ($A663) in system RAM for outputting to the CRT. This 3-byte location contains a JMP instruction. If you wish to redirect output to another device such as a printer, change the 2-byte address part of the JMP instruction to point to the alternate devices software driver.

KEYBOARD RAE vectors thru INVEC ($A660) in system RAM for inputting from the keyboard. This 3-byte location contains a JMP instruction. If you wish to redirect output to another device such as a TTY, change the 2-byte address part of the JMP instruction to point to the alternate devices software driver.

PRINTER RAE reserves 3-bytes starting at $00B6 which the user can use to vector to a routine which drives an alternate output device. On cold start, RAE enters an RTS at this vector. When an >HA SET command is initiated, program control is transferred thru this vector for driving an alternate output device while outputting to the CRT at the same time. Register A will contain the ASCII character to be output. Registers X and Y should be preserved and the decimal mode bit in the PSR should be left cleared. Outputting thru this vector is terminated via >HA CLEAR.

USER When a user command is entered, RAE "JUMPS" to location $0003 for execution of user supplied instructions. RAE does not enter any default code at this location. The user supplied routine can reenter RAE via a JMP to the warm start address ($B003). None of the registers need be preserved.

APPENDIX C

CONVERTING MOS TECHNOLOGY/SYSTEM 65 ASSEMBLY LANGUAGE PROGRAMS TO RAE

This table shows by example, how to translate from MOS Technology/System 65 syntax to RAE syntax.

LINE		**MOS TECHNOLOGY**		
NO.		**SYSTEM 65**		**RAE**
.0006		* = $A600		.BA $A600
.0007	SCPBUF	* = *+$20	SCPBUF	.DS $20
.0008	RAM	= *	RAM	.DI = ;or just RAM
.0024	RC	= SCRD	RC	.DI SCRD
.0087	PADA	= $A400	PADA	.DE $A400
.0252		BYT $FF,$FF,$FF		.BY $FF $FF $FF
.0430		BNE *+5		.BNE = +5
.0434		STX $FF		STX *$FF
.1589	ASCMI	= *-1	ASCM1	.DI = -1
.1711	STDVAL	.DBY $D54C,$2410	STDVAL	BY $D5 $4C
.1723		.WORD $C000		.SE $C000
.1724		.WORD TTY		.SI TTY
.1778		.END		.EN
		.LDA #>EXPRESSION		LDA #H,EXPRESSION
		.LDA #<EXPRESSION		LDA #L,EXPRESSION

1. The following RAE directives do not have equivalent functions in the MOS Technology/System 65 assembler:

   .LS, **.LC**, **.OS**, **.OC**, **.RC**, **.MD**, **.ME**, **.EC**, **.ES**, **.RS**

2. The following RAE directives have similar functions in the System 65 assembler:

   .CE is an intrinsic attribute of the System 65 assembler.

   .EJ is ".PAG" in the System 65 assembler.

   .MC is implemented in a constrained fashion on System 65 i.e., the "entire" object program may be assembled into a different memory space than the one specified for execution.

   .CT is available on System 65 for source stored in multiple disk files via the FILE directive.

APPEDIX D

```
0010 ;****RELOCATING LOADER FOR SYNERTEK SYSTEMS RAE-1
0020 ;
0030 ;
0040 ;
0050            .OS
0060 ;
0070 ;***COPYRIGHT 1979 BY SYNERTEK SYSTEMS CORP.***
0080 ;***    ALL RIGHTS RESERVED.                ***
0090 ;
0100 ;
0110 ;
0120 ;
0130 ;++++++  USER INPUTTED VARIABLES BEFORE EXECUTION  ++++++
0140 FILE/NO     .DE $0110     ;FILE NUMBER (0-99)
0150 OFFSET      .DE $E0 ;RELOCATOR OFFSET (2 BYTES)
0160 BUFFER      .DE $C8 ;ADDRS. OF R.L. BUFFER
0170 ;
0180 ;
0190 ;
0200 ;           RELOCATOR DIRECTIVES
0210 ;
0220 ;  DIRECTIVE              DESCRIPTION
0230 ;  ---------              -----------
0240 ;      0F       EXTERNAL 2 BYTE ADDRS. PRECEEDS
0250 ;               DON'T RELOCATE. OTHERWISE RELOCATE.
0260 ;
0270 ;      1F       #L,  DATA PRECEEDS
0280 ;
0290 ;      2F       #H,  DATA PRECEEDS, LO PART FOLLOWS.
0300 ;
0310 ;      3F       .AS OR .HS BYTE FOLLOWS.
0320 ;
0330 ;      4F       .SE OR .SI 2 BYTE ADDRS. FOLLOWS.
0340 ;
0350 ;      5F       TURN RELOCATOR ON (VIA .RS).
0360 ;               RESOLVE ADDRESSES AND RELOCATE CODE
0370 ;
0380 ;      6F       TURN RELOCATOR OFF (VIA .RC).
0390 ;               RESOLVE ADDRESSES BUT DO NOT
0400 ;               RELOCATE CODE
0410 ;
0420 ;      7F       .DS - 2 BYTE BLOCK VALUE FMLLMUS,
0430 ;
0440 ;
0450            .BA $0200
0460 ;
0470 ;TAPE INPUT PARMS
0480 LORD/NO     .DE  $0180    0: NO STORE; 1: STORE
0490 TSTART      .DE  $A64C    LOAD BEGINNING AT TSTART
0500 TEND        .BE  $A64A    STOP LOADING AT TEND
0510 ;
0520 ;
0530 ;HEADER INPUT DATA
0540 HFILE/NO    .DE  $017A    HEADER FILE NUMBER
0550 HSTART      .DE  $017B    HEADER START
```

```
                 0560 HEND        .DE $017D HEADER END
                 0570 ;
                 0580 ;
                 0590 ;VARIABLES
                 0600 SCRAT       .DE $11E SCRATCH AREA
                 0610 TEMP1       .DE $11F SCRATCH AREA
                 0620 TEMP2       .BE $120 SCRATCH AREA
                 0630 SAVE        .BE $121 SCRATCH AREA
                 0640 ADDRS       .DE $DC 4 BYTES OF ADDRESS INFO.
                 0650 BUFF.END    .DE $0123 END OF 256 BYTE BUFFER
                 0660 BUFF.INDEX  .DE $0124 PRESENT ACCESSED DATA FROM BUFFER
                 0670 ;
                 0680 ;
                 0690 ;R(X)=00:   RELOCATOR ON
                 0700 ;R(X)=02:   RELOCATOR OFF
                 0710 ;
                 0720 ;BEGIN EXECUTION AT LABEL START
                 0730 ;
0200- A2 FF      0740 START       LDX #$FF
0202- 9A         0750             TXS INITIALIZE STACK
0203- E8         0760             INX R(X)=00: SET RELOCATOR INITIALLY TO ON
0204- 20 86 8B   0770             JSR ACCESS
0207- D8         0780             CLD
0208- 8E 21 01   0790             STX SAVE R(X)=00
020B- 20 E6 02   0800             JSR LOAD<BUFF
020E- 4C 14 02   0810             JMP ENTY
0211- 20 74 03   0820 LOOP1       JSR GET<DATA
                 0830 ;
0214- C9 7F      0840 ENTY        CMP #$7F     ;CKG. FOR .DS
0216- D0 03      0850             BNE PRO.SF
0218- 4C AA 03   0860             JMP PRO.7F   ;JUMTO PROCESS DIR. 7F
021B- C9 3F      0870 PRO.SF      CMP #$3F CKG. FOR RELOCATOR DIRECTIVE
021D- D0 0B      0880             BNE OP<CKG
021F- 20 74 03   0890             JSR GET<DATA
0222- 81 DC      0900             STA (ADDRS,X)
0224- 20 88 03   0910             JSR INC<ADDRS
0227- 4C 11 02   0920             JMP LOOP1
022A  C9 4F      0930 OP<CKG      CMP #$4F CKG. FOR .SE, .SI
022C- D0 03      0940             BNE W:
022E- 4C AD 02   0950             JMP TWO<BYT<AD
0231- C9 5F      0960 W:          CMP #$5F CKG. FOR RELOCATOR ON
0233- D0 04      0970             BNE CKNX
0235- A2 00      0980             LDX #$00
0237- F0 D8      0990             BEQ LOOP1
                 1000 ;
0239- C9 6F      1010 CKNX        CMP #$6F CKG. FOR RELOCATOR OFF
023B- D0 04      1020             BNE NO<REL
023D- A2 02      1030             LDX #$02
023F- D0 D0      1040             BNE LOOP1
0241- 81 DC      1050 NO<REL      STA (ADDRS,X) STORE OP CODE
0243- 20 88 03   1060             JSR INC<ADDRS
0246- C9 00      1070             CMP #$00 CKG.
0248- F0 C7      1080             BEQ LOOP1
024A- C9 20      1090             CMP #$20 OKG. FOR JSR INSTR.
024C- F0 5F      1100             BEQ TWO<BYT<AD
024E- 8D 21 01   1110             STA SAVE SAVE R(A), IT CONTAINS OP CODE
0251- 29 9F      1120             AND #$9F
```

```
0253- F0 BC       1130             BEQ LOOP1
0255- AD 21 01    1140             LDA SAVE RESTORE OP CODE
0258- 29 1D       1150             AND #$1D
025A- C9 08       1160             CMP #$08 CKG. FOR ONE BYTE INSTR.
025C- F0 B3       1170             BEQ  LOOP1
025E- C9 18       1180             CMP  #$18 CKG. FOR ONE BYTE INSTR.
0260- F0 AF       1190             BEQ  LOOP1
                  1200 ;
                  1210 ;NOW, TEST FOR INSTR. CONTAINING 2 BYTES
                  1220 ;OF ADDRESS INFORMATION
                  1230 ;
0262- AD 21 01    1240             LDA  SAVE RESTORE OP CODE
0265- 29 1C       1250             AND  #$1C
0267- C9 1C       1260             CMP  #$1C
0269- F0 42       1270             BEQ  TWO<BYT<AD
026B- C9 18       1280             CMP  #$18
026D- F0 3E       1290             BEQ  TWO<BYT<AD
026F- C9 0C       1300             CMP  #$0C
0271- F0 3A       1310             BEQ  TWO<BYT<AD
                  1320 ;
                  1330 ;THE REMAINING CONTAIN ONE BYTE OF
                  1340 ;ADDRESS INFORMATION
                  1350 ;
                  1360 ;PROCESSING OF ONE BYTE ADDRESSES AND IMMEDIATE DATA
                  1370 ;
0273- 20 74 03    1380 ONE<BYT<AD JSR  GET<DATA
0276- 81 DC       1390             STA  (ADDRS,X)
0278- 20 88 03    1400             JSR  INC<ADDRS
027B- 20 74 03    1410             JSR  GET<DATA
027E- C9 2F       1420             CMP  #$2F CKG. FOR RELOCATOR DIRECTIVE
0280- F0 14       1430             BEQ  IMM<HI CKG. FOR #H,
0282- C9 1F       1440             CMP  #$1F CKG. FOR RELOCATOR DIRECTIVE
0284- D0 BE       1450             BNE  ENTY
                  1460 ;
                  1470 ;PROCESS    #L, DATA FOR RELOCATION
0286- 20 95 03    1480 IMM<LO      JSR  DEC<ADDRS
0289- 10          1490             CLC
028A- A1 DC       1500             LDA  (ADDRS,X)
028C- 65 E0       1510             ADC  *OFFSET+00 ADD OFFSET LOW PART FOR #L
028E- 81 DC       1520             STA  (ADDRS,X)
0290- 20 88 03    1530             JSR  IMC<ADDRS
0293- 4C 11 02    1540 BACK<TO<L1 JMP  LOOP1
                  1550 ;PROCESS #H, DATA FOR RELOCATION
0296- 20 74 03    1560 IMM<HI      JSR  GET<DATA LOW BYTE FOLLOWS REL. DIR.
0299- 18          1570             CLC
029A- 65 E0       1580             ADC  *OFFSET FROM THE LO ADDRS. PART
029C- 08          1590             PHP
029D- 20 95 03    1600             JSR  DEC<ADDRS
02A0- 28          1610             PLP
02A1- A1 DC       1620             LDA  (ADDRS,X)
02A3- 65 E1       1630             ADC  *OFFSET+$1 NOW FORM THE EFFECTIVE #H,
02A5- 81 DC       1640             STA  (ADDRS,X)
02A7- 20 88 03    1650             JSR  INC<ADDRS
02AA- 4C 11 02    1660             JMP  LOOP1
                  1670 ;
                  1680 ;PROCESSING OF TWO BYTE ADDRESSES
02AD- A0 82       1690 TWO<BYT<AD LDY  #$02
```

```
02AF- 98          1700 XX         TYA
02B0- 48          1710            PHA  SAVE R(Y)
0281- 20 74 03    1720            JSR  GET<DATA
0284- 81 DC       1730            STA  (ADDRS,X)
02B6- 20 88 03    1740            JSR  INC<ADDRS
02B9- 68          1750            PLA
02BA- A8          1760            TAY  RESTORE R(Y)
02BB- 88          1770            DEY
02BC- D0 F1       1780            BNE  XX
02BE- 20 74 03    1790            JSR  GET<DATA
02C1- C9 0F       1800            CMP  #$0F CKG. FOR RELOCATOR DIRECTIVE
02C3- D0 03       1810            BNE  XY
02C5- 4C 11 02    1820            JMP  LOOP1
02C8- 48          1830 XY         PHA
02C9- 20 95 03    1840            JSR  DEC<ADDRS
02CC- 20 95 83    1850            JSR  DEC<ADDRS
                  1860 ;DECREMENT BACK TO ADDRESS START
                  1870 ;
02CF- A1 DC       1880            LDA  (ADDRS,X)
02D1- 18          1890            CLC
02D2- 65 E0       1900            ADC  *OFFSET AND OFFSET LO
02D4- 81 DC       1910            STA  (ADDRS,X)
02D6- 20 88 03    1920            JSR  INC<ADDRS
02D9- A1 DC       1930            LDA  (ADDRS,X)
02DB- 65 E1       1940            ADC  *OFFSET+$1 ADD OFFSET HI
02DD- 81 DC       1950            STA  (ADDRS,X)
2DDF- 20 88 03    1960            JSR  INC<ADDRS
02E2- 68          1970            PLA
02E3- 4C 14 02    1980            JMP  ENTY
                  1990 ;
                  2000 ;SUBROUTINE LOAD BUFFER WITH DATA FROM TAPE
                  2010 ;
02E6- A9 7A       2020 LOAD<BUFF  LDA  #$7A ADDLO OF START OF HEADER
02E8- 88 4C A6    2030            STA  TSTART+$00
02EB- A9 7F       2040            LDA  #$7F ADDLO OF END OF HEADER
02ED- 88 4A AB    2050            STA  TEND+$00
02F0- A9 01       2060            LDA  #$01 HI ADDRS
02F2- 8D 4D A6    2070            STA  TSTART+$01
02F5- 8D 4B A6    2080            STA  TEND+$01
02F8- 8D 80 01    2090            STA  LOAD/NO 01: INDICATE TO LOAD
02FB- 20 D5 03    2100            JSR  USER/LOAD USER LDA^BD FROM TAPE ROUTINE
                  2110 ;
                  2120 ;THE ABOVE SETS UP AND LOADS HEADER INFORMATION
                  2130 ;FROM TAPE. THE HEADER CONTAINS THE MODULE FILE
                  2140 ;NUMBER, AND STARTING AND ENDING ADDRESSES OF
                  2150 ;FOLLOWING DATA.
                  2160 ;
                  2170 ;
02FE- D0 4D       2180            BNE  ERROR IF Z-BIT FALSE, ERROR IN LOADING
0300- A2 00       2190            LDX  #$00
                  2200 ;
0302- AD 78 01    2210            LDA  HEND+$00
0305- 3B          2220            SEC
0306- ED 7B 01    2230            SBC  HSTART+$00
                  2240 ;CALCULATE NUMBER OF BYTES IN FOLLOWING DATA
                  2250 ;
0309- 8D 23 01    2260            STA  BUFF.END INITIALIZE BUFFER END
```

```
030C- AD 7E 01    2270               LDA  HEND+$01 POINTER
030F- ED 7C 01    2280               SBC  HSTART+$01
0312- D0 39       2290               BNE  ERROR ONLY 256 BYTE BUFFER ALLOWED
0314- A5 C8       2300               LDA  *BUFFER
0316- 8D 4C A6    2310               STA  TSTART
0319- 18          2320               CLC
031A- 6D 23 01    2330               ADC  BUFF.END # BYTES
031D- BD 4A A6    2340               STA  TEND
0320- A5 C9       2350               LDA  *BUFFER+01
0322- 8D 4D A6    2360               STA  TSTART+$01
0325- 69 00       2370               ADC  #$00
0327- 8D 4B A6    2380               STA  TEND+$D1
                  2390 ;NOW THE START AND END ADDRESS PARMS HAVE BEEN
                  2400 ;SET UP TO LOAD FROM TARE INTO THE BUFFER.
                  2410 ;
032A- AD 10 01    2420               LDA  FILE/NO USER ENTERED FILE NUMBER
032D- F0 08       2430               BEQ  STORE.DATA IF F# = 00 . LOAD ANYWAY
032F- CD 7A 01    2440               CMP  HFILE/NO CMP WITH USER VERSUS THAT ON
TARE
0332- F0 03       2450               BEQ  STORE.DATA
0334- 8E 80 01    2460               STX  LOAD/NO R( X )=0; NO STORE
0337- 20 D5 03    2470 STORE.DATA JSR  USER/LOAD
                  2480 ;
                  2490 ;THE ABOVE LOADS IN DATA INTO BUFFER DEPENDING
                  2500 ;ON THE STATE OF LOAD/NO
                  2510 ;
033A- D0 11       2520               BNE  ERROR Z-BIT = FALSE THEN ERROR
033C- A2 00       2530               LDX  #$00
033E- AD 7A 01    2540               LDA  HFILE/NO
0341- C9 EE       2550               CMP  #$EE COMPARE IF END OF FILE
0343- D0 0C       2560               BNE  BUFFLOADED
0345- A9 00       2570               LDA  #$00 INDICATE GOOD LOAD
0347- 00          2580 B             BRK
0348- EA          2590               NOP
0349- EA          2600               NOP
034A- 4C 00 02    2610               JMP  START
034D- A9 EE       2620 ERROR         LDA  #$EE INDICATE ERROR IN LOAD
034F- D0 F6       2630               BNE  B
                  2640 ;
                  2650 ;
                  2660 ;NOW GET ADDRS. INFO AND PUT IN ADDRS+$2, +$3
                  2670 ;ADDRS. INFO  IS IN FIRST TWO BYTES OF BUFFER
                  2680 ;
0351- AD 80 01    2690 BUFFLORDED    LDA  LOAD/NO CKG. IF PROPER DRTR
0354- F0 90       2700               BEQ  LOAD<BUFF
0356- AE 21 01    2710               LDX  SAVE RESTORE R(X)
0359- A0 00       2720               LDY  #$0
035B- B1 C8       2730               LDA  (BUFFER),Y
035D- 85 DE       2740               STA  *ADDRS+$2
035F- C8          2750               INY
0360- B1 C8       2760               LDA  (BUFFER),Y
0362- 85 DF       2770               STA  *ADDRS+$3
0364- 8C 24 01    2780               STY  BUFF.INDEX SET BUFFER DATA POINTER
                  2790 ;
                  2800 ;SET RELOCATION ADDRS. IN ADDRS+$0, +$1
0367- A5 DE       2810               LDA  *ADDRS+$2
0369- 18          2820               CLC
```

```
036A- 65 E0        2830                ADC  *OFFSET
036C- 85 DC        2840                STA  *ADDRS
036E- A5 E1        2850                LDA  *OFFSET+$1
8370- 65 DF        2860                ADC  *ADDRS+$3
0372- 85 DD        2870                STA  *ADDRS+$1
                   2880 ;
0374- 8E 21 01     2890 GET<DATA       STX  SAVE X IN CASE WE BR. TO LOAD/BUFF
0377- EE 24 01     2900                INC  BUFF.INDEX INC. 256 BYTE BUFFER POINTER
037A- AC 24 01     2910                LDY  BUFF.INDEX
037D- CC 23 01     2920                CPY  BUFF.END
0380- 90 03        2930                BCC  WX BR. IF NOT AT END OF DATA IN BUFFER
0382- 4C E6 02     2940                JMP  LOAD<BUFF RELOAD BUFFER
0385- B1 C8        2950 WX             LDA  (BUFFER),Y
0387- 60           2960                RTS
                   2970 ;
                   2980 ;
                   2990 ;INCREMENT ADDRS+$0, +$1 AND ADDRS+$2, +$3
                   3000 ;
0388- E6 DC        3010 INC<ADDRS      INC  *ADDRS
038A- D0 02        3020                BNE  SKIP<INC1
038C- E6 DD        3030                INC  *ADDRS+$1
038E- E6 DE        3040 SKIP<INC1      INC  *ADDRS+$2
0390- D0 02        3050                BNE  SKIP<INC2
0392- E6 DF        3080                INC  *ADDRS+$3
0394- 60           3070 SKIP<INC2      RTS
                   3080 ;
                   3090 ;
                   3100 ;DECREMENT ADDRS+$0, +1 AND ADDRS+$2, +$3
                   3110 ;
0395- C6 DC        3120 DEC<ADDRS      DEC  *ADDRS
0397- A5 DC        3130                LDA  *ADDRS
0399- C9 FF        3140                CMP  #$FF
039B- D0 02        3150                ONE  SKIP<DEC1
039D- C6 DD        3160                DEC  *ADDRS+$1
039F- C6 DE        3170 SKIP<DEC1      DEC  *ADDRS+$2
03A1- A5 DE        3180                LDA  *ADDRS+$2
03A3- C9 FF        3190                CMP  #$FF
03A5- D0 02        3200                BNE  SKIP<DEC2
03A7- C6 DF        3210                DEC  *ADDRS+$3
03A9- 60           3220 SKIP<DEC2      RTS
                   3230 ;
                   3240 ;
                   3250 ;7F LO HI      -- PCL PCH 7F LO HI
                   3260 ;
03AA- 20 74 03     3270 PRO.7F         JSR  GET<DATA
03AD- 48           3280                PHA  ;SAVE LO
03AE- 20 74 03     3290                JSR  GET<DATA
03B1- A8           3300                TAY  ;SAVE HI IN R(Y)
03B2- AD 24 01     3310                LDA  BUFF.INDEX
03B5- C9 05        3320                CMP  #$05      ;NO PROC. IF <= 4
03B7- 90 18        3330                BCC  NO.PROC
03B9- 18           3340 PROC.DS        CLC
03BA- 68           3350                PLA  ;GET LO
03BB- 48           3360                PHA
03BC- 65 DC        3370                ADC  *ADDRS
03BE- 85 DC        3380                STA  *ADDRS
03C0- 98           3390                TYA  ;GET HI
```

```
03C1- 65 DD       3400            ADC  *ADDRS+1
03C3- 85 DD       3410            STA  *ADDRS+1
03C5- 68          3420            PLA
03C8- 48          3430            PHA  ;GET LO
03C7- 18          3440            CLC
03C8- 65 DE       3450            ADC  *ADDRS+2
03CA- 85 DE       3480            STA  *ADDRS+2
03CC- 98          3470            TYA  ;GET HI
03CD- 65 DF       3480            ADC  *ADDRS+3
03CF- 85 DF       3490            STA  *ADDRS+3
03D1- 68          3500 NO PROC    PLA
03D2- 4C 11 02    3510            JMP  LOOP1
                  3520 ;
                  3530 ;
                  3540 ;
                  3550 ;      ***SYSTEM MONITOR CASSETTE INTERFACE***
                  3560 ;
                  3570 ;
                  3580 ;
                  3590 ;    DEFINITIONS:
                  3600 SAVER      .DE  $8188
                  3610 ACCESS     .DE  $8B86
                  3620 ID         .DE  $A64E
                  3630 MODE       .DE  $FD
                  3640 CONFIG     .DE  $89A5
                  3650 ZERCK      .DE  $832E
                  3650 P2SCR      .DE  $829C
                  3670 LOADT      .DE  $8C78
                  3680 MACCESS    .DE  $8B9C
                  3690 RESXAF     .DE  $81B8
                  3700 ;
                  3710 ;
03D5  20 88 81    3720 USER/LOAD  JSR  SAVER     ;SAVE REGISTERS
03D8- A9 FF       3730            LDA  #$FF      ;ID=FF FOR USER RANGE
03DA- 85 4E A6    3740            STA  ID
03DD- A0 80       3750            LDY  #$80
03DF- 84 FD       3760            STY  *MODE     ;BIT 7=1 FOR H.S.
03E1- A9 09       3770            LDA  #$09
03E3- 20 A5 89    3780            JSR  CONFIG
03E6- 20 2E 83    3790            JSR  ZERCK
03E9- 20 9C 82    3800            JSR  P2SCR
03EC- 20 7B 8C    3810            JSR  LOADT+$3   ;ENTRY IN TAPE LOAD
03EF- D8          3820            CLD
03F0- A9 00       3830            LDA  #$00      ;Z-BIT = T
03F2- 90 02       3840            BCC  SKPERRU/L
03F4- A9 01       3850            LDA  #$01      ;Z-BIT = F
03F6- 4C B8 81    3860 SKPERRU/L  JMP  RESXAF ;RESTORE REGS. EXCEPT A.PSR
                  3870 ;
                  3880 ;
                  3890 END.PGM    .EN
```

```
LABEL FILE: [ / = EXTERNAL ]

/FILE/NO=0110            /OFFSET=00E0             /BUFFER=00C8
/LOAD/NO=0180            /TSTART=A64C             /TEND=A64A
/HFILE/NO=017A           /HSTART=017B             /HEND=017D
/SCRAT=011E              /TEMP1=011F              /TEMP2=0120
/SAVE=0121               /ADDRS=00DC              /BUFF.END=0123
/BUFF.INDEX=0124         START=0200               LOOP1=0211
ENTY=0214                PRO.SF=021B              OP<CKG=022A
W:=0231                  CKNX=0239                NO<REL=0241
ONE<BYT<AD=0273          IMM<LO=0286              BACK<TO<L1=0293
IMM<HI=0296              TWO<BYT<AD=02AD          XX=02AF
XY=02C8                  LOAD<BUFF=02E6           STORE.DATA=0337
B=0347                   ERROR=034D               BUFFLOADED=0351
GET<DATA=0374            WX=0385                  IMC<ADDRS=0388
SKIP<INC1=038E           SKIP<INC2=0394           DEC<ADDRS=0395
SKIP<DEC1=039F           SKIP<DEC2=03A9           PRO.7F=03AA
PROC.DS=03B9             NO.PROC=03D1             /SAVER=8188
/ACCESS=8B86             /ID=A64E                 /MODE=00FD
/CONFIG=89A5             /ZERCK=832E              /P2SCR=829C
/LOADT=8C78              /NACCESS=8B9C            /RESXAF=81B8
USER/LOAD=03D5           SKPERRU/L=03F6           END.PGM=03F9

//0000,03F9,03F9
```

APPENDIX E

The GoertzWorks! Ram Model

Synertek Systems released RAE-1 in 1979. At that time mass storage devices, large memory arrays, CRT based I/O devices, and fast modems were not within reach to the home user. This becomes obvious while reading the users manual. It discusses assembling to cassette tape (no large memory arrays) and how to attach a TTY (no CRT based I/O). RAE-1 was, and still is, a very powerful editor / assembler combination. The GoertzWorks! Ram model helps get around these limitations.

The GoertzWorks! Ram model consists of the following:

RAE-1 in its entirety. However, it is loaded into ram via floppy. There are no ROMs. It still occupies $B000-$BFFF and $E000-$EFFF.

Boot executives to automatically modify the SET limits to:

Text file:	**$1000-$7FFC**
Label file:	**$D000-$DFFC**
Object file reserve:	**$0200-$0FFF**
RELOCATABLE buffer	**Not needed but can be located anywhere free**

Attachments to load and save source and object files to floppy.

9600 baud on a 1mhz system.

Two full handshake RS-232 I/O ports with the ability to input and output to both ports at the same time.

Modem support.

Real time disassembling debugger.

RAE-1 formatted disassembler.

Standard hex dump.

Some of the above require modifications to the SYM-1 hardware but are minor.

The SYM-1 the GoettzWorks! Ram model runs on contains only the **SUPERMON** and **SYMDOS** ROMs. The remaining address space is occupied completely with ram.

These additions make RAE-1 very usable, even in today's "WINTEL" world. It is rally nice to be able to edit / assemble / debug / run on the same machine - no need to transfer object code from a WINTEL system.

I have found nothing to replace it.

Leland Goertz
kd6mzu@attitude.com

APPENDIX F

SOFTWARE LISTING

Enter range limits for hex dump: B000-BFFF

```
     00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
B000 4C ED BF 4C AE B0 0C E6 B4 E8 BB E8 2D EC 4D B4,ED  Lm?L.0 f4h;h-lM4
B010 AF E3 A2 FF 9A E8 86 EE 86 EF 8E 13 01 86 E3 8E,24  /c"□ h n o    c
B020 1F 01 8E 53 A6 20 EB E5 AD 02 A0 09 80 8D 02 A0,C2     S& ke-
B030 A9 01 8D 20 01 8D 21 01 8D 0F 01 85 DB 8E 0C 01,61  )     !     [
B040 8E 0D 01 8E 0E 01 A0 47 20 1E B5 A9 0B 85 EA 20,B7         G  5)  j
B050 F7 EA 20 96 B0 8E 28 01 20 0F E3 20 18 E3 A2 00,84  wj  0 (   c  c"
B060 D8 20 86 8B A9 00 8D 53 A6 8D 13 01 8D 33 01 20,3E  X   )  S&     3
B070 CA E3 A2 FF 8E 12 01 8E 11 01 9A E8 8E 14 01 8E,80  Jc"□       h
B080 15 01 20 64 B2 A0 00 20 02 B5 C0 50 B0 E4 20 AE,B5     d2    5@P0d .
B090 B6 A2 ED 4C 4D B4 AD 00 01 85 D3 AD 01 01 85 D4,55  6"mLM4-   S-   T
B0A0 8A A0 02 91 D3 60 A9 80 8D 53 A6 00 EA EA 20 F7,DF      S`)  S& jj w
B0B0 EA 4C 5E B0 A2 0C 20 E6 B2 4C 60 B0 20 49 E4 4C,7E  jL^0"  f2L`0 IdL
B0C0 5E B0 20 A0 B6 8D 0F 01 C9 43 D0 03 8E 0F 01 20,3C  ^0  6   ICP
B0D0 FF B4 C0 50 B0 13 8E 11 01 A9 01 8D 13 01 20 4A,17  □4@P0    )     J
B0E0 E2 E6 D1 A5 D1 29 1F 85 EA 4C 60 B0 20 A0 B6 8D,3C  bfQ%Q)  jL`0  6
B0F0 0E 01 C9 43 D0 03 8E 0E 01 4C 60 B0 8E 14 01 8E,54    ICP    L`0

     00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
B100 13 01 98 48 20 37 BF 20 5F E0 8E 11 01 68 A8 8E,FB     H 7? _`   h(
B110 17 01 20 A0 B6 C9 4E F0 07 C9 4C D0 06 EE 17 01,88       6INp ILP n
B120 20 FF B4 20 FC B1 AD 08 01 8D 2F 01 AD 09 01 8D,DF   □4 |1-   / -
B130 30 01 4C C7 BB A9 01 8E 11 01 8D 13 01 20 4A E2,15  0 LG;)        Jb
B140 20 FF B4 20 49 B1 4C 60 B0 4C EB EF B9 35 01 C9,3C   □4 I1L`0Lko95 I
B150 2F D0 18 A9 FF 8D 09 01 20 14 B2 AD 1C 01 85 DD,A4  /P )□     2-   ]
B160 AD 1D 01 85 DE 20 CB B1 4C 60 B0 20 FC B1 20 B5,6C  -   ^ K1L`0 |1 5
B170 B1 4C 60 B0 A5 C8 8D 29 01 18 6D 22 01 8D 2B 01,FE  1L`0%H )  m"  +
B180 A5 C9 8D 2A 01 69 00 8D 2C 01 8E 22 01 20 8F E3,8A  %I * i  ,  "   c
B190 84 CE 20 7C E5 A4 CE 20 9D E3 60 8E 10 01 20 A0,2E   N |e$N  c`
B1A0 B6 C9 46 D0 0F C8 A2 0A 20 E6 B2 AD 0A 01 8D 10,53  6IFP H"  f2-
B1B0 01 20 02 B5 60 F0 10 90 12 A0 02 88 30 0D B9 0A,57     5`p      0 9
B1C0 01 D1 DD F0 F6 B0 04 20 E8 B1 60 20 B2 B3 AD 0E,F9   Q]pv0  h1` 23-
B1D0 01 D0 06 20 32 B6 20 D9 E3 20 A8 B5 20 CA E3 20,1E   P  26 Yc (5 Jc
B1E0 A4 B3 A0 02 B1 DD D0 D1 20 EF B1 20 CA E3 60 86,B9  $3  1]PQ o1 Jc`
B1F0 E3 A9 2F 20 A4 E3 A9 2F 20 A4 E3 60 A2 08 20 E6,AA  c)/ $c)/ $c`"  f
```

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
B200 B2 C0 50 90 07 A9 FF 8D 0B 01 D0 08 20 02 B5 A2,95   2@P  )□    P    5"
B210 0A 20 E6 B2 20 BF B4 A0 02 B1 DD D0 02 18 60 88,EC    f2 ?4  1]P   `
B220 30 FB B1 DD D9 08 01 F0 F6 90 01 60 A5 DD 8D 1C,89   0{1]Y  pv  `%]
B230 01 A5 DE 8D 1D 01 20 A4 B3 4C 17 B2 84 CE AC 22,64    %^     $3L 2 N,"
B240 01 91 C8 EE 22 01 A4 CE 60 AD 13 01 10 08 A9 4F,72     Hn" $N`-     )O
B250 EE 11 01 20 70 BE 4C 11 BE A0 02 B1 DD 60 EE 13,6C   n    p>L >  1]`n
B260 01 4C B8 BB 20 BB B5 A0 00 B9 35 01 C9 30 90 04,D8    L8; ;5  95 I0
B270 C9 3A 90 01 60 A2 08 20 42 B6 84 E1 AD 1A 01 38,F3   I:  `"  B6 a-  8
B280 E5 E1 8D 1A 01 20 64 B3 A4 E1 20 02 B5 C0 50 B0,B4   ea     d3$a  5@P0
B290 05 A4 E1 20 FC B2 20 F1 B2 F0 27 AD 08 01 18 F8,AC    $a |2 q2p'-    x
B2A0 6D 0C 01 8D 08 01 AD 09 01 6D 0D 01 8D 09 01 D8,5D   m      -  m      X
B2B0 20 35 B6 20 BB B5 A0 00 84 E1 20 C5 B2 D0 C6 20,4A    56 ;5    a E2PF
B2C0 CA E3 4C 72 B0 A0 00 20 02 B5 B9 35 01 C9 2F D0,93   JcLr0     595 I/P
B2D0 03 D9 36 01 60 20 42 B6 C0 50 B0 09 A5 DF F0 05,60    Y6 ` B6@P0 %_p
B2E0 A2 00 4C 44 B4 60 20 D5 B2 A9 20 D9 35 01 D0 F0,E5   " LD4` U2) Y5 Pp
B2F0 60 18 AD 0C 01 6D 0D 01 8D 15 01 60 98 48 20 AD,42   ` -  m       ` H -

      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
B300 B4 AD 1A 01 38 65 D3 48 8A 65 D4 CD 03 01 90 0E,A8   4-  8eSH eTM
B310 F0 03 4C 3E B4 A8 68 CD 02 01 B0 F6 48 98 85 D4,98   p L>4(hM  0vH  T
B320 68 85 D3 20 B6 B4 4C 33 B3 20 E9 B4 20 E8 B4 A1,2E   h S 64L33 i4 h4!
B330 DF 81 E1 A5 DD C5 DF D0 F0 A5 DE C5 E0 D0 EA A0,D7   _ a%]E_Pp%^E`Pj
B340 00 AD 08 01 91 DD AD 09 01 C8 91 DD 68 AA C8 BD,7F    -    ]-  H ]h*H=
B350 35 01 E8 91 DD CC 1A 01 D0 F4 09 80 91 DD A2 00,4F   5 h ]L  Pt    ]"
B360 20 A0 B0 60 20 14 B2 D0 01 60 B0 FD 20 A4 B4 20,7B     0`  2P `0} $4
B370 A4 B3 A0 02 B1 DD D0 0F A5 DF 85 D3 A5 E0 85 D4,9B   $3  1]P %_ S%` T
B380 20 A0 B0 20 14 B2 60 A1 DD 81 DF 20 D7 B4 20 D6,D0     0  2`!] _ W4 V
B390 B4 88 D0 F3 A1 DD 81 DF 08 20 D7 B4 20 D6 B4 28,32   4 Ps!] _  W4 V4(
B3A0 10 F2 30 CE 20 D7 B4 20 D7 B4 A1 DD 10 F9 20 D7,06    r0N W4 W4!] y W
B3B0 B4 60 20 26 B6 B1 DD 8D 08 01 C8 B1 DD 8D 09 01,27   4` &61]    H1]
B3C0 C8 AD 0F 01 F0 2D B1 DD C9 3B F0 27 C9 20 F0 03,4E   H-  p-1]I;p'I p
B3D0 20 09 B4 20 F9 B3 A6 EA C9 3B F0 17 20 09 B4 20,8F     4 y3&jI;p   4
B3E0 F9 B3 C9 3B F0 0A 20 09 B4 20 F9 B3 C9 3B D0 03,B9   y3I;p   4 y3I;P
B3F0 20 02 B4 20 09 B4 4C F3 B3 B1 DD C8 C9 20 F0 F9,86     4  4Ls31]HI py

      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
B400 88 60 E8 8A 29 07 D0 FA 60 E0 50 B0 1B B1 DD 30,F3    `h ) Pz``P0 1]0
B410 0E C9 09 F0 1A 9D 35 01 C8 E8 C9 20 D0 EB 60 29,8D    I p  5 HhI Pk`)
B420 7F C9 09 F0 03 9D 35 01 8A A8 A2 00 68 68 60 20,C8   □I p  5   (" hh`
B430 02 B4 C8 D0 D4 E8 E8 E8 E8 E8 E8 E8 E8 E8 E8 E8,E2    4HPThhhhhhhhhhh
B440 E8 E8 E8 E8 E8 E8 E8 E8 E8 E8 E8 E8 E8 8A 48 A2,1E   hhhhhhhhhhhhh H"
B450 00 20 B7 E7 AD 13 01 10 06 8E 13 01 8D 12 01 A5,9A     7g-             %
B460 E3 85 CF 86 E3 A9 07 20 A4 E3 20 42 BF 20 CA E3,7F   c O c)  $c B? Jc
B470 A9 21 20 A4 E3 68 20 E2 E3 98 48 A0 00 20 1E B5,B0   )! $ch bc H     5
B480 68 A8 20 32 B6 A9 2F 20 A4 E3 AD 28 01 20 E2 E3,02   h( 26)/ $c-(  bc
B490 20 CA E3 AD 12 01 D0 05 A5 CF 85 E3 60 A9 FF 85,CD    Jc-  P %O c`)□
B4A0 DB 4C 58 B0 A5 DD 85 DF A5 DE 85 E0 60 A5 D3 85,27   [LX0%] _%^ ``%S
B4B0 DF A5 D4 85 E0 60 A5 D3 85 E1 A5 D4 85 E2 60 AD,0F   _%T ``%S a%T b`-
B4C0 00 01 85 DD AD 01 01 85 DE 60 AD 04 01 85 DD AD,A5      ]-   ^`-   ]-
B4D0 05 01 85 DE 60 E8 E8 E8 E8 E8 E8 E8 8A 0A AA F6,FA      ^`hhhhhhh  *v
B4E0 D3 D0 02 F6 D4 A2 00 60 E8 E8 E8 E8 E8 E8 E8 8A,4D   SP vT" `hhhhhhh
B4F0 0A AA D6 D3 B5 D3 C9 FF D0 02 D6 D4 A2 00 60 20,98    *VS5SI□P VT" `
```

```
     00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
B500 10 B5 B9 35 01 C8 C0 50 B0 05 C9 20 F0 F4 88 60,8E   595 H@P0 I pt `
B510 B9 35 01 C8 C0 50 B0 05 C9 20 D0 F4 88 60 B9 2A,82  95 H@P0 I Pt `9*
B520 B5 D0 01 60 20 A4 E3 C8 D0 F4 20 41 54 20 4C 49,05  5P ` $cHPt AT LI
B530 4E 45 00 0D 0A 0A 0A 4C 41 42 45 4C 20 46 49 4C,1E  NE     LABEL FIL
B540 45 3A 20 20 5B 20 2F 20 3D 20 45 58 54 45 52 4E,DA  E:  [ / = EXTERN
B550 41 4C 20 5D 0D 0A 0A 00 52 45 41 44 59 20 46 4F,2F  AL ]    READY FO
B560 52 20 50 41 53 53 20 32 0D 0A 00 50 41 47 45 20,7E  R PASS 2   PAGE
B570 00 0D 0A 52 41 45 20 56 31 2E 30 0D 0A 43 4F 50,6B     RAE V1.0  COP
B580 59 52 49 47 48 54 20 31 39 37 39 20 53 59 4E 45,9B  YRIGHT 1979 SYNE
B590 52 54 45 4B 20 53 59 53 54 45 4D 53 20 43 4F 52,2D  RTEK SYSTEMS COR
B5A0 50 2E 20 0D 0A 0D 0A 00 A2 00 8A 48 BD 35 01 20,80  P.      "  H=5
B5B0 A4 E3 68 AA E8 88 10 F2 A2 00 60 86 E3 A9 3E 20,FD  $ch*h  r" ` c)>
B5C0 A4 E3 20 26 B6 20 BD E3 C9 0D D0 08 C8 8C 1A 01,5D  $c &6 =cI P H
B5D0 20 CA E3 60 C9 08 F0 09 C9 7F D0 1E A9 5C 20 A4,53   Jc`I p I□P )\ $
B5E0 E3 88 30 08 A9 20 99 35 01 4C C5 B5 20 CA E3 AD,CE  c 0 )  5 LE5 Jc-
B5F0 15 01 F0 C7 20 35 B6 4C BD B5 C9 18 F0 EE 99 35,F1    pG 56L=5I pn 5

     00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
B600 01 C9 09 F0 10 C8 C0 46 D0 05 A9 07 20 A4 E3 C0,7E   I p H@FP )  $c@
B610 50 B0 B9 D0 B0 98 48 20 DC E3 C8 98 29 07 D0 F7,CD  P09P0 H \cH ) Pw
B620 68 A8 C8 4C 0F B6 A0 55 A9 20 99 35 01 88 10 FA,D5  h(HL 6 U)  5   z
B630 C8 60 20 DC E3 AD 09 01 20 E2 E3 AD 08 01 20 E2,30  H` \c-   bc-   b
B640 E3 60 A9 00 9D 00 01 9D 01 01 85 DF 85 E0 20 7E,C0  c`)        _ ` ~
B650 B6 D0 03 A2 00 60 48 A5 DF F0 09 AD 11 01 F0 04,C3  6P " `H%_p -  p
B660 A2 00 68 60 C8 98 48 A0 04 1E 00 01 3E 01 01 88,60  " h`H H     >
B670 D0 F7 68 A8 68 1D 00 01 9D 00 01 4C 4E B6 20 A0,6B  Pwh(h      LN6
B680 B6 C9 30 90 18 C9 3A 90 0F C9 41 90 10 C9 47 B0,CE  6I0  I:  IA  IG0
B690 0C 29 0F 18 69 09 E6 DF 29 0F E6 E0 60 A9 00 60,C8   )  i f_) f``) `
B6A0 B9 35 01 C9 61 90 06 C9 7B B0 02 29 DF 60 A9 00,7E  95 Ia  I{0 )_`)
B6B0 F0 06 A9 01 D0 02 A9 FF 85 E1 84 E2 A4 E2 A5 E1,70  p ) P )□ a b$b%a
B6C0 F0 0E 10 06 BD C7 B8 D0 0D 60 BD 41 B8 D0 07 60,EA  p   =G8P `=A8P `
B6D0 BD 41 B7 D0 01 60 8D 1A 01 20 A0 B6 CD 1A 01 08,DE  =A7P `     6M
B6E0 98 38 E5 E2 C8 E8 28 F0 22 C9 01 F0 05 C9 02 F0,D9   8ebHh(p"I p I p
B6F0 02 E8 E8 E8 A5 E1 30 05 D0 C2 4C BC B6 BD C6 B8,D9   hhh%a0 PBL<6=F8

     00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
B700 29 0F 86 E0 38 65 E0 AA 4C BC B6 48 A5 E1 D0 07,01  )  `8e`*L<6H%aP
B710 68 C9 01 D0 A9 F0 05 68 C9 02 D0 A2 A5 E1 F0 03,BF  hI P)p hI P"%ap
B720 86 D0 60 BD 41 B7 85 E1 BD 42 B7 85 E2 68 68 E0,5D   P`=A7 a=B7 bhh`
B730 85 B0 06 20 FF B4 4C 3C B7 20 02 B5 A2 00 6C E1,70   0  □4L<7  5" la
B740 00 42 52 A6 B0 43 4C 52 B0 50 55 24 E5 46 4F C2,F0   BR&0CLR0PU$eFOB
B750 B0 50 52 4C B1 41 55 B4 B0 41 53 FC B0 50 41 B7,C1  0PRL1AU40AS|0PA7
B760 E2 52 55 35 B1 4D 41 EC B0 4F 55 A7 BB 4F 4E 50,4D  bRU51MAl0OU';ONP
B770 E3 4F 46 33 E3 48 41 12 E9 47 45 BC B0 4C 41 00,E4  cOF3cHA iGE<0LA
B780 BF 45 44 1C E6 4E 55 FD E3 44 45 90 EA 46 49 16,59  ?ED fNU}cDE jFI
B790 E6 4D 4F 87 EA 43 4F 44 E9 53 45 BD EA 55 53 F7,E9  fMO jCODiSE=jUSw
B7A0 EF 44 55 4E EB 45 4E 84 EB 4C 4F 95 EB 44 43 A6,F4  oDUNkEN kLO kDC&
B7B0 EB 00 2E 53 A6 38 6E 53 A6 60 00 00 00 00 00 00,05  k .S&8nS&`
B7C0 00 00 00 00 00 00 53 49 46 BA 42 41 5A BA 45 4E,CB        SIF:BAZ:EN
B7D0 63 BB 42 59 C4 BA 53 45 4F BA 44 49 FE BA 4C 53,87  c;BYD:SEO:DI~:LS
B7E0 3B BA 4C 43 40 BA 4D 43 9A BA 4F 43 31 BA 4F 53,08  ;:LC@:MC :OC1:OS
B7F0 2C BA 43 45 27 BA 43 54 36 BA 52 53 22 BA 44 45,E8  ,:CE':CT6:RS":DE
```

```
     00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
B800 07 BB 52 43 37 E4 44 53 B0 B9 45 53 41 EE 45 43,A9   ;RC7dDS09ESAnEC
B810 46 EE 45 4A 0B BA 4D 44 8D EE 4D 45 33 EF 00 00,F1  FnEJ :MD nME3o
B820 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00,F1
B830 00 00 00 00 00 00 00 41 20 CA BD 00 29 20 0A BE,EA         A J= )  >
B840 00 54 41 58 AA 54 41 59 A8 54 53 58 BA 54 58 41,BD   TAX*TAY(TSX:TXA
B850 8A 54 58 53 9A 54 59 41 98 43 4C 43 18 43 4C 44,23   TXS TYA CLC CLD
B860 D8 43 4C 49 58 43 4C 56 B8 53 45 43 38 53 45 44,B7  XCLIXCLV8SEC8SED
B870 F8 53 45 49 78 4E 4F 50 EA 52 54 49 40 52 54 53,07  xSEIxNOPjRTI@RTS
B880 60 44 45 58 CA 44 45 59 88 49 4E 58 E8 49 4E 59,43  `DEXJDEY INXhINY
B890 C8 50 48 41 48 50 48 50 08 50 4C 41 68 50 4C 50,4D  HPHAHPHP PLAhPLP
B8A0 28 42 52 4B 00 00 42 43 43 90 42 43 53 B0 42 45,BB  (BRK  BCC BCS0BE
B8B0 51 F0 42 4D 49 30 42 4E 45 D0 42 50 4C 10 42 56,2F  QpBMI0BNEPBPL BV
B8C0 43 50 42 56 53 70 00 52 4F 52 C5 C1 6E 7E 66 76,5E  CPBVSp ROREAn~fv
B8D0 6A 41 44 43 E8 DA 6D 7D 79 65 75 71 61 69 41 4E,59  jADChZm}yeuqaiAN
B8E0 44 E8 DA 2D 3D 39 25 35 31 21 29 41 53 4C C5 C1,3D  DhZ-=9%51!)ASLEA
B8F0 0E 1E 06 16 0A 42 49 54 82 80 2C 24 43 4D 50 E8,88       BIT  ,$CMPh

     00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
B900 DA CD DD D9 C5 D5 D1 C1 C9 43 50 58 83 82 EC E4,9A  ZM]YEUQAICPX  ld
B910 E0 43 50 59 83 82 CC C4 C0 44 45 43 C4 C0 CE DE,B7  `CPY  LD@DECD@N^
B920 C6 D6 45 4F 52 E8 DA 4D 5D 59 45 55 51 41 49 49,BC  FVEORhZM]YEUQAII
B930 4E 43 C4 C0 EE FE E6 F6 4A 4D 50 92 00 4C 6C 4C,16  NCD@n~fvJMP  LlL
B940 44 41 E8 DA AD BD B9 A5 B5 B1 A1 A9 4C 44 58 A5,62  DAhZ-=9%51!)LDX%
B950 A2 AE BE A6 B6 A2 4C 44 59 C5 C2 AC BC A4 B4 A0,3E  ".>&6"LDYEB,<$4
B960 4C 53 52 C4 C1 4E 5E 46 56 4A 53 52 81 00 20 4F,DB  LSRDAN^FVJSR   O
B970 52 41 E8 DA 0D 1D 19 05 15 11 01 09 52 4F 4C C5,5A  RAhZ        ROLE
B980 C1 2E 3E 26 36 2A 53 42 43 E8 DA ED FD F9 E5 F5,64  A.>&6*SBChZm}yeu
B990 F1 E1 E9 53 54 41 E7 D8 8D 9D 99 85 95 91 81 53,08  qaiSTAgX       S
B9A0 54 58 83 A0 8E 86 96 53 54 59 83 C0 8C 84 94 00,68  TX     STY @
B9B0 AD 17 01 F0 0E AD 13 01 C9 01 D0 07 20 BF BE A9,D3  -  p -  I P  ?>)
B9C0 05 85 E7 20 4A E2 AD 1A 01 F0 06 8C 12 01 4C 49,82    g Jb-  p    LI
B9D0 B4 A5 D1 18 65 D7 85 D7 A5 D2 65 D8 85 D8 A5 D1,E3  4%Q eW W%ReX X%Q
B9E0 18 65 D9 85 D9 A5 D2 65 DA 85 DA AD 13 01 10 18,95   eY Y%ReZ Z-
B9F0 A9 7F EE 11 01 20 70 BE A5 D1 EE 11 01 20 70 BE,CF  )□n   p>%Qn   p>

     00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
BA00 A5 D2 EE 11 01 20 70 BE 4C FD BB AD 13 01 F0 0F,58  %Rn   p>L};-  p
BA10 30 0D AD 17 01 F0 08 AD 1F 01 F0 03 20 D1 E8 4C,37  0 -  p -  p  QhL
BA20 FD BB A9 5F 4C 9B BD 8E 12 01 F0 17 8C 16 01 F0,D6  };)_L =   p    p
BA30 12 8E 16 01 F0 0D 8C 14 01 F0 08 8C 17 01 F0 03,BA      p    p    p
BA40 8E 17 01 4C FD BB 20 4A E2 EE 1E 01 4C 49 B2 20,24     L}; Jbn  LI2
BA50 4A E2 A2 00 8E 1E 01 4C 49 B2 A5 DD 48 A5 DE 48,7B  Jb"    LI2%]H%^H
BA60 20 4A E2 AD 1A 01 F0 06 8C 12 01 4C 49 B4 68 85,5A   Jb-  p    LI4h
BA70 DE 68 85 DD A5 D1 85 D7 85 D9 A5 D2 85 D8 85 DA,65  ^h ]%Q W Y%R X Z
BA80 AD 35 01 C9 20 F0 08 AD 13 01 D0 03 20 57 BB AD,9C  -5 I p -  P  W;-
BA90 13 01 10 03 20 ED BE 4C FD BB AD 12 01 48 8C 12,38       m>L};-  H
BAA0 01 20 4A E2 68 8D 12 01 A5 D1 85 D9 A5 D2 85 DA,37    Jbh   %Q Y%R Z
BAB0 4C FD BB 20 4A E2 A5 D1 20 EB BA 20 02 B5 C0 50,A9  L}; Jb%Q k:  5@P
BAC0 B0 D5 90 0A C0 50 90 06 20 43 B4 4C FD BB C9 3B,8D  0U  @P   C4L};I;
BAD0 F0 C5 C9 27 D0 DD C8 C0 50 B0 ED B9 35 01 C9 27,33  pEI'P]H@P0m95 I'
BAE0 D0 03 C8 D0 D6 20 EB BA B8 50 EB 48 AD 13 01 10,45  P HPV k:8PkH-
BAF0 08 A9 3F EE 11 01 20 70 BE 68 20 70 BE 60 20 2F,E8   )?n   p>h p>` /
```

```
     00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
BB00 BB 20 57 BB 4C FD BB 20 2F BB D0 1D A0 01 C8 B1,EA  ; W;L}; /;P   H1
BB10 CE 10 FB B1 CE C8 91 CE 88 88 C0 01 D0 F5 C8 A9,70  N {1NH N   @ PuH)
BB20 2F 91 CE 20 DB B4 20 5F E0 20 57 BB 4C FD BB 20,62  / N {4 _` W;L};
BB30 4A E2 AD 35 01 C9 20 D0 06 20 48 B4 4C FD BB A0,F0  Jb-5 I P  H4L};
BB40 00 20 1A E1 A5 DD 85 CE A5 DE 85 CF AD 13 01 F0,68    a%] N%^ O-  p
BB50 05 8E 1E 01 A9 FF 60 A0 00 A5 D1 91 CE C8 A5 D2,D6      )□`  %Q NH%R
BB60 91 CE 60 20 47 BF A5 BC D0 0A A5 BF D0 0A A5 BB,94   N` G?%<P %?P %;
BB70 D0 0A F0 0D A2 21 D0 06 A2 22 D0 02 A2 23 20 4D,CC  P p "!P ""P "# M
BB80 B4 AD 13 01 F0 0E 30 06 AD 17 01 4C FE BE 20 ED,4F  4-  p 0 -  L~> m
BB90 BE 4C 58 B0 EE 13 01 AD 14 01 F0 1C 20 9D E3 A0,71  >LX0n  -  p   c
BBA0 2E 20 1E B5 4C 58 B0 8E 22 01 20 9B B1 8E 11 01,A3  .  5LX0 "    1
BBB0 A2 FF 8E 13 01 8E 0A 01 AD 2F 01 8D 08 01 AD 30,CF  "□        -/     -0
BBC0 01 8D 09 01 20 14 B2 A2 00 86 E6 86 DC 86 DB A2,C0           2"  f \ ["
BBD0 FF 9A E8 8E 16 01 86 C4 86 C5 86 C2 86 C3 86 BB,4D  □ h     D E B C ;
BBE0 86 BC 86 BF 86 C1 A9 00 85 D7 85 D9 A9 02 85 D8,86   < ? A)  W Y)   X
BBF0 85 DA AD 13 01 10 11 20 F0 BE 4C 08 BC A2 00 68,AF   Z-      p>L <" h

     00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
BC00 85 DD 68 85 DE 20 A4 B3 A5 DE 48 A5 DD 48 8E 1E,94   ]h ^ $3%^H%]H
BC10 01 20 59 B2 D0 54 A5 BB F0 05 A2 27 20 4D B4 E6,09    Y2PT%;p "' M4f
BC20 C4 D0 09 E6 C5 D0 05 A2 2F 20 4D B4 AD 14 01 D0,AA  DP fEP "/ M4-  P
BC30 08 A9 01 8D 12 01 4C 46 B4 20 47 BF 20 CA E3 20,55   )     LF4 G? Jc
BC40 CA E3 20 CA E3 A5 BA 8D 10 01 20 71 E4 A9 00 8D,77  Jc Jc%:     qd)
BC50 11 01 AD 2F 01 8D 08 01 AD 30 01 8D 09 01 20 14,A5    -/     -0
BC60 B2 F0 CE A2 FF 9A E8 4C 08 BC A2 00 AD 17 01 F0,9F  2pN"□ hL <" -  p
BC70 15 20 47 BF AD 13 01 C9 01 D0 0B A5 BC F0 04 A5,3A    G?-  I P %<p %
BC80 C1 F0 03 20 CA E3 20 B2 B3 84 E9 E6 E6 A0 00 B9,D2  Ap  Jc 23 iff  9
BC90 35 01 C9 3B F0 03 20 FF B4 86 BD A5 BB F0 08 A2,0F  5 I;p  □4 =%;p "
BCA0 24 20 44 ED 4C FD BB A5 BF F0 08 A2 19 20 44 ED,F0  $ DmL};%?p "  Dm
BCB0 4C FD BB A0 00 B9 35 01 C9 3B D0 03 4C FD BB C9,27  L};  95 I;P L};I
BCC0 20 F0 08 AD 13 01 D0 03 20 79 BF 20 FF B4 B9 35,EC   p -  P  y? □495
BCD0 01 C9 3B F0 E7 C0 50 B0 E3 C9 2E D0 18 C8 B9 37,02   I;pg@P0cI.P H97
BCE0 01 C9 20 F0 06 20 4A B4 4C FD BB A2 85 20 AE B6,AF   I p  J4L};"  .6
BCF0 A2 00 4C E5 BC B9 38 01 C9 20 D0 50 20 B2 B6 F0,B1  " Le<98 I PP 26p

     00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
BD00 09 BD 41 B8 20 70 BE 4C FD BB A2 65 20 B2 B6 F0,41   =A8 p>L};"e 26p
BD10 34 BD 41 B8 20 70 BE 20 FF B4 AD 13 01 F0 23 20,40  4=A8 p> □4-  p#
BD20 4A E2 18 A5 D1 E5 D7 48 A5 D2 E5 D8 D0 0B 68 C9,3E  Jb %QeWH%ReXP hI
BD30 80 90 0F 20 4C B4 4C FD BB C9 FF D0 F6 68 C9 80,C0      L4L};I□PvhI
BD40 90 F1 4C 04 BD A2 00 20 B6 B6 D0 03 4C 8B EC 20,32   qL ="  66P L l
BD50 02 B5 B9 35 01 C9 23 D0 79 A2 0B 20 3D BE A2 55,CC   595 I#Py"  =>"U
BD60 20 44 ED C8 B9 35 01 C9 27 D0 06 B9 36 01 4C 04,DA   DmH95 I'P 96 L
BD70 BD A2 00 20 4A E2 A5 D1 4C 04 BD 20 4A E2 A5 D1,CA  ="  Jb%QL = Jb%Q
BD80 20 70 BE AD 1E 01 F0 26 A9 1F D0 0F 20 4A E2 A5,92   p>-  p&) P  Jb%
BD90 D2 20 70 BE AD 1E 01 F0 15 A9 2F 48 AD 13 01 10,74  R p>-  p )/H-
BDA0 10 68 EE 11 01 48 20 70 BE 68 C9 2F F0 07 4C FD,22   hn  H p>hI/p L}
BDB0 BB 68 4C FD BB A5 D1 EE 11 01 4C 04 BD 20 70 BE,1A  ;hL};%Qn  L = p>
BDC0 AD 1E 01 D0 E9 A9 0F 4C 9B BD A2 0C 20 3D BE 4C,10  -  Pi) L ="  =>L
BDD0 FD BB A2 F6 20 AE B6 AA B9 35 01 C9 2A D0 0A C8,12  };"v .6*95 I*P H
BDE0 20 C1 E0 F0 3A A2 3F D0 1A C9 28 D0 0F C8 20 C1,41   A`p:"?P I(P H A
BDF0 E0 A2 FB 20 AE B6 A2 34 D0 09 D0 07 20 C1 E0 F0,79  `"{ .6"4P P  A`p
```

```
     00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
BE00 0C A2 4A 20 44 ED AA 4C C8 BA E8 E8 E8 E8 20 3D,37   "J Dm*LH:hhhh =
BE10 BE A5 D1 20 70 BE A5 D2 4C BD BD E8 E8 E8 E8 E8,7E  >%Q p>%RL==hhhhh
BE20 E8 E8 E8 E8 20 3D BE A5 D1 20 70 BE A5 D2 F0 0A,6E  hhhh =>%Q p>%Rp
BE30 A2 00 AD 13 01 F0 03 20 4D B4 4C FD BB 86 DD A6,F2  " -  p  M4L}; ]&
BE40 D0 BD C8 B8 85 DE A9 04 85 DF BD C7 B8 C6 DF 30,84  P=H8 ^)   _=G8F_0
BE50 10 0A 90 01 E8 C6 DD D0 F4 B0 12 A2 00 20 42 B4,F8      hF]Pt0 "  B4
BE60 60 A5 DE 0A 90 01 E8 C6 DD D0 F8 90 EE BD C8 B8,84  `%^   hF]Px n=H8
BE70 48 A2 00 AD 13 01 F0 57 30 59 AD 17 01 F0 31 A5,8A  H" -  pW0Y-  p1%
BE80 BC F0 04 A5 C1 F0 29 A5 E7 D0 07 A9 05 85 E7 20,56  <p %Ap)%gP )  g
BE90 BF BE A5 E7 C9 0E D0 09 20 47 BF 20 CA E3 4C 8B,D9  ?>%gI P  G? JcL
BEA0 BE 20 DC E3 A5 E7 18 69 03 85 E7 68 48 20 E2 E3,87  > \c%g i  ghH bc
BEB0 AD 16 01 F0 1A 68 81 D9 20 DA B4 20 D9 B4 60 A5,77  -  p h Y Z4 Y4`%
BEC0 D8 20 E2 E3 A5 D7 20 E2 E3 A9 2D 20 A4 E3 60 68,DA  X bc%W bc)- $c`h
BED0 B8 50 E5 68 20 3C B2 AD 11 01 D0 06 20 DA B4 4C,CC  8Peh <2-  P  Z4L
BEE0 E5 BE CE 11 01 AD 22 01 C9 FF B0 01 60 20 74 B1,3D  e>N  -" I□0 ` t1
BEF0 8E 22 01 A5 D7 20 3C B2 A5 D8 20 3C B2 60 F0 03,56   " %W <2%X <2`p

     00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
BF00 20 62 E2 20 EF B1 20 0F BF 20 19 BF 4C 5E B0 A5,5F   bb o1  ?  ?L^0%
BF10 DC 20 E2 E3 A5 DB 4C E2 E3 A9 2C 20 A4 E3 A5 D8,AA  \ bc%[Lbc), $c%X
BF20 20 E2 E3 A5 D7 20 E2 E3 A9 2C 20 A4 E3 A5 DA 20,0B   bc%W bc), $c%Z
BF30 E2 E3 A5 D9 4C E2 E3 AD 04 01 85 D5 AD 05 01 85,A3  bc%YLbc-   U-
BF40 D6 60 98 48 4C 55 BF 98 48 AD 13 01 C9 01 D0 22,76  V` HLU? H-  I P"
BF50 AD 17 01 F0 1D A5 E6 F0 19 A4 E7 A5 BC D0 13 C8,73  -  p %fp $g%<P H
BF60 20 DC E3 C0 10 90 F8 A4 E9 20 35 B6 20 DC E3 20,41   \c@  x$i 56 \c
BF70 A8 B5 86 E7 68 A8 86 E6 60 A5 D5 85 B9 A5 D6 85,9F  (5 gh( f`%U 9%V
BF80 BA A5 BC F0 12 20 66 E0 D0 02 18 60 20 76 E0 D0,B2  :%<p  f`P  ` v`P
BF90 24 20 89 E0 4C A2 BF 20 66 E0 D0 19 20 47 BF 20,A1  $  `L"? f`P  G?
BFA0 7E E0 84 E4 20 94 E0 A6 E4 20 A0 E0 20 A0 E0 20,E5  ~` d  `&d  `   `
BFB0 A0 E0 4C BC BF 84 E4 20 94 E0 A6 E4 BD 35 01 C9,6E   `L<? d  `&d=5 I
BFC0 40 90 1D 20 A0 E0 BD 35 01 C9 20 F0 1D A5 BC F0,35  @     `=5 I p %<p
BFD0 07 BD 35 01 C9 29 F0 12 BD 35 01 20 54 E2 90 E3,DF   =5 I)p =5  Tb c
BFE0 20 5F E0 A2 00 20 41 B4 38 60 4C 00 E0 A9 60 85,47   _`"  A48`L `)`
BFF0 B6 AD 00 A0 29 7F 8D 00 A0 4C 12 B0 00 00 00 00,2D  6-  )□   L 0
```

Memory block $B000-$BFFF checksum: 672D

APPENDIX G

SOFTWARE LISTING

Enter range limits for hex dump: E000-EFFF

```
     00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
E000 B1 D5 09 80 91 D5 38 98 65 D5 85 D5 A2 00 8A 65,6A  1U   U8 eU U"  e
E010 D6 85 D6 CD 07 01 F0 2E B0 33 20 5F E0 A4 E4 20,78  V VM  p.03 _`$d
E020 1A E1 A5 DE 48 A5 DD 48 20 A8 E0 20 59 B2 F0 0C,D7   a%^H%]H (` Y2p
E030 68 85 DD 68 85 DE 20 47 B4 4C 50 E0 68 85 DD 68,35  h ]h ^ G4LP`h ]h
E040 85 DE A4 E4 18 60 A5 D5 CD 06 01 90 CD 20 3F B4,56   ^$d `%UM   M ?4
E050 A5 B9 85 D5 A5 BA 85 D6 20 5F E0 A4 E4 38 60 A9,F0  %9 U%: V _`$d8`)
E060 00 A0 02 91 D5 60 B9 35 01 C9 21 D0 08 D9 36 01,19      U`95 I!P Y6
E070 D0 03 D9 37 01 60 B9 35 01 C9 2E 4C 6B E0 A5 C4,43  P Y7 `95 I.Lk`%D
E080 99 36 01 A5 C5 99 37 01 60 A5 C2 99 36 01 A5 C3,4D   6 %E 7 `%B 6 %C
E090 99 37 01 60 A0 00 A5 D7 91 D5 A5 D8 C8 91 D5 60,0B   7 `  %W U%XH U`
E0A0 BD 35 01 C8 91 D5 E8 60 A0 02 B1 DD C9 2E F0 04,8F  =5 H Uh`  1]I.p
E0B0 C9 21 D0 09 20 D7 B4 20 D7 B4 20 D7 B4 20 A4 B3,CA  I!P  W4 W4 W4 $3
E0C0 60 8E 1E 01 C0 50 90 04 20 43 B4 60 86 D1 86 D2,A1  `   @P   C4` Q R
E0D0 8E 1A 01 4C E5 E0 B9 35 01 C9 2B F0 07 C9 2D F0,1B     Le`95 I+p I-p
E0E0 1E C9 20 60 C8 20 1A E1 B0 03 EE 1A 01 18 AD 18,FE   I `H  a0 n   -
E0F0 01 65 D1 85 D1 AD 19 01 65 D2 85 D2 4C D6 E0 C8,AA   eQ Q-  eR RLV`H

     00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
E100 20 1A E1 B0 03 EE 1A 01 38 A5 D1 ED 18 01 85 D1,8B    a0 n  8%Qm   Q
E110 A5 D2 ED 19 01 85 D2 4C D6 E0 20 CA B4 86 C0 84,CA  %Rm   RLV` J4 @
E120 E1 8E 18 01 8E 19 01 B9 35 01 C9 25 D0 1B C8 B9,43  a      95 I%P H9
E130 35 01 C9 30 F0 06 C9 31 F0 05 38 60 18 90 01 38,D0  5 I0p I1p 8`   8
E140 2E 18 01 2E 19 01 4C 2E E1 20 9A E1 B0 06 B9 35,F9  .  .  L.a  a0 95
E150 01 4C A8 E1 C8 B9 35 01 20 9A E1 B0 F7 98 48 88,30   L(aH95   a0w H
E160 B9 35 01 84 CE 20 9A E1 90 20 F0 17 A8 BD 90 E1,99  95  N  a  p (= a
E170 18 6D 18 01 8D 18 01 BD 95 E1 6D 19 01 8D 19 01,3E   m     = am
E180 88 D0 EA A4 CE E8 E0 05 D0 D5 68 A8 A2 00 38 60,AE   Pj$Nh` PUh(" 8`
E190 01 0A 64 E8 10 00 00 00 03 27 C9 30 90 04 C9 3A,CF    dh     'I0  I:
E1A0 90 02 18 60 29 0F 38 60 C9 24 F0 4A C9 3D F0 39,FF     `) 8`I$pJI=p9
E1B0 A6 E1 A0 02 B1 DD C9 2F D0 01 C8 B1 DD F0 03 4C,14  &a  1]I/P H1]p L
E1C0 BD E2 A2 00 A4 E1 C8 B9 35 01 20 54 E2 90 F7 C9,37  =b" $aH95  Tb wI
E1D0 21 F0 04 C9 2E D0 08 A5 BD D0 EB A5 BC D0 E7 AD,FD  !p I.P %=Pk%<Pg-
E1E0 13 01 F0 03 20 45 B4 18 60 A5 D7 8D 18 01 A5 D8,34    p  E4 `%W   %X
E1F0 8D 19 01 C8 38 60 A2 18 C8 20 42 B6 A5 E0 D0 48,72     H8`" H B6%`PH
```

```
     00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
E200 20 40 B4 18 60 B1 DD 08 29 7F DD 35 01 D0 11 E8,18   @4 `1] )□]5 P h
E210 C8 28 10 F1 8A A8 B9 35 01 20 54 E2 B0 0B 90 01,CC  H( q (95  Tb0
E220 28 A2 00 20 A8 E0 B8 50 87 98 AA A0 00 B1 DD 8D,CA  ("  (`8P  *  1]
E230 18 01 C8 B1 DD 8D 19 01 C8 B1 DD C9 2F F0 03 EE,0F    H1]   H1]I/p n
E240 1E 01 8A A8 A2 00 E6 C0 38 60 20 C1 E0 D0 01 60,32      (" f@8` A`P `
E250 20 43 B4 60 C9 2E 90 08 C9 3D F0 04 C9 7B 90 01,07   C4`I.  I=p I{
E260 38 60 A0 09 20 1E B5 20 CA B4 86 CF 20 CA E3 A0,9B  8`     5 J4 O Jc
E270 02 B1 DD D0 04 20 CA E3 60 20 B2 B3 AD 35 01 C9,5D   1]P  Jc` 23-5 I
E280 21 F0 04 C9 2E D0 06 20 A8 E0 4C 6F E2 98 18 65,99  !p I.P  (`Lob  e
E290 CF 85 CF 20 A8 B5 A9 3D 20 A4 E3 20 35 B6 20 A8,99  O O (5)= $c 56 (
E2A0 E0 A4 CF A4 CF 20 DC E3 C8 84 CF C0 12 F0 C0 C0,9B  `$O$O \cH O@ p@@
E2B0 24 F0 BC B0 B5 D0 EC 8E 11 01 4C 5E B2 B1 DD C9,DF  $p<05Pl   L^21]I
E2C0 2E D0 1B A5 BC F0 3E C8 B1 DD C5 C2 D0 37 C8 B1,E4  .P %<p>H1]EBP7H1
E2D0 DD C5 C3 D0 30 BD 35 01 C9 2E D0 29 F0 2A C9 21,30  ]ECP0=5 I.P)p*I!
E2E0 D0 14 C8 B1 DD C5 C4 D0 1C C8 B1 DD C5 C5 D0 15,A4  P H1]EDP H1]EEP
E2F0 A5 BD F0 14 D0 15 A5 BD D0 0B BD 35 01 C9 2E F0,06  %=p P %=P =5 I.p

     00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
E300 04 C9 21 D0 07 4C 21 E2 E8 E8 E8 C8 4C 05 E2 AD,7A    I!P L!bhhhHL b-
E310 0C A0 29 DF 8D 0C A0 60 AD 00 A0 09 80 8D 00 A0,CA     )_    `-
E320 60 AD 0C A0 09 20 8D 0C A0 60 AD 00 A0 29 7F 8D,C7  `-  ¯       `-  )□
E330 00 A0 60 A2 08 20 E6 B2 AD 08 01 D0 06 20 0F E3,C7     `"  f2-  P   c
E340 4C 60 B0 C9 01 F0 03 4C 37 B4 20 18 E3 4C 60 B0,8E  L`0I p L74  cL`0
E350 A2 08 20 E6 B2 AD 08 01 D0 06 20 21 E3 4C 60 B0,FC  "  f2-  P  !cL`0
E360 C9 01 D0 E3 20 2A E3 4C 60 B0 20 6D E3 20 70 E3,E5  I Pc *cL`0 mc pc
E370 20 73 E3 A2 60 8E 2D 01 8E 2E 01 EE 2D 01 D0 FB,BD   sc"` -  . n- P{
E380 EE 2E 01 D0 F6 A2 00 60 20 2A E3 20 73 E3 60 20,C5  n. Pv" ` *c sc`
E390 21 E3 20 6A E3 60 20 73 E3 20 18 E3 60 20 73 E3,FD  !c jc` sc  c` sc
E3A0 20 0F E3 60 48 84 E4 86 E5 AD 1F 01 F0 05 68 48,FC     c`H d e-  p hH
E3B0 20 B4 E8 68 20 AF EB A4 E4 A6 E5 D8 60 84 E4 86,13   4hh /k$d&eX` d
E3C0 E5 20 10 EC A4 E4 A6 E5 D8 60 20 D3 E3 A9 0A 20,08  e  l$d&eX` Sc)
E3D0 A4 E3 60 A9 0D 20 A4 E3 60 20 DC E3 A9 20 20 A4,18  $c`)  $c` \c)  $
E3E0 E3 60 48 4A 4A 4A 4A 20 EF E3 68 20 EF E3 60 29,A0  c`HJJJJ och oc`)
E3F0 0F 09 30 C9 3A 90 02 69 06 20 A4 E3 60 20 03 E4,FA     0I:  i  $c`  d

     00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
E400 4C 5E B0 20 FC B1 AD 0B 01 C9 FF D0 03 4C 3C B4,B1  L^0 |1-  I□P L<4
E410 A0 00 18 F8 08 C0 02 D0 0F 20 A4 B3 28 90 03 4C,88     x @ P  $3(  L
E420 3D B4 B1 DD D0 EA D8 60 28 B9 0A 01 79 08 01 99,00  =41]PjX`(9  y
E430 08 01 91 DD C8 D0 DD A9 6F 4C 9B BD A2 08 20 E6,58     ]HP])oL ="  f
E440 B2 84 CE 20 14 B2 A4 CE 60 20 67 E4 AD 28 01 C9,1E  2 N  2$N` gd-( I
E450 EE F0 0E AD 23 01 F0 03 4C 5E B0 20 89 E4 4C 4C,4D  np -# p L^0  dLL
E460 E4 20 A0 B0 4C 5E B0 20 9B B1 20 A0 B6 C9 20 D0,96  d  0L^0  1  6I P
E470 06 20 96 B0 18 90 04 C9 41 D0 0B A5 D3 85 DD A5,12     0    IAP %S ]%
E480 D4 85 DE 18 90 03 20 3C E4 20 88 E3 20 11 E5 8D,62  T ^     <d  c  e
E490 23 01 20 5D EF D0 69 A5 DD 8D 24 01 A5 DE 8D 25,94  #  ]oPi%] $ %^ %
E4A0 01 38 AD 2B 01 ED 29 01 48 AD 2C 01 ED 2A 01 AA,A1   8-+ m) H-, m* *
E4B0 68 85 D1 18 65 DD 8D 26 01 8A 85 D2 65 DE 8D 27,45  h Q e] &    Re^ '
E4C0 01 A9 00 8D 23 01 AD 10 01 F0 05 CD 28 01 D0 1F,38   )  # -  p M( P
E4D0 EE 23 01 AD 27 01 CD 03 01 90 14 D0 08 AD 26 01,40  n# -' M     P -&
E4E0 CD 02 01 90 0A A9 01 8D 12 01 A2 00 4C 3E B4 20,F4  M     )      " L>4
E4F0 5D EF D0 0C A2 00 20 97 E5 20 96 E3 20 AA E5 60,02  ]oP "    e  c *e`
```

```
     00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
E500 A2 00 AD 23 01 F0 03 8D 12 01 20 A0 B0 20 36 B4,82  " -# p      0 64
E510 60 A9 28 8D 24 01 A9 2D 8D 26 01 A9 01 8D 25 01,4C  `)( $ )- & )  %
E520 8D 27 01 60 20 72 EB 20 A0 B6 C9 20 F0 48 C9 58,96   ' ` rk  6I pHIX
E530 D0 0E A9 EE 8D 10 01 8D 09 01 20 41 E4 C8 D0 0A,27  P )n        AdHP
E540 20 9B B1 C0 50 B0 2F 20 3C E4 A5 DD 8D 29 01 A5,A0    1@P0/ <d%] ) %
E550 DE 8D 2A 01 20 02 B5 A2 08 20 E6 B2 A0 02 B1 DD,9F  ^ *    5"  f2  1]
E560 F0 0A 20 2C B2 B0 05 10 03 20 A4 B3 A5 DD 8D 2B,10  p  ,20     $3%] +
E570 01 A5 DE 8D 2C 01 20 8D B1 4C 5E B0 20 11 E5 AD,C9   %^ ,    1L^0  e-
E580 10 01 8D 28 01 20 93 E5 A0 03 B9 29 01 99 24 01,6C     (    e  9)  $
E590 88 10 F7 20 8A EF 60 AD 23 01 F0 0D AD 26 01 85,1B    w  o`-# p -&
E5A0 D3 AD 27 01 85 D4 20 A0 B0 60 A9 46 20 A4 E3 AD,2F  S-'  T  0`)F $c-
E5B0 28 01 20 E2 E3 20 D9 E3 A5 D2 20 E2 E3 A5 D1 20,0B  (  bc Yc%R bc%Q
E5C0 E2 E3 AD 23 01 F0 20 20 D9 E3 AD 25 01 20 E2 E3,45  bc-# p  Yc-%  bc
E5D0 AD 24 01 20 E2 E3 A9 2D 20 A4 E3 AD 27 01 20 E2,50  -$  bc)- $c-'  b
E5E0 E3 AD 26 01 20 E2 E3 20 CA E3 60 BD 0C E6 9D 00,65  c-&  bc Jc`= f
E5F0 01 E8 E0 08 90 F5 AD 14 E6 85 C8 AD 15 E6 85 C9,A5   h`  u- f H- f I

     00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
E600 A2 00 20 96 B0 20 37 BF 20 5F E0 60 00 02 FC 0B,8B  "    0 7? _``  |
E610 00 0C FC 0E 00 0F 8C 31 01 4C 32 E6 8E 31 01 86,18     |     1 L2f 1
E620 DF 20 7E B6 F0 0C A5 DF D0 08 4C C7 E7 A2 00 4C,8B  _ ~6p %_P LGg" L
E630 38 B4 A9 25 85 CF 86 DB 86 DC EE 33 01 A2 02 B9,DB  84)% O [ \n3 " 9
E640 35 01 85 CD C8 98 9D 8F 01 E0 01 D0 05 AD 31 01,85  5  MH       ` P -1
E650 D0 16 B9 35 01 99 8F 01 C8 C0 4C B0 D0 C5 CD D0,39  P 95     H@L0PEMP
E660 E8 98 CA 9D 8F 01 D0 E1 AD 90 01 18 ED 91 01 F0,26  h J    Pa-    m  p
E670 BC A2 00 8E 92 01 20 02 B5 B9 35 01 C9 25 D0 0D,36  <"        595 I%P
E680 C8 B9 35 01 85 CF C8 20 02 B5 B9 35 01 C9 2A D0,92  H95  OH   595 I*P
E690 05 EE 92 01 D0 07 C9 23 D0 06 CE 92 01 20 FF B4,E5   n  P I#P N    □4
E6A0 20 FC B1 A0 02 B1 DD D0 03 4C 34 E7 20 A5 E7 8E,56   |1  1]P L4g %g
E6B0 18 01 AE 91 01 AC 18 01 C4 CE F0 02 B0 20 BD 8F,14    .  ,  DNp 0 =
E6C0 01 C5 CF F0 15 C5 CD D0 07 A2 00 8C 8E 01 F0 24,E8   EOp EMP "    p$
E6D0 D9 35 01 F0 05 EE 18 01 D0 D8 E8 C8 D0 DA A2 00,97  Y5 p n  PXhHPZ"
E6E0 20 A4 B3 A0 02 88 30 09 B9 0A 01 D1 DD F0 F6 90,59   $3   0 9  Q]pv
E6F0 43 4C A3 E6 20 B7 E7 AD 31 01 D0 07 AD 92 01 30,55  CL#f 7g-1 P -  0

     00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
E700 5B F0 59 AD 92 01 30 0F 20 DC E3 AD 18 01 20 E2,1F  [pY-  0  \c-   b
E710 E3 20 2B E8 20 CA E3 AD 31 01 D0 B9 A9 2A 20 A4,01  c +h Jc-1 P9)* $
E720 E3 20 BD E3 20 A3 B6 48 20 CA E3 68 C9 53 F0 AE,54  c =c #6H JchISp.
E730 C9 58 D0 09 20 EF B1 20 0F BF 4C 60 B0 C9 4D F0,5E  IXP  o1  ?L`0IMp
E740 94 C9 41 F0 17 C9 06 D0 09 20 DD E7 20 CA E3 4C,A8   IAp I P  ]g JcL
E750 DE E6 C9 44 D0 BE 20 64 B3 4C E0 E6 AD 31 01 F0,1F  ^fIDP> d3L`f-1 p
E760 03 4C D5 E6 AD 8E 01 A8 38 ED 18 01 AA 20 7B E8,78   LUf-  (8m   * {h
E770 CA D0 FA AE 90 01 BD 8F 01 C5 CD F0 07 20 52 E8,7B  JPz.  =  EMp  Rh
E780 C8 E8 D0 F2 8C 18 01 20 95 E8 B0 03 4C F1 E6 AD,B2  HhPr      h0 Lqf-
E790 92 01 10 08 A9 00 20 A4 E3 4C B2 E6 20 2B E8 20,E4      )  $cL2f +h
E7A0 CA E3 4C B2 E6 AD 0F 01 48 8E 0F 01 20 B2 B3 68,05  JcL2f-  H     23h
E7B0 8D 0F 01 C8 84 CE 60 F8 18 A5 DB 69 01 85 DB A5,1B     H N`x %[i  [%
E7C0 DC 69 00 85 DC D8 60 20 FC B1 10 0E B0 0C 20 A5,65  \i  \X` |1  0  %
E7D0 E7 20 A8 B5 20 CA E3 20 DD E7 4C 60 B0 A0 00 20,96  g (5 Jc ]gL`0
E7E0 BD E3 C9 06 D0 26 A9 3E 20 A4 E3 20 BD E3 C9 06,18  =cI P&)> $c =cI
E7F0 F0 ED 8D 8D 01 C8 C4 CE 90 03 F0 01 60 B9 34 01,3C  pm   HDN  p `94
```

```
     00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
E800 48 20 A4 E3 68 CD 8D 01 D0 EB F0 D3 C9 08 F0 09,36   H $chM  PkpSI p
E810 C9 7F D0 0B A9 5C 20 A4 E3 20 7B E8 4C DF E7 C9,63   I□P )\ $c {hL_gI
E820 0D D0 17 20 95 E8 90 11 20 CA E3 20 DC E3 20 32,93    P   h   Jc \c 2
E830 B6 20 D9 E3 A4 CE 20 A8 B5 60 C9 04 D0 0D 20 CA,08   6 Yc$N (5`I P  J
E840 E3 88 84 CE C8 20 97 E8 4C 26 E8 20 52 E8 C8 4C,F4   c  NH  hL&h RhHL
E850 DF E7 48 8C 19 01 A4 CE B9 35 01 99 36 01 88 30,91   _gH   $N95  6  0
E860 05 CC 19 01 B0 F2 68 C8 99 35 01 C0 4C 90 01 88,42    L  OrhH 5 @L
E870 E6 CE A5 CE C9 4C 90 02 C6 CE 60 88 10 01 C8 98,FD   fN%NIL  FN`   H
E880 48 C8 B9 35 01 99 34 01 C4 CE 90 F5 C6 CE 10 02,87   HH95  4 DN uFN
E890 E6 CE 68 A8 60 A4 CE C8 C0 52 90 02 A0 51 8C 1A,20   fNh(`$NH@R   Q
E8A0 01 A2 00 20 64 B3 AC 1A 01 C0 02 90 06 A0 00 20,D9    "  d3,  @
E8B0 FC B2 38 60 20 B6 00 C9 0A D0 56 EA EA EA EE 20,BA   |28` 6 I PVjjjn
E8C0 01 AD 20 01 C9 04 F0 1D C9 40 D0 45 A9 0A 20 A4,F8    -  I p I@PE)  $
E8D0 E3 AD 20 01 C9 3F D0 04 A9 0A D0 D8 C9 46 D0 EC,AB   c-  I?P ) PXIFPl
E8E0 A9 04 8D 20 01 A9 24 8D 2D 01 20 DC E3 CE 2D 01,69   )     )$ -  \cN-
E8F0 D0 F8 98 48 A0 41 20 1E B5 68 A8 AD 21 01 48 20,2C   Px H A  5h(-! H

     00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
E900 E2 E3 68 F8 18 69 01 D8 8D 21 01 20 CA E3 20 CA,11   bchx i X !  Jc J
E910 E3 60 20 A0 B6 C9 50 F0 1C 8E 1F 01 C9 53 D0 06,8F   c`  6IPp     ISP
E920 8E 20 01 EE 1F 01 20 FF B4 C0 50 B0 05 A2 21 20,C7      n   □4@P0 "!
E930 E6 B2 4C 60 B0 AD 1F 01 F0 EC 98 48 20 D1 E8 68,85   f2L`0-  pl H Qhh
E940 A8 4C 26 E9 20 4A E9 4C 5E B0 20 02 B5 A2 08 20,D6   (L&i JiL^0  5"
E950 E6 B2 AD 08 01 48 AD 09 01 48 84 CE 20 14 B2 10,B3   f2-  H-  H N  2
E960 03 20 A4 B3 A5 DD 85 E1 A5 DE 85 E2 A4 CE 20 02,93     $3%] a%^ b$N
E970 B5 20 FC B1 08 AD 0B 01 C9 FF D0 03 4C 3C B4 20,CD   5 |1 -  I□P L<4
E980 A4 B4 28 10 03 20 A4 B3 20 7A EA 20 17 B2 10 03,57   $4(   $3 zj  2
E990 20 A4 B3 68 8D 09 01 68 8D 08 01 38 A5 DD E5 DF,49    $3h   h   8%]e_
E9A0 85 D7 A5 DE E5 E0 85 D8 A5 D7 18 65 D3 85 D9 48,BC    W%^e` X%W eS YH
E9B0 A5 D8 65 D4 85 DA 48 CD 03 01 F0 05 90 0A 4C 3E,03   %XeT ZHM  p   L>
E9C0 B4 A5 D9 CD 02 01 B0 F6 A5 E2 C5 DE F0 04 90 08,61   4%YM  0v%bE^p
E9D0 B0 38 A5 E1 C5 DD B0 32 A5 E0 C5 E2 F0 04 90 08,0B   08%aE]02%`Ebp
E9E0 B0 09 A5 DF C5 E1 B0 03 4C 3B B4 A2 02 18 B5 DD,2A   0 %_Ea0 L;4"  5]
E9F0 65 D7 95 DD B5 DE 65 D8 95 DE CA CA 10 EF A2 00,50   eW ]5^eX ^JJ o"

     00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
EA00 20 F2 B4 20 EC B4 A1 D3 81 D9 A5 E2 C5 D4 D0 F0,84    r4 l4!S Y%bETPp
EA10 A5 E1 C5 D3 D0 EA A5 DF 48 A5 E0 48 A5 E1 85 D7,D7   %aESPj%_H%`H%a W
EA20 A5 E2 85 D8 A5 DD C5 DF D0 06 A5 DE C5 E0 F0 0D,DC   %b X%]E_P %^E`p
EA30 A1 DF 81 E1 20 D6 B4 20 D5 B4 4C 24 EA 68 85 E0,38   !_ a V4 U4L$jh `
EA40 68 85 DF 68 85 D4 68 85 D3 20 A0 B0 A5 DE 48 A5,65   h _h Th S  0%^H%
EA50 DD 48 A0 02 B1 E1 48 A9 00 91 E1 8D 0A 01 8D 0B,51   ]H  1aH)  a
EA60 01 A5 D7 85 DD A5 D8 85 DE B1 DD F0 03 20 10 E4,A5    %W ]%X ^1]p   d
EA70 68 91 E1 68 85 DD 68 85 DE 60 AD 0A 01 8D 08 01,C2   h ah ]h ^`-
EA80 AD 0B 01 8D 09 01 60 20 4A E9 20 72 B3 4C 5E B0,64   -      ` Ji r3L^0
EA90 20 02 B5 C0 40 90 03 4C 3C B4 20 FC B1 F0 1B 08,EA     5@@  L<4 |1p
EAA0 20 A4 B4 20 7A EA 28 10 03 20 A4 B3 20 17 B2 10,91    $4 zj(   $3  2
EAB0 03 20 A4 B3 20 72 B3 4C 5E B0 4C 3B B4 EE 13 01,E7     $3 r3L^0L;4n
EAC0 8E 11 01 C0 50 B0 30 8A 48 A2 00 20 4A E2 68 AA,49      @P00 H"  Jbh*
EAD0 A5 D1 9D 00 01 E8 A5 D2 9D 00 01 E8 20 02 B5 E0,F9   %Q   h%R   h  5`
EAE0 08 90 E0 C0 50 B0 10 A2 00 20 4A E2 A5 D1 85 C8,F2     `@P0 "  Jb%Q H
EAF0 A5 D2 85 C9 EA EA EA A2 00 20 CA E3 BD 01 01 20,C3   %R Ijjj"  Jc=
     00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
EB00 E2 E3 BD 00 01 20 E2 E3 E0 00 D0 08 A9 2D 20 A4,7D  bc=   bc` P )- $
EB10 E3 4C 1B EB E0 04 F0 F4 20 D9 E3 E8 E8 E0 08 90,9E  cL k` pt Ychh`
EB20 DB A5 C9 20 E2 E3 A5 C8 20 E2 E3 20 CA E3 A5 D4,64  [%I bc%H bc Jc%T
EB30 20 E2 E3 A5 D3 20 E2 E3 20 D9 E3 A5 D6 20 E2 E3,E2   bc%S bc Yc%V bc
EB40 A5 D5 20 E2 E3 20 CA E3 20 E8 B1 20 5E B0 20 9B,B0  %U bc Jc h1 ^0
EB50 B1 8E 10 01 20 71 E4 AD 28 01 8D 10 01 C9 EE F0,90  1    qd-(    Inp
EB60 0E CD 0A 01 F0 09 20 72 EB 20 8D B1 4C 51 EB 4C,1E   M  p  rk  1LQkL
EB70 5E B0 A2 01 BD 00 01 9D 29 01 B5 D3 9D 2B 01 CA,6F  ^0" =   ) 5S + J
EB80 10 F2 E8 60 86 EF C0 50 B0 02 E6 EF 20 92 EB 4C,AE   rh` o@P0 fo  kL
EB90 5E B0 6C F0 00 86 EE C0 50 B0 02 E6 EE 20 A3 EB,D0  ^0lp  n@P0 fn #k
EBA0 4C 5E B0 6C F2 00 20 AC EB 4C 5E B0 6C EC 00 20,11  L^0lr  ,kL^0ll
EBB0 D7 EB 20 66 A6 B0 01 60 20 66 A6 B0 FB 48 20 10,5F  Wk f&0 ` f&0{H
EBC0 EC C9 0F D0 0C A9 0D 20 0D EC A9 0A 20 0D EC 68,02  lI P )   l)   lh
EBD0 60 C9 11 D0 E9 68 60 29 7F 48 A5 E3 F0 02 68 60,EF  `I Pih`)□H%cp h`
EBE0 68 48 C9 00 F0 22 C9 1B F0 1E C9 0D F0 1A C9 0A,1F  hHI p"I p I p I
EBF0 F0 16 C9 07 F0 12 C9 08 F0 0E C9 20 B0 0A 48 A9,5A  p I p I p I 0 H)

     00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
EC00 5E 20 0D EC 68 18 69 40 20 0D EC 68 60 4C 63 A6,30  ^  lh i@  lh`Lc&
EC10 A9 00 85 E3 20 6F EC C9 00 F0 F9 C9 1A D0 03 4C,70  )  c olI pyI P L
EC20 5E B0 C9 03 D0 03 4C A6 B0 C9 19 D0 03 4C F1 EF,A0  ^0I P L&0I P Lqo
EC30 C9 02 D0 08 A9 80 8D 53 A6 4C 00 C0 C9 07 D0 03,A1  I P )  S&L @I P
EC40 4C 72 89 C9 0F D0 03 E6 E3 60 C9 14 D0 20 20 6F,18  Lr I P fc`I P  o
EC50 EC C9 30 F0 0F C9 31 D0 15 AD 00 A0 49 80 8D 00,7E  lI0p I1P -  I
EC60 A0 4C 6C EC AD 0C A0 49 20 8D 0C A0 A9 18 60 20,FE   Lll-  I    ) `
EC70 60 A6 29 7F 48 AD 33 01 D0 0A 68 48 C9 11 F0 09,32  `&)□H-3 P hHI p
EC80 C9 09 F0 05 68 20 D7 EB 60 68 60 A2 00 B9 38 01,FF  I p h Wk`h`" 98
EC90 C9 20 D0 03 20 44 ED A9 01 85 BD 20 1A E1 B0 06,C9  I P  Dm)  =  a0
ECA0 20 4B B4 4C FD BB A5 C0 F0 F6 B9 35 01 C9 20 F0,FF   K4L};%@pv95 I p
ECB0 06 20 4B B4 4C FD BB AD FF 01 CD 19 01 F0 0A B0,66    K4L};-□ M  p 0
ECC0 10 A2 20 20 4D B4 4C FD BB AD FE 01 CD 18 01 90,7F   "  M4L};-~ M
ECD0 F0 AD 19 01 48 AD 18 01 48 86 BD 8E 8F 01 20 FF,0C  p-  H-  H =    □
ECE0 B4 C0 50 B0 39 C9 3B F0 35 C9 28 F0 08 20 43 B4,E2  4@P09I;p5I(p  C4
ECF0 68 68 4C FD BB C8 20 02 B5 C0 50 B0 21 B9 35 01,25  hhL};H  5@P0!95

     00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
ED00 C9 29 F0 1A 8A 48 A2 00 20 C1 E0 68 AA A5 D1 9D,7B  I)p  H"  A`h*%Q
ED10 90 01 A5 D2 E8 9D 90 01 E8 EE 8F 01 D0 D8 A2 00,49    %Rh   hn  PX"
ED20 68 85 DD 68 85 DE 48 A5 DD 48 20 47 BF A9 01 85,45  h ]h ^H%]H G?)
ED30 BE E6 BC A5 BC C9 20 90 08 A2 24 8E 12 01 4C 4D,87  >f<%<I   "$   LM
ED40 B4 4C 6A BC B9 35 01 20 A3 B6 DD 86 ED D0 2A B9,18  4Lj<95  #6] mP*9
ED50 36 01 20 A3 B6 DD 87 ED D0 1F B9 37 01 20 A3 B6,72  6  #6] mP 97  #6
ED60 DD 88 ED D0 14 BD 89 ED 85 E1 BD 8A ED 85 E2 68,44  ] mP = m a= m bh
ED70 68 C8 C8 C8 A2 00 4C 39 B7 E8 E8 E8 E8 E8 BD 86,AD  hHHH" L97hhhhh=
ED80 ED D0 C1 A2 00 60 49 46 45 F8 ED 49 46 4E 08 EE,B9  mPA" `IFExmIFN n
ED90 49 46 50 15 EE 49 46 4D 21 EE 53 45 54 4B EE 2A,D5  IFP nIFM!nSETKn*
EDA0 2A 2A E6 ED 2E 45 4E 63 BB 00 2E 4D 45 EB ED 2E,A1  **fm.ENc; .MEkm.
EDB0 4D 44 F0 ED 2E 45 4E 63 BB 00 2C 58 29 1B BE 29,9D  MDpm.ENc; ,X) >)
EDC0 2C 59 1C BE 00 2C 58 20 1E BE 2C 59 20 1D BE 00,FC  ,Y > ,X  >,Y  >
EDD0 2C 58 20 0C BE 2C 59 20 0B BE 00 23 4C 2C 7B BD,AB  ,X  >,Y  > #L,{=
EDE0 23 48 2C 8C BD 00 86 BF 4C FD BB 86 BB 4C FD BB,19  #H, =  ?L}; ;L};
EDF0 A2 29 20 4D B4 4C FD BB 20 2D EE A5 D1 D0 06 A5,35  ") M4L}; -n%QP %
```

```
     00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
EE00 D2 D0 02 86 BF 4C FD BB 20 2D EE A5 D1 D0 F4 A5,3C  RP  ?L}; -n%QPt%
EE10 D2 F0 F2 D0 EE 20 2D EE A5 D2 30 02 86 BF 4C FD,20  RprPn -n%R0  ?L}
EE20 BB 20 2D EE A5 D2 10 02 86 BF 4C FD BB 84 BF 20,4B  ; -n%R   ?L}; ?
EE30 02 B5 20 4A E2 AD 1A 01 F0 06 8C 12 01 20 49 B4,C8   5 Jb-  p     I4
EE40 60 84 C1 4C FD BB 86 C1 4C FD BB 20 1A E1 B0 03,8A  ` AL}; AL};  a0
EE50 4C 3A EE A5 C0 D0 08 A2 2A 20 4D B4 4C FD BB 20,4C  L:n%@P "* M4L};
EE60 02 B5 B9 35 01 C9 3D F0 06 20 43 B4 4C FD BB A5,AE   595 I=p  C4L};%
EE70 DE 48 A5 DD 48 C8 20 2F EE 68 85 DD 68 85 DE A0,D8  ^H%]HH /nh ]h ^
EE80 00 A5 D1 91 DD A5 D2 C8 91 DD 4C FD BB A5 BC F0,BE   %Q ]%RH ]L};%<p
EE90 0C A5 BE D0 20 A2 29 20 4D B4 4C FD BB E6 BB AD,5B   %>P ") M4L};f;-
EEA0 13 01 D0 0E A0 00 68 91 DD 68 48 C8 91 DD 88 B1,E2    P   h ]hHH ] 1
EEB0 DD 48 4C FD BB E6 C2 D0 09 E6 C3 D0 05 A2 2E 20,C5  ]HL};fBP fCP ".
EEC0 4D B4 86 BE 86 BD AD 8F 01 F0 5B B9 35 01 C9 28,B5  M4 > =-  p[95 I(
EED0 F0 08 A2 25 20 4D B4 4C 3C EF 86 E5 C8 20 02 B5,16  p "% M4L<o eH  5
EEE0 C9 3B F0 3A C0 50 B0 36 C9 29 F0 32 84 E4 20 1A,F0  I;p:@P06I)p2 d
EEF0 E1 B0 07 A4 E4 20 79 BF B0 DD 84 E4 A0 00 A6 E5,88  a0 $d y?0] d  &e

     00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
EF00 BD 90 01 91 DD E8 BD 90 01 E8 C8 91 DD A4 E4 86,DB  =   ]h=  hH ]$d
EF10 E5 A2 00 CE 8F 01 30 BA 20 FF B4 4C DD EE AD 8F,D0  e" N  0: □4L]n-
EF20 01 D0 AF 4C FD BB B9 35 01 C9 3B F0 F6 C0 50 B0,ED   P/L};95 I;pv@P0
EF30 F2 90 9F A5 BC D0 05 86 BB 4C FD BB 86 E6 C6 BC,77  r  %<P  ;L}; fF<
EF40 30 11 F0 0A C6 C2 A5 C2 C9 FF D0 02 C6 C3 68 68,94  0 p FB%BI□P FChh
EF50 4C FD BB 86 BC A2 2B 20 4D B4 4C FD BB A5 EE F0,4F  L}; <"+ M4L};%np
EF60 07 20 65 EF 60 6C F6 00 20 88 81 A9 FF 20 C4 EF,30    eo`lv    )□ Do
EF70 84 FD A9 09 20 A5 89 20 2E 83 20 9C 82 20 7B 8C,E7   })  %  .    {
EF80 D8 A9 00 90 02 A9 01 4C E5 EF A5 EF F0 07 20 92,01  X)   ) Leo%op
EF90 EF 60 6C F4 00 20 88 81 A9 00 20 C4 EF AD C5 8F,56  o`lt    )  Do-E
EFA0 C9 3F F0 0C A9 01 8D 30 A6 20 87 8E D8 4C E5 EF,94  I?p )  0&   XLeo
EFB0 20 B6 8D A9 07 8D 02 A4 EE 02 A4 A2 01 20 9A 8E,59   6 )   $n $"
EFC0 D8 4C E5 EF 20 86 8B 8D 4E A6 AD 24 01 8D 4C A6,54  XLeo    N&-$  L&
EFD0 AD 25 01 8D 4D A6 AD 26 01 8D 4A A6 AD 27 01 8D,5A  -%  M&-&  J&-'
EFE0 4B A6 A0 80 60 20 9C 8B 4C B8 81 20 B2 B7 6C D1,5D  K&  `   L8  27lQ
EFF0 00 20 B2 B7 4C 00 00 20 B2 B7 4C 03 00 00 00 00,0A    27L    27L
```

Memory block $E000-$EFFF checksum: B20A

APPENDIX H

SOFTWARE LISTING

Enter range limits for raw ASCII dump: B000-BFFF

```
4CEDBF4CAEB00CE6B4E8BBE82DEC4DB4AFE3A2FF9AE886EE86EF8E130186E38E1F
018E53A620EBE5AD02A009808D02A0A9018D20018D21018D0F0185DB8E0C018E0D
018E0E01A047201EB5A90B85EA20F7EA2096B08E2801200FE32018E3A200D82086
8BA9008D53A68D13018D330120CAE3A2FF8E12018E11019AE88E14018E15012064
B2A0002002B5C050B0E420AEB6A2ED4C4DB4AD000185D3AD010185D48AA00291D3
60A9808D53A600EAEA20F7EA4C5EB0A20C20E6B24C60B02049E44C5EB020A0B68D
0F01C943D0038E0F0120FFB4C050B0138E1101A9018D1301204AE2E6D1A5D1291F
85EA4C60B020A0B68D0E01C943D0038E0E014C60B08E14018E130198482037BF20
5FE08E110168A88E170120A0B6C94EF007C94CD006EE170120FFB420FCB1AD0801
8D2F01AD09018D30014CC7BBA9018E11018D1301204AE220FFB42049B14C60B04C
EBEFB93501C92FD018A9FF8D09012014B2AD1C0185DDAD1D0185DE20CBB14C60B0
20FCB120B5B14C60B0A5C88D2901186D22018D2B01A5C98D2A0169008D2C018E22
01208FE384CE207CE5A4CE209DE3608E100120A0B6C946D00FC8A20A20E6B2AD0A
018D10012002B560F0109012A00288300DB90A01D1DDF0F6B00420E8B16020B2B3
AD0E01D0062032B620D9E320A8B520CAE320A4B3A002B1DDD0D120EFB120CAE360
86E3A92F20A4E3A92F20A4E360A20820E6B2C0509007A9FF8D0B01D0082002B5A2
0A20E6B220BFB4A002B1DDD00218608830FBB1DDD90801F0F6900160A5DD8D1C01
A5DE8D1D0120A4B34C17B284CEAC220191C8EE2201A4CE60AD13011008A94FEE11
012070BE4C11BEA002B1DD60EE13014CB8BB20BBB5A000B93501C9309004C93A90
0160A2082042B684E1AD1A0138E5E18D1A012064B3A4E12002B5C050B005A4E120
FCB220F1B2F027AD080118F86D0C018D0801AD09016D0D018D0901D82035B620BB
B5A00084E120C5B2D0C620CAE34C72B0A0002002B5B93501C92FD003D936016020
42B6C050B009A5DFF005A2004C44B46020D5B2A920D93501D0F06018AD0C016D0D
018D150160984820ADB4AD1A013865D3488A65D4CD0301900EF0034C3EB4A868CD
0201B0F6489885D46885D320B6B44C33B320E9B420E8B4A1DF81E1A5DDC5DFD0F0
A5DEC5E0D0EAA000AD080191DDAD0901C891DD68AAC8BD3501E891DDCC1A01D0F4
098091DDA20020A0B0602014B2D00160B0FD20A4B420A4B3A002B1DDD00FA5DF85
D3A5E085D420A0B02014B260A1DD81DF20D7B420D6B488D0F3A1DD81DF0820D7B4
20D6B42810F230CE20D7B420D7B4A1DD10F920D7B4602026B6B1DD8D0801C8B1DD
8D0901C8AD0F01F02DB1DDC93BF027C920F0032009B420F9B3A6EAC93BF0172009
B420F9B3C93BF00A2009B420F9B3C93BD0032002B42009B44CF3B3B1DDC8C920F0
F98860E88A2907D0FA60E050B01BB1DD300EC909F01A9D3501C8E8C920D0EB6029
7FC909F0039D35018AA8A2006868602002B4C8D0D4E8E8E8E8E8E8E8E8E8E8E8E8
E8E8E8E8E8E8E8E8E8E8E8E88A48A20020B7E7AD130110068E13018D1201A5E385
CF86E3A90720A4E32042BF20CAE3A92120A4E36820E2E39848A000201EB568A820
32B6A92F20A4E3AD280120E2E320CAE3AD1201D005A5CF85E360A9FF85DB4C58B0
A5DD85DFA5DE85E060A5D385DFA5D485E060A5D385E1A5D485E260AD000185DDAD
010185DE60AD040185DDAD050185DE60E8E8E8E8E8E8E88A0AAAF6D3D002F6D4A2
0060E8E8E8E8E8E8E88A0AAAD6D3B5D3C9FFD002D6D4A200602010B5B93501C8C0
50B005C920F0F48860B93501C8C050B005C920D0F48860B92AB5D0016020A4E3C8
D0F4204154204C494E45000D0A0A4C4142454C2046494C453A20205B202F203D
2045585445524E414C205D0D0A0A0052454144592064F52205041535320320D0A
0050414745200000D0A524145205631 2E300D0A434F5059524947485420313937 39
2053594E455254454B20535953544D5320434F52502E200D0A0D0A00A2008A48
BD350120A4E368AAE88810F2A2006086E3A93E20A4E32026B620BDE3C90DD008C8
8C1A0120CAE360C908F009C97FD01EA95C20A4E3883008A9209935014CC5B520CA
E3AD1501F0C72035B64CBDB5C918F0EE993501C909F010C8C046D005A90720A4E3
C050B0B9D0B0984820DCE3C8982907D0F768A8C84C0FB6A055A9209935018810FA
C86020DCE3AD090120E2E3AD080120E2E360A9009D00019D010185DF85E0207EB6
D003A2006048A5DFF009AD1101F004A2006860C89848A0041E00013E010188D0F7
```

```
68A8681D00019D00014C4EB620A0B6C9309018C93A900FC9419010C947B00C290F
186909E6DF290FE6E060A90060B93501C9619006C97BB00229DF60A900F006A901
D002A9FF85E184E2A4E2A5E1F00E1006BDC7B8D00D60BD41B8D00760BD41B7D001
608D1A0120A0B6CD1A01089838E5E2C8E828F022C901F005C902F002E8E8E8A5E1
3005D0C24CBCB6BDC6B8290F86E03865E0AA4CBCB648A5E1D00768C901D0A9F005
68C902D0A2A5E1F00386D060BD41B785E1BD42B785E26868E085B00620FFB44C3C
B72002B5A2006CE1004252A6B0434C52B0505524E5464FC2B050524CB14155B4B0
4153FCB05041B7E2525535B14D41ECB04F55A7BB4F4E50E34F4633E3484112E947
45BCB04C4100BF45441CE64E55FDE3444590EA464916E64D4F87EA434F44E95345
BDEA5553F7EF44554EEB454E84EB4C4F95EB4443A6EB002E53A6386E53A6600000
0000000000000000000534946BA42415ABA454E63BB4259C4BA53454FBA4449FE
BA4C533BBA4C4340BA4D439ABA4F4331BA4F532CBA434527BA435436BA525322BA
444507BB524337E44453B0B9455341EE454346EE454A0BBA4D448DEE4D4533EF00
000000000000000000000000000000000000000000004120CABD0029200ABE
00544158AA544159A8545358BA5458418A5458539A54594198434C4318434C44D8
434C4958434C56B85345433853454F8534549784E4F50EA525449405254536044
4558CA44455988494E58E8494E59C85048414850485008504C4168504C50284252
4B000042434390424353B0424551F0424D4930424E45D042504C10425643504256
537000524F52C5C16E7E66766A414443E8DA6D7D796575716169414E44E8DA2D3D
39253531212941534CC5C10E1E06160A42495482802C24434D50E8DACDDD9C5D5
D1C1C94350588382ECE4E04350598382CCC4C0444543C4C0CEDEC6D6454F52E8DA
4D5D594555514149494E43C4C0EEFEE6F64A4D5092004C6C4C4441E8DAADBDB9A5
B5B1A1A94C4458A5A2AEBEA6B6A24C4459C5C2ACBCA4B4A04C5352C4C14E5E4656
4A53528100204F5241E8DA0D1D190515110109524F4CC5C12E3E26362A534243E8
DAEDFDF9E5F5F1E1E9535441E7D88D9D998595918153545883A08E869653545983
C08C849400AD1701F00EAD1301C901D00720BFBEA90585E7204AE2AD1A01F0068C
12014C49B4A5D11865D785D7A5D265D885D8A5D11865D985D9A5D265DA85DAAD13
011018A97FEE11012070BEA5D1EE11012070BEA5D2EE11012070BE4CFDBBAD1301
F00F300DAD1701F008AD1F01F00320D1E84CFDBBA95F4C9BBD8E1201F0178C1601
F0128E1601F00D8C1401F0088C1701F0038E17014CFDBB204AE2EE1E014C49B220
4AE2A2008E1E014C49B2A5DD48A5DE48204AE2AD1A01F0068C12014C49B46885DE
6885DDA5D185D785D9A5D285D885DAAD3501C920F008AD1301D0032057BBAD1301
100320EDBE4CFDBBAD1201488C1201204AE2688D1201A5D185D9A5D285DA4CFDBB
204AE2A5D120EBBA2002B5C050B0D5900AC05090062043B44CFDBBC93BF0C5C927
D0DDC8C050B0EDB93501C927D003C8D0D620EBBAB850EB48AD13011008A93FEE11
012070BE682070BE60202FBB2057BB4CFDBB202FBBD01DA001C8B1CE10FBB1CEC8
91CE8888C001D0F5C8A92F91CE20DBB4205FE02057BB4CFDBB204AE2AD3501C920
D0062048B44CFDBBA000201AE1A5DD85CEA5DE85CFAD1301F0058E1E01A9FF60A0
00A5D191CEC8A5D291CE602047BFA5BCD00AA5BFD00AA5BBD00AF00DA221D006A2
22D002A223204DB4AD1301F00E3006AD17014CFEBE20EDBE4C58B0EE1301AD1401
F01C209DE3A02E201EB54C58B08E2201209BB18E1101A2FF8E13018E0A01AD2F01
8D0801AD30018D09012014B2A20086E686DC86DBA2FF9AE88E160186C486C586C2
86C386BB86BC86BF86C1A90085D785D9A90285D885DAAD1301101120F0BE4C08BC
A2006885DD6885DE20A4B3A5DE48A5DD488E1E012059B2D054A5BBF005A227204D
B4E6C4D009E6C5D005A22F204DB4AD1401D008A9018D12014C46B42047BF20CAE3
20CAE320CAE3A5BA8D10012071E4A9008D1101AD2F018D0801AD30018D09012014
B2F0CEA2FF9AE84C08BCA200AD1701F0152047BFAD1301C901D00BA5BCF004A5C1
F00320CAE320B2B384E9E6E6A000B93501C93BF00320FFB486BDA5BBF008A22420
44ED4CFDBBA5BFF008A2192044ED4CFDBBA000B93501C93BD0034CFDBBC920F008
AD1301D0032079BF20FFB4B93501C93BF0E7C050B0E3C92ED018C8B93701C920F0
06204AB44CFDBBA28520AEB6A2004CE5BCB93801C920D05020B2B6F009BD41B820
70BE4CFDBBA26520B2B6F034BD41B82070BE20FFB4AD1301F023204AE218A5D1E5
D748A5D2E5D8D00B68C980900F204CB44CFDBBC9FFD0F668C98090F14C04BDA200
20B6B6D0034C8BEC2002B5B93501C923D079A20B203DBEA2552044EDC8B93501C9
27D006B936014C04BDA200204AE2A5D14C04BD204AE2A5D12070BEAD1E01F026A9
1FD00F204AE2A5D22070BEAD1E01F015A92F48AD1301101068EE1101482070BE68
```

```
C92FF0074CFDBB684CFDBBA5D1EE11014C04BD2070BEAD1E01D0E9A90F4C9BBDA2
0C203DBE4CFDBBA2F620AEB6AAB93501C92AD00AC820C1E0F03AA23FD01AC928D0
0FC820C1E0A2FB20AEB6A234D009D00720C1E0F00CA24A2044EDAA4CC8BAE8E8E8
E8203DBEA5D12070BEA5D24CBDBDE8E8E8E8E8E8E8E8203DBEA5D12070BEA5D2
F00AA200AD1301F003204DB44CFDBB86DDA6D0BDC8B885DEA90485DFBDC7B8C6DF
30100A9001E8C6DDD0F4B012A2002042B460A5DE0A9001E8C6DDD0F890EEBDC8B8
48A200AD1301F0573059AD1701F031A5BCF004A5C1F029A5E7D007A90585E720BF
BEA5E7C90ED0092047BF20CAE34C8BBE20DCE3A5E718690385E7684820E2E3AD16
01F01A6881D920DAB420D9B460A5D820E2E3A5D720E2E3A92D20A4E36068B850E5
68203CB2AD1101D00620DAB44CE5BECE1101AD2201C9FFB001602074B18E2201A5
D7203CB2A5D8203CB260F0032062E220EFB1200FBF2019BF4C5EB0A5DC20E2E3A5
DB4CE2E3A92C20A4E3A5D820E2E3A5D720E2E3A92C20A4E3A5DA20E2E3A5D94CE2
E3AD040185D5AD050185D66098484C55BF9848AD1301C901D022AD1701F01DA5E6
F019A4E7A5BCD013C820DCE3C01090F8A4E92035B620DCE320A8B586E768A886E6
60A5D585B9A5D685BAA5BCF0122066E0D00218602076E0D0242089E04CA2BF2066
E0D0192047BF207EE084E42094E0A6E420A0E020A0E020A0E04CBCBF84E42094E0
A6E4BD3501C940901D20A0E0BD3501C920F01DA5BCF007BD3501C929F012BD3501
2054E290E3205FE0A2002041B438604C00E0A96085B6AD00A0297F8D00A04C12B0
00000000
```

Memory block $B000-$BFFF checksum: 672D

APPENDIX I

SOFTWARE LISTING

Enter range limits for hex dump: E000-EFFF

```
B1D5098091D5389865D585D5A2008A65D685D6CD0701F02EB033205FE0A4E4201A
E1A5DE48A5DD4820A8E02059B2F00C6885DD6885DE2047B44C50E06885DD6885DE
A4E41860A5D5CD060190CD203FB4A5B985D5A5BA85D6205FE0A4E43860A900A002
91D560B93501C921D008D93601D003D9370160B93501C92E4C6BE0A5C4993601A5
C599370160A5C2993601A5C399370160A000A5D791D5A5D8C891D560BD3501C891
D5E860A002B1DDC92EF004C921D00920D7B420D7B420D7B420A4B3608E1E01C050
90042043B46086D186D28E1A014CE5E0B93501C92BF007C92DF01EC92060C8201A
E1B003EE1A0118AD180165D185D1AD190165D285D24CD6E0C8201AE1B003EE1A01
38A5D1ED180185D1A5D2ED190185D24CD6E020CAB486C084E18E18018E1901B935
01C925D01BC8B93501C930F006C931F00538601890013 82E18012E19014C2EE120
9AE1B006B935014CA8E1C8B93501209AE1B0F7984888B9350184CE209AE19020F0
17A8BD90E1186D18018D1801BD95E16D19018D190188D0EAA4CEE8E005D0D568A8
A2003860010A64E8100000000327C9309004C93A90021860290F3860C924F04AC9
3DF039A6E1A002B1DDC92FD001C8B1DDF0034CBDE2A200A4E1C8B935012054E290
F7C921F004C92ED008A5BDD0EBA5BCD0E7AD1301F0032045B41860A5D78D1801A5
D88D1901C83860A218C82042B6A5E0D0482040B41860B1DD08297FDD3501D011E8
C82810F18AA8B935012054E2B00B900128A20020A8E0B8508798AAA000B1DD8D18
01C8B1DD8D1901C8B1DDC92FF003EE1E018AA8A200E6C0386020C1E0D001602043
B460C92E9008C93DF004C97B90013860A009201EB520CAB486CF20CAE3A002B1DD
D00420CAE36020B2B3AD3501C921F004C92ED00620A8E04C6FE2981865CF85CF20
A8B5A93D20A4E32035B620A8E0A4CFA4CF20DCE3C884CFC012F0C0C024F0BCB0B5
D0EC8E11014C5EB2B1DDC92ED01BA5BCF03EC8B1DDC5C2D037C8B1DDC5C3D030BD
3501C92ED029F02AC921D014C8B1DDC5C4D01CC8B1DDC5C5D015A5BDF014D015A5
BDD00BBD3501C92EF004C921D0074C21E2E8E8E8C84C05E2AD0CA029DF8D0CA060
AD00A009808D00A060AD0CA009208D0CA060AD00A0297F8D00A060A20820E6B2AD
0801D006200FE34C60B0C901F0034C37B42018E34C60B0A20820E6B2AD0801D006
2021E34C60B0C901D0E3202AE34C60B0206DE32070E32073E3A2608E2D018E2E01
EE2D01D0FBEE2E01D0F6A20060202AE32073E3602021E3206AE3602073E32018E3
602073E3200FE3604884E486E5AD1F01F005684820B4E86820AFEBA4E4A6E5D860
84E486E52010ECA4E4A6E5D86020D3E3A90A20A4E360A90D20A4E36020DCE3A920
20A4E360484A4A4A4A20EFE36820EFE360290F0930C93A9002690620A4E3602003
E44C5EB020FCB1AD0B01C9FFD0034C3CB4A00018F808C002D00F20A4B32890034C
3DB4B1DDD0EAD86028B90A017908019908019 1DDC8D0DDA96F4C9BBDA20820E6B2
84CE2014B2A4CE602067E4AD2801C9EEF00EAD2301F0034C5EB02089E44C4CE420
A0B04C5EB0209BB120A0B6C920D0062096B0189004C941D00BA5D385DDA5D485DE
189003203CE42088E32011E58D2301205DEFD069A5DD8D2401A5DE8D250138AD2B
01ED290148AD2C01ED2A01AA6885D11865DD8D26018A85D265DE8D2701A9008D23
01AD1001F005CD2801D01FEE2301AD2701CD03019014D008AD2601CD0201900AA9
018D1201A2004C3EB4205DEFD00CA2002097E52096E320AAE560A200AD2301F003
8D120120A0B02036B460A9288D2401A92D8D2601A9018D25018D2701602072EB20
A0B6C920F048C958D00EA9EE8D10018D09012041E4C8D00A209BB1C050B02F203C
E4A5DD8D2901A5DE8D2A012002B5A20820E6B2A002B1DDF00A202CB2B005100320
A4B3A5DD8D2B01A5DE8D2C01208DB14C5EB02011E5AD10018D28012093E5A003B9
29019924018810F7208AEF60AD2301F00DAD260185D3AD270185D420A0B060A946
20A4E3AD280120E2E320D9E3A5D220E2E3A5D120E2E3AD2301F02020D9E3AD2501
20E2E3AD240120E2E3A92D20A4E3AD270120E2E3AD260120E2E320CAE360BD0CE6
9D0001E8E00890F5AD14E685C8AD15E685C9A2002096B02037BF205FE0600002FC
0B000CFC0E000F8C31014C32E68E310186DF207EB6F00CA5DFD0084CC7E7A2004C
38B4A92585CF86DB86DCEE3301A202B9350185CDC8989D8F01E001D005AD3101D0
16B93501998F01C8C04CB0D0C5CDD0E898CA9D8F01D0E1AD900118ED9101F0BCA2
```

```
008E92012002B5B93501C925D00DC8B9350185CFC82002B5B93501C92AD005EE92
01D007C923D006CE920120FFB420FCB1A002B1DDD0034C34E720A5E78E1801AE91
01AC1801C4CEF002B020BD8F01C5CFF015C5CDD007A2008C8E01F024D93501F005
EE1801D0D8E8C8D0DAA20020A4B3A002883009B90A01D1DDF0F690434CA3E620B7
E7AD3101D007AD9201305BF059AD9201300F20DCE3AD180120E2E3202BE820CAE3
AD3101D0B9A92A20A4E320BDE320A3B64820CAE368C953F0AEC958D00920EFB120
0FBF4C60B0C94DF094C941F017C906D00920DDE720CAE34CDEE6C944D0BE2064B3
4CE0E6AD3101F0034CD5E6AD8E01A838ED1801AA207BE8CAD0FAAE9001BD8F01C5
CDF0072052E8C8E8D0F28C18012095E8B0034CF1E6AD92011008A90020A4E34CB2
E6202BE820CAE34CB2E6AD0F01488E0F0120B2B3688D0F01C884CE60F818A5DB69
0185DBA5DC690085DCD86020FCB1100EB00C20A5E720A8B520CAE320DDE74C60B0
A00020BDE3C906D026A93E20A4E320BDE3C906F0ED8D8D01C8C4CE9003F00160B9
34014820A4E368CD8D01D0EBF0D3C908F009C97FD00BA95C20A4E3207BE84CDFE7
C90DD0172095E8901120CAE320DCE32032B620D9E3A4CE20A8B560C904D00D20CA
E38884CEC82097E84C26E82052E8C84CDFE7488C1901A4CEB93501993601883005
CC1901B0F268C8993501C04C900188E6CEA5CEC94C9002C6CE60881001C89848C8
B93501993401C4CE90F5C6CE1002E6CE68A860A4CEC8C0529002A0518C1A01A200
2064B3AC1A01C0029006A00020FCB2386020B600C90AD056EAEAEAEE2001AD2001
C904F01DC940D045A90A20A4E3AD2001C93FD004A90AD0D8C946D0ECA9048D2001
A9248D2D0120DCE3CE2D01D0F89848A041201EB568A8AD21014820E2E368F81869
01D88D210120CAE320CAE36020A0B6C950F01C8E1F01C953D0068E2001EE1F0120
FFB4C050B005A22120E6B24C60B0AD1F01F0EC984820D1E868A84C26E9204AE94C
5EB02002B5A20820E6B2AD080148AD09014884CE2014B2100320A4B3A5DD85E1A5
DE85E2A4CE2002B520FCB108AD0B01C9FFD0034C3CB420A4B428100320A4B3207A
EA2017B2100320A4B3688D0901688D080138A5DDE5DF85D7A5DEE5E085D8A5D718
65D385D948A5D865D485DA48CD0301F005900A4C3EB4A5D9CD0201B0F6A5E2C5DE
F0049008B038A5E1C5DDB032A5E0C5E2F0049008B009A5DFC5E1B0034C3BB4A202
18B5DD65D795DDB5DE65D895DECACA10EFA20020F2B420ECB4A1D381D9A5E2C5D4
D0F0A5E1C5D3D0EAA5DF48A5E048A5E185D7A5E285D8A5DDC5DFD006A5DEC5E0F0
0DA1DF81E120D6B420D5B44C24EA6885E06885DF6885D46885D320A0B0A5DE48A5
DD48A002B1E148A90091E18D0A018D0B01A5D785DDA5D885DEB1DDF0032010E468
91E16885DD6885DE60AD0A018D0801AD0B018D090160204AE92072B34C5EB02002
B5C04090034C3CB420FCB1F01B0820A4B4207AEA28100320A4B32017B2100320A4
B32072B34C5EB04C3BB4EE13018E1101C050B0308A48A200204AE268AAA5D19D00
01E8A5D29D0001E82002B5E00890E0C050B010A200204AE2A5D185C8A5D285C9EA
EAEAA20020CAE3BD010120E2E3BD000120E2E3E000D008A92D20A4E34C1BEBE004
F0F420D9E3E8E8E00890DBA5C920E2E3A5C820E2E320CAE3A5D420E2E3A5D320E2
E320D9E3A5D620E2E3A5D520E2E320CAE320E8B1205EB0209BB18E10012071E4AD
28018D1001C9EEF00ECD0A01F0092072EB208DB14C51EB4C5EB0A201BD00019D29
01B5D39D2B01CA10F2E86086EFC050B002E6EF2092EB4C5EB06CF00086EEC050B0
02E6EE20A3EB4C5EB06CF20020ACEB4C5EB06CEC0020D7EB2066A6B001602066A6
B0FB482010ECC90FD00CA90D200DECA90A200DEC6860C911D0E96860297F48A5E3
F00268606848C900F022C91BF01EC90DF01AC90AF016C907F012C908F00EC920B0
0A48A95E200DEC68186940200DEC68604C63A6A90085E3206FECC900F0F9C91AD0
034C5EB0C903D0034CA6B0C919D0034CF1EFC902D008A9808D53A64C00C0C907D0
034C7289C90FD003E6E360C914D020206FECC930F00FC931D015AD00A049808D00
A04C6CECAD0CA049208D0CA0A918602060A6297F48AD3301D00A6848C911F009C9
09F0056820D7EB606860A200B93801C920D0032044EDA90185BD201AE1B006204B
B44CFDBBA5C0F0F6B93501C920F006204BB44CFDBBADFF01CD1901F00AB010A220
204DB44CFDBBADFE01CD180190F0AD190148AD18014886BD8E8F0120FFB4C050B0
39C93BF035C928F0082043B468684CFDBBC82002B5C050B021B93501C929F01A8A
48A20020C1E068AAA5D19D9001A5D2E89D9001E8EE8F01D0D8A2006885DD6885DE
48A5DD482047BFA90185BEE6BCA5BCC9209008A2248E12014C4DB44C6ABCB93501
20A3B6DD86EDD02AB9360120A3B6DD87EDD01FB9370120A3B6DD88EDD014BD89ED
85E1BD8AED85E26868C8C8C8A2004C39B7E8E8E8E8BD86EDD0C1A20060494645
F8ED49464E08EE49465015EE49464D21EE5345544BEE2A2A2AE6ED2E454E63BB00
```

```
2E4D45EBED2E4D44F0ED2E454E63BB002C58291BBE292C591CBE002C58201EBE2C
59201DBE002C58200CBE2C59200BBE00234C2C7BBD23482C8CBD0086BF4CFDBB86
BB4CFDBBA229204DB44CFDBB202DEEA5D1D006A5D2D00286BF4CFDBB202DEEA5D1
D0F4A5D2F0F2D0EE202DEEA5D2300286BF4CFDBB202DEEA5D2100286BF4CFDBB84
BF2002B5204AE2AD1A01F0068C12012049B46084C14CFDBB86C14CFDBB201AE1B0
034C3AEEA5C0D008A22A204DB44CFDBB2002B5B93501C93DF0062043B44CFDBBA5
DE48A5DD48C8202FEE6885DD6885DEA000A5D191DDA5D2C891DD4CFDBBA5BCF00C
A5BED020A229204DB44CFDBBE6BBAD1301D00EA0006891DD6848C891DD88B1DD48
4CFDBBE6C2D009E6C3D005A22E204DB486BE86BDAD8F01F05BB93501C928F008A2
25204DB44C3CEF86E5C82002B5C93BF03AC050B036C929F03284E4201AE1B007A4
E42079BFB0DD84E4A000A6E5BD900191DDE8BD9001E8C891DDA4E486E5A200CE8F
0130BA20FFB44CDDEEAD8F01D0AF4CFDBBB93501C93BF0F6C050B0F2909FA5BCD0
0586BB4CFDBB86E6C6BC3011F00AC6C2A5C2C9FFD002C6C368684CFDBB86BCA22B
204DB44CFDBBA5EEF0072065EF606CF600208881A9FF20C4EF84FDA90920A58920
2E83209C82207B8CD8A9009002A9014CE5EFA5EFF0072092EF606CF400208881A9
0020C4EFADC58FC93FF00CA9018D30A620878ED84CE5EF20B68DA9078D02A4EE02
A4A201209A8ED84CE5EF20868B8D4EA6AD24018D4CA6AD25018D4DA6AD26018D4A
A6AD27018D4BA6A08060209C8B4CB88120B2B76CD10020B2B74C000020B2B74C03
00000000
```

Memory block $E000-$EFFF checksum: B20A

END OF DOCUMENT