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 omission in the manual as it stands now. The maximum terminal width is 118 and not 128, as it stated in the manual. Even though the input buffer is 128 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 "S". 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 instruction if the zero flag was reset while:

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.

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.
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 that 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 $BCCH
MVI A,'VDMBASE

will load the accumulator with BCCH. Where as:

POTT5 EQU $MFPH
MVI E,'POTT5

will load register 'E' with MFPH.

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 (OIAH thru 1ICB). 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 as '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 to 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
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 $01E8H

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 preceded by a valid "END" pseudo in the source file so the assembler can change its internal pass indicator. Now 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
              LOA      SWCH1   FIRST TIME IN?
              ORA      A        NO IF NOT ZERO
              JNZ      ENTRY    GET CHARACTER FROM READER
              CALL     CHR      MAKE IT NON ZERO
              ORA      A        NOW PROCESS THE RECEIVED CHARACTER
              JZ       ENTRY    SCAN PAST LEADING ZEROS
              STA      SWCH1
              JMP      GOTON

DLOOP         CALL     CHR
              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 CHR
              CPI      7FH
              JZ       DLOOP    BYPASS SYSTEM FILL CHR
              INX      H
              JMP      DLOOP

REWIND        CALL     REND    ROUTINE FOR DEVICE REWIND
              MV1      A,8      PRETEND IT'S THE FIRST TIME THROUGH
              STA      SWCH1
              JMP      ENTRY    GO SCAN PAST LEADING ZEROS AGAIN

* CHR        THIS ROUTINE GETS ONE CHARACTER FROM THE
          INPUT DEVICE

* CHR        EQU $
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.

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, ($D_{16}$), 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 $0 $0 $0 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:

$D9$0: 16 30 30 30 35 20 40 44 41 20 41 42 55 46 $D$
$D9$0: $01
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 39 and 39 and that the $00_{16}$ value identifies the end of each RECORD.

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.
CRLF EQU $E216H

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

SCRN EQU $E389H

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

DUMO EQU $E56FH
HOUT EQU $E577H
HOTB EQU $E586H
DOTU EQU $E34BH
COUT EQU $E353H
COTB EQU $E359H

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
DOTU Output as Decimal
COUT Output as Octal
COTB Output as above but with space following

Up to two levels of stack may be used.

Title: ADOUT 16 Bit Value Output

Function: 

ADOUT EQU $E55CH
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.
BLKO EQU $E361H

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.

The following Application Bulletins specify and describe the system subroutines for converting binary values to their ASCII representation.
**Title:** Binary to ASCII Conversion  
**Release Date:** 10/7/76  
**Revision No.:** 6  
**Level No.:** K3  
**Page:** 2 of 2

**Function:**

```assembly
BINH EQU $E390H  
BIND EQU $E3A0H  
BINO EQU $E3C8H
```

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 MCON, 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  
**Release Date:** 11/10/76  
**Revision No.:** 6  
**Level No.:** K4  
**Page:** 1 of 1

**Function:**

```assembly
SEAR EQU $E257H
```

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.

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

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.

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

<table>
<thead>
<tr>
<th>NAME</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMODE</td>
<td>ASCII to Current Mode Setting</td>
</tr>
<tr>
<td>ADEC</td>
<td>ASCII to Decimal</td>
</tr>
<tr>
<td>AHEX</td>
<td>ASCII to Hexadecimal</td>
</tr>
<tr>
<td>AOCT</td>
<td>ASCII to Octal</td>
</tr>
</tbody>
</table>

```
EMODE EQU $E2F3H
ADEC EQU $E09AH
AHEX EQU $E1FAH
AOCT EQU $E333H
```

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.
Title: System Conversion Routines

Entry Point: As Specified

<table>
<thead>
<tr>
<th>Entry Conditions</th>
<th>A</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>C Low byte pointer</td>
<td>S</td>
<td>P</td>
</tr>
<tr>
<td>D</td>
<td>CY</td>
<td>AC</td>
</tr>
<tr>
<td>E</td>
<td>Z</td>
<td></td>
</tr>
</tbody>
</table>

Exit Conditions

<table>
<thead>
<tr>
<th>Exit Conditions</th>
<th>A</th>
<th>H High order value</th>
</tr>
</thead>
<tbody>
<tr>
<td>B Through scan</td>
<td>L Low order value</td>
<td></td>
</tr>
<tr>
<td>C Through scan</td>
<td>S P Error flag</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>CY Flag AC</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>Z</td>
<td></td>
</tr>
</tbody>
</table>

Buffer Area used: _______ to _______

Registers Altered: All

Subroutines called:

Subroutines levels used:

Stack level used: 1

Comments:

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

© 1977 Processor Technology Corporation
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 IDDR 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 19 with for its execution. Assume the ALS-8 received the command:

```
POTTAM /ABCDE/ 1057 ALE5
```
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 ADCF
JC WHAT  CARRY SET MEANS ERROR
SLLD PARAL STORE FIRST
LXI B,ABUF+7
LDA X  GET CHR OF SECOND PARAMETER
ORA A  SET FLAGS
JZ WHAT NO PARAMETER
CALL ADRC CONVET 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 SCNE GO DO THE "S" PART
CPI 'A' 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, GETP, 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 $D1E4H INPUT BUFFER
ABUF EQU $D1DEH ASCII BUFFER
FBUF EQU $D077H NAME BUFFER
BBUF EQU $D0EFH BINARY BUFFER

VCHK EQU $E51CH
WHAT EQU $E77DH
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 0.000,001 to 0.999,999) 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 \(2^{15}-1\) (7FFFH) values in length, and determines that the execution time per number is \(10^{-9}\) sec. 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 = 7FFFH), as shown in the first execution (Figure 5) each slot contains the required 80 values except the last slot from which word FF80 is missing. In the second execution, only 1/4 of the possible values are sampled (count limit = 1FFFH). It is apparent that the values cluster around the proper average value (20H), and range from 11H to 2FH, confirming the evenness of the number distribution.
0000 000C • RANDE2
0000 0010 •
0000 0015 • WRITTEN BY ERIC G. FAHISON, 11/20/76
0000 0020 •
0000 0025 • RANDE2 IS A 15 BIT PSEUDO RANDOM
0000 0030 • NUMBER GENERATOR. EACH
0000 0035 • FRN IS OBTAINED FROM THE
0000 0040 • ONE BY SETTING CARRY=PARITY OF BITS
0000 0045 • 1 AND 2 OF THE LEAST SIGNIFICANT BYTE
0000 0050 • (BIT 0 IS ALWAYS CEPD AND DONE)
0000 0055 • TWO FRNS ON THE TWO BYTE WORD.
0000 0060 • STEPPING BY 21 FRIEDNES NEAR NEIGHBOR
0000 0065 • CORRELATIONS AND STILL YIELDS A FRN
0000 0070 • IN UNDER 920 MICROSECONDS, FASTER.
0000 0075 • EXECUTION CAN BE OBTAINED BY REPLACING
0000 0080 • 23 IN LINE 145 BY 19, 17, 13, 11 OR EVEN 1.
0000 0085 • FOR WHICH THE CALL TIME DEPENS TO 110 MICRO
0000 0090 • SEC AT THE COST OF INCREASING NEAR-NEIGHBOR
0000 0095 • CORRELATIONS. ALL 32767 15-BIT FRN'S
0000 0100 • ARE GENERATED, AND THEN THE CYCLE REPEATS.
0000 0105 • FFEEH DOES NOT OCCUR.
0000 0110 •
0000 0115 • TO RANDOMISE FROM RUN TO RUN, RESET FRNUM
0000 0120 • USING ALT-8 COMMAND ENTR, PERHAPS USING THE
0000 0125 • DATE OR TIME. ODD VALUES ARE EQUIVALENT TO
0000 0130 • (VALUES-1), DON'T USE FFEEH OR FFEEH.
0000 0135 •
0000 0140 •
0000 0145 • PUSH B
0000 0150 • MVI B, 02 SET LOOP COUNTER
0000 0155 • LCALL FRNAM GET LAST FRN
0000 0160 •
0000 0165 •
0000 0170 • LDPP ANI 6 GET BITS 1-2
0000 0175 • STC CARRY=PARITY
0000 0180 •
0000 0185 •
0000 0190 • JPE CONT
0000 0195 •
0000 019A • MVI H,A SHFT H
0000 01A0 •
0000 01A5 • MVI D,L NOW L
0000 01B0 •
0000 01B5 • MVI A,L NEW FRN
0000 01C0 •
0000 01C5 • MVI L,R DCR B
0000 01D0 •
0000 01D5 • MVI H,P SLB FRNUM REFRM
0000 01E0 •
0000 01E5 • MVI B, E XCHG PUT TO B,E
0000 01F0 •
0000 01F5 • MVI D,P POP E
0000 0200 •
0000 0205 • MVI C,R RET
0000 0210 •
0000 0215 • 8255 FRNUM DW 255CH AS GOOD AS ANY STARTER
0000 0220 •
0000 0225 •
0000 0230 •
0000 0235 •
0000 0240 •
0000 0245 •
0000 0250 •
0000 0255 •

**Figure 1.** Assembly listing of RANDE2.

---

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

<table>
<thead>
<tr>
<th>EXEC B00</th>
</tr>
</thead>
<tbody>
<tr>
<td>A249</td>
</tr>
<tr>
<td>F3AE</td>
</tr>
<tr>
<td>BEFC</td>
</tr>
<tr>
<td>16FE</td>
</tr>
<tr>
<td>F9D4</td>
</tr>
<tr>
<td>2C08</td>
</tr>
<tr>
<td>AFRM</td>
</tr>
<tr>
<td>7E42</td>
</tr>
<tr>
<td>CEBE</td>
</tr>
<tr>
<td>EE32</td>
</tr>
<tr>
<td>94CC</td>
</tr>
<tr>
<td>E796</td>
</tr>
<tr>
<td>CBEE</td>
</tr>
<tr>
<td>A040</td>
</tr>
<tr>
<td>2662</td>
</tr>
</tbody>
</table>

**READY**
Figure 3. This program checks the length of the PRN sequence, and shows that the execution time per number is \( \frac{30.2 \text{ sec}}{32,767} \) or approximately \( \frac{0.00092 \text{ sec}}{\text{per number}} \)."
The following Video Display Module Driver was submitted by:

Don Minilitch
29 Hoyt Street
Stamford, CT 06905

C900 1000 *VDM - VIDEO DISPLAY MODULE DRIVER
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 1070 VDM PUSH H
C900 1080 LMD VDMFT
C901 1090 GET NEXT CHAR ADDR
C904 1090 MOV A,B
C905 1090 INSERT THE CHAR
C906 1100 CPI CR
C908 1120 IS IT CR CHAR?
C90C 1130 YES
C90E 1140 IS IT OTHER CTL CHAR?
C912 1150 YES - IGNORE IT
C912 1160 JZ VDMRT
C916 1170 CPI DEL
C918 1180 IS IT DEL CHAR?
C91C 1190 YES - IGNORE IT
C91C 1210 VDMX GOTO CALC NXT CHAR
C91C 1220 VDMCR MOV A,L CALC NXT LINE ADDR
C916 1230 MOV A,H
C91E 1240 VDMFT
C920 1250 VDMRT
C924 1260 MO VDPAG+4 IS IT OFF THE VDM
C926 1270 VDPAG SET TO START OF VDM
C928 1280 VDPAG SAVE CHAR ADDR
C92C 1290 VDPAG+80H INSERT CURSOR
C92E CALL CALC SCREEN OFFSET
C930 1300 MO VDPAG+10 SET SCREEN OFFSET
C932 1310 OFFH SEE IF HARDCOPY WANTED
C934 1320 CALL COPY CHAR OR NULL
C936 1330 CALL SYSUB
C938 1340 CALL POP
C93A 1350 CALL POP
C93C 1360 CALL POP
C93E 1370 CALL POP
C940 1380 CALL POP
The following Utility Routines for use with the ALS-8 were submitted by:

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

UTILITY ROUTINES TO BE USED WITH THE ALS-8
CRRM, CRRS, CRRU, CRRD, CRRN, NUMB

ALL CASSETTE ROUTINES ARE SPECIFICALLY FOR THE TARBE LL CASSETTE INTERFACE.

CARR: CASSETTE READS DATA FROM TAPE & Puts IT IN MEMORY
CARR
- 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 (MAKL).

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

CARR XXXX YYYY
- THIS COMMAND IS USED FOR FILES THAT HAVE BEEN
  BLENDED ON TO 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.

CASSW: CASSETTE WRITES DATA ON TAPE FROM MEMORY
CASSW
- 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.

CASSW 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.
RENUN: RENUMBER A SOURCE FILE

RENUN XXXX YYYY
- THIS ROUTINE WILL RENUMBER BY ANY GIVEN
INCREMENT. THE INCREMENT MUST BE SPECIFIED
AND MAY PUSH A LINE NUMBER PAST THE MAX 9999.
VALUE IN WHICH CASE THE FIFTH DIGIT IS DROPPED.
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/ YYYY
- 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 XXXX.

NUMB YYYY 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 NUML DO A CONTROL X AND READY WILL
BE PRINTED.
1690  * WRITE OUT A BYTE TO CASSETTE
1700  COUT  PUSH  PSW
1710  CLOP  IN  CASC  READ CASSETTE STATUS
1720  ANI  2BH
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+1
1960  JMP  GO
1970  * BEGIN LN AT 0000
1980  INTLN  LXI  D,ABUF+7
1990  MV1  A,'0'
2000  MV1  B,4
2010  ZER0  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  ENDCX
2080  CALL  ADD
2090  CALL  LOAD
2100  CALL  ADDR
2110  JMP  GO

2120  * ROUTINE TO SUBTRACT ASCII BIAS FROM NUMBERS
2130  BIAS  LHLD  INCPT
2140  XCHG
2150  MV1  B,4
2160  ABIN  LDAX  D
2170  SUI  3BH
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  18

2250  MOV  M.A
2260  DCX  H
2270  MOV  M.A
2280  INX  D
2290  CPI  '9'+1
2300  RC
2310  CALL  INC
2320  RET
2330  * IS THIS THE END?
2340  ENDCX  MOV  M.A
2350  CPI  1
2360  RMZ
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  MV1  D,0
2430  CALL  CLEAR
2440  CALL  LIST
2450  JMP  EORNS

2460  * ROUTINE TO MOVE CONTENTS OF NO. BUFFER TO FILE
2470  LOAD  MV1  B,4
2480  MOV  A,B
2490  CALL  ADDR  ADD 4 TO ADDRESS
2500  LOAD1  LDAX  D
2510  MOV  M.A
2520  DCX  D
2530  DCX  H
2540  DCR  B
2550  JNZ  LOAD1
2560  MOV  M.A
2570  RET

2580  * ADD INCREMENT TO WORK BUFFER
2590  ADD  PUSH  H
2600  LHLD  INCPT  POINT TO INC VALUE BUFFER
2610  XCHG
2620  LHLD  LIMPT  POINT TO LINE NO. BUFFER
2630  MV1  E,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
2080 CLEAR XRA A
2082 MOVE M.A
2084 DCR B
2086 JNZ CLR
2087 RET

2088 * ROUTINE TO CONCATENATE OR INSERT FILES INTO ONE ANOTHER
2089 * FIRST ARG CONTAINS FILE TO BE INSERTED
2090 * SECOND ARG CONTAINS LINE NO. INSERT WILL FOLLOW

2091 INCR LXI H.ABUF
2092 CALL FIND
2093 MOV R.M
2094 CALL ADDR
2095 DCX H
2096 MV1 M.2
2097 SMDS INS

2098 * POINT TO FILE TO BE INSERTED AND GET BDFP & EDFF
2099 CALL FSEA
2100 LXI D.NLEN
2101 DAD D
2102 LDAX D
2103 ORA A
2104 JZ Win
2105 CALL LDMD
2106 PSH B
2107 MOV A.D
2108 SUB B
2109 MOV L.A
2110 MOV A.E
2111 SBB C
2112 MOV H.A
2113 XCHG
2114 SMDS EDFP
2115 PSH H
2116 DAR D
2117 SMDS EDFP
2118 POP D

2119 * CREATE SPACE FOR INSERTION
2120 MV1 C.2
2121 CALL RMV

2122 * MOVE FILE TO BE INSERTED INTO SPACE
2123 POP B
2124 MOV E.B
2125 MOV D.C
2126 SMDS INS
2127 MV1 M.ffc
2128 INX H
2129 MV1 C.1
2130 CALL LMDV

2131 * SET UP TO RENUMBER FILES BY INCR OF 1
2132 XRA A
2133 LXI D.ABUF INC VALUE BUFFER POINTER
2134 MV1 B.3
2135 PLAC STAX D

2136 LXI D.INX

2370 DCR B
2372 JNZ PLACE
2374 INR A
2376 STAX D
2378 INX D
2380 JMP INTLH RENUM COMBINED FILES

2381 * ROUTINE TO PROVIDE AUTOMATIC LINE NUMBERING
2382 * FIRST ARG DEFINES STARTING LINE NO.

2383 * OPTIONAL SEC ARG DEFINES INC VALUE - DEFAULTS TO 5
2384 * CTRL/X RETURNS YOU TO NORMAL OPERATION

2385 NUMB CALL CRLF
2386 CALL ARG1 CHECK FOR FIRST ARG
2387 JZ WHAI
2389 XRA A SET UP BEGINNING OF LINE FLAG
2390 STA NFLAG
2391 LXI H.SAVE STORE NEW LINE NO & INC VALUE POINTERS
2392 SHLD INCP
2393 LXI H.SAVE+4
2394 SHLD LINP

2395 * ROUTINE TO SAVE ASCII ARG ABUF & ABUF+7
2396 * ABUF TO ABUF+10 IS USED BY ALS-8 DELETE ROUTINE
2397 * LXI H.ABUF POINT TO INC VALUE BUFFER
2398 CALL LDMD LOAD IN GECD
2399 LXI H.ABUF+7 POINT TO INC VALUE
2400 CALL LDMD
2401 LXI H.SAVE+3 POINT TO SAVE BUFFER
2402 CALL STOM COPY INC VALUE
2403 CMP M CHECK FOR SECOND ARG
2404 JZ INC5
2406 CALL BIAS SUB ASCII BIAS
2407 JMP NEWIN ONWARD WITH PRESCRIBED INC

2408 * INITIALIZE INCREMENT TO 5
2409 INC5 MV1 B.3
2411 INC MV1 M.8
2413 INX H
2415 DCR B
2417 JNZ LINP
2419 MV1 M.5

2420 * CHANGE INPUT DRIVER
2421 NEWIN LHLD INPB+1
2422 SMDS JIVE+1
2424 LXI H.BOGGED
2425 SMDS INPB+1
2426 JMP SYSC

2427 * START OF NEW INPUT DRIVER
2428 BOGED LDA NFLAG
2429 JNZ JIVE NOT A NEW LINE THEN JIVE TO SYSTEM INPUT
2430 MV1 E.2+5 ADVANCE CHR COUNT
2432 PSH D 4 SAVE
2434 MV1 A.1 SET FLAG TO BE CLEARED BY A CR
2436 STA NFLAG

2437 PRINT LINE NO.
3920  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 A, M  TO INPUT STREAM
3980  MOV D, A  SAVE ACC
3990  CALL OUTPB
4000  INX H  NEXT PLEASE
4010  INX D
4020  DCR C
4030  JNZ VIEW
4040  * PRINT A SPACE
4050  MVI A, 0
4060  MOV D, A
4070  MOV B, A
4080  CALL OUTPB
4090  INX H
4100  CALL ADDH  ADD INC TO LINE NO
4110  POP D  CHRL 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  STR NFLAG
4200  MOV A, B
4210  RET
4220  * RESTORE POINTERS
4230  EXIT LXI H, ABUF
4240  SMLD INCPT
4250  LXI H, ABUF+7
4260  SMLD LINPT
4270  JMP EORMS BACK TO MONITOR & PRINT READY
4280  *
4290  * DEFINE CONSTANTS & STORAGE
4300  ASCR EQU 00H
4310  NMLN EQU 0005H
4320  TEMP EQU 0000H
4330  CASC EQU 06H
4340  CASC EQU 6FH
4350  PSW EQU 6
4360  NFLAG DS 1
4370  SAVE DS 0
4380  INCPT DW ABUF
4390  LINPT DW ABUF+7
4400  *
4410  ** ALS-8 ROUTINE ADDRESSES
4420  *
4430  BOFF EQU 00035H
4440  EOFF EQU 00007H
4450  MAXL EQU 00009H
4460  IMPH EQU 0000CH
4470  OUTPB EQU 000CFH
4480  ABUF EQU 000DFH
The following Page Printer Program for the ALS-8
was submitted by:

Bill Gunn
875 West Broadway
Vancouver, B.C. V5L1J9

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 MARGIN IN COUNTER
0150 PRINT LXI H,ABUF+2 POINT TO SECOND LAST CHAR IN PARAM BUFF
0160 MOV A, M
0170 ORR A, ANYTHING THERE
0180 JZ WHAT 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 CALL 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 LCNT LINE COUNT
0260 STA LCNT LAST CHAR OUTPUTED
0270 MOV 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 MOV M, '0' INSTALL AN ASCII ZERO IN THE LAST BYTE
0340 CALL CRLF RESET PRINTER HEAD
0350 CALL PNUM 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 SMLO OUTPUT+1
0400 SET UP FOR TEXT COMMAND
0410 LXI H,TEXT POINT TO ASCII "TEXT" IN ALS-8
0420 CALL LCDM LOAD ASCII CHARs INTO REG B C D E
0430 LXI H,ABUF+3 POINT TO INPUT COMMAND BUFFER
0440 CALL STOM STUFF ASCII TEXT HERE
0450 CALL 2BUF 0 ZERO PARAMETER BUFFER. ABUF TO ABUF+2R
0460 LXI H,SYSE+6 ADDRESS TO BE USED AFTER THE NEXT RET

0470 PUSH H PUT IT ON THE STACK
0480 LXI H,1BUF+4 POINT TO JUST AFTER TEXT
0490 JMP CENTER+4 ADD A CR & EOF INDICATOR BEFORE COMP STRINGS
0500 - CENTER IS A POINT IN THE ALS-8 WHERE I MAY
0510 - CONTINUE AFTER ENTERING A COMMAND VIA THIS PROGRAM
0520 GENERAL DRIVER FOR VDM-1 AND LA3B
0530 SENSE SWITCH 8 DOWN VDM DRIVER IS USED
0540 SENSE SWITCH 8 UP LA3B DRIVER IS USED
0550 FLY IN SENSE
0570 RAL
0580 JC NOCPY
0590 RAR
0600 RAR
0610 JC HARD
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 LCNT YES - INCREMENT LINE COUNT
0690 INR A
0700 CPI 'D' PRINTED 51 LINES?
0710 JNZ STCNT NO - STORE LINE COUNT
0720 MOV MV1 A, BSH YES - SET FLAGS. IGNORE DELETES & SET TOP OF F
0730 STA DFLAG SAVE FLAGS
0740 JMP OUTS OUTPUT CARRIAGE RETURN, LINE FEED & TWO DELETES
0750 STCNT STRCNT LINE COUNT
0760 LDA SAVE GET LAST CHAR PRINTED
0770 CPI IS IT A SPACE?
0780 JZ KEEPM YES - HOLD PRINT HEAD WHERE IT IS
0790 MV1 A, BSH NO - SET FLAG TO IGNORE NEXT TWO DELETE CHAR
0800 STA DFLAG SAVE FLAGS
0810 JMP OUTS
0820 KEEPM POP B YES - GET OUT OF CRLF ROUTINE BY PULLING RET
0830 ADDR FROM STACK
0840 MV1 A, BSHL SKIP THE CR & DO A LINE FEED ONLY
0850 CALL OUTS BECAUSE YOU'VE ALREADY SET LEFT MARGIN
0860 MV1 A, '2' SET UP LAST CHAR PRINTED AS LEFT MARGIN 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 MOV MV1 A, BSH NO - THEN OUTPUT IT
0910 TEST FOR TWO DELETE CHAR OUTPUT AFTER A CR
0920 LDA DFLAG GET FLAGS
0930 RAL IF MOST SIG BIT SET, OUTPUT DELETE AS A NULL
0940 JC OUTS
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 BUTF ER & OUTPUT THAT MANY SPACES
0990 LA3B DRIVER
1000 OUTS IN USTA CHECK STATUS
1018 ANI BAH READY TO OUTPUT?
1028 JZ OUTS NO - CHECK AGAIN
1590  RZ
1600   ADD   D,A   ADD IN APPROPRIATE BINARY VALUE OF DECIMAL DIGIT   
1610   MOV   A,D   & STORE IN D   
1620   JMP   CONV+4   
1630   * ROUTINE FOR MULTIPLE CHAR OUTPUT   
1640   OVER   MVI   C,63H   MOVE PAGE NO OVER TO THE RIGHT   
1650   MVI   D,D7   BY INSERTING MULTIPLE SPACES   
1660   JMP   SHOT   
1670   PAGE   MVI   C,7   SKIP OVER PAGE PERFORATIONS   
1680   MVI   D,ASLF   BY DOING MULTIPLE LINE FEEDS   
1690   SHOT   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   SPED   TEST FOR SPECIAL CASE OF NO CHAR TO BE OUTPUT   
1730   CALL   OUTS   
1740   JMP   SPACE   AGAIN   
1750   SPED   POP   B   FINISHED - SO RESTORE ORG CHAR TO BE OUTPUT   
1760   XRA   A   CLEAR ACC   
1770   RET   
1780   * INCREMENT & PRINT CONTENTS OF PAGE NO. BUFFER   
1790   PNUM   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,00H   YES - REPE AT 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,00H   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   CALF   OUTPUT CR, LF & NULLS   
1950   RET   
1960   MSG   ASC   *PAGE*   
1970   BUFF   DS   4   
1980   DW   002EH   A PERIOD FOLLOWED BY A CR   
1990   *   
2000   * DEFINE CONSTANTS   
2010   VIDEO   EDU   0FE7H   
2020   ASLF   EDU   0AH   
2030   ASCR   EDU   0CH   
2040   DELETE   EDU   7FH   
2050   CHANS   EDU   3   
2060   USTR   EDU   0   
2070   DAV   EDU   40H   
2080   TBE   EDU   88H   
2090   OUT64   EDU   080CH   
2100   ABUF   EDU   000FH   
2110   IBUF   EDU   014H   
2120   EORMS   EDU   0060H   
2130   SVS   EDU   0870H   
2140   CENTRY   EDU   0193H
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 CARRIAGE 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 (CLBPC) 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 RETRIEVE 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.B/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 DB0SH. 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 FCNK 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.
0010 ****************************
0010 3C90      0400   PGBRK EQU $  END OF PAGE
0010 3C91      0500   XRA A
0010 3C92      0510   STA LCTR CLEAR LINE COUNTER
0010 3C93      0520   PUSh B
0010 3C94      0530   MVI C.33H SKIP 3 LINES FIRST
0010 3C95      0540   CALL CRLF
0010 3C96      0550   CALL CONV CONVERT BINARY TO ASCII
0010 3C97      0560   PUSh H
0010 3C98      0570   MVI B.28H SPACES
0010 3C99      0580   MVI C.27 27 SPACES
0010 3C9A      0590   CALL PUT OUTPUT SPACES BEFORE "PAGE"
0010 3C9B      05A0   LXI H,MSG1
0010 3C9C      05B0   CALL MSG OUTPUT PAGE NUMBER
0010 3C9D      05C0   MVI C.84H SKIP 4 LINES AFTER
0010 3C9E      05D0   CALL CRLF
0010 3C9F      05E0   MVI B.2DH HYPHENS
0010 3CA0      05F0   CALL RCR OUTPUT SEPARATOR
0010 3CA1      0600   MVI C.5 SKIP 6 LINES LAST
0010 3CA2      0610   CALL CRLF
0010 3CA3      0620   POP H
0010 3CA4      0630   POP B
0010 3CA5      0640   JMP OUT FINALLY OUTPUT CHARACTER
0010 3CA6      0650   0730  CRLF EQU $  REG C HOLDS LOOP FACTO R
0010 3CA7      0660   MVI B.80H CARRIAGE RETURN
0010 3CA8      0670   CALL OUT
0010 3CA9      0680   MVI B.8AH LINE FEED
0010 3CAA      0690   CALL OUT
0010 3CAB      06A0   DCR C
0010 3CAC      06B0   JMP CRLF LOOP BACK
0010 3CAD      06C0   0820  CONV EQU $  CONVERTS PAGE COUNT TO
0010 3CBE      06D0   ASCII
0010 3CF0      06E0   LDA PGCT PAGE COUNT
0010 3CF1      06F0   INR A
0010 3CF2      0700   STA PGCT SAVE
0010 3CF3      0710   DAA
0010 3CF4      0720   MOV B.A SAVE IN B
0010 3CF5      0730   ANI $FH FIRST 4 BITS
0010 3CF6      0740   ORI $3B MAKE IT ASCII
0010 3CF7      0750   STA ASCI
0010 3CF8      0760   MOV A.B
0010 3CF9      0770   RRC
0010 3CFA      0780   0490 ****************************
3CC0 0810 ***************
3CC0 0800 * THIS PROGRAM LOADS SAVED PROGRAMS *
3CC0 0800 * DIRECTLY INTO CORE STARTING AT THE *
3CC0 0800 * LOCATION POINTED TO BY THE CURRENT *
3CC0 0800 * FILE POINTER AT "DOES" *
3CC0 0800 * FILE NAME IS MATCHED AGAINST ENTRY *
3CC0 0800 * IN INPUT LINE AREA *
3CC0 0800 * IN TAPED CLEAR TAPE REGISTER *
3CC0 DB $03
3CC2 CD 2C 3D 0100 CALL CRLF
3CC5 DB $03 0110 IN TAPED
3CC7 
3CC7 0130 * GET FILE NAME *
3CC7 0140 *
3CC7 21 E4 D1 0150 LXI H,IBUF BUFFER POINT
3CCE 0E 32 0160 MOV VI C,50 MAX SEARCH LENGTH
3CCC 07E 0170 FNAM EQU $
3CCC 7E 0180 MOV A,M
3CCD 23 0190 INX H
3CCD 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 C5 CC 3C 0230 JNZ FNAM LOOP BACK
3CD7 0240 *
3CD7 0250 * STORE FILE NAME LOCATION *
3CD7 0260 *
3CD7 2B 0270 DCX H BACKSPACE ONE
3CDB 22 51 3D 0280 SHLD STARTLOC
3CDB 0290 *
3CDB 0300 * TRY TO MATCH WITH TAPE *
3CDB 0310 *
3CDB 0320 MATCH EQU $
3CDB 0330 LHL D STARTLOC
3CDE 0E 0340 MOV G,M
3CDE 4E 0350 MATI EQU $
3CDE CD 11 3D 0360 CALL TAPE GET DATA
3CEE B9 0370 CMP C
3CEE CF DF 3C 0380 JNZ MATH NO MATCH
3CEF 0390 MAT2 EQU $
3CEF 23 03A0 INX H
3CEF 03B0 *
3CEF 04C0 * TEST MIDDLE CHARACTERS *
3CEF 04D0 *
3CEF 04E0 MOV G,M
3CEF 04F0 CALL TAPE
3CF0 FE 3E 0460 CPI *
3CF0 CA F7 3C 0470 JZ MAT3 END OF TAPE NAME?
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 1
3CFD 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
3DE0 0590 MAIN EQU $
3DE0 05A0 CALL TAPE
3DE0 70 05B0 MOV H,B
3DE0 05C0 INX H
3DE6 FE 81 05D0 CPI I TEST FOR END OF FILE
3DE8 C2 81 3D 05E0 JNZ MAIN
3DE8 21 32 3D 05F0 LXI H,MSG
3DEE C3 E0 E7 0600 JMP MESS
3DF1 0610 *
3DF1 0620 * GET DATA FROM TAPE *
3DF1 0630 *
3DF1 0640 *
3DF1 0650 *
3DF1 0660 TAPE EQU $
3DF1 0670 IN KBSTAT
3DF2 E6 81 0680 ANI MASK1
3DF2 FA 03 0690 JZ TAPE2
3DF2 0700 IN KBDATA
3DF3 EA 6F 7F 0710 ANI 127
3DF3 FE 8B 0720 CPI ESC
3DF3 FA 0730 JZ EORMS RETURN IMMEDIATELY
3DF4 CA 60 E0 0740 GLD TAPE2 EQU $
3DF4 0750 IN TAPES
3DF4 0760 IN ANI MASK1
3DF4 0770 IN TAPE
3DF5 0780 IN TAPED GET DATA
3DF6 0790 IN BA SAVE
3DF6 07A0 RET
3DF6 07B0 *
3DF6 07C0 SEND CR TO VDM
3DF8 0770 *
3DF8 0780 IN CR LF EQU $
3DF8 0790 MOV B,ODH
3DF8 07A0 CALL VDM
3DFD C9 07B0 RET
3DFD 07C0 *
3DFD 07D0 END OF PROGRAM
3DFD 07E0 *
3DFD 52 45 53 5A 07F0 MSG ASC "RESTORE COMPLETE, FCHK FILE 1"
0010 ********************
0018 0020 * THIS PROGRAM WRITTEN WITH THE ALS-* *
0028 0030 * SYSTEM BY DENNIS H. ROSENTHAL *
0038 0048 * THIS PROGRAM SAVES CURRENT FILE ON TAPE*
0048 0058 * ALONG WITH NAME, NAME IS ENCLOSED BY *
0058 0068 * THE CHARACTERS '<' >'. ENTER TWO DUMMY *
0068 0078 * NUMBERS TO SATISFY NORMAL PARAMETER *
0078 0088 * CHECKING *
0088 0098 ********************
0108 0118 * FIRST OUTPUT FILE NAME TO TAPE *
0118 0128 ********************
0138 CALL CRLF RETURN CURSOR
0148 * PUT INPUT BUFFER LOC IN H,L
0158 0168 LXI H,IBUF
0168 0178 MVI C,68h MAX LENGTH TO SEARCH
0178 0188 0170 F1 MOV A,M
0188 0198 0168 INX H
0198 0208 0198 DCR C
0208 0218 0200 JP Z WHAT ERROR CONDITION *
0218 0228 0210 CPI *'
0228 0238 0220 JNZ FI
0238 0248 0238 MOV B,A
0248 0258 0240 F2 CALL OUTN PUT NAME ON TAPE
0258 0268 0258 MOV A,M
0268 0278 0260 INX H
0278 0288 0270 CPI *'
0288 0298 MOV B,A
0298 0308 * CONTINUE UNTIL END INDICATOR IS FOUND
0308 0318 0300 JNZ FE
0318 0328 0318 CALL OUTN
0328 0338 0328 MVI B,0DH CR
0338 0348 CALL OUTN
0348 0358 0340 *************
0358 0368 * NOW PUT FILE TO TAPE ***
0368 0378 ********************
0378 0388 CALL CRLF
0388 0398 LHL D 85 search
0398 0408 MAIN EQU $5
0408 0418 0408 MOV B,M TRANSFER TO REG
0418 0428 0418 INX H
0428 0438 0428 CALL OUTN TRANSFER TO TAPE
0438 0448 0438 MOV A,B CHECK FOR END OF FILE
0448 0458 0448 CPI 1 END OF FILE ?
0458 0468 0458 JNZ MAIN
0468 0478 0468 JMP EORMS BACK TO SYSTEM
0478 0488 0478 *************
0488 0498 0488 SEND REG B TO TAPE *
3D9B 0490 ****************
3D9B 0500 OUTN EQU $3
3D9B DB 02 0510 IN TAPES
3D9B E6 02 0520 ANI MASK1
3D9F CA 9B 3D 0530 JZ OUTN
3DA2 7E 0540 MOV A,B
3DA2 D3 03 0550 OUT TAPED
3DA5 C9 0560 RET
3DA6 0570 ****************
3DA6 0580 * SEND CR TO VDM *
3DA6 0590 ****************
3DA6 0600 CRLF EQU $5
3DA6 0610 E0H MVI B,0DH
3DA6 CD 00 DE 0620 CALL VDM
3DAB C9 0630 RET
3DAC 0640 MASK1 EQU 2
3DAC 0650 FPTR EQU 30005H
3DAC 0660 **********************
3DAC 0670 * END OF PGM *
3DAC 0680 **********************

CRLF 3DA6 0130 0370
F1 3D48 0220
F2 3D74 0380
FPTR D005 0380
MAIN 3D8D 0450
MASK1 0002 0520
OUTN 3D9B 0240 0310 0330 0428 0530