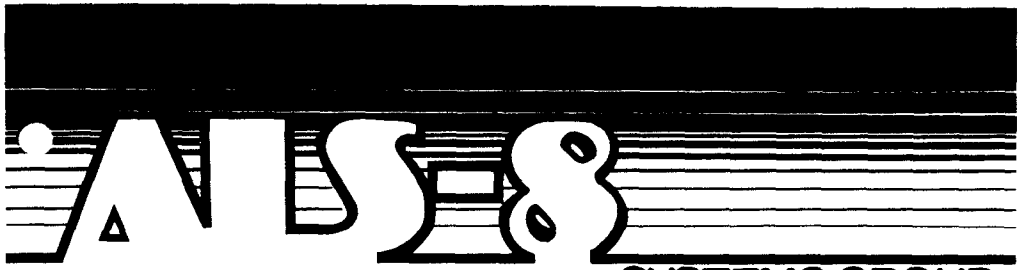


VOLUME ONE, NUMBER ONE MARCH 15, 1977

Drew Rogge and Ed Wyman
Co-Editors, Co-Authors and
Chief Cooks and Bottle Washers

P.O. Box 2072
Livermore, CA 94550

Well, here it is! The first ALS-8 SYSTEMS GROUP Newsletter. So, welcome to the Group, and let us tell you a little about ourselves. We're like most of you -- small system hackers. We're not professional writers or editors, so what you'll see might not be of the highest literary quality, but we will try to do our best to give you comprehensive answers to your questions, along with information on the ALS-8 system.

You may be wondering about the relationship of the ALS-8 SYSTEMS GROUP to Processor Technology. So, we'd like to tell you just how it works. We receive support from Processor Technology in the form of a dollar-for-dollar (minimum) matching of membership dues, assistance in typing and mailing of the Newsletter and, of course, the best part of the relationship: They are our source of information that enables us to pass along, to you, the whys and wherefores of the ALS-8.

Those whys and wherefores will consist mainly of:

-- where the utility subroutines are located in ALS-8,
-- which registers they use,
-- any other subroutines they call,
-- what data they require, and,
-- where that data needs to be located.

If we have enough time between issues, we will try to include, among other things, some routines that will enhance the editing capabilities of the TXT-2 Editor.

One of the relatively minor duties of the Newsletter will be to notify you of changes and improvements made to ALS-8 by Processor Technology. But more about that in future newsletters.

In order to make this newsletter the best it can possibly be, we'll need feedback from you, the user. The feedback can be suggestions on handling user supplied software, standardization of subroutine labels, or just comments on the contents of the Newsletter itself. Your comments and suggestions will help determine the contents of future issues, Although the suggestions will be greatly appreciated, we usually won't have much time for individual replies.

We are planning to publish as many of the user supplied programs as space allows. We feel that this is the most efficient way to distribute these programs, since listings are the least expensive and most practical method.

One thing you might consider now is if you'd like your name, address and telephone number included in a listing of ALS-8 owners. If you do, please drop us a line.

As of now, we plan to publish 10 or 11 issues per year. That will be very flexible, depending on the amount of material we receive and, most important, the amount of time we have.

Well, enough of this kind of stuff. It's on to the kind of things we think you've been waiting for.

MANUAL ERRORS

There are two errors and one ommission in the manual as it stands now. The maximum terminal width is 118 and not 12Ø, as it stated in the manual. Even though the input input buffer is 12Ø bytes long, two bytes need to be reserved for the carriage return and an end of file marker.

The item that was left out is an operator in the form of a "$". This operator is equivalent to the address that the next instruction will be assembled at. If we had an instruction such as:

```
JNZ $+3
```

And the next instruction was three bytes in length, then this would skip the next next instruction if the zero flag was reset while:

```
JMP $
```

would not accomplish anything, as it would just pass control to the next instruction. I suggest that you try using this operator in a variety of ways, as it can prove to be a very powerful tool.

PROGRAM ERRORS

Two minor errors have been found in the ALS-8 PROMS. The first one has to do with the 'TERM' command which sets the length of the input buffer. If input is made to the end of the system RAM including the custom command table is cleared. This is because the carriage return is treated as a character and there is no room left for it. The best way to get around this error (until it is corrected) is to set the terminal width to two more than the width of your input device.

The other error concerns the 'ASC' pseudo-op. When using a terminal width over 80 and when in the 'FORM' mode, the closing delimiter of the string is missed by the assembler. As a result, any comment that is on the same line will become part of the 'ASC' string. Since there seem to be a lot of variables that enter into this error, the best way to avoid an error is to use the unformatted mode when assembling programs that contain 'ASC' strings.

There is also an error in the TXT-2 Editor PROMS. If you use a CONTROL-H to delete the last character of a line that is exactly 64 characters long, the file structure of the line is altered. This error can be avoided by replacing the last character with a space when it needs to be deleted.

The above mentioned errors (term width, 'ASC', and CTRL-H) have been corrected in the PROMS from serial number 500 on. For those PROMS before the 500, these errors will be corrected during an update service that will be announced in the next newsletter.

'<' & '>' OPERATORS

Also included in this update will be the addition of the high half and low half operators to the assembler. These operators allow you to use either the H or L part of a 16 bit value assigned to a label as an eight bit value. Here's an example:

```
VDMBASE   EQU   ØCCØØH
          MVI   A,< VDMBASE
```

will load the accumulator with ØCCH. Where as:

```
POTTS     EQU   ØEØDFH
          MVI   E,>POTTS
```

will load register 'E' with ØDFH.

APPLICATION NOTES

Included in this newsletter is a collection of system application notes and user supplied routines. The notes are self-explanatory and it is suggested that you try some of the functions in the application notes to become familiar with them.

If we receive questions in the mail about some of the material contained in the notes, we will expand on the problem areas in future newsletters. One of the application notes shows the ALS-8 parameter area.

Included in this area is a 39 BYTE section that has been reserved for user parameters (D1A1H thru D1C8H). If there is enough interest in making this a global parameter area for all user supplied software, we will try to work up a definition for this area from your suggestions.

RANDOM COMMENTS

We would like to standardize subroutine, system oriented custom command, and storage area names in order to make routines that are distributed among ALS-8 users as universal as possible. Then the subroutine is in the ALS-8 PROMS, the label that was used in either the ALS-8 application notes or in the Newsletter should be used. That way, the label can be entered in the Global Symbol Table and be called without using a long string of equates at the end of each program. Any time a subroutine is needed in a program, try to use one that exists in ALS-8 even if it means having to save a register pair that is altered by the subroutine. What this all boils down to is: Let's make our routines as BYTE efficient and as universal as possible.

Since this is a Systems Group Newsletter, we are only interested in programs that are related to operating systems in general, and ALS-8 in particular. Games such 'LIFE', 'NIM', and 'STAR TREK' are fun, and definitely have

a place in the computer world, but they are not the concern nor are they the subject matter of this newsletter.

Programs that would be of benefit to members of this group could be routines, such as WRITE CURRENT FILE ON CASSETTE, special I/O drivers, or ones that, in general, enhance the ALS-8 operating system.

One item that we've thought of is software to be used with one of the semicommercial type tape transports, such as the PHI-DECK which would provide mass storage capability with many of the characteristics of disk. If anyone has used a high speed tape unit with ALS-8, we would like to hear about it so that the information can be passed on the the users group.

Does anyone have a need for some special drivers or other software in order to make ALS-8 do a particular job for them? If so, let us know what it is, and we'll pass the request on through the Newsletter. Someone else may have already solved the problem.

We hope that the Newsletter will benefit all ALS-8 users, and that we can have a little something for everyone.

Drew and Ed

Title ASSI Input Drivers

Function

Release Date: 9/3/76
Revision No. Ø
Level No. B2
Page 1 of 3

The ASSI command of the ALS-8 allows the assembler of the system to produce object code from files stored on mass storage devices such as disk, cassette or paper tape. While the type of assembly is generally slower (being I/O bound) it allows larger source programs than the more convenient memory files.

This note describes the requirements of the input driver used for the ASSI command. In all cases the term FILE is used in the context, "a collection of data" and should be interpreted as meaning the date for the assembly process.

ASSI INPUT DRIVER SPECIFICATION

ALS-8

Program Development System

The ASSI extension of the ALS-8 assembler frees the user from memory files and allows assemblies from paper tape, cassette, disk or any other extended storage medium. In order for the input driver to function with the assembler, it must bring in data from the device on a RECORD by RECORD basis.

This requirement is met by reading lines from the file on a line by line basis instead of character by character. In other words, the driver gets one whole line of input instead of one character as do standard drivers.

These lines are read into a memory location so that the data can be processed by the assembler. Whether or not the file has line numbers determines the exact location as follows:

Line numbered lines start at IBUF-5
Non-line numbered lines start at IBUF
where:

```
IBUF    EQU    ØD1E4H
```

Each line must be read into the proper memory location on an incrementing basis and a CR (13 decimal) must terminate the line.

Also, because the assembler performs the assembly in two passes, some provision must be made for rewinding the file when the "END" is encountered on each pass.

This "END" for a file must have been preceeded by a valid "END" pseudo in the source file so the assembler can change its internal pass indicator. How this end is synchronized to the actual physical end is dependent on the characteristics of the device reading the file.

Example

The file is on paper tape to be read by a high speed reader. The format of the tape is such that a leader and trailer of zero's were punched on the tape before and after the actual file code. Line numbers are present and the last line of the file is an END statement.

```
*
*       SAMPLE ASSI INPUT DRIVER
*
ENTRY    LXI    H,IBUF-5    FOR LINE NUMBERS
         LDA    SWCH1       FIRST TIME IN?
         ORA    A
         JNZ    ENTRY       NO IF NOT ZERO
         CALL   CHR         GET CHARACTER FROM READER
         ORA    A
         JZ     ENTRY       SCAN PAST LEADING ZEROS
*
         STA    SWCH1       MAKE IT NON ZERO
         JMP    GOTON       NOW PROCESS THE RECEIVED CHARACTER
*
DLOOP    CALL   CHR
GOTON    ORA    A           TEST FOR A ZERO
         JZ     REWIND      WE'RE AT THE END IF ZERO
         MOV    M,A
         CPI    CR          CARRIAGE RETURN?
         RZ                 RETURN TO ASSEMBLER IF SO
         CPI    ' '
         JC     DLOOP       BYPASS ALL CONTROL CHRS
         CPI    7FH
         JZ     DLOOP       BYPASS SYSTEM FILL CHRS
         INX    H
         JMP    DLOOP
*
*
REWIND   CALL   REWD        ROUTINE FOR DEVICE REWIND
         MVI    A,Ø
         STA    SWCH1       PRETEND IT'S THE FIRST TIME THROUGH
         JMP    ENTRY       GO SCAN PAST LEADING ZEROS AGAIN
*
*  CHR    THIS ROUTINE GETS ONE CHARACTER FROM THE
*          INPUT DEVICE
CHR      EQU    $
*
```

Title ALS-8 File Structure

Function

Release Date: 9/19/76
Revision No. Ø
Level No. A5
Page 1 of 3

The memory files of the ALS-8 System provide a convenient means of creating and modifying assembly language programs. Small to medium size programs can be assembled directly to memory and tested in a quick, efficient manner.

Sometimes, after a system crash the resident file may have been modified, making it unsuitable for use by the system file commands.

In order to directly examine and "fix" a file, some knowledge of the file structure is necessary. Also, special utilities can be written to search and modify files for higher level editing operations.

ALS-8 FILE STRUCTURE

ALS-8

Program Development System

System memory files are structured on a RECORD basis. That is, each line in the file is considered to be a RECORD whose length is contained in the first byte of the RECORD.

Most files used with the system contain assembly language programs, and in the example that follows this will be the case. The system itself looks only for a correct record length and correct record terminators with valid ASCII characters between them. It makes no assumptions as to file content until an assembly is attempted.

Each RECORD of a file is structured as follows:

RL	DATA	13

Where, RL indicates the number of characters in the RECORD including the RL and terminator.

13 is a decimal 13 value, ($ØD_{16}$), as a RECORD terminator.

DATA is any characters greater than 20_{16} in value.

Thus the file line: 5 LDA ABUF entered as a one line file would be stored in memory as follows:

16	Ø	Ø	Ø	5			L	D	A		A	B	U	F	13	1

Here the RL and terminator are given as decimal while the DATA is shown as ASCII. When dumped by the system dump command the RECORD would take the form:

```
ØØØ1: 1Ø 3Ø 3Ø 3Ø 35 2Ø 2Ø 4C 44 41 2Ø 41 42 55 46 ØD
ØØ11: Ø1
```

Since the file contains only this one RECORD, the RL of the next RECORD contains a 1. This value indicates the End of File and must always be present at the end of each file.

It is recommended that an actual file be created and viewed as a dump. Notice how the line numbers are evident by the string of Hex values between 3Ø and 39 and that the $ØD_H$ value identifies the end of each RECORD.

Title General Purpose System Output Routines
Function
Release Date: 10/1/76
Revision No. Ø
Level No. K2
Page 1 of 6

The ALS-8 System contains many routines for outputting data to the current I/O driver.

Each of these are described in the following pages along with the specifications for register usage.

SYSTEM OUTPUT ROUTINES

ALS-8

Program Development System

Title: CRLF

Function: Output a Carriage Return/Linefeed

Release Date: 10/2/76
Revision No. Ø
Level No. K2
Page 2 of 6

CRLF EQU ØE216H

This routine outputs a CRLF followed by two delete characters. Output is made to the current output driver and registers A and B are altered.

One level of stack is used by the routine.

Entry Point: E216

Title: SCRN - String Output Routine

Function: Output a String of Characters

Release Date: 10/5/76
Revision No. Ø
Level No. K2
Page 3 of 6

SCRN EQU ØE38ØH

This routine outputs a string of characters from incrementing memory until a decimal 13 is found. This string is assumed to be ASCII in a form suitable for the output device.

On entry, registers H & L should point to the first character of the string.

Registers A and B are altered on return and H & L will point to the termination character.

Example

```
        LXI     H,MESS      POINT TO MESSAGE
        CALL    SCRN
        JMP     DO IT
                .
                .
                .
                .
MESS    ASC     'ARE YOU THERE?' THE MESSAGE
        DB      13          THE TERMINATOR
```

SCRN also sets a system parameter XOUT whose use is described in an additional system bulletin.

Title: Data Output Routines

Function: Output Values in HEX, OCTAL or DECIMAL

Release Date: 10/5/76
Revision No. Ø
Level No. K2
Page 4 of 6

```
DUMO    EQU    ØE56FH
HOUT    EQU    ØE577H
HOTB    EQU    ØE586H
DOUT    EQU    ØE348H
OOUT    EQU    ØE353H
OOTB    EQU    ØE359H
```

Each of these routines is used to output the binary value in register A to the current output device as ASCII characters. In every case registers A,B and HL are altered.

```
DUMO    Output in accord with current mode
HOUT    Output as Hexadecimal
HOTB    As above but output a space following
DOUT    Output as Decimal
OOUT    Output as Octal
OOTB    Output as above but with space following
```

Up to two levels of stack may be used.

Title: ADOUT 16 Bit Value Output

Function:

Release Date: 10/5/76
Revision No. Ø
Level No. K2
Page 5 of 6

ADOUT EQU ØE55CH

This routine outputs the 16 bit value in registers D & E. The output is made in accord with the system mode and in all cases to the current output driver.

All registers are altered and the stack is used to two levels.

Title: BLKO Output Spaces

Function:

Release Date: 10/5/76
Revision No. Ø
Level No. K2
Page 6 of 6

BLKO EQU ØE361H

This routine outputs spaces to the current output driver. On call the number of spaces should be in register 'C' with zero giving 256 spaces.

Registers A,B and C are altered and the stack is used to one level.

Title System Conversion Routines

Function Binary to ASCII Conversion

Release Date 10/7/76
Revision No. Ø
Level No. K3
Page 1 of 2

The following Application Bulletins specify and describe the system subroutines for converting binary values to their ASCII representation.

SYSTEM BINARY TO ASCII CONVERSION

ALS-8

Program Development System

Title: Binary to ASCII Conversion

Function: _____

Release Date: 10/7/76
Revision No. Ø
Level No. K3
Page 2 of 2

```
BINH    EQU    ØE39ØH
BIND    EQU    ØE3ADH
BINO    EQU    ØE3C8H
```

```
BINH    Binary to Hexadecimal
BIND    Binary to Decimal
BINO    Binary to Octal
```

In each routine the value in Register 'A' is converted to ASCII characters which are then stored in memory.

Registers A,B and HL are altered by the routines and one level of the stack is used.

On return, the three byte memory area, HCON, will contain the ASCII characters on an incrementing MSB to LSB basis. Hex conversion produces two ASCII characters while both Decimal and Octal produce three.

Title: SEAR String Search and Compare

Function: _____

Release Date: 11/10/76
Revision No. Ø
Level No. K4
Page 1 of 1

SEAR EQU ØE257H

This subroutine checks two character strings for equality. The strings are pointed to by HL and DE. On call the desired length of the comparison should be in Register 'C' and on return DE and HL will point to the next address after the length of the compare.

If the strings are identical the zero flag will be set.

```
        * DE HAVE OTHER ADDRESS
          LXI     H,YMES
          MVI     C,3           LENGTH OF COMPARE
          CALL    SEAR          DO COMPARE
          JNZ     NEXT
          MOV     A,M           GET FIRST ADDRESS
          INX     H
          MOV     L,M
          MOV     H,A
          PCHL                  BRANCH TO YES
YMES      ASC     'YES'
          DW      YES
NEXT
                  •
                  •
                  •
                  •
                  •
```

Title System Conversion Routines

Function ASCII to Binary

Release Date: 10/15/76
Revision No. Ø
Level No. K1
Page 1 of 3

The following Application Bulletin specifies and describes the System subroutines for converting ASCII parameters to their binary values.

In all cases the routine returns with a standard RET and errors must be handled by the calling program.

SYSTEM ASCII TO BINARY CONVERSION ROUTINES

ALS-8

Program Development System

The ALS-8 contains subroutines for conversion of ASCII parameters to binary values. The following routines are available:

NAME	
EMODE	ASCII to Current Mode Setting
ADEC	ASCII to Decimal
AHEX	ASCII to Hexadecimal
AOCT	ASCII to Octal

```
EMODE   EQU   ØE2F3H
ADEC    EQU   ØEØØAH
AHEX    EQU   ØE2FAH
AOCT    EQU   ØE333H
```

On entry, registers B,C must point to the first ASCII digit and the routine scans incrementing memory for additional digits. The scan stops when a binary zero is found.

On return, H & L have the converted value, and carry is set if an error was detected.

Release Date 10/15/76
Revision No. Ø
Level No. K1
Page 3 of 3

Title: System Conversion Routines

Function: _____

Entry Point: As Specified

Entry Conditions

A	H		
B High byte pointer	L		
C Low byte pointer	Flags	S	P
D		CY	AC
E		Z	

Exit Conditions

A	H High order value		
B Through scan	L Low order value		
C Through scan	Flags	S	P
D		CY Error flag	AC
E		Z	

Buffer Area used: _____ to _____

Registers Altered: All

Subroutines called:

_____ _____

_____ _____

_____ _____

Subroutines levels used: _____

Stack level used: 1

Coments: _____

Title ALS-8 Parameter Passing

Function _____

Release Date 11/1/76
Revision No. Ø
Level No. B1
Page 1 of 5

This bulletin describes the techniques for passing parameters to commands, routines or programs external to the ALS-8.

While sufficient information is given for normal usage, the experienced programmer is also referred to the parameter conversion bulletins.

PARAMETER PASSING FROM THE ALS-8 SYSTEM

ALS-8

Program Development System

When a command is received by the ALS-8, all parameters given with the command are stripped off and placed in "buffers" for the command to use. Whether or not the parameters are required, how they are interpreted and what function they perform depends strictly on the command itself. The ALS-8 serves only to locate the parameters in a specific place and to provide routines to process or convert the parameters into a usable form.

Name Extension

The syntax requirements of the system require only four characters for command identification. The command itself can use any additional characters for its own use as with the ASSM, ASSME, ASSMX and ASSMS commands. With this system command the ALS-8 reacts to the 'ASSM' while the assembler itself uses the fifth character within its own code.

ASCII Parameters

In addition to direct name extension the ALS-8 also allows up to five ASCII characters, enclosed within slashes (/), to be passed. The system FILE and IODR are examples of ASCII parameter use.

Numeric Values

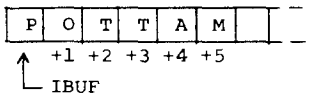
The system also allows two numeric values to be input with the command. These may be interpreted as HEX, OCTAL or DECIMAL values by the external routine.

How to locate these parameters is specified here, and some techniques for processing them are explained in detail sufficient for normal operations.

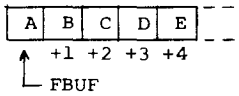
As an example, assume the entry "POTT" were entered as a custom command along with the address 1Ø for its execution. Assume the ALS-8 received the command:

POTTAM /ABCDE/ 1057 A1E5

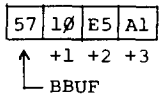
The system would preprocess the parameters and then make a programatic call to address 1Ø with the buffers set as follows:



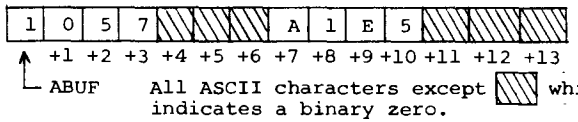
ALL ASCII CHARACTERS



ALL ASCII CHARACTERS



Binary values in low byte, high byte format



All ASCII characters except ▨ which indicates a binary zero.

With this example the values in the BINARY BUFFER, BBUF, assume the system MODE was set to hexadecimal. The system, prior to branching to the command, converted the values to binary and stored them as such in BBUF as received. If the system mode had been octal in this case, the first parameter would have been converted to its binary equivalent, but the second would have caused an error. The error is present because the characters "A" and "E" are not legal for the octal mode. The system would NOT have gone to the command in this case. It would have output the error message "WHAT?" instead.

If the custom command had wanted the values converted in accord with the system mode, the LHLD instruction should be used for direct loading of the values.

```
LHLD    BBUF        GETS FIRST PARAMETER
LHLD    BBUF+2      GETS SECOND PARAMETER
```

There are times where a value is always expected as Decimal, Octal or Hexadecimal. In this case the ASCII values in ABUF can be converted by using the appropriate ALS-8 conversion routine. The conventions for these routines require that they be called with registers B & C pointing to the first ASCII character and that a binary zero terminate the sequence. (See the corresponding system bulletin for further information.) If we wanted each parameter to be interpreted as Decimal, the following sequence would be used:

```
CALL    VCHK        MAKE SURE THE FIRST IS PRESENT
LXI     B,ABUF
CALL    ADEC        CONVERT VALUE
JC      WHAT        CARRY SET MEANS ERROR
SHLD    PARA1       STORE FIRST
LXI     B,ABUF+7
LDAX    B           GET CHR OF SECOND PARAMETER
ORA     A           SET FLAGS
JZ      WHAT        NO PARAMETER
CALL    ADEC        CONVERT SECOND
JC      WHAT        ERROR
        •
        •
        •
        •
        •
```


The command name extensions, as in the "AM" added to the example command "POTT", are best picked up with a direct LDA instruction.

```
LDA     IBUF+4      GET FIFTH CHR
CPI     'S'         IS IT S?
JZ      SONE        GO DO THE "S" PART
CPI     'A'
JNZ     WHAT        ONLY A OR S FOR THE EXAMPLE
        •
        •
        •
        •
```

The characters in FBUF are used to provide a name for the command functions that follow. This can be the name of a File or I/O Driver as with the ALS-8 Command Set or some other interpretation as per the requirements of a Custom Command.

A command, GETF, for example, could read a cassette tape until the specified name was found and then read it into memory locations starting at the first parameter but not to exceed the second.

Also, a command, Save, could search the file tables for the file name, pick up the file parameters and then save them on tape. The name could also indicate the desired I/O Driver for use with the command execution. In this case the I/O Driver Table would be searched to find the correct value.

The use of the File and I/O Driver search is more complex than parameter passing. Many options are available and fertile minds are referred to the bulletin describing the available routines.

```
IBUF    EQU     ØD1E4H      INPUT BUFFER
ABUF    EQU     ØDØDFH      ASCII BUFFER
FBUF    EQU     ØDØD7H      NAME BUFFER
BBUF    EQU     ØDEFH       BINARY BUFFER


VCHK    EQU     ØE51CH
WHAT    EQU     ØE7DDH
```

Title: RAND2 Eric G. Rawson 11/20/76

Function: 15-bit Random Number Generator

Many software algorithms exist for random number generation. Most involve multiplication and truncation. None generate truly random numbers, but instead give a finite sequence of numbers (which may be very long) which then repeats exactly, over and over. Such numbers are called pseudo-random numbers (PRN). A good PRN generator gives numbers which are very irregularly positioned within its given numerical range, and generates all possible numbers within that range before repeating. For example, a PRN generator having a range of 0.0 to 1.0, and claiming 6 significant figures, should generate all 10^6-1 numbers (from .000,001 to .999,999) before the sequence repeats. Unfortunately, not all algorithms generate as complete a set of numbers as their precision implies. The program described here uses a software analogue of the electronic technique for generating maximal length bit sequences (m-sequences) in the field of data transmission. An n-bit shift register, whose first bit input is derived from an exclusive-or of the nth bit and one other suitable bit (often n-1 or n-2) generates an m-sequence 2^n-1 bits long. Not as widely known is the fact that, in so doing, the n bits in the shift register present a complete set of PRN's. Each shift generates a new PRN, and the sequence cycles thru all 2^n-1 values before repeating exactly.

In this program (Figure 1) the carry bit is set equal to the parity of bits 1 and 2 of the least significant byte (bit zero is always zero) and RAR operations effect the required right shift. The resulting sequence is complete but retains some near-neighbor correlation due to the shift mechanism used. The latter is eliminated by skipping thru the sequence 23 numbers at a time; this preserves the sequence length and its irregularity while eliminating the short-term correlations.

Figure 2 is a test program which illustrates the use of the program by listing the first 150 PRN's provided by RAND2.

Figure 3 is a test program which confirms that the sequence is 2^{15}-1 ($=7FFE_H$) values in length, and determines that the execution time per number is ~920 μsec.

Figure 4 shows a test of the evenness of the distribution of PRN's. The numbers are sorted and counted in 256 "bins," according to the value the most significant byte. When the complete sequence is so sorted (count limit = $7FFF_H$), as shown in the first execution (Figure 5) each slot contains the required 80_H values except the last slot from which word FFFE is missing. In the second execution, only 1/4 of the possible values are sampled (count limit = $1FFF_H$). It is apparent that the values cluster around the proper average value (20_H), and range from 11_H to $2E_H$, confirming the evenness of the number distribution.

```
0800                    0005 * RAND2
0800                    0010 *
0800                    0015 *    WRITTEN BY ERIC G. RAWSON, 11/20/76
0800                    0020 *
0800                    0025 *    RAND2 IS A 15 BIT PSEUDO RANDOM
0800                    0030 *    NUMBER (PRN) GENERATOR. EACH
0800                    0035 *    PRN IS OBTAINED FROM THE PREVIOUS
0800                    0040 *    ONE BY SETTING CARRY=PARITY OF BITS
0800                    0045 *    1 AND 2 OF THE LEAST SIGNIF. BYTE
0800                    0050 *    (BIT 0 IS ALWAYS ZERO) AND DOING
0800                    0055 *    TWO RAR'S ON THE TWO BYTE WORD.
0800                    0060 *    STEPPING BY 23 REMOVES NEAR NEIGHBOR
0800                    0065 *    CORRELATIONS AND STILL YIELDS A PRN
0800                    0070 *    IN UNDER 920 MICROSECONDS. FASTER
0800                    0075 *    EXECUTION CAN BE OBTAINED BY REPLACING
0800                    0080 *    23 IN LINE 145 BY 19,17,13,11, OR EVEN 1,
0800                    0085 *    FOR WHICH THE CALL TIME DROPS TO 110 MICRO
0800                    0090 *    SEC AT THE COST OF INCREASING NEAR-NEIGHBOR
0800                    0095 *    CORRELATIONS.  ALL 32,767 15-BIT PRN'S
0800                    0100 *    ARE GENERATED, AND THEN THE CYCLE REPEATS.
0800                    0105 *    FFFE(H) DOES NOT OCCUR.
0800                    0110 *
0800                    0115 *    TO RANDOMISE FROM RUN TO RUN, RESET PRNUM
0800                    0120 *    USING ALS-8 COMMAND ENTR, PERHAPS USING THE
0800                    0125 *    DATE OR TIME. ODD VALUES ARE EQUIVALENT TO
0800                    0130 *    (VALUE)-1. DON'T USE FFFF(H) OR FFFE(H).
0800                    0135 *
0800 C5                 0140        PUSH   B
0801 06 17              0145        MVI    B,23    SET LOOP COUNTER
0803 2A 22 08           0150        LHLD   PRNUM   GET LAST PRN
0806 54                 0155        MOV    D,H     SAVE IT
0807 5D                 0160        MOV    E,L
0808 7D                 0165        MOV    A,L
0809 E6 06              0170 LOOP   ANI    6       GET BITS 1,2
080B 37                 0175        STC    .       CARRY=PARITY
080C EA 10 08           0180        JPE    CONT
080F 3F                 0185        CMC
0810 7C                 0190 CONT   MOV    A,H     SHIFT H
0811 1F                 0195        RAR
0812 67                 0200        MOV    H,A
0813 7D                 0205        MOV    A,L     NOW L
0814 1F                 0210        RAR
0815 E6 FE              0215        ANI    0FEH    STRIP BIT 0
0817 6F                 0220        MOV    L,A     A NEW PRN
0818 05                 0225        DCR    B
0819 C2 09 08           0230        JNZ    LOOP    DO 23 TIMES
081C 22 22 08           0235        SHLD   PRNUM   RESAVE PRN
081F EB                 0240        XCHG   .       PUT TO D,E
0820 C1                 0245        POP    B
0821 C9                 0250        RET
0822 C8 25              0255 PRNUM  DW     25C8H   AS GOOD AS ANY STARTER

CONT    0810    0180
LOOP    0809    0230
PRNUM   0822    0150 0235
```

Figure 1. Assembly listing of RAND2.

```
ASSMX B00

0B00                    0005 * TST3  LIST FIRST 150 NUMBS OF RAND2
0B00 CD 16 E2           0010        CALL   CRLF
0B03 0E 0F              0015        MVI    C,15    LINE CNTR
0B05 06 0A              0020        MVI    B,10    WORD CNTR
0B07 CD 00 08           0025 LOOP   CALL   RAND2
0B0A C5                 0030        PUSH   B
0B0B CD 5C E5           0035        CALL   ADOUT   ALS-8 4HEX OUTPUT
0B0E C1                 0040        POP    B
0B0F 05                 0045        DCR    B
0B10 CA 1D 0B           0050        JZ     CRTN
0B13 C5                 0055        PUSH   B
0B14 06 20              0060        MVI    B,20H
0B16 CD A9 D0           0065        CALL   OUTP8   ALS-8 CHAR OUTPUT
0B19 C1                 0070        POP    B
0B1A C3 07 0B           0075        JMP    LOOP
0B1D C5                 0080 CRTN   PUSH   B
0B1E CD 16 E2           0085        CALL   CRLF    ALS-8 CR + LF SUBR
0B21 C1                 0090        POP    B
0B22 06 0A              0092        MVI    B,10    RESET WORD CNTR
0B24 0D                 0095        DCR    C
0B25 C2 07 0B           0100        JNZ    LOOP
0B28 C3 60 E0           0105        JMP    EORMS   DONE
0B2B                    0110 RAND2  EQU    800H

CRTN    0B1D    0050
LOOP    0B07    0075 0100
RAND2   0800    0025

EXEC B00

A248 928C E824 E962 6262 82AC FC3C 667C 0EAA 7FF6
FA3E A7F8 FC8A 2B7C 2EC0 4746 B41A 7110 5536 C200
BFDC EA1E B3E0 7394 47B4 D39A 4144 759E 8330 01BC
16FE 5F62 A28E E98C 0862 C2F2 D85C DA4A 9348 32A4
F9D4 2F7A ADC6 A484 2D08 DA44 9548 F2A0 FBF4 7B78
2C38 6444 55A8 9E00 3FAE DD5E FBCC 6078 CC2E 6DD4
AFA4 D906 5CCA 038C 48FC 9E92 642E 6D28 CE24 2956
7E42 16BE 7762 A2B2 F00C F876 CAFA DD50 FDCC A07C
CE0E 39D6 2E5A 19C6 246A 47C8 E21A B16C 6416 7628
2E32 20C6 844E 51B8 1406 7C60 433C 069C 62FA DDAC
9C4C 502C EC86 2CE4 11C4 25E6 B048 D296 E6C4 456A
E798 C0EA 975E BBA2 5A18 B008 FA96 E6F8 5CEA 178C
C8E2 9252 9C24 692C 4CA2 3A94 E7D8 E8EA 9762 A222
AA0C B800 FF9A C17E 6FDE EB26 08E0 9372 2824 E9C2
2662 02CA C3FC 7E5C 1ABE F768 A6F2 580A FB8A 4B78

READY
```

Figure 2. A test program which lists the first 150 PRN's

```
ASSMX 4100

4100                    0005 * COUN2,  COUNT PRN'S BEFORE SEQUENCE REPEATS
4100 CD 16 E2           0010         CALL   CRLF    ALS-8 CR + LF SUBR
4103 21 00 00           0015         LXI    H,0
4106 22 22 08           0020         SHLD   PRNUM   PRESET PRNUM TO 0000
4109 01 00 00           0025         LXI    B,0     ZERO COUNTER
410C CD 00 08           0030 LOOP    CALL   RAND    GET ONE
410F 7A                 0035         MOV    A,D     IS IT ZERO?
4110 B3                 0040         ORA    E
4111 CA 18 41           0045         JZ     DONE    IF SO,QUIT
4114 03                 0050         INX    B
4115 C3 0C 41           0055         JMP    LOOP
4118 50                 0060 DONE    MOV    D,B     PRINT COUNT
4119 59                 0065         MOV    E,C
411A CD 5C E5           0070         CALL   ADOUT   USING ALS-8 4HEX OUTPUT SUBR
411D C3 60 E0           0075         JMP    EORMS   RET TO ALS-8
4120                    0080 RAND    EQU    800H
4120                    0085 PRNUM   EQU    822H

DONE    4118    0045
LOOP    410C    0055
PRNUM   0822    0020
RAND    0800    0030

EXEC 4100
7FFE
READY
```

elapsed time = 30.2 sec. Thus, time per number $\leq \frac{30.2}{32,767}$ $\leq 920\,\mu sec$

correct count:

$7FFE_H = 2^{15} - 1 = 32,767$

Figure 3. This program checks the length of the PRN sequence, and shows that the execution time per number is $\leq$920 µsec.

```
ASSMX 3320

3320                    0005 * RMAP, COUNT OCCURENCES OF ALL 256 MOST
3320                    0010 *     SIGNIFICANT 8 BITS. STORE COUNTS IN 3400(H)
3320                    0015 *     TO 34FF(H).
3320                    0020 *
3320 21 00 00           0025         LXI    H,0
3323 22 22 08           0030         SHLD   PRNUM  PRESET PRNUM TO 0000
3326 21 00 34           0035         LXI    H,3400H
3329 01 FF 7F           0040         LXI    B,7FFFH  SET COUNT LIMIT
332C 36 00              0045 PT1     MVI    M,0     ZERO 3400-34FF
332E 23                 0050         INX    H
332F 7D                 0055         MOV    A,L
3330 FE 00              0060         CPI    0
3332 C2 2C 33           0065         JNZ    PT1
3335 CD 00 08           0070 PT2     CALL   RAND2
3338 26 34              0075         MVI    H,34H   RESET H
333A 6A                 0080         MOV    L,D     SET L TO MSBYTE OF PRN
333B 34                 0085         INR    M       INCREMENT COUNT
333C CA 48 33           0090         JZ     OFLOW   MUST NOT EXCEED FF(H)
333F 0B                 0095         DCX    B       DECR LOOP COUNT
3340 78                 0100         MOV    A,B     IS LOOP COUNT ZERO?
3341 B1                 0105         ORA    C
3342 C2 35 33           0110         JNZ    PT2     IF NOT, LOOP
3345 C3 60 E0           0115         JMP    EORMS   DONE, RET TO ALS-8
3348 21 4E 33           0120 OFLOW   LXI    H,OFMSG
334B C3 E0 E7           0125         JMP    MESS    ALS-8 MSG RETURN
334E 20 4F 56 45        0130 OFMSG   ASC    " OVERFLOW"
     52 46 4C 4F
     57
3357 0D                 0135         DB     13      TERMINATOR
3358                    0140 RAND2   EQU    800H
3358                    0145 PRNUM   EQU    822H

OFLOW   3348    0090
OFMSG   334E    0120
PRNUM   0822    0030
PT1     332C    0065
PT2     3335    0110
RAND2   0800    0070
```

Figure 4. Checks the evenness of the PRN distribution by sorting into 256 number "bins." See executions in Fig. 5.

```
CUSTE /RMAP/3320
RMAP
READY
DUMP 3400 34FF
3400: 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
3410: 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
3420: 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
3430: 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
3440: 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
3450: 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
3460: 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
3470: 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
3480: 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
3490: 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
34A0: 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
34B0: 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
34C0: 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
34D0: 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
34E0: 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
34F0: 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 7F

ENTR 332B
1F/

RMAP
READY
DUMP 400 34FF+++++++++
WHAT?
DUMP 3400 34FF
3400: 28 22 22 2B 1D 23 25 22 2E 27 21 1F 1A 25 1E 1D
3410: 21 26 25 22 1C 1D 20 29 1F 1E 21 1D 1F 20 25 17
3420: 1E 1C 23 1E 1C 22 1F 21 1F 18 17 20 16 21 21 22
3430: 1F 29 1F 11 20 1F 24 1E 24 2B 27 1E 1D 21 23 28
3440: 16 20 21 1C 25 20 21 1E 1E 25 1B 27 1F 1F 24 1F
3450: 20 24 1F 1F 18 25 1B 21 1E 1C 24 1F 1F 15 26 20
3460: 21 23 1B 1C 20 26 24 19 20 24 1E 16 21 24 1F 20
3470: 25 21 24 25 27 2A 1D 18 1D 24 1F 22 20 2B 26 21
3480: 21 20 19 22 21 1A 23 1A 19 1B 20 21 1E 1C 20 16
3490: 19 22 20 1C 1B 19 2A 1E 24 15 20 16 23 21 25 1C
34A0: 17 1C 29 21 27 20 24 1B 25 20 22 21 1E 22 20 19
34B0: 2C 2D 19 1B 21 1F 20 1F 1A 22 26 15 29 1D 27 20
34C0: 1C 1D 1E 1C 1D 19 1C 1E 22 22 22 21 24 19 1E 23
34D0: 1B 1C 24 1E 21 18 26 20 25 20 21 28 1C 21 1F 25
34E0: 1D 1C 23 2B 25 1A 20 1F 1D 20 1D 22 20 25 22 1A
34F0: 17 22 24 26 25 1B 24 1C 1F 1E 1E 1E 26 24 1C 1C
```

Figure 5. Execution of RMAP (Fig. 4) for a full sequence (top) and a 1/4-length sequence (bottom). For the full sequence, all possible even numbers occur, except $FFFE_H$ is missing from the topmost bin. For the 1/4-sequence, the numbers average 20_H, and range from 11_H to $2E_H$.

The following Video Display Module Driver was submitted by:

Don Minlitch
29 Hoyt Street
Stamford, CT 06905


```
C900                    1000 *
C900                    1010 *VDM - VIDEO DISPLAY MODULE DRIVER
C900                    1020 *
C900                    1030 *ON ENTRY
C900                    1040 * B=CHARACTER TO BE OUTPUT
C900                    1050 *
C900                    1060            COM     VDM
C900 E5                 1070 VDM        PUSH    H
C901 2A 40 C9           1080            LHLD    VDMPT           GET NXT CHAR ADDR
C904 78                 1090            MOV     A,B
C905 77                 1100            MOV     M,A             INSERT THE CHAR
C906 FE 0D              1110            CPI     CR              IS IT CR CHAR
C908 CA 18 C9           1120            JZ      VDMCR           YES
C90B FE 20              1130            CPI     ' '             IS IT OTHER CTL CHAR
C90D DA 25 C9           1140            JC      VDMRT           YES - IGNORE IT
C910 FE 7F              1150            CPI     DEL             IS IT DEL CHAR
C912 CA 25 C9           1160            JZ      VDMRT           YES - IGNORE IT
C915 C3 1C C9           1170            JMP     VDMNX           GOTO CALC NXT CHAR
C918 7D                 1180 VDMCR      MOV     A,L             CALC NXT LINE ADDR
C919 F6 3F              1190            ORI     3FH
C91B 6F                 1200            MOV     L,A
C91C 23                 1210 VDMNX      INX     H               GET NXT CHAR ADDR
C91D 7C                 1220            MOV     A,H
C91E FE D0              1230            CPI     VDPAG+4         IS IT OFF THE VDM
C920 DA 25 C9           1240            JC      VDMRT           NO
C923 26 CC              1250            MVI     H,VDPAG         SET TO START OF VDM
C925 22 40 C9           1260 VDMRT      SHLD    VDMPT           SAVE CHAR ADDR
C928 36 8D              1270            MVI     M,CR+80H        INSERT CURSOR
C92A 29                 1280            DAD     H               CALC SCREEN OFFSET
C92B 29                 1290            DAD     H
C92C 7C                 1300            MOV     A,H
C92D 3C                 1310            INR     A
C92E E6 0F              1320            ANI     0FH
C930 D3 C8              1330            OUT     VDMIO           SET SCREEN OFFSET
C932 C5                 1340            PUSH    B
C933 DB FF              1350            IN      0FFH            SEE IF HARDCOPY WANTED
C935 0F                 1360            RRC
C936 9F                 1370            SBB     A
C937 A0                 1380            ANA     B
C938 47                 1390            MOV     B,A
C939 CD A9 D0           1400            CALL    SYSOU           COPY CHAR OR NULL
C93C C1                 1410            POP     B
C93D E1                 1420            POP     H
C93E 78                 1430            MOV     A,B
C93F C9                 1440            RET     .               RETURN
C940 00 CC              1450 VDMPT      DW      VDPAG*256
```

```
VDM     C900
VDMCR   C918    1120
VDMNX   C91C    1170
VDMPT   C940    1080 1260
VDMRT   C925    1140 1160 1240

SYML
PSW     0006    TIP     CB00    TOP     CB0A    INCH    D0CC
OUCH    D0CF    PRMC    D0D7    PRMD1   D0E0    PRMD2   D0E6
PRMB1   D0EF    PRMB2   D0F1    BUF     D1E4    STACK   D12F
CRLF    E216    SWCH1   D0FD    SWCH2   D0FE    EOR     E0B7
EORNS   E0D1    EORMS   E060    WHAT    E7DD    MESS    E7E0
CR      000D    LF      000A    HT      0009    FF      000C
BS      0008    BEL     0007    STX     0002    ETX     0003
DEL     007F    CAN     0018    ESC     001B    VDPAG   00CC
VDMIO   00C8    VDM     C900    SYSIN   D098    SYSOU   D0A9

D3FC
```


The following Utility Routines for use with the ALS-8 were submitted by:

Bill Gunn
875 West Broadway
Vancouver, B.C. V5ZlJ9

UTILITY ROUTINES TO BE USED WITH THE ALS-8
CASR, CASW, RENUM, INSRT, NUMB

ALL CASSETTE ROUTINES ARE SPECIFICALLY FOR THE TARBELL CASSETTE INTERFACE.

CASR: CASSETTE READS DATA FROM TAPE & PUTS IT IN MEMORY

CASR
-AFTER A FILE HAS BEEN INITIALIZED THIS COMMAND WILL READ IN A SOURCE FILE AND UPDATE END OF FILE POINTER (EOFP) AS WELL AS MAXIMUM LINE NUMBER (MAXL).

CASR XXXX
-READS DATA FROM CASSETTE AND DUMPS IT INTO MEMORY AT SPECIFIED STARTING ADDRESS XXXX. USED IN CONJUNCTION WITH CASW XXXX YYYY (IE. FIRST TEST TWO BYTES OF FILE ARE LENGTH-1 OF REMAINING FILE)

CASR XXXX YYYY
-THIS COMMAND IS USED FOR FILES THAT HAVE BEEN BLASTED ONTO TAPE WITH NO HEADER INFORMATION. XXXX IS THE STARTING ADDRESS TO WHICH THE DATA WILL BE DUMPED IN MEMORY & YYYY IS THE LENGTH OF THE DUMP.

IF THE CHECKSUM CALCULATED DURING THE READ DOES NOT AGREE WITH THE CHECKSUM WRITTEN ON THE TAPE AT THE END OF THE FILE A "READ ERROR" MESSAGE WILL BE PRINTED ON THE SYSTEM OUTPUT DEVICE. OTHERWISE A "READ COMPLETE" WILL BE PRINTED.

CASW: CASSETTE WRITES DATA ON TAPE FROM MEMORY

CASW
-WRITES THE CURRENT SOURCE FILE ONTO CASSETTE. THE FIRST 4 BYTES WRITTEN ARE THE MAX LINE NO. THE FOLLOWING 2 BYTES ARE THE LENGTH-1 OF THE FILE.

CASW XXXX YYYY
-WRITES A BLOCK OF DATA FROM MEMORY STARTING AT ADDRESS XXXX TO ADDRESS YYYY.

"WRITE COMPLETE" WILL BE WRITTEN ON THE CURRENT SYSTEM OUTPUT DEVICE UPON COMPLETION OF WRITE TO CASSETTE.

```
RENUM: RENUMBER A SOURCE FILE
     RENUM XXXX YYYY
          -THIS ROUTINE WILL RENUMBER BY ANY GIVEN
           INCREMENT.  THE INCREMENT MUST BE SPECIFIED
           & MAY PUSH A LINE NUMBER PAST THE MAX 9999
           VALUE IN WHICH CASE THE FIFTH DIGIT IS DROPPED
           XXXX IS THE MANDATORY INCREMENT VALUE WHILE
           YYYY IS THE OPTIONALLY SPECIFIED FILE LINE
           NUMBER THAT RENUMBERING WILL START AT.

           THIS ROUTINE IS GOOD FOR CREATING SPACE IN YOUR
           SOURCE FOR ADDING OTHER ROUTINES

INSRT: INSERT OR CONCATENATE TWO FILES
     INSRT /FNAME/ XXXX
          -IF YOU CREATE YOUR SOURCE IN A MODULAR FORM
           AND WISH TO ADD ROUTINES TO A SOURCE BODY
           THEN MAKE ONE FILE CURRENT SPECIFYING THE FILE
           TO BE INSERTED AS FNAME FOLLOWED BY THE LINE
           NO.  IN THE CURRENT FILE THE INSERTED FILE WILL
           FOLLOW.

           DANGER:  IT'S EASY TO RUN INTO VULNERABLE
                    MEMORY WITH THIS COMMAND.

 NUMB: AUTOMATIC LINE NUMBERING FOR INPUT MODE
      NUMB XXXX
          -THIS ROUTINE WILL CHANGE INPUT DRIVERS TO
           ALLOW THE AUTOMATIC PRINTING OF A 4 DIGIT
           LINE NUMBER FOLLOWED BY A SPACE.  XXXX WILL BE
           THE FIRST LINE NUMBER PRINTED AND EACH
           SUCCEEDING LINE NUMBER, INITIATED BY A CARRIAGE
           RETURN, WILL BE INCREMENTED BY 0005.

      NUMB XXXX YYYY
          -SAME AS ABOVE EXCEPT YOU SPECIFY THE INCREMENTING
           VALUE IN YYYY.

           A CARRIAGE RETURN AFTER AN AUTOMATIC NUMBER HAS
           BEEN PRINTED RESULTS IN THAT NUMBER BEING DELETED
           FROM THE FILE.

           TO EXIT FROM NUMB DO A CONTROL X AND READY WILL
           BE PRINTED.
```



```
0010 *
0020 *
0030 *        UTILITY ROUTINES TO BE USED WITH THE ALS-8
0040 *     ASSEMBLED ON THE ALS-8 PROGRAM DEVELOPMENT SYSTEM
0050 *     WRITTEN BY:  BILL GUNN
0060 *                  875 WEST BROADWAY
0070 *                  VANCOUVER, B.C.       V5Z 1J9
0080 *                  AREA CODE 604  879-0574
0090 *
0100 *     AUGUST 16,1976
0110 *
0120 *   CASR:  CASSETTE (TARBELL) READ
0130 *        3 MODES - 1.  SPECIFY NO ARGUMENTS, READ SOURCE
0140 *                  2.  SPECIFY FIRST ARG, READ BINARY WITH
0150 *                      LENGTH HEADER ON TAPE
0160 *                  3.  SPECIFY BOTH ARGS, READ BINARY
0170 *
0180 *   CASW:  CASSETTE WRITE
0190 *        2 MODES - 1.  SPECIFY NO ARGUMENTS, WRITE SOURCE
0200 *                  2.  SPECIFY FIRST ARG, WRITE BINARY WITH
0210 *                      LENGTH HEADER ON TAPE
0220 *
0230 *   RENUM:  RENUMBER A SOURCE FILE
0240 *        2 MODES - 1.  SPECIFY FIRST ARG, RENUM FROM THIS
0250 *                      EXISTING LINE NO.  BY INCS OF 0005
0260 *                  2.  SPECIFY BOTH ARGS, SEC ARG DEFINES
0270 *                      INC SIZE
0280 *
0290 *   INSRT:  CONCATENATE OR INSERT FILES INTO ONE ANOTHER
0300 *        1 MODE - 1.  SPECIFY FILE TO INSERT INTO CURRENT
0310 *                     FILE & LINE NO.  INSERTED FILE IS TO
0320 *                     FOLLOW IN CURRENT FILE
0330 *
0340 *   NUMB:  AUTOMATIC LINE NUMBERING FOR INPUT MODE
0350 *        2 MODES - 1.  SPECIFY FIRST ARG, LINE NO.  BEGINS
0360 *                      AT THIS NO.  & CONT.  WITH INC OF 0005
0370 *                  2.  SECOND ARG SPECIFYS INC VALUE
0380 *
0390 *
0400 * CASSETTE INPUT ROUTINE FOR SOURCE & BINARY
0410 *
0420 CASR   CALL   CRLF
0430        MVI    A,10H  RESET INTERFACE
0440        OUT    CASC
0450        CALL   ARG1   ANY ARGUMENTS?
0460        JZ     SREAD
0470        CALL   ARG2   A SEC.  ARG?
0480        JZ     HEADER
0490        LHLD   BBUF+2
0500        XCHG
0510        LHLD   BBUF
0520        JMP    IN+1
0530 HEADER CALL   LENGT
0540        LHLD   BBUF      START ADDR OF BINARY BUFFER
0550        JMP    IN
0560 * GET HERE IF NO ARGS PRESENT & READ 4 BYTES OF MAX LN NO
```

```
0570 SREAD  LXI     H,MAXL
0580        MVI     B,4
0590 SRIN   CALL    LOOP
0600        MOV     M,A
0610        INX     H
0620        DCR     B
0630        JNZ     SRIN
0640 * READ 2 BYTE LENGTH TO FIX END FILE POINTER
0650        CALL    LENGT
0660        LHLD    BOFP    LOAD BEGIN OF FILE POINTER
0670        DAD     D       ADD LENGTH-1 TO GET
0680        SHLD    EOFP    END OF FILE POINTER
0690        LHLD    BOFP
0700 IN     INX     D
0710        MVI     B,0
0720 * READ IN DATA
0730 LOP    CALL    LOOP    READ DATA FROM CASSETTE
0740        MOV     M,A
0750        ADD     B       ADD CHECKSUM TO A
0760        MOV     B,A
0770        INX     H
0780        DCX     D
0790        XRA     A       FINISHED?
0800        CMP     D
0810        JNZ     LOP
0820        CMP     E
0830        JNZ     LOP
0840 * READ IN CHECKSUM BYTE
0850        CALL    LOOP
0860        CMP     B
0870        LXI     H,MES2
0880        JNZ     ERR
0890        LXI     H,MES1
0900 ERR    CALL    SCRN
0910        JMP     EORNS
0920 * INPUT A BYTE FROM CASSETTE
0930 LOOP   IN      CASC
0940        ANI     10H
0950        JNZ     LOOP
0960        IN      CASD
0970        RET
0980 * GET LENGTH-1 OF FILE
0990 LENGT  CALL    LOOP
1000        MOV     E,A
1010        CALL    LOOP
1020        MOV     D,A
1030        RET
1040 * CHECK FOR PRESENCE OF FIRST ARG IN ASCII BUFFER
1050 ARG1   LXI     H,ABUF
1060        MOV     A,M
1070        ORA     A
1080        RET
1090 * CHECK FOR PRESENCE OF SECOND ARG
1100 ARG2   LXI     H,ABUF+7
1110        MOV     A,M
1120        ORA     A
```



```
1130        RET
1140 *
1150 * CASSETTE WRITE ROUTINE FOR SOURCE & BINARY
1160 *
1170 CASW   CALL    CRLF
1180        MVI     A,3CH   GET START BYTE
1190        CALL    COUT
1200        MVI     A,0E6H  GET SYNC BYTE
1210        CALL    COUT
1220 * IF THERE IS NOTHING IN ABUF THEN COPY A SOURCE FILE
1230        CALL    ARG1
1240        JZ      SRCPY
1250 * NOT SOURCE, BETTER HAVE A SECOND ARG
1260        CALL    ARG2
1270        JZ      WHA1
1280        LHLD    BBUF        GET BEGIN OF LOAD POINTER
1290        LXI     D,BBUF+2    POINT TO END OF LOAD
1300        JMP     HWLNG
1310 SRCPY  LXI     H,MAXL  WRITE 4 BYTES OF MAX LINE NO.
1320        MVI     B,4
1330 SROUT  MOV     A,M
1340        CALL    COUT
1350        INX     H
1360        DCR     B
1370        JNZ     SROUT
1380        LHLD    BOFP
1390        LXI     D,EOFP
1400 * FIND LENGTH OF LOAD
1410 HWLNG  LDAX    D
1420        SUB     L
1430        MOV     C,A
1440        CALL    COUT
1450        INX     D
1460        LDAX    D
1470        SBB     H
1480        MOV     B,A
1490        CALL    COUT
1500        INX     B
1510        MVI     E,0
1520 * WRITE DATA ONTO TAPE
1530 LOPO   MOV     A,M
1540        CALL    COUT
1550        ADD     E       ADD CHECKSUM TO A
1560        MOV     E,A
1570        INX     H
1580        DCX     B
1590        XRA     A       FINISHED?
1600        CMP     B
1610        JNZ     LOPO
1620        CMP     C
1630        JNZ     LOPO
1640        MOV     A,E
1650        CALL    COUT    WRITE CHECKSUM BYTE
1660        LXI     H,MES3
1670        CALL    SCRN
1680        JMP     EORNS
```

```
1690 * WRITE OUT A BYTE TO CASSETTE
1700 COUT   PUSH   PSW
1710 CLOP   IN     CASC   READ CASSETTE STATUS
1720        ANI    20H
1730        JNZ    CLOP
1740        POP    PSW
1750        OUT    CASD   OUTPUT DATA TO CASSETTE
1760        RET
1770 * MESSAGE BUFFER
1780 MES1   ASC    'LOAD COMPLETE'
1790        DB     13
1800 MES2   ASC    'READ ERROR'
1810        DB     13
1820 MES3   ASC    'WRITE COMPLETE'
1830        DB     13
1840 *
1850 * RENUMBER A SOURCE FILE
1860 * ALWAYS PUT INCREMENTING VALUE AS FIRST ARG
1870 * OPTIONALLY PUT LINE NO. TO START RENUM AT AS SEC. ARG
1880 * DEFAULTS TO FIRST LINE NO. IN FILE
1890 *
1900 RENUM  CALL   BIAS   SUBTRACT ASCII BIAS FROM FIRST ARG
1910 * CHECK IF THERE IS A SECOND ARG
1920        CALL   ARG2
1930        JZ     INTLN
1940        LXI    H,ABUF+10
1950        CALL   FIND+3
1960        JMP    GO
1970 * BEGIN LN AT 0000
1980 INTLN  LXI    D,ABUF+7
1990        MVI    A,'0'
2000        MVI    B,4
2010 ZERO   STAX   D
2020        INX    D
2030        DCR    B
2040        JNZ    ZERO
2050 * SO START AT THE BEGINNING & RENUMBER IT ALREADY
2060        LHLD   BOFP
2070 GO     CALL   ENDCK
2080        CALL   ADD
2090        CALL   LOAD
2100        CALL   ADR
2110        JMP    GO
2120 * ROUTINE TO SUBTRACT ASCII BIAS FROM NUMBERS
2130 BIAS   LHLD   INCPT
2140        XCHG
2150        MVI    B,4
2160 ABIN   LDAX   D
2170        SUI    30H
2180        STAX   D
2190        INX    D
2200        DCR    B
2210        JNZ    ABIN
2220        RET
2230 * ROUTINE TO INCREMENT LINE NO. DUE TO A CARRY
2240 INC    SUI    10
```



```
2250        MOV    M,A
2260        DCX    H
2270        MOV    A,M
2280        INR    A
2290        CPI    '9'+1
2300        RC
2310        CALL   INC
2320        RET
2330 * IS THIS THE END ?
2340 ENDCK  MOV    A,M
2350        CPI    1
2360        RNZ
2370 * YES THIS IS THE END - UPDATE MAX LINE NO.
2380        LXI    D,ABUF+10
2390        LXI    H,MAXL-1
2400        CALL   LOAD
2410        LXI    H,ABUF  SET UP TO LIST
2420        MVI    B,8
2430        CALL   CLEAR
2440        CALL   LIST
2450        JMP    EORNS
2460 * ROUTINE TO MOVE CONTENTS OF NO. BUFFER TO FILE
2470 LOAD   MVI    B,4
2480        MOV    A,B
2490        CALL   ADR    ADD 4 TO ADRESS H&L
2500 LOAD1  LDAX   D
2510        MOV    M,A
2520        DCX    D
2530        DCX    H
2540        DCR    B
2550        JNZ    LOAD1
2560        MOV    A,M
2570        RET
2580 * ADD INCREMENT TO WORK BUFFER
2590 ADD    PUSH   H
2600        LHLD   INCPT  POINT TO INC VALUE BUFFER
2610        XCHG
2620        LHLD   LINPT  POINT TO LINE NO. BUFFER
2630        MVI    B,4
2640        PUSH   H
2650 ADINC  LDAX   D
2660        ADD    M
2670        CPI    '9'+1
2680        CNC    INC
2690        MOV    M,A
2700        INX    D
2710        POP    H
2720        INX    H
2730        PUSH   H
2740        DCR    B
2750        JNZ    ADINC
2760        LXI    D,ABUF+10
2770        POP    H
2780        POP    H
2790        RET
2800 * ZERO OUT A BUFFER
```

```
2810 CLEAR  XRA    A
2820 CLER   MOV    M,A
2830        DCR    B
2840        JNZ    CLER
2850        RET
2860 *
2870 * ROUTINE TO CONCATENATE OR INSERT FILES INTO ONE ANOTHER
2880 * FIRST ARG CONTAINS FILE TO BE INSERTED
2890 * SECOND ARG CONTAINS LINE NO.  INSERT WILL FOLLOW
2900 *
2910 INSRT  LXI    H,ABUF
2920        CALL   FIND
2930        MOV    A,M
2940        CALL   ADR
2950        DCX    H
2960        MVI    M,2
2970        SHLD   INSP
2980 * POINT TO FILE TO BE INSERTED AND GET BOFF & EOFF
2990        CALL   FSEA
3000        LXI    D,NMLEN
3010        DAD    D
3020        LDAX   D
3030        ORA    A
3040        JZ     WHA1
3050        CALL   LODM
3060        PUSH   B
3070        MOV    A,D
3080        SUB    B
3090        MOV    L,A
3100        MOV    A,E
3110        SBB    C
3120        MOV    H,A
3130        XCHG
3140        LHLD   EOFP
3150        PUSH   H
3160        DAD    D
3170        SHLD   EOFP
3180        POP    D
3190 * CREATE SPACE FOR INSERTION
3200        MVI    C,2
3210        CALL   RMOV
3220 * MOVE FILE TO BE INSERTED INTO SPACE
3230        POP    B
3240        MOV    E,B
3250        MOV    D,C
3260        LHLD   INSP
3270        MVI    M,ASCR
3280        INX    H
3290        MVI    C,1
3300        CALL   LMOV
3310 *SET UP TO RENUMBER FILES BY INCR OF 1
3320        XRA    A
3330        LXI    D,ABUF   INC VALUE BUFFER POINTER
3340        MVI    B,3
3350 PLACE  STAX   D
3360        INX    D
```



```
3370        DCR    B
3380        JNZ    PLACE
3390        INR    A
3400        STAX   D
3410        INX    D
3420        JMP    INTLN  RENUM COMBINED FILES
3430 *
3440 * ROUTINE TO PROVIDE AUTOMATIC LINE NUMBERING
3450 * FIRST ARG DEFINES STARTING LINE NO.
3460 * OPTIONAL SEC ARG DEFINES INC VALUE - DEFAULTS TO 5
3470 * CTRL/X RETURNS YOU TO NORMAL OPERATION
3480 *
3490 NUMB   CALL   CRLF
3500        CALL   ARG1   CHECK FOR FIRST ARG
3510        JZ     WHA1
3520        XRA    A      SET UP BEGINNING OF LINE FLAG
3530        STA    NFLAG
3540        LXI    H,SAVE  STORE NEW LINE NO.  & INC VALUE POINTERS
3550        SHLD   INCPT
3560        LXI    H,SAVE+4
3570        SHLD   LINPT
3580 * ROUTINE TO SAVE ASCII ARG ABUF & ABUF+7
3590 * ABUF TO ABUF+10 IS USED BY ALS-8 DELETE ROUTINE
3600        LXI    H,ABUF  POINT TO INC VALUE BUFFER
3610        CALL   LODM   LOAD IN BCDE
3620        LXI    H,SAVE+7
3630        CALL   STOM
3640        LXI    H,ABUF+7 POINT TO INC VALUE
3650        CALL   LODM
3660        LXI    H,SAVE+3  POINT TO SAVE BUFFER
3670        CALL   STOM   COPY INC VALUE
3680        CMP    M      CHECK FOR SECOND ARG
3690        JZ     INC5
3700        CALL   BIAS   SUB ASCII BIAS
3710        JMP    NEWIN  ONWARD WITH PRESCRIBED INC
3720 * INITIALIZE INCREMENT TO 5
3730 INC5   MVI    B,3
3740 LINC   MVI    M,0
3750        INX    H
3760        DCR    B
3770        JNZ    LINC
3780        MVI    M,5
3790 * CHANGE INPUT DRIVER
3800 NEWIN  LHLD   INP8+1
3810        SHLD   JIVE+1
3820        LXI    H,BOOGEE
3830        SHLD   INP8+1
3840        JMP    SYS8
3850 * START OF NEW INPUT DRIVER
3860 BOOGEE LDA    NFLAG
3870        JNZ    JIVE   NOT A NEW LINE THEN JIVE TO SYSTEM INPUT
3880        MVI    E,2+5  ADVANCE CHAR COUNT
3890        PUSH   D      & SAVE
3900        MVI    A,1    SET FLAG TO BE CLEARED BY A CR
3910        STA    NFLAG
3920 * PRINT LINE NO.
```

```
3930         LXI    D,SAVE+4  POINT TO LINE NO. BUFFER
3940         LXI    H,IBUF    POINT TO INPUT STREAM
3950         MVI    C,4
3960 VIEW    LDAX   D
3970         MOV    M,A       TO INPUT STREAM
3980         MOV    B,A       SAVE ACC
3990         CALL   OUTP8
4000         INX    H         NEXT PLEASE
4010         INX    D
4020         DCR    C
4030         JNZ    VIEW
4040 * PRINT A SPACE
4050         MVI    A,' '
4060         MOV    M,A
4070         MOV    B,A
4080         CALL   OUTP8
4090         INX    H
4100         CALL   ADD       ADD INC TO LINE NO.
4110         POP    D         CHAR COUNT IN E
4120 * INPUT ROUTINE
4130 JIVE    CALL   TEMP      SYSTEM INPUT ADDRESS TO APPEAR
4140         CPI    24        CHECK FOR CONTROL X
4150         JZ     EXIT
4160         CPI    ASCR      CHECK FOR CR
4170         RNZ
4180         XRA    A         INITIALIZE START OF LINE FLAG
4190         STA    NFLAG
4200         MOV    A,B
4210         RET
4220 * RESTORE POINTERS
4230 EXIT    LXI    H,ABUF
4240         SHLD   INCPT
4250         LXI    H,ABUF+7
4260         SHLD   LINPT
4270         JMP    EORMS     BACK TO MONITOR & PRINT READY
4280 *
4290 * DEFINE CONSTANTS & STORAGE
4300 ASCR    EQU    0DH
4310 NMLEN   EQU    0005H
4320 TEMP    EQU    0000H
4330 CASC    EQU    6EH
4340 CASD    EQU    6FH
4350 PSW     EQU    6
4360 NFLAG   DS     1
4370 SAVE    DS     8
4380 INCPT   DW     ABUF
4390 LINPT   DW     ABUF+7
4400 *
4410 ** ALS-8 ROUTINE ADDRESSES
4420 *
4430 BOFP    EQU    0D005H
4440 EOFP    EQU    0D007H
4450 MAXL    EQU    0D009H
4460 INP8    EQU    0D0CCH
4470 OUTP8   EQU    0D0CFH
4480 ABUF    EQU    0D0DFH
```



```
4490 BBUF    EQU    0D0EFH
4500 INSP    EQU    0D1C9H
4510 IBUF    EQU    0D1E4H
4520 EORMS   EQU    0E060H
4530 SYS8    EQU    0E078H
4540 EORNS   EQU    0E0D1H
4550 CRLF    EQU    0E216H
4560 SCRN    EQU    0E380H
4570 FSEA    EQU    0E68AH
4580 WHA1    EQU    0E7DDH
4590 FIND    EQU    0E941H
4600 ADR     EQU    0E96BH
4610 LMOV    EQU    0E970H
4620 RMOV    EQU    0E979H
4630 LODM    EQU    0E982H
4640 STOM    EQU    0E98AH
4650 LIST    EQU    0E9DBH
4660         END
```

The following Page Printer Program for the ALS-8
was submitted by:

Bill Gunn
875 West Broadway
Vancouver, B.C. V5Z1J9

```
0010 *            ************************
0020 *            *     PAGE PRINTER     *
0030 *            *      FOR ALS-8       *
0040 *            ************************
0050 * THIS PROGRAM IS TO BE USED FOR HARD COPY OUTPUT
0060 * TO SET LEFT HAND MARGINS ONCE FOR ENTIRE PRINT OUT
0070 * AND TO ENCODE SPACES IN AN EFFICIENT MANNER. PAGING
0080 * IS ALSO ACCOMPLISHED WITH 50 LINES PRINTED PER
0090 * PAGE. TOP LINE CONTAINS PAGE NUMBER. ROUTINE CAN BE USED
0100 * AS CUSTOMER COMMAND AS:         PRINT 12
0110 * 12 DEFINING THE NUMBER OF SPACES TO BE LEFT IN THE LEFT
0120 * MARGIN.
0130 *
0140 * SET LEFT HAND MARGIN IN COUNTER
0150 PRINT  LXI    H,ABUF+2  POINT TO SECOND LAST CHAR IN PARAM BUFF
0160        MOV    A,M
0170        ORA    A         ANYTHING THERE?
0180        JZ     WHA1      NO - PRINT WHAT? & RETURN TO ALS-8
0190        CALL   ADECB     YES - CONVERT IT INTO BINARY NO.
0200        STA    MARG      STORE AS LEFT HAND MARGIN VALUE
0210        XRA    A         INITIALIZE THE FOLLOWING
0220 * DFLAG CONTAINS SWITCHES FOR TOP OF PAGE EJECT, & CONTROL
0230 * CHAR. FOR SPACING OR NULLS FOR CR
0240        STA    DFLAG
0250        STA    LNCNT     LINE COUNT
0260        STA    SAVE      LAST CHAR. OUTPUTTED
0270        MVI    B,3       ZERO OUT FIRST 3 BYTES OF PAGE NO. BUFFER
0280        LXI    H,BUFF    POINT TO PAGE NO. BUFFER
0290 ZERO   MOV    M,A       ZERO IT
0300        INX    H         NEXT BYTE
0310        DCR    B
0320        JNZ    ZERO
0330        MVI    M,'0'     INSTALL AN ASCII ZERO IN THE LAST BYTE
0340        CALL   CRLF      RESET PRINTER HEAD
0350        CALL   PGNUM     PRINT FIRST PAGE NO.
0360        CALL   PAGE      FOLLOWED BY 7 LINE FEEDS
0370 * CHANGE OUTPUT DRIVER
0380        LXI    H,FLY     NEW OUTPUT DRIVER ADDRESS
0390        SHLD   OUTP8+1
0400 * SET UP FOR TEXT COMMAND
0410        LXI    H,ATEXT   POINT TO ASCII 'TEXT' IN ALS-8
0420        CALL   LODM      LOAD ASCII CHARS. INTO REG B C D E
0430        LXI    H,IBUF+3  POINT TO INPUT COMMAND BUFFER
0440        CALL   STOM      STUFF ASCII 'TEXT' HERE
0450        CALL   ZBUF      ZERO PARAMETER BUFFER, ABUF TO ABUF+20
0460        LXI    H,SYS8+6  ADDRESS TO BE USED AFTER THE NEXT RET
```



```
0470        PUSH   H         PUT IT ON THE STACK
0480        LXI    H,IBUF+4  POINT TO JUST AFTER 'TEXT'
0490        JMP    CENTRY    ADD A CR & EOF INDICATOR BEFORE COMP STRINGS
0500 *       -CENTRY IS A POINT IN THE ALS-8 WHERE I MAY
0510 *        CONTINUE AFTER ENTERING A COMMAND VIA THIS PROGRAM
0520 *
0530 * GENERAL DRIVER FOR VDM-1 AND LA30
0540 * SENSE SWITCH 8 DOWN VDM DRIVER IS USED
0550 * SENSE SWITCH 8 UP LA30 DRIVER IS USED
0560 FLY    IN     SENSE
0570        RAL
0580        JC     NOCPY
0590        RAR
0600        RAR
0610        JC     HARDC
0620 * STANDARD VIDEO DRIVER (VDM-1)
0630        CALL   VIDEO
0640        RET
0650 HARDC  MOV    A,B       PUT CHAR IN ACC. SO WE CAN LOOK AT IT
0660        CPI    ASCR      IS IT A CR?
0670        JNZ    DEL       NO - SEE IF IT'S A DELETE
0680        LDA    LNCNT     YES - INCREMENT LINE COUNT
0690        INR    A
0700        CPI    51D       PRINTED 51 LINES?
0710        JNZ    STCNT     NO - STORE LINE COUNT
0720        MVI    A,82H     YES - SET FLAGS, IGNORE DELETES & SET TOP OF F
0730        STA    DFLAG     SAVE FLAGS
0740        JMP    OUT8      OUTPUT CARRIAGE RETURN ,LINE FEED & TWO DELETES
0750 STCNT  STA    LNCNT
0760        LDA    SAVE      GET LAST CHAR PRINTED
0770        CPI    ' '       IS IT A SPACE?
0780        JZ     KEEPM      YES - HOLD PRINT HEAD WHERE IT IS
0790        MVI    A,80H     NO - SET FLAG TO IGNORE NEXT TWO DELETE CHAR
0800        STA    DFLAG     SAVE FLAGS
0810        JMP    OUT8
0820 KEEPM  POP    B         YES - GET OUT OF CRLF ROUTINE BY PULLING RET
0830 *                       ADDRESS FROM STACK
0840        MVI    B,ASLF    SKIP THE CR & DO A LINE FEED ONLY
0850        CALL   OUT8      BECAUSE YOU'VE ALREADY SET LEFT MARGIN
0860        MVI    A,' '      SET UP LAST CHAR PRINTED AS LEFT MARG READY
0870        STA    SAVE      STUFF IT IN SAVE
0880        RET              GET ANOTHER CHARACTER TO BE PRINTED
0890 DEL    CPI    DELETE    CHECK FOR DELETE CHAR TO SET SPACES
0900        JNZ    OUT8      NO - THEN OUTPUT IT
0910 * TEST FOR TWO DELTETE CHAR OUTPUT AFTER A CR
0920        LDA    DFLAG     GET FLAGS
0930        RAL              IF MOST SIG BIT SET, OUTPUT DELETE AS A NULL
0940        JC     OUT8
0950 * BIT NOT SET, USE NEXT TWO CHAR. AS NO. OF SPACES
0960        INX    H         POINT TO CHAR  FOLLOWING DELETE CHAR
0970        CALL   ADECB     CHAR. WILL BE DECIMAL SO CONVERT TO BINARY
0980        JMP    BUTTER    & OUTPUT THAT MANY SPACES
0990 * LA30 DRIVER
1000 OUT8   IN     USTA      CHECK STATUS
1010        ANI    80H       READY TO OUTPUT?
1020        JZ     OUT8      NO - CHECK AGAIN
```

```
1030         PUSH   B       YES - SAVE IT & WAIT TILL LA30 CATCHES UP WITH
1040 *                       3P+S
1050         LXI    B,HUM   LOAD WAIT FACTOR
1060 TIMIT   DCR    B
1070         JNZ    TIMIT
1080         DCR    C
1090         JNZ    TIMIT
1100         POP    B       FINISHED WAITING, LETS HAVE THE CHARACTER
1110         MOV    A,B     MOVE IT TO ACC SO WE CAN OUTPUT IT
1120         OUT    CHAN3
1130         STA    SAVE    SAVE IT SO WE CAN CHECK IT LATER
1140         CPI    DELETE   WAS IT A DELETE?
1150         RNZ            NO - BEAT IT
1160         LDA    DFLAG    YES - CHECK FOR FLAGS
1170         RAR            AT THE LEAST SIG BIT - LAST CHAR A DELETE?
1180         JC     LMARG   YES - SET MARG, MAYBE TOP OF PAGE
1190         CMC            NO - SET FLAG FOR NEXT TIME - 2ND DELETE CHAR
1200         RAL            PUT IT IN PLACE
1210         STA    DFLAG   & STORE IT
1220         RET
1230 * WAIT TILL SENSE SW 15 HAS BEEN TOGGLED BACK
1240 NOCPY   IN     SENSE
1250         RAL
1260         JC     NOCPY
1270         JMP    EORMS   BACK TO ALS-8, REPLACING I/O DRIVER & PRINT REA
1280 * SET LEFT MARGIN
1290 LMARG   RAR            CHECK FOR NEW PAGE FLAG
1300         JNC    LEFT    NOT SET THEN SET MARGIN
1310         CALL   PAGE    GET TO TOP OF NEXT PAGE
1320         STA    LNCNT   STORE ZEROED LINE COUNT SO YOU CAN DO A CR
1330         STA    DFLAG   ZERO FLAG
1340         CALL   PGNUM   PRINT PAGE NUMBER
1350         CALL   PAGE    LEAVE SOME SPACE &
1360         STA    LNCNT   INITIALIZE LINE COUNT
1370         RET
1380 * THE SPACE GENERATOR
1390 LEFT    LDA    MARG    GET PREVIOUSLY DEFINED LEFT MARG POSITION
1400 BUTTER  MOV    C,A     STUFF NO. OF SPACES IN REG C
1410         CALL   OVER+2   SPACE IT OUT
1420         STA    DFLAG   RESET FLAG
1430         RET
1440 * CONVERT TWO ASCII DEC DIGITS TO BINARY
1450 * LEAVING THE RESULT IN REG D
1460 ADECB   MVI    D,0     INITIALIZE BINARY COUNT
1470         MVI    C,10D   HOW MANY 10'S
1480         CALL   CONV
1490         INX    H       NEXT ASCII DIGIT
1500         MVI    C,1     HOW MANY 1'S
1510         CALL   CONV
1520         RET
1530 * CONVERT ASCII DECIMAL TO BINARY
1540 CONV    MOV    A,M     PUT ASCII DIGIT IN ACC
1550         SUI    '0'-1   SUBTRACT ASCII BIAS
1560         MOV    B,A     & STORE IN B
1570         MOV    A,D .   D CONTAINS RESULT OF PREVIOUS DIGIT CONVERSION
1580         DCR    B
```



```
1590         RZ
1600         ADD    C       ADD IN APPROPRIATE BINARY VALUE OF DECIMAL DIGIT
1610         MOV    D,A     & STORE IN D
1620         JMP    CONV+4
1630 * ROUTINE FOR MULTIPLE CHAR. OUTPUT
1640 OVER    MVI    C,63D   MOVE PAGE NO. OVER TO THE RIGHT
1650         MVI    D,' '   BY INSERTING MULTIPLE SPACES
1660         JMP    SHOOT
1670 PAGE    MVI    C,7     SKIP OVER PAPER PERFORATIONS
1680         MVI    D,ASLF    BY DOING MULTIPLE LINE FEEDS
1690 SHOOT   PUSH   B       SAVE PREVIOUS CHAR. TO BE OUTPUT
1700         MOV    B,D     PUT  CHAR. TO BE OUTPUT IN REG B
1710 SPACE   DCR    C       NO. OF CHAR. TO OUTPUT IN REG C
1720         JM     SPCED   TEST FOR SPECIAL CASE OF NO CHAR TO BE OUTPUT
1730         CALL   OUT8
1740         JMP    SPACE   AGAIN
1750 SPCED   POP    B       FINISHED - SO RESTORE ORIG CHAR TO BE OUTPUT
1760         XRA    A       CLEAR ACC
1770         RET
1780 * INCREMENT & PRINT CONTENTS OF PAGE NO. BUFFER
1790 PGNUM   LXI    H,BUFF+3  POINT TO END OF PAGE NO. BUFFER
1800 DIGIT   INR    M       INCREMENT DIGIT
1810         MOV    A,M     STUFF IN ACC. TO LOOK AT IT
1820         CPI    3AH     INCREMENTED DIGIT PAST 9?
1830         JNZ    GETOUT   NO - THEN PRINT WHOLE BUFFER
1840         MVI    M,'0'   YES - RESET THIS DIGIT POSITION TO 0
1850         DCX    H       LOOK AT NEXT DIGIT
1860         MOV    A,M     STUFF IN ACC. TO LOOK AT IT
1870         ORA    A       ANYTHING HERE?
1880         JNZ    DIGIT   YES - INCREMENT IT
1890         MVI    M,'0'   NO - PUT IN ASCII BIAS
1900         JMP    DIGIT
1910 GETOUT  CALL   OVER    POSITION PRINTER HEAD FOR PAGE NO.
1920         LXI    H,MSG   & PRINT PAGE NO.
1930         CALL   SCRN    OUTPUT A STRING OF ASCII CHAR TILL CR
1940         CALL   CRLF    OUTPUT CR, LF & 2 NULLS
1950         RET
1960 MSG     ASC    "PAGE "
1970 BUFF    DS     4
1980         DW     0D2EH   A PERIOD FOLLOWED BY A CR
1990 *
2000 * DEFINE CONSTANTS
2010 VIDEO   EQU    0FE77H
2020 ASLF    EQU    0AH
2030 ASCR    EQU    0DH
2040 DELETE  EQU    7FH
2050 CHAN3   EQU    3
2060 USTA    EQU    0
2070 DAV     EQU    40H
2080 TBE     EQU    80H
2090 OUTP8   EQU    0D0CFH
2100 ABUF    EQU    0D0DFH
2110 IBUF    EQU    0D1E4H
2120 EORMS   EQU    0E060H
2130 SYS8    EQU    0E078H
2140 CENTRY  EQU    0E193H
```

```
2150 CRLF   EQU    0E216H
2160 ZBUF   EQU    0E26AH
2170 SCRN   EQU    0E380H
2180 ATEXT  EQU    0E3ECH
2190 HUM    EQU    0015H
2200 SENSE  EQU    0FFH
2210 MARG   DS     1
2220 DFLAG  DS     1
2230 LNCNT  DS     1
2240 SAVE   DS     1
2250 WHA1   EQU    0E7DDH
2260 LODM   EQU    0E982H
2270 STOM   EQU    0E98AH
2280 ADEC   EQU    0E2FAH
2290        END
```


The following three programs (Output Driver for TTY, Load Saved Programs and Save Current File on Tape) were submitted by:

Dennis H. Rosenthal

PROGRAM DESCRIPTION:

THE TTY PRE-DRIVER IS USED TO CORRECT A DEFICIENCY IN THE ALS-8 ASSEMBLY OUTPUT. WHEN LINE OUTPUT EXCEEDS A PREDEFINED LIMIT (72 FOR A TTY) THIS PROGRAM OUTPUTS A CARRAGE RETURN AND LINE-FEED, AND SPACES OVER TO THE COMMENT AREA OF THE ASSEMBLER LISTING. ALSO INCLUDED IS A ROUTINE TO KEEP TRACK OF THE NUMBER OF LINES PRINTED. WHEN A PRESET LIMIT IS REACHED A PAGE BREAK OCCURS, OUTPUTING A PAGE NUMBER, LINE SKIPS, AND A LINE OF HYPHENS. A SEPERATE CUSTOM COMMAND IS INCLUDED (CLRPC) TO MANUALLY RESET PAGE AND LINE COUNT. THE STANDARD OUTPUT DRIVER IS BRANCHED TO FROM THIS PROGRAM.

THE OTHER TWO PROGRAMS ARE USED TO SAVE AND RETREIVE PROGRAMS THAT HAVE BEEN STORED ON CASSETTE TAPE. THEY ALLOW MULTIPLE PROGRAMS ON ONE TAPE AND ALWAYS WORK WITH THE ALS-8 SYSTEMS CURRENT FILE. THE PROGRAM NAME IS ENCLOSED BY '<' AND '>', ALTHOUGH ANY CHARACTERS COULD HAVE BEEN SELECTED. EXAMPLE: SAVE 0 0 <PROG A,8/1/76>. THE TWO ZEROS ARE USED TO SATISFY THE PARAMETER CHECKING THAT IS DONE BY THE ALS-8 SYSTEM. THE PROGRAMS SEARCH THE INPUT BUFFER AREA FOR THE CONTROL CHARACTERS AND IF NOT FOUND THEY BRANCH TO 'WHAT'. THE FILE LOCATION IS TAKEN FROM THE ALS-8 CURRENT FILE POINTER LOCATED AT D005H. IF THE ESCAPE KEY IS PRESSED ON THE ADDRESSED KEYBOARD, THE PROGRAMS RETURN CONTROL BACK TO THE SYSTEM IMMEDIATELY. AT THE END OF A RESTORE, THE FCHK COMMAND MAY BE USED TO COMPLETE THE FILE IDENTIFICATION TO THE ALS-8 SYSTEM.

IN ALL THESE PROGRAMS, I CONSIDER THE VDM AND ASSOCIATED KEYBOARD AS THE MASTER OR 'SYSIO', WITH THE TTY FOR LISTINGS. THE PROGRAMS ARE NOT MINIMIZED, AND PERHAPS BETTER USE COULD HAVE BEEN MADE OF THE SWITCHABLE I/O DRIVER OPTIONS.

```
3C00                    0010 ****************************************
3C00                    0020 * THIS PROGRAM WRITTEN WITH THE ALS-8   *
3C00                    0030 *  SYSTEM BY DENNIS H. ROSENTHAL        *
3C00                    0040 * OUTPUT DRIVER FOR TTY                 *
3C00                    0050 *  PRODUCES FORMATED OUTPUT LISTING     *
3C00                    0060 * FOR USE WITH THE ASSEMBLER.           *
3C00                    0070 *  LINE CARRY-OVERS ARE SPACED TO MATCH *
3C00                    0080 * COMMENT FORMAT. AFTER EVERY "LLIM"    *
3C00                    0090 * LINES A PAGE BREAK OCCURS WITH PAGE   *
3C00                    0100 * NUMBER PRINTED AT END OF PAGE AND A   *
3C00                    0110 * STRING OF HYPHENS AS SEPERATOR.       *
3C00                    0120 * THE MAXIMUM PAGE NUMBER IS 99 DECIMAL *
3C00                    0130 *  A SEPERATE CLEAR COMMAND FOR LINE    *
3C00                    0140 * AND PAGE COUNTS IS PROVIDED.          *
3C00                    0150 * THE STANDARD I/O DRIVER IS USED FOR   *
3C00                    0160 *  ACTUAL CHARACTER OUTPUT              *
3C00                    0170 ****************************************
3C00                    0180 START  EQU   $
3C00 78                 0190        MOV   A,B     TEST CHARACTER
3C01 FE 0D              0200        CPI   0DH     CARRAGE RETURN ?
3C03 CA 15 3C           0210        JZ    CCHRC   CLEAR CHARACTER COUNT
3C06 3A BA 3C           0220        LDA   CCTR    INCREMENT CHAR COUNT
3C09 3C                 0230        INR   A
3C0A FE 4A              0240        CPI   CLIM    CHARACTER LIMIT ?
3C0C CA 28 3C           0250        JZ    SPLN    SPACE LINE
3C0F 32 BA 3C           0260        STA   CCTR
3C12 C3 A9 D0           0270        JMP   OUT     OUTPUT CHARACTER
3C15                    0280 ****************************************
3C15                    0290 CCHRC  EQU   $       CLEAR CHAR COUNT
3C15 AF                 0300        XRA   A
3C16 32 BA 3C           0310        STA   CCTR    RESET CHAR COUNT
3C19                    0320 LNCHK  EQU   $       CHECK NUMBER OF LINES
3C19 3A BC 3C           0330        LDA   LCTR    INCREMENT LINE COUNT
3C1C 3C                 0340        INR   A
3C1D FE 32              0350        CPI   LLIM    PAGE LIMIT REACHED ?
3C1F CA 39 3C           0360        JZ    PGBRK   PAGE BREAK
3C22 32 BC 3C           0370        STA   LCTR
3C25 C3 A9 D0           0380        JMP   OUT     OUTPUT CHARACTER
3C28                    0390 ****************************************
3C28                    0400 SPLN   EQU   $       ADD SPACES
3C28 C5                 0410        PUSH  B       SAVE B
3C29 0E 32              0420        MVI   C,SPLEN  NUMBER OF SPACES
3C2B 3E 31              0430        MVI   A,SPLEN-1
3C2D 32 BA 3C           0440        STA   CCTR    SET NEW CHAR COUNT
3C30 06 20              0450        MVI   B,20H   SPACES TO BE PRINTED
3C32 CD A1 3C           0460        CALL  PUT     OUTPUT SPACES
3C35 C1                 0470        POP   B
3C36 C3 19 3C           0480        JMP   LNCHK   INCREMENT LINE COUNT
```



```
3C39                    0490 ****************************************
3C39                    0500 PGBRK  EQU   $       END OF PAGE
3C39 AF                 0510        XRA   A
3C3A 32 BC 3C           0520        STA   LCTR    CLEAR LINE COUNTER
3C3D C5                 0530        PUSH  B
3C3E 0E 03              0540        MVI   C,03H   SKIP 3 LINES FIRST
3C40 CD 6A 3C           0550        CALL  CRLF
3C43 CD 79 3C           0560        CALL  CONV    CONVERT BINARY TO ASCI
                                                  I
3C46 E5                 0570        PUSH  H
3C47 06 20              0580        MVI   B,20H   SPACES
3C49 0E 1B              0590        MVI   C,27    27 SPACES
3C4B CD A1 3C           0600        CALL  PUT     OUTPUT SPACES BEFORE "
                                                  PAGE"
3C4E 21 B1 3C           0610        LXI   H,MSG1
3C51 CD 96 3C           0620        CALL  MSG     OUTPUT PAGE NUMBER
3C54 0E 04              0630        MVI   C,04H   SKIP 4 LINES AFTER
3C56 CD 6A 3C           0640        CALL  CRLF
3C59 06 2D              0650        MVI   B,2DH   HYPHEN
3C5B 0E 46              0660        MVI   C,70    70 HYPHENS
3C5D CD A1 3C           0670        CALL  PUT     OUTPUT SEPERATOR
3C60 0E 05              0680        MVI   C,5     SKIP 6 LINES LAST
3C62 CD 6A 3C           0690        CALL  CRLF
3C65 E1                 0700        POP   H
3C66 C1                 0710        POP   B
3C67 C3 A9 D0           0720        JMP   OUT     FINALLY OUTPUT CHARACT
                                                  ER
3C6A                    0730 ****************************************
3C6A                    0740 CRLF   EQU   $       REG C HOLDS LOOP FACTO
                                                  R
3C6A 06 0D              0750        MVI   B,0DH   CARRAGE RETURN
3C6C CD A9 D0           0760        CALL  OUT
3C6F 06 0A              0770        MVI   B,0AH   LINE FEED
3C71 CD A9 D0           0780        CALL  OUT
3C74 0D                 0790        DCR   C
3C75 C8                 0800        RZ
3C76 C3 6A 3C           0810        JMP   CRLF    LOOP BACK
3C79                    0820 ****************************************
3C79                    0830 CONV   EQU   $       CONVERTS PAGE COUNT TO
                                                   ASCII
3C79 3A BB 3C           0840        LDA   PGCT    PAGE COUNT
3C7C 3C                 0850        INR   A
3C7D 32 BB 3C           0860        STA   PGCT    SAVE
3C80 27                 0870        DAA
3C81 47                 0880        MOV   B,A     SAVE IN B
3C82 E6 0F              0890        ANI   0FH     FIRST 4 BITS
3C84 F6 30              0900        ORI   30H     MAKE IT ASCII
3C86 32 B8 3C           0910        STA   ASC0
3C89 78                 0920        MOV   A,B
3C8A 0F                 0930        RRC
```

```
3C8B 0F                0940          RRC
3C8C 0F                0950          RRC
3C8D 0F                0960          RRC
3C8E E6 0F             0970          ANI     0FH     NEXT 4 BITS
3C90 F6 30             0980          ORI     30H     MAKE IT ASCII
3C92 32 B7 3C          0990          STA     ASC1
3C95 C9                1000          RET
3C96                   1010 ****************************************
3C96                   1020 MSG      EQU     $       OUTPUTS MESSAGE POINTE
                                                     D TO BY HL
3C96 7E                1030          MOV     A,M     TAKE FROM MEMORY
3C97 B7                1040          ORA     A       SET COND CODES
3C98 C8                1050          RZ      .       RETURN IF ZERO
3C99 47                1060          MOV     B,A     SAVE IN B
3C9A CD A9 D0          1070          CALL    OUT
3C9D 23                1080          INX     H
3C9E C3 96 3C          1090          JMP     MSG
3CA1                   1100 ****************************************
3CA1                   1110 PUT      EQU     $       OUTPUT REG B  C TIMES
3CA1 CD A9 D0          1120          CALL    OUT     OUTPUT TO PRINTER
3CA4 0D                1130          DCR     C
3CA5 C8                1140          RZ
3CA6 C3 A1 3C          1150          JMP     PUT     LOOP BACK
3CA9                   1160 ****************************************
3CA9                   1170 * THIS SECTION SHOULD BE EXECUTED BY A
3CA9                   1180 *  A CUSTOM COMMAND SUCH AS CLRPC
3CA9                   1190 CLRPC    EQU     $       RESET CHAR AND LINE CO
                                                     UNTERS
3CA9 AF                1200          XRA     A
3CAA 32 BB 3C          1210          STA     PGCT
3CAD 32 BC 3C          1220          STA     LCTR
3CB0 C9                1230          RET
3CB1                   1240 ****************************************
3CB1 50 41 47 45       1250 MSG1     ASC     "PAGE  "
     20 20
3CB7 00                1260 ASC1     DB      0
3CB8 00                1270 ASC0     DB      0
3CB9 00                1280          DB      0       END OF MSG 1
3CBA 00                1290 CCTR     DB      0       CHAR COUNT WITHIN A LI
                                                     NE
3CBB 00                1300 PGCT     DB      0       PAGE COUNT
3CBC 00                1310 LCTR     DB      0       USED TO SAVE NUMBER OF
                                                      LINES
3CBD                   1320 SPLEN    EQU     50      NUMBER OF SPACES FOR
                                                     OVERFLOW
3CBD                   1330 CLIM     EQU     74      CHAR LIMIT PER LINE
3CBD                   1340 OUT      EQU     0D0A9H  STANDARD OUTPUT DRIVE
                                                     R **
3CBD                   1350 LLIM     EQU     50      MAX LINES PER PAGE
3CBD                   1360 ****************************************
```



```
3CBD                   1370          COM     CLRPC  ALSO PUT IN CUST TABLE
3CBD                   1380 ****************************************

ASC0    3CB8    0910
ASC1    3CB7    0990
CCHRC   3C15    0210
CCTR    3CBA    0220 0260 0310 0440
CLIM    004A    0240
CLRPC   3CA9
CONV    3C79    0560
CRLF    3C6A    0550 0640 0690 0810
LCTR    3CBC    0330 0370 0520 1220
LLIM    0032    0350
LNCHK   3C19    0480
MSG     3C96    0620 1090
MSG1    3CB1    0610
OUT     D0A9    0270 0380 0720 0760 0780 1070 1120
PGBRK   3C39    0360
PGCT    3CBB    0840 0860 1210
PUT     3CA1    0460 0600 0670 1150
SPLEN   0032    0420 0430
SPLN    3C28    0250
START   3C00
```

```
3CC0                    0010 ************************************
3CC0                    0020 * THIS PROGRAM LOADS SAVED PROGRAMS  *
3CC0                    0030 * DIRECTLY INTO CORE STARTING AT THE *
3CC0                    0040 * LOCATION POINTED TO BY THE CURRENT *
3CC0                    0050 * FILE POINTER AT "D005"            *
3CC0                    0060 * FILE NAME IS MATCHED AGAINST ENTRY *
3CC0                    0070 * IN INPUT LINE AREA                *
3CC0                    0080 ************************************
3CC0 DB 03              0090        IN      TAPED   CLEAR TAPE REGISTER
3CC2 CD 2C 3D           0100        CALL    CRLF
3CC5 DB 03              0110        IN      TAPED
3CC7                    0120 *
3CC7                    0130 * GET FILE NAME
3CC7                    0140 *
3CC7 21 E4 D1           0150        LXI     H,IBUF  BUFFER POINTER
3CCA 0E 32              0160        MVI     C,50    MAX SEARCH LENGTH
3CCC                    0170 FNAME  EQU     $
3CCC 7E                 0180        MOV     A,M
3CCD 23                 0190        INX     H
3CCE 0D                 0200        DCR     C
3CCF CA DD E7           0210        JZ      WHAT    NOT PROPER INPUT
3CD2 FE 3C              0220        CPI     '<'     TEST FOR BEG OF NAME
3CD4 C2 CC 3C           0230        JNZ     FNAME   LOOP BACK
3CD7                    0240 *
3CD7                    0250 * STORE FILE NAME LOCATION
3CD7                    0260 *
3CD7 2B                 0270        DCX     H       BACKSPACE ONE
3CD8 22 51 3D           0280        SHLD    STARTLOC
3CDB                    0290 *
3CDB                    0300 * TRY TO MATCH WITH TAPE
3CDB                    0310 *
3CDB                    0320 MATCH  EQU     $
3CDB 2A 51 3D           0330        LHLD    STARTLOC
3CDE 4E                 0340        MOV     C,M
3CDF                    0350 MAT1   EQU     $
3CDF CD 11 3D           0360        CALL    TAPE    GET DATA
3CE2 B9                 0370        CMP     C
3CE3 C2 DF 3C           0380        JNZ     MAT1    NO MATCH
3CE6                    0390 MAT2   EQU     $
3CE6 23                 0400        INX     H
3CE7                    0410 *
3CE7                    0420 * TEST MIDDLE CHARACTERS
3CE7                    0430 *
3CE7 4E                 0440        MOV     C,M
3CE8 CD 11 3D           0450        CALL    TAPE
3CEB FE 3E              0460        CPI     '>'     END OF TAPE NAME ?
3CED CA F7 3C           0470        JZ      MAT3
3CF0 B9                 0480        CMP     C       TEST AGAINST MEMORY
```



```
3CF1 CA E6 3C           0490        JZ      MAT2    MATCH SO FAR
3CF4 C3 DB 3C           0500        JMP     MATCH   NO MATCH
3CF7                    0510 MAT3   EQU     $
3CF7 B9                 0520        CMP     C
3CF8 C2 DB 3C           0530        JNZ     MATCH   NO MATCH !
3CFB CD 11 3D           0540        CALL    TAPE    REMOVE EXTRA CR
3CFE                    0550 *
3CFE                    0560 * LOAD FILE INTO MEMORY
3CFE                    0570 *
3CFE 2A 05 D0           0580        LHLD    FPTR    FILE POINTER
3D01                    0590 MAIN   EQU     $
3D01 CD 11 3D           0600        CALL    TAPE
3D04 70                 0610        MOV     M,B
3D05 23                 0620        INX     H
3D06 FE 01              0630        CPI     1       TEST FOR END OF FILE
3D08 C2 01 3D           0640        JNZ     MAIN
3D0B 21 32 3D           0650        LXI     H,MSG
3D0E C3 E0 E7           0660        JMP     MESS
3D11                    0670 *
3D11                    0680 * GET DATA FROM TAPE
3D11                    0690 *
3D11                    0700 TAPE   EQU     $
3D11 DB 0A              0710        IN      KBSTAT
3D13 E6 01              0720        ANI     MASK1
3D15 CA 21 3D           0730        JZ      TAPE2
3D18 DB 08              0740        IN      KBDATA
3D1A E6 7F              0750        ANI     127
3D1C FE 1B              0760        CPI     ESC
3D1E CA 60 E0           0770        JZ      EORMS   RETURN IMMEDIATELY
3D21                    0780 TAPE2  EQU     $
3D21 DB 02              0790        IN      TAPES
3D23 E6 01              0800        ANI     MASK1
3D25 CA 11 3D           0810        JZ      TAPE
3D28 DB 03              0820        IN      TAPED   GET DATA
3D2A 47                 0830        MOV     B,A     SAVE
3D2B C9                 0840        RET
3D2C                    0850 *
3D2C                    0860 * SEND CR TO VDM
3D2C                    0870 *
3D2C                    0880 CRLF   EQU     $
3D2C 06 0D              0890        MVI     B,0DH
3D2E CD 00 DE           0900        CALL    VDM
3D31 C9                 0910        RET
3D32                    0920 *
3D32                    0930 * END OF PROGRAM
3D32                    0940 *
3D32 52 45 53 54        0950 MSG    ASC     "RESTORE COMPLETE, FCHK FILE
     4F 52 45 20                            ! "
     43 4F 4D 50
```

```
     4C 45 54 45
     2C 20 46 43
     48 4B 20 46
     49 4C 45 20
     21 20
3D50 0D                  0960         DB      0DH
3D51                     0970 MASK1   EQU     1
3D51                     0980 FPTR    EQU     0D005H
3D51                     0990 STARTLOC DS     2
3D53                     1000 MESS    EQU     0E7E0H
3D53                     1010 *

CRLF     3D2C      0100
FNAME    3CCC      0230
FPTR     D005      0580
MAIN     3D01      0640
MASK1    0001      0720 0800
MAT1     3CDF      0380
MAT2     3CE6      0490
MAT3     3CF7      0470
MATCH    3CDB      0500 0530
MESS     E7E0      0660
MSG      3D32      0650
START    3D51      0280 0330
TAPE     3D11      0360 0450 0540 0600 0810
TAPE2    3D21      0730
```



```
3D60                     0010 ****************************************
3D60                     0020 * THIS PROGRAM WRITTEN WITH THE ALS-8   *
3D60                     0030 *  SYSTEM BY DENNIS H. ROSENTHAL       *
3D60                     0040 *THIS PROGRAM SAVES CURRENT FILE ON TAPE*
3D60                     0050 *ALONG WITH NAME .  NAME IS ENCLOSED BY *
3D60                     0060 *THE CHARACTERS '< >' . ENTER TWO DUMMY *
3D60                     0070 * NUMBERS TO SATISFY NORMAL PARAMETER   *
3D60                     0080 *   CHECKING .                          *
3D60                     0090 ****************************************
3D60                     0100 **********************************
3D60                     0110 * FIRST OUTPUT FILE NAME TO TAPE *
3D60                     0120 **********************************
3D60 CD A6 3D            0130         CALL    CRLF    RETURN CURSOR
3D63                     0140 * PUT INPUT BUFFER LOC IN H,L
3D63 21 E4 D1            0150         LXI     H,IBUF
3D66 0E 3C               0160         MVI     C,60    MAX LENGTH TO SEARCH
3D68 7E                  0170 F1      MOV     A,M
3D69 23                  0180         INX     H
3D6A 0D                  0190         DCR     C
3D6B CA DD E7            0200         JZ      WHAT    ERROR CONDITION *
3D6E FE 3C               0210         CPI     '<'
3D70 C2 68 3D            0220         JNZ     F1
3D73 47                  0230         MOV     B,A
3D74 CD 9B 3D            0240 F2      CALL    OUTN    PUT NAME ON TAPE
3D77 7E                  0250         MOV     A,M
3D78 23                  0260         INX     H
3D79 FE 3E               0270         CPI     '>'
3D7B 47                  0280         MOV     B,A
3D7C                     0290 * CONTINUE UNTIL END INDICATOR IS FOUND
3D7C C2 74 3D            0300         JNZ     F2
3D7F CD 9B 3D            0310         CALL    OUTN
3D82 06 0D               0320         MVI     B,0DH CR
3D84 CD 9B 3D            0330         CALL    OUTN
3D87                     0340 **************************
3D87                     0350 * NOW PUT FILE TO TAPE ***
3D87                     0360 **************************
3D87 CD A6 3D            0370         CALL    CRLF
3D8A 2A 05 D0            0380         LHLD    FPTR    GET BEG OF FILE
3D8D                     0390 MAIN    EQU     $
3D8D 46                  0400         MOV     B,M     TRANSFER TO REG
3D8E 23                  0410         INX     H
3D8F CD 9B 3D            0420         CALL    OUTN    TRANSFER TO TAPE
3D92 78                  0430         MOV     A,B     CHECK FOR END OF FILE
3D93 FE 01               0440         CPI     1       END OF FILE ?
3D95 C2 8D 3D            0450         JNZ     MAIN
3D98 C3 60 E0            0460         JMP     EORMS   BACK TO SYSTEM
3D9B                     0470 **********************
3D9B                     0480 * SEND REG B TO TAPE *
```

```
3D9B                    0490 **********************
3D9B                    0500 OUTN    EQU     $
3D9B DB 02              0510         IN      TAPES
3D9D E6 02              0520         ANI     MASK1
3D9F CA 9B 3D           0530         JZ      OUTN
3DA2 78                 0540         MOV     A,B
3DA3 D3 03              0550         OUT     TAPED
3DA5 C9                 0560         RET
3DA6                    0570 ******************
3DA6                    0580 * SEND CR TO VDM *
3DA6                    0590 ******************
3DA6                    0600 CRLF    EQU     $
3DA6 06 0D              0610         MVI     B,0DH
3DA8 CD 00 DE           0620         CALL    VDM
3DAB C9                 0630         RET
3DAC                    0640 MASK1   EQU     2
3DAC                    0650 FPTR    EQU     0D005H
3DAC                    0660 ****************************************
3DAC                    0670 *             END OF PGM                *
3DAC                    0680 ****************************************

CRLF    3DA6    0130 0370
F1      3D68    0220
F2      3D7A    0300
FPTR    D005    0380
MAIN    3D8D    0450
MASK1   0002    0520
OUTN    3D9B    0240 0310 0330 0420 0530
```