VOLUME I

INTEL SOFTWARE INDEX
AND TECHNOLOGY EXCHANGE

PROGRAM LIBRARY MANUAL
INTEL USER'S LIBRARY
PROGRAM MANUAL

APRIL 1978

TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>SECTION</th>
<th>CONTENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>INTRODUCTION</td>
</tr>
<tr>
<td>2</td>
<td>SALES OFFICE LISTINGS</td>
</tr>
<tr>
<td>3</td>
<td>MEMBERSHIP PROGRAM SUBMITTAL</td>
</tr>
<tr>
<td>4</td>
<td>OPERATING, TESTING, DEBUGGING PROGRAMS</td>
</tr>
<tr>
<td>5</td>
<td>MATH NUMERICAL MANIPULATION PROGRAMS</td>
</tr>
<tr>
<td>6</td>
<td>CROSS PRODUCT SOFTWARE</td>
</tr>
<tr>
<td>7</td>
<td>UNCLASSIFIED PROGRAMS</td>
</tr>
<tr>
<td>8</td>
<td>GAMES</td>
</tr>
<tr>
<td>9</td>
<td>RESIDENT LANGUAGE TRANSLATORS</td>
</tr>
<tr>
<td>10</td>
<td>CROSS REFERENCE TO PL/M PROGRAMS</td>
</tr>
<tr>
<td>11</td>
<td>RMX-80</td>
</tr>
</tbody>
</table>
INTRODUCTION

PROGRAM LIBRARY DEFINITION

Welcome to Insitex™, Intel's software index and technology exchange. We think you will be impressed and enthusiastic about your new library of programs. Within the contents of Insitex™, you will find a diversified collection of programs, subroutines, procedures, and macros written by the users of Intel's microprocessors. Thanks to your contributions and continued participation in this Library, we are able to maintain and facilitate this unrestricted interchange of non-proprietary programs among our subscribers.

Each individual subscriber is responsible for making his own independent analysis to determine which programs in the Library are applicable to his requirements. The programs in the Library have been collected and published for your convenience and are not guaranteed nor supported by Intel. The verification of each program will be left up to the individual user.

PROGRAM LIBRARY MANUAL

This Program Library Manual includes submittal forms and source listings for those programs three pages and under. Longer programs are represented solely by an abstract which indicates the function of the routine, the required hardware and software and memory requirements. Should you find that you are interested in one of the longer programs, the listing and paper tape may be purchased from Intel at reproduction cost (see Section 3, Ordering Procedure). The Program Library Manual is divided into five major areas of program classification. At the beginning of each section you will find a listing of the contents of that section. At the end of the manual is an alphabetical index of all programs in the Library.

SOURCE TAPES

Source paper tapes and listings of most programs in the Program Library Manual are available for purchase at a minimum reproduction cost. For further information see Section 3, Ordering Procedure.

PROGRAM DISKETTES

Program Diskettes are available for most programs in your Program Library Manual. See Section 3 for prices and ordering procedure.

LIBRARY UPDATES

By filling out and mailing the REGISTRATION CARD in the front of your Library you will be placed on the Membership List and will receive the bi-monthly Program Library Update Package. This package will include all new programs which have been accepted into the Library over the preceding two months. During your subscription year you will not only receive the Library's current programs, but also six update packages. However, unless you mail in your Registration Card you will not receive these updates.

QUALITY ASSURANCE

As it is impossible to test each program submitted to the User’s Library, the verification of each program will be left up to users. Included in this section are several “Program Certification and Review Forms”. The purpose of this form is to determine whether or not a program functions as its author claims. This form should not be used as a vehicle for program enhancement or elaboration. We appreciate your responses which let us know the accuracy of library programs.

PROGRAM REVISION

Program revisions submitted by the original author are handled in a manner identical to original program submission. That is, the author should submit a completed “Microcomputer User’s Library Submittal Form” along with the entire package or relevant documentation. The program which has been revised should be referred to in a cover letter by the reference number which appears in the upper right-hand corner of the original Library Submittal Form.

One member may not revise another member’s program. Correctable errors should be documented on a “Program Certification and Review Form” and sent to the User’s Library Manager for dissemination to the original authors to allow them to submit a revision.

Enhancement or reprogramming of a program already in the User’s Library should be submitted as a new package of documentation and will be treated as an original submission.
DISKETTE AND PAPER TAPE
ORDERING PROCEDURE

DISKETTE ORDERING PROCEDURE

The guidelines below apply to all diskette orders, with the exception of Resident Language Translators (Section 9). Program diskettes, unless otherwise noted on the author's program abstract, are available in source code only.

1. Three program minimum on all diskette orders.
2. A three program diskette is priced at $55.00; add $15.00 for each additional program.
3. Your check or money order is required with orders under $100.00.

Listings will not be included with diskette orders. However, when required, appropriate User's Guide will be sent. Diskette Program Order Forms are enclosed in your Insite™ Manual, Section 3. Send your order to Intel Corporation, User's Library, 3065 Bowers Avenue, Santa Clara, CA 95051.

RESIDENT LANGUAGE TRANSLATORS

Due to program length, programs in Section 9 are priced separately. Insite members can either order the programs individually or as a complete package. Program Diskettes are available in source code unless otherwise noted.

<table>
<thead>
<tr>
<th>Reference Number</th>
<th>Description</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1</td>
<td>Small</td>
<td>$35.00</td>
</tr>
<tr>
<td>F2</td>
<td>ML/80 Structured Assembler for 8080</td>
<td>$70.00</td>
</tr>
<tr>
<td>F3</td>
<td>Basic M</td>
<td>$35.00</td>
</tr>
<tr>
<td>F7</td>
<td>LLL Basic Interpreter</td>
<td></td>
</tr>
<tr>
<td>F8</td>
<td>Octal Debugging Routine</td>
<td>$35.00</td>
</tr>
<tr>
<td>F9</td>
<td>4004/4040 Cross Assembler for Intellec 800</td>
<td>$25.00</td>
</tr>
<tr>
<td></td>
<td>(Paper Tape and Diskette available in object code only)</td>
<td></td>
</tr>
<tr>
<td>F10</td>
<td>8008 Cross Assembler for Intellec 800</td>
<td>$25.00</td>
</tr>
<tr>
<td></td>
<td>(Paper Tape and Diskette available in object code only)</td>
<td></td>
</tr>
<tr>
<td>F11</td>
<td>8080 Macro Assembler for Intellec 8/MOD80</td>
<td>$15.00</td>
</tr>
<tr>
<td></td>
<td>(Source Listing available only)</td>
<td></td>
</tr>
<tr>
<td>F12</td>
<td>8008 Macro Assembler for Intellec 8/MOD8</td>
<td>$15.00</td>
</tr>
<tr>
<td></td>
<td>(Source Listing available only)</td>
<td></td>
</tr>
<tr>
<td>F13</td>
<td>Sequential Pascal Compiler PAS80</td>
<td>$70.00</td>
</tr>
<tr>
<td></td>
<td>(Available on diskette only)</td>
<td></td>
</tr>
<tr>
<td>F14</td>
<td>RIA80</td>
<td>$25.00</td>
</tr>
<tr>
<td>F15</td>
<td>LLL/Chernack Basic Interpreter</td>
<td>$35.00</td>
</tr>
</tbody>
</table>

Package Price for all Resident Language Translators (Section 9) .... $320.00
PAPER TAPE ORDERING PROCEDURE

The majority of Library Programs are available in source code. However, paper tapes for Resident Language Translators, Section 9, are only offered in object code. Program Order Forms are in this section of your Insite Program Library Manual. A prepaid $15.00 handling fee is charged for each program, which includes paper tape and listing. Send your order to Intel Corporation, User's Library, 3065 Bowers Avenue, Santa Clara, California 95051.
PROGRAM SUBMITTAL PROCEDURE

All programs submitted for our review must follow the guidelines listed below:

1. Programs must be written in standard Intel Assembly Language, PL/M or FORTRAN. These languages are documented in the following manuals:
   a. 8080/8085 Assembly Language Programming Manual #9800301C
   b. 8008 Assembly Language Programming Manual #98-019B
   c. 4004/4040 Assembly Language Programming Manual #98-025A
   d. 8008/8080 PL/M Programming Manual #98-108A
   e. PL/M 80 Programming Manual #98-268B
   f. MCS 48/UCI 41 Assembly Language Manual #98-255B
   g. FORTRAN-80 Programming Manual #9800481

2. Submitted programs must be error free. No consideration will be given to partial programs or duplication of existing programs. All accepted programs should assemble or compile correctly without syntax errors. An example of program operation must be included.

3. Programs must be supplied in both machine readable source form and machine generated listing along with a completed submittal form (copies of this form can be found in this section). On the back of the Library Submittal Form are detailed instructions for program submittal which should be closely adhered to. These documentation standards are maintained to assure the usability of each library program by other users.
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program Title</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td></td>
</tr>
<tr>
<td>Required Hardware</td>
<td></td>
</tr>
<tr>
<td>Required Software</td>
<td></td>
</tr>
<tr>
<td>Input Parameters</td>
<td></td>
</tr>
<tr>
<td>Output Results</td>
<td></td>
</tr>
<tr>
<td>Registers Modified:</td>
<td></td>
</tr>
<tr>
<td>Programmer:</td>
<td></td>
</tr>
<tr>
<td>RAM Required:</td>
<td></td>
</tr>
<tr>
<td>Company:</td>
<td></td>
</tr>
<tr>
<td>ROM Required:</td>
<td></td>
</tr>
<tr>
<td>Address:</td>
<td></td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td></td>
</tr>
<tr>
<td>City:</td>
<td></td>
</tr>
<tr>
<td>Assembler/Compiler Used:</td>
<td></td>
</tr>
<tr>
<td>State:</td>
<td></td>
</tr>
</tbody>
</table>
# PROGRAM CERTIFICATION AND REVIEW FORM

Please check all statements made by the submitting author before noting program discrepancies. Any comments relating to program improvement are welcome; however, program revisions or rewrites must be sent in as original submissions.

<table>
<thead>
<tr>
<th>PROGRAM NAME:</th>
<th>REFERENCE NUMBER:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Does the program work?</td>
<td>Yes □ No □</td>
</tr>
<tr>
<td>Were the author's comments accurate?</td>
<td>Yes □ No □</td>
</tr>
<tr>
<td>Was the documentation sufficient?</td>
<td>Yes □ No □</td>
</tr>
<tr>
<td>Was modification necessary?</td>
<td>Yes □ No □</td>
</tr>
<tr>
<td>Were usage instructions adequate?</td>
<td>Yes □ No □</td>
</tr>
<tr>
<td>Should the program be revised?</td>
<td>Yes □ No □</td>
</tr>
</tbody>
</table>

**COMMENTS:** (Please elaborate on deficiencies noted above and include any necessary modifications)

---

© Intel Corporation, 1976
Please check all statements made by the submitting author before noting program discrepancies. Any comments relating to program improvement are welcome; however, program revisions or rewrites must be sent in as original submissions.

<table>
<thead>
<tr>
<th>PROGRAM NAME:</th>
<th>REFERENCE NUMBER:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Does the program work?</td>
<td>Was modification necessary?</td>
</tr>
<tr>
<td>Yes □ No □</td>
<td>Yes □ No □</td>
</tr>
<tr>
<td>Were the author’s comments accurate?</td>
<td>Were usage instructions adequate?</td>
</tr>
<tr>
<td>Yes □ No □</td>
<td>Yes □ No □</td>
</tr>
<tr>
<td>Was the documentation sufficient?</td>
<td>Should the program be revised?</td>
</tr>
<tr>
<td>Yes □ No □</td>
<td>Yes □ No □</td>
</tr>
</tbody>
</table>

**COMMENTS:** (Please elaborate on deficiencies noted above and include any necessary modifications)

© Intel Corporation, 1976
<table>
<thead>
<tr>
<th>REFERENCE NUMBER</th>
<th>PROGRAM</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>AA1</td>
<td>RAM TEST PROGRAM</td>
<td>4-5</td>
</tr>
<tr>
<td>AA2</td>
<td>8080 RAM MEMORY TEST</td>
<td>4-81</td>
</tr>
<tr>
<td>AA3</td>
<td>MEMORY DIAGNOSTIC PROGRAM</td>
<td>4-83</td>
</tr>
<tr>
<td>AA4</td>
<td>PUNCH TEST OR TTY READER/PUNCH TEST</td>
<td>4-98</td>
</tr>
<tr>
<td>AA5</td>
<td>READER TEST</td>
<td>4-103</td>
</tr>
<tr>
<td>AA6</td>
<td>DIAGNOSTIC 1003 - MEMORY VALIDITY CHECK</td>
<td>4-186</td>
</tr>
<tr>
<td>AA7</td>
<td>LIST DEVICE PROGRAM</td>
<td>4-190</td>
</tr>
<tr>
<td>AA8</td>
<td>TRACE - PROGRAM TRACE AND DEBUGGER</td>
<td>4-203</td>
</tr>
<tr>
<td>AA9</td>
<td>MEMORY TEST FOR THE 8080</td>
<td>4-250</td>
</tr>
<tr>
<td>AA10</td>
<td>RAM CHECK</td>
<td>4-269</td>
</tr>
<tr>
<td>AA11</td>
<td>MEMORY TEST PROGRAM</td>
<td>4-271</td>
</tr>
<tr>
<td>AA12</td>
<td>8080 CPU EXERCISE ROUTINE</td>
<td>4-340</td>
</tr>
<tr>
<td>AA13</td>
<td>INTELLEC/MDS DIAGNOSTIC TEST VERSION 1.1</td>
<td>4-360</td>
</tr>
<tr>
<td>AA14</td>
<td>TTY DIAGNOSTIC</td>
<td>4-414</td>
</tr>
<tr>
<td>AA15</td>
<td>SAMPLE AUTOMATIC TEST EQUIPMENT</td>
<td>4-456</td>
</tr>
<tr>
<td>AA16</td>
<td>DISKETTE RECOVERY PROGRAM, RECOVERY 1</td>
<td>4-486</td>
</tr>
<tr>
<td>AA17</td>
<td>SBC 80/10 8255 TEST</td>
<td>4-537</td>
</tr>
<tr>
<td>AA18</td>
<td>USCOPE 820 TEST INSTRUMENT,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ISBC 80/10 DIAGNOSTIC PROGRAM</td>
<td></td>
</tr>
<tr>
<td>AA19</td>
<td>I/O TEST PROGRAM FOR SBC 80/20 - IOTEST</td>
<td></td>
</tr>
<tr>
<td>AA20</td>
<td>INPUT/OUTPUT DIAGNOSTIC 8080</td>
<td></td>
</tr>
<tr>
<td>Page</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>4-295</td>
<td>ICE-80 DISASSEMBLER</td>
<td></td>
</tr>
<tr>
<td>4-297</td>
<td>DELETE COMMENTS</td>
<td></td>
</tr>
<tr>
<td>4-301</td>
<td>TYPE</td>
<td></td>
</tr>
<tr>
<td>4-302</td>
<td>SDK 80 KEYBOARD MONITOR</td>
<td></td>
</tr>
<tr>
<td>4-303</td>
<td>DISK DUMP ROUTINE FOR ICOM F DOS -II/MOD 80 FLOPPY DOS.</td>
<td></td>
</tr>
<tr>
<td>4-307</td>
<td>LIST/PRINT/TYPF &quot;LIST SRC&quot; ON DISKETTE</td>
<td></td>
</tr>
<tr>
<td>4-325</td>
<td>STATEMENT COUNTER</td>
<td></td>
</tr>
<tr>
<td>4-327</td>
<td>9600 INITIALIZE CRT 9600 UART FOR BAUD.</td>
<td></td>
</tr>
<tr>
<td>4-334</td>
<td>RTM (REAL TIME MONITOR)</td>
<td></td>
</tr>
<tr>
<td>4-336</td>
<td>FORMAT INTEL DATA</td>
<td></td>
</tr>
<tr>
<td>4-338</td>
<td>LIST - LIST ERRORS OR SPECIFIED LINES ON CONSOLE</td>
<td></td>
</tr>
<tr>
<td>4-344</td>
<td>BTP (BINARY TAPE PROGRAM)</td>
<td></td>
</tr>
<tr>
<td>4-346</td>
<td>LERR - LIST ASSEMBLY ERRORS</td>
<td></td>
</tr>
<tr>
<td>4-353</td>
<td>ADCCP REMAINDER ROUTINE</td>
<td></td>
</tr>
<tr>
<td>4-356</td>
<td>WIPE - FILE DELETER</td>
<td></td>
</tr>
<tr>
<td>4-358</td>
<td>TABS - EXPANDS FILES TO INCLUDE CONTROL-I</td>
<td></td>
</tr>
<tr>
<td>4-362</td>
<td>INTELCC 8 TEXT EDITOR</td>
<td></td>
</tr>
<tr>
<td>4-364</td>
<td>2708 PROM PROGRAMMER FOR INTELCC 8 MOD80</td>
<td></td>
</tr>
<tr>
<td>4-366</td>
<td>INTELCC 8 MOD60 MONITOR</td>
<td></td>
</tr>
<tr>
<td>4-368</td>
<td>INTELCC MDS MONITOR VERSION 2.0</td>
<td></td>
</tr>
<tr>
<td>4-370</td>
<td>UTILITY MACROS FOR 8000</td>
<td></td>
</tr>
<tr>
<td>4-372</td>
<td>SBC 80/10 PORT I/O EXERCISOR</td>
<td></td>
</tr>
<tr>
<td>4-374</td>
<td>FLY READER DRIVER</td>
<td></td>
</tr>
<tr>
<td>4-378</td>
<td>NON-ENCODED KEYBOARD SUBROUTINE</td>
<td></td>
</tr>
<tr>
<td>4-380</td>
<td>LOAD</td>
<td></td>
</tr>
<tr>
<td>4-386</td>
<td>DRIVER FOR TEKTRONIX 4010 GRAPHIC SCREEN</td>
<td></td>
</tr>
<tr>
<td>4-388</td>
<td>T.I. SILENT 700-SBC 80 MONITOR INTERFACE</td>
<td></td>
</tr>
<tr>
<td>4-390</td>
<td>GLANCE</td>
<td></td>
</tr>
<tr>
<td>4-392</td>
<td>KEYBOARD SCANNER</td>
<td></td>
</tr>
<tr>
<td>4-400</td>
<td>P2708 PROM PROGRAMMING ROUTINE</td>
<td></td>
</tr>
<tr>
<td>4-404</td>
<td>DATA GENERAL TO INTELCC MDS DISKETTE TRANSPORT PACKAGE</td>
<td></td>
</tr>
<tr>
<td>4-406</td>
<td>MAIN ROUTINE DDUMP (DISKETTE DUMP ROUTINES)</td>
<td></td>
</tr>
<tr>
<td>4-412</td>
<td>I/O ROUTINE FOR T.I. SILENT 700 TERMINAL</td>
<td></td>
</tr>
<tr>
<td>4-418</td>
<td>RECOVR - TEXT EDITOR RECOVERY PROGRAM</td>
<td></td>
</tr>
<tr>
<td>4-422</td>
<td>MSAVE/MLOAD</td>
<td></td>
</tr>
<tr>
<td>4-424</td>
<td>DISPLAY</td>
<td></td>
</tr>
<tr>
<td>4-426</td>
<td>HAZELTINE 2000 CRT FUNCTION</td>
<td></td>
</tr>
<tr>
<td>4-432</td>
<td>INTEL FORMAT HEX DATA FILE LOAD/READ</td>
<td></td>
</tr>
<tr>
<td>4-436</td>
<td>&quot;DATCON B1&quot; ANALOG TO DIGITAL CONVERSION PROGRAM</td>
<td></td>
</tr>
<tr>
<td>4-438</td>
<td>FAST AND SLOW</td>
<td></td>
</tr>
<tr>
<td>4-450</td>
<td>5 LEVEL (BARBOT) TO 8 LEVEL (ASCII)</td>
<td></td>
</tr>
<tr>
<td>4-452</td>
<td>PAPER TAPE CONVERSION</td>
<td></td>
</tr>
<tr>
<td>4-458</td>
<td>SBC COMMUNICATOR</td>
<td></td>
</tr>
<tr>
<td>4-460</td>
<td>IBM SELLECTRIC INPUT PROGRAM</td>
<td></td>
</tr>
<tr>
<td>4-462</td>
<td>INSERT TAB CHARACTERS FOR SPACES</td>
<td></td>
</tr>
<tr>
<td>4-464</td>
<td>SDK PROM PROGRAMMER</td>
<td></td>
</tr>
<tr>
<td>4-470</td>
<td>EXEC</td>
<td></td>
</tr>
<tr>
<td>4-474</td>
<td>MDS BACK TO BACK DATA TRANSFER (PL/M)</td>
<td></td>
</tr>
<tr>
<td>4-480</td>
<td>FDUMP</td>
<td></td>
</tr>
<tr>
<td>4-482</td>
<td>ENHANCED MDS TEXT EDITOR X111</td>
<td></td>
</tr>
</tbody>
</table>
PRINT - LIST FILE ON LP UP TO 255 TIMES ........................................... 4-488
TYPE - LIST FILE ON CONSOLE BY PAGE ............................................. 4-490
JOIN - MERGE 2 HEX FILES ...................................................................... 4-492
PRINT PROGRAM FOR G.E. TERMINET-1200 PRINTER. .............................. 4-494
PRINT OUT SOURCE FILE ON FLOPPY DISK ........................................... 4-496
ONLINE, UPLOAD, DOWNLOAD. ................................................................. 4-502
8048 - SEVEN SEGMENT DISPLAY INTERFACE ........................................ 4-520

AP29 "USING THE 8085 SERIAL I/O LINES".............................................. 4-522
8048 TUNE GENERATOR ......................................................................... 4-524
CARD READER DRIVER: HOLLERITH TO ASCII CONVERSION ............... 4-533
COMPARE FILES ..................................................................................... 4-535
CUT AND PASTE EDITOR <PL/M> ............................................................. 4-554
HEWLETT-PACKARD CALCULATOR TO MDS800/ .................................... 4-563
CONTROLLER FOR HEWLETT-PACKARD 9871A PRINTER. ....................... 4-569
PRINT TEXT FOR SBC 80/10. ................................................................. 4-580
MONITOR FOR ISBC80/05 OR 80/04. ....................................................... 4-590
MONITOR FOR ISBC80/10 OR 80/10A ....................................................... 4-592
MONITOR FOR ISBC80/20 OR 80/20-4 ..................................................... 4-594
EXTRACT SELECTED LINES FROM SOURCE OR PRINT FILE - ............... 4-600
BINARY PUNCH TAPE FOR PROM PROGRAMMER - PUNCH ................. 4-608
HEX-FIELD FORMAT TO OBJ. - ............................................................... 4-611
MDT TEXT EDITOR RECOVERY PROGRAM - RETRIEVE ......................... 4-617
EID - CALIBRATION FOR MAPPING WITH THE SURFACE ...................... 4-622
TRANSFORMATION ROUTINE FOR COORDINATE DIGITS ....................... 4-624
FROM EID INTO X-Y COORDINATES OF CRT - ...................................... 4-624
CALCULATE TWO VERIFICATION DIGITS FOR DATA STRING - .......... 4-626
FORMFEED: FORMFEED TO LINEFEED, CONVERSION PROGRAM. ....... 4-642
MONITOR ROUTINES FOR A 3M DCD1 CASSETTE TAPE DRIVE. .............. 4-650
ALPHA: AN ALPHABETIZED LISTING OF THE DISK DIRECTORY. .......... 4-653
BLOCK: LARGE HEX FILE INTO PROM LENGTH .................................... 4-662
PROM BUS MAPPER ............................................................................ 4-669
FLEXIBLE NAME LIST MANAGER .......................................................... 4-671
IDENTI - FRONT PAGE IDENTIFIER PRINTER PROGRAM. ..................... 4-678
MON830 - MONITOR FOR ISBC80/30 .................................................... 4-679
UPI-41 8-DIGIT LED DISPLAY CONTROLLER ....................................... 4-680
UPI-41A SENSOR MATRIX CONTROLLER ............................................. 4-688
LRC PRINTER CONTROLLER - AP27 ....................................................... 4-698
UPI-41A COMBINATION I/O DEVICE <UNI0D> ....................................... 4-700
SDK-86 SERIAL MONITOR, V1.1 .......................................................... 4-702
SDK-86 KEYPAD MONITOR, V1.1 ......................................................... 4-704
TAPE - AUDIO TAPE INTERFACE ......................................................... 4-706
TEXT PROCESSOR. ............................................................................... 4-708
AB146 TEXT EDITOR

AB147 8278 KEYBOARD/DISPLAY CONTROLLER - UPI-41A.

AB148 8295 DOT MATRIX PRINTER CONTROLLER - UPP-41A.

AB149 OLIVETTI 20-COLUMN PRINTER CONTROLLER - UPI-41.

AB150 8292 (GPIB CONTROLLER) IMPLEMENTATION.

AB151 SEND 48 DOWNLOAD TO PROMPT48 FOR SERIES II.

AB152 PROMPT48 OR PROMPT80 INTERFACE - OUTIN.

AB153 REMOTE48 - INTERACTIVE CONTROLLR OF PROMPT48.

AB154 SYMBOL TABLE INSERTER FOR AB22.

AB155 SDK-80 PSEUDO DISASSEMBLER.

AB156 SELECTIVE FILE LINE PRINTER AND SCANNER.

AB157 SDK85 - MONITOR FOR THE 8080 SYSTEM DESIGN KIT.

AB158 RATE - BAUD RATE SELECTION FOR MDS SYSTEM.

AB159 KEYWORD FILE SEARCH FOR ISIS-II ENVIRONMENT.

AB160 TIMER - MEASURES EXECUTION TIMES OF USER PROGRAMS.

AB161 KAPIAR - VERSION 1.2 - GENERAL PURPOSE MACROPROCESSOR.

AB162 ECCO PAPER TAPE READER.

AB163 TRACE.ICE.

AB164 DOWNS0 - DOWNLOAD FROM SERIES-II TO PROMPT80/85.

AB165 IBM BI-SYNC CRCL6 GENERATION SUBROUTINE.

AB166 HXEDIT - HEXADECIMAL DISK FILE EDITOR.

AB167 ALPHANUMERIC INPUT FROM NUMERIC KEYBOARD.

AB168 MODIFIED SDK-80 RESTART ROUTINE.

AB169 MDS SERIES-II: DUMB TERMINAL.

AB170 FILE GENERATOR: FROM OFF-LINE TERMINAL TO
ISIS-II FILE.
AC1 8080 I/O SYSTEM STATUS DISPLAY ........................................ 4-25
AC2 LIST 1 - HIGH-SPEED LIST PROGRAM FOR INTELLEC 8 .................. 4-30
AC3 I/O SIMULATION MACROS .................................................... 4-34
AC4 TALLY R2050 HSPTR DRIVER ............................................... 4-108
AC5 TALLY - USE TALLY 220 LINE PRINTER IN ASSEMBLY STAGE OF PROGRAMMING ........................................ 4-112
AC6 MODEL 101 CENTRONICS PRINTER HANDLER ............................. 4-114
AC7 HIGH-SPEED PAPER TAPE READER WITH STEPPER MOTOR CONTROL .................................................. 4-119
AC8 INTELLEC 8/MD 80 - SILENT 700 INTERFACE ............................ 4-128
AC9 READ/WRITE ROUTINES FOR INTERCHANGE TAPES .................... 4-164
AC10 HANDLER FOR TALLY PTP ................................................... 4-208
AC11 T.I. SILENT 700 INTERFACE - INTELLEC MDS ........................ 4-215
AC13 ASSEMBLER ORIENTED CENTRONICS 306 LINE PRINTER HANDLER AND ERROR ONLY ASSEMBLER ......................... 4-248
AC14 MP8208 A/D CONVERTER ROUTINE .................................... 4-277
AC15 PAGE BREAK FOR TEKTRONIX 4010 MODO GRAPHICS TERMINAL ... 4-280
AC16 C RTBZ/GET - INTERFACE WITH HP2640 CRT TERMINAL ............ 4-285
AC18 INTERFACING THE MDS AND HP 2644A .................................. 4-308
AC19 VIDEO DRIVER ................................................................. 4-309
AC20 BASIC DIGITAL PANEL METER CALL ..................................... 4-313
AC21 APL GRAPHIC DISPLAY ON A 5 x 7 DOT MATRIX .................... 4-317
AC22 BUFFERED LINE PRINTER DRIVER FOR CENTRONIX 101A - $BLPT ........................................ 4-318
<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>AD1</td>
<td>SAVE/RESTORE CPU STATE ON AN INTERRUPT.</td>
<td>4-1</td>
</tr>
<tr>
<td>AD2</td>
<td>READ AND INTERRUPT MODIFICATIONS FOR INTELLEQ 8/MOD80.</td>
<td>4-27</td>
</tr>
<tr>
<td>AD3</td>
<td>INTERRUPT SERVICE ROUTINE</td>
<td>4-130</td>
</tr>
<tr>
<td>AD4</td>
<td>INTERRUPT HANDLER (RE-ENTRANT).</td>
<td>4-132</td>
</tr>
<tr>
<td>AD5</td>
<td>SOFTWARE STACK ROUTINES FOR 8080.</td>
<td>4-176</td>
</tr>
<tr>
<td>AD6</td>
<td>8089 -- BREAK.89.</td>
<td>4-757</td>
</tr>
<tr>
<td>AD7</td>
<td>PROGRAMMABLE SOFTWARE TIMERS.</td>
<td>4-765</td>
</tr>
<tr>
<td>AE1</td>
<td>LIST.</td>
<td>4-319</td>
</tr>
<tr>
<td>AE2</td>
<td>TRACE ROUTINE.</td>
<td>4-320</td>
</tr>
<tr>
<td>AE3</td>
<td>8080 SYMBOL TABLE DUMP.</td>
<td>4-321</td>
</tr>
<tr>
<td>AE4</td>
<td>SNAP DUMP 8080.</td>
<td>4-330</td>
</tr>
<tr>
<td>AE5</td>
<td>TRACE VERSION 7.0.</td>
<td>4-351</td>
</tr>
<tr>
<td>AE6</td>
<td>SCAN.</td>
<td>4-384</td>
</tr>
<tr>
<td>AE8</td>
<td>CROSS REFERENCE FOR PAS80 PASCAL PROGRAMS--;XREF80</td>
<td>4-408</td>
</tr>
<tr>
<td>AE9</td>
<td>SBC 80/10 INTERACTIVE MONITOR</td>
<td>4-410</td>
</tr>
<tr>
<td>AE10</td>
<td>SDK 80 TRAP</td>
<td>4-466</td>
</tr>
<tr>
<td>AE11</td>
<td>TRACE &amp; REGISTER PRINT OUT.</td>
<td>4-468</td>
</tr>
<tr>
<td>AE12</td>
<td>STEP.</td>
<td>4-506</td>
</tr>
<tr>
<td>AE13</td>
<td>PROGRAM TEST LOAD.</td>
<td>4-513</td>
</tr>
<tr>
<td>AE14</td>
<td>SYMBOL TABLE PROGRAM FOR 8080/8085 V1.2</td>
<td>4-515</td>
</tr>
<tr>
<td>AE15</td>
<td>WRITEP - OUTPUT PROCEDURES FOR PL/M-80.</td>
<td>4-635</td>
</tr>
<tr>
<td>AE16</td>
<td>ERRORP/MESSGP - ERROR MESSAGE POINT SUBROUTINE (ISIS-II AND USER).</td>
<td>4-639</td>
</tr>
<tr>
<td>AE17</td>
<td>CERROR - PLM-80 COMPILER ERROR DISPLAY PROGRAM.</td>
<td>4-673</td>
</tr>
<tr>
<td>AE18</td>
<td>DDKDUMP - ISIS DISK FILE DUMP.</td>
<td>4-733</td>
</tr>
<tr>
<td>AE19</td>
<td>DDUMP - DISKETTE DDUMP.</td>
<td>4-735</td>
</tr>
</tbody>
</table>
SAVE/RESTORE CPU STATES ON AN INTERRUPT

Saves the CPU registers and flags in memory at the start of interrupt processing and restores the CPU registers and flags after the interrupt has been processed.

A LIFO register (hardware external push down stack) to allow saving registers and flags without using L and H.

In the present program, SVAD (saveguard address) is the output address of that LIFO register and RSAD (restitution address) is the input.

None

None

None

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Assembler/Compiler Used:</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>MAC8</td>
</tr>
<tr>
<td>RAM Required:</td>
<td>Programmer:</td>
</tr>
<tr>
<td>6</td>
<td>Robert Arouete</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>Company:</td>
</tr>
<tr>
<td>60 bytes</td>
<td>A 2 M</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td>Address: 78 170 La Celle</td>
</tr>
<tr>
<td>0</td>
<td>St. Cloud, FRANCE</td>
</tr>
</tbody>
</table>
ORG 100H

0100  D310  OUT SVAD
0102  78   MOV A,B
0103  D310  OUT SVAD
0105  79   MOV A,C
0106  D310  OUT SVAD
0108  7A   MOV A,D
0109  D310  OUT SVAD
010B  7B   MOV A,E
010C  D310  OUT SVAD
010E  7C   MOV A,H
010F  D310  OUT SVAD
0111  7D   MOV A.L
0112  D310  OUT SVAD

MVI A,0
MOV B,A
JZ S1I
MVI A,0
JP S2I
MVI B,64
JPE S1I
INR B
JPO S1I
INR B
RAR
ORA B
OUT SVAD
RET

SVAD  EQU 16

XRA A
IN RSAD
RAL
MOV B,A
INR B
DCR B
IN RSAD
MOV L,A
0118 DB04           IN RSAD
0120 67            MOV H, A
0121 DB04           IN RSAD
0122 DF            MOV E, A
0123 DB04           IN RSAD
0124 67            MOV D, A
0125 DB04           IN RSAD
0126 4F            MOV C, A
0127 DB04           IN RSAD
0128 47            MOV B, A
0129 DB04           IN RSAD
012A C9            RET
012B RSAD         EQU 4
012D END


RAM TEST PROGRAM

Performs write and read of all zeros and ones, checkerboard test and unique address test. The RAM to be tested is successively initialized to a value and then tested. The values are zero, all ones, alternate bit word (Ø252) and its complement (Ø125). Next, increasing consecutive values are stored and then tested.

Teletypewriter

Requires teletypewriter I/O subroutines. (Other subroutines included.)

After an "A" is printed, the operator types the initial and final RAM addresses separated by a comma and ended with a carriage return. All numbers are in octal.

For each error found, the address, contents and the value which should have been read are printed with the heading. No errors are signified by printing "A" for the next test.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Maximum Subroutine Nesting Level:</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>4 or 5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RAM Required:</th>
<th>Assembler/Compiler Used:</th>
</tr>
</thead>
<tbody>
<tr>
<td>20 bytes</td>
<td>MACB</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ROM Required:</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>429 bytes</td>
<td>S Kung/D Wallace/L Breiden</td>
</tr>
<tr>
<td></td>
<td>Varian Instrument Division</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Company:</th>
</tr>
</thead>
<tbody>
<tr>
<td>2700 Mitchell Drive</td>
</tr>
<tr>
<td>Walnut Creek, Calif.</td>
</tr>
</tbody>
</table>

4-5
TTY Binary Load Routine

This program will load memory via TTY which are formatted into blocks of 256 or less binary bytes and tests each block against a checksum frame for any read errors. The tape format requires a rubout to indicate start of block followed by the starting page address, the starting byte address, the word count, up to 256 bytes of data and a checksum in that order. A block of data may overlap pages but may not exceed 256 bytes in length. The last block of data should be followed by two consecutive rubouts to indicate end of data. The program will then branch to page 000 byte 000.

REQUIRED HARDWARE: ASR-33 teletype, teletype interface to 8008

Intel MCS-8 hardware assembler page 000 or TTY input and output routines

None

Program will be loaded into allocated memory locations. A checksum error will cause a program halt. Interrupting with a No-Op will restart the program.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Assembler/Compiler Used:</th>
</tr>
</thead>
<tbody>
<tr>
<td>A, B, C, D, E, H, L</td>
<td>MAC8</td>
</tr>
<tr>
<td>RAM Required:</td>
<td>Programmer:</td>
</tr>
<tr>
<td>73 bytes</td>
<td>David Crenlen</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>Company:</td>
</tr>
<tr>
<td>MCS-8 hardware assembler,</td>
<td>Continental Design</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td>pp. 000</td>
</tr>
<tr>
<td></td>
<td>Address:</td>
</tr>
<tr>
<td></td>
<td>7723 Convoy Court</td>
</tr>
<tr>
<td></td>
<td>San Diego, Ca. 92111</td>
</tr>
</tbody>
</table>

98-034C 4-7
DESCRIPTION OF TEST DATA, DEFINITION OF TERMS, ADDITIONAL INFORMATION FOR THE TTY BINARY LOADER AND TTY BINARY DUMP ROUTINES.

It is recommended that the TTY Binary Dump routine be assembled first. A known section of memory, preferably ROM should be dumped using the Binary dump routine. By manually checking the first few bytes with an octal dump of the same area the user can determine whether or not he's on the right track.

The complete program checkout procedure would be as follows:

Load a page of octal data using the octal loader.
Dump the page in binary.
Load the page back in using the binary loader.
Dump the page using the octal dump program.

The tape produced by the octal loader in the final step should compare exactly with the tape loaded in initially.

SOME TERMS DEFINED; AND CONVENTIONS USED:

The Checksum produced by the dump program and checked by the loader is a modulo-$256_{10}$ sum of all data in the block. Assuming a valid punch operation, a faulty checksum indicates a read error.

The byte count is the number of bytes to be loaded into memory. At the expiration of the byte count, the checksum is examined.

The dump produces, and the loader requires, an eight bit rubout preceding the binary address of the data. Thus, the format is:

```
xxxxxx.xxx  rubout
xxxxxx.xxx  starting address, page
xxxxxx.xxx  starting address, byte
xxxxxx.xxx  byte count
.
.
.
'byte count' frames
.
.
.
xxxxxx.xxx  checksum

xxxxxx.xxx  rubout
xxxxxx.xxx  rubout ...stop code
```
;REF. NO AB1
;PROGRAM TITLE TTY BINARY LOAD ROUTINE

;TTY BINARY LOAD ROUTINE
BEGIN:

0000 CD3500 CALL READ; READ A FRAME
0003 FEFF CPI 3770; START OF BLOCK?
0005 C20000 JNZ BEGIN; NO.
0008 CD3500 CALL READ
000B FEFF CPI 3770; 2 RUBOUTS = STOP CODE
000D CA4700 JZ DONE
0010 67 MOV H, A;
0011 CD3500 CALL READ
0014 6F MOV L, A;
0015 CD3500 CALL READ
0018 5F MOV E, A;
0019 1600 MVI D, 000;
001B CD3500 CALL READ
LOAD:
001E 77 MOV M, A;
001F 82 ADD D;
0020 57 MOV D, A;
0021 2C INR L;
0022 C22600 JNZ SKIP; NO PAGE LAG
0025 24 INR H
SKIP:
0026 1D DCR E;
0027 C21800 JNZ LOAD; LOOP UNTIL BYTE COUNT = 0
002A CD3500 CALL READ;
002D 92 SUB D;
002E CA0000 JZ BEGIN; READ NEXT BLOCK
STOP:
0031 76 HLT; 000
READ:
0035 45 MOV B, L;
0036 4C MOV C, H
0037 214800 LXI H, SAVER
003A 70 MOV M, B
003B 2C INR L
003C 71 MOV M, C
003D 2C INR L
003E 72 MOV M, D

ERROR STOP. PROGRAM CAN BE RESTA WITH A "NOP" INTERRUPT.

SAVE REGISTERS
003F AF   XRA   A;   ACC=0; FLAGS SET
0040 CF   RST   TYINP; DATA INPUT FROM TTY
0041 56   MOV   D, M; NOT SEND TO PRINTER
0042 2D   DCR   L; RESTORE REGISTERS
0043 4E   MOV   C, M
0044 2D   DCR   L
0045 6E   MOV   L, M
0046 61   MOV   H, C
0047 C9   DONE: RET
0048     SAVER: DS 3
0001     TYINP EQU 1
0000     END
TTY BINARY DUMP ROUTINE

This program will punch the contents of memory via TTY which are formatted into blocks of 256 or less binary bytes with checksum. The tape format begins with a rubout to indicate start of block followed by the starting page address, the starting byte address, the word count, up to 256 bytes of data and a checksum in that order. The start and end memory locations are entered from the TTY keyboard.

ASR-33 teletype, teletype interface to 8008

Intel MCS-8 Hardware assembler page 000 or TTY input and output routines and TTY octal to binary converting routine (BILDE).

On TTY keyboard enter D followed by colon then ending page (PPP), ending byte (BBB), starting page (ppp), starting byte (bbb).

D: PPP: BBB: ppp: bbb:

Program will punch binary tape from designated memory locations and calculate and punch a checksum. Tape feed frames are punched between blocks and a stop code (two consecutive rubouts) are punched at end of dump. The program will end by branching to page 000 byte 000.

Registers Modified: A, B, C, D, E, H, L

RAM Required: 

ROM Required:
MCS-8 Hardware assembler pg. 000

Maximum Subroutine Nesting Level: 3

Assembler/Compiler Used: MAC8

Programmer: David Crelten

Company: Continental Design

Address: 7723 Convoy St.

City: San Diego

State: CA 92111
TTY BINARY DUMP ROUTINE

BEGIN:

0000 AF XRA A
0001 FF RST TTYIN;
0002 F680 ORI 2000Q;
0004 FEC4 CPI 304Q;
0006 C20000 JNZ BEGIN;
0009 AF XRA A
000A FF RST TTYIN
000B F680 ORI 2000Q
000D FEBA CPI 2720Q;
000F C20000 JNZ BEGIN;
0012 218E00 LXI H, END;
0015 CD9F00 CALL BILDE;

318 73 MOV M.E;
0019 2C INR L;
001A CD9F00 CALL BILDE;
001D 73 MOV M.E;
001E CD9F00 CALL BILDE;
0021 63 MOV H.E;
0022 CD9F00 CALL BILDE;
0025 6B MOV L.E;

PADR:

0026 CD6A00 CALL FEED;
0029 CD7600 CALL BYTEC;
002C 7C MOV A.H;

002D F7 RST TTYOT;
002E 7D MOV A.L;
002F F7 RST TTYOT;
0030 AF XRA A;
0031 93 SUB E
0032 F7 RST TTYOT;
0033 1E00 MVI E, 000;

PTEXT:

0035 7E MOV A, M;
336 83 ADD E;
0037 5F MOV E, A;
0038 7E MOV A, M;

; REF. NO. AB2
; PROGRAM TITLE TTY BINARY DUMP
; 
; TTY BINARY DUMP ROUTINE

; INPUT TTY-SEND TO PRINTER
; ADD PARITY
; D?
; NO.
; COLON?
; NO.
; POINT TO ENDING VALUE
; ACCEPT 3-DIGIT OCTAL FROM TTY
; CONVERT TO BINARY REPRESENTATION
; STORE ENDING PAGE
; POINT TO BYTE END VALUE
; ACCEPT OCTAL FROM TTY
; STORE ENDING BYTE
; ACCEPT OCTAL FROM TTY
; STORE START PAGE
; ACCEPT OCTAL FROM TTY
; STORE START BYTE
; GENERATE LEADER FOR PUNCH
; CALCULATE FIRST-BLOCK BYTE COUNT
; MOVE PAGE ADDRESS TO ACC FOR BINARY PUNCH. NOTE THAT THE
; REQUIRED LEADING RUBOUT HAS BEEN PUNCHED BY THE FEED ROUTINE PRE-
; CEEDING EACH BLOCK
; PUNCH STARTING ADDRESS
; MOVE BYTE ADDRESS TO ACC
; PUNCH STARTING ADDRESS -- BYTE
; COMPUTE BYTE COUNT
; BYTE COUNT IN ACC--PUNCH ONTO TA
; INITIALIZE CHECKSUM

FETCH DATUM FROM MEMORY
COMPUTE CHECKSUM
UPDATE CHECKSUM
CHECKSUM CALCULATION DESTROYED D IN ACC. IS FETCHED FROM MEMORY A
PUNCH:

0039 F7      RST  TTYOT;         PUNCH DATUM
003A 7C      MOV  A, H;          SAVE PAGE VALUE
003B 4D      MOV  C, L;          SAVE BYTE VALUE
003C 218E00  LXI  H, ENDR;       POINT TO ENDING VALUE
003F BE      CMP  M, P;         LAST PAGE?
0040 C24E00  JNZ  P2;           NO.
0043 47      MOV  B, A          MOVE BYTE ADDRESS TO ACC FOR COM
0044 79      MOV  A, C;          POINT TO LAST BYTE
0045 2C      INR  L;            LAST BYTE?
0046 BE      CMP  M;            YES--GO TO END OF JOB
0047 78      MOV  A, B
0048 CA5C00  JZ   PEND;         NEXT BYTE

P2:
004B 0C      INR  C;            RESTORE H & L REGISTERS
004C C25700  JNZ  P3            RESTORE H & L REGISTERS
004F 69      MOV  L, C;         RESTORE H & L REGISTERS
0050 67      MOV  H, A;
0051 24      INR  H
0052 7B      MOV  A, E
0053 F7      RST  TTYOT;       PUNCH CHECKSUM
    254 C32600  JMP  PADR;       GO PUNCH NEXT BLOCK

P3:
0057 69      MOV  L, C;
0058 67      MOV  H, A;
0059 C33500  JMP  PTEXT        PUNCH CHECKSUM

PEND:
005C 7B      MOV  A, E;
005D F7      RST  TTYOT         GENERATE TRAILER AND 1 OF
005E CD6A00  CALL  FEED;       2 NEEDED RUBOUTS FOR STOP CODE

      0061 3EFF  MVI  A, 3770     PUNCH RUBOUT
5       0063 F7      RST  TTYOT;     MORE TRAILER
0064 CD6A00  CALL  FEED;       JMP  DONE
0067 C30000  JMP  DONE

;THE VALUE OF DONE IS EQUATED TO
;0000 000 IN THIS ASSEMBLY. THIS ADDRESS
;IS THE DESIRED BRANCH AFTER THE DUMP.
;IS COMPLETED. A VALUE OF 000 000 RETURNS
;CONTROL TO THE TTY OPERATING SYSTEM SO
;FURTHER COMMANDS MAY BE MADE. IF THE
;DUMP PROGRAM IS TO BE MADE A CALLED
;SUBROUTINE, A "RET" INSTRUCTION WOULD
;BE APPROPRIATE. THE PROGRAM COULD ALSO
;BE MADE TO HALT HERE, IF DESIRED.

FEED:

36A 1E28  MVI  E, 040;       INTER-RECORD GAP OF 32 FRAMES
F:
006C AF      XRA  A;          INSURE TAPEFEED IS PUNCHED
006D F7      RST  TTYOT;      PUNCH TAPEFEED
I.S 8080 MACRO ASSEMBLER, V1.0

006E 1D DCR E;
006F C26C00 JNZ F;
0072 3EFF MVI A, 3770;
0074 F7 RST TTYOT;
0075 C9 RET ;

BYTEC:
0076 44 MOV B, H;
0077 4D MOV C, L;
0078 7C MOV A, H;
0079 21800 LXI H, ENDER;
007C BE CMP M;
007D C2800 JNZ NEQ;
0080 2C INR L;
0081 7E MOV A, M;
0082 91 SUB C;
0083 C3800 JMP SKIP

NEQ:
0086 2C INR L;
0087 7E MOV A, M;

SKIP:
0088 EEFF XRI 3770;
008A 5F MOV E, A;
008B 60 MOV H, B;
008C 69 MOV L, C;
008D C9 RET

ENDER:
008E DS 2
0000 DONE EQU 0;

0006 TTYOT EQU 6;
0007 TTYIN EQU 7;

009F BILDE EQU 0002370;

0000 END

REDUCE "FRAMES REMAINING" COUNT LOOP UNTIL DONE
LOAD LEADING RUBOUT FOR BLOCK START OF BLOCK RE-ENTER PROGRAM
SAVE H SAVE L GET INITIAL H TO ACC POINT TO FINAL H INITIAL PAGE = FINAL PAGE? NO.
POINT TO FINAL L GET FINAL L ACC=FINAL - INITIAL
POINT TO FINAL L GET FINAL L

1's COMPLEMENT STORE BYTE COUNT IN E
STORED ADDRESS IS STORED HERE
THIS IS THE BRANCH ADDRESS ON PR COMPLETION
TTY OUTPUT ROUTINE TTY INPU ROUTINE -- ECHOES BACK TO PRINTER. EQUATE TO 000010 IF DATA ENTRY IS PREFERRED
ACCEPTS 3-DIGIT OCTAL TE COLON AND CONVERTS TO BINARY WITH RESULT IN "E"
MEMORY DUMP
Lists memory in octal: start & stop point user definable

TTY -- standard Intel setup

TTY routines
Carriage return & linefeed routines

None

Lists memory on teletype

Registers Modified:
All

RAM Required:
Last 7 words of 013
1 RAM or ROM for pgm storage

ROM Required:

Maximum Subroutine Nesting Level:

Assembler/Compiler Used:
MAC8

Programmer:
LeRoy J. Kniskern

Company:
Poly-Scientific Div.
Litton Systems, Inc.

Address:
1213 N. Main St.
Blacksburg, Va. 24060
;REF. NO. AB3
;PROGRAM TITLE MEMORY DUMP
;
;
;
;
;
;
;
;
;
;
;
;
;
;
;
;
;
;

DUMPS SELECTED MEMORY IN OCTAL
;
USER INSTRUCTIONS
CALL THIS SUBROUTINE
TTY WILL TYPE F
TYPE FIRST ADDRESS TO BE DUMPED
IN FORMAT HHH:LLL:
EXAMPLE 010:356:
ADDRESS IN OCTAL, 3 DIGITS IMMEDIATELY
PRECEEDING ARE INTERPRETED AS ADDRESS
TTY WILL TYPE L
TYPE LAST ADDRESS TO BE DUMPED IN SAME FORMAT
ADDRESSES NEED NOT BE ON SAME PAGE

EXTERNAL SUBROUTINES REQUIRED:
TTYIN-INTEL TELETYPE INPUT ROUTINE
TOUT - INTEL TELETYPE OUTPUT ROUTINE
CRLF - CARRIAGE RET & LINE FEED ROUTINE

REGISTERS USED - ALL
RAM USED - LOCATIONS 371 THRU 377 IN 013

OUTPUT FORMAT - ADDRESS LISTED EVERY 8 LOCATIONS
CONTENTS OF M IN OCTAL

BEGIN:

0000 260B   MVI       H, 0130
0002 CD2800   CALL      CRLF;  CARRIAGE RETURN & LINE FEED
0005 0646   MVI       B, 'F'
0007 CD8500   CALL      TOUT
000A CD2800   CALL      CRLF
000D CDA600   CALL      TAKE3;  GET FIRST 'H'
0010 2EFC   MVI       L, 3740
0012 73   MOV       M, E
0013 CDA600   CALL      TAKE3;  LET FIRST 'L'
0016 2EFF   MVI       L, 3730
318 73   MOV       M, E
J019 CD2800   CALL      CRLF
001C 064C   MVI       B, 'L'
001E CD8500   CALL      TOUT

4-19
0021 CD2800 CALL CRLF
0024 CDA600 CALL TAKE3; GET LAST ‘H’
0027 2EFA MVI L, 3720
0029 73 MOV M, E
002A CDA600 CALL TAKE3; GET LAST ‘L’
002D 2EF9 MVI L, 3710
002F 73 MOV M, E
0030 2C INR L
0031 2C INR L
0032 46 MOV B, M
0033 2C INR L
0034 4E MOV C, M
0035 68 MOV L, B
0036 61 MOV H, C

PRT1:
0037 CD5800 CALL PRADR

PRT:
003A 5E MOV E, M
003B CD8200 CALL PROC
003E CD6E00 CALL ADCK
0041 CA5400 JZ DONE
344 2C INR L
0045 C24900 JNZ PRT2
0048 24 INR H

PRT2:
0049 7D MOV A, L
004A E607 ANI 007
004C F000 CPI 000
004E CC5800 CZ PRADR
0051 C33A00 JMP PRT1

DONE:
0054 CD2800 CALL CRLF
0057 C9 RET ; END OF DUMP (COULD BE JUMP INSTR

PRADR:
0058 CD2800 CALL CRLF
005B 5C MOV E, H
005C CD8200 CALL PROC
005F 5D MOV E, L
0060 CD8200 CALL PROC
0063 062D MVI B, ‘ ’
0065 CD8500 CALL TOUT
0068 063B MVI B, ‘ ’ ; SPACE
006A CD8500 CALL TOUT
006D C9 RET

ADCK:
006E 45 MOV B, L
36F 4C MOV C, H
J079 21F916 LXI H, 0133710
0073 7E MOV A, M
0074 B8 CMP B
0075 C27F00   JNZ NOTIT
0078 2C       INR L
0079 7E       MOV A, M
007A B9       CMP C
007B C27F00   JNZ NOTIT
007E C9       RET

NOTIT:
007F 68       MOV L, B
0080 61       MOV H, C
0081 C9       RET

PROCT:
0082 7B       MOV A, E
0083 E6C0     ANI 3000
0085 07       RLC
0086 07       RLC
0087 CD9F00   CALL OCT1
008A 7B       MOV A, E
008B E638     ANI 0700
008D 0F       RRC
008E 0F       RRC
008F 0F       RRC
390 CD9F00   CALL OCT1
0093 7B       MOV A, E
0094 E607     ANI 007
0096 CD9F00   CALL OCT1
0099 063B     MVI B, 057; SPACE
009B CD8500   CALL TOUT
009E C9       RET

OCT1:
009F C630     ADI '0'; 060 OCTAL
00A1 47       MOV E, A
00A2 CD8500   CALL TOUT
00A5 C9       RET

TAKE3:
00A6 CD3900   CALL TTYIN
00A8 78       MOV A, B
00AA FE3A     CPI ?
00AC CABB00   JZ IN3
00AF 2EFF     MVI L, 3770
00B1 56       MOV D, M
00B2 70       MOV M, B
00B3 2D       DCR L
00B4 5E       MOV E, M
00B5 72       MOV M, D
00B6 2D       DCR L
00B7 73       MOV M, E
3B8 C3A600    JMP TAKE3

IN3:
00BB CDCA00   CALL CONOCT
00BE D0       RNC ; NO ERROR, BINARY NOW IN -E-
0000 063F    MVIB, 'z'
0001 CD8500 CALL TOUT
0004 CD2000 CALL CRLF
0007 C3A600 JMP TAKE3

CONOC1:

000A 2EFF MVIL, 2770
000C CDF2000 CALL CON1
0010 D8 RC
0012 5F MOV E, A
0014 2D DCR L
0016 CDF2000 CALL CON1
0018 D8 RC
001A 07 RLC
001C 07 RLC
001E 83 ADD E
0020 5F MOV E, A
0022 2D DCR L
0024 CDF2000 CALL CON1
0026 D8 RC
0028 47 MOV B, A
002A D604 SUI 004
002C D2EE00 JNC NOTOK
002E 78 MOV A, B
0030 0F RRC
0032 0F RRC
0034 83 ADD E
0036 5F MOV E, A
0038 AF XRA A
003A 1F RAR
003C C9 RET

NOTOK:

003E 3E01 MVIA, 001
0040 1F RAR
0042 C9 RET

CON1:

0044 7E MOV A, M
0046 C6C8 ADI 3180
0048 D8 RC
004A D6F8 SUI 3700
004C 83 RAR

004E C9 RET

0050 CRLF EQU 500; CHANGE AS REQUIRED
0055 TOUT EQU 2050; CHANGE AS REQUIRED
0059 TTYIN EQU 710; CHANGE AS REQUIRED
005C END
PROM PROGRAMMER FOR INTELLEC 8

Changes programmer from fixed timing to PROM dependent timing. Programs 50% more than minimum required, ensuring permanency.

Intellec 8 with PROM programming board and system monitor PROMs.

Works with Intellec 8 system monitor version 1.0 and 1.1.

Interfaces with user exactly as original version.

Noticeably faster programming with no requirement for two (2) passes to ensure permanency. Locations that cannot be programmed result in return to system with a printout of the location.

<table>
<thead>
<tr>
<th>Registers Modified: Register D used in addition to others used in original version.</th>
<th>Maximum Subroutine Nesting Level:</th>
</tr>
</thead>
<tbody>
<tr>
<td>None additional</td>
<td>N/A</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RAM Required:</th>
</tr>
</thead>
<tbody>
<tr>
<td>None additional</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ROM Required:</th>
</tr>
</thead>
<tbody>
<tr>
<td>None additional</td>
</tr>
</tbody>
</table>

Assembler/Compiler Used: Intellec 8/Mod 8 Macro Assembler, Ver. 1.0

Programmer: William Haskett
Northeast Electronics

Company: Box 649
Concord, N.H. 03301
; REF. NO. AB4
; PROGRAM PROM PROGRAMMER FOR INTELLEC 8

3CCB INCAD SET 3CCB
3BDE START SET 3BDEH
3EF2 HEXLAP SET 3EF2H ; MONITOR SUBROUTINES
3C75 LONG SET 3C75H
3C72 SHORT SET 3C72H
3B9A ORG 3B9AH
3B9A 1608 NEW MVI D, 08H : SET MINIMUM OVERCHARGE
3BC9 79 MOV A, C : SET ADDRESS TO PROGRAM
3B9D EEFF XRI 0FFH : COMPLIMENT IT
3B9F D30A OUT 10 : SEND TO PROM
3B91 DB02 OLD IN 2 : READ DATA IN PROM
3B93 EEFF XRI 0FFH
3B95 BE CMP M : COMP WITH DESIRED DATA
3BA6 CABA3B JZ DOWN : IF OK, GOTO OVERCHARGE
3B98 14 INR D : INCREMENT PASS COUNT
3BAA E70 MVI A, 70H : GET MAXIMUM PASSES
3BAC E8 CMP D : COMPARE WITH COUNT
3BAD CADA3B JZ ERROR : CAN'T PROGRAM, GIVE UP
3BB0 14 INR D : ADD TWO TO PASS COUNT
3B81 14 INR D : SUB ONE FROM COUNT
3B92 15 DOWN DCR D : FINISHED IF ZERO
3B93 CACC3B JZ DONE : GET DATA TO PROGRAM
3B96 EEFF MVI A, 0FFH : SEND IT TO PROM
3B98 AE XRA M : ENABLE PROG PULSES
3B99 D30B OUT 11 : WAIT THREE PULSES
3B9B E87 MVI A, 0B7H
3B9D D309 OUT 9 : STOP PULSES
3B9F CD753C CALL LONG
3B92 E37 MVI A, 37H
3B94 D309 OUT 9 : COOL OFF
3B96 CD723C CALL SHORT : DO IT AGAIN
3B98 C3A13B JMP OLD
3B9A 0C DONE INR C : STEP NEXT PROM ADD
3B9C 0A3B JZ START : QUIT IF OVERFLOW
3B9D 1C CALL INCAD : STEP MEMORY ADDRESS
3B9E 0A3C DCR E : DEC COUNT OF-pos TO PROG
3B99 0A3E JNZ NEW : START PROG NEW LOCATION
3B9D 03E3B JMP START : RETURN TO SYSTEM
3BDA 79 ERROR MOV A, C : GET ADD WHERE FAILED
3BDE 00 CALL HEXAD : PRINT IT
3B00 00 NOP : ENDS AT START OF MONITOR
8080 I/O SYSTEM STATUS DISPLAY

Display current I/O assignment information when invoked by the INTELLEC8/MOD 80 MONITOR (Version 1.0)

INTELLEC 8/MOD 80, 1 1702A PROM

INTELLEC 8/MOD 80 MONITOR, Version 1.0

Calls IOCHK and uses the returned value for driving the display

SYSTEM STATUS

CONSOLE  TTY
READER    PTR
PUNCH     PTP
LIST      TTY

Registers Modified:
N/A

Maximum Subroutine Nesting Level:
3 bytes

RAM Required:
NONE

Assembler/Compiler Used:
8080 Resident Macro Assembler

ROM Required:
1 1702A

Programmer:
K. Burgett

Company:
**Read and Interrupt Modifications to the 8080 System Monitor**

These modifications allow printing of headings or operator instructions at the beginning of a Read operation (from the front of the Hex Load tape). They also allow console control of the interrupt enable F-F. This prevents interrupts until the service subroutines are loaded.

**Required Hardware**
- Intellec 8-80 (or equal)
- PROM Programmer

**Required Software**
- 8080 Monitor, Version 2.0

**Input Parameters**

Monitor Commands:
- "R"  Read Hex Tape and print heading (all characters before the first ":") on the console.
- "IE"  Enable the 8080 interrupt F-F.
- "ID"  Disable the 8080 interrupt F-F.

**Output Results**

**Registers Modified:**
- Not Important

**RAM Required:**
- None

**ROM Required:**
- 48H (or 1702A)

**Maximum Subroutine Nesting Level:**
- 0 (Not important)

**Assembler/Compiler Used:**
- 8080 Macro Assembler

**Programmer:**
- C. Vincent Phillips

**Company:**
- Alkon Corporation

**Address:**
- 5329 N. High St.

**City:**
- Columbus

**State:**
- Ohio 43214

© Intel Corporation, 1976
; REF. NO. AD2
; PROGRAM TITLE READ AND INTERRUPT MODIFICATIONS

; LINKAGES

3C20 LER EQU 3C20H ; ERROR ROUTINE
3D59 Expr EQU 3D59H ; GET EXPRESSION
3CAD CRLF EQU 3CADH ; CARRIAGE RETURN, LINE FEED
3EEE RIX EQU 3EEEH ; READ, CHECK AND MASK PARI
3F6D TI EQU 3F6DH ; INPUT FROM CONSOLE AND E
386A START EQU 386AH ; MAIN CONTROL LOOP
3C32 CO EQU 3C32H ; CONSOLE OUTPUT

; THIS CHIP CONTAINS MONITOR COMMAND OVERFLOW
; THE UNUSED BRANCHES IN THE COMMAND BRANCH
; TABLE HAVE BEEN MOVED TO THIS CHIP TO EASE
; FUTURE MODIFICATIONS.
; THE FOLLOWING MODIFICATIONS MUST BE MADE TO
; CHIP 3800 (COMMAND BRANCH TABLE)

LOCATI ON OLD NEW
38A3 2 203C 0036
38A5 203C 0336
38A7 203C 0636
38AF 203C 0936
38B3 203C 0C36
38B5 203C 0F36
38BB 203C 1236
38BD 203C 1536

3600 ORG 3600H ; CHIP 3600H

; NEW COMMAND BRANCH TABLE

3600 C33636 JMP INTER ; I ENTRY POINT
3603 C3203C JMP LER ; J ENTRY POINT
3606 C3203C JMP LER ; K ENTRY POINT
3609 C3203C JMP LER ; O ENTRY POINT
360C C3203C JMP LER ; Q ENTRY POINT
360F C31836 JMP READ ; R ENTRY POINT
3612 C3203C JMP LER ; U ENTRY POINT
3615 C3203C JMP LER ; V ENTRY POINT

; READ ROUTINE
; THIS SUBROUTINE CAUSES A HEADING TO BE
; PRINTED ON THE CONSOLE FROM THE INPUT TAPE
; BEFORE THE BALANCE OF THE TAPE IS READ IN
; THE NORMAL MANNER. ALL CHARACTERS BEFORE
; THE FIRST : EXCEPT NULLS ARE PRINTED. THIS
; IS USEFUL WHEN LOADING FROM MAG TAPE OR A
; DISK.
; READ:
3618 0D DCR C ; GET ONE ADDRESS
3619 CD593D CALL EXPR
361C CDAD3C CALL CRLF ; SPACE UP
361F CDAD3C CALL CRLF

RA:
3622 E1 POP H ; GET BIAS ADDRESS
3623 E5 PUSH H
3624 CDEE3E CALL RIX ; READ CHARACTER
3627 CA2236 JZ RA ; SKIP IF NULL
362A 4F MOV C,A
362B CD323C CALL CO ; TYPE ON CONSOLE
362E D63A SU1 ; ; LOOKFOR RECORD MARK
3630 C22236 JNZ RA
3633 C3EE3A JMP 3AEEH ; JUMP TO READ ROUTINE

; INTERRUPT CONTROL ROUTINE
; IE ENABLE INTERRUPT
; ID DISABLE INTERRUPT
; THIS SUBROUTINE ALLOWS THE CONTROL OF
; THE INTERRUPT ENABLE FROM THE CONSOLE.
; THIS IS USEFUL TO PREVENT INTERRUPTS
; UNTIL THE INTERRUPT SERVICE SUBROUTINES
; ARE LOADED.
; THE FOLLOWING MODIFICATIONS MUST BE MADE
; TO THE MONITOR:

; LOCATION OLD NEW
; 386A FB 00
; 3F9F FB 00

; INTER:
3636 CD6D3F CALL TI ; INPUT CHARACTER
3639 FE45 CPI 'E' ; COMPARE WITH "E"
363B C24236 JNZ I1 ; JUMP IF NOT "E"
363E FB EI ; ENABLE INTERRUPTS
363F C36A38 JMP START ; RETURN

I1:
3642 FE44 CPI 'D' ; COMPARE WITH "D"
3644 C2203C JNZ LER ; ERROR IF NOT "D"
3647 F3 DI ; DISABLE INTERRUPTS
3648 C36A38 JMP START ; RETURN

0000 END
LIST 1

HIGH SPEED LIST DEVICE FOR INTELLEC 8: REQUIRES NO ADDITIONAL OUTPUT BOARD BECAUSE THE PROM PROGRAMMER SOCKET IS USED FOR THE PRINTER CONNECTION.

INTELLEC 8, ONE PROM CHIP, A HIGH SPEED PRINTER SUCH AS CENTRONICS 306, TELETEYPEDWRITER

INTELLEC MONITOR VER. 1.0

COMMAND "AL=1" IN MONITOR WILL CAUSE EXTERNAL PRINTER (CONNECTED VIA PROM PROGRAMMER) TO BE USED FOR "LIST" INFORMATION; (E.G. ASSEMBLER PRINTOUT IN PASS 2 OR 4).

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Maximum Subroutine Nesting Level:</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCUMULATOR</td>
<td>0</td>
</tr>
<tr>
<td>RAM Required:</td>
<td>Assembler/Compiler Used:</td>
</tr>
<tr>
<td>NONE OTHER THAN MONITOR'S STACK</td>
<td>8080 ver 2.0</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>Programmer:</td>
</tr>
<tr>
<td>35 BYTES</td>
<td>Doug Raymond</td>
</tr>
<tr>
<td>INSTALL AS CHIP #7, PROM BOARD 3</td>
<td>Zehntel</td>
</tr>
<tr>
<td></td>
<td>Company:</td>
</tr>
<tr>
<td></td>
<td>2440 Stanwell Drive</td>
</tr>
<tr>
<td></td>
<td>Concord, Ca. 94520</td>
</tr>
</tbody>
</table>
; REF. NO. AC2
; PROGRAM TITLE HIGH SPEED LIST FOR INTELLEC 8
;
***AN EASILY CONNECTED LIST DEVICE***
;
INTELLEC 8 WITH 8080 CPU
;
DOUG RAYMOND, ZEHNTL, INC.
;
THE FOLLOWING IS A SUBROUTINE WHICH
PERMITS THE USE OF AN EXTERNAL LIST
DEVICE (I.E. HIGH SPEED PRINTER) WITH
THE INTELLEC 8 MONITOR VERSION 1.0
;
PRINTER CONNECTION IS MADE VIA THE
PROM PROGRAMMING SOCKET ON THE
INTELLEC'S FRONT PANEL.
;
USE OF A HIGH SPEED PRINTER SPEEDS
ASSEMBLY TREMENDOUSLY, BECAUSE
ONLY TWO PASSES (P=1, P=4) ARE REQUIRED
2) HIGH SPEED READER IS NOT SLOWED
DOWN BY THE SLOW PRINTING ON THE CONSOLE.
;
PRIOR TO RUNNING THE ASSEMBLER, TYPE
"AL=1" INTO THE MONITOR, AND THIS
SUBROUTINE WILL BE USED TO SEND
"LIST" INFORMATION TO THE EXTERNAL
HIGH SPEED PRINTER.
;
THE ROUTINE IS SPECIFICALLY FOR A
CENTRONICS 306, WHOSE CHARACTERISTICS
ARE: DATA--------PARALLEL 7 LINE ASCII
STROBE-------NORMALLY HIGH
BUSY---------HIGH WHEN BUSY
THE CENTRONICS 306 HAS SEVERAL
JUMPER OPTIONS. TWO OF THESE ARE
IMPORTANT FOR PROPER APPEARANCE
OF THE LISTED MATERIAL.
AUTOMATIC LINEFEED SHOULD BE DISABLED.
OPTION DSC SHOULD BE ENABLED.
;
DATA 0 OUT ON PIN 3
1 2
2 1
; 3  21
; 4  20
; 5  19

; 6  18
;STROBE OUT ON PIN 17
;BUSY IN ON PIN 11
;GROUND BY A CLIP LEAD TO SLOT IN CHASSIS
;BOTTOM.
;A 24 PIN WIREWRAP SOCKET CAN BE USED
;FOR A CONNECTOR.

3712
3712 C31A37
371A

ORG 3712H; ENTRY POINT FOR AL=1
JMP PRINT
ORG 371AH; OR OTHER AVAILABLE PROM
PRINT:  MVI A, 0
OUT 1  ; ENABLES SENSING OF BUSY
PBZ:   IN 2
RLC
JNC PBZ ; CARRY IS TRUE IF BUSY WAS TRUE

MVI A, 80H
OUT 1  ; RESTORES CONTROLS TO NORMAL
MOV A, C ; CHARACTER IS IN C REG
ORI 80H ; BIT 7 IS USED AS STROBE
CPI 0FFH

FEFF R2 ; DON'T SEND RUBOUT TO CENTRONICS 306
C8  CMA ; I/O BOARD IS INVERTING
2F  OUT 2
D302 ORI 80H
F680 OUT 2 ; STROBE IS NOW ACTIVE
D302 ANI 7FH
D302 OUT 2 ; STROBE IS RETIRED
C9  RET
0000 END
I/O SIMULATION MACROS

WHEN ASSEMBLED INTO A PROGRAM, THESE TWO MACROS, 'INPUT' AND 'OUTPUT', ALLOW SIMULATION OF ALL 'IN' AND 'OUT' 8080 INSTRUCTIONS. ALL 'IN'S' AND 'OUT'S' ARE LOGGED ON THE INTELLEC 8/MOD80 CONSOLE

INTELLEC 8/MOD80 AND TTY

MOD80 MONITOR

FOR OUT XX: OUTPUT XX = YY
FOR IN XX: INPUT XX = (EXPECT INPUT VALUE)

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Maximum Subroutine Nesting Level:</th>
</tr>
</thead>
<tbody>
<tr>
<td>A, FLAGS</td>
<td>~ 8 BYTES</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RAM Required:</th>
<th>Assembler/Compiler Used:</th>
</tr>
</thead>
<tbody>
<tr>
<td>185 BYTES</td>
<td>8080 MACRO ASSEMBLER</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ROM Required:</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>K. BURGETT</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Company:</th>
</tr>
</thead>
</table>
Tape Duplicator

Duplicates a tape read in from the H.S. tape reader by punching a copy on the TTY terminal with a leader added at both ends.

TTY
H.S. Reader
Intellec 8

Program Object Tape

Load the program with the system monitor. Use the system monitor command(s) to change the memory at address Ø2 to Ø1H. Place the tape to be copied in the H.S. reader. Turn on the tape punch and press reset on the console.

A copy of the tape being read into the H.S. reader will be generated by the TTY punch with a leader at both the beginning and end of the tape. The program automatically returns to the system monitor.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Maximum Subroutine Nesting Level:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Assembler/Compiler Used:</td>
</tr>
<tr>
<td></td>
<td>Intellec 8 Macro Assembler</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RAM Required:</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Jon Zoller</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ROM Required:</th>
<th>Company:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Dorsett Electronics</td>
</tr>
</tbody>
</table>
; REF. NO. AB5
; PROGRAM TITLE TAPE DUPLICATOR

; INPUT FROM H. S. READER-OUTPUT TO TTY
LED  MACRO  BLNK
   MVI E, BLNK
CLEAR: MVI B, 00
       MOV C, B
       CALL PO
       DCR E
       JNZ CLEAR
ENDM

3806   RI   EQU 3806H
380C   PO   EQU 380CH
0100   ORG 100H
0100 310020  LXI SP, 2000H
103 CD0638 START: CALL RI  ; GET A CHARACTER
0106 57   MOV D, A
0107 A7   ANA A    ; IS CHARACTER BLANK?
0108 CA0301 JZ START  ; YES, IGNORE
          +   LED 150
010B 1E96 +   MVI E, 00096H
010D 0600 +CLEAR: MVI B, 00
010F 48 +   MOV C, B
0110 CD0C38 +   CALL PO
0113 1D +   DCR E
0114 C20D01 +   JNZ CLEAR

0117 4A   MOV C, D
0118 1E0F AGAIN: MVI E, 15
011A CD0C38 GO: CALL PO  ; PUNCH CHARACTER
011E CD0638 CALL RI  ; GET ANOTHER CHARACTER
0120 4F   MOV C, A
0121 A7   ANA A
0122 1D   DCR E
0123 C23501 JNZ JOHN
           +   LED 100
0126 1E64 +   MVI E, 00064H
0128 0600 +CLEAR: MVI B, 00
012A 48 +   MOV C, B
012D CD0C38 +   CALL PO
012E 1D +   DCR E
012F C22801 +   JNZ CLEAR
0132 C20038 JMP 3800H
0135 A7   JOHN: ANA A
0136 CA1A01 JZ GO
0139 C31801 JMP AGAIN
0000 END
Program Title: CBC GEN

Function: Generate a 16 bit cyclic redundancy check (CRC) for a data string of up to 2^16 bytes. The generator polynomial and initial conditions are defined by the user.

Required Hardware: None

Required Software: None

Input Parameters:
- BC contains size of data string in bytes
- HL points to beginning of data string
- DE contains initial value of CRC, usually all zeros or all ones
- RAM required - 2 bytes plus data string
- RAM or ROM - 60 bytes (subroutine only)

Output Results:
- All registers are modified
- DE contains value of CRC
- The data string is unmodified

Registers Modified:

<table>
<thead>
<tr>
<th>RAM Required:</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 bytes plus character string</td>
</tr>
<tr>
<td>ROM Required:</td>
</tr>
<tr>
<td>60 bytes</td>
</tr>
</tbody>
</table>

Assembler/Compiler Used:

| MAC80 |

Programmer:

| Geoffrey Karlin |

Company:

| Applied Digital Data Systems, Inc. |

Address:

| 11787 |
| 100 Marcus Blvd., Hauppauge, N.Y. |
;REF. NO.: A66
;PROGRAM TITLE: CRC GEN-GENERATE A 16 BIT CYCLIC REDUNDANCY CHECK

;CALLING PARAMETERS
;
; (BC) - NO. OF BYTES
; HL - BEGINNING OF FILE
; DE - INITIAL CONDITION OF CRC REGISTER

;RETURNING PARAMETERS
;
; (DE) - CRC

0100 ORG 100H

0100 0000 POINT: DW 0000
DATA: DB 0FEH
0103 00 DATA: DB 00
0104 00 DB 00 ; IF CRC REGISTER IS INIT.
105 05 DB 05H ; TO ALL ONES, CRC IS 1E07
0106 00 DB 00

0021 GX1 EQU 21H ; FOR THIS EXAMPLE
0010 GX2 EQU 10H ; G(X)=X(16)+X(12)+X(5)+1

; CALLING PROGRAM FOR SUBROUTINE VERIFICATION
;
0107 310001 CALLP: LXI SP,0100H ; SET UP PARAMETERS TO ALL
010A 010500 LXI B,0005H ; CRCGEN
010D 210201 LXI H,DATA
0110 11FFFF LXI D,0FFFFH
0113 CD1D01 CALL CRCGEN
0118 EB XCHG
011C 220001 SHLD POINT ; TEST OVER, STORE CRC IN POINT
011A C30038 JMP 3000H ; RETURN TO 8080 MONITOR

; ACTUAL SUBROUTINE STARTS HERE
;
011D 220001 CRCGEN: SHLD POINT ; POINT IS LOC. OF NEXT P(X)
0120 C5 PUSH B
0121 E1 POP H ; HL - BYTE CTR
0122 0E08 MVI C,8 ; C - BIT CTR

0124 E5 CRC2: PUSH H ; LOAD (B) @ POINT
0125 2A0001 LHL0D POINT ; (LOAD B NEXT P(X))
0128 46 MOV B,M
0129 23        INX  H
012A 220001     SHLD  POINT
012D E1        POP  H  ;SHIFT CRC LEFT (DE)
                
012E 7B        MOV  A,E  ;SHIFT P (X) LEFT (B)
012F 83        ADD  E
0130 5F        MOV  E,A
0131 7A        MOV  A,D
0132 8A        ADC  D
0133 57        MOV  D,A
0134 DA5401     JC  CRC3  ;JUMP IF MSB = 1
                
0137 78        MOV  A,B  ;JUMP IF MSB = 0
0138 80        ADD  B
0139 47        MOV  B,A
013A D24501    JNC  CRCOUT  ;JUMP IF MSB = 0
                
013D 7B        CRCXOR:MOV  A,E  ;XOR CRC & G(X)
013E EE21      XRI  GX1
0140 5F        MOV  E,A
0141 7A        MOV  A,D
142 EE10       XRI  GX2
0144 57        MOV  D,A
0145 0D        CRCOUT:DCR  C  ;CK BIT CNT
                
0146 C2E01      JNZ  CRC4  ;CK WORD CNT
0149 0E08      MVI  C,8
014B 2D        DCR  L
014C C2401     JNZ  CRC2
014F 25        DCR  H
0150 F2401     JP  CRC2
0153 C9        RET
                
0154 78        CRC3:MOV  A,B  ;SHIFT LEFT P(X)
0155 80        ADD  B
0156 47        MOV  B,A
0157 DA4501     JC  CRCOUT  ;JUMP IF MSB = 1
015A C33001    JMP  CRCXOR  ;JUMP TO XOR, MSB = 0
                
0000          END
16 bit CRC for polynomial X16+X12+X5+1 (polynomial for SDLC)

Function
Produces a 16 bit CRC with 8 bit input bytes. Care should be taken with Most/Least bit feeding of the data byte and CRC residue. Does not require a table or contain any loops. Requires 24H memory locations and executes in 72.5 usec.

Required Hardware
None

Required Software
Main program to include routine as in-line code or modified routine to be called as subroutine.

Input Parameters
Register "A" is data byte
Register "B" is low byte of CRC residue
Register "C" is high byte of CRC residue

Output Results
CRC residue in registers B & C

Registers Modified:
A, B, C, D, E, L

Assembler/Compiler Used:
8080 Macro Assembler, Ver 1.0

RAM Required:
24H bytes, OR

Programmer:
Bill McGonigal

ROM Required:
24H bytes

Company:
Sycor

Maximum Subroutine Nesting Level:

Address: 100 Phoenix Drive
Ann Arbor, Michigan 48104
; REF NO. AB7
; PROGRAM 16 BIT CRC FOR POLYNOMIAL X16+X12+X5+1 (POLYNOMIAL FOR
; ;
; ;
; CRC SUBROUTINE
; POLYNOMIAL IS X16 +X12 +X5 +1
; CRC USES MOST SIGNIFICANT BIT (BACKWARDS)
; REG B IS LOW BYTE OF CRC RESIDUE
; REG C IS HIGH BYTE OF CRC RESIDUE
; DATA BYTE IS EXPECTED IN THE
; ACCUMULATOR
;

0100     ORG $100H
0100 A9   ENTRY: XRA C            ; X-OR DATA BYTE
           MOV D, A            ; WITH CRC HIGH BYTE
1002 0F   RRC
1003 0F   RRC
1004 0F   RRC
1005 0F   RRC
1006 5F   MOV E, A
1007 AA   XRA D
1008 E6F0 ANI $0F0H
100A A8   XRA B
100B 6F   MOV L, A            ; RESTORE (TEMP SAVE)
100C 7B   MOV A, E
100D 07   RLC
100E E61F ANI $1FH            ; MASK
1010 AD   XRA L
1011 6F   MOV L, A            ; RESTORE
1012 7A   MOV A, D
1013 07   RLC
1014 E601 ANI $1            ; MASK
1016 A8   XRA B
1017 AD   XRA L
1018 47   MOV B, A            ; LOW BYTE IS COMPLETE
1019 7B   MOV A, E
101A E60F ANI $0FH            ; MASK
101C AA   XRA D
101D 4F   MOV C, A            ; STORE
101E 7B   MOV A, E
11F AA   XRA D
120 07   RLC
121 E6E0 ANI $0E0H            ; MASK
123 A9   XRA C
0124 4F  MOV  C, A
0125 76  HLT

0000  END

; (NOT PART OF
; SUBROUTINE)
CRC16

This macro calculates a CRC16 check word using the generation polynomial $X^{16}+X^{15}+X^2+1$. It can be used to generate a check word for a record that doesn't include one or to check that a record including a check word is correct.

8080

MAC80 for assembly

"BUFF" is the address to a word that contains the address to the first byte in the record to be included in the CRC16 accumulation.

"LEN" is the address to a word that contains the number of bytes to be included in the CRC16 accumulation not including the two check bytes. The record for which the CRC16 is to be calculated must occupy consecutive bytes in memory. If a CRC16 is to be generated for a record that doesn't contain one, the two bytes following the record where the CRC16 usually lies must be zero.

The CRC16 check word is left in the BC register pair, B being the most significant byte. The HL register pair contains the address to the least significant check byte in the record. The CPU zero flag is set if and only if the BC register pair = 0. This occurs if the record contains a correct CRC16 check word. The A, D, and E registers are left undefined. The record address vector and record length are not changed.
; REF. NO. A88.
; PROGRAM NAME CRC16

; TITLE CRC16

; FUNCTION
; THIS MACRO CALCULATES A CRC16 CHECK WORD
; THE GENERATION POLYNOMIAL X**16+X**15+X**2+1. IT
; CAN BE USED TO GENERATE A CHECK WORD FOR A RECORD THAT
; DOESN'T INCLUDE ONE OR TO CHECK THAT A RECORD INCLUDES
; A CHECK WORD IS CORRECT

; REQUIRED HARDWARE 8080
; REQUIRED SOFTWARE -- MAC80 FOR ASSEMBLY

; INPUT PARAMETERS
; THE MACRO CALL INCLUDES 2 PARAMETERS

BUFF    "BUFF" IS THE ADDRESS TO A WORD THAT CONTAINS THE ADDRESS TO THE FIRST BYTE IN THE RECORD TO BE INCLUDED IN THE CRC16 ACCUMULATION.

LEN     "LEN" IS THE ADDRESS TO A WORD THAT CONTAINS THE NUMBER OF BYTES TO BE INCLUDED IN THE CRC16 ACCUMULATION NOT INCLUDING THE TWO CHECK BYTES. THE RECORD FOR WHICH CRC16 IS TO BE CALCULATED MUST BE CONSECUTIVE BYTES IN MEMORY. IF A CRC16 IS TO BE GENERATED FOR A RECORD THAT DOESN'T CONTAIN ONE, THE BYTES FOLLOWING THE RECORD WHERE THE CRC16 USUALLY MUST BE ZERO.

; OUTPUT RESULTS

; DESCRIPTION

4-50
THE MACRO USES A SUBROUTINE XCRCS TO GENERATE THE CRC16 CHECK WORD. THE CODE FOR THIS SUBROUTINE IS GENERATED BY THE MACRO AT THE FIRST CALL TO THE MACRO ONLY. XCRCS USES 4 BYTES OF RAM MEMORY IN THE STACK FOR TEMPORARY DATA STORAGE AND IS THEREFORE REENTRANT.

TIMING 71600 STATES PER 100 BYTES

RAM REQUIRED 6 BYTES IN STACK

RCM REQUIRED FIRST CALL 64 BYTES

OTHER CALLS 10 BYTES

1000

ORG 1000H ; USED FOR TEST PURPOSES ONLY

CRC16

-----

FFFF XCRCF SET NOT 0

CRC16 MACRO BUFF, LEN

LHLD LEN

XCHG

LHLD BUFF

IF XCRCF

; GENERATE CODE IF FIRST CALL TO

XCRCF SET 0

; XCRCS:

MOV B, M

; FIRST BYTE OF RECORD TO 8

INX H

MOV C, M

; SECOND BYTE TO C

MOV A, M

; PROCESS NEXT BYTE

INX H

; ADDRESS TO NEXT BYTE IN HL

MOV A, M

; NEW BYTE TO A

PUSH H

; STORE CURRENT ADDRESS IN STACK

PUSH D

LXI H, 0A001H

; GENERATION POLYNOMIAL TO HL

MVI E, 8

; INITIALIZE BIT COUNT IN E

NBIT:

RAR

MOV D, A

; SAVE NEW BYTE IN D

MOV A, C

RAR

MOV C, A

; AND INTO C

MOV A, B

; SHIFTFIT MOST SIGNIFICANT BIT IN

RAR

MOV B, A

JNC NODIV

; HERE IF SDIVISABLE

XRA L

;
MOV  B, A
MOV  A, C
XRA  H
MOV  C, A

NODIV:
MOV  A, D
DCR  E
JNZ  NBIT

HERE IF BYTE DONE
POP  D
POP  H
ADDRESS TO LAST BYTE PROCESSED
DCR  E
DECREASE LENGTH COUNT BY 1
JNZ  NBYTE
TAKE NEXT BYTE IF NOT DONE
DCR  D

HERE IF RECORD DONE
JP   NBYTE

MOV  A, B
ANA  A
RNZ
ADD  C
RET

BYPAS:
ENDIF
CALL  XCRC6
ENDM
END

0000
CRECH

Computes CRC characters for IBM compatible floppy disk. Also works for Synchronous Data Link Control (SDLC).

Required Hardware

None

Required Software

None

Input Parameters

HL = start address of data
DE = end address of data

Output Results

BC = CRC characters

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Assembler/Compiler Used:</th>
</tr>
</thead>
<tbody>
<tr>
<td>A,B,C,H,L</td>
<td>8080 Assembler</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RAM Required:</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>50.10</td>
<td>Dick Springer</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ROM Required:</th>
<th>Company:</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>General Microwave Corporation</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Maximum Subroutine Nesting Level:</th>
<th>Address:</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>155 Marine Street</td>
</tr>
<tr>
<td></td>
<td>Farmingdale, N.Y. 11735</td>
</tr>
</tbody>
</table>
:REF NO. AB3
:PROGRAM CRECH
:
:
:
:
:
:
:
:
:
:
:

SUBROUTINE CRECH COMPUTES THE CRO CHECK WORDS REQUIRED FOR
READING FROM OR WRITING ON IBM-FORMAT FLOPPY DISKS. THE TWO-
BYTE CHECK WORD FOR A BLOCK OF DATA IS THE WORD PRODUCED BY
THE GENERATING POLYNOMIAL X**16+X**12+X**5+1 WHEN THE CHECK
WORD IS INITIALIZED TO THE VALUE 0FFFFH.

THE SUBROUTINE INITIALIZES THE CHECK WORD TO 0FFFFH AND
PROCESSES A BLOCK OF DATA WITH INITIAL MEMORY LOCATION IN
REGISTER PAIR (H.L) AND FINAL MEMORY LOCATION IN REGISTER
PAIR (D.E). THE CHECK WORD APPEARS IN REGISTER PAIR (B.C)
WITH THE FIRST BYTE ON THE DISK IN REGISTER C AND THE SECOND
BYTE IN REGISTER B.

IF A DIFFERENT CHECK WORD INITIALIZATION IS DESIRED, AS MAY
BE THE CASE IF PROCESSING IS TO BE CONTINUED AFTER PROCESSING
A BLOCK ELSEWHERE IN MEMORY, THE INITIAL VALUE SHOULD BE
INSERTED IN (B.C) AND THE ROUTINE ENTERED AT LOCATION CRECH+3.

AFTER PROCESSING, REGISTER PAIR (B.C) CONTAINS THE CHECK WORD,
REGISTER PAIR (D.E) CONTAIN THE INPUT FINAL ADDRESS, AND
REGISTER PAIR (H.L) CONTAIN THE ADDRESS FOLLOWING THAT OF
THE LAST BYTE PROCESSED.

IF THE FINAL ADDRESS PRECEDES THE INITIAL ADDRESS ONE BYTE
IS PROCESSED.

0000 01FFFF CRECH:  LXI B,0FFFFH
0003 D5        PUSH D
0004 7E        MOV A,M
0005 A9        XRA C
0006 57        MOV D,A
0007 0F        RRC
0008 0F        RRC
0009 0F        RRC
000A 0F        RRC
000B E60F       ANI 0FH
000C AA        XRA D
000D 5F        MOV E,A
000E 0F        RRC
000F 0F        RRC
0010 0F        RRC
0011 57        MOV D,A
0012 E61F       ANI 1FH
0015 A8        XRA B
0016 4F
0017 7A
0018 E6E0
001A AB
001B 47
001C 7A
001D 0F
001E E6F0
0020 A9
0021 4F
0022 23
0023 D1
0024 7A
0025 BC
0026 D8
0027 C20300
002A 7B
002B 8D
002C D8
002D C30300
0000

MOV C, A
MOV A, D
ANI 0E0H
XRA E
MOV B, A
MOV A, D
RRC
ANI 0F0H
XRA C
MOV C, A
INX H
POP D
MOV A, D
CMP H
RC
JNZ CRECH+3
MOV A, E
CMP L
RC
JMP CRECH+3
END
Legible paper tape

The program punches legible characters on paper tape, useful for tape labelling. The characters are formed in the conventional 5x7 matrix and the 64 character sub-set of ASCII is represented. The characters to be punched are taken from the console device and decoded as an index into a table of bit patterns. Console input is convenient where a separate punch is available but is not a necessary part of the logic: a list of characters could as easily be referenced. Carriage return, line feed and the control characters are echoed but not punched.

Hardware: console and paper tape punch

Console I/O and punch output routines of Intellec 3 monitor.

ASCII characters in A

Legible paper tape: bit patterns presented to punch output routine in B

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Assembler/Compiler Used:</th>
</tr>
</thead>
<tbody>
<tr>
<td>A, B, C, D, E, H, L</td>
<td>ISIS 8080 Macro Assembler, V1.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RAM Required:</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>387 decimal</td>
<td>Intellec 3 Macro assembler</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ROM Required:</th>
<th>Company:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P.R. Cave - DR</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Maximum Subroutine Nesting Level:</th>
<th>Address:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Dept. Mechanical Engrg.</td>
</tr>
<tr>
<td></td>
<td>The University, Dundee</td>
</tr>
<tr>
<td></td>
<td>DD1 4EH England</td>
</tr>
</tbody>
</table>
Banner Print & Punch

Create, on a listing device or tape perforator, a graphic representation of certain ASCII characters.

Required Hardware

CBAN Console Keyboard, List Device
LBAN List Device
CPBN Console Keyboard, Tape Perforator
PBAN Tape Perforator

Required Software

None

Input Parameters

CBAN Console Keyboard Input
CPBN

Output Results

LBAN The B and C registers contain the high and low bytes respectively of the address of a buffer. The buffer contains the ASCII codes for the characters to be printed or punched, and is terminated by the ASCII code for carriage return (ODH). Care should be taken that the graphic representations of the specified characters will fit on one line of the list device. Legal characters include the alphabet, the digits and special characters: Space, comma, *, +, -, *, /, :, ;, <, =, >, ?, and @.

PBAN

CBAN Graphic representations of the input characters drawn on the output from the listing device. Each character is drawn in a 1, 3, or 5 by 7 matrix with 2 columns between characters.

CPBN

PBAN Graphic representations of the input characters drawn on perforated tape. Each character is drawn in a 1, 3, or 5 by 7 matrix with 2 blanks between characters.

Registers Modified:

All

Assembler/Compiler Used:

8008 Macro Assembler Ver. 2D

RAM Required:

16 bytes

Programmer:

C. E. Ohme

ROM Required:

657 bytes

Company:

Ohme

Maximum Subroutine Nesting Level:

3

Address: 144750 Winding Lane
Fremont, CA 94538
; REF. NO. AB11
; PROGRAM TITLE BANNER PRINT & PUNCH
;
1000 ORG 1000H
;
BANNER ROUTINES
;
MACROS TO SIMULATE 8080 INSTRUCTIONS
; (VALID ONLY IN THESE ROUTINE)
0000 CR EQU 0DH ; CARRIAGE RETURN
0001 LF EQU 0AH ; LINE FEED
;
THE CBAN ROUTINE ACCEPTS CHARACTERS
; FROM THE CONSOLE AND DRAWS THEM ON
; THE LIST DEVICE. A CARRIAGE RETURN
; TERMINATES EACH LINE. PLEASE STAY
; ON THE PAPER.

1000 CD5E11 CBAN: CALL CRLF ; PRINT CR LF
1003 1640 MVI D, 40H ; MASK FOR LINE 1
1005 21C812 LXI H, BUFF ; BUFFER ADDR
1008 22C512 SHLD BADR ; INIT BADR
100B 7C MOV A, H
100C 45 MOV B, L
100D 21C512 LXI H, BADR
1010 70 MOV M, B
1011 23 INX H
1012 CDC112 CALL INCHL
1015 77 MOV M, A
1016 67 MOV H, A
1017 68 MOV L, B
1018 22C712 CB1: SHLD CADR ; SAVE CAHR ADDR
101B 7C MOV A, H
101C 45 MOV B, L
101D 21C712 LXI H, CADR
1020 70 MOV M, B
1021 23 INX H
1022 CDC112 CALL INCHL
1025 77 MOV M, A
1026 67 MOV H, A
1027 68 MOV L, B
1028 CDB512 CALL CI ;INPUT A CHAR
102B E67F ANI 7FH ;REMOVE PARITY
102D 5F MOV E,A ;SAVE CHAR
102E 2AC712 LHLH CADR ;RESTORE CHAR ADDR

1031 21C712 LXI H,CADR
1034 7E MOV A,M
1035 23 INX H
1036 CDC112 CALL INCHL

1039 66 MOV H,M
103A 6F MOV L,A
103B 7B MOV A,E ;CHAR TO DRAW
103C 77 MOV M,A ;STORE CHAR

103D FE0D CPI CR ;CAP RET
103F CA6210 JZ EOP ;IF END OF INPUT
1042 CDEC10 CALL CONV ;CONVERT CHAR
1045 DA1810 JC CB1 ;IF ILLEGAL
348 CD1E11 CALL BCH ;PRINT LINE 1
104B 2AC712 LHLH CADR ;RESTORE CHAR ADDR
104E 21C712 LXI H,CADR
1051 7E MOV A,M
1052 23 INX H
1053 CDC112 CALL INCHL

1056 66 MOV H,M
1057 CDC112 CALL INCHL
105A 6F MOV L,A
105B 23 INX H ;FOR NEXT CHAR
105C CDC112 CALL INCHL
105F C31810 JMP CB1 ;LOOP
1062 CDB210 EOP: CALL EOL ;COMPLETE PICTURE
1065 CD5E11 CALL CRLF ;PRINT CR LF
1068 CD5E11 CALL CRLF ;PRINT CR LF
106B C30010 JMP CBAN ;FOR NEXT PICTURE

; THE LBAN SUBROUTINE ACCEPTS THE ADDRESS OF A 
; BUFFER IN THE B AND C REGISTERS. ASCI 
; CHARACTERS IN THE BUFFER ARE DRAWN ON THE 
; LIST DEVICE UNTIL AN ASCII CARRIAGE RETURN 
; IS ENCOUNTERED.

106E 60 LBAN: MOV H,B ;BUFFER ADDR H
106F 69 MOV L,C ;BUFFER ADDR L
1070 22C512 SHLD BADR ;SAVE BUFFER ADDR
1073 7C  MOV  A, H
1074 45  MOV  B, L
1075 21C512 LXI  H, BADR
1076 70  MOV  M, B
1077 23  INX  H
1078 CDC112 CALL  INCHL
107D 77  MOV  M, A
107E 67  MOV  H, A
107F 68  MOV  L, B
1080 CD5E11 CALL  CRLF  ; PRINT CR LF
1083 1640 MVI  D, 40H  ; MASK FOR LINE 1
1085 2AC512 LB1: LHLDBADR  ; BUFFER ADDR
1088 21C512 LXI  H, BADR
108B 7E  MOV  A, M
108C 23  INX  H
108D CDC112 CALL  INCHL
1090 66  MOV  H, M
1091 6F  MOV  L, A
1092 7E  LB2: MOV  A, M  ; CHAR TO DRAW
1093 FE0D CPI  CR  ; CAR RET
1095 CAB210 JZ  EOL  ; IF END OF LINE
1098 CDEC10 CALL  CONV  ; CONVERT CHAR
109B D41E11 CNC  BCH  ; IF LEGAL CHAR
109E 2AC712 LHLDCADR  ; RESTORE CHAR ADDR
10A1 21C712 LXI  H, CADR
10A4 7E  MOV  A, M
10A5 23  INX  H
10A6 CDC112 CALL  INCHL
10A9 66  MOV  H, M
10AA 6F  MOV  L, A
10AB 23  INX  H  ; FOR NEXT CHAR
10AC CDC112 CALL  INCHL
10AF C39210 JMP  LB2  ; LOOP
10B2 CD5E11 EOL: CALL  CRLF  ; PRINT CRLF
10B5 7A  MOV  A, D  ; MASK
10B6 1F  RAR  ; FOR NEXT LINE
10B7 E67F ANI  7FH  ; REMOVE TOP BIT
10B9 C8  RZ  ; IF PICTURE DONE
10BA 57  MOV  D, A  ; MASK
10BB C38510 JMP  LB1  ; LOOP

; THE CPBN ROUTINE ACCEPTS CHARACTERS
; FROM THE CONSOLE AND DRAWS THEM ON
; THE PUNCH DEVICE.
10BE CDB512 CPBN: CALL CI ; INPUT A CHAR
10C1 E67F ANI 7FH ; REMOVE PARITY
10C3 CDEC10 CALL CONV ; CONVERT CHAR
10C6 D46811 CNC PCH ; IF LEGAL CHAR
10C9 C3BE10 JMP CPBN ; LOOP

; THE PBAN SUBROUTINE ACCEPTS THE ADDRESS OF A
; BUFFER IN THE B AND C REGISTERS. ASCII
; CHARACTERS IN THE BUFFER ARE DRAW ON THE
; PUNCH DEVICE UNTIL AN ASCII CARRIAGE RETURN
; IS ENCOUNTERED.
10CC 60 PBAN: MOV H, B ; BUFFER ADDR H
10CD 69 MOV L, C ; BUFFER ADDR L
10CE 7E PB1: MOV A, M ; CHAR TO DRAW
10CF FE0D CPI CR ; CAR RET
10D1 C8 RZ ; IF END OF BUFFER
10D2 CDEC10 CALL CONV ; CONVERT CHAR
10D5 D46811 CNC PCH ; IF LEGAL CHAR
10DB 2AC712 LHLD CADR ; RESTORE CHAR ADDR
10DB 21C712 LXI H, CADR
10DE 7E MOV A, M
10DF 23 INX H
10E0 CDC112 CALL INCHL
10E3 66 MOV H, M
10E4 6F MOV L, A
10E5 23 INX H ; FOR NEXT CHAR
10E6 CDC112 CALL INCHL
10E9 C3CE10 JMP PB1 ; LOOP

; CONV CONVERTS THE CHARACTER IN A TO A CRAFT
; TABLE ORDINAL IN H AND L. THE CHARACTER IS
; SAVED IN E. THE ORIGINAL H AND L ARE STORED
; AT CADR. D IS PRESERVED.
10EC 5F CONV: MOV E, A ; SAVE CHAR
10ED 22C712 SHLD CADR ; SAVE CHAR ADDR
10F0 7C MOV A, H
10F1 45 MOV B, L
10F2 21C712 LXI H, CADR
10F5 70 MOV M, B
10F6 23 INX H
10F7 CDC112 CALL INCHL
10FA 77 MOV M, A
I. 8080 MACRO ASSEMBLER, V1.1

10FB 67 MOV H,A
10FC 68 MOV L,B

10FD 78 MOV A,E ;CHAR TO CONVERT
10FE D620 SUI ' ' ;SPACE
1100 D8 RC ;IF < SPACE
1101 CA1211 JZ FND ;IF SPACE
1104 D60A SUI '*' ;* = 0
1106 D8 RC ;IF *
1107 FE31 CPI 'Z'+1+*/' ;CHECK FOR > Z
1109 C21B11 JNZ CNVER ;IF > Z
110C C601 ADI 1 ;* = 1
110E 47 MOV B,A ;CHAR INDEX
1110 87 ADD A ;2 * CHAR INDEX
1111 87 ADD A ;4 * CHAR INDEX
1112 80 ADD B ;5 * CHAR INDEX
1112 C6A2 FND: ADI GRAF AND 0FFH ;GRAF BASE L
1114 6F MOV L,A ;GRAF L
1115 3E11 MVI A,GRAF SHR 8 ;GRAF BASE H
1117 CE00 ACI 0 ;GRAF H
1119 67 MOV H,A ;GRAF H
111A C9 RET
111B C66F CNVER: ADI 0FFH ;SET CARRY
111D C9 RET

; BCH PRINTS ONE LINE OF THE CHARACTER
; WHOSE GRAF ORDINAL IS IN H AND L UPON
; ENTRY. THE LINE PRINTED IS DETERMINED
; BY THE MASK IN D.

111E 0E05 BCH: MVI C,5 ;COL COUNT
1120 22C912 BCH1: SHLD GADR ;SAVE GRAF ADDR
1123 7C MOV A,H
1124 45 MOV B,L
1125 21C912 LXI H,GADR
1128 70 MOV M,B
1129 23 INX H
112A CDC112 CALL INCHL

112D 77 MOV M,A
112E 67 MOV H,A
112F 68 MOV L,B

1130 7E MOV A,M ;GRAF CODE
1131 A7 ANA A ;SET FLAGS
1132 FA5411 JM BCH3 ;IF END OF CHAR
1135 43 MOV B,E ;CHAR TO PRINT
1136 A2 ANA D ;GRAF BIT
1137 C23C11 JNZ BCH2 ;TO PRINT CHAR
113A 0620 MVI B, ' ' ;SPACE
113C CD9C12 BCH2: CALL LO ;PRINT CHAR OR SP
113F 2AC912 LHLH GADR ;RESTORE GRAF ADDR
1142 21C912 LXI H, GADR
1145 7E MOV A, M
1146 23 INX H
1147 CDC112 CALL INCHL
114A 66 MOV H, M
114B 6F MOV L, A
114C 23 INX H ;FOR NEXT GRAF
114D CDC112 CALL INCHL
1150 0D DCR C ;DECR COL CNT
1151 C2011 JNZ BCH1 ;IF MORE
1154 0620 BCH3: MVI B, C ;SPACE
1156 CD9C12 CALL LO ;BETWEEN CHARs
1159 0620 MVI B, L ;SPACE
115B C39C12 JMP LO ;SPACE AND RET
115E 060D CRLF: MVI B, CR ;CAR RET
1160 CD9C12 CALL LO ;LIST
1163 050A MVI B, LF ;LINE FEED
1165 C39C12 JMP LO ;LIST AND RET

; PCH PUNCHES THE CHARACTER WHOSE GRAF
; ORDINAL IS IN H AND L UPON ENTRY
1168 0E05 PCH: MVI C, 5 ;CHAR COUNT
116A 22C912 PCH1: SHLD GADR ;SAVE GRAF ADDR
116D 7C MOV A, H
116E 45 MOV B, L
116F 21C912 LXI H, GADR
1172 70 MOV M, B
1173 23 INX H
1174 CDC112 CALL INCHL
1177 77 MOV M, A
1178 67 MOV H, A
1179 68 MOV L, B
117A 7E MOV A, M ;GRAF CODE
117B A7 ANA A ;SET FLAGS
117C FA9811 JM PCH2 ;IF END OF CHAR
117F 47 MOV B, A ;GRAF CODE
1180 CDB012 CALL PO ;PUNCH CODE
1183 2AC912 LHLH GADR ;RESTORE GRAF ADDR
1186 21C912 LXI H, GADR
1189 7E MOV A, M
118A 23     INX     H
118B CDC112 CALL INCHL
118E 66     MOV     H, M
118F 6F     MOV     L, A
1190 23     INX     H ; FOR NEXT GRAF
1191 CDC112 CALL INCHL
1194 0D     DCR     C ; DECR CHAR CNT
1195 C26A11 JNZ     PCH1 ; IF MORE
1198 0600   PCH2:   MVI     B, 0 ; BLANK
119A CDB012 CALL PO ; BETWEEN CHARS
119D 0600   MVI     B, 0 ; BLANK
119F C3B012 JMP     PO ; BLANK AND RET

THE GRAF TABLE CONTAINS THE GRAPHIC
DESCRIPTION OF THE LEGAL CHARACTERS.

11A2 00000000 GRAF:
  .A6 00
  .A7 14081480
  .A8 80
  .A9 081C0880
  .A0 80
  .A1 03080300
  .A2 80
  .A3 00080080
  .A4 80
  .A5 01080080
  .A6 80
  .A7 02040810
  .A8 80
  .A9 3E414141
  .A9 3E
  .A9 217F0180
  .A8 80
  .A8 21434549
  .A3 36
  .A4 22414949
  .A8 36
  .A9 0C14247F
  .A8 04
  .A9 7A494949
  .A9 46
  .A9 3E494949
  .A8 26
  .A9 43444850
  .A8 60

DB 00H, 00H, 00H, 00H, 00H ; SP
DB 14H, 08H, 14H, 08H, 08H ; *
DB 08H, 1CH, 08H, 08H, 08H ; +
DB 03H, 80H, 08H, 08H, 08H ; ,
DB 08H, 08H, 08H, 08H, 08H ; ;
DB 01H, 80H, 08H, 08H, 08H ; -
DB 03H, 04H, 08H, 10H, 08H ; /
DB 3EH, 41H, 41H, 41H, 3EH ; 0
DB 21H, 7FH, 01H, 08H, 08H ; 1
DB 21H, 43H, 45H, 49H, 36H ; 2
DB 22H, 41H, 49H, 49H, 36H ; 3
DB 08H, 14H, 24H, 7FH, 04H ; 4
DB 7AH, 49H, 49H, 49H, 46H ; 5
DB 3EH, 49H, 49H, 49H, 26H ; 6
DB 43H, 44H, 48H, 50H, 60H ; 7
<table>
<thead>
<tr>
<th>Address</th>
<th>Contents</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>11ED 36494949</td>
<td>DB 36H, 49H, 49H, 49H, 36H</td>
<td>:8</td>
</tr>
<tr>
<td>11F1 36</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11F2 32494949</td>
<td>DB 32H, 49H, 49H, 49H, 3EH</td>
<td>:9</td>
</tr>
<tr>
<td>11F6 3E</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11F7 12008080</td>
<td>DB 12H, 80H, 80H, 80H, 80H</td>
<td>::</td>
</tr>
<tr>
<td>11FB 80</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11FC 13008080</td>
<td>DB 13H, 80H, 80H, 80H, 80H</td>
<td>::</td>
</tr>
<tr>
<td>1200 80</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1201 08142241</td>
<td>DB 08H, 14H, 22H, 41H, 41H</td>
<td>:&lt;</td>
</tr>
<tr>
<td>1205 41</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1206 14141480</td>
<td>DB 14H, 14H, 14H, 80H, 80H</td>
<td>:=</td>
</tr>
<tr>
<td>1208 41</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1209 80</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1210 08142214</td>
<td>DB 41H, 41H, 22H, 14H, 08H</td>
<td>::</td>
</tr>
<tr>
<td>1214 30</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1215 7F404F41</td>
<td>DB 7FH, 40H, 4FH, 41H, 7FH</td>
<td>@</td>
</tr>
<tr>
<td>1219 7F</td>
<td></td>
<td></td>
</tr>
<tr>
<td>121A 7F484848</td>
<td>DB 7FH, 48H, 48H, 48H, 7FH</td>
<td>@A</td>
</tr>
<tr>
<td>121E 7F</td>
<td></td>
<td></td>
</tr>
<tr>
<td>121F 7F494949</td>
<td>DB 7FH, 49H, 49H, 49H, 36H</td>
<td>@B</td>
</tr>
<tr>
<td>:23 36</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1224 7F414141</td>
<td>DB 7FH, 41H, 41H, 41H, 41H</td>
<td>:C</td>
</tr>
<tr>
<td>1228 41</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1229 7F414141</td>
<td>DB 7FH, 41H, 41H, 41H, 3EH</td>
<td>:D</td>
</tr>
<tr>
<td>122D 3E</td>
<td></td>
<td></td>
</tr>
<tr>
<td>122E 7F494949</td>
<td>DB 7FH, 49H, 49H, 49H, 41H</td>
<td>:E</td>
</tr>
<tr>
<td>1232 41</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1233 7F484848</td>
<td>DB 7FH, 48H, 48H, 48H, 48H</td>
<td>:F</td>
</tr>
<tr>
<td>1237 40</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1238 7F414149</td>
<td>DB 7FH, 41H, 41H, 41H, 49H</td>
<td>:G</td>
</tr>
<tr>
<td>123C 4F</td>
<td></td>
<td></td>
</tr>
<tr>
<td>123D 7F0000008</td>
<td>DB 7FH, 08H, 08H, 08H, 7FH</td>
<td>:H</td>
</tr>
<tr>
<td>1241 7F</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1242 417F4180</td>
<td>DB 41H, 7FH, 41H, 80H, 80H</td>
<td>:I</td>
</tr>
<tr>
<td>1246 80</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1247 03010101</td>
<td>DB 03H, 01H, 01H, 01H, 7FH</td>
<td>:J</td>
</tr>
<tr>
<td>124B 7F</td>
<td></td>
<td></td>
</tr>
<tr>
<td>124C 7F081422</td>
<td>DB 7FH, 08H, 14H, 22H, 41H</td>
<td>:K</td>
</tr>
<tr>
<td>1250 41</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1251 7F010101</td>
<td>DB 7FH, 01H, 01H, 01H, 01H</td>
<td>:L</td>
</tr>
<tr>
<td>1255 01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1256 7F201020</td>
<td>DB 7FH, 20H, 10H, 20H, 7FH</td>
<td>:M</td>
</tr>
<tr>
<td>1259 7F</td>
<td></td>
<td></td>
</tr>
<tr>
<td>125B 7F300006</td>
<td>DB 7FH, 30H, 08H, 06H, 7FH</td>
<td>:N</td>
</tr>
<tr>
<td>125F 7F</td>
<td></td>
<td></td>
</tr>
<tr>
<td>:60 7F414141</td>
<td>DB 7FH, 41H, 41H, 41H, 7FH</td>
<td>:O</td>
</tr>
<tr>
<td>:64 7F</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1265 7F484848</td>
<td>DB 7FH, 48H, 48H, 48H, 78H</td>
<td>:P</td>
</tr>
<tr>
<td>1269 78</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4-60H
; THE FOLLOWING SUBROUTINES ARE SUBSTITUTES
; FOR MONITOR ROUTINES. NORMALLY THE
; CORRESPONDING ROUTINES IN MONITOR 2.1
; WOULD BE USED.

; OUTPUT A CHARACTER TO THE LIST DEVICE

129C 3E02  L0:   MVI    A, 2
129E D30A  L01:  OUT    0AH
12A0 DB01   L02:  IN     1
12A2 E604   ANI    4
12A4 C2A012  JNZ    L02
12A7 78     MOV    A, B
12A8 EEFF   XRI    0FFH
12AA D308   OUT    8
12AC AF     XRA    A
12AD D30A   OUT    0AH
12AF C9     RET

; OUTPUT A CHARACTER TO THE PUNCH

12B0 3E01  PO:   MVI    A, 1
12B2 C39E12  JMP    L01

; INPUT A CHARACTER FROM THE CONSOLE

12B5 DB01   CI:    IN    1
12B7 E601   ANI    1
12B9 C2B512  JNZ    CI
12BC DB00     IN     0
12BE EEFF    XRI     OFFH
12C0 C9     RET

; INCREMENT H AND L REGISTERS

12C1 2C  INCHL:  INR     L
12C2 00    RNZ
12C3 24  INR     H
12C4 C9     RET

; TEMPORARY STORAGE

12C5  BADR:  DS      2 ; BUFFER ADDR
12C7  CADR:  DS      2 ; CHAR ADDR
12C9  GADR:  DS      2 ; GRAF ADDR
12CB  BUFF:  DS      40 ; BUFFER
00000   END
Tape Labeler for MDS

Responds to "0" command in MDS monitor and allows user to insert a label on a paper tape he or she may be generating.

MDS-800, punch, and a PROM resident monitor

See listing

```
.O<TEXT FOR LABEL>,
```

Text is punch in large character on paper tape

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Assembler/Compiler Used:</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>MAC80</td>
</tr>
<tr>
<td>RAM Required:</td>
<td>Programmer:</td>
</tr>
<tr>
<td>Amount to store text</td>
<td>Matt Townsend</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>Company:</td>
</tr>
<tr>
<td>512 bytes</td>
<td>Address:</td>
</tr>
</tbody>
</table>
REF. NO. AB12
PROGRAM TITLE TAPE LABELER FOR MDS

; MDS MONITOR CALL ENTERED VIA THE "Q" COMMAND THAT WILL
; PROVIDE A LABEL FOR PAPER TAPES. REQUIRES A CHANGE IN
; THE MONITOR'S "CTBL" TO SET THE "Q" COMMAND JUMP ADDR
; FROM AN ERROR CONDITION TO THE DATA 00F6.
; THIS PROGRAM IS ENTERED FROM THE START ROUTINE IN THE
; MDFOR, WILL ACCEPT CHARACTERS, THEN UPON RECEIVING
; A <CR-LF> WILL PUNCH THE TITLE TO PTP. THIS PROGRAM
; UTILIZES MDS SUBROUTINES TO DO THE DIRTY WORK SO I'LL
; START BY DEFINING THEM

FEC0  LEAD   EQU 0FEC0H ; LEADER PUNCH
F80F  PUN   EQU 0F80FH  ; LISTER OUTPUT
FCA2  CI    EQU 0FCA2H  ; CONSOLE INPUT
FD3A  CO    EQU 0FD3AH  ; CONSOLE OUTPUT
3000  ORG 3000H

; NITTY GRITTY STARTS HERE

3000  21FF3D  LABEL:  LXI H, 3DFFH ; SET UP CHARACTER COUNTER
3003  1600    MV1 D, 00H
3005  CD92FC   PCMD:  CALL CI ; GO GET A CHARACTER
3008  4F      MOV C, A
3009  CD3AFD   CALL CO ; ECHO IT
300C  FE0D    CPI 0DH ; IS IT A CARR. RET?
300E  CA1730   JZ BANN ; JZ= YES, GO PUNCH THE CHARS.
3011  14      INR D ; INCR THE CHAR CNTR
3012  71      MOV M, C
3013  2B      DCX H ; SAVE THE CHAR IN C REG.
3014  C30530   JMP PCMD ; GO FOR MORE
3017  0E0A  BANN:  MV1 C, 0AH ; LF TO CONSOLE
3019  CD3AFD   CALL CO
301C  CDC0FE   CALL LEAD
301F  21FF3D   LXI H, 3DFFH
3022  1E0A    MV1 E, 0AH
3024  4E      BAN1:  MOV C, M ; FETCH 1ST CHAR
3025  CD3F30   CALL BANP ; PUNCH IT
3028  15      DCR D ; DECR CHAR CNTR
3029  CA2B30   JZ EXT ; JZ= EXT, GET MORE CHARS
302C  2B      DCX H ; POINT TO MEM LOC
302D  1D      DCR E
302E  C22430   JNZ BAN1
3031  0E0D    MV1 C, 0DH
3033  CD0FF8   CALL PUN
3036  1E0A    MV1 E, 0AH
3038  C32430   JMP BAN1

4-62
3038 CDC0FE
303E C9

EXT: CALL LEAD ; PUNCH TRAILER
       RET ; BACK TO MONITOR

; PUNCHES AN ASCII CHARACTER AS A BIG ONE ON PAPER
; TAPE ALLOWING MDS TAPES TO BE NAMED. ENTERS WITH THE
; CHARACTER IN C REG. USES MDS PUNCH SUBROUTINE TO THE
; TTY.

303F E5
3040 D5
3041 C5
3042 79
3043 E63F
3045 4F
3046 07
3047 07
3048 81
3049 216930
304C D25030
304F 24
3050 5F
3051 1600
3053 1F
3054 0605
3056 4E
3057 CD0FF8
3059 23
305B 05
305C C25630
305F 0E00
3061 CD0FF8
3064 CD0FF8
3067 CD0FF8
306A C1
306B D1
306C E1
306D C9

BANP: PUSH H
       PUSH D
       PUSH B
       MOV A,C
       ANI 3FH
       MOV C,A
       RLC
       RLC ; MULT BY FOUR
       ADD C ; MULT BY 5
       JNC BAN
       INR H
       INCR HIGH M LOC

BAN:  MOV E,A
       MVI D,00H
       DAD D ; SET H AND L TO TBL LOC
       MVI B,05H ; SET UP TABLE CNTR
       PBAN: MOV C,M ; FETCH TABLE CHAR
       CALL PUN
       INX H
       DCR B
       JNZ PBAN ; Z=END OF PUNCH

306E FC121112
3072 FC
3073 FF898989
3077 76
3078 7E818109
307C 42
307D FF818181
3081 7E
3082 FF898989
3086 81
3087 FF090909
3088 01
308C 7E818191

PTBL: DB 3740, 220, 210, 220, 3740 ; A
       DB 3770, 2110, 2110, 2110, 1660 ; B
       DB 1760, 2010, 2010, 1020 ; C
       DB 3770, 2110, 2110, 2110, 2010 ; E
       DB 3770, 110, 110, 110, 10 ; F
       DB 1760, 2010, 2610, 2210, 1620 ; G
3090 72
3091 FF008008
3095 FF
309F 00C9FFC9
309F 00
309F 4080C97F
309F 01
30A0 FF081C22
30A4 C1
30A5 FF080808
30A9 50
30AA FF020402
30AE FF
30AF FF040820
30B3 FF
30B4 7E818181
30B8 7E
30B9 FF090909
30BD 06
30BE 7E8181A1
30C2 7E
30C3 FF091929
30C7 C6
30C8 86898989
30CC 71
30CD 0101FF01
30D1 01
30D2 7F008060
30D6 7F
30D7 1F609060
30DB 1F
30DC FF402040
30E0 FF
30E1 E31C081C
30E5 E3
30E6 0304F804
30EA 03
30EB C1A19189
30EF 87
30F0 00FF8181
30F4 00
30F5 03041860
30F9 80
30FA 08181FF
30FE 00
30FF 0402FF02
3103 04
3104 2070A820
3108 20
3109 00000000

DB 3770, 100, 100, 100, 3770 ; H
DB 00, 201, 3770, 201, 00 ; I
DB 1000, 2000, 201, 1770, 10 ; J
DB 3770, 100, 340, 420, 3010 ; K
DB 3770, 2000, 2000, 2000, 2000 ; L
DB 3770, 20, 40, 20, 3770 ; M
DB 3770, 40, 100, 400, 3770 ; N
DB 3770, 110, 110, 110, 60 ; P
DB 1760, 2010, 2010, 2410, 1760 ; Q
DB 3770, 110, 310, 510, 3060 ; R
DB 2060, 2110, 2110, 2110, 1610 ; S
DB 10, 10, 3770, 10, 10 ; T
DB 1770, 2000, 2000, 2000, 1770 ; U
DB 370, 1400, 2000, 1400, 370 ; V
DB 3770, 1000, 400, 1000, 3770 ; W
DB 3430, 340, 100, 340, 3430 ; X
DB 30, 40, 3700, 40, 30 ; Y
DB 3010, 2410, 2210, 2110, 2070 ; Z
DB 00, 3770, 2010, 2010, 00 ; LEFT BRACKET
DB 30, 40, 300, 1400, 2000 ; /
DB 00, 2010, 2010, 3770, 00 ; RIGHT BRACKET
DB 40, 20, 3770, 20, 40 ; UP ARROW
DB 400, 1600, 2500, 400, 400 ; LEFT ARROW
DB 0, 0, 0, 0, 0 ; BUMMER
318A  7E
318B A6A50000
318F  00
3190 B6760000
3194  00
3195 03102442
3199  41
319A 24242424
319E  24
319F 81422410
31A3  08
31A4 02B90909
31A8  06
0000  END

DB  2460, 2460, 00, 00, 00
DB  2660, 1660, 00, 00, 00
DB  100, 200, 440, 1020, 1010
DB  440, 440, 440, 440, 440
DB  2010, 1020, 440, 200, 100
DB  20, 2710, 110, 110, 60
Page listing program

Provide facility for listing information in a paginated, numbered format. This is accomplished thru the system software with the console printer.

8080 Development System, Console device.

Monitor (Ver 2.0), tape to be listed.

The desired heading is typed on the console. (First page foot, then mark, then actual heading. Rubout echoes and deletes previous character, control 'I' is horizontal tab, type page #1 followed by a control 'P'. Include spaces for necessary decimal carry digits. A control 'D' terminates entry. Then the reader device supplies the body text. Finish by typing 'LC' until bottom of last page is marked by an extra heading.

The desired formated listing is obtained.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Assembler/Compiler Used:</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM Required:</td>
<td></td>
</tr>
<tr>
<td>CDH Bytes</td>
<td>8080 Resident Assembler</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>Programmer:</td>
</tr>
<tr>
<td>Monitor I/O Routines</td>
<td>Michael Christian</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td>Company:</td>
</tr>
<tr>
<td>Three levels</td>
<td>ARION CORPORATION</td>
</tr>
<tr>
<td></td>
<td>Address:</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
; REF NO. AB13
; PROGRAM NAME LISTING PROGRAM

; LINKAGE ADDRESSES:
3803   CI  EQU  3803H
3806   RI  EQU  3806H
3809   CO  EQU  3809H

; ASCII CONSTANTS:
000D   CR  EQU  00H
000A   LF  EQU  0AH
007F   RUBOUT EQU  7FH

; MAIN PROGRAM:

0040   ORG  40H     ; RE-ENTRY POINT, OR TO
0040 C32702   JMP  THEAD     ; RE-ENTRY POINT, OR TO

0200   ORG  200H     ; USE DEFAULT HEADING.

; NORMAL ENTRY POINT OF PROGRAM:
0200 219E02 BEGIN: LXI  H, MHEAD
0203 CD4002   CALL  PRINT     ; PRINT HEADING MESSAGE
0206 21C102   LXI  H, HEAD
0209 1600    MVI  D, 0
020B CD0338   B0:  CALL  CI
020E E67F    ANI  RUBOUT     ; TO REMOVE PARITY BIT
0210 FE04  CPI  'D'-'@'     ; CONTROL D = FINISHED
0212 CA2302  JZ  B1         ; IF HEADING COMPLETE
0215 FE7F    CPI  RUBOUT
0217 CA9002  JZ  RUB        ; DELETE LAST CHARACTER TYPED
021A 23     INX  H
021B 77      MOV  M, A      ; SAVE CHARACTOR
021C CD6502   CALL  PRT     ; ECHO CHARACTOR
021F 14      INR  D
0220 C30802   JMP  B0
0223 21C102  B1:  LXI  H, HEAD
0226 72      MOV  M, D      ; SAVE HEADING LENGTH
0227 0642  THEAD: MVI  B, 66     ; SET # OF LINES/PAGE
0229 21C102   LXI  H, HEAD
022C CD4002   CALL  PRINT
022F CD0638   READ: CALL  RI
0232 DA2F02   JC  READ      ; IF NOT RECEIVED, TRY AGAIN
0235 E67F    ANI  RUBOUT    ; REMOVE PARITY BIT
0237 CD6502  SEND:  CALL  PRT
023A DA2702   JC  THEAD     ; IF PAGE FULL, PRINT HEADING
023D C32F02     JMP   READ

; SUBROUTINE:
;
; PRINT MESSAGE. STARTING AT <HL>. FIRST BYTE IS LENGTH:
0240 56     PRINT: MOV   D,M
0241 23     PRO: INX   H
0242 7E     MOV   A,M
0243 CD6502   CALL  PRT       ; PRINT CHARACTER
0246 E5     PUSH  H
0247 FE10   CPI   'P'-@'     ; CONTROL 'P'= INCREMENT PAGE #
0249 CC5202   CZ  INCPN
024C E1     POP    H
024D 15     DCR    D
024E C24102  JNZ   PRO       ; IF NOT FINISHED
0251 C9     RET

; INCREMENT ASCII NUMBER STORE IN STRING
; IMMEDIATELY PRECEDING <HL> ADDRESS.
0252 2B     INCPN: DCX   H
0253 3E56   MVI    A,' '
255 BE     CMP    M
0256 C25B02  JNZ    $+5        ; SUBSTITUTE FOR LEADING BLANK
0259 3630   MVI    M,'0'
025B 34     INR    M
025C 3E39   MVI    A,'9'
025E BE     CMP    M        ; TEST FOR CARRY
025F F0     RP        ; IF ALL DONE
0260 3630   MVI    M,'0'
0262 C35202  JMP   INCPN       ; TO PROCESS CARRY

; PRINT CHARACTER RECEIVED IN <A>:
; IF <CONTROL I>. EXECUTE HORIZONTAL TAB.
; IF <CR>, ALSO RESET TAB COLUMN COUNTER (<E>).
; IF <LF>, ALSO DECREMENT LINES COUNTER (<B>).
; RETURNING CARRY SET IF 'BOTTOM OF PAGE'.
0265 FE09   PRT: CPI   'I'-'@'     ; CONTROL 'I'= HORIZONTAL TAB
0267 4F     MOV    C,A
0268 CD0938   CALL  CO
026B 79     MOV    A,C
026C FE0D   CPI    CR
026E C27302  JNZ    $+5     ; RESET COLUMN COUNTER
0271 1E00   MVI    E,0
0273 FE0A   CPI    LF
0275 C27B02  JNZ    $+6     ; ONE LINE PRINTED
0278 05     DCR    B        ; IF BOTTOM OF PAGE
~?79 37     STC
27A C8     RZ
027B FE20   CPI    
027D FA8102  JM    $+4      ; IF NON-PRINTING CHARACTER
0280 1C     INR     E       ; TO COUNT COLUMN PRINTED
0281 A7     ANA     A       ; CLEAR CARRY
0282 C9     RET
0283 0E20    TAB:     MVI     C, '/'
0285 CD0938  CALL    CO
0288 1C     INR     E
0289 3E07    MVI     A, 7
028B A3     ANA     E
028C C28302  JNZ     TAB     ; IF NOT DONE
028F C9     RET
0290 7A     RUB:     MOV     A, D
0291 A7     ANA     A
0292 CA0B02  JZ      B0     ; IF NUL STRING
0295 7E     MOV     A, M
0296 CD6502  CALL    PRT     ; ECHO CHARACTER REMOVED
0299 15     DCR     D
029A 2B     DCX     H
029B C30B02  JMP     B0     ; GET NEXT CHARACTER

; MESSAGES:
?9E 2200DA54 MHEAD:  DB     L1, CR, LF, 'TYPE PAGE HEADING:'
02A2 59504520
02A6 5041745
02AA 20484541
02AE 44494E47
02B2 3A

02B3 0D0A3C42  DB     CR, LF, '<BODY END>', CR, LF
02B7 4F445920
02BB 454E443E
02BF 0D0A

0022     L1     EQU    $-MHEAD-1
02C1 0A0A0A0A HEAD:  DB     L2, LF, LF, LF, LF, ; (DEFAULT HEADING)
02C5 0A
02C6 2E0D0A0A  DB     '. ', CR, LF, LF, LF, LF
~3CA 0A0A

000A     L2     EQU    $-HEAD-1
0000     END
**SOURCE PAPER TAPE TO MAGNETIC CASSETTE**

Will copy a source paper tape onto a magnetic cassette. End statement must be followed by a carriage return. Program will ignore leading blanks.

**TI Silent 700 Terminal connected in place of TTY.**

Supplied with Silent 700 terminal. Also see User's Library program Ref. No. AC8

**None**

Paper tape copied on magnetic cassette A X off character follows the end statement.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Assembler/Compiler Used:</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>8080 Macro Ver 3.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RAM Required:</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>157 decimal</td>
<td>John H. Brandt</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ROM Required:</th>
<th>Company:</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>E-Systems, Inc.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Maximum Subroutine Nesting Level:</th>
<th>Address:</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 bytes</td>
<td>Box 1056, Dept. 54221</td>
</tr>
<tr>
<td></td>
<td>Greenville, Texas 75401</td>
</tr>
</tbody>
</table>
SOURCE PAPER TAPE TO MAGNETIC CASSETTE

END STATEMENT IS FOLLOWING BY XOFF
PROGRAM Ignores LEADING BLANKS

3806 RI EQU 3806H ; READER ENTRY POINT
3763 TO EQU 3763H ; TAPE OUTPUT ENTRY POINT
0013 XOFF EQU 13H
000D CR EQU 00H
0044 DD EQU 44H
0045 EE EQU 45H
004E N EQU 4EH
0020 SPA EQU 20H
3D25 DELAY EQU 3D25H ; DELAY ENTRY POINT
100 ORG 100H
0100 2E00 MVI L, 00H
0102 1600 MVI D, 00H
0104 62 MOV H, D
0105 CD0638 START: CALL RI
0108 DA0501 JC START
010E 4F MOV C, A
010C 7A MOV R, D
010D FE01 CPI 01H
010F C21A01 JNZ COMCK
0112 79 MOV R, C
0113 FE0D CPI CR
0115 C26D01 JNZ TAPE
0118 1600 MVI D, 00H
011A 79 COMCK: MOV R, C
011B FE3B CPI 3BH ; COMMENT DELIMITER
011D CA5B01 JZ FLSET
0120 7C MOV A, H
0121 FE01 CPI 01H
0123 C26001 JNZ SPCK
0126 79 MOV R, C
0127 FE0D CPI CR
0129 C6E01 JZ SPEL
012C FE45 CPI EE
012E C26D01 JNZ TAPE
0131 CD6337 CALL TO
134 CD0638 CALL RI
0137 4F MOV C, A
0138 FE4E CPI N
013A C26D01      JNZ   TAPE
013D CD6337      CALL  TO
0140 CD0638      CALL  RI
0143 4F          MOV  C,A
0144 FE44        CPI   DD
0146 C26D01      JNZ  TAPE
0149 CD6337      CALL  TO
014C CD0638      CALL  RI
014F 4F          MOV  C,A
0150 FE0D        CPI   CR
0152 C26D01      JNZ  TAPE
0155 0E13        MVI  C,XOFF
0157 CD6337      CALL  TO
015A 76          HLT
015B 1601        FLSET:  MVI  D,01H
015D C36D01      JMP  TAPE
0160 79          SPCK:  MOV  A,C
0161 FE20        CPI   SPA
0163 C26D01      JNZ  TAPE
0166 2601        MVI  H,01H
0168 C36D01      JMP  TAPE
16B 2600        SPEL:  MVI  H,00H
016D 79          TAPE:  MOV  A,C
016E FE0D        CPI   CR
0170 CA7A01      J2Z  RESET
0173 2C          INR  L
0174 CD6337      CALL  TO
0177 C30501      JMP  START
017A 7D          RESET:  MOV  A,L ; CHECK FOR SHORT LINE
017B D619        SUI  25D
017D FA8801      JM   DELY ; ADD DELAY
0180 2E00        MVI  L,00H
0182 CD6337      CALL  TO
0185 C30501      JMP  START
0188 2F          DELY:  CMA
0189 C601        ADI  01H
018E 2E0D        MVI  L,00H
0190 CD253D      CALL  DELAY
0193 2D          DCR  L
0191 C28D01      JNZ  4
0194 3D          DCR  A
0195 C28B01      JNZ  DELY+3
0198 2E00        MVI  L,00H
019A CD6337      CALL  TO
019D C30501      JMP  START
0000
I Command (INSERT DATA in hexadecimal form from the TTY into RAM)

This program loads hexadecimal code into sequential RAM locations beginning at the address specified. An upper address limit may also be specified, if desired. It is useful for loading hex machine code directly into RAM for corrections, debugging, execution, or PROM programming.

TTY (Intellec 8/Mod 8 configuration uses port 0, 1 and 8)

INTELLEC 8/Mod 8 MONITOR, VERSION 2.1
Modified by changing two bytes in the command brand table as follows:

- address 3885_H was 43_H change to B0_H
- address 3886_H was 3C_H change to 3F_H

SEE ATTACHMENT

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Assembler/Compiler Used:</th>
</tr>
</thead>
<tbody>
<tr>
<td>A11:A,B,C,D,E,H,L</td>
<td>INTELLEC 8/Mod8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RAM Required:</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>(03,04,05,08,09,0A,0B) plus</td>
<td>G. Bechthold</td>
</tr>
<tr>
<td>bytes to be loaded by program</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ROM Required:</th>
<th>Company:</th>
</tr>
</thead>
<tbody>
<tr>
<td>(eq 3FB0 - 3FFF)</td>
<td>National Research Council</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Maximum Subroutine Nesting Level:</th>
<th>Address:</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 levels</td>
<td>Montreal Road</td>
</tr>
<tr>
<td></td>
<td>Ottawa, Ontario K1A 0R8</td>
</tr>
</tbody>
</table>
ATTACHMENT

Input Parameters

FORMAT

I low address, high address CR (or ,)
or I low address, CR (or ,)

xxxx (sp) HH (sp) HH (sp) HH (sp) ....

Note:
1. A space (sp) may be substituted for the , when entering commands
2. Parameters with underscore are printed by the I routine
3. XXXX - is the address of the byte beginning each line (similar to the D command DISPLAY DATA)
4. HH - two ASCII characters, entered via the TTY, converted to hexadecimal values and placed in a byte of memory
5. The high address in the command may be used when loading RAM to prevent overwriting data above that address. If it is omitted data may be written from the low address to the top of available RAM. Entry in either mode may be terminated by typing CR or (sp)
or ,

Error Conditions

1. If the low address or high address is greater than 16 bits, only the last 4 hex digits of the argument will be used as the address
2. If the low address is greater than the high address, only one byte at the low address will be inserted.
3. If the low address, high address or data contains an invalid character, or if the high address is omitted and two delimiters (, or (sp) followed by , or (sp) or CR) do not follow the low address the monitor will immediately type *(CR)(1F) and await the next command.
ATTACHMENT  Output Results

The hexadecimal address of the first memory location to be loaded is printed followed by a space. Two hexadecimal characters may then be typed and will be loaded into a single byte of memory, followed by an automatic space. This procedure continues, loading successive memory addresses, until a maximum of 16 bytes have been loaded. An automatic CR, LF, and the address of the first byte of this new line will then be printed. All lines are blocked into integral multiples of 16, as in the Display Data command, so the first and last lines may be less than 16 bytes in order to synchronize the display. Data entry continues until a CR is typed, the high address (if specified following the I command) is loaded, or the top address of available RAM is reached. The Insert operation is then terminated and control returned to the monitor.
; 8080 MACRO ASSEMBLER, V1.0

; REF. NO. AB15.
; PROGRAM <INSERT DATA IN HEXADECIMAL FORM FROM THE TTY INTO RAM>

; INSERT DATA IN HEXADECIMAL FORM
; FROM THE TTY INTO RAM.

3FB0  ORG 3FB0H
3FB0 CD003D INSRT: CALL 3D80H ; EXP-ENTER ADDRESS LIMITS
3FB3 CDCC7C CALL 3CC7H ; CR/LF
3FB6 CDCF3D CALL 3DCFH ; GETAD-RETRIVE ADDRESS
3FB9 AF XRA A ; CHECK IF UPPER LIMIT
3FBA 92 SUB D ; OF ADDRESS =0 <ADDRESS
3FBB 9B SBB E ; NOT SPECIFIED
3FBC CAF33F JZ UPLIM ; YES
3FBD CDFB3D CALL 3DFBH ; LADR-PRINT ADDRESS
3FC2 CD4B3C NXTBY: CALL 3C4BH ; BLANK
3FC5 CD443F CALL 3F44H ; RII-READ IN 1ST. CHAR.
3FC8 CD533E CALL 3E53H ; NIBBLE-CONV. TO HEX, CHECK ILLE
3FCB DA433C JC 3C43H ; LER-PRINT * RET. TO MONITOR
3FCC 07 RLC
3FCD 07 RLC
3FDE 07 RLC
3FF0 07 RLC
3FF1 07 RLC
3FF2 4F MOV C,A ; TEMP. STORE 1ST. CHAR.
3FF3 CD443F CALL 3F44H ; RII-READ IN 2ND. CHAR.
3FF6 CD533E CALL 3E53H ; NIBBLE-CONV. TO HEX, CHECK ILLE
3FF9 DA433C JC 3C43H ; LER-PRINT * RET. TO MONITOR
3FFC B1 ORA C
3FD0 CDCF3D CALL 3DCFH ; GETAD-RETRIVE ADDRESS
3FE0 77 MOV M,A ; STORE BYTE
3FE1 CD003D CALL 3D00H ; HILO-CHECK UPPER LIMIT
3FE4 DA438 JC 3B44H ; YES. RET. TO MONITOR
3FE7 CD2F3F CALL 3F2FH ; SAVIT-STORE ADDRESS
3FE8 79 MOV A,C ; CHECK FOR COMPLETION OF
3FE9 E60F ANI 0FH ; LINE <16 BYTES>
3FEA C2C23F JNZ NXTBY ; GET NEXT TWO CHAR.
3FF0 C3B33F JMP INSRT+3 ; NEW LINE, PRINT ADDRESS
3FF3 CD3E3E UPLIM: CALL 3E3EH ; MEMCK-FIND UPPER LIM. RAM
3FF6 210A00 LXI H,000AH ; LOAD UPPER LIMIT
3FF9 71 MOV M,C ; LOCATIONS OF THE
3FFA 2C INR L ; I PROGRAM
3FFB 77 MOV M,A
3FFC C3B63F JMP INSRT+6 ; PRINT ADDRESS & BEGIN
000 END
8080 RAM MEMORY TEST

MEMORY TEST for Intellec/80 system

Intellec/80 I/O board
(TTY data output port $08H)
(TTY status input port $01H)
(DISPLAY output port $FFH)

Required Hardware

None

Required Software

TTY may be disabled (for long test periods) by typing 2 successive keys (causing over-run error) and turning it off. Assembler Values "TOP" and "BOTTOM" define the limits of the RAM addresses under test (normally $0000H to $2000H).

Input Parameters

Four types of error messages can occur (if the TTY is enabled):

1) Instead of $67H, $65H was written into address $1234H;
   -$67-$65-$1234 ($02H shown on front panel display)

2) Instead of $67H, $65H was read from address $1234H;
   -$67-$67-$1234 ($02H shown on front panel display)

3) Address $1234 not in RAM under test;
   X-$55-$FF-$1234 ($55H shown on front panel display)

4) $31H was written into address $1234 while addressing another
   address in memory (i.e. "Cross-Write error")
   X-$55-$31-$1234 ($55H shown on front panel display)

NOTE---The error bit pattern will always be displayed.

Registers Modified:

ALL

Assembler/Compiler Used:
MAC80 or Resident Assembler

RAM Required:
NONE (except RAM under test)

Programmer:
B. Searle

ROM Required:
256 Bytes (1 chip)

Company:
Ministry of Transport (CANADA)

Maximum Subroutine Nesting Level:
No stack used

Address: T.A.C.D., Floor 10-C
          Tower "C", Place De Ville
          OTTAWA, CANADA

98-0348
4-81
Memory Diagnostic Program

Writes test bytes in any range of memory and compares the written bit combination with what is read. Upon detection of a defective memory location, an error message is printed specifying the address, reference and actual values.

TTY on Port 0 + 1

Intellec 80 System Monitor

Address range input from TTY-console:

<low address>,<high address> <CR>

Delimiter between values = ','
- Closing input parameters with CR
- Parameters in HEX

Error message if program finds a defective memory location:

ADDR:  REF:  ACT:  (TABLE TITLES)
XXXX  XX  XX  (ERROR MESSAGE)
  .  .  .

Registers Modified:

<table>
<thead>
<tr>
<th></th>
<th>Assembler/Compiler Used:</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>Intellec 80 Macro Assembler</td>
</tr>
<tr>
<td>RAM Required:</td>
<td>Programmer: P. Schneider/H.P. Teufer</td>
</tr>
<tr>
<td>12H</td>
<td>Company: Landis &amp; Gyr AG</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>Address: CH-6301, Zug Switzerland</td>
</tr>
<tr>
<td>181H</td>
<td></td>
</tr>
</tbody>
</table>
Program Title
Compare Object Code Tape with Memory

This program extends the Intellec 8 System Monitor's 'Compare' command to check the data from a hex format tape against the current information stored in memory.

Required Hardware
Intellect 8/80 Development System or equivalent (with tape reader)

Required Software
8080 Monitor, Version 2.0 (for other versions, linkage addresses must be corrected)

Input Parameters
To compare Prom, type "C(address)(CR)" (no change)
To compare a tape, ready the tape and type "C(CR)" (omitting the memory address, as the address is specified by the hex format tape.)

Output Results
If all data checks, control is passed back to the Monitor's command loop. If a mismatch is found, it is reported in the following format: (Mem address) (Mem contents) (Compare data)
Also strings of mismatches are separated by a blank in the 'Compare tape' routine only.

Registers Modified:
All

RAM Required:
None

ROM Required:
90H

Maximum Subroutine Nesting Level:
(not significant)

Assembler/Compiler Used:
8080 Macro Assembler

Programmer:
Michael Christian

Company:
Arion Corporation

Address:
825 Boone Avenue North
Minneapolis, MN  55427
; REF NO. AB16.
; PROGRAM NAME COMPAR OBJECT CODE TAPE WITH MEMORY
;
;
; COMPARE OBJECT CODE TAPE WITH MEMORY
; 8/20/75
;
; THIS PROGRAM EXTEND THE MONITOR 'COMPARE'
; COMMAND TO CHECK THE DATA FROM A HEX FORMAT
; OBJECT CODE TAPE AGAINST THE CURRENT INFORMATION
; STORED IN MEMORY, AND REPORTS ANY MISMATCHES
; FOUND.
;
; TO COMPARE A PROM, TYPE "C<ADDRESS><CR>".
; TO COMPARE A TAPE, LOAD THE TAPE TO BE COMPARED,
; AND TYPE "C<CR>" (OMITTING THE ADDRESS).
;
; THE FORMAT OF ANY MISMATCHES FOUND IS:
; <MEM ADDRESS> <MEM CONTENTS> <COMPARE DATA>
;
; MONITOR <VER 2.0> ADDRESSES REFERENCED -
;
395A COMP EQU 395AH
3F6E TI EQU 3F6EH
3E11 NIBBLE EQU 3E11H
3D71 EX1 EQU 3D71H
000D CR EQU 00H
3C3E LER EQU 3C3EH
3EA0 RI EQU 3EA0H
3C7D BYTE EQU 3C7DH
3CC8 CRLF EQU 3CC8H
3DBA LADR EQU 3DBAH
3DC2 LBYTE EQU 3DC2H
3C4E BLK EQU 3C4EH
3C50 CO EQU 3C50H
386B START EQU 386BH
;
;******************************************************************************
;
; ENTRY POINT - FIND OUT IF COMPARE IS WITH
; PROM OR WITH TAPE.
;
400 ORG 3400H ; OR ANY AVAILABLE ROM ADDRESS
COMPARE:
3400 0D DCR C ; GET ONE PARAMETER
3401 21E239 LXI H, COMP+4
3404 E5 PUSH H ; SET UP RETURN ADDRESS FROM EXPR
3405 210000 LXI H, 0
3408 CD6E3F CALL Ti ; GET FIRST CHARACTER
340B 47 MOV B, A ; SAVE TO TEST AS DELIMITER
340C CD11E3 CALL NIBBLE
340F D2713D JNC EX1,7 ; IF HEX DIGIT, REJOIN EXPR CODE
3412 D1 POP D ; BUMP DUMMY RETURN ADDRESS
3413 78 MOV A, B
3414 FE0D CPI CR
3416 C23E3C JNZ LER ; NOT VALID DELIMITER, ERROR

;**********************************************
; READ HEX TAPE - GET NEXT BYTE AND TEST MEMORY ADDRESS
; SPECIFIED IN RECORD. REPORT MISMATCH, IF FOUND.
;
3419 E5 PUSH H ; RESET MISMATCH FLAG & ZERO COUNTER
341A DA03E3 RED0: CALL RI
341D D93E3C JC LER ; IF END OF TAPE, ERROR
3420 063A MVI B, :/
3422 90 SUB B
3423 C21A34 JNZ RED0 ; LOOP UNTIL START OF RECORD
3426 57 MOV D, A ; ZERO CHECKSUM
3427 CD7D3C CALL BYTE
342A CA7B34 JZ FINISH ; ZERO RECORD LENGTH, ALL DONE
342D 5F MOV E, A
342E CD7D3C CALL BYTE ; GET MSB OF ADDRESS
3431 F5 PUSH PSW
3432 CD7D3C CALL BYTE ; GET LSB OF ADDRESS
3435 E1 POP H
3436 6F MOV L, A ; SET UP STARTING ADDRESS IN HL
3437 CD7D3C CALL BYTE ; RECORD TYPE
343A CD7D3C RED1: CALL BYTE ; GET DATA
343D BE CMP M
343E CA6034 JZ MATCH ; IF GOOD DATA, SKIP REPORT
3441 F5 PUSH PSW
3442 CDC83C CALL CRFL
3445 CDBA3D CALL LADR ; PRINT ADDRESS OF MISMATCH
3448 CD4E3C CALL BLK
344B 7E MOV A, M
344C CDC23D CALL LBYTE ; PRINT STORED DATA
344F CD4E3C CALL BLK
3452 F1 POP PSW
3453 CDC23D CALL LBYTE ; PRINT READ DATA
3456 E3 XTHL
3457 23 INX H ; COUNT ONE MISMATCH
3458 3E80 MVI A, 80H

4-88
345A B4 ORA H
345B 67 MOV H.A SET MISMATCH FLAG
345C E3 XTHL
345D C36B34 JMP NEXT

; UPDATE POINTER AND TEST IF END OF ONE RECORD.
; MATCH: XTHL ; GET COUNTER AND FLAG
3460 E3 MOV A.H
3461 7C ORA A
3462 B7 CM CRLF ; MARK END OF MISMATCH STRING
3463 FCC83C MOV A.7FH
3464 3E7F MVI A
3465 A4 ANA H ; CLEAR FLAG
3466 67 MOV H.A
3467 E3 XTHL
3468 23 NEXT: INX H ; POINT TO NEXT MEMORY LOCATION
3469 C1D DCR E ; ONE RECORD READ
346A C23A34 JNZ RED1 ; IF NOT DONE, GET NEXT BYTE
346B CD7D3C CALL BYTE ; GET CHECKSUM
346C 473 CA1A34 JZ RED0 ; IF OK, GET NEXT RECORD
346D 0E2A MVI C, '*' ; FINISH: CALL BLK
346E CD503C CALL CO ; FLAG CHECKSUM ERROR
346F CD4E3C CALL BLK
3470 2B DCX H
3471 CDBA3D CALL LADR ; PRINT LAST ADDRESS CHECKED
3472 CDC83C CALL CRLF
3473 E1 POP H
3474 3E7F MVI A.7FH
3475 A4 ANA H
3476 67 MOV H.A
3477 CD6A3D CALL LADR ; PRINT # OF MISMATCHES
3478 C36B38 JMP NEXT ; RETURN TO MONITOR

; ORG 3896H
3896 0034 DW COMPARE ; MODIFY THE MONITOR'S COMMAND BRANCH TABLE TO JUMP TO NEW ROUTINE.

; END
Program Title

K, PROGRAM TRAP

Function

This program provides two traps (search/wait) for debugging other programs which use RAM memory. Displays the contents of five registers and the trap address when the trap occurs.

Required Hardware

Intellec 8 computer and TTY on Port 0.

Required Software

Monitor VER. 1.1 as the trap program is written. For Monitor VER. 2.1 change the trap program equates (See the attached sheet).

Input Parameters

See the attached sheet.

Output Results

See the attached sheet.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>A, B, C, D, E, H, L</td>
<td>R. T. Kirk</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RAM Required:</th>
<th>Company:</th>
</tr>
</thead>
<tbody>
<tr>
<td>10H to 1FH</td>
<td>Kenway Incorporated (801)292-2401</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ROM Required:</th>
<th>Address:</th>
</tr>
</thead>
<tbody>
<tr>
<td>3100 to 323E</td>
<td>525 W. 350</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Maximum Subroutine Nesting Level:</th>
<th>City:</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>N. Bountiful</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Assembler/Compiler Used:</th>
<th>State:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intellec Assembler Ver. 1.0</td>
<td>UTAH 84010</td>
</tr>
</tbody>
</table>
; TITLE: K, PROGRAM TRAP
; A PROGRAM BREAK POINT OR TRAP IS SET BY THIS PROGRAM.
; FOLLOWING POWER ON START, TYPE G3230 CR TO INITIALIZE
; THE TRAP PROGRAM.
; THIS PROGRAM CAN ONLY BE USED BY RAM TYPE MEMORY.
; TO SET A TRAP TYPE K FOLLOWED BY TRAP ADDRESS AND CR.
; TO CLEAR ONE OR TWO TRAPS TYPE K FOLLOWED BY CR.
; A MAXIMUM OF TWO TRAPS CAN BE SET. IF THE MAXIMUM IS
; EXCEEDED TERR WILL BE PRINTED ON THE TTY.
; WHEN A TRAP IS ENCOUNTERED THE CONTENTS OF THE A,B,C,
; D,E REGISTERS ARE PRINTED FOLLOWED BY THE TRAP
; ADDRESS. THE FORMAT IS: AB CD E0 ADDRESS
; THE TRAP IS CLEARED WHEN ENCOUNTERED.
; A TRAP requires 3 MEMORY LOCATIONS. THEREFORE NEVER
; ALLOW THE TRAP TO AFFECT AN ENTRY POINT.
; EXAMPLE: A TRAP CANBE SET AT TR1: MOV A,M, DEC C OR
; JNZ TR10 BUT NOT AT TR2: MVI D,OFFH UNLESS
; IT CANBE DETERMINED THAT PROGRAM EXECUTION
; WILL NEVER ARRIVE AT TR3 BEFORE THE TRAP.
; IS ENCOUNTERED.
; TR1: MOV A,M
; DEC C
; JNZ TR10
; TR2: MVI D,OFFH
; TR3: MVI E,OFFH
; FOR MONITOR VER 1.1 CHANGE THE CONTENTS OF MEMORY
; LOCATIONS 3C21 AND 3C22 TO 3100 HEX. FOR MONITOR
; VER 2.1 CHANGE THE CONTENTS OF MEMORY LOCATIONS
; 38B9 AND 38B0 TO 3100 HEX.
; FOR MONITOR VERSION 2.1 CHANGE THE FOLLOWING EQUATES TO:
; EXPR EQU 3D80H
; INCHL EQU 3DEBH
; START EQU 3844H
; TTY EQU 3C57H
; VERO EQU 3835H
;
DEBUG - A two prom debugging package to be used in minimum 8080 systems.

To inspect, dump, move and find data in memory.

<table>
<thead>
<tr>
<th>Input Parameters</th>
<th>SEE WRITEUP</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Program Title</th>
<th>DEBUG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function</td>
<td></td>
</tr>
</tbody>
</table>

| Required Hardware |             |
| Required Software |             |

| Output Results    |             |

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL</td>
<td>Tom Lafleur</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RAM Required: Stack and one location above stack for flag</th>
<th>Company:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Naval Electronics Lab Ctr.,</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ROM Required: 2-1702 (512 location)</th>
<th>Address:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>La Posta Observatory</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Maximum Subroutine Nesting Level:</th>
<th>City:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Campo</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Assembler/Compiler Used: 8080 Macro Assembler</th>
<th>State:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CA 92006</td>
</tr>
</tbody>
</table>

© Intel Corporation, 1976
COMMAND SYNTAX

Inspect Memory
I ADDR = XXXX
 (Address to inspect)
     (XXXX) DD XX
    (Space bar to go to next location)
    (New data (two hex character)
    (CTL 'A' (01H) will ask for new address
    (Current Data
    (Address

Dump Memory in Hex
D ADDR = XXXX, XXXX
  (Last location)
  (Dump range)
  (First location)
Will display 16 memory locations per line. (XXXX) DD DD DD DD DD.

Move Block of Memory
M ADDR = XXXX, XXXX TO XXXX
  (New address of block)
  (Last address of block)
  (First address of block

Find Byte
F ADDR = XXXX, XXXX, D=DD, N=DD
  (Carriage return if no replacement
  (or replacement data)
  (Search data)
  (Last Address of search range
  (First address of search range
This command will print out the address of all locations that contain the search data. An option is also available that will replace the search data with a new data byte.

GOTO
G ADDR = XXXX
  (Jump address
Will exit with interrupt on.
Notes:
1. Underlined data is user entered.
2. Dump and find routine allows for a return to the command loop by pressing any key on the TTY.
3. All routine allows for exit by pressing ALT-MODE (7DN) or break (00H) key in response to any input, will respond with '?'.

Options
Commands are given for jumps to user-written high speed I/O devices.
The commands are:
- **R** (Jump 3500H)  Read hex
- **W** (Jump 3503H)  Write hex routine
- **E** (Jump 3506H)  End file

Output Results
First time through debug will give an ID message; then will type out a '?' when it's ready for input.
PUNCH TEST or TTY READER/PUNCH TEST

1) Tests paper tape punch (using High Speed or TTY Reader)
2) Complete TTY READER/PUNCH TEST

INTELLEC/80: I/O board for TTY output

INTELLEC/80 monitor (or equivalent I/O drivers).

None.

1) Binary count tape (& TTY listing if TTY punch is used).
2) On errors, TTY prints actual & expected data before exiting to monitor

Stop CPU and advance reader tape one character to simulate a character error.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Assembler/Compiler Used:</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL</td>
<td>Mac-80 resident or cross Assembler</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RAM Required:</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$ (except stack)</td>
<td>B. Searle</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ROM Required:</th>
<th>Company:</th>
</tr>
</thead>
<tbody>
<tr>
<td>3F Hex</td>
<td>Ministry of Transport (CANADA)</td>
</tr>
</tbody>
</table>

Maximum Subroutine Nesting Level: about 4

Address: TACD, Tower C, Place de Ville
          Ottawa, Ontario, CANADA; K1A 0N8
; REF. NO. AA4.
; PROGRAM NAME PUNCH TEST OR TTY READER/PUNCH TEST
;
;
TITLE ' ----- RPTEST ----- ' 
;
;
;******************************************************************************
;******************************************************************************
;
TELETEYPE READER/PUNCH TEST
;
START TEST AT LOCATION 1000 (HEX), THEN
PUT PUNCHED LEADER IN READER & START
READER WHILE BINARY COUNT IS PUNCHING

IF ERROR OCCURS, THE ACTUAL CHARACTER
READ IN AND THE EXPECTED CHARACTER ARE
TYPED CUT. THE TEST PROGRAM THEN EXITS
TO THE INTELLEC 80 MONITOR.

AUTHOR: B. SEARLE
MINISTRY OF TRANSPORT
OTTAWA, CANADA
(613) 996-0221

DATE: JANUARY 30, 1975

;******************************************************************************
;******************************************************************************
;
;****** INTELLEC 8/80 MONITOR ENTRY POINTS, VERSION 1.1
;VERSION 2.0 ENTRY POINT
;
3C30          BLK    EQU    3C30H     ;3CE4
3CAD          CRLF   EQU    3CADH    ;3CC8
3DB0          LBYTE  EQU    3DB0H    ;3DC2
7C0           LEAD   EQU    3D0CH    ;3DD2
-26C          PO     EQU    3E6CH    ;3E78
3E94          RI     EQU    3E94H    ;3E90
3800          MONITR EQU    3800H    ;3800
; "PUNCH" CONDITIONAL ASSEMBLY CONTROL

0000 FALSE EQU 0000H
00FF TRUE EQU 00FFH
00FF PUNCH EQU TRUE

3640 ORG 3640H

; START UP

3640 FB START: EI
3641 CDAD3C CALL CRLF ; PRINT CR & LF

IF PUNCH ;;;;;;;;

; FIRST PUNCH LEADER, SYNC CHARACTER, AND
; ONE FULL BINARY COUNT PATTERN

3644 CDC03D INITP: CALL LEAD ; PUNCH LEADER
3647 0EFF MVI C, 00FFH ; PUNCH SYNC CHARACTER
3649 CD6C3E CALL PO
364C 0E00 MVI C, 00H ; INITIALIZE PUNCH CHAR
364E CD6C3E INIT1: CALL PO ; PUNCH & INCREMENT CHAR
3651 0C INR C ; LOOP UNTIL FULL COUNT
3652 C24E36 JNZ INIT1
3653 ENDF IF ;;;;;;;;

; NOW INITIALIZE READER TAPE POSITION

3655 CD943E INITR: CALL RI ; READ UNTIL SYNC CHARAC
3658 3C INR A ; INITIALIZE COUNT PATTE
3659 C25536 JNZ INITR
365C 5F MOV E, A

; TEST: READ TAPE & COMPARE TO COUNT PATTERN

;
365D  CD943E  TEST1: CALL RI ; READ NEXT CHARACTER
3660   93    SUB E ; TEST AND BRANCH IF ERR
3661  C46C3E   CNZ ERROR ; PUNCH CHARACTER
3664   4B    MOV C,E
3665  CD6C3E   CALL PO
3668   1C    ENDF ; LOOP TO NEXT CHARACTER
3669  C35D3E   JMP TEST1

;****** ERROR PRINT OF ACTUAL & EXPECTED CHARACTER

366C   83    ERROR: ADD E ; RESTORE ACTUAL CHARACTER
366D  CD803D   CALL LBYTE ; PRINT ACTUAL CHARACTER
3670  CD303C   CALL BLK ; PRINT BLANK
3673   7B    MOV A,E ; PRINT EXPECTED CHARACTER
3674  CDB03D   CALL LBYTE
3677  CDAD3C   CALL CRLF
367A  C30038   IF PUNCH ; PRINT CR, LF
367D  C34036   JMP MONITR
367E  C30038   ENDF

0000   END ; OR IF PRINT & PUNCH ARE DIFFERENT, COULD
           ; RE-SYNC & CONTINUE TEST
**READER TEST**

Tests High Speed paper tape reader or teletype reader.

**INTELLEC 8/80:** I/O board for reader control, I/O board for TTY output.

**INTELLEC 8/80** monitor (or equivalent I/O drivers).

Binary count tape (use a patched-up continuous loop for a continuous test).

**Actual reader data & expected data are typed out on TTY for any read error, and the input tape is re-synchronized (in case of dropped or extra characters).**

**Stop CPU and advance reader tape one character to simulate a character error.**

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Assembler/Compiler Used:</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL</td>
<td>Mac-60 resident or cross Assembler</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RAM Required:</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ø (except stack)</td>
<td>B. Searle</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ROM Required:</th>
<th>Company:</th>
</tr>
</thead>
<tbody>
<tr>
<td>25 Hex</td>
<td>Ministry of Transport (CANADA)</td>
</tr>
</tbody>
</table>

| Maximum Subroutine Nesting Level: | Address: | |
|-----------------------------------|----------|
| about 4                           | TACD, Tower C, Place de Ville |
|                                   | OTTAWA, ONTARIO, CANADA K1A ON6 |
; REF. NO. AA5
; PROGRAM TITLE READER TEST
;
;
TITLE: "----- RPTEST -----"
;
;******************************************************************************
;******************************************************************************
;
TELETYPE READER PUNCH TEST
;
START TEST AT LOCATION 1000 (HEX), THEN
PUT PUNCHED LEADER IN READER & START
READER WHILE BINARY COUNT IS PUNCHING
;
IF ERROR OCCURS, THE ACTUAL CHARACTER
READ IN AND THE EXPECTED CHARACTER ARE
TYPED OUT. THE TEST PROGRAM THEN EXITS
TO THE INTELLEC 80 MONITOR.
;
AUTHOR: S. SEARLE
MINISTRY OF TRANSPORT
OTTAWA, CANADA
(613) 996-0221
;
DATE: JANUARY 30, 1975
;
;******************************************************************************
;******************************************************************************
;
;***** INTELLEC 8/80 MONITOR ENTRY POINTS, VERSION 1.1
; ; VERSION 2.0 ENTRY POINT
;
3C30 BLK EQU 3C30H ;3CE4
3C40 CRLF EQU 3C40H ;3CC8
3DB0 LBYTE EQU 3DB0H ;3DC2
3DC0 LEAD EQU 3DC0H ;3DD2
3E94 PO EQU 3E94H ;3E78
3E94 RI EQU 3E94H ;3EA0
3800 MONITR EQU 3800H ;3800
;
4-105
**** "PUNCH" CONDITIONAL ASSEMBLY CONTROL ****

0000      FALSE  EQU  0000H
00FF      TRUE   EQU  00FFH
0000      PUNCH   EQU  FALSE

3600      ORG  3600H

****** START UP ******

3600 FB    START: EI
3601 CD403C CALL  CRLF ;PRINT CR & LF

IF PUNCH ;;;;;;
****** FIRST PUNCH LEADER, SYNC CHARACTER, AND ******
****** ONE FULL BINARY COUNT PATTERN ******

INITP: CALL  LEAD ;PUNCH LEADER
        MVI   C,00FFH ;PUNCH SYNC CHARACTER
        CALL  PO
        MVI   C,00H
INIT1: CALL  PO
        INR   C
        JNZ   INIT1 ;LOOP UNTIL FULL COUNT P
        ENDF   ;;;;;;

****** NOW INITIALIZE READER TAPE POSTION ******

3604 CD943E INITR: CALL  RI ;READ UNTIL SYNC CHARACT
3607 3C      INR   A
3608 C20436   JNZ   INITR
3608 5F      MOV   E,A ;INITIALIZE COUNT PATTERN

****** TEST: READ TAPE & COMPARE TO COUNT PATTERN ******
360C CD943E TEST1: CALL RI ;READ NEXT CHARACTER
360F 93 SUB E ;TEST AND BRANCH IF ERROR
3610 C41736 CNZ ERROR ;PUNCH CHARACTER
IF PUNCH;;;;;
MOV C,E
CALL PO
ENDIF ;LOOP TO NEXT CHARACTER
;
3613 1C INR E
3614 C30C36 JMP TEST1

;****** ERROR PRINT OF ACTUAL & EXPECTED CHARACTER

3617 83 ERROR: ADD E ;RESTORE ACTUAL CHARACTER
3618 CD003D CALL LBYTE ;PRINT ACTUAL CHARACTER
361B CD303C CALL BLK ;PRINT BLANK
361E 7B MOV A,E ;PRINT EXPECTED CHARACTER
361F CD003D CALL LBYTE
3622 CD403C CALL CR LF
IF PUNCH;;;;;
JMP MONITR ;PRINT CR, LF
ENDIF ;EXITS TO MONITOR
3625 C30036 JMP START ;OR IF PRINT & PUNCH DEV
;ARE DIFFERENT, COULD...
;RE-SYNC & CONTINUE TEST

0000 END
TALLY R2050 HS PTR DRIVER

Extension to the Intel 8080 monitor to handle a Tally model R2050 photoelectric tape reader at 200 cps.

**Intellec 8 Mod 80**
- Tally model R2050 photoelectric tape reader
- Power supply for the reader
- Connecting cables

**Intel 8080 monitor**

None

Data read is passed back to the monitor

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Assembler/Compiler Used:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data in A all others, saved &amp; restored</td>
<td>8080 Macro Assembler, V2.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RAM Required:</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>J.J. deZubeldia</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ROM Required:</th>
<th>Company:</th>
</tr>
</thead>
<tbody>
<tr>
<td>44 bytes</td>
<td>IKASLAN S.A.</td>
</tr>
</tbody>
</table>

Maximum Subroutine Nesting Level: 2

Address: Barrio de Zabalondo MUNGUIA (Vizcaya) SPAIN
REF NO. AC4.
PROGRAM NAME TALLY R2050 HSPTK DRIVER

INTEL 8080 MONITOR VER. 1.0
EXTENSION TO HANDLE A TALLY MODEL R 2050
PHOTOELECTRIC TAPE READER AT 200 CPS.
USING THE INTELLEC INPUT PORT #3 AS TAPE
READER DATA INPUT AND BIT #2 (00000100)
ON OUTPUT PORT #1 AS TAPE ADVANCE.
READER OUTPUTS INVERTED DATA.
HANDLER RETURNS NON-INVERTED DATA IN A.

JZ 16. 4. 75
VER. 2. 1

J003   DIPR  EQU  3  ;DATA INPUT PORT
0001   COPR  EQU  1  ;COMMAND OUTPUT PORT
0004   CBIT  EQU  4  ;COMMAND BIT ON COPR

3706   ORG   3706H  ;TABLE ENTRY FOR USER
3706 C31B37  JMP   371BH  ;JUMP TO DRIVER
371B   ORG   371BH
371B DB03  IN    DIPR  ;INPUT DATA
371D  2F    CMN   CBIT  ;COMPLEMENT DATA
371E B7    ORA   A  ;CLEAR CARRY
371F F5    PUSH  PSW
3720 AF    XRA   A  ;CLEAR ACCUMULATOR
3721 D301  OUT   COPR  ;SEND A 100 MICROSEC.
3723 CD2F37  CALL  DCMI  ;ADVANCE PULSE TO
3726 3E04  MVI   A, 12  ;THE READER THROUGH
3728 D301  OUT   COPR  ;PORT COPR
372A CD3837  CALL  DLYM  ;5 MSEC. DELAY
372D F1    POP   PSW
372E C9    RET

100 MICROSEC. DELAY SUBROUTINE

72F F5    DCMI  ;PUSH  PSW
3730 3E0C  MVI   A,12
3732 3D    DCM0  ;DCR   A
3733 C23237       JNZ       DCM0
3736 F1           POP       PSW
3737 C9           RET

; 5 MILLISEC. DELAY SUBROUTINE

3738 F5           DLYM:     PUSH PSW
3739 3E2A         MVI       A, 42
373B CD2F37       DLY0:     CALL DCM1
373E 3D           DCR       A
373F C23237       JNZ       DLY0
3742 F1           POP       PSW
3743 C9           RET

0000           END
Tally - allows Tally 2200 line printer to be used in the assembly stage of programming with Intellec 80.

Program first tests to see if the printer is ready. Then it checks the data in the output line to see if it is output or control data. If output it is sent to the Tally buffer until the buffer is full. With the control for line feed, null or buffer full, the line is printed. Page control is also given.

One interface board connected to a Tally 2200 line printer. I/O ports connected to J4 J5 on second I/O board. (Full details in Figure 1 of attachment)

Assembler and monitor

Gate 7 Bit 0 -- TEST = 1 when the printer is ready
1 BUFB = 1 if hardware is still in use
2 HDWB = 1 if hardware is still in use
3 CRST = 1 when control signal has been received
4 BUFF = 1 when buffer is full
5 VFU1 = 1 when end of page is reached
6 VFU2 = 1 for start of new page

Gate 6 8-bit ASCII data

Gate 7 Bit 0
1 SLEW = 1 to start paper feed
2 CSCD = 1 to strobe data into Tally
3 PCMD = 1 to start printing

Registers Modified: A

Assembler/Compiler Used:
8080 MDS Macro Assembler Ver 1.0

RAM Required: 0

Programmer: Miguel Angel Ainsa/S. Pick

ROM Required: 96 bytes (1 chip)

Company: ITTLS - Standard Electrica

Maximum Subroutine Nesting Level: 1

Address: Avenida America KM7.2
Madrid 27 SPAIN
Model 101 Centronics Printer Handler

Accepts character output for Model 101 Centronics printer from assembler or other source. Buffers print characters in RAM performing TTY compatible operations with control characters. Causes line to be printed upon receipt of line feed. Counts lines and keeps track of pagination. Inserts title at top of each page.

Model 101 Centronics printer with appropriate connectors and cable. Out Port 3 goes to data and strobe of Model 101, bits 0-6 are data and bit 7 is strobe. Input Port 1, bit 7 is ready status.

No additional software required.

C register to contain ASCII character to be printed.

A,C register modified. All others preserved.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Assembler/Compiler Used:</th>
</tr>
</thead>
<tbody>
<tr>
<td>A,C</td>
<td>8080 Macro Assembler</td>
</tr>
<tr>
<td>RAM Required:</td>
<td>Programmer:</td>
</tr>
<tr>
<td>11016</td>
<td>T. Riddell</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>Company:</td>
</tr>
<tr>
<td>None</td>
<td>Data Test Corporation</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td>Address:</td>
</tr>
<tr>
<td>3 + 8 bytes stack storage</td>
<td>2450 Whitman Road</td>
</tr>
</tbody>
</table>

Concord, Ca.
;REF. NO. AC6
;PROGRAM TITLE MODEL 101 CENTRONICS PRINTER HANDLER
;
;ASSEMBLER ORIENTED LINE PRINTER HANDLER

ORG 3700H ;ABS JUMP VECTORS
JMP 3C9CH ;COMMAND IN VECTOR TTY
JMP ASYPRT ;COMMAND OUT VECTOR
ORG 3712H
JMP ASYPRT ;LIST OUT VECTOR

ORG 3720H
LISTOUT EQU *
MOV A,C ;COMPLEMENT CHARACTER
CMA
MOV C,A

STATLOOP:    IN STATUS ;GET STATUS FROM PRTR
            JZ STATLOOP ;LOOP TIL READY
            MVI A,7FH ;MARK CHAR NEED
            ANA C
            OUT LPRT ;MARK CHAR OUT
            ADI 80H ;STROBE CHAR OT
            OUT LPRT
            MVI A,7FH ;MARK CHAR OFF
            ANA C
            OUT LPRT ;MARK CHAR OUT
            MOV A,C
            STA CHARSV ;RESTORE CHARACTER
            MOV C,A
            RET ;***EXIT

STATUS EQU 1 ;STATUS IN PORT
LPRT EQU 3 ;LINE PRT DATA OUT

ASYPRT EQU *
MOV A,C ;SAVE ENVIFONMENT
MOV A,C
STA CHARSV
PUSH B ;SAVE REGS
PUSH D
PUSH H
PUSH PSW

CKLC:    LDA LINCNT
ANI OFFH
JNZ CKCR ;BR IF LINE COUNT NOT ZERO
MVI C,0CH
CALL LISTOUT ;FORM FEED
3751 3ECA       MVI     A, -54
3753 320436     STA     LINCNT ; SET COUNT=-54
3756 210636     LXI     H. HEAD
3759 CD2636     CALL    PRTLINE ; PRINT HEAD-2L5T
375C CD9A37     CALL    CLRBUF
375F C36A37     JMP     CSET1

3762 3A0536     LDA     CHARSY ; GET CHARACTER
3765 FE0D       CPI     0DH    ; CARRIAGE RETURN
3767 C27337     JNZ     CKLF    ; BR IF NOT
376A 213536     XLX     H. BUF ; RESET CHAR PTR
376D 220036     SHLD    CHARPOS
3770 C38A37     JMP     ASYEND

3773 FE0A       CKLF    CPI     0AH    ; LINE FEED
3775 C28F37     JNZ     DOPRT    ; BR IF NOT
3777 218436     LXI     H. BUF+79 ; CCR TO BUF
3778 360D       MVI     M. 0DH
377D 213536     LXI     H. BUF
3778 CD2636     CALL    PRTLINE
3773 CD9A37     CALL    CLRBUF
3778 210436     LXI     H. LINCNT ; LINE COUNT + 1
3779 34         INR     M
377A F1        ASYEND: POP     PSW ; RESTORE REGISTERS
377B E1        POP     H
377C D1        POP     D
377D C1        POP     B
377E C9        RET ;********EXIT

377F 2A0036     DOPRT   LHLD    CHARPOS ; PUT CHAR IN BUF
3779 77        MOV     M. A
377A 23         INX     H
3779 220036     SHLD    CHARPOS
377A C38A37     JMP     ASYEND

379A           CLRBUF   EQU     $ ; ROUTINE ENTRY
379A 0650      MVI     B, 80 ; CHAR CNT
379C 213536     LXI     H. BUF
379F 3E0D       MVI     A, 0DH ; SPACE
37A1 77        CLRLP    MOV     M. A ; TO MEMORY
37A2 23        INX     H
37A3 05         DCR     B
37A4 C2A137     JNZ2    CLRLP ; DO WHOLE BUFFER
37A7 C9        RET ;********EXIT
3600 ORG 3600H
3600 0000   CHARPOS: DW     0
3602 00   PRTOFF: DB     0
3603 00   LINOFF: DB     0
3604 00   LINCNT: DB     0
3605 00   CHARSY: DB     0
3606     HEAD EQU $  
3606 09093830 DB '  
360A 3830204D  
360E 4143524F  
3612 20415353  
3616 454D424C  
361A 45522056  
361E 45522032  
3622 2E30  
  
3624 0D DB 0DH  
3625 00 DB 0  
3626 PRTLNE EQU $  
526 4E MOV C,M  ; GET CHARACTER (X)  
3627 CD2037 CALL LISTOUT  
362A 79 MOV A,C  
362B FE0D CPI 0DH  ; COMPARE WITH CARRIAGE RETURN  
362D CA3436 JZ LINEND  
3630 23 INX H  
3631 C32636 JMP PRTLNE  
3634 C9 LINEND: RET  ; ****EXIT  
3635 BUF: DS 80  ; LINE PRINTER BUFFER,  
0000 END  ; COMMENT
High Speed Paper Tape Reader with Stepper Motor Control

This circuit and program allow paper tape to be read at approximately 150 characters per second. The reader is assigned by monitor command "AR=1." The program uses electronic damping, under software control, of a stepper motor to increase stepping speed and precision.

Intellect 8-80
(See attached schematic).
Output port 3, input ports 1 and 3 shared with TTY interface.

8080 Monitor

Monitor Commands:

"AR=1" assigns reader device
"R" read hex tape or any other reader commands

The high speed reader will perform the same functions as TTY tape reader under command of monitor, text editor, macro assembler, or other user defined programs.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Maximum Subroutine Nesting Level:</th>
</tr>
</thead>
<tbody>
<tr>
<td>A, Flags</td>
<td>0</td>
</tr>
<tr>
<td>RAM Required:</td>
<td>Assembler/Compiler Used:</td>
</tr>
<tr>
<td>2 Bytes in Stack</td>
<td>Intellec Macro Assembler</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>Programmer:</td>
</tr>
<tr>
<td>80 Bytes (50H)</td>
<td>J. Garner</td>
</tr>
<tr>
<td></td>
<td>Company: Bell Helicopter</td>
</tr>
<tr>
<td></td>
<td>Box 482, Ft. Worth, Texas 76101</td>
</tr>
</tbody>
</table>
; REF. NO. AC7
; PROGRAM HIGH SPEED PAPER TAPE READER WITH STEPPER MOTOR CONTROL
;
;
; HIGH SPEED READER
; FOR USE WITH INTELLEO 8/80
; I/O PORTS ARE SHARED WITH
; ITY INTERFACE.
; THE STEPPER MOTOR CONTROL
; MAY BE MODIFIED FOR OTHER
; MOTORS BY CHANGING THE
; ACCELERATION AND
; DECELERATION TIMES.

3706 ORG 3706H ; OR 3709H FOR READER #2
3706 C32137 JMP 3721H
721 ORG 3721H
3721 DB01 IN 1 ; READ MOTOR STATUS
3723 2F CMA
3724 17 RAL ; SET CARRY IF NO TAPE
3725 D8 RC ; RETURN ON NO TAPE
3726 E5 PUSH H ; STORE REGISTER CONTENTS
3727 E6C0 ANI 0C0H ; MASK FOR STATUS
3729 FA6F37 JM OUT1 ; LOAD PATTERN "001"
372C CA7437 JZ OUT2 ; LOAD PATTERN "010"
372F 3E04 MVI A, 4
3731 D303 OUTM: OUT 3 ; STEP MOTOR
DELAY MACRO TDEL ; DELAY MACRO
MVI H, TDEL

CNT1: MVI L, 0CH ; LOAD INNER LOOP COUNT
CNT2: DCR L
       JNZ CNT2
       DCR H
       JNZ CNT1
ENDM ; END DELAY MACRO
       + DELAY 17H ; STEP FORWARD (ACCELERATE)

3733 2617 + MVI H, 00017H
+ 3735 2E0C +CNT1: MVI L, 0CH ; LOAD INNER LOOP COUNT
3737 2D +CNT2: DCR L
       JNZ CNT2
       DCR H
373C C23537 + JNZ CNT1
373F 2617   MVI   H.00017H
3741 2E0C   CNT1:   MVI   L.0CH   ; LOAD INNER LOOP COUNT
3743 2D   CNT2:   DCR   L
3744 C24337   JNZ   CNT2
3747 25   DCR   H
3748 C24137   JNZ   CNT1
3748 0F   RRC
374C D303   OUT   3   ; STEP BACK (DECELERATE)
374E 2606   +   DELAY   6
3750 2E0C   +CNT1:   MVI   L.0CH   ; LOAD INNER LOOP COUNT
3752 2D   +CNT2:   DCR   L
3753 C25237   +   JNZ   CNT2
3756 25   +   DCR   H
3757 C25037   +   JNZ   CNT1
375A 07   RLC
375B D303   OUT   3   ; STEP FORWARD (HOLD)
375D 260C   +   DELAY   0CH   ; ALLOW TO STOP
375F 2E0C   +CNT1:   MVI   L.0CH   ; LOAD INNER LOOP COUNT
3761 2D   +CNT2:   DCR   L
3762 C26137   +   JNZ   CNT2
3765 25   +   DCR   H
3766 C25F37   +   JNZ   CNT1
3769 DB03   IN   3   ; READ DATA
376B 2F   CMA
376C E1   POP   H   ; RESTORE REGISTERS
376D B7   ORA   A   ; CLEAR THE CARRY BIT
376E C9   RET
376F 3E09   OUT1:   MVI   A.09H   ; LOAD PATTERN "100" DRIVE
3771 C33137   JMP   OUTM
3774 3E02   OUT2:   MVI   A.2
3776 C33137   JMP   OUTM
0000   END
Terminal Editor

Procedure for controlling an ASR733 Texas Instruments terminal equipped with RDC (remote control device) option. Search a line in a file contained in cassette 1 with or without copying on cassette 2. The procedure is linked to the Intellec monitor.

Intellec 8/Mod 80

Intellec 80 monitor

The procedure is accessed via the "U" and "V" monitor commands
<Search without copying command>:=U<String>
<Search copying command>:=V<String>

String can be of any length from 1 to 86 characters, the convention for EOF is a "$" in column 1. The line typed in need not be of the same length of the line to be found. A carriage return as first character of the string means to search the previous string in the following file.

When the line is not found before an EOF the terminal prints a blank line.
When the string is found, the terminal prints an equal sign ("=")

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Assembler/Compiler Used:</th>
</tr>
</thead>
<tbody>
<tr>
<td>NA</td>
<td>PLM80</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RAM Required:</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>130</td>
<td>Enrico Massetti</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ROM Required:</th>
<th>Company:</th>
</tr>
</thead>
<tbody>
<tr>
<td>187H</td>
<td>Laboratorio Massetti</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Maximum Subroutine Nesting Level:</th>
<th>Address:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Note: monitor modified</td>
<td>20133 Milano, Via Ronchi 17</td>
</tr>
<tr>
<td></td>
<td>ITALY</td>
</tr>
</tbody>
</table>
**//REF NO. AB19 */**
**//PROGRAM TITLE TERMINAL EDITOR*/**

**/* PROCEDURE FOR CONTROLLING AN ASR733 TEXAS TERMINAL EQUIPED
** WITH RDC (REMOTE DEVICE CONTROL) OPTION. */**

00001 1 DECLARE (CAR, BLOCK) (86) BYTE;
00002 1 CR LITERALLY '13';
00003 1 /* LINKAGE TO 8080 MONITOR. */
00004 1 DECLARE CO LITERALLY '3C32H';
00005 1 TTYOUT: PROCEDURE (CHAR);
00006 2 DECLARE CHAR ADDRESS;
00007 2 CHAR=SHL (CHAR, 8);
00008 2 GO TO CO;
00009 2 END TTYOUT;
00010 1 DECLARE CI LITERALLY '3C76H';
00011 1 TTYIN: PROCEDURE BYTE;
00012 2 GO TO CI;
00013 2 END TTYIN;
00014 1 DLEOUT: PROCEDURE (CARATTERE);
00015 2 DECLARE CARATTERE BYTE;
00016 2 CALL TTYOUT(16); /* DLE */
00017 2 CALL TTYOUT(CARATTERE); /* CARATTER */
00018 2 RETURN;
00019 2 END DLEOUT;
00020 1 READS$BLOCK: PROCEDURE (INBLOCK);
00021 2 DECLARE INBLOCK ADDRESS,
00022 2 BLOCK BASED INBLOCK BYTE,
00023 2 N BYTE,
00024 2 TEMP BYTE;
00025 2 IF (TEMP$= TTYIN)=CR THEN RETURN;
00026 2 ELSE BLOCK(0)= TEMP;
00027 2 DO N = 1 TO 85;
00028 2 BLOCK (N)= TTYIN;
00029 2 IF BLOCK(N)=CR THEN RETURN;
00030 3 END;
00031 2 RETURN;
00032 2 END READS$BLOCK;
00033 1 /*
00034 1 */
00035 1 ON$BLOCK: PROCEDURE BYTE;
00036 2 CALL DLEOUT(55); /* DLE 7 = BLOCK FWD */
00037 2 CALL READS$BLOCK (.BLOCK);
00038 2 IF BLOCK = 36 THEN DO; /* 36=$ END OF FILE */
00039 2 CALL TTYOUT (20); /* DC4 = RECORD OFF */
00040 2 CALL DLEOUT (57); /* DLE 9 = PRINTER ON */
00041 3 RETURN 0;
00042 3 END;
00043 2 ELSE RETURN OFFH;
00044 2 END ON$BLOCK;
00045 1 PREDISP: PROCEDURE BYTE;
00046 2 DECLARE STATUS BYTE;
00047 2 /* CASSETTE 1 IN PLAYBACK 2 IN RECORD */
00048 2 CALL DLEOUT (48); /* DLE 0 = PRINTER OFF */
00049 2 CALL DLEOUT (54); /* DLE 6 = CASS1 PLY, CASS2 RECORD */
00050 2 CALL DLEOUT (60); /* DLE < = STATUS REQUEST */
00051 2 STATUS = TTYIN; /* VERIFY TERMINAL STATUS */
00052 2 VERIFICA$STATU: IF (STATUS AND 1FH)< 1FH THEN DO; /* STATUS NOT OK */
00053 2 CALL DLEOUT (57); /* DLE 9 = PRINTER ON */
00054 2 RETURN 0;
00061 3  END;
00062 2  ELSE RETURN OFFH;
00063 2  END PREDISP;
00064 1  /* THE PROCEDURE RETURNS 0 IF THE TERMINAL IS NOT PREDISPOSED, 1 IF ALL OK*/
00065 1  /* PROCEDURE FOR CONFRONTING THE CAR AND BLOCKS*/
00066 1  BLOCK$CONFIR: PROCEDURE BYTE;
00067 2  DECLARE N BYTE;
00068 2  DU N=0 TO 85;
00069 2  IF CAR(N)<>BLOCK(N) THEN DO;
00070 3  IF CAR(N)=CR THEN RETURN 1;
00071 4  ELSE RETURN 0;/*BLOCKS DIFFERENT*/
00072 4  END;
00073 3  ELSE IF CAR(N)=CR THEN RETURN OFFH;/*IDENTICAL BLOCK*/
00074 3  END;
00075 2  RETURN OFFH;/*IDENTICAL BLOCKS*/
00076 2  END BLOCK$CONFIR;
00077 1  /* THE PROCEDURE RETURNS TO THE CALLING POINT THE FOLLOWING VALUES:
00078 1  0 = BLOCK NOT EQUALS
00079 1  1 = BLOCK CAR IS A SUBSET OF BLOCK BLOCK
00080 1  OFFH = BLOCKS IDENTICAL */
00081 1  /* PROCEDURE FOR CONTROLLING AN ASR 733 TEXAS TERMINAL EQUIPED WITH RDC
00082 1  OPTION:
00083 1  SEARCH OF A LINE IN FILE CONTAINED INTO CASSETTE 1 WITH OR WITHOUT
00084 1  PARAMETER COPIA=0 : SEARCH WITHOUT COPYING; PARAMETER COPIA=1 : SEARCH
00085 1  COPYING. */
00086 1  RICERCAS$BLOCK: PROCEDURE (COPIA);
00087 2  DECLARE (COPIA,TEMP) BYTE;
00088 2  ENTER$BLOCK: CALL READ$BLOCK (.CAR);
00089 2  /* THE LINE TO BE FOUND COMES FROM THE KEYBOARD AND IS PUT IN THE MEMORY
00090 2  IN THE CAR POSITIONS. THE LINE CAN BE OF ANY LENGTH FROM 1 86 CHARACTERS
00091 2  THE CONVENTION FOR AND OF FILE IS THE ASCII CHARACTER $ IN COLUMN 1.
00092 2  WHEN THIS CHARACTER IS FOUND IN COLUMN 1 THE TERMINAL PRINTS <> AND THE
00093 2  PROGRAM RETURNS TO THE CALLING POINT.
00094 2  THE LINE TYPED IN NEEDS NOT TO BE OF THE SAME LENGTH OF THE LINE TO BE FOUND
00095 2  FOR EXAMPLE:
00096 2  --------------------------------------------------------
00097 2  FIRST,SECOND,THIRD
00098 2  (THIS IS THE LINE TO BE FOUND)
00099 2  FIRST
00100 2  (THIS IS A VALID INDICATION TO FIND THAT LINE)
00101 2  --------------------------------------------------------
00102 2  */
00103 2  IF PREDISP=0 THEN RETURN; /* TERMINAL NOT PREDISPOSED*/
00104 2  IF COPIA =1 THEN CALL TIOUT(18); /*DC2=RECORD ON*/
00105 2  LOOP: IF ONE$BLOCK=0 THEN DO; /*BLOCK NOT FOUND.FOUND EOF ($) PRINTS<> */
00106 2  CALL TIOUT (60); /* < */
00107 3  CALL TIOUT (62); /* > */
00108 3  RETURN; END;
00109 2  /* BLOCK FOUND. START CONFRONT*/
00110 2  IF (TEMP=BLOCKS$CONFIR)=0 THEN GO TO LOOP; /*BLOCKS DIFFERENT*/
00111 2  CALL TIOUT (20); /*DC4=RECORD OFF*/
00112 2  CALL DLEUUT (41); /*DLE9=PRINTER ON*/
00113 2  CALL TIOUT (61); /*=*/
00114 2  RETURN;
00115 2  END RICERCAS$BLOCK;
00116 1  EDF
NO DGRAM ERRORS
Intellec 8/MOD 80-Silent 700 Interface

This note describes an interface between the Intellec 8/Mod 80 Microcomputer Development System and the Texas Instruments "Silent 700" Electronic Data Terminal.

The interface consists of an Intellec/Silent 700 adapter. In addition, the "Silent 700" terminal and Intellec 8/Mod 80 must be modified. The adapter and equipment modifications are described in this note.

The interface was designed for use with the basic Intellec configuration (imm8-84A) and a Model 733ASR terminal equipped with the 1200 Baud Transmission option and either the Automatic Device Control (ADC) option or Remote Device Control (RDC) option.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Assembler/Compiler Used:</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM Required:</td>
<td>Programmer: Phil Jensen</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>Company:</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td>Address: 6530 LBJ Freeway, #178 Dallas, Texas 75240</td>
</tr>
</tbody>
</table>
Interrupt Service Routine

Handles multiple-level interrupts, saving all registers and flags and outputting the status of the current interrupt to an external status latch. See additional sheet.

Priority Interrupt Control Unit (8214) with status latch inputs connected to appropriate output port (#255 in sample program).

INTX and ROUTX for each interrupt level to be handled, plus necessary routines to service the interrupts.

A RESTART instruction is forced onto the data bus by the interrupt control unit.

The status of the current interrupt routine is output on bits 0, 1, and 2 of output port 255. The service routine is executed and all registers and flags from the calling program are save and restored after the interrupt routine is complete.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Maximum Subroutine Nesting Level:</th>
</tr>
</thead>
<tbody>
<tr>
<td>All registers are saved</td>
<td>12 bytes per interrupt level</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RAM Required:</th>
<th>Assembler/Compiler Used:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 byte plus stack</td>
<td>8080 Macro Assembler V4.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ROM Required:</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>28 bytes + 21 bytes per interrupt level.</td>
<td>Donald E. Shorter</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Company:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Telcom, Inc.</td>
<td></td>
</tr>
<tr>
<td>8027 Leesburg Pike</td>
<td></td>
</tr>
</tbody>
</table>

Vienna, Va. 22180 4-130
Interrupt Service Routine

This routine is designed to be used with an external hardware interrupt control unit and interrupt status latch. A type 8214 Priority Interrupt Control Unit may be used to generate the appropriate RESTART instructions; the current status latch pins should be connected to bits 0, 1, and 2 of output port 255.

With any priority interrupt situation, a problem arises when one interrupt service routine is interrupted by a higher priority signal. When the higher priority routine begins, it must output its status to the current status latch; after it is finished, control is transferred to the lower priority routine which was interrupted. This requires that the lower priority status must be sent to the status latch so that any higher priority interrupt may take control again.

This routine uses one byte of RAM to store any current status (this byte should be set to the lowest priority at the beginning of the main program). When an interrupt occurs, this previous status is pushed into the stack and the current status is output to the status latch. At the end of the routine, the previous status is popped off the stack and output to the latch just before the program goes into the routine to restore the registers and return to the calling program. In this way, the status latch will always contain the current status, even if several interrupts are stacked one on top of another.

The interrupt service routine saves the contents of all registers and flags and restores them at the end of the service program. Each level of interrupt requires 12 bytes of RAM for stack operations, plus whatever is required to actually service the interrupt. Interrupts may be nested as far as the available stack will permit.

In the sample program, INTX is the routine which resides in an eight-byte section of the first 64 bytes of system memory; "x" is the number, 0-7, of the interrupt routine. One INTX is required for each interrupt level desired. INTX saves the contents of all registers and flags and then jumps to the service routine.

ROUTX is the service routine which saves the previous status in the stack and outputs the current status to the external latch. "x" is the number, 0-7, of the interrupt routine. The interrupt flip-flop is enabled and any instructions necessary to service the interrupt are executed. Then
ROUTX disables the interrupt flip-flop and restores the previous status to RAM and also to the status latch on port 255. One ROUTX is required for each interrupt level desired.

SAVE is the subroutine which performs the actual status updating and outputting operations. Only one SAVE is required.

RSTOR is the subroutine which restores the previous status to the latch and the RAM location (STATU). Only one RSTOR is required.

FINIS is the routine which restores all registers and flags to their original condition and returns to the calling program. One FINIS is required.

Test Program

A test program was developed to test the interrupt service routine; it requires the Monitor, Version 2.0 in the Intellec 8/Mod 80.

The six INTX routines (INT2 - INT7) are loaded in RAM starting at location 10H (the first 16 bytes are required for operation of the Monitor and cannot be used). The main program is then loaded in RAM starting at location 200H.

The main program continuously prints a message on the TTY; at any time during this message, a RST2 - RST7 may be entered on the data bus of the Intellec by setting up the appropriate code on the data switches and pressing the INT switch. When it receives the interrupt, the program will begin to print the message corresponding to the number of that interrupt. Another interrupt may then be inserted and the corresponding message will be printed. When a message has been completely sent, the program will revert back to the remainder of the previous message, etc., until it is back to the main program message.

A delay routine was incorporated to slow the printing so that the operator has time to set up and generate another interrupt signal. As each interrupt routine is active, the current status appears on the lower three bits of the front panel lights connected to output port 255. A RST1 will cause the program to be transferred back to the system Monitor.

Donald E. Shorter
Telcom, Inc.
8027 Leesburg Pike
Vienna, Virginia 22180
Interrupt Handler Re-entrant

On processor receipt of an interrupt instruction (RST 0-7), this program saves the machine state and previous interrupt level on the stack, transmits the new service level to the interrupt control unit (ICU), executes a subroutine corresponding to the level interrupt received, then restores the machine and ICU to their pre-interrupt state before resuming executing the interrupted program.

Interrupt circuitry; test routine assumes TTY on 0 and 1; test interrupts were from Intellec 8/80 front panel.

Service programs appropriate to the various interrupt levels expected background initialization; TTY test routine uses "CO" and "CRLF" in monitor or equiv.

Restart instructions, RST 0-7, either from ICU or programmed.

Interrupt routine causes branches to routines according to pointers: SERV1, SERV2, etc. supplied by user for his system. Test program types the level of interrupt received on the assigned monitor console device. See attached sheet.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Assembler/Compiler Used:</th>
</tr>
</thead>
<tbody>
<tr>
<td>None; state restored on exit</td>
<td>Intellec 8 Macro Assembler</td>
</tr>
<tr>
<td>RAM Required: 6 byte variables, plus 10 bytes/nesting level on stack, to stack limit</td>
<td>Programmer:</td>
</tr>
<tr>
<td>ROM Required: bytes 0H through 3FH, plus 44H bytes elsewhere = 84H</td>
<td>John M. Mills, Lecturer</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level: Fully re-entrant; stack limit</td>
<td>Company:</td>
</tr>
<tr>
<td></td>
<td>University of Wisconsin</td>
</tr>
<tr>
<td></td>
<td>Address: Dept. of Mechanical Engr.</td>
</tr>
<tr>
<td></td>
<td>Madison, Wisconsin 53706</td>
</tr>
</tbody>
</table>
8008 DISASSEMBLER

This program is used to obtain an assembly language listing of a machine coded program in memory.

INTELLEC 8 with console I/O device.
12K of RAM Memory (ORG commands can be changed for less).

INTEL 8008 MONITOR program ver 2.1. (Monitor subroutine addresses are listed at the start of the program for updating to other monitor versions.)

The Monitor 'G' command is used to enter the program with a response of a carriage return and line feed. The low address and the high address is then entered in hexadecimal format, separated by a comma, and followed by a carriage return.

A listing on the console output device of the hexadecimal address, the hexadecimal machine instruction, and the assembly language instruction.

Invalid machine instruction codes are listed as hexadecimal numbers.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Assembler/Compiler Used:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>INTELLEC 8 MACRO ASSEMBLER</td>
</tr>
<tr>
<td>RAM Required:</td>
<td>Programmer: Hal King</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>Company: Brooks Research &amp; Mfg</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td>Address: 5612 Brighton 64130 Kansas City, Missouri</td>
</tr>
</tbody>
</table>
8080 DIS-ASSEMBLER (SEE NOTE BELOW)

This program inputs an ISIS-II file in HEX format and generates a symbolic assembly language program suitable for editing and/or assembling.

MDS-800, Dual Diskette Drives
32K to operate
64K to compile

PLM-80 Diskette Resident Compiler (to compile the source)
ISIS-II (to execute the program)

ISIS-II file (disk file, :HR:, etc.) in Intel's HEX format
(use OBJHEX if it's originally in ISIS-II, 8080 object module format)

REVISED 10/78
by Gary Carleton
Intel Corp.

Assembly language source program

NOTE: Program source is in PLM-80 Resident compiler form.

Available on diskette only
DISASM (8080 Disassembler)

DISASM is intended as a software development and debugging aid. Operating on resident object code, it produces an assembly language equivalent which is printed on a TTY terminal. In this present form, the program starts at a given memory address and steps sequentially through memory until manually halted.

TTY on parts 0 and 1 (as Intellec system configuration)

Intellec 8/80 Monitor console output routine (or equivalent)

See reference #AB 154, Page 4-725 for SYMBOL TABLE INERTER

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM Required: 2 Bytes (pluss Stack)</td>
<td>Company: General Electric</td>
</tr>
<tr>
<td>ROM Required: 791 Bytes</td>
<td>Address: Bldg 50-1</td>
</tr>
<tr>
<td></td>
<td>2901 East Lake Road</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level: 2 levels</td>
<td>4 bytes</td>
</tr>
<tr>
<td>Assembler/Compiler Used: 8080 Macro</td>
<td>City: Erie</td>
</tr>
<tr>
<td>Assembler, V 3.0</td>
<td>State: PA 16531</td>
</tr>
</tbody>
</table>
; REF. NO. AB22
; PROGRAM NAME DISASM (8080 DISASSEMBLER)

1800 ORG 1800H
3809 EQU 3809H

1B00 2A081D RBYTE: LHLD PC
1B03 7E MOV A,M
1B04 23 INX H
1B05 22081D SHLD PC
1B08 C9 RET

1B09 3C RGPRNT: INR A
1B0A E607 ANI 07
1B0C FE06 CPI 06
30E DA131B JC RGP1
1B11 C603 ADI 03
1B13 FE05 RGP1: CPI 05
1B15 DA131B JC RGP2
1B18 C602 ADI 02
1B1A C641 RGP2: ADI 41H
1B1C 4F MOV C,A
1B1D C30938 JMP CO

1B20 47 DECODE: MOV B,A
1B21 E6F0 ANI 0F0H
1B23 0F RRC
1B24 0F RRC
1B25 0F RRC
1B26 0F RRC
1B27 C690 ADI 90H
1B29 27 DAA
1B2A CE40 ACI 40H
1B2C 27 DAA
1B2D 4F MOV C,A
1B2E CD0938 CALL CO
1B31 78 MOV A,B
1B32 E6F0 ANI 0F0H
1B34 C690 ADI 90H
1B36 27 DAA
1B37 CE40 ACI 40H
1B39 27 DAA
1B3A 4F MOV C,A
1B3B C30938 JMP CO
<table>
<thead>
<tr>
<th>Line</th>
<th>Opcode</th>
<th>Label</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1B3E</td>
<td>0604</td>
<td>PRINT</td>
<td>MVI B, 4</td>
</tr>
<tr>
<td>1B40</td>
<td>4E</td>
<td>P1:</td>
<td>MOV C, M</td>
</tr>
<tr>
<td>1B41</td>
<td>CD0938</td>
<td>CALL</td>
<td>CO</td>
</tr>
<tr>
<td>1B44</td>
<td>23</td>
<td>INX</td>
<td>H</td>
</tr>
<tr>
<td>1B45</td>
<td>05</td>
<td>DCR</td>
<td>B</td>
</tr>
<tr>
<td>1B46</td>
<td>C2401B</td>
<td>JNZ</td>
<td>P1</td>
</tr>
<tr>
<td>1B49</td>
<td>0E20</td>
<td>MVI</td>
<td>C, 20H</td>
</tr>
<tr>
<td>1B4B</td>
<td>C30938</td>
<td>JMP</td>
<td>CO</td>
</tr>
<tr>
<td>1B4E</td>
<td>7A</td>
<td>XTRACT</td>
<td>MOV A, D</td>
</tr>
<tr>
<td>1B4F</td>
<td>E638</td>
<td>ANI</td>
<td>38H</td>
</tr>
<tr>
<td>1B51</td>
<td>0F</td>
<td>RRC</td>
<td></td>
</tr>
<tr>
<td>1B52</td>
<td>0F</td>
<td>RRC</td>
<td></td>
</tr>
<tr>
<td>1B53</td>
<td>0F</td>
<td>RRC</td>
<td></td>
</tr>
<tr>
<td>1B54</td>
<td>C9</td>
<td>RET</td>
<td></td>
</tr>
<tr>
<td>1B55</td>
<td>CD04E1B</td>
<td>CCPRNT</td>
<td>CALL XTRACT</td>
</tr>
<tr>
<td>1B58</td>
<td>87</td>
<td>ADD</td>
<td>A</td>
</tr>
<tr>
<td>1B59</td>
<td>4F</td>
<td>MOV</td>
<td>C, A</td>
</tr>
<tr>
<td>1B5A</td>
<td>21881E</td>
<td>LXI</td>
<td>H, CCODE</td>
</tr>
<tr>
<td>1B5D</td>
<td>09</td>
<td>DAD</td>
<td>B</td>
</tr>
<tr>
<td>1B5E</td>
<td>4E</td>
<td>MOV</td>
<td>C, M</td>
</tr>
<tr>
<td>1B5F</td>
<td>CD0938</td>
<td>CALL</td>
<td>CO</td>
</tr>
<tr>
<td>1B62</td>
<td>23</td>
<td>INX</td>
<td>H</td>
</tr>
<tr>
<td>1B63</td>
<td>4E</td>
<td>MOV</td>
<td>C, M</td>
</tr>
<tr>
<td>1B64</td>
<td>CD0938</td>
<td>CALL</td>
<td>CO</td>
</tr>
<tr>
<td>1B67</td>
<td>0E20</td>
<td>MVI</td>
<td>C, 20H</td>
</tr>
<tr>
<td>1B69</td>
<td>CD0938</td>
<td>CALL</td>
<td>CO</td>
</tr>
<tr>
<td>1B6C</td>
<td>C30938</td>
<td>JMP</td>
<td>CO</td>
</tr>
<tr>
<td>1B6F</td>
<td>CD04E1B</td>
<td>RPRNT</td>
<td>CALL XTRACT</td>
</tr>
<tr>
<td>1B72</td>
<td>E606</td>
<td>ANI</td>
<td>06</td>
</tr>
<tr>
<td>1B74</td>
<td>FE06</td>
<td>CPI</td>
<td>06</td>
</tr>
<tr>
<td>1B76</td>
<td>C2091B</td>
<td>JNZ</td>
<td>RGPRNT</td>
</tr>
<tr>
<td>1B79</td>
<td>0E53</td>
<td>MVI</td>
<td>C, 53H</td>
</tr>
<tr>
<td>1B7B</td>
<td>CD0938</td>
<td>CALL</td>
<td>CO</td>
</tr>
<tr>
<td>1B7E</td>
<td>0E50</td>
<td>MVI</td>
<td>C, 50H</td>
</tr>
<tr>
<td>1B80</td>
<td>C30938</td>
<td>JMP</td>
<td>CO</td>
</tr>
<tr>
<td>1B83</td>
<td>0E0D</td>
<td>DISASM</td>
<td>MVI C, 0DH</td>
</tr>
<tr>
<td>1B85</td>
<td>CD0938</td>
<td>CALL</td>
<td>CO</td>
</tr>
<tr>
<td>1B88</td>
<td>0E0A</td>
<td>MVI</td>
<td>C, 0AH</td>
</tr>
<tr>
<td>1B8A</td>
<td>CD0938</td>
<td>CALL</td>
<td>CO</td>
</tr>
<tr>
<td>1B8D</td>
<td>2A081D</td>
<td>LHLD</td>
<td>PC</td>
</tr>
<tr>
<td>1B90</td>
<td>7C</td>
<td>MOV</td>
<td>A, H</td>
</tr>
<tr>
<td>1B91</td>
<td>CD201B</td>
<td>CALL</td>
<td>DECODE</td>
</tr>
<tr>
<td>1B94</td>
<td>7D</td>
<td>MOV</td>
<td>A, L</td>
</tr>
<tr>
<td>1B95</td>
<td>CD201B</td>
<td>CALL</td>
<td>DECODE</td>
</tr>
<tr>
<td>1B98</td>
<td>0E20</td>
<td>MVI</td>
<td>C, 20H</td>
</tr>
<tr>
<td>1B9A</td>
<td>CD0938</td>
<td>CALL</td>
<td>CO</td>
</tr>
</tbody>
</table>
1B9D CD0938 CALL CO
1BA0 CD001B CALL RDBYTE
1BA3 57 MOV D, A
1BA4 210A1D LXI H, TABLE
1BA7 011100 LXI B, 11H
1BA9 BE D1: CMP M
1BBA CAF01C JZ TG1
1BB0 23 INX H
1BB1 0D DCR C
1BB6 C2A1B JNZ D1
1BB3 0E0A MVI C, 0AH
1BB5 BE CMP M
1BB6 BE D2: CMP M
1BB7 CAE61C JZ TG2
1BB8 23 INX H
1BB9 0D DCR C
1BBC C2B51B JNZ D2
1BBF 0E06 MVI C, 6
1BC1 BE D3: CMP M
1BC2 CAC1C JZ TG3
1BC5 23 INX H
1C6 0D DCR C
1BC7 C2C1B JNZ D3
1BCA 6C0 ANI 0C0H
1BCC FE40 CPI 40H
1BCE CAB01C JZ MG0
1BD0 FE80 CPI 80H
1BD3 0AA1C JZ MG1
1BD6 7A MOV A, D
1BD7 E6C7 ANI 0C7H
1BD9 D604 SUI 04
1BDA 9A92C1C JZ MG2
1BDE 3D DCR A
1BDF CAB81C JZ MG3
1BE2 3D DCR A
1BE3 CA781C JZ MG4
1BE6 7A MOV A, D
1BE7 E6C0 ANI 0C0H
1BE9 CA4C1C JZ MG5
1BEC 7A MOV A, D
1BED E6C7 ANI 0C7H
1BEF D600 SUI 0C0H
1BF1 CA411C JZ MG6
1BF4 D602 SUI 02
1BF6 CA361C JZ MG7
1BF9 D602 SUI 02
1FB0 CA2B1C JZ MG8
1FE D603 SUI 03
1C00 CA1F1C JZ MG9
1C03 7A MOV A, D
1C04 E607  ANI  07
1C06 4F  MOV  C, A
1C07 21FF1D  LXI  H, PPOP-1
1C08 09  DAD  B
1C0B CD3E1B  CALL  PRINT
1C0E CD4E1B  CALL  XTRACT
1C11 FE06  CPI  06
1C13 C29B1C  JNZ  D6
1C16 21FC1D  LXI  H, PPSW
1C19 CD3E1B  CALL  PRINT
1C1C C3831B  JMP  DISASM
1C1F 21F81D  LG9:  LXI  H, PRST
1C22 CD3E1B  CALL  PRINT
1C25 CD201B  CALL  DECODE
1C28 C3831B  JMP  DISASM
1C2B 0E43  MG8:  MVI  C, 43H
1C2D CD0938  CALL  C0
1C30 CD551B  CALL  CCRPRNT
1C33 C3D51C  JMP  D7
1C36 0E4A  MG7:  MVI  C, 4AH
1C38 CD0938  CALL  C0
1C3B CD551B  CALL  CCRPRNT
1C3E C3D51C  JMP  D7
1C41 0E52  MG6:  MVI  C, 52H
1C43 CD0938  CALL  C0
1C46 CD551B  CALL  CCRPRNT
1C49 C3831B  JMP  DISASM
1C4C 21E01D  MG5:  LXI  H, PLXI
1C4F 7A  MOV  A, D
1C50 E60F  ANI  0FH
1C52 3D  DCR  A
1C53 CA6A1C  JZ  MG51
1C56 FE04  CPI  04
1C58 DA5D1C  JC  D4
1C5B D605  SUI  05
1C5D 87  D4:  ADD  A
1C5E 87  ADD  A
1C5F 4F  MOV  C, A
1C60 09  DAD  B
1C61 CD3E1B  CALL  PRINT
1C64 CD6F1B  CALL  RPRPRNT
1C67 C3831B  JMP  DISASM
1C6A CD3E1B  MG51:  CALL  PRINT
1C6D CD6F1B  CALL  RPRPRNT
1C70 0E2C  MVI  C, 2CH
1C72 CD0938  CALL  C0
1C75 C3D51C  JMP  D7
1C78 21DC1D  MG4:  LXI  H, PMVI
1C7B CD3E1B  CALL  PRINT
1C7E CD4E1B  CALL  XTRACT
1C81 CD091B CALL RGPRNT
1C84 0E2C MVI C, 2CH
1C86 CD0938 CALL C0
1C89 C3F11C JMP D8
1C8C 21D81D MG3: LXI H, PDCR
1C8F C3951C JMP D5
1C92 21D41D MG2: LXI H, PINR
1C95 CD3E1B D5: CALL PRINT
1C98 CD4E1B CALL XTRACT
1C9B CD091B D6: CALL RGPRNT
1C9E C3831B JMP DISASM
1CA1 7A MG1: MOV A, D
1CA2 E638 ANI 38H
1CA4 0F RRC
1CA5 4F MOV C, A
1CA6 21B41D LXI H, PADD
1CA9 09 DAD B
1CAE CD3E1B CALL PRINT
1CAD C3C11C JMP D9
1CB0 21B01D MG0: LXI H, PMOV
1CB3 CD3E1B CALL PRINT
1CB6 CD4E1B CALL XTRACT
1CB9 CD091B CALL RGPRNT
1CBC 0E2C MVI C, 2CH
1CBE CD0938 CALL C0
1CC1 7A D9: MOV A, D
1CC2 E607 ANI 07
1CC4 CD091B CALL RGPRNT
1CC7 C3831B JMP DISASM
1CCA 79 TG3: MOV A, C
1CCB 87 ADD A
1CCC 87 ADD A
1CCD 4F MOV C, A
1CEE 21941D LXI H, TAB3-4
1CD1 09 DAD B
1CD2 CD3E1B CALL PRINT
1CD5 CD001B D7: CALL RDBYTE
1CD8 57 MOV D, A
1CD9 CD001B CALL RDBYTE
1CDC CD201B CALL DECODE
1CDF 7A MOV A, D
1CE0 CD201B CALL DECODE
1CE3 C3831B JMP DISASM
1CE6 79 TG2: MOV A, C
1CE7 87 ADD A
1CE8 87 ADD A
1CE9 4F MOV C, A
1CEA 216C1D LXI H, TAB2-4
1CED 09 DAD B
1CEE CD3E1B CALL PRINT
1CF1 CD001B  DB: CALL RDBYTE
1CF4 CD201B  CALL DECODE
1CF7 C3831B  JMP DISASM
1CFA 79      TG1: MOV A, C
1CFB 87      ADD A
1CFC 87      ADD A
1CFD 4F      MOV C, A
1CFE 21271D  LXI H, TAB1-4
1D01 09      DAD B
1D02 CD3E1B  CALL PRINT
1D05 C3831B  JMP DISASM
1D08        PC: DS 2
1D0A 00070F17 TABLE: DB 000H, 007H, 00FH, 017H
1D0E 1F272F37 DB 01FH, 027H, 02FH, 037H
1D12 3F76C9E3 DB 03FH, 076H, 0C9H, 0E3H
1D16 E9E8F9  DB 0E9H, 0EBH, 0F3H, 0F9H
1D1A FBC6CED3 DB 0FBH, 0C6H, 0CEH, 0D3H
1D1E D6DBEE6 DB 0D6H, 0DBH, 0DEH, 0E6H
1D22 EEFF22  DB 0EEH, 0FFH, 0FH, 022H
1D26 2A323AC3 DB 02AH, 032H, 03AH, 03CH
1D2A CD      DB 0CDH
1D2B 45492020 TAB1: DB 'EI ', 'SPHL', 'DI ', 'XCHG'
1D2F 20535048
1D33 4C44920
1D37 20584348
1D3B 47

1D3C 5043484C DB 'PCHL', 'XTHL', 'RET ', 'HLT '
1D40 5854484C
1D44 52455420
1D48 484C5420

1D4C 434D4320 DB 'CMC ', 'STC ', 'CMA ', 'DAA '
1D50 53544320
1D54 434D4120
1D58 44414120

1D5C 52415220 DB 'RAR ', 'RAL ', 'RRC ', 'RLC '
1D60 52414C20
1D64 52524320
1D68 524C4320

1D6C 4E4F5020 DB 'NOP '
1D70 43504920 TAB2: DB 'CPI ', 'ORI ', 'XRI ', 'ANI '
1D74 4F524920
1D78  58524920
1D7C  41E4920

1D80  53424920  DB  'SBI ', 'IN ', 'SUI ', 'OUT '
1D84  49E2020
1D88  53554920
1D8C  4F555420

1D90  41434920  DB  'ACI ', 'ADI '
1D94  41444920
1D98  43414C4C  TAB3:  DB  'CALL ', 'JMP ', 'LDA ', 'STA '
1D9C  4A4D5020
1DA0  4C44120
1DA4  53544120

1DA8  4C484C44  DB  'LHLD ', 'SHLD '
1DAC  53484C44

1DB0  4D4F5620  PMOV:  DB  'MOV '
1DB4  41444220  PADD:  DB  'ADD ', 'ADC ', 'SUB ', 'SBB '
1DB8  41444320
1DBC  53554220
1DC0  53424220

1DC4  41E4120  DB  'ANA ', 'XRA ', 'ORA ', 'CMP '
1DC8  58524120
1DCC  4F524120
1DD0  434D5020

1DD4  49E5220  PINR:  DB  'INR '
1DD8  44435220  PDCR:  DB  'DCR '
1DDE  4D564920  PMVI:  DB  'MVI '
1DE0  4C584920  PLXI:  DB  'LXI ', 'STAX ', 'INX ', 'DAD '
1DE4  53544158
1DE8  49E5820
1DEC  44414420

1DF0  4C444158  DB  'LDAX ', 'DCX '
1DF4  44435820

1FF8  52535420  PRST:  DB  'RST '
1FFC  50535720  PFSW:  DB  'PSW '
1E00  504F5020  PPOP:  DB  'POP ', 'PUSH '
1E04  50555348
CODE: DB 'NZ', 'Z', 'NC', 'C'
1E8  4E5A5A20
1E0C 4E434320

DB 'PO', 'PE', 'P', 'M'
1E10  504F5045
1E14 50204D20

END

0000
BINLB -- 8080 System Loader

Loads Hex Format Paper Tape produced by Macro-Assembler on GE Time Sharing into 8080 system. Also provides TTY input and output subroutines. BINLD can also produce a binary dump of itself for bootstrap loading.

TTY -- This program uses a serial TTY interface and a hardware timer. This can be easily changed.

NONE

Hex format paper tape

Loaded program

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Maximum Subroutine Nesting Level:</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM Required:</td>
<td>N/A</td>
</tr>
<tr>
<td>256 Bytes</td>
<td></td>
</tr>
<tr>
<td>ROM Required:</td>
<td>Assembler/Compiler Used:</td>
</tr>
<tr>
<td>NONE</td>
<td>MACRO/8080</td>
</tr>
<tr>
<td></td>
<td>Programmer:</td>
</tr>
<tr>
<td></td>
<td>George Miler</td>
</tr>
<tr>
<td>Company:</td>
<td>MILERTRONICS</td>
</tr>
<tr>
<td></td>
<td>525-A Airport Rd.</td>
</tr>
<tr>
<td></td>
<td>Greenville, S.C. 29607</td>
</tr>
</tbody>
</table>
; REF. NO. AB23.
; PROGRAM NAME 8080 SYSTEM LOADER
; 
; 
; 4,22,75
; WRITTEN BY
; MILERTRONICS DIV. 
; GEORGE MILER INC.
; 525-A AIRPORT RD.
; GREENVILLE, S. C. 29607
; (803)242-9232
; BINLD LOADS INTEL HEX FORMAT
; OBJECT TAPE ASSEMBLER
; REG B USED TO GENERATE CHARACTER 
; REG C USED TO FOR BIT COUNT 8SERIAL I/O)
; REG D USED FOR TEMP STORAGE
; REG E USED FOR CHARACTER COUNT
; 
; 
0E00 ORG 0E00H

0E00 3EC9 MVI A, 0C9H ; STORE A RET AT THE TTY
0E02 321800 STA 18H ; INTERRUPT ADDRESS-18H
0E05 F3 START: DI ; DISABLE INTERRUPT
0E06 31100F LXI SP, 0F10H ; LOAD STACK POINTER

; BEGINNING OF MAIN BINLD PROGRAM

0E09 AF BINLD: XRA A ; CLEAR SUM
0E0A 32D10E STA CHECK ; CHECK LOCATION.
0E0D CDA80E CALL TTY ; GET CHARACTER
0E10 FE3A CPI 3AH ; CHECK FOR COLON
0E12 C2090E JNZ BINLD ; GET NEXT CHARACTER IN NOT :
0E15 CDA80E CALL TTY ; GET FIRST LENGTH HEX
0E18 50 MOV D, B ; SAVE FIRST HEX
0E19 CDA80E CALL TTY ; GET SECONO LENGTH
0E1C CD600E CALL PACK ; PACK LENGTH
0E1F B7 ORA A ; STOP IF END
0E20 CAC90E JZ STOP
0E23 58 MOV E, B ; STORE LENGTH IN REG E
0E24 CDA80E CALL TTY ; GET 4 HEX ADDRESSES
0E27 50 MOV D, B
0E28 CDA80E CALL TTY ; GET SECOND HEX
0E2B CD600E CALL PACK ; AND PACK
0E2E 60 MOV H, B ; MOVE LSB TO H
0E2F CDA80E CALL TTY ; GET THIRD HEX
0E32 50 MOV D, B ;
; GET LAST ADDRESS HEX
0E33 CDA80E CALL TTY
0E36 CD600E CALL PACK
0E39 68 MOV L,B
0E3A CDA80E CALL TTY
0E3D CDA80E CALL TTY

; PACK AND STORE DATA IN MEMORY
0E40 CDA80E STORE: CALL TTY
0E43 50 MOV D,B
0E44 CDA80E CALL TTY
0E47 CD600E CALL PACK
0E4A 70 MOV M,B
0E4B 23 INX H
0E4C 1D DCR E
0E4D C2400E JNZ STORE
0E50 CDA80E CALL TTY
0E53 50 MOV D,B
0E54 CDA80E CALL TTY
0E57 CD600E CALL PACK
0E59 C2780E JNZ ERROR
0E5D C3090E JMP BINLD

; PACK TAKES DATA IN REG B AND D
; AND PACKS INTO REG B
; REG D CONTAINS MOST SIGNIFICANT HEX
0E60 7A PACK: MOV A,D
0E61 FE3A CPI 3AH
0E63 CA7B0E JZ ERROR
0E66 07 RLC
0E67 07 RLC
0E68 07 RLC
0E69 07 RLC
0E6A 57 MOV D,A
0E6B 78 MOV A,B
0E6C FE3A CPI 3AH
0E6E CA7B0E JZ ERROR
0E71 B2 ORA D
0E72 47 MOV B,A
0E73 3AD10E LDA CHECK
0E76 80 ADD B
0E77 32D10E STA CHECK
0E7A C9 RET

; ERROR HALT
; HALT IF A COLON IS DETECTED IN THE DATA
; HALT IF AN ERROR IS DETECTED
; 0E7B 0613  ERR: MVI B, 13H ; STOP TAPE READER
0E7D CDD20E CALL OUTPT ; AND HALT THE LOADER
0E80 76 HLT

; TIMER LOADS DELAY FROM ACC TO CCM TIMMER
; INTERRUPTS ARE NOT USED-- TIME IS IN MS

0E81 D301 TIMER: OUT 1 ; SEND TIME TO CCM
0E83 DB03 LOOP1: IN 3 ; READ TIME FROM CCM
0E85 B7 ORA A ; TEST TIME
0E86 C2830E JNZ LOOP1 ; WAIT FOR TIME
0E89 C9 RET

; TTY READS CHARACTER FROM TTY
; USES SERIAL I/O
; RETURNS ONLY HEX OR COLON
; CONVERTS ASCII TO HEX
; REG C USED FOR BIT COLOUNT

; 080 FB INPUT: EI ; ENABLE INTERRUPTS
0E8B 76 HLT ; AND WAIT FOR TTY
0E8C F3 DI ; DISABLE INTERRUPTS
0E8D 010900 LXI B, 9 ; CLEAR B, SET BIT COUNT TO 9
0E90 3E0D MVI A, 13 ; SKIP 1.5 BIT TIMES
0E92 C390E JMP JUMP3
0E95 DB17 LOOP2: IN 17H ; INPUT FROM TTY
0E97 B0 ORA B ; OR BIT INTO BYTE
0E98 0F RRC ; ROTATE RIGHT
0E99 47 MOV B, A ; SAVE BYTE
0E9A 3E09 MVI A, 9 ; WAIT FOR
0E9C CD810E JUMP3: CALL TIMER ; NEXT BIT
0E9F 0D DCR C ; DECREMENT BIT COUNTER
0EAC C2950E JNZ LOOP2
0EAE 78 MOV A, B ; SAVE CHARACTER
0EAF E67F ANI 7FH ; REMOVE PARITY
0EB 47 MOV B, A
0EB7 C9 RET
0EBA CD90E TTY: CALL INPUT ; GET NEXT CHARACTER
0EB8 78 MOV A, B ; RETURN CHARACTER TO MCG
0EBD FE2F CPI 2FH ; IS CHARACTER LESS THAN 0
0EE0 DAA80E JC TTY ; LOOK FOR NEXT CHARACTER
0EE1 FE3A CPI 3AH ; CHECK FOR 0 TO 9
0EE2 DAC50E JC JUMP1
0EE5 CAC70E JZ JUMP2 ; JUMP2 IF COLON
0EE8 FE41 CPI 41H ; CHECK FOR A
0EEB DAA80E JC TTY ; GET NEXT CHARACTER
0EEC FE47 CPI 47H ; GET NEXT CHARACTER
0EC0 D2A0E JNC TTY ; IF GREATER THAN F
0EC3 C609  ADI  9  ;CORRECT HEX
0EC5 E60F  JUMP1: ANI 0FH  ;REMOVE LEFT 4 BITS
0EC7 47  JUMP2: MOV B,A
0EC8 C9  RET

;PROGRAM LOADED CORRECTLY
;STOP TTY AND JUMP TO PROGRAM

0EC9 0613  STOP: MVI B,13H  ;STOP TTY
0ECA CDD20E  CALL OUTPT
0EEC C30001  JMP 100H  ;STARTING ADDRESS OF PROGRAM
0ED1 CHECK: DS 1  ;CHECK SUM STORAGE
;TTY OUTPUT SUBROUTINE

0ED2 0E09  OUTPT: MVI C,9  ;SET BIT COUNTER
0ED4 AF  XRA A  ;CLEAR ACC
0ED5 D317  OUT 17H  ;SEND 0
0ED7 3E09  RT1: MVI A,9  ;SET TIMMER
0ED9 CD810E  CALL TIMER
0EDC 78  MOY A,B  ;DATA TO ACC
0D D317  OUT 17H  ;SEND TO TTY
0EDF 0F  RRC  ;ROTATE DATA
0EE0 47  MOV B,A  ;SAVE
0EE1 0D  DCR C
0EE2 C2D70E  JNZ RT1
0EE5 3E01  MVI A,1  ;SEND MARK
0EE7 D317  OUT 17H  ;TO TTY
0EE9 3E1C  MVI A,28
0EEB CD810E  CALL TIMER
0EEE9 C9  RET

;BINARY DUMP PROGRAM
;PUNCHES BINARY MEMORY IMAGE
;OUTPT ROUTINE PUNCHES BINARY ON TTY
;MEMORY IMAGE IS IN REVERSE ORDER

0EEF 31FE0F  DUMP: LXI SP, 0FFEH  ;SET STACK POINTER
0EF2 21FD0E  LXI H, ENDPG ;START OF DUMP
0EF5 16FF  MVI D, 0FFH  ;LENGTH OF DUMP
0EF7 46  LOOP3: MOV B,M  ;LOAD BYTE
0EF8 CDD20E  CALL OUTPT  ;PUNCH BYTE
0EFB 2B  DEX H  ;DECREMENT ADDRESS
0EFC 15  DCR D  ;DECREMENT COUNT
0EFD C2F70E  ENDPG: JNZ LOOP3  ;GET NEXT BYTE
0000  END
Boot

To allow for bootstrap loading of programs and for patching of programs or data in memory via the teletype. The program uses less than 200 bytes of memory and may be placed in ROM or entered manually.

8080, teletype and interface.

None.

The input format is described in comments at the beginning of the program listing.

Code or data may be loaded into any locations in memory. Loading or patching of memory locations need not be in sequential order.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Assembler/Compiler Used:</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>MAC 80</td>
</tr>
<tr>
<td>RAM Required:</td>
<td>Programmer:</td>
</tr>
<tr>
<td>x'9D' (157 bytes)</td>
<td>Jeff Kravitz</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>Company:</td>
</tr>
<tr>
<td>Same as above</td>
<td>Grumman Data Systems</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td>Address:</td>
</tr>
<tr>
<td>5</td>
<td>197 Fairhaven Blvd.</td>
</tr>
<tr>
<td></td>
<td>Woodbury, N.Y., 11797</td>
</tr>
</tbody>
</table>
; REF. NO. AB24
; PROGRAM NAME BOOT
;
; BOOTLOADER & DEBUGGING TOOL FOR INTEL 8080 WITH TELETYPES
;
; THIS PROGRAM ACCEPTS INPUT FROM THE TTY KEYBOARD OR READ
; IN HEXADECIMAL FORMAT. IT WILL ENTER PROGRAMS OR DATA IN
; ANY LOCATION IN MEMORY AND BRANCH TO ANY LOCATION.
;
; THE FOLLOWING CHARACTERS HAVE SPECIAL MEANING TO THE PRO
;
; (COLON) INDICATES THAT THE NEXT FOUR HEX DIGITS
; ARE TO BE USED AS THE ADDRESS OF ANY SUB
; DATA TO BE LOADED. THE COLON MAY BE USED
; OF TIMES TO ENTER PATCHES INTO DIFFERENT
; MEMORY OR IF LOADING FROM PAPER TAPE, TO
; SPECIFY THE INITIAL LOAD ADDRESS AND ANY
; ADDRESSES.
;
; T
; INDICATES THAT THE INPUT IS COMING FROM
; TAPE. THIS TURNS OFF ECHOING OF CHARACTE
; PRINTER. (THIS WILL NOT WORK IF THE PROG
; ONLY MEMORY). IF A PAPER TAPE CONTAINS A
; ITS FIRST CHARACTE, THE TTY WILL NOT EC
; THE T MAY ONLY BE USED AS THE FIRST CHAR
; THE PROGRAM ONLY CHECKS FOR IT ONCE.
;
; G
; INDICATES THAT THE NEXT FOUR HEX DIGITS
; TO BE USED AS A BRANCH ADDRESS. THE PROG
; BRANCH TO THE SPECIFIED LOCATON.
;
; X
; INDICATES THAT THE LOADING OR PATCHING O
; IS COMPLETED. THE PROGRAM WILL HALT WITH
; ENABLED.
;
; BLANKS AND CARRIAGE RETURNS MAY BE FREEL
; FOR CONVENIENCE AND READABILITY. CARRIAGE
; WILL ECHO AS CARRIAGE RETURN, LINE FEED.
;
; A SAMPLE INPUT MIGHT BE AS FOLLOW;

4-148
:0100 A2F4 A345 97B6 98 45 0909 4343 G 0400

THIS WILL PLACE 'A2E4 A354... ETC.' AT LOCATION 100
AND THEN BRANCH TO LOCATION 400 (ALL NUMBERS IN HEX)

ANOTHER EXAMPLE;
:0304 F2 :0309 46 :0807 87 X

THIS WOULD PLACE 'F2' AT 304, '46' AT 309, '87' AT 807 A

;***************************************************************************
;0000 31B800 INIT: LXI SP,ENDP+10H ; INITIALIZE THE STACK P
;0001 210001 LXI H,START ; LOAD DEFAULT STARTING
;0002 CD5900 CALL TESTX ; TEST FOR C'?' (NO ECHO
;0003 CD6900 LOOP1: CALL TREAD ; READ, TEST, CONVERT, E
;0004 77 MOV M,A ; STORE THE BYTE IN MEMO
;0005 22 INX H ; INCREMENT THE LOAD ADD
;0006 C20900 JMP LOOP1 ; CONTINUE

L200 START EQU 100H ; DEFAULT LOAD ADDRESS

;***************************************************************************
; ; DEFAULT LOAD ADDRESS
;

0011 CD7B00 GO: CALL GETHL ; GET H&L FROM TTY
0014 E9 PCHL ; BRANCH TO USER ADDRESS

;***************************************************************************
; ; ERROR ROUTINE
;

0015 063F ERROR: MVI B,'?' ; TYPE A QUESTION MARK
0017 CD3200 CALL OUTT
001A FB ERR2: EI
001B 76 HLT

;***************************************************************************
; READ AND ECHO SUBROUTINES

; NOTE THIS PROGRAM ASSUMES THE TTY IS AT 70 AND 71
; THE ADDRESSES IN THE NEXT TWO ROUTINES MUST BE MODIFIED
; IF THIS IS NOT THE CASE.

001C DB71 READ: IN 71H ; READ TTY STATUS BYTES BYTE
001E E8FB ANI 0FFH-TBE ; REMOVE THE 'TRANSMIT BUFFER EMP
0020 FE01  CPI  01H    ;TEST FOR DATA AVAILABLE BIT
0022 F21500  JP  ERROR    ;OTHER BIT ON. ERROR FROM TTY
0025 FA1C00  JM  READ    ;DATA NOT YET AVAIL. KEEP TRYING
0028 EB70  IN  70H    ;READ A DATA BYTE
002A FEE0  CPI  00H    ;SKIP BLANK LEADER TAPE
002C CA1C00  JZ  READ    ;TURN OFF 'PARITY' BIT
0031 47  MOV  B, A    ;PUT CHAR IN B FOR ECHO ROUTINE

;**********************************************************************
; ECHO SUBROUTINE
;**********************************************************************

0032 DB71  OUTT:  IN  71H    ;GET TTY STATUS
0034 E604  ANI  TBE    ;TEST FOR TRANSMIT BUFFER EMPTY
0036 CA1D00  JZ  OUTT    ;NOT EMPTY, TEST AGAIN
0039 78  MOV  A, B    ;GET DATA BYTE IN A
003A D270  NOPE:  OUT  70H    ;OUT THE BYTE
003C FE20  CPI  87H    ;TEST FOR A BLANK
003E CA1C00  JZ  READ    ;YES, READ AGAIN
0041 FEAF  CPI  LF    ;TEST FOR A LINE FEED
0044 4F  CA1C00  JZ  READ    ;YES READ AGAIN FEED
0047 FEF0  CPI  CR    ;TEST FOR CARRIAGE RETURN
004A D0  RNZ    ;NO, RETURN TO CALLER
004D 060A  MVI  B, LF    ;ECHO A LINE FEED
0056 C31200  JMP  OUTT

;**********************************************************************
; SPECIAL CHARACTER TESTING ROUTINE
;**********************************************************************

0059 CD1C00  TESTX:  CALL  READ    ;READ A BYTE FROM THE TTY & ECHO
005C FE51  CPI  CHART    ;IS THE THE 'NO ECHO' CHAR?
005E C2600  JNZ  TEST    ;NO, CONTINUE
0061 2E00  MVI  A, NOPE    ;NOP THE ECHO ROUTINE
0064 223A00  STA  NOPE

0066 223B00  STA  NOPE+1
0069 CD1C00  TREAD:  CALL  READ    ;READ ANOTHER OTHER CHARACTER

006C FE47  TEST:  CPI  CHARG    ;IS THIS THE "GO" CHARACTER
006E CA1100  JZ  GO    ;YES "GO"
0071 FE58  CPI  CHARX    ;IS THIS THE "DONE" CHARACTER
0073 CA1A00  JZ  ERR2    ;YES, HALT
0076 FE3A  CPI  CHARA    ;IS THIS THE "ADDRESS" CHARACTER

4-150
0000  C29700  JNZ    CONV1 ; NO TREAT AS HEX DIGIT
0008  CD8400  GETHL: CALL CONV ; READ A 4 HEX DIGIT ADDRESS IN P
0010  BC7E  MOV    H.A ; HI ORDER BYTE OF ADDRESS
0012  CD8400  CALL CONV ; GET LOW ORDER ADDRESS
0014  5F  MOV    L.A
0015  C9  RET

0017  NOPI  EQU  00H ; NOP INSTRUCTION
001A  CHART EQU  54H ; C'T' 'NO ECHO' CHAR
001D  CHARG EQU  47H ; C'G' 'GO' CHAR
0020  CHARY EQU  50H ; C'X' 'DONE' CHAR
0023  CHARA EQU  3AH ; C'.' 'ADDRESS' CHAR

;***************************************************************************
; *                        ASCII TO HEX ASSEMBLY ROUTINE                       *
; *                                                                          *
;***************************************************************************
0034  CD1C00  CONV: CALL READ ; READ A BYTE FROM THE TTY & ECHO
003C  CD9500  CONV1: CALL TLATE ; TRANSLATE THE BYTE FROM ASCII TO T
0040  87  ADD    A ; SHIFT LEFT 2
0041  87  ADD    A ; SHIFT LEFT ANOTHER 2
0042  5F  MOV    E.A ; SAVE IN B
0043  CD1C00  CALL READ ; READ ANOTHER BYTE
004B  CD9500  CALL TLATE ; TRANSLATE FROM ASCII TO HEX
0053  E0  ORA    B ; COMBINE TWO HALF BYTES
0054  C9  RET ; RETURN TO CALLER

;***************************************************************************
; *                        ASCII TO HEX TRANSLATION ROUTINE                     *
; *                                                                          *
;***************************************************************************
0055  D630  TLATE: SUI '0' ; SUBTRACT THE LOWEST PERMISSABLE
005D  FA1500  JM ERROR ; TOO LOW, BAD HEX CHAR
0061  FE8A  CPI '9'-'0'+1 ; TEST FOR A 9
0065  F8  RM ; BETWEEN ZERO AND NINE, DONE
0066  D611  SUI 'A'-'0' ; SUBTRACT THE DIFFERENCE TO A
006F  FA1500  JM ERROR ; BETWEEN '0' AND 'A', ERROR
0073  FE65  CPI 'F'-'A'+1 ; TEST FOR HIGHER THAN 'F'
0077  F8  RM ; BETWEEN 'A' AND 'F', OK
0078  CD1500  JMP ERROR ; GREATER THAN 'F', ERROR
007B  ENDP  EQU $  
007F  END
Program Title: Octal PROM Programming

Function: See below

Required Hardware: Standard INTELLEc 8/MOD 80 with teletype

Required Software: System Monitor

Input Parameters: This program accepts sets of 3 octal numbers. The fourth character (unless it is a rubout) will cause the BYTES to be placed in memory starting at 100H. Any invalid octal character input in the first 3 positions will cause a carriage return and line feed to be output to the teletype and the line to be ignored. Any number of sets may be input (up to the practical limit of 100H). Whenever a "bell" is typed, the address of the last valid byte on Page 1 will be displayed on the registerflag lights and control will pass to the system monitor. To program the PROM, type P100, lNN,0 where NN is the HEX number displayed on the lights.

Output Results: The program may be tested by inputting your choice of octal numbers and verifying that the PROM accepted them.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Assembler/Compiler Used:</th>
</tr>
</thead>
<tbody>
<tr>
<td>A11</td>
<td>INTELLEc 8/MOD 80 VER. 3.0</td>
</tr>
<tr>
<td>RAM Required:</td>
<td>Programmer:</td>
</tr>
<tr>
<td>6E</td>
<td>Bob Gill</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>Company:</td>
</tr>
<tr>
<td>None</td>
<td>Applied Physics Laboratory</td>
</tr>
</tbody>
</table>

Maximum Subroutine Nesting Level: 1 - plus whatever is in Cl and CO

Address: 1013 NE 40th Seattle, Washington 98195
; REF. NO. AB25
; PROGRAM NAME OCTAL FROM PROGRAMMING

ORG 18H

SET STACK POINTER

0010 31F001 LXI SP, 1F0H
0013 1600 MVI D, 0
0015 210001 MVI H, 100H
0018 0600 BB: MVI B, 0
001A 1E00 MVI E, 2-2
001C CD0003 CALL 380H
001F 4F MOV C, A
0020 77 MOV M, A
0021 CD0938 CALL 3809H
0024 7E MOV A, M
0027 E67F MOV 7FH
0029 FE7F CPI 7FH
002C CA5700 JZ CRLF
002E FA0F7 CPI 07H
0031 FA6000 JZ OUTY
0033 FA5700 CPI 30H
0036 FE38 JMP CRLF
0038 FE25700 CPI 38H
003B E607 ANI 0
003D 08 ADD B
003E 47 MOV B, A
003F 7B MOV A, E
0040 FE00 CPI 0
0042 CA4E00 JZ FIN
0045 78 MOV A, B
0046 07 RLC
0047 07 RLC
0048 07 RLC
0049 47 MOV B, A
004A 1C INR E
004B C3100 JMP AA
004E CD0338 FIN: CALL 3803H
0051 E67F ANI 7FH
0053 78 MOV A, B
0054 77 MOV M, A
0055 14 INR D
0056 23 INX H
0057 0E00 CRLF: MVI C, 0DH
0059 CD0938 CALL 3809H
005C 0E0A MVI C, 0AH
005E CD0938 CALL 3809H

; SAVE IT
; ECHO IT
; RELOAD CHAR
; MASK TO 7 BITS
; IGNORE LINE IF RUBOUT
; IS IT A BELL
; GO TO PROGRAMMING PORTI
; IGNORE LINE IF #<0
; IGNORE LINE IF #>7
; CONVERT ASCII TO BINARY
; ADD CHAR TO ACCUM
; IF LOOP=0, HAVE LOOKED A
; MULT ACCUM BY 7
; INCREMENT LOOP COUNTER
; GET 4TH CHAR
; MASK TO 7 BITS
; IF NON-RUBOUT, STORE AC
; INCREMENT # OF BYTES CO
; MOVE ADDRESS POINTER-G
; OUTPUT CR
; OUTPUT LF
0061 C31800 JMP BB
0064 7A OUTY: MOV A, D
0065 3D DCR A
0066 D3FF OUT 255

;DISPLAY LAST ADDRESS ON PAGE 1 ON REGISTER/FLAG LIGHTS

0068 C20038 JMP 2800H
0000 END
Program Title: 8080 IDLE Analyzer for Approximating CPU utilization

Function:
Displays amount of time 8080 would have spent in an Idle loop. When RUN time is compared with Idle time, the percent of CPU utilization can be calculated. Time display is in memory, in ASCII.

Required Hardware:
8080

Required Software:
Routine to call "IDLE", Routine to restore Registers after an interrupt, Routine to call "Time Display" when job is complete

Input Parameters:
NONE

Output Results:
ASCII time, format:

\[ \text{bIDLEbTIMEbMMbSS.S} \]

where \( m \) = minutes
\( s \) = seconds

Registers Modified:
IDLE: A, B, H, L
Time Display: ALL

RAM Required:
16 BYTES (Hex)

ROM Required:
99 BYTES (Hex)

Maximum Subroutine Nesting Level:
Uses 2 BYTES of stack
N/A

Assembler/Compiler Used:
8080 MACRO ASSEMBLER
Ver 1.0

Programmer:
J. William McConigal

Company:
Sycor, Inc.

Address:
100 Phoenix Drive
Ann Arbor, MI 48103
REF. NO. AB26.
PROGRAM 8080 IDLE ANALYZER FOR APPROXIMATING CPU UTILIZATION

8080 IDLE ANALYZER FOR APPROXIMATING
CPU UTILIZATION

J W MCGONIGAL
MAY 7, 1975

THE FIRST ROUTINE (IDLE), SHOULD BE CALLED WHENEVER
THE 8080 IS WAITING FOR AN EVENT (INTERRUPT) TO OCCUR.

WHEN THE JOB HAS COMPLETED PROCESSING, CALL
"TIME DISPLAY" (TIMED). IDLE TIME WILL BE STORED IN
MEMORY, IN ASCII, STARTING AT LOCATION "ITIME".
MAXIMUM TIME IS 54 MINUTES, 36.7 SECONDS.
RESOLUTION IS 0.1 SECONDS.

USER IS RESPONSIBLE FOR SAVING REGISTERS AGTER AN
INTERRUPT, AND IS ALSO RESPONSIBLE FOR RESTORING THEM
BEFORE AGAIN CALLING "IDLE". NEITHER "IDLE" NOR
"TIMED" REQUIRE INITIAL PARAMETERS. STACK POINTER IS
ASSUMED TO BE PRE-INITIALIZED FOR "TIMED".

0100
ORG 100H ;MAY BE AT ANY LOCATION

*** IDLE ROUTINE ***
R(A) = 1.92 MILISECOND COUNTER
R(B) = 7.5 MICROSECOND COUNTER
R(H.L) = 50 MILISECOND COUNTER
OTHER REGISTERS, STACK, AND STACK POINTER ARE
NEITHER USED NOR MODIFIED.

*** TIME DISPLAY ***
CALLS CLOCK CONVERSION ROUTINE (INCLUDED)
R(A) = WORK REGISTER
R(B.C) = MASKS
R(D.E) = BINARY COUNTER
R(H.L) = DATA POINTER
USES STACK POINTER AND THE STACK
REQUIRES TWO BYTES OF STACK SPACE

THIS SECTION MAY BE IN ROM

0100 3A9001 IDLE: LDA FLAG1 ;INITIALIZE
0103 C600 ADI 0 ;ACTIVATE CONDITION FLAGS
0105 C21501  JNZ  IDLE1 ;JUMP IF NOT FIRST PASS
0108 3C  INR A
0109 329001  STA  FLAG1 ;TURN FLAG ON
010C 0600  MVI B, 0 ;ZERO R(B)
010E 210000  LXI H, 0 ;ZERO R(H,L)
0111 229101  SHLD BINCT ;ZERO THE BINARY COUNTER
0114 AF  XRA A ;CLEAR THE ACCUMULATOR
0115 04  IDLE1:  INR B ;INCREMENT R(B)
0116 C21501  JNZ IDLE1 ;LOOP FOR 1.92 MILLISECONDS
0119 3C  INR A ;INCREMENT THE ACCUMULATOR
011A FE1A  CPI 26D ;CHECK FOR 26 PASSES
011C C21501  JNZ IDLE1 ;LOOP IF NOT 49.92 MILLISECONDS
011F 23  INX H
0120 229101  SHLD BINCT ;INCR 50 MILLISECOND COUNTER
0123 C3FF00  JMP IDLE -1 ;LOOP

; TIME DISPLAY

0126 3E30  TIMED:  MVI A, '0' ;RESET DISPLAY TO ASCII ZERO
0128 219D01  LXI H, IMINT ;SET H.L TO TENS OF MINUTES
012B 77  MOV M, A ;ZERO TENS OF MINUTES
012C 23  INX H ;BUMP POINTER TO MINUTES
012D 77  MOV M, A ;ZERO MINUTES
012E 23  INX H ;SKIP THE COLON
012F 23  INX H ;POINT TO TENS OF SECOND
0130 77  MOV M, A ;ZERO
0131 23  INX H
0132 77  MOV M, A ;ZERO SECOND
0133 23  INX H ;SKIP THE PERIOD
0134 23  INX H
0135 77  MOV M, A ;ZERO TENTH OF SECONDS

; START TIME CONVERSION

0136 2A9101  LHLD BINCT ;GET BINARY TIME
0139 EB  XCHG ;PUT IN R(D,E)
013A 219D01  LXI H, IMINT ;SET POINTER TO TENS OF MINUTES
013D 01E02E  LXI B, 2EE0H ;TENS OF MINUTES MASK
0140 CD7701  TIME1:  CALL CCONV ;CALL CLOCK CONVERTER
0143 CA4B01  JZ TIME2 ;JUMP IF NO TENS OF MINUTES
0146 86  ADD M ;INCREMENT DISPLAY COUNT
0147 77  MOV M, A ;STORE
0148 C34001  JMP TIME1 ;LOOP
014B 23  TIME2:  INX H ;ADVANCE TO MINUTES
014C 01B004  LXI B, 04B0H ;MASK FOR MINUTES
014F CD7701  TIME3:  CALL CCONV ;CALL THE CLOCK CONVERTER
0151 01C000  LXI B, 00C0H ;MASK FOR TEN OF SECONDS
0155 CD7701  TIME5:  CALL CCONV ;CALL CLOCK CONVERTED
0158 CA6001  JZ TIME6 ;JUMP IF NO TENS
015B 86  ADD M

4-159
015C 77  MOV  M, A  ; INCREMENT DISPLAY COUNT
015D C35501  JMP  TIME5  ; LOOP
0160 23  TIME6:  INX  H  ; ADVANCE TO SECONDS
0161 01400  LXI  B, 0014H  ; MASK FOR SECONDS
0164 CD7701  TIME7:  CALL  CCONV  ; CALL CLOCK CONVERTER
0167 CA6F01  JZ  TIME8  ; JUMP IF NO SECONDS
016A 86  ADD  M
016B 77  MOV  M, A  ; INCREMENT DISPLAY COUNT
016C C36401  JMP  TIME7  ; LOOP
016F 23  TIME8:  INX  H  ; ADVANCE TO TENTHS OF SECONDS
0170 23  INX  H
0171 AF  XRA  A  ; CLEAR ACCUMULATOR AND THE CARRY
0172 7B  MOV  A, E  ; GET REMAINING BINARY TIME
0173 1F  RAR  ; SHIFT OFF HUNDREDTHS
0174 86  ADD  M
0175 77  MOV  M, A  ; STORE TENTHS OF SECONDS SECONDS
0176 C9  RET  ; RETURN TO USER

; CLOCK CONVERTER

; RETURN: A=0 IF DATA LESS THAN MASK
; A=1 IF DATA GREATER THAN MASK, ALSO PERFORMS SUBTRACTION

; RETURN/ENTER WITH: R(B,C) = 16 BIT MASK
; R(D,E) = 16 BIT DATA

0177 7A  CCONV:  MOV  A, D
0178 B8  CMP  B
0179 DA8401  JC  CCON1  ; RETURN A=0 IF A LT. B
017C C28601  JNZ  CCON2  ; JUMP TO SUBTRACT IF A GT. B
017F 7B  MOV  A, E  ; A = B; CHECK LOWER BYTE
0180 B9  CMP  C
0181 D28601  JNC  CCON2  ; JUMP IF A GT. OR EQ. C
0184 AF  CCON1:  XRA  A  ; ZERO THE ACCUMULATOR
0185 C9  RET  ; RETURN
0186 AF  CCON2:  XRA  A  ; CLEAR A AND CARRY BIT
0187 7B  MOV  A, E  ; GET DATA
0188 99  SBB  C  ; SUBTRACT MASK
0189 5F  MOV  E, A  ; SAVE IT
018A 7A  MOV  A, D
018B 98  SBB  B  ; DO UPPER BYTE
018C 57  MOV  D, A  ; STORE IT
018D AF  XRA  A  ; ZERO ACCUMULATOR
018E 3C  INR  A  ; SET A=1
\8F C9  RET  ; RETURN

; THIS SECTION MUST BE IN RAM AND CAN BE RELOCATED
BY AN ORG STATEMENT OF INCLUSION WITH OTHER CODE.

; FIRST PASS FLAG
0190 00  FLAG1: DB  0
0191 00  BINCT: DB  0 ; BINARY 50 MILISECOND COUNTER
0192 00  DB  0
0193 20494440 ITIME: DB ' IDLE TIME' ; MAY BE MODIFIED OR OMITTED
0197 45205449
0198 4D45
019D 20  IMINT: DB ' ' ; TENS/UNITS OF MINUTES
019E 20  DB ' ' ; SPACE
019F 20  ISECS: DB ' ' ; TENS/UNITS OF SECONDS
01A0 20  DB ' ' ; PERIOD
01A1 20  ITENTH: DB ' ' ; TENTHS OF SECONDS
01A2 20  DB ' ' ; BLANK Padding
0000  END

4-161
REAL TIME EXECUTIVE

PERFORMS PROCESSOR INITIALIZATION, PERIODIC AND DEMAND SCHEDULING, ROUTINE TERMINATION, AND WAITS DURING IDLE TIME.

8080, TIMER THAT CAUSES A PERIODIC INTERRUPT (i.e. 1ms), ROM, AND RAM

NONE

<table>
<thead>
<tr>
<th>ROUTINE</th>
<th>PARAMETER</th>
<th>REGS</th>
<th>RESULT</th>
</tr>
</thead>
<tbody>
<tr>
<td>?CLKI</td>
<td>CLOCK INTERRUPT</td>
<td>----</td>
<td>?CLOK IS INCREMENTED &amp; PERIODIC ROUTINES MAY BE ADDED TO DISPATCH QUEUE</td>
</tr>
<tr>
<td>?PERQ</td>
<td>ROUTINE ADDRESS</td>
<td>D,E</td>
<td>THE PASSED ROUTINE IS ADDED TO THE TIME QUEUE TO BE ACTIVATED AT TIME IN A.</td>
</tr>
<tr>
<td></td>
<td>TIME</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>?INIT</td>
<td>ROUTINE ADDRESS</td>
<td>D,E</td>
<td>ADDS ROUTINE TO LIFO DISPATCH QUEUE EXITS TO NEXT TASK ON QUEUE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>?DSPC</td>
<td>----</td>
<td>----</td>
<td>WAITS FOR AN INTERRUPT (SYSTEM IDLE)</td>
</tr>
<tr>
<td>?WAIT</td>
<td>----</td>
<td>----</td>
<td>CLEARS RAM AND SETS UP QUEUES AND STACKS</td>
</tr>
<tr>
<td>?INTP</td>
<td>----</td>
<td>----</td>
<td></td>
</tr>
</tbody>
</table>

Registers Modified:

<table>
<thead>
<tr>
<th>ALL</th>
</tr>
</thead>
</table>

Assembler/Compiler Used:

8080 MDS MACRO ASSEMBLER -L.0

RAM Required:

22 BYTES MIN → 42 BYTES REC.

ROM Required:

$0FEH = 254 BYTES

Maximum Subroutine Nesting Level:

2

Programmer: TED J. CLOWES

Company: CUBIC CORP.

Address: 4285 PONDEROSA
          SAN DIEGO, CA. 92123
MICROCOMPUTER USER'S
LIBRARY SUBMITTAL FORM

Program Title
Read/write Routines for Interchange Tapes

Function
Subroutines read and write blocks and characters for any common audio cassette recorder. Variable redundancy allows high-speed or highly reliable operation.

Required Hardware
8080 with one-chip cassette tape interface

Required Software
Self-sufficient

Input Parameters
Block I/O: Specify array address and block length
Character I/O: Specify word size

Output Results
Writer: Recorded cassette tape
Reader: Blocks of data in memory

Registers Modified:
All

Assembler/Compiler Used:
Macro Assembler

RAM Required:
User's data

Programmer:
J.L. Ogdin

ROM Required:
306 bytes

Company:
Microcomputer Technique, Inc.

Maximum Subroutine Nesting Level:
3

Address: 11227 Handlebar Road
           Reston, Va. 22091
PROPORTIONAL POWER CONTROL IMAGE BUILDER

This program builds an "ON-OFF" image in RAM to allow proportional power control using zero crossing solid state relays. After an image is built using this program, consecutive image words are accessed by an interrupt routine (not shown here) that causes solid state zero crossing relays to turn on and off as the interrupt program steps through the RAM image. Assuming a 60 Hz line interrupt frequency, one complete image cycle takes \(256/60 = 4.4\) sec. The image is built so that power is applied nearly evenly throughout the cycle. For example, the most significant value bit controls every other word; the next most significant bit in a value word controls every other remaining word and so on. Each value word controls one output bit stream. The present program is set to take 32 value words of 8 bits each, and produce a switching image; \(32/8 \times 100H = 400H\) locations long to control 4 output ports or 32 relays.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Assembler/Compiler Used:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PLM1, PLM2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RAM Required:</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>(130 + p (100H))</td>
<td>R. Yodlowski/617-890-2000 X494</td>
</tr>
<tr>
<td>(p = # \text{ of ports used})</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ROM Required:</th>
<th>Company:</th>
</tr>
</thead>
<tbody>
<tr>
<td>15AH</td>
<td>LFE Corporation</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Maximum Subroutine Nesting Level:</th>
<th>Address:</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1601 Trapelo Rd, Waltham, Mass. 02154</td>
</tr>
</tbody>
</table>
/*REF. NO. AB28 */
/*PROGRAM TITLE PROPORTIONAL POWER CONTROL IMAGE BUILDER*/
200H: DECLARE VALUE (32) BYTE;
DECLARE MSB$USED LITERALLY '7';
DECLARE PWRIMBRET ADDRESS;

/ / PWRIMB.RY PWRIMB.RY APRIL 9, 1975*/

DO:  / /****POWER IMAGE BUILDER****/

/* THIS PROGRAM SEGMENT DEVELOPS A BIT-BY-BIT RAM IMAGE OF THE POWER PATTERNS TO BE APPLIED TO SOLID STATE RELAYS TO ACHIEVE SYNCHRONOUS SWITCHING PROPORTIONAL POWER CONTROL. THE DESIRED POWER LEVELS ARE MADE AVAILABLE IN A BYTE VECTOR: VALUE (N). EACH VALUE SPECIFIES THE NUMBER OF ON POWER CYCLES OUT OF A POSSIBLE FULL POWER MAXIMUM OF 1+2**(MSB$USED+1) CYCLES. */

DECLARE ( /*POWER IMAGE BUILDER VARIABLES*/
STARTOPPRT, /* POINTS TO 1ST VALUE OF PORIS. */
BITSEL, /* SELECTS BITS FROM VALUE BYTES */
BITINBR, /* NUMBER OF BIT WITHIN VALUE BYTE */
BIT$HOLDER, /* HOLDS BITS SELECTED FROM VALUES */
PORTINBR, /* NUMBER OF PORT */
RLYCFPORT, /* WHICH RLY OF CURRENT PORT */
0, /* HOLDER VARIABLE */
STRSCYC, /* CYCLES FROM IMAGE TO START */
INCRCYC) /* OF CYCLES BETWEEN PATTERNS */
BYTE;

DECLARE CYC$CTR ADDRESS; /* CYCLE COUNT DOWN IN IMAGE*/
DECLARE IMAGE (1024) BYTE;

/* FOR EACH PORT */
DO STARTOFPORT = 0 TO LAST(VALUE) BY 8;
PORINBR = SHR (STARTOFPORT, 3);
BITSEL = 1000$0000B;

/* FOR EACH BIT OF VALUE USED */
DO BITNBR = 0 TO MSBSUSED;
BITSHOLDER = 0;
BITSEL = ROL (BITSEL, 1);

/* FOR EACH OUTPUT RELAY OF THE PORT */
DJ RLYOFPORT = 0 TO 7;
BITSHOLDER = ROL (BITSHOLDER, 1);

/* COLLECT THE CORRESP. BITS INTO BIT HOLDER */
BITSHOLDER = BITSHOLDER OR (BITSEL AND VALUE (STARTOFPORT +
RLYOFPORT));
END;

/* CORRECT BIT POSITIONS OF BITSHOLDER */
IF BITNBR <> 0
THEN BITSHOLDER = ROR (BITSHOLDER, BITNBR);

/* COMPUTE START CYCLE NUMBER */
I = 7 - MSBSUSED + BITNBR;
STARTSCYC = SHR (255, I +1);

/* COMPUTE CYCLE INCREMENT */
IF I <> 0
THEN INCRSCYC = 1 + SHR (255,I);
ELSE INCRSCYC = 255;

/* PLACE BITSHOLDER INTO THE (STARTSCYC) TH IMAGE */
00121 4 AND EVERY (INCR$CYC) IMAGES THEREAFTER*/
00122 4
00123 4
00124 4 DO CYC$CIR = STRT$CYC TO 255 BY INCR$CYC;
00125 4
00126 4 IMAGE ((256*PORTNBR) + CYC$CIR) = BIT$HOLDER;
00127 5
00128 5 END; /* PLACEMENT OF BIT$HOLDER */
00129 4
00130 4 END; /* BITS OF VALUE USED */
00131 3
00132 3 END; /* PORTS */
00133 2
00134 2 END; /*****POWER IMAGE BUILDER*****/
00135 1
00136 1 GO TO PWRIMBRET;
00137 1
00138 1
00139 1 EOF;
NO PROGRAM ERRORS
FLAG PROCESSING ROUTINE
A Routine for contact closure debouncing and processing

1. Simultaneously check 8 input flags to find those flags which have been present for 3 successive iterations of routine.
2. Report flags identified above.
3. Clear reported flags that have been serviced.
4. Remember serviced flags until input flags are cleared.

Input Interface (s) to bring in Contact Closures (Flags)

Routine to assemble up to 8 flags into an 8 bit word before each iteration of this routine. Routine to recycle thru this routine as frequently as desired.

8 bit word at RAM location "Flags" assembled with latest State of Contact Closures.
8 bit word at RAM location "Flags + 4" showing actions to be deleted from "Do" list.

8 bit word at RAM location "Flags + 5" showing actions "to do". This is a normally used output.
8 bit word at RAM locations "Flags + 3" showing actions needed.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Assembler/Compiler Used:</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL</td>
<td>Intellec 8/mode Macro assembler Ver 2.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RAM Required:</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 bytes</td>
<td>B. Weston</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ROM Required:</th>
<th>Company:</th>
</tr>
</thead>
<tbody>
<tr>
<td>49 bytes</td>
<td>TRANSCOM INCORPORATED</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Maximum Subroutine Nesting Level:</th>
<th>Address:</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td>580 Spring Street</td>
</tr>
<tr>
<td></td>
<td>Windsor Locks, Comm. 06096</td>
</tr>
</tbody>
</table>
INSTRUCTIONS
1. ASSEMBLE INPUT FLAGS INTO MEMLOC "FLAGS"
2. "OR" IN FLAGS THAT HAVE BEEN SERVICED
   INTO MEMLOC "FLAGS + 1"
   INTO MEMLOC "FLAGS + 4"
3. RUN ROUTINE
4. WORD IN MEMLOC "FLAGS + 5" REPORTS BIT
   POSITION OF FLAGS WHICH NEED TO BE
   SERVICED
5. WORD IN MEMLOC "FLAG + 3" REPORTS BIT
   POSITION OF FLAGS WHICH HAVE BEEN
   PRESENT FOR 3 OR MORE SUCCESSIVE
   ROUTINE ITERATIONS AND ARE STILL PRESENT
   OR HAVE NOT BEEN REPORTED TO HAVE BEEN
   SERVICE.

0041  FLAG   EQU  41H
0141  FLAGS  EQU  141H
002E  ORG   002EH

FLAG PROCESSING ROUTINE

002E 214101 LXI   H,FLAGS ; POINTER TO FLAG
0031  7E   MOV   A,M ; STORE CURRENT WORD IN A
0032  47   MOV   B,A ; AND ALSO IN B
0033  2C   INR   L ; POINTER TO PAST WORD
0034  AE   XRA   M ; FIND PAST/CURRENT CHANGES
0035  A6   ANA   M ; FIND DELETED FLAGS
0036  4F   MOV   C,A ; SAVE DELETED FLAGS IN C
0037  AE   XRA   M ; REMOVE DELETED FLAGS FROM
                  ; PAST WORD
0038  77   MOV   M,A ; RE-STORE CORRECTED PAST WORD
0039  2C   INR   L ; POINTER TO PREVIOUS PAST WORD
003A  79   MOV   A,C ; BRING IN DELETED FLAGS
003B  EEFF  XRI   OFFH ; INVERT
003C  A6   ANA   M ; DELETE FROM PREVIOUS PAST
                  ; WORD
003D  57   MOV   D,A ; SAVE RESULT IN D
003E  2E45 MVI   L,FLAG+4 ; POINT TO ACTION TAKEN
0041  7E   MOV   A,M ; STORE IN A
0042  2D   DCR   L ; POINT TO ACTION NEEDED
RUN ROUTINE AGAIN, MEMORY WILL BE CHANGED TO-

F8 F8 F8 7E 08 76

REPEAT ROUTINE AGAIN, RESULTING IN-

F8 F8 F8 FE 08 F6

CONTINUED EXECUTIONS WILL RESULT IN NO FURTHER CHANGES.

0000 END
SOFTWARE STACK ROUTINES FOR 8008

Subroutines provided for PUSH, POP, and EXCHANGE A with top of stack, and to save processor state on stack in case of an interrupt, and to restore it again.

NONE

NONE

E register must be initialized to empty stack (e.g. 255) at beginning of main program.

H and L registers left pointing to top of stack.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>A, (D), E, H, L</td>
<td>Tom Pittman</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RAM Required:</th>
<th>Company:</th>
</tr>
</thead>
<tbody>
<tr>
<td>4-255 Bytes (for stack)</td>
<td>MicroComputer Consultant</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ROM Required:</th>
<th>Address:</th>
</tr>
</thead>
<tbody>
<tr>
<td>8410 Bytes</td>
<td>PO 23189</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Maximum Subroutine Nesting Level:</th>
<th>City:</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0-</td>
<td>San Jose</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Assembler/Compiler Used:</th>
<th>State:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Macro 8</td>
<td>California 95153</td>
</tr>
</tbody>
</table>
SOFTWARE STACK ROUTINES FOR 8008

To permit interrupts in a 8008 microcomputer system requires either extra hardware for saving and restoring the processor status external to the CPU, or two dedicated registers, used only in the service of the interrupt (for holding the H and L registers, so that the processor state may be stored in memory).

The subroutines presented here are designed to get extra mileage out of one of those dedicated registers by designating it as a stack pointer for a stack which may be used by the main routine as well as the interrupt service routine. If the system has the additional hardware to disable interrupts during the save and restore sequences, the interrupt service routines themselves may also be interruptible, to the maximum subroutine nest depth allowed by the main routine and all nested interrupts.

Two categories of subroutines are presented here. The first is a set of subroutines to push, pop, and exchange the A register with the top of the stack. These routines are re-entrant, and are designed to be callable with a RST instruction, though of course they may be placed anywhere in memory to be called with the CAL instruction.

The second category consists of the processor state save and restore routines, which may be called from or straight line coded into the beginning and end of the interrupt service routine. These routines are not re-entrant, since they use the dedicated registers. If interrupts may be nested, they must be disabled during these save and restore routines. These routines assume an arbitrary processor state, and save/restore the flags, and the A, H, and L registers in the stack. If it is desired to save the B and C registers in the stack also, the instructions to do this are easily added.

In these subroutines, the D register is dedicated for the interrupt save/restore routines, and may not be used for any other purpose while interrupts are enabled. The E register is designated the stack pointer, and always points to the first available empty cell in memory adjacent to the top of the stack. It is assumed that the stack is wholly contained within a defined page in RAM, designated "PAGE" in the source code; this value is loaded into the H register for all stack operations. All stack operations (except the restore sequence) leave the H and L registers pointing to the top of the stack (i.e. the item last pushed or next to be popped), to facilitate stack arithmetic.

Using the stack for temporary storage effectively compensates for the loss of the D and E registers for general programming.
SOFTWARE STACK SUBROUTINES FOR 8080

0010 PAGE  SET  16 ; PAGE IN RAM OF STACK
0028 ORG  40 ; FOR USE AS RST $0
0028 00 NOP
0029 00 NOP

; STACK PUSH (RE-ENTRANT)

002A 2610 PUSH1:  MVI H, PAGE ; SET PAGE OF STACK
002C 6B MOV L, E ; GET POINTER
002D 1D DCR E ; DECREMENT STACK POINTER
002E 77 MOV M, A ; STORE A IN STACK
002F C9 RET

; STACK POP (RE-ENTRANT)

0030 2610 POP1:  MVI H, PAGE ; SET PAGE OF STACK
0032 6B MOV L, E ; GET POINTER
0033 2C INR L ; POINT TO STACK TOP
0034 7E MOV A, M ; FETCH IT INTO A
0035 5D MOV E, L ; UPDATE STACK POINTER
0036 2C INR L ; LEAVE L AT STACK TOP
0037 C9 RET

; EXCHANGE A WITH STACK TOP (RE-ENTRANT)

0038 2610 XCHA:  MVI H, PAGE ; SET PAGE OF STACK
003A 6B MOV L, E ; GET POINTER
003B 1D DCR E ; DECREMENT STACK POINTER
003C 77 MOV M, A ; SAVE OLD A
003D 2C INR L ; POINT TO OLD TOP
003E 7E MOV A, M ; FETCH IT
003F 6B MOV L, E ; PUSH INTO STACK.
0040 1D DCR E ; (AT NEW TOP)
0041 77 MOV M, A
0042 2C INR L ; RETRIEVE OLD A
0043 7E MOV A, M
0044 2C INR L ; STORE IN OLD TOP
0045 77 MOV M, A
0046 2D DCR L ; RECOVER NEW A
0047 2D DCR L
0048 7E MOV A, M
0049 2C    INR  L    ; DISCARD TEMP CELLS
004A 5D    MOV  E, L    ; AT STACK TOP
004B 2C    INR  L    ; LEAVE L AT STACK TOP
004C C9    RET

; PUSH A, H, L, & FLAGS INTO STACK

004D 55    INTS: MOV  D, L    ; SAVE L IN (DEDICATED) D
004E 6B    MOV  L, E    ; GET STACK POINTER
004F 5C    MOV  E, H    ; SAVE H IN E
0050 2610  MVI  H, PAGE    ; SET PAGE OF STACK
0052 77    MOV  M, A    ; PUSH A
0053 3E00  MVI  A, 0    ; ENCODE FLAGS:
0055 CA5F00 JZ  **+10    ; ZERO
0058 3EA0  MVI  A, 160    ; NONZERO
005A FA5F00 JM  **+5    ; MINUS
005D 3E60  MVI  A, 96    ; PLUS
005F 1F    RAR    ; CARRY (SHIFTED IN )
0060 EA6500 JPE  **+5    ; EVEN PARITY
0063 F608  ORI  S    ; ODD PARITY
0065 2D    DCR  L    ; PUSH INTO STACK
0066 77    MOV  M, A
0067 2D    DCR  L    ; NOW PUSH H
0068 73    MOV  M, E    ; (WAS IN E)
0069 2D    DCR  L    ; *PUSH L*
006A 72    MOV  M, D    ; *REQUIRED ONLY IF NESTED*
006B 5D    MOV  E, L    ; PUT STACK POINTER
006C 1D    DCR  E    ; BACK INTO E
006D C9    RET

; POP A, H, L, & FLAGS FROM STACK

006E 2610  INTR: MVI  H, PAGE    ; SET PAGE OF STACK
0070 6B    MOV  L, E    ; GET POINTER
0071 2C    INR  L    ; *POP L VALUE INTO D*
0072 56    MOV  D, M    ; *OMIT IF NOT PUSHED*
0073 2C    INR  L    ; POP H VALUE INTO E
0074 5E    MOV  E, M
0075 2C    INR  L    ; POP FLAGS INTO A
0076 7E    MOV  A, M
0077 2C    INR  L    ; SET FLAGS
0078 87    ADD  A
0079 7E    MOV  A, M    ; POP A FROM STACK
007A 63    MOV  H, E    ; RE-ARRANGE REGISTERS:
007B 5D    MOV  E, L    ; STACK POINTER TO E FROM L
007C 6A    MOV  L, D    ; H&L FROM E&D
007D C9    RET

0000 END
Symbol Table List Routine

This program will print the user's symbols in alphabetical order followed by the address the 8008 Macro Assembler has assigned to each symbol.

It can be run after pass 1 of the Assembler. The remaining passes, if any, can then be run without rerunning pass 1. Either the V2.0 or the V2.2 equate statements can be set equal to 1 or 0 to assemble the program for use with either version 2.0 or 2.2 of the Intellec 8 Monitor.

ASR 33 TTY, teletype interface to Intellec 8 Mod 8

Intellec 8 Assembler and Monitor

None

After running pass 1 of the Assembler when this routine is called, it will print a table of user's symbol addresses in alphabetical order.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Assembler/Compiler Used:</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>MAC 8</td>
</tr>
<tr>
<td>RAM Required:</td>
<td>Programmer:</td>
</tr>
<tr>
<td>None</td>
<td>Robert Uleski</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>Company:</td>
</tr>
<tr>
<td>177 Bytes</td>
<td>Gilford Instrument Labs</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td>Address:</td>
</tr>
<tr>
<td>4</td>
<td>132 Artino, Oberlin, OH. 44074</td>
</tr>
</tbody>
</table>
;REF. NO. AB30.
;PROGRAM NAME SYMBOL TABLE LIST ROUTINE
;
;
;

3300 ORG 3300H

0001 V20 EQU 1; TRUE FOR MONITOR VER 32.0
0000 V22 EQU 0; TRUE FOR MONITOR VER 2.2

3300 CD9533 CALL CRLF
3303 0E16 MVI C,22
3305 CD933 CALL BLK0;
;
3308 218733 LXI H, MSG;
330B 0E0E MVI C,14

SYM0:

330D 46 MOV B,M
330E CD3B3D CALL INCHL
3311 CD573C CALL TTY
3314 0D DCR C
3315 C20D33 JNZ SYM0
3318 11D01D LXI D, 1DD0H;

LOOP:

331B CD2D33 CALL LOOP0;
331E CD2D33 CALL LOOP0
3321 CD2D33 CALL LOOP0
3324 CD2D33 CALL LOOP0
3327 CD9533 CALL CRLF
332A C31B33 JMP LOOP

LOOP0:

332D 2602 MVI H,D;
332F 7B MOV A,E
3330 C605 ADI 5
3332 6F MOV L,A
3333 7E MOV A,M
3334 FE02 CPI 02H;
3336 C24438 JNZ 3844H;
3339 68 MOV L,E
333A 0E05 MVI C,5

LOOP1:

333C 46 MOV B,M;
333D CD3B3D CALL INCHL
3340 CD573C CALL TTY
3340 0D          DCR C
3341 C23C33      JNZ LOOP1
3342 0620        MVI B, 20H
3343 CDEB3D      CALL INCHL
3344 CD573C      CALL TTY
3345 0620        MVI B, 20H
3346 CD573C      CALL TTY
3347 CDEB3D      CALL INCHL
3348 CD6A33      CALL BYTE;
3349 CD9F33      CALL DECHL
334A CD6A33      CALL BYTE;
334B CD8033      CALL NEX
334C CD733       CALL BLANK;
334D C9         RET

BYTE:
334E A7E1       MOV A, M;
334F 0F98       RRC
3350 0F98       RRC
3351 0F98       RRC
3352 0F98       RRC
3353 0F98       RRC
3354 0F98       RRC
3355 E60F       ANI 0FH
3356 C7A33      CALL HXD
3357 7E         MOV A, M
3358 E60F       ANI 0FH
3359 C7A33      JMP HXD

H XD:
335A CD8C3C      CALL CONV;
335B C3573C      JMP TTY

NEX:
335C 7B         MOV A, E;
335D C608       ADI 8
335E 5F         MOV E, A
335F 0D         RNC
3360 14         INR D
3361 C9         RET

3362 53594D42 MSG:  DB 'SYMBOL TABLE', CR, LF
3363 4F4C2054
3364 41424C45
3365 0D0A

CRLF:
3366 060D       MVI B, CR;
3367 CD573C     CALL TTY

PRINT TWO SPACES
PRINT MSB OF ADDRESS
PRINT LSB OF ADDRESS
PRINT 6 SPACES
PRINT A BYTE AS 2 ASCII CHARS
CALLS MONITOR CONVERT SR
SET D&E TO NEXT SYMBOL
PRINT CR, LF
339A 060A   MVI B, LF
339C C3573C JMP TTY

DECHL:
339F 2D   DCR L; DECREMENT H & L SR
33A0 2C   INR L
33A1 C2A533 JNZ DECL
33A4 25   DCR H

DECL:
33A5 2D   DCR L
33A6 C9   RET

BLANK:
33A7 0E06 MVI C, 6; PRINT 6 BLANKS
33A9 0620 MVI B, 20H; PRINT THE NUMBER OF BLANKS IN
33AB CD573C CALL TTY
33AE 0D   DCR C
33AF C8   RZ
33B0 C3A933 JMP BLK0

3C8C CONV EQU 3C8CH
3C57 TTY EQU 3C57H
3DEB INCHL EQU 3DEBH

ENDIF

IF V20

ENDIF

CONV EQU 3CC1H
TTY EQU 3C5CH
INCHL EQU 3DF0H

ENDIF

000D CR EQU 0DH; ASCII CARRIAGE RETURN
000A LF EQU 0AH; ASCII LINE FEED

0000 END
Diagnostic 1003

To verify that a section of memory is correct by calculating a checksum. The checksum is an ordinary 16-bit sum of the 8-bit bytes contained in the storage region examined.

The routine is explicitly designed to operate entirely in PROM and the CPU without ever using RAM. Thus, the stack cannot be used for anything, including subroutine calls and saving the machine state.

8080 Chip and ROM Chip.

None.

The starting ROM address, the ending ROM address and the location to which the routine exits.

The A register contains zero if the checksum is correct and is non-zero otherwise.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Assembler/Compiler Used:</th>
</tr>
</thead>
<tbody>
<tr>
<td>All except SP</td>
<td>8080 Assembler, Version 3.0</td>
</tr>
<tr>
<td>RAM Required:</td>
<td>Programmer:</td>
</tr>
<tr>
<td>None</td>
<td>M. J. Gralia</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>Company:</td>
</tr>
<tr>
<td>43 bytes</td>
<td>Applied Physics Laboratory</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td>Address:</td>
</tr>
<tr>
<td>Zero</td>
<td>Laurel, Maryland 20810</td>
</tr>
</tbody>
</table>
; REF. NO. AA6
; PROGRAM TITLE DIAGNOSTIC 1003

; DIAGNOSTIC 1003

; PURPOSE:
; TO VERIFY THAT A SECTION OF MEMORY IS
; CORRECT BY CALCULATING A CHECKSUM. THE CHECKSUM
; IS AN ORDINARY 16-BIT SUM OF THE EIGHT-BIT
; BYTES CONTAINED IN THE STORAGE REGION EXAMINED.
; THE ROUTINE IS EXPLICITLY DESIGNED TO
; OPERATE IN PROM AND THE CPU WITHOUT EVER USING
; RAM. THUS, THE SDACK CANNOT BE USED FOR ANYTHING,
; INCLUDING SUBROUTINE CALLS AND SAVING THE MACHINE
; STATE.

; INPUTS:
; THE ROUTINE IS ENTERED VIA A "JUMP"
; INSTRUCTION. THE "CALL" INSTRUCTION MUST NOT BE
; USED BECAUSE OF ITS ACCESS OF RAM.
; THE MEMORY REGION TO BE VALIDATED IS
; CONTAINED IMPLICITLY WITHIN THE PROGRAM; IT
; IS SPECIFIED AT ASSEMBLY TIME.
; A PRE-COMPUTED VALUE FOR THE REGION'S
; CHECKSUM IS ALSO CONTAINED WITHIN THE CODE, AND
; FIXED AT ASSEMBLY TIME.

; OUTPUTS:
; THE ROUTINE TERMINATES VIA A "JUMP"
; INSTRUCTION. THE "CALL/RETURN" INSTRUCTION COUPLE
; MUST NOT BE USED BECAUSE OF ITS ACCESS TO RAM.
; UPON RECEIPT OF CONTROL, THE "A" REGISTER
; IS ZERO IF NO ERRORS WERE DETECTED.
; THE SP REGISTER IS NOT MODIFIED BY THIS
; ROUTINE, BUT ALL OTHER REGISTERS AND FLAGS ARE
; MODIFIED. NOTE: THEY CANNOT BE TEMPORARILY
; SAVED BECAUSE RAM MUST NOT BE USED.

010B

ORG 010BH

; DIAGNOSTIC 1003
; CHECKSUM PROM. DON'T USE RAM.
; INPUTS ALL IMPLICIT
; OUTPUTS: ONLY A=0 IF NO ERRORS, A=0FH IF ERRORS
; REGS:
; A - UTILITY
; BC - MEMORY POINTER
; DE - MEMORY CELL CONTENTS
; HL - CHECKSUM

DI1003:
; INITIALIZE

0100 210000 LXI H, 0: 0 -> SUM
010E 010038 LXI B, START; FIRST MEM LOC -> MEM PNTR
0111 1600 MVI D, 0

DOWHIL:
; DO-WHILE (CURRENT LOCATION) <= (FINISH LOCATION)

0113 3E00 MVI A, <<NOT FINISH AND 0FFH>
0115 81 ADD C
0116 3EC0 MVI A, <<NOT FINISH SHR 8>
0118 88 ADC B
0119 DA2301 JC DOEND

; SUM = SUM + MEMORY<BC>
011C 0A SUM = SUM + MEMORY<BC>
CONT: LDAX B
011D 5F MOV E, A
011E 19 DAD D

; BC = BC + 1
011F 03 INX B
0120 C31301 JMP DOWHIL

DOEND:
; END DO-WHILE

0123 3E65 IF (EXPECT, NE, SUM) THEN A=0FH ELSE A=0
0125 BC MVI A, <<EXPECT SHR 8>
0126 C23301 CMP H
0129 3E4D JNZ THEN
012B BD MVI A, <<EXPECT AND 0FFH>
012C C23301 CMP L
012F AF ELSE: XRA A
0130 C30000 JMP XXXX
0133 3E0F THEN: MVI A, 0FH
0135 C30000 JMP XXXX

; PARAMETERS
65D4 EXPECT EQU 065D4H; EXPECTED CHECKSUM
3FF3 FINISH EQU 03FF3H; LAST MEMORY LOCATION SUMMED
3000 START EQU 03000H; FIRST MEMORY LOCATION SUMMED
0000 XXXX EQU 0H; EXIT LOCATION: NEXT ROUTINE

0000 END
LIST DEVICE TEST PROGRAM

Tests selected list device and its interface.

Minimum system with list device
CRT is used with enclosed program

System monitor

None

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Assembler/Compiler Used:</th>
</tr>
</thead>
<tbody>
<tr>
<td>All except H &amp; L</td>
<td>8080 MDS Macro Assembler</td>
</tr>
<tr>
<td>RAM Required:</td>
<td>Version 1.0</td>
</tr>
<tr>
<td>256 x 8</td>
<td>Programmer:</td>
</tr>
<tr>
<td></td>
<td>Leland B. Myers</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>Company:</td>
</tr>
<tr>
<td>System monitor</td>
<td>North Electric Co.</td>
</tr>
<tr>
<td></td>
<td>Paul H. Henson Research Center</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td>Address:</td>
</tr>
<tr>
<td>None</td>
<td>Box 20345</td>
</tr>
<tr>
<td></td>
<td>Columbus, Ohio 43220</td>
</tr>
</tbody>
</table>
; REF. NO. AAT
; PROGRAM TITLE LIST DEVICE TEST PROGRAM
;
; NORTH ELECTRIC CO., DELAWARE, OHIO
; PAUL H. HENSON RESEARCH CENTER
; L. B. MYERS NOVEMBER 21, 1975
;
; LIST DEVICE TEST PROGRAM
;
;       ORG 030H
;
;       030 1E40  MVI E, 040H ; 64 LINES.
;       032 0E1F  MVI C, 01FH
;       034 060A  LOOP1:  MVI B, TAB ; MARGIN.
;       036 3E20  MVI A, 020H
;       038 CD6100  TABS:  CALL PRINT
;       03B 05  DCR B
;       03C C23800  JNZ TABS
;       03F 0641  MVI B, 041H ; 65 CHAR'S.
;       041 79  MOV A, C
;       042 3C  LOOP0:  INR A
;       043 FE60  CPI 060H
;       045 C24A00  JNZ CONT
;       048 3E20  MVI E, 020H
;       04A 4F  CONT:  MOV C, A
;       04B CD6100  CALL PRINT
;       04E 05  DCR B ; COUNT CHAR'S.
;       04F C24200  JNZ LOOP0
;       052 3E0D  MVI A, 00H ; CR
;       054 CD6100  CALL PRINT
;       057 3E0A  MVI A, 0AH ; LF
;       059 CD6100  CALL PRINT
;       05C 1D  DCR E ; COUNT LINES.
;       05D C23400  JNZ LOOP1
;       060 C7  RST 0 ; FINISHED WHEN E=0.
;       061 57  PRINT:  MOV D, A ; SAVE CHAR.
;       062 DBF7  LOOP:  IN INAD
;       064 E601  ANI MASK
;       066 CA6200  JZ LOOP ; LOOP UNTIL READY.
MOV A, D
OUT OUTAD ; PRINT CHAR.
RET
END
PUNCH BINARY TAPE

To transfer the contents of a given section of memory to paper tape in binary form.

The paper tape format is that used by DATA I/O and other PROM programming devices.

TTY on Port 0 and 1
TTY Printer/Punch on output Port 0
TTY Printer/Punch status on input Port 1

The program contains the necessary TTY output subroutine. This subroutine assumes that the data will be inverted between the accumulator output and the TTY serial output connector, as in the case of the Intellec 8/MOD 80 using the imm8-61 INPUT/OUTPUT card.

Data to be output to paper tape must be stored in consecutive memory locations starting at 100 Hex.

The number of bytes to be punched must be entered in the body of the program at location 0031 Hex.

The default value is 32 decimal or 1F Hex.

The execution of this program will cause a paper tape to be punched by the TTY.

After punching six inches of NULL characters as LEADER the program will punch one ASCII "RUBOUT" to indicate that the next tape location contains the first data byte from memory location 100H. The selected number of bytes will then be punched on paper tape. After the data has been punched the program will cause six inches of NULL characters to be punched as TRAILER.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Assembler/Compiler Used:</th>
</tr>
</thead>
<tbody>
<tr>
<td>A,B,C,H &amp; L</td>
<td>Intellec 8/MOD 80 Macro Assembler</td>
</tr>
<tr>
<td>RAM Required:</td>
<td>Programmer: Rea</td>
</tr>
<tr>
<td>512 Bytes</td>
<td>Robert W. Rea</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>Company: Atlantic Research Corp.</td>
</tr>
<tr>
<td>None</td>
<td>Address: 5390 Cherokee Ave. Alexandria, Va. 22314</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td></td>
</tr>
<tr>
<td>Zero</td>
<td></td>
</tr>
</tbody>
</table>

Register/Compiler Used:
Intellec 8/MOD 80 Macro Assembler

Programmer:
Robert W. Rea

Company:
Atlantic Research Corp.

Address:
5390 Cherokee Ave.
Alexandria, Va. 22314
;REF. NO. AB31
;PROGRAM TITLE PUNCH BINARY TAPE
;
;
;NAME: PUNCH BINARY TAPE
;
;FUNCTION: PUNCH A BINARY TAPE IN FORMAT NEEDED BY CERTAIN PROM PROGRAMMERS
;
;DATA TO BE OUTPUT MUST BE STORED IN CONSECUTIVE LOCATIONS STARTING AT 100H
;
;THE NUMBER OF BYTES TO BE PUNCHED MUST BE ENTERED IN THE BODY OF PROGRAM AT LOCATION 0031
;THE DEFAULT VALUE IS 32 DECIMAL OR 1F HEX
;
;PROGRAM WILL PUNCH SIX INCHES OF NULL CHARACTERS AND THEN PUNCH ONE ASCII RUBOUT CHARACTER IN THE TAPE LOCATION IMMEDIATELY PRECEDING THE FIRST EIGHT BIT BINARY WORD
;
;AFTER THE SELECTED NUMBER OF BYTES HAVE BEEN PUNCHED, THE PROGRAM WILL PUNCH SIX INCHES OF NULLS AS TRAILER AND THEN CAUSE THE CPU TO ENTER THE HALT STATE
;
;START AT 10 HEX
;
0010 ORG 10H
0010 063C MVI B, 60
0012 0E00 LEADR:
0014 DB01 MVI C, 00
316 E504 IN 01
818 C21400 ANI 04
001B 79 JNZ LOOP1
001C 2F MOV A, C
001D CMA

;SET B AS NULL COUNTER
;00 HEX = ASCII NULL
;READ TTY STATUS
;TEST TTY STATUS
;LOOP TIL TTY READY
;MOVE C TO A
;COMPLEMENT A

4-196
001D D300  OUT 00 ;OUTPUT A TO TTY  
001F 05   DCR B ;DECREMENT NULL COUNTER  
0020 C21200 JNZ LEADR ;LOOP TO LEADR IF NOT  
         ;SIXTY NULLS

0023 0EFF MVI C, 0FFH ;SET C TO ASCII RUBOUT  
0025 DB01  LOOP2: IN 01 ;READ TTY STATUS  
0027 E604  ANI 04 ;TEST TTY STATUS  
0029 C22500 JNZ LOOP2 ;LOOP TIL TTY READY  
002C 79   MOV A, C ;MOVE C TO A  
002D 2F   CMA ;COMPLEMENT A  
002E D300 OUT 00 ;OUTPUT A TO TTY

0030 061F  FROM: MVI B, 1FH ;SET BYTE COUNTER  
0032 00   SIZE: NOP ;32 BYTE = 1F HEX  
         ;64 BYTE = 3F HEX  
         ;128 BYTE = 7F HEX  
         ;256 BYTE = FF HEX

0033 210001 LXI H, 100H ;SET H & L TO 100 HEX  
0036 4E   MOV C, M ;MOVE MEMORY TO C  
0037 DB01  LOOP3: IN 01 ;READ TTY STATUS  
0039 E604  ANI 04 ;TEST TTY STATUS  
003B C23700 JNZ LOOP3 ;LOOP TIL TTY READY  
003E 79   MOV A, C ;MOVE C TO A  
003F 2F   CMA ;COMPLEMENT A  
0040 D300 OUT 00 ;OUTPUT A TO TTY

0042 23   PDATA: INX H ;INCREMENT H & L  
0043 4E   MOV C, M ;MOVE MEMORY TO C  
0044 DB01  LOOP4: IN 01 ;READ TTY STATUS  
0046 E604  ANI 04 ;TEST TTY STATUS  
0048 C24400 JNZ LOOP4 ;LOOP TIL TTY READY  
004B 79   MOV A, C ;MOVE C TO A  
004C 2F   CMA ;COMPLEMENT A  
004D D300 OUT 00 ;OUTPUT A TO TTY  
004F 05   DCR B ;DECREMENT BYTE COUNTER  
0050 C24200 JNZ PDATA ;GET ANOTHER CHARACTER  
         ;IF BYTE COUNTER NOT ZERO

0053 063C MVI B, 60 ;SET B AS NULL COUNTER  
0055 0E00   TRALR: MVI C, 00 ;00 HEX = ASCII NULL  
0057 DB01  LOOP5: IN 01 ;READ TTY STATUS  
0059 E604  ANI 04 ;TEST TTY STATUS  
005B C25700 JNZ LOOP5 ;LOOP TIL TTY READY  
005E 79   MOV A, C ;MOVE C TO A  
0060 2F   CMA ;COMPLEMENT A  
0060 D300 OUT 00 ;OUTPUT A TO TTY

0062 05   DCR B ;DECREMENT NULL COUNTER  
0063 C25500 JNZ TRALR ;LOOP TO TRALR IF NOT
SIXTY NULLS

0066 76 HLT
0000 END

PUT CPU IN HALT STATE
MEMORY COMPARE

This program extends the version 3.0 monitors functions to include a memory to memory compare. The program as written occupies locations 3000 to 30FF but may be reassembled to occupy any 256 bytes of memory.

MCS Intellec 8/MOD 80


When started, the program types "C" to show it is active. There are two legal inputs that may be entered at this time, "C" or "H". "C" means the user wants to compare 256 bytes of memory (The C stands for Chip). "H" means the user wants to compare 4096 bytes of memory (The H stands for Whole). The program then waits for two arguments to be entered. The arguments are the address at which the comparisons are to begin. If a mistake is made, type Rubout and the program will return control to the monitor. The arguments may be separated by a space or a comma.

The output results are similar to the results of the compare memory to PROM function as supplied by the monitor. If no differences are found, nothing is printed. If differences are found, the first address followed by its contents followed by the second address followed by its contents are printed. This printout occurs for all locations which do not match.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Maximum Subroutine Nesting Level:</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL (A, B, C, D, E, H, L)</td>
<td>Intellec 8/MOD 80 VER 3.0</td>
</tr>
<tr>
<td>RAM Required:</td>
<td>Assembler/Compiler Used:</td>
</tr>
<tr>
<td>STACK USE ONLY</td>
<td>R. E. Considine</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>Programmer:</td>
</tr>
<tr>
<td>MAY BE PLACED IN 256 BYTE ROM</td>
<td>SRL MEDICAL, INC.</td>
</tr>
<tr>
<td>2 levels</td>
<td>Company:</td>
</tr>
<tr>
<td></td>
<td>2676 Indian Ripple Rd.</td>
</tr>
<tr>
<td></td>
<td>Dayton, Ohio 45440</td>
</tr>
</tbody>
</table>
; TITLE 'MEMORY COMPARE ROUTINE'

3CCD CRLF EQU 3CCDH
3C55 CO EQU 3C55H
3F73 TI EQU 3F73H
3C53 BLK EQU 3C53H
3DBF LADR EQU 3DBFH
3DC7 LBYTE EQU 3DC7H
3D70 EXPR EQU 3D70H

3000 ORG 3000H

3000 CDCCD3C COMPR: CALL CRLF
3003 0E43 MVI C, 'C'
3005 CD553C CALL CO
3007 CD733F CALL TI
300B FE43 CPI 'C'
300D CA1D30 JZ PART
3010 FE48 CPI 'H'
3012 CA2230 JZ WHOLE
3015 0E3F MVI C, '?'
3017 CD553C CALL CO
301A C30030 JMP COMPR
301D 0E01 PART: MVI C, 1
301F C32430 JMP HOLE1

3022 0E10 WHOLE: MVI C, 16
3024 C5 HOLE1: PUSH B
3025 CD3430 CALL GETPR
3028 CD3C30 HOLE2: CALL CM256
302B C1 POP B
302C 0D DCR C
302D C5 PUSH B
302E C22830 JNZ HOLE2
3031 C30030 JMP COMPR

3034 0E02 GETPR: MVI C, 2
3036 CD703D CALL EXPR
3039 D1 POP D
303A E1 POP H
303B C9        RET
303C 0E00    CM256: MVI C, 0
303E C5        CMLUP: PUSH B
303F 7E        MOV A, M
3040 EB        XCHG
3041 46        MOV B, M
3042 EB        XCHG
3043 B8        CMP B
3044 CA6330    JZ NDIFF
3047 CD033C    DIFF: CALL CRLF
304A CDBF3D    CALL LADR
304D CD533C    CALL BLK
3050 7E        MOV A, M
3051 CDC73D    CALL LBYTE
3054 CD533C    CALL BLK
3057 EB        XCHG
3058 CDBF3D    CALL LADR
305B CD533C    CALL BLK
305E 7E        MOV A, M
305F CDC73D    CALL LBYTE
3062 EB        XCHG
3063 23        NDIFF: INX H
3064 EB        XCHG
05 23          INX H
3066 EB        XCHG
3067 C1        POP B
3068 0C        INR C
3069 C23E30    JNZ CMLUP
306C C9        RET

0000        END
TRACE

Provide a program debug trace facility by dumping the address of the calling location and the contents of the registers.

TTY on ports 0 and 1.

The routine calls four TTY utilities in the monitor. However, they are very short and could be included when the monitor is not available.

All of the registers and the first location of the stack (return address).

A header description line and the dump below. For example, if trace were called from location 100H the console output would be:

   *T*   A   F   B   C   D   E   H   L   M   SP
   0103 0A02 1823 0001 5678 0101 0088

Note that the return address is three greater than the calling location and that the M reference is considered 2 bytes.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Assembler/Compiler Used:</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>Intellec 8 macro assembler</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RAM Required:</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>66 bytes</td>
<td>Lorne Douglas</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ROM Required:</th>
<th>Company:</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>Telaid Systems</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Maximum Subroutine Nesting Level:</th>
<th>Address:</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>6725 Variel, Canoga Park, Cal, 91303</td>
</tr>
</tbody>
</table>
; REF. NO. AAS
; PROGRAM TITLE TRACE
;
;
; **********************
; * * TRACE * *
; **********************
;
; THIS ROUTINE DUMPS THE ADDRESS OF
; THE CALLING LOCATION & THE VALUES
; OF THE REGISTERS. IT IS INTENDED
; A DEBUGGING AID POSSIBLY STORED
; IN PROM.
;
; UPON ENTRY ALL OF THE VALUES TO BE
; DUMPED ARE PUSHED INTO THE STACK.
; THE HEADER IS THEN PRINTED AND THE
; ACTUAL VALUES ARE TAKEN FROM THE
; STACK. ALL REGISTERS ARE THEN
; RESTORED AND CONTROL IS RETURNED.
;
; THIS VERSION WAS INTENDED FOR USE
; WITH AN INTELLEC 8/MOD. 80 WITH
; MONITOR VERSION 3.0. THE FORMAT OF
; THE DISPLAY WAS DESIGNED FOR USE
; WITH A CRT WHICH HAS A PRINT LINE
; OF 40 CHR.
;
; THE FOLLOWING ARE MONITOR UTILITIES
; THAT WRITE TO THE CONSOLE. IF USING
; A DIFFERENT MONITOR CHANGE TO
; APPROPRIATE ADDRESSES.
;
3C53   BLK   EQU 3C53H ; WRITE BLANK
3CCD   CRLF  EQU 3CCDH ; END OF LINE
3C55   CO   EQU 3C55H ; WRITE 1 CHR
3DC7   LBYTE EQU 3DC7H ; WRITE HEX BYTE

07E0   ORG 07E0H
07E0   HEAD   EQU $ ; HEADER
07E4   2A542A20 DB {*T* A F B C D} 
07E8   20412046
07EC   32042044
07F0   20452020 DB '{ E H L M SP}'
07F4   48204C20
07F8 204D2020
07FC 20205350
0020 TITCH EQU $-HEAD
; 0030 TRACE EQU $ ; * ENTRY
; 0040 F5 PUSH PSW ; THE DATA TO BE
0041 C5 PUSH B ; DISPLAYED IS ALL
0042 D5 PUSH D ; PUSHED INTO THE
0043 E5 PUSH H ; STACK IN THE
0044 46 MOV B.M ; ORDER OF THE DUMP
0045 23 INX H ; M IS CONSIDERED A
0046 4E MOV C.M ; 16 BIT LOCATION
0047 C5 PUSH B
0048 210C00 LXI H,00CH ; STACK VALUE PRIOR
0049 39 DAD SP ; TO TRACE CALL
004A E5 PUSH H
004B 2B DCX H ; CALLING ADDR
004C E5 PUSH H ; SAVE FOR DUMP
; ; NOW DUMP THE TRACE TITLE LINE
; 0050 F5 0620 MVI B,TITCH ; NUMBER OF CHR
0051 21E007 LXI H,HEAD ; CHR STRING
LOOP:
0054 4E MOV C.M ; PICK UP CHR
0055 CD53C CALL CO ; PRINT IT
0057 05 DCR B ; SEE IF DONE
0058 CA2008 JZ DONE
005A 23 INX H ; ADDR OF NEXT CHR
005B 31408 JMP LOOP
DONE:
; ; 0070 CDCDC3C CALL CRLF ; NEXT LINE
; ; NOW THAT THE TITLE IS PRINTED ITS
; TIME TO DUMP THE DATA IN THE STACK
0073 E1 POP H ; RESTORE POINTER
; 0074 0607 MVI B,7 ; # BYTE PAIRS
0075 05 LOOP2: PUSH B ; SAVE COUNT
0076 7E MOV A.M ; PICK UP FIRST 1
0077 CDC73D CALL LBYTE ; & DUMP IT
0078 2B DCX H ; NEXT 1
0079 7E MOV A.M
007A CDC73D CALL LBYTE
007B 2B DCX H ; 4 NEXT TIME
007C 53CD CALL BLK ; PRINT BLANK
0834 C1   POP B      ; LOOP COUNT
0835 05   DCR B     ; TEST IF DONE
0836 C22608 JNZ LOOP2
0839 CDCD3C CALL CRLF ; NEW LINE

; ALL DONE NOW RESTORE REGISTERS
; AND EXIT

083C E1   POP H      ; THESE 2 SYNC TO
083D E1   POP H      ; THE TRUE H

083E E1   POP H      ; RESTORE`M
083F D1   POP D      ;
0840 C1   POP B      ;
0841 F1   POP PSW    ;
0842 C9   RET        ; ** ADIOS **

; * * * THAT`S ALL FOLKS * * *

; TEST TEST TEST TEST TEST TEST TEST
; ***** ***** ***** ***** ***** ***** *****

0200 ORG 200H
0200 3E0A MVI A, 0AH
0202 012C1B LXI B, 1B2CH
0205 114E3D LXI D, 3D4EH
0208 217856 LXI H, 5678H
020B 310001 LXI SP, 0100H
020E CD0008 CALL TRACE
0211 CD0008 CALL TRACE

0214 C30000 JMP 0

; * * * * * * * * * * * * * * * * * * * * *

0000 END
Handler for Tally paper tape punch

The program TALLY is a handler that interfaces to the 8080 monitor. The Tally punch requires a 4.5 msec drive pulse and has a maximum punch rate of 60 cps. To obtain the long drive pulse the step line of the Tally was wired to the drive direction line of the MDS. The Tally punch has high true logic which requires software inversion of the signals. On entry to the subroutine the C register contains the data to be punched. The data output to device FBH. The drive bit (bit 4) on device F9H is set and a 4.5 msec time delay is generated. The drive bit is removed and a 12.2 msec delay is generated. The data lines are cleared and a return is made. The PSW, and B register pairs are saved on entry and restored before return.

Tally model PD-420 paper tape punch and MDS-800

MDS-800 monitor

C register contains data to be punched

Punch on paper tape

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>None</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assembler/Compiler Used:</td>
<td>Intelsec MDS Assembler Ver 1.0</td>
</tr>
<tr>
<td>RAM Required:</td>
<td>2DH bytes</td>
</tr>
<tr>
<td>Programmer:</td>
<td>S. Graf</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>None</td>
</tr>
<tr>
<td>Company:</td>
<td>Sentrol Systems Ltd.</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td>None</td>
</tr>
<tr>
<td>Address:</td>
<td>4401 Steele Avenue West</td>
</tr>
<tr>
<td></td>
<td>Toronto, Ontario M3N 2S4</td>
</tr>
</tbody>
</table>
; REF. NO. AC10
; PROGRAM TITLE HANDLER FOR TALLY PTP
;
;
; TALLY PUNCH HANDLER
;
;
TALLY:

0000 F5  PUSH  PSW ; SAVE
0001 C5  PUSH  B
0002 79  MOV   A, C
0003 2F  CMA    ; POSITIVE TRUE
0004 D3F8 OUT  0F8H ; OUTPUT DATA
0006 AF  XRA   A ; CLEAR A
0007 D3F9 OUT  0F9H ; SET DRIVE BIT
0009 0602 MVI  B, 2
000B 3EFF UP1: MVI  A, 0FFH
000D 3D  UP2: DCR  A ; DELAY 4.5 MS
000E C20D00 JNZ  UP2
0111 05  DCR  B
0012 C20B00 JNZ  UP1
0015 3E10 MVI  A, 10H ; REMOVE DRIVE BIT
0017 D3F9 OUT  0F9H
0019 0607 MVI  B, 7
001B 3EFF UP3: MVI  A, 0FFH ; DELAY (16.7-4.5) MS
001D 3D  UP4: DCR  A
001E C21D00 JNZ  UP4
0021 05  DCR  B
0022 C21B00 JNZ  UP3
0025 3EFF MVI  A, 0FFH
0027 D3F8 OUT  0F8H ; CLEAR DATA LINES
0029 C1  POP   B ; RESTORE
002A F1  POP   PSW
002B C9  RET

0000    END
CRCC-16

This program generates a 2-byte cyclic redundancy check character, obtained as remainder of the division of the serial data stream by a selected polynomial. This character is stored in register pair B (most significant byte) and C.

8251 as input device.

None

The selected polynomial is given in the form of two constants stored in instructions 410 and 480. The selected polynomial for this case is

\[ x^{16} + x^{15} + x^2 + 1 \]

coming as constants 01000000B, and 00000001B, respectively. Any polynomial of type \( x^{16} + \sum_{i=1}^{15} a_i x^i + 1 \)

(the a's being boolean), has to be introduced as the sequences of bits \( a_1, a_2, \ldots, a_7, \emptyset \) and \( a_8, a_9, \ldots, a_{15} \) as immediate fields of instructions 410 and 480 respectively.

Registers B and C store the check character. After the read-out of one block of data followed by this character, the content of B & C has to be \( \emptyset \).
; REF. NO. AB33
; PROGRAM TITLE CRCC-16

; THIS PROGRAM GENERATES A TWO BYTE CYCLIC
; REDUNDANCY CHECK CHARACTER, OBTAINED AS REMAINDER
; OF THE DIVISION OF THE SERIAL DATA STREAM BY A
; SELECTED POLYNOMIAL
; THE DATA STREAM IS READ BYTE AFTER BYTE THROUGH
; INPUT DEVICE #0 ASSUMED TO BE AN 8251.
; THE CRCC IS STORED IN REGISTER PAIR B (MORE
; SIGNIFICANT BYTE) AND C (L.S. BYTE).
; THE SELECTED POLYNOMIAL IS GIVEN IN THE FORM OF
; IMMEDIATE CONSTANTS STORED IN INSTRUCTIONS
; 410 AND 480
; THE PRESENT PROGRAM SELCTS POLYNOMIAL
; X**16 + X**15 + X**2 + 1
; COMING AS CONSTANTS 40H AND 01H.
; ANY POLYNOMIAL OF TYPE
; X**16 + A(15)*X**15 + .... + A(1)*X + 1
; THE A'S BEING BOOLEAN, HAS TO BE INTRODUCED AS TWO
; CONSTANTS, NAMELY THE SEQUENCE OF BINARY DIGITS
; A(1), A(2), ....... , A(7), 0 AND THE SEQUENCE
; OF BINARY DIGITS A(8), A(9), ...., A(15) AS
; IMMEDIATE FIELD OF INSTRUCTIONS
; 410 AND 480 RESPECTIVELY.

; ORG 100H
0100 010000 ORG 100H
0103 DB000 XI B, 0H ; INITIALIZATION - CLEAR B AND C
0105 1E08 IN 0H ; INPUT DATA
0107 57 MVI E, 0H ; SET BYTE COUNTER
0108 A9 MOV D, A ; CHECK BIT CALCULUS
0109 1F XRA C
010A 78 MOV A, B ; CONDITIONAL JUMP ON CHECK BIT
010B D2101 JNC BITO ; UPDATING DUE TO THE CHOSEN POLYNOM
010E EE40 XRI 40H ; IN MS BYTE CRC REGISTER (B)

0110 37 STC
0111 1F RAR
0112 47 MOV B, A
0113 79 MOV A, C
0114 1F RAR
0115 EE01 XRI 01H ; UPDATING DUE TO THE CHOSEN POLYNOM
}
0117 4F  BITE:  MOV  C, A
0118 7A  MOV  A, D
0119 1F  RAR
011A 1D  DCR  E
011B C20701  JNZ  BITF  ; TO NEW DATA BIT
011E C30301  JMP  W1  ; TO NEW DATA BYTE
0121 1F  BITO:  RAR
0122 47  MOV  B, A
0123 79  MOV  A, C
0124 1F  RAR
0125 C31701  JMP  BITE
0000  END
Intellec MDS to TI Silent 700 Interface

To interface a Ti Silent 700 Terminal with magnetic tape cassettes to an Intellec MDS and permit both the printer and cassettes to run at 30 characters per second speed. Both hardware and software modifications are required to the MDS and hardware changes are required for the TI Silent 700.

Intellec MDS
One (1) MDS-406 6K PROM board
Nine (9) 1702A PROMs
TI Silent 700 ASR Terminal with current loop interface
UPP PROM Programmer or equivalent capability

MDS Monitor Version 1 or Version 2
MDS Boot PROM V1.0 or V1.1

Program Listing Available Only.
Paper Tape is not offered.

Revised 8/8/77

8/8/77
COMPARE

This program reads a HEX format tape and compares it to the corresponding memory locations. When the two don't match, the address is printed on the system console.

Included are three options:

1. Print contents of tape & addr.
2. Skip if address was originally Ø.
3. Restore memory to original value.

8080

Several utility routines available in the MCS 80 monitor were called. These could be substituted or rewritten for use on other systems (see listing).

HEX format tape in reader.

The program lists all locations that have been modified and, optionally, the original and modified values. Also, optionally, the program may restore the modified locations to the original value.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Assembler/Compiler Used:</th>
</tr>
</thead>
<tbody>
<tr>
<td>A,B,C,H,L</td>
<td>Intellec 8 Macro Assembler</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RAM Required:</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>16 BYTES</td>
<td>LORNE DOUGLAS</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ROM Required:</th>
<th>Company:</th>
</tr>
</thead>
<tbody>
<tr>
<td>113 BYTES (can be in RAM)</td>
<td>TELAID SYSTEMS, INC.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Maximum Subroutine Nesting Level:</th>
<th>Address:</th>
</tr>
</thead>
<tbody>
<tr>
<td>16 BYTES IN STACK</td>
<td>6725 Variel Avenue</td>
</tr>
<tr>
<td></td>
<td>Canoga Park, Calif. 91303</td>
</tr>
</tbody>
</table>
; REF. NO. AB34
; PROGRAM TITLE COMPARE
;

; ************************************************************
; * COMPARE *
; ************************************************************

; THIS PROGRAM READS A HEX FORMAT TAPE
; AND COMPARES IT TO THE CORRESPONDING
; MEMORY LOCATIONS. WHEN THE TWO DON'T
; MATCH THE ADDRESS IS PRINTED ON THE
; SYSTEM CONSOLE.
;
0000  NO    EQU  0
0001  YES   EQU  1

; INCLUDED ARE THREE OPTIONS:
; 1 PRINT CONTENTS OF TAPE & ADDR.
; 2 SKIP IF ADDR WAS ORIGINALLY 0.
; 3 RESTORE MEM TO ORIGINAL VALUE.
;
0001  PRNTM  SET  YES
0001  SKIPM  SET  YES
0001  RSTRM  SET  YES

; THE BULK OF THIS PROGRAM IS AN EXACT
; COPY OF THE MONITOR READ ROUTINE. IT
; WAS ASSEMBLED FOR USE ON AN INTELLEC
; 8 MOD 80 RUNNING ON VERSION 4. THERE
; ARE SEVERAL MONITOR UTILITIES CALLED
; THAT COULD BE SUBSTITUTED FOR USE ON
; OTHER SYSTEMS.
;
3D70  EXPR   EQU  3D70H ; GET OFFSET ADDR
3EFF  RIX    EQU  3EFFH ; GET CHR FROM READR
3C82  BYTE   EQU  3C82H ; READ TWO ASCII CHR
;       DECODE TO BINARY
3C55  CO     EQU  3C55H ; PRINT CHR IN C
3C53  BLK    EQU  3C53H ; PRINT BLANK
3CCD  EOL    EQU  3CCDH ; PRINT CR LF
3DBF  LADR   EQU  3DBFH ; PRINT ADDR IN H&L

4-219
3DC7  LBYTE EQU 3DC7H ;CONVERT 1 BINARY
            ; BYTE TO HEX & PRT
386B  START EQU 386BH ;MONITOR RESTART
3C43   LER EQU 3C43H ;ERROR RETURN
            ;
0300   ORG 300H
            ; READ:
0300 0D  DCR C     ; GET ONE ADDR
0301 CD703D CALL EXPR
            ; REDO:
0304 E1  POP H      ; GET BIAS ADDR
0305 E5  PUSH H
0306 CDF3E CALL RIX
0309 063A MYI B, :/
030B 90  SUB B
030C C20403 JNZ REDO ; SCAN TO REC MRK
030F 57  MOV D,A ; CLEAR CHECKSUM
310 CD82C CALL BYTE
0313 CA6003 JZ RED2 ; ZERO RECORD L
            ; LENGTH ALL DONE
0316 5F  MOV E,A ; EC-RECORD LENGTH
0317 CD82C CALL BYTE ; MSB OF LOAD ADR
031F F5  PUSH PSW ; SAVE IT
            ; RED1:
031B CD82C CALL BYTE
            ;** COMPARE INSERT ***************
            ;
            ; IF
0327 B7  ORA A ; TEST TO SEE IF 0
0328 CA5203 JZ ENEDIT ; AND EXIT
            ; ENDIF
032B F5  PUSH PSW ; SAVE BYTE FROM TAP
03C 96  SUB M ; *TEST* IF MATCH
32D CA5003 JZ POPIT ; IF SO ADIOS
            ; THIS CODE IS EXECUTED IF THE LOCATION
0330  C8CD3C  CALL  EOL     ;START A NEW LINE
0333  C8DF3D  CALL  LADR     ;PRINT LOCATION
         IF
0336  CD533C  CALL  BLK     ;PRINT A BLANK
0339  0E54   MVI  C, 'T'     ;PRINT A 'T' TO
033B  CD553C  CALL  CO      ;IDENTIFY TAPE
033E  F1    POP  PSW      ;GET BACK THE CHR
033F  F5    PUSH PSW     ;FROM THE TAPE
0340  4F    MOV  C,A      ;AND PRINT IT
0341  CDC73D  CALL  LBYTE   ;
0344  CD533C  CALL  BLK     ;PRINT A BLANK
0347  0E4D   MVI  C, 'M'     ;PRINT 'M' FOR
0349  CD553C  CALL  CO      ;MEMORY
034C  7E    MOV  A,M      ;PRINT THE CONTENTS
034D  CDC73D  CALL  LBYTE   ;OF MEMORY
         ENDF

0350  F1    POPIT: POP  PSW     ;CHR FROM TAPE
         IF
351  77    MOV  M,A      ;RESTORE VALUE
         ENDF

;ENDIT:

;** END INSERT **********************************************

0352  23    INX  H
0353  1D    DCR  E
0354  C22403  JNZ  RED1     ;LOOP UNTIL DONE
0357  CD823C  CALL  BYTE    ;READ CHECKSUM
035A  C2432C  JNZ  LER      ;CHECKSUM ERROR
035D  C30403  JMP  RED0     ;ANOTHER RECORD

RED2:

0360  CD823C  CALL  BYTE    ;GET MSB TRANSFER
         ;ADDRESS

0363  67    MOV  H,A
0364  CD823C  CALL  BYTE
0367  6F    MOV  L,A
0368  B4    ORA  H
0369  CA6D03  JZ  RED3
036C  E9    PCHL

RED3:

036D  E1    POP  H
03E C36B38  JMP START

0000  END
INPUT and OUTPUT COMMANDS FOR INTELLEC/MDS

These two pseudo monitor commands allow the user to gain console control over "IN" and "OUT" ports in the MDS system. Data can be read into the console using the "I" command; data can be sent out using the "O" command.

Standard MDS system with console.

INTELLEC/MDS Monitor Version 1.0, resident on Prom.

Letter I or O from console input and data:

<i><PORT NUMBER> (CONSOLE RETURNS INPUT PORT DATA)

<o><PORT NUMBER>, <DATA TO BE OUTPUTTED>

The I command prints on the console the value of the chosen input port.

The O command causes the chosen data value to be sent to the output port.

Registers Modified: -

Assembler/Compiler Used:
Intellec 80 MACRO Assembler

RAM Required:
3 BYTES

Programmer:
Paul G. St. Amand

ROM Required:
41 (decimal) BYTES

Company:
Bell Telephone Labs

Maximum Subroutine Nesting Level: -

Address:
Room 3E-25
1600 Osgood Street
No. Andover, MA. 01845
; REF. NO. AB36
; PROGRAM TITLE INPUT & OUTPUT COMMANDS FOR MDS
;
;
; PAUL G. ST. AMAND          BELL TELEPHONE LABS  24 JAN 76
;
; ADDITIONAL MONITOR COMMANDS FOR INTELLEC/MDS SYSTEM
;
; THIS ROUTINE ALLOWS THE USER TO GAIN ACCESS TO THE INPUT AND
; PORTS OF HIS MDS SYSTEM, THROUGH USE OF TWO MONITOR-LIKE COMM
; THE USER=5 EXISTING MONITOR SOFTWARE MUST BE MODIFIED TO ALLO
; THE TWO NEW COMMANDS OF <IN> AND <OUT> TO BE RECOGNIZED.
; THE MONITOR EPROM AT LOCATION 0F800H MUST BE CHANGED AS SHOWN
;
; LOCATION    NEW DATA
; INPUT -I-    F86F    00
;              F870    F7
; OUTPUT -O-    F878    15
;              F87C    F7
;
; THIS WILL CAUSE THE MONITOR TO BRANCH TO THE NEW PROM CODE AT
; APPROPRIATE LOCATIONS.
;
;---------------------------------------------------------------
;
; SYMBOL TABLE FOR LINK POINTS TO MDS MONITOR

FE7F  EXPR  EQU  0FE7FH ; BRING IN DATA FROM CONSOLE
FEAA  LBYTE EQU  0FEAAH ; DATA OUTPUT ROUTINE TO CONSOLE
FECB  PARAM EQU  0FECBH ; BRINGS IN ONE BYTE OF DATA FROM CONSOLE
;
;
; SYMBOL TABLE FOR PROGRAM LOCATIONS

0010  PATCH EQU  10H ; USER RAM, TO BE USED BY ROUTINES.
F700  ROM EQU  0F700H ; FIRST PROM BELOW MONITOR. (ARBITRARY)
;
;---------------------------------------------------------------
;
; F700  ORG  ROM
;
;
; INPUT ROUTINE
;
; THE USER WILL TYPE IN <I><NUMBER>< ). THE SYSTEM WILL THEN
; ANSWER WITH <-ANSWER>. WHERE <NUMBER> IS THE INPUT PORT NUMBER
; IN HEX, AND <-ANSWER> IS THE HEX VALUE OF THE SELECTED INPUT
PORT. FOR EXAMPLE, IF INPUT PORT 08 CONTAINS THE VALUE 79 THE
ROUTINE WILL RETURN:

.108 -79

INPORT:

F700 CD0BFE CALL PARAM ;FETCH THE PORT NUMBER
F703 40 MOV C.L ;SAVE PORT NUMBER IN C REGISTER
F704 211000 LXI H.PATCH ;LOAD RAM PATCH LOCATION
F707 36D8 MVI M.0D8H ;LOAD <IN> COMMAND
F709 23 INX H
F70A 71 MOV M.C ;LOAD PORT NUMBER
F70B 23 INX H
F70C 36C9 MVI M.0C9H ;LOAD <RETURN> COMMAND

RAM WILL NOW HAVE THE PATCH CODE:

IN
PORT NUMBER
RETURN

F70E CD1000 CALL PATCH ;EXECUTE THE CODE
F711 CDAAFE CALL LBYTE ;DISPLAY THE DATA
F714 C9 RET ;RETURN TO MONITOR COMMAND LOOP

OUTPUT PORT ROUTINE

THE USER TYPES IN <0><NUMBER><DATA>. WHEN THE CARRIAGE
RETURN IS ACTIVATED, THE MONITOR WILL OUTPUT TO PORT NUMBER
<NUMBER> THE VALUE <DATA>, WHERE BOTH <NUMBER> AND <DATA> ARE
IN HEX FORMAT.
AN EXAMPLE WOULD BE:

.03C.42
THIS WOULD CAUSE HEX 42 TO BE SENT TO PORT NUMBER 3C.

OUTPORT:

F715 CD7FFE CALL EXPR ;FETCH PORT NUMBER AND DATA
F718 211000 LXI H.PATCH ;LOAD ADDRESS FOR RAM PATCH
F71B 36D3 MVI M.0D3H ;LOAD <OUT> COMMAND
F71D 23 INX H
F71E C1 POP B ;DATA VALUE TO OUTPUT IS IN C REGISTER
F71F D1 POP D ;PORT NUMBER WILL BE IN D REGISTER
F720 73 MOV M.E ;LOAD PORT NUMBER
F721 79 MOV A,C ;MOVE DATA VALUE INTO REGISTER A
22 23 INX H
F723 36C9 MVI M.0C9H ;LOAD <RETURN> INTO RAM
THE RAM PATCH AREA WILL NOW HOLD:

CALL PATCH ; EXECUTE THE CODE
RET ; RETURN TO MONITOR COMMAND LOOP

END
MLOAD

To allow MDS-DOS users to load a binary file (created by the HEXBIN command) using ISIS, at an address modified with a bias. MDS-DOS users cannot currently create, assemble, load, and transfer to PROM a program which uses memory addresses less than 12,000. The reason for this problem is that such programs, when loaded into MDS using the DEBUG command, attempt to overlay the ISIS control program. An error will result, aborting the attempted load. By using MLOAD, such programs may be loaded at an address above 12,000, and then transferred to PROM.

Standard MDS-DOS configuration, with either 1 or 2 disk drives, 32K + memory, and console.

ISIS control program.

MLOAD is initiated by an ISIS-like command, as is EDIT,ASM80, etc.

While system is under ISIS control, and is expecting an ISIS-like command, enter:

\[
\text{MLOAD } \langle \text{filename} \rangle \text{=} \langle \text{address} \rangle \text{H}
\]

where:

\( \langle \text{filename} \rangle \) is a binary file, created by HEXBIN.
\( \langle \text{address} \rangle \) is the biased address of the load in HEX.

The binary file, specified in \( \langle \text{filename} \rangle \) is loaded into MDS RAM, beginning at the address biased by \( \langle \text{address} \rangle \). If no error occurs, control is returned to the MDS Monitor to allow PROM programming. Otherwise, an error message will be output on the system console, and control is returned to ISIS. Error codes are standard ISIS error codes, with the exception of the MLOAD syntax error, which is error code 100.

--SEE ATTACHED NOTE--

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Assembler/Compiler Used:</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEE ATTACHED NOTE</td>
<td>MDS 8080 Macro Assembler</td>
</tr>
<tr>
<td></td>
<td>Programmer:</td>
</tr>
<tr>
<td></td>
<td>William J. Long</td>
</tr>
<tr>
<td>RAM Required:</td>
<td>Company:</td>
</tr>
<tr>
<td>SEE ATTACHED NOTE</td>
<td>Sullivan, Long &amp; Hagerty</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>Address:</td>
</tr>
<tr>
<td>NONE</td>
<td>P.O. Box 2247 Birmingham, AL 35201</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td></td>
</tr>
<tr>
<td>SEE ATTACHED NOTE</td>
<td></td>
</tr>
</tbody>
</table>
MLOAD is designed to operate as an ISIS System command (e.g. EDIT, DEBUG, etc.). MLOAD is therefore called in the same way as an ISIS command, and is provided with parameters in the same way. MLOAD will use about 500 bytes of memory (between addresses 3000 and 31FF Hex.) and will alter all registers. The user must assure that programs loaded by MLOAD will not overlay this area of memory as unpredictable results will occur. After performing the MLOAD function, control is returned to the MDS Monitor if the load is successful, else control is returned to ISIS.

MLOAD does not perform a Relocate function, and the binary file which is loaded is not usually executable. MLOAD simply loads the binary file at an adjusted starting address, usually for the purposes of PROM programming.

No test program is provided because any ISIS binary file will suffice. However, a sample load procedure is shown below.

ABBREVIATED TEST PROGRAM LISTING

001 0000 F3    DI;
002 0001 06FF  MVI B,OFFH;
003 0003 0EFF  STLOP: MVI C,OFFH;
004 0005 0D    STLLP: DCR C;
005 0006 C20500 JNZ STLLP;
006 0009 05    DCR B;

- ETC -

Now suppose that this program was assembled, and the output of the assembly, when passed through HEXBIN was contained in a file called :F1:BIN.FIL. If the user wanted to load this program onto PROM chips, a DEBUG command could not be used. MLOAD would be used in this case, and the following command would be entered on the MDS console:

MLOAD :F1:BIN.FIL$=3200H

This would cause the binary file to be loaded into MDS RAM, beginning at address 3200H (AAAA + BIAS, which in this case, AAAA = 0000H and BIAS = 3200H). Control would then pass to the MDS Monitor, where the following command:

D3200,3209

would produce the following list:

3200 F3 06 FF 0E FF 0D C2 05 00 05

The PROM could then be programmed by entering:

PTX3200,3209,00
**MDS/DOS**

**MEMORY LOAD FROM DISK**

**WITH BIAS OFFSET**

---

```
0000 OPEN EQU 0;
0001 CLOSE EQU 1;
0003 READ EQU 3;
0006 LOAD EQU 6;
000C ERROR EQU 12;
0009 EXIT EQU 9;
0040 ISIS EQU 64;

3200 ORG 03000H;
3203 319F31 START: LXI SP,SPADD;
3205 0E 03 MVI C,READ;
3207 1B430 LXI D,RDDCB;
3209 CD4000 CALL ISIS;
320B 3A7B31 LDA STATUS;
320D E7 ORA A;
320F C25A20 JNZ ERRTN;
3212 2A3E31 LHLD ACTUAL;
3215 EB XCHG;
3218 21BE30 LXI H,BUFFER;
321B 01A31 LXI B,LDFIL;
321C 7E FILNL: MOV A,M;
321D FE24 CPI $'/';
321F C22C30 JNZ GTBIA;
3222 02 STAX B;
3223 23 INX H;
3224 03 INX B;
3225 15 DCR D;
3226 CA6530 JZ STXER;
3229 C31C30 JMP FILNL;
322C 3E20 GTBIA: MOV A,'/';
322E 02 STAX B;
322F 0606 MVI B,006H;
3231 23 INX H;
3232 7E MOV A,M;
3233 FE3D CPI '=';
3235 C25530 JNZ STXER;
3238 C27E30 CALL CONVB;
323B 324331 STA LDBIA+1;
```

LOAD STACK POINTER

READ CONSOLE

JUMP IF ERROR

GET BYTEYTE COUNT

TRANSFER FILE NAME TO LOAD CALL

TEST FOR END OF FILE NAME

STORE IN LOAD CALL FILE NAME BUF

INCREMENT BUFFER ADDRESSES

DECREMENT COUNT

JUMP TO ERROR IF COUNT EXCEEDED

INSERT BLANK AFTER FILENAME

LOAD COUNT

CHECK FOR =

JUMP IF SYNTAX ERROR

CONVERT 2 HIGH ORDER HEX DIGITS

STORE IN LOAD CALL BIAS WORD
303E  CD7630   CALL    CONVB;
3041  324231   STA     LDBIA;
3044  23       INX     H;
3045  7E       MOV     A, M;
3046  FE48     CPI     'H';
3048  C26530   JNZ     STXER;
304B  0E06     MVI     C, LOAD;
304D  114031   LXI     D, LDDCB;
3050  CD4000   CALL    ISIS;
3053  3A7B31   LDA     STATUS;
3056  B7       ORA     A;
3057  CA6D30   J2      FINISH;
305A  0E0C     ERRTN: MVI   C, ERROR;
305C  117B31   LXI     D, ERCDB;
305F  CD4000   CALL    ISIS;
3062  C36D30   JMP     FINISH;
3065  2E64     STXER: MVI   A, 100;
3067  327B31   STA     STATUS;
306A  C35A30   JMP     ERRTN;
306D  0E09     Finish: MVI   C, EXIT;
306F  114031   LXI     D, LDDCB;
3072  CD4000   CALL    ISIS;
3075  76       HLT;
3076  CD4A30   Conv: CALL   GETBI;
3079  17       RAL;
307A  17       RAL;
307B  17       RAL;
307C  17       RAL;
307D  E6F0     ANI    0F0H;
307F  F5       Push   PSW;
3080  CD4A30   CALL    GETBI;
3083  E6F0     ANI    00F0H;
3085  4F       MOV    C, A;
3086  F1       POP    PSW;
3087  B1       ORA    C;
3088  C9       Ret;
3089  F5       CVTHB: Push   PSW;
308A  AF       XRA    A;
308C  57       MOV    D, A;
308C  016A31   LXI    B, CONTA;
308F  F1       POP    PSW;
3090  5F       MOV    E, A;
3091  0A       CVTLP: LDAX    B;
3092  FE47     CPI    'G';
3094  CAA030   JZ     CVRER;
3097  BB       CMP    E;
3098  CAA230   JZ     CVRED;
309B  03       INX    B;
309C  13       INX    D;
309D  C39130   JMP    CVTLP;

CONVERT 2 LOW ORDER HEX DIGITS TO MONITOR, ELSE ERROR
CHECK FOR 'H' AFTER ADDRESS KEY
ERROR IF NOT 'H'
LOAD FILE AND TRANSFER CONTROL
OUTPUT ERROR MESSAGE
JUMP TO EXIT
LOAD ERROR CODE
GET HEX VALUE CONVERT
SAVE
MASK OUT HIGH-ORDER 4 BITS
CONVERSION ROUTINE HEX-TO-BINARY
ZERO BINARY COUNTER
CONVERSION TABLE
GET HEX CHARACTER
GET TABLE VALUE
TEST FOR END OF TABLE
END OF TABLE? JUMP TO ERROR
TEST FOR MATCH
JUMP IF MATCH
INCREMENT TABLE POINTER
INCREMENT BINARY COUNTER
GO BACK AND TRY, TRY AGAIN
LOAD ERROR RETURN CODE
LOAD BINARY VALUE IN ACC.
INCREMENT TABLE ADDRESS
TEST FOR ERROR
JUMP IF ERROR
STACK ADJUST
STACK ADJUST
JUMP TO SYNTAX ERROR ROUTINE
CONSOLE READ AFT
READ BUFFER
BYTE COUNT
POINTER TO FILENAME
BIAS
RETURN SWITCH
ENTRY POINTY SFFTRDD (IGNORED)
FILE NAME
ASCII CONVERSION TABLE
EOF
STACK AREA
HIGH STACK ADDRESS
END;
HEX TAPE LOADER FOR SDK
LOADS HEX ASSEMBLER OBJECT TAPE INTO RAM OF SDK

INTEL MCS-80 SDK, ASR-33 TTY

SDK MONITOR PROM

PAPER TAPE HEX TAPE FORMATTED BY INTEL OR TIMESHARE ASSEMBLERS.

INSTRUCTIONS: 1) LOAD TAPE IN READER

2) TYPE (AFTERMONITOR PROMPT) G 1380 (CR)

3) TURN ON READER

HEX INFORMATION IS LOADED INTO MEMORY AND CONTROL RETURNS TO MONITOR.

A CHECKSUM ERROR CAUSES THE SIGNON MESSAGE TO BE PRINTED. A GOOD LOAD RESULTS IN A MONITOR PROMPT "." FOLLOWED BY SOME "*" ERROR SYMBOLS UNTIL READER IS STOPPED.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Assembler/Compiler Used:</th>
</tr>
</thead>
<tbody>
<tr>
<td>N/A</td>
<td>ISIS 8080 MACRO ASSM., V1.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RAM Required:</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>256 BYTES (KIT SUPPLIED)</td>
<td>A.C. MARSHALL</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ROM Required:</th>
<th>Company:</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDK MONITOR PROM</td>
<td>PROTEON ASSOCIATES, INC.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Maximum Subroutine Nesting Level:</th>
<th>Address:</th>
</tr>
</thead>
<tbody>
<tr>
<td>NA</td>
<td>24 CRESCENT STREET</td>
</tr>
<tr>
<td></td>
<td>WALTHAM, MA 02154</td>
</tr>
</tbody>
</table>
; REF. NO. AB38
; PROGRAM TITLE HEX TAPE LOADER FOR SDK
; A. C. MARSHALL, FEB 4, 1976
; HEX TAPE LOADER FOR SDK
; LOAD INTO MEMORY USING I COMMAND
; POSITION IN LEADER OF '0'S', TYPE I,
; THEN TURN ON READER.
; TO RUN TAPE LOADER TYPE G1380 CR.
; POSITION TAPE WITHIN 1 INCH OF BEGIN
; RECORD MARK THEN TURN ON READER.
; A GOOD LOAD WILL RETURN TO MONITOR, WITH
; SOME '*' FOR EACH BLANK CHAR READ FROM
; TRAILER.
; A CHECKSUM ERROR WILL RETURN THROUGH
; MONITOR SIGNON MESSAGE.
; TO SEE WHAT HAPPENED USE X COMMAND.
; E REG CONTAINS ERROR ON READ.

1380 060A  ENTR:  MVI B,10 ;LEADER LENGTH 1"  
1382 CD1B02  GET:  CALL GETCH ;GET A CHAR  
1385 05  DCR B ;1 "?  
1386 CA2B00  JZ GETCM ;NO RECORD FOUND, RET TO MON  
1389 FE3A  CPI ' ' ;BEGIN REC?  
138B C28213  JNZ GET ;NO, GET ANOTHER  
138E 1E00  MVI E,0 ;CLEAR E REG  
1390 CDB313  CALL INPBY  
1393 A7  ANA A ;CLEAR ACCUM  
1394 CA2B00  JZ GETCM ;RECORD LENG=00, RET TO MON  
1397 42  MOV B,D ;SAVE RECORD SIZE IN B  
1398 CDB313  CALL INPBY ;GET HI ADDR  
139B 62  MOV H,D ;PUT IN H REG  
139C CDB313  CALL INPBY ;GET LO ADDR  
139F 6A  MOV L,D ;PUT IN L REG  
13A0 CDB313  CALL INPBY ;RECORD TYPE, IGNORE  
13A3 05  DCR B ;FINISHED REC?  
13A6 72  MOV M,D ;STORE DATA  
13A8 23  INX H ;INCREMENT MEMORY ADDR  
13A9 C2A313  JNZ DATA ;LAST ONE?  
13AC CDB313  CALL INPBY ;YES, GET CHECKSUM  
13AF CA8013  JZ ENTR ;CHECKSUM IS OK, START OVER  
13B2 CF  RST 1 ;CHECKSUM ERROR RTN TO MON
; SUBROUTINE
13B3 CD1B02 INPBY: CALL GETCH ; GET A CHAR
13B6 4F MOVC,A ; PUT IN A
13B7 CDDA01 CALL CNVBN ; CONVERT TO BINARY
13BA 0F RRC
13BB 0F RRC
13BC 0F RRC
13BD 0F RRC ; MOVE IT TO MSD
13BE 57 MOVD,A ; SAVE FIRST HALF
13BF CD1B02 CALL GETCH ; INP CHAR
13C2 4F MOVC,A ; PUT IN C
13C3 CDDA01 CALL CNVBN ; CONVERT TO BIN
13C6 82 ADD D ; FORM BYTE
13C7 57 MOVD,A ; CALC CKSUM
13C8 83 ADD E ; SAVE NEW CKSUM
13C9 5F MOVE,A ; SAVE CKSUM
13CA C9 RET

; EQUATES

01DA CNVBN EQU 1DAH ; MONITOR ROUTINE
21B GETCH EQU 21BH ; MONITOR ROUTINE
000F LSD EQU 0FFH
002B GETCM EQU 02BH ; MONITOR ROUTINE
0000 END
HEX FORMAT PAPER TAPE DUMP

DUMP SECTIONS OF STORED PROGRAM IN HEX FORMAT (AS IS PRODUCED BY INTEL ASSEMBLER). NO LEADER IS PRODUCED SO THAT MULTIPLE SECTIONS MAY BE CONCATENATED INTO ONE TAPE

NOTE: EACH "." MONITOR PROMPT MUST BE "RUB OUT" OVERPUNCHED TO LOAD WITH ATTACHED LOADER.

INTEL MCS-80 SDK, ASR-33 TTY.

SDK MONITOR PROM

MEMORY START, FINISH ADDRESSES:

INSTRUCTIONS: IN MONITOR

1) TYPE .G 1300 (CR)
    XXXX,YYYY (TURN ON PUNCH) (CR)

HEX FORMAT PAPER TAPE. LEADERS SHOULD BE PRODUCED MANUALLY, OFF LINE.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Assembler/Compiler Used:</th>
<th>Programmer:</th>
<th>Company:</th>
<th>Address:</th>
</tr>
</thead>
<tbody>
<tr>
<td>NA</td>
<td>ISIS 8080 MACRO ASSEMBLER, V1.0</td>
<td>A.C. MARSHALL</td>
<td>PROTEON ASSOCIATES, INC.</td>
<td>24 CRESCENT ST. WALTHAM, MA 02154</td>
</tr>
<tr>
<td>RAM Required:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SDK SUPPLIED (256 BYTE)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ROM Required:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SDK MONITOR PROM</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NA</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
; REF. NO. AB39
; PROGRAM TITLE HEX FORMAT PAPER TAPE DUMP
; 
; A.C. MARSHALL, FEB 4, 1976
; HEX FORMAT FORMAT DUMP ROUTINE
; USES MONITOR IN SDK
; COMMAND IS GI300
; THEN GIVE TWO ADDRESSES IN STD MON
; FORMAT.
; TURN ON PUNCH THEN HIT CR

1300
1300 0E02  DUMP:  ORG 1300H
1302 CD5702  MVI C, 2 ; INIT TO 2 ADDR
1305 D1  CALL GETNM
1306 E1  POP D ; END ADDR
307 CDEE01 DCM05: CALL CROUT ; RETN
130A 0E3A  MVI C, :) ; RECORD FORM
130C CDFA03  CALL CO ; OUTPUT IT
130F 7B  MOV A, E ; LSD OF ADDR END
1310 95  SUB L ; LSD OF PRLSENT START ADDR
1311 FE0F  CPI NEWLN
1313 D24C13  JNC MAX ; >-=?, YES
1316 3C  INR A ; EQU MEANS ONE LOCATION
1317 47  MOV B, A ; SAVE A
1318 AF  FULL:  XRA A ; CALEAR CKSUM
1319 326713  STA TEMP ; CLEAR CKSUM
131C 78  MOV A, B ; RECALL A
131D CD5A13  CALL CKSUM ; OUTPUT REC LENGTH
1320 7C  MOV A, H ; GET HI ADDR
1321 CD5A13  CALL CKSUM ; OUTPUT HI ADDR
1324 7D  MOV A, L ; GET LO ADDR
1325 CD5A13  CALL CKSUM ; OUTPUT LO ADDR
1328 AF  XRA A ; CLEAR A
1329 CD5A13  CALL CKSUM ; OUTPUT REC TYPE
132C 7E  DCM10:  MOV A, M ; GET BYTE
132D CD5A13  CALL CKSUM ; OUTPUT BYTE
1330 CD9C02  CALL HILO ; DONE?
1333 DA4313  JC CKOUT ; YES, FINISH OFF
1336 23  INX H ; NO, NEW ADDR
1337 7D  MOV A, L ; ADDR TEST
338 E60F  ANI NEWLN ; NEW LINE?
133A C22C13  JNZ DCM10 ; NO, NEXT BYTE
133D CD5113 RCEND:  CALL CKEND ; YES, SEND CKSUM
1340 C30713  JMP DCM05 ; DO NEW RECORD
1343 CD5113  CKOUT: CALL CKEND ;SEND CKSUM
1346 CDEE01  CALL CROUT ;
1349 C32B00  JMP GETCM ;DONE REENTER MONITOR
134C 0610  MAX: MVI B,10H ;MAX RECORD SIZE
134E C31813  JMP FULL
1351 3A6713  CKEND: LDA TEMP ;GET CKSUM
1354 2FCMA
1355 3C  INRA ;NEGATE
1356 CDC302  CALL NMOUT ;OUTPUT
1359 09 RET
135A 47  CKSUM: MOV B,A ;SAVE B
135B 3A6713  LDA TEMP ;GET OLD CKSUM
135E 80  ADDB ;ADD NEW TO OLD
135F 326713  STA TEMP ;SAVE NEW CKSUM
1362 78  MOV A,B ;RESTORE A
1363 CDC302  CALL NMOUT ;OUTPUT DIGIT
1366 C9 RET ;RETURN
1367 00  TEMP: DB 0 ;CKSUM STORAGE
0257  GETNM EQU 0257H ;MON ROUTINE
01EE  CROUT EQU 1EEH ;MON ROUTINE
03FA  CO EQU 3FAH ;MON ROUTINE
2C3  NMOUT EQU 2C3H ;MON ROUTINE
029C  HILO EQU 29CH ;MON ROUTINE
000F  NEWLN EQU 0FH ;MASK
002B  GETCM EQU 2BH ;MON ENTRY POINT
0000  END
PAPER TAPE REFORMATTER

REFORMATS THE "DUMP" (D INSTRUCTION) TAPE OF THE 80-SDK MONITOR INTO A COMPATIBLE FORMAT FOR RELOADING WITH THE "INSERT" (I) INSTRUCTION.

MCS-80-SDK ASR-33 OR EQUIVALENT PAPER TAPE TERMINAL.
NO MODIFICATIONS OF KIT NECESSARY.

MCS 80-SDK PROM MONITOR REVISION AS LISTED IN INTEL SDK USERS GUIDE PRELIMINARY EDITION (98-203A)

PAPER TAPE FORMED BY USING D INSTRUCTION WITH LEADER.
PROGRAM LOOKS FOR 2 CR, LF CHARACTERS TO BEGIN EXECUTION.

INSTRUCTIONS: 1) TURN ON PUNCH
               2) LOAD TAPE IN READER
               3) TYPE IN (AFTER MONITOR PROMPT) .G 1300 (CR)
               4) TURN ON READER

PAPER TAPE IS OUTPUT WITH A LEADER OF ASCII "\0". THIS ALLOWS READ AS FOLLOWS:

INSTRUCTIONS: 1) LOAD OBJECT TAPE SET TO NEAR END OF "\0" LEADER
               2) TYPE IN .I
               3) TURN ON READER
               4) REMAINDER OF INSERT INSTRUCTION IS ON TAPE

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Assembler/Compiler Used: ISIS 8080 MACRO ASSEMBLER V1.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>NA</td>
<td>Programmer: A.C. MARSHALL</td>
</tr>
<tr>
<td>RAM Required:</td>
<td>Company: PROTEON ASSOCIATES, INC.</td>
</tr>
<tr>
<td>KIT SUPPLIED 256 Bytes</td>
<td>Address: 24 CRESCENT STREET WALTHAM, MA 02154</td>
</tr>
</tbody>
</table>

| ROM Required:       | Maximum Subroutine Nesting Level: NA                  |
| SDK MONITOR         |                                                       |
; REF. NO. AB40
; PROGRAM TITLE PAPER TAPE REFORMATTER
; 
; A. C. MARSHALL, FEB 4, 1976
; TAPE REFORMATTING ROUTINE
; TAPE IN THE FORM OF A D COMMAND
; FROM THE SDK IS REFORMATTED IN
; THE FORM FOR AN INSERT 'I' COMMAND
; FOR THE MONITOR
; START THE OBJECT TAPE IN THE READER
; AT THE FIRST CHARACTER, WHEN THE 'I'
; FROM THE MONITOR IS RECEIVED, TURN ON THE
; TAPE READER.

01E3 CO EQU 01E3H ;MONITOR ROUTINE
000A LF EQU 0AH ;MONITOR ROUTINE
022B GETCM EQU 02BH ;MONITOR ROUTINE
01EE CROUT EQU 1EEH ;MONITOR ROUTINE
021B CI EQU 021BH ;MONITOR ROUTINE
021B GETCH EQU 021BH ;MONITOR ROUTINE
001B ESC EQU 1BH
000D CR EQU 0DH
0014 LDRLN EQU 20

1300 ORG 1300H
1300 1614 ENTRY: MVI D,LDRLN ;SIZE OF LEADER
1302 0E30 MVI C,'0' ;ZERO FOR LEADER
1304 CDE301 LEADER: CALL CO ;OUTPUT LEADER
1307 15 DCR D ;DONE?
1308 C20413 JNZ LEADER ;NO LOOP
130B CD1B02 START: CALL GETCH ;INPUT CHARACTER
130E FE0A CPI LF ;LINE FEED?
1310 C20813 JNZ START ;NO, GET ANOTHER
1313 CD1B02 CALL GETCH ;RECORD
1316 CD1B02 CALL GETCH ;/LENGTH, IGNORE
1319 1E06 MVI E,6 ;6 CHAR
131B 1604 MVI D,4 ;YES, GET ADDR.
131D CD7113 ADRI: CALL CARECH ;OUTPUT CHAR
1320 15 DCR D ;ALL 4?
1321 C21D13 JNZ ADRI ;NO, REPEAT
1324 CD1B02 CALL GETCH ;BLANK- OMIT
327 0E0D MVI C,CR ;SET UP CR
329 CDE301 CALL CO ;OUTPUT IT
132C CD7113 DATA: CALL CARECH ;DIGIT 1
132F CD7113 CALL CARECH ;DIGIT 2
1332 3E0D    MVI    A, CR   ;BLANK - OMIT
1334 3E0D    MVI    A, CR   ;3RD CHAR IN A
1336 B9    CMP    C     ;CR?
1337 C22C13  JNZ    DATA   ;NO, REPEAT
133A 0E2C   MVI    C, \   ;YES, LINE DONE
133C CDE301  CALL   GETCH   ;LINE FEED - NO OUT
133F CD1B02  CALL   GETCH   ;HX DGT OR 
1342 CD1B02  CALL   GETCH   ;HX DGT OR 
1345 3E2E   MVI    A, \    ;CHECK END
1347 B9    CMP    C     ;END?
1348 C26113  JNZ    NXTLN  ;NO, DO AGN
134B 0E1B   MVI    C, ESC  ;YES
134D CDE301  CALL   CO     ;OUTPUT ESC
1350 0E00    ENDTP:  MVI    C, 0   ;BLANK FOR TRAILER
1352 1614   MVI    D, LDRLN  ;SIZE OF LEADER
1354 CDE301  TRAIL:  CALL   CO     ;OUTPUT BLANK
1357 15    DCR    D     ;DONE?
135B C25413  JNZ    TRAIL  ;NO, CONTINUE
135F CDEE01  CALL   CROUT  ;YES, RETURN
135E C32B00  JMP    GETCM  ;TO MONITOR
1361 1604   NXTLN:  MVI    D, 4   ;SET COUNTER TO 4
1363 1D    DCR    E     ;DONE 5 REC?
1364 C95013  JZ     ENDTP   ;YES END
1367 CD1B02  NADR:  CALL   GETCH  ;NO, GET A CHAR
136A 15    DCR    D     ;DONE?
136B C26713  JNZ    NADR   ;NO, LOOP
136E C52C13  JMP    DATA   ;DO NEXT RECORD

;SUBROUTINES:

1371 CD1B02  CARECH: CALL   GETCH   ;GET CHAR
1374 CDE301  CALL   CO     ;ECHO IT
1377 C9    RET     ;RETURN

0000    END
DATA IO PROM TAPE PROCESSOR V1.0 DATA IO

CREATE, EDIT AND PUNCH PAPER TAPE IN THE BINARY FORMAT REQUIRED FOR DATA IO PROM PROGRAMMERS

CONSOLE, INTELLEC MDS-800, PAPER TAPE READER, PAPER TAPE PUNCH

MDS-800 MONITOR

HEX INPUT FROM CONSOLE, HEX OR BINARY TAPE INPUT

PAPER TAPE IN BINARY IMAGE FORMAT, AND LISTINGS

Program offered on diskette only.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Assembler/Compiler Used:</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL</td>
<td>ASM80 MACRO ASSEMBLER</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RAM Required:</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>FCBH - 4043D</td>
<td>J. WILLOTT</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ROM Required:</th>
<th>Company:</th>
</tr>
</thead>
<tbody>
<tr>
<td>NONE</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Maximum Subroutine Nesting Level:</th>
<th>Address:</th>
</tr>
</thead>
<tbody>
<tr>
<td>LESS THAN 6</td>
<td>1487 BONGATE COURT</td>
</tr>
<tr>
<td></td>
<td>SAN JOSE, CA. 95130</td>
</tr>
</tbody>
</table>
Program Title

CENTRONICS  Assembler oriented
306 Line Printer Handler and Error Only Assembly

Function

Allows assemblies to be performed on Centronics 306
Line Printer without top of form option.

Two listings are available  1) errors only
                              2) normal

Required Hardware

See attached for interface detail.

Line printer on output port 7
       input port 4

Required Software

Normal 8080 Monitor and Assembler

Input Parameters

G 37E0    Error Only Listing
B 37F0    Final Listing

These can be set or changed at any time by re-entering the
monitor with a console reset.

Output Results

The routine can be modified to provide both listings on
a teletype by altering the following instructions and
re-assembling.

Line 15   ANI 80H to ANI 04H
          29 Status EQU 4 to Status EQU 1
          30 LPRT EQU 7 to LPRT EQU 0

Delete lines 20 thru 25  ADI 80H to out LPRT
Replace LINED:  RET with          LINEND:  MVI A,0AH
                                             OUT LPRT
                                             RET

Registers Modified:

Assembler/Compiler Used:

8080

RAM Required:

1 page

Programmer:

R. Perry and D. Rush

ROM Required:  *may be located in

1 page  RAM if desired

Company:

Westinghouse Transportation

Maximum Subroutine Nesting Level:

Address:  2001 Lebanon Road
          West Mifflin, PA  15122
Memory Test 8080

To perform a functional test of the ram memory within a reasonable time. Using modified walking ones and zeroes test pattern.

MDS-800 with List output device... May use console.

MDS-800 monitor list output subroutine for printing error messages.

4 equate card describing the dimensionality of the memory chips, the start/end address of memory to be tested, the origin of the program and the address of the character output subroutine.

Returns to monitor if no errors... else prints each memory location in error, the data received from memory, and the test data written. Test time for 4K bytes of memory is one minute.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Assembler/Compiler Used: ISIS 8080</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>Macro Assembler, V1.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RAM Required:</th>
<th>Programmer: S. G. Thompson</th>
</tr>
</thead>
<tbody>
<tr>
<td>286 Bytes</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ROM Required:</th>
<th>Company: Harris Controls</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel MDS-800 Monitor I/O Subroutines</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Maximum Subroutine Nesting Level:</th>
<th>Address: P. O. Box 430</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 + 4 pushes</td>
<td>Melbourne, Florida 32901</td>
</tr>
</tbody>
</table>
; REF. NO. A49
; PROGRAM TITLE MEMORY TEST 8080

MEMORY TEST

1000 MEMS EQU 1000H ;MEMORY START ADR FOR TESTING
4000 MEME EQU 4000H ;MEMORY END ADR

0040 MDEM EQU 64 ; MEMORY CHIP DIMENSIONALITY 64 BY 64
0000 DATAS EQU 000H ;STARTING MEMORY DATA PATTERN (BACKGROUN
0041 MPINC EQU MDEM+1 ; POINTER INC. VALUE FOR NEXT ALT P

MDS-800 SUBROUTINES USED
L0 (LIST OUTPUT I/O)
START (OF MONITOR)

30F L0 EQU 0F80FH
+826 START EQU 0F826H ; MONITOR ADR. . STACK RESET

0100 ORG 100H ;PGM START ADR
0100 311E02 LXI SP, STACK ; SET STACK ADR.
0103 0E00 MVI C, 0 ;FIRST TIME FLAG
0105 0600 MVI B, DATAS ;B=DATA
0107 NPAT EQU $ ;DO NEXT PATTERN
0107 AF XRA A ;CLEAR OFSET
0108 32EB01 STA OFSET
010B 210810 BEGIN: LXI H, MEMS ; HL= MEM START ADR
010E CD5001 CALL BASE ; GO TO CALC BASE ADR FOR READ WRIT
0111 CD5C01 WRITE: CALL WCHK ;WRITE PROPER PATTERN AT THIS ADR.
0114 23 INX H ;NEXT MEM ADR
0115 7C MOV A, H ;CHECK FOR DONE WITH WRITE
0116 FE40 CPI MEME/256 ; HIDH ADR. CHECK
0118 C21101 JNZ WRITE ; LOOP TIL DONE
011B 7D MOV A, L
011C FE00 CPI MEME-(MEME/256)*256 ; LOW
011E C21101 JNZ WRITE
0121 210010 LXI H, MEMS ; BACK TO START FOR READ
0124 CD5001 CALL BASE ;GO CALC BASE ADR FOR READ
0127 CD6F01 READ: CALL RCHK ; READ & CHECK FOR OK
012A 23 INX H
012B 7C MOV A, H ;CHECK FOR DONE WITH READ
~12C FE40 CPI MEME/256 ; HIDH ADR. CHECK
12E C22701 JNZ READ
0131 7D MOV A, L
0132 FE00 CPI MEME-(MEME/256)*256 ; LOW
0134 C22701    JNZ    READ
            ; CHECK FOR DONE . . OFSET=MDEM
0137 3AE0B1    LDA    OFSET
013A 3C        INR    A
013B 32E0B1    STA    OFSET
013E FE41      CPI    MDEM+1
0140 C20B01    JNZ    BEGIN    ; CONTINUE TILL DONE
0143 53        NXT    EQU    $   
0143 79        MOV    A,C
0144 B7        ORA    A
0145 C226F8    JNZ    START    ; BACK TO MONITOR
0148 78        MOV    A,B
0149 2F        CMP    
014A 47        MOV    B.A     ; DATA NOW COMPLEMENTED
014B 0EFF      MVI    C,0FFH    ; SET SECOND TIME FLAG
014D C30701    JMP    NPAT    ; GO TO NXT PATTERN

; SUBROUTINE TO CALC BASEADR. FOR ALT PATTERN START
0150 110010    BASE    EQU    $   
0150 3AE0B1    LDA    OFSET
0153 83        ADD    E
0155 5F        MOV    E.A     ; MEM START
0158 AF        XRA    A
0159 8A        ADC    D     ; ADD IN ANY CARRY
015A 57        MOV    D.A     ; RESTORE
015B C9        RET

; SUBROUTINE TO WRITE DATA PATTERN INTO MEMORY
015C 7C        WCHK    EQU    $   
015C 7C        MOV    A.H     ; CHECK FOR ALTERNATE DATA PATTERNADR
015D BA        CMP    D
015E C26D01    JNZ    W2     ; NOT ADR... WRITE BACKGROUND
0161 7D        MOV    A.L
0162 BB        CMP    E
0163 C26D01    JNZ    W2     ; NOT ADR... WRITE BAKGND ELSE...
0166 78        MOV    A.B     ; COMPLEMENT PATTERN AND...
0167 2F        CMA
0168 77        MOV    M.A     ; WRITE
0169 CD9301    CALL    INCD    ; INC ALT DATA PATTERN POINTER
016C C9        RET
016D 78        W2:    MOV    M.B     ; STORE BACKGROUND DATA PATTERN
016E C9        RET

; SUBROUTINE TO READ AND CHECK MEMORY DATA
016F 7C        RCHK    EQU    $   
016F 7C        MOV    A.H     ; CHECK ADR
0170 BA        CMP    D

4-253
0171 C28A01  JNZ R2
0174 7D  MOV A, L ; B = CALC. DATA WRITTEN
0175 BB  CMP E ; A = DATA FROM MEMORY
0176 C28A01  JNZ R2
0179 C5  PUSH B ; GET DATA
017A 78  MOV A, B
017B 2F  CMP A, B ; CHECK CELL FOR ERROR
017C 47  MOV B, A
017D 7E  MOV A, M
017E B8  CMP B ; GO TO PRINT OUT
017F CA8501  JZ $+6 ; RESTORE REGS
0182 CD9B01  CALL ERROR ; INC. ALT DATA PATTERN POINTER
0185 C1  CALL INCD ; INC. ALT DATA PATTERN POINTER
0186 CD9301  CALL INC ; INC. ALT DATA PATTERN POINTER
0189 C9  RET ; IF ERROR PRINT ELSE BACK TO CALLER
018A 7E  R2: MOV A, M
018B B8  CMP B
018C CA9201  JZ $+6
018F CD9B01  CALL ERROR
0192 C9  RET

; SUBROUTINE TO INC. ALT DATA PATTERN POINTER
0193 INCD EQU $; SAVE HL
0193 E5  PUSH H
0194 214100 LXI H, MPINC ; GET INC. VALUE
0197 19  DAD D ; HL <= ALT. PATTERN POINTER + INC.
0198 EB  XCHG ; BACK IN DE
0199 E1  POP H ; RESTORE HL
019A C9  RET

; MEMORY ERROR PRINTING SUBROUTINE
; LINE FORMAT:
; AAAAA XXM YYT
; XX= MEMORY DATA IN ERROR
; YY= TEST DATA PATTERN
; AAAAA= ADDRESS OF MEM FAILURE
; A, X, Y ALL IN HEX
; ERROR EQU $
019B C5  PUSH B ; BC SAVED
019C F5  PUSH PSW ; ACC SAVED
019D 0E0D MVI C, $0DH ; CARRIAGE RET
019F CD0FF8 CALL L0
01A2 0E0A MVI C, $0AH ; LINE FEED
01A4 CD0FF8 CALL L0
01A7 CDC701 CALL DADR ; MEM ADR PRINT
01AA 0E20 MVI C, $ ; BLANK
01AC CD0FF8 CALL L0
01AF F1  POP PSW
01B0 CDCF01 CALL DBYTE ; PRINT MEM DATA IN ERROR
01B3 0E4D  MVI  C, 'M'
01B5 CD0FF8 CALL  L0   ;FLAG AS 'M'
01B8 0E20  MVI  C, '/'  ;BLANK
01BA CD0FF8 CALL  L0
01BD C1    POP  B
01BE 78    MOV  A, B   ;PRINT DAT A WRITTEN
01BF CDCF01 CALL  DBYTE
01C2 0E54  MVI  C, 'T'   ;T = TEST PATTERN
01C4 C30FF8 JMP  L0

; DUMP MEMORY ADR REG TO L0

01C7 DADR  EQU  $
01C7 7C    MOV  A, H
01C8 CDCF01 CALL  DBYTE
01CB 7D    MOV  A, L
01CC C3CF01 JMP  DBYTE

; DISPLAY ON L0 DATA BYTE IN ACC
; REG-C DESTROYED

01CF F5  DBYTE:  PUSH  PSW
01D0 0F    RRC
01D1 0F    RRC
01D2 0F    RRC
01D3 0F    RRC
01D4 CDE101 CALL  CONV   ;CONVERT TO HEX ASCII
01D7 CD0FF8 CALL  L0   ;PRINT
01DA F1    POP  PSW   ;RESTORE LSD
01DB CDE101 CALL  CONV   ;CONV TO ASCII HEX
01DE C30FF8 JMP  L0

; CONVERT 4 BIT BINARY INTO ASCII HEX
; 4 BITS IN LSD OF ACC, ASCII IN REG C

01E1  CONV  EQU  $
01E1 E60F  ANI  0FH
01E3 C690  ADI  90H
01E5 27    DAA
01E6 CE40  ACI  40H
01E8 27    DAA
01E9 4F    MOV  C, A
01EA C9    RET
01EB OFFSET EQU  $
01EB  DS  1   ; ALT PATTERN BASE ADR. <= OFFSET+ME
01EC  DS  50
021E STACK  EQU  $
0300  END
SYM

SYMBOL TABLE DUMP FOR INTELLEC 8/80

INTELLEC MCS, PRINTER (CONSOLE OR LIST DEVICE)

INTEL ASSEMBLER VERSION 4.0

SYMBOL TABLE AS LEFT AFTER AN ASSEMBLY

THE ASSEMBLED SYMBOLS FOLLOWED BY THE ASSIGNED ADDRESS (VALUE). THE OUTPUT IS SIMILAR TO THE MDS ASSEMBLER SYMBOL TABLE DUMP.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Assembler/Compiler Used:</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL</td>
<td>INTEL MACRO ASSM /4.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RAM Required:</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>NONE</td>
<td>LORNE DOUGLAS</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ROM Required:</th>
<th>Company:</th>
</tr>
</thead>
<tbody>
<tr>
<td>630 BYTES</td>
<td>TELAID SYSTEMS</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Maximum Subroutine Nesting Level:</th>
<th>Address:</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>6725 Varuel</td>
</tr>
<tr>
<td></td>
<td>Canoga Park, CA. 91303</td>
</tr>
</tbody>
</table>
; REF NO. AB42
; PROGRAM TITLE SYMBOL TABLE DUMP FOR MOD 80
;
;
;
;
;
;
;
;
;
;
;
;
;
;
;
;
;
;
;

*  
*** SYMBOL TABLE DUMP FOR INTELLECT  
*

; THIS ROUTINE PRINTS THE SYMBOL TABLE  
; AFTER AN ASSEMBLY IS COMPLETE. THIS  
; PARTICULAR VERSION RUN UNDER MONITOR  
; VERSION 3.0 AND ASSEMBLER VERSION 4.0.  
; THIS PROGRAM HAS BEEN ASSEMBLED TO  
; RUN IN RAM AND I HAVE MODIFIED THE  
; BRANCH TABLE TO PRINT SYMBOLS BY  
; TYPING V UNDER THE MONITOR.  
;
;
; TO ADAPT TO ANOTHER ASSEMBLER OR  
; MONITOR ONE WOULD HAVE TO FIND THE  
; START OF THE SYMBOL TABLE IN RAM AND  
; SUBSTITUTE THE UTILITY ROUTINE ADDR.  
;
;
;
3368  ORG 3368H ;FOR PROM
3368 218E1D
336B C0CD3C
336E 16FD

SYM:
   LXI H,1D8EH;STARTING ADDR OF
   ;SYMBOL TABLE
   CALL CRLF ;NEW LINE TO LIST
   MVI D,-3 ;3 SYMBOLS PER LINE
   CALL PRINT ;PRINT SYMBOL CHR
   CALL BLK ;
   INX H ;INCREMENT TO HEX
   INX H ;VALUE AND PRINT
   MOV A,M ;
   CDC73D CALL LBYTE ;
   JC2B DCX H ;
   MOV A,M ;
   CDC73D CALL LBYTE ;
3381 23  INX H ; NOW INCR TO NEXT
3382 23  INX H ; SYMBOL
3383 CD533C  CALL BLK ;
3386 CD533C  CALL BLK ;
3389 CD533C  CALL BLK ;
338C 14  INR D ;
338D FA7033  JM NXTSYM ;
3390 C36B33  JMP NXTLN
;
PRINT:
3393 1EFB  MVI E,-5 ; 5 CHR PER SYMBOL
PR2:
3395 7E  MOV A, M ; TEST FOR END OF
3396 FE8D  CPI 0DH ; TABLE
3398 C29E33  JNZ PR3 ;
339B C36B38  JMP START ; DONE - ADIOS
PR3:
339E 4E  MOV C, M ; PRINT SYMBOL
339F CD553C  CALL CO ;
33A2 23  INX H ; NEXT CHR
33A3 1C  INR E ; CHR COUNT
33A4 C8  RZ ;
33A5 C39533  JMP PR2 ;
;
** MONITOR UTILITIES
;
3C53  BLK  EQU 3C53H ; PRINT 1 BLANK
3C55  CO  EQU 3C55H ; PRINT 1 ASCII CHR
3CCD  CRLF  EQU 3CCDH ; END OF LINE SEQUENCE
3DC7  LBYTE  EQU 3DC7H ; PRINT HEX BYTE
386B  START  EQU 386BH ; MONITOR RESTART
;
0000  END
MON256

Provides the most commonly used monitor debug functions in a single 256-byte EPROM: Go To, Substitute, Display, Hex Arithmetic, Find Byte, and Move.

As assembled requires INTELLEC 8/MOD 80 with TTY on ports 0 and 1. Can easily be reassembled with different input/output assignments for use with any user designed hardware which has ASCII I/O capability.

None

ASCII Characters 0 – 9, A – F, G, H, M and S. Addresses must be entered as four characters. Bytes must be entered as two characters. Leading zeros are significant.

ASCII characters corresponding to hexadecimal memory addresses and contents.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Assembler/Compiler Used:</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL</td>
<td>8080 MACRO ASSEMBLER, VER 3.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RAM Required:</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>11 BYTES FOR STACK</td>
<td>FRANK FAFF</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ROM Required:</th>
<th>Company:</th>
</tr>
</thead>
<tbody>
<tr>
<td>256 BYTES</td>
<td>ATLANTIC RESEARCH CORP.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Maximum Subroutine Nesting Level:</th>
<th>Address:</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>5390 CHEROKEE AVENUE ALEXANDRIA, VA. 22314</td>
</tr>
</tbody>
</table>
CHARACTER INTERPRETED MEMORY DUMP

Prints contents of selected portion of memory. Each line contains beginning address followed by 16 bytes in hex format and same 16 bytes repeated in character format. Minimum dump contains 256 bytes. All dumps are spaced to place three 256 byte blocks on an 11 inch page with column identification at the top of each page.

Intellec 8/Mod 80 with TTY or equivalent.

Intellec Monitor Version 2.0

Program prompts for two addresses with an '@'. Addresses are entered in hex separated by a comma or blank as in the Monitor dump command.

Prints memory in 256 byte blocks as described above.

Registers Modified: Monitor is reinitialized at exit.

RAM Required: Program requires 242 bytes. Will execute in either

ROM Required: RAM or ROM.

Maximum Subroutine Nesting Level:

Assembler/Compiler Used: ISIS 8080 Macro Assembler, V1.0

Programmer: Jim Squires

Company: Allan Hancock College

Address: Santa Maria, Ca 93454
; REF. NO. AB44
; PROGRAM TITLE CHARACTER INTERPRET MEMORY DUMP
; ;
; MEMORY DUMP WITH CHARACTER INTERPRET
; DEFINE EXTERNAL REFERENCES
;
3D6B  EXPR       EQU 3D6BH
3C8  CRLF       EQU 3C8H
3DBA  LADR      EQU 3DBAH
3C4E  BLK        EQU 3C4EH
3DC2  LBYTE      EQU 3DC2H
3D9F  HILO       EQU 3D9FH
386B  START      EQU 386BH
3C50  CO         EQU 3C50H
3C3E  LER        EQU 3C3EH
;
; INITIALIZATION
;
100  DUMP: ORG 7000H  ; ENTRY POINT
7000  0E40  MVI C, '0'  ; GET PROMPT CHAR
7002  CD503C  CALL CO  ; PRINT IT
7005  CD4E3C  CALL BLK  ; AND A BLANK
7008  0E02  MVI C, 2  ; SET UP FOR TWO ADDRS
700A  CD6B3D  CALL EXPR  ; GET TWO ADDRESSES
700D  D1  POP D  ; ENDING ADDR
700E  E1  POP H  ; START ADDR
700F  3EF0  MVI A, 0F0H  ; STRIP RIGHT
7011  A5  ANA L  ; 4 BITS
7012  6F  MOV L, A  ; FROM START ADDR
7013  0E02  MVI B, 2  ; SKIP TWO LINES
7015  CD9970  CALL PB0TM  ; TO GET STARTED
;
; PAGE LOOP
;
7018  E5  DUMP2: PUSH H  ; SAVE DUMP POINTER
7019  0607  MVI B, 7  ; 7 LINE SKIP
701B  CDA270  CALL LSKIP  ; DO IT
701E  21AA70  LXI H, HEAD  ; GET ADDR OF PRINT LINE
7021  0649  MVI B, 73  ; GONNA PRINT 73 CHAR
7023  4E  DUMP3: MOV C, M  ; GET NEXT CHAR
7024  CD503C  CALL CO  ; PRINT IT
7027  23  INX H  ; BUMP TO NEXT CHAR
7028  05  DCR B  ; DECREMENT COUNTER
7029  C22370  JNZ DUMP3  ; DO ANOTHER IF NEED BE
72C  E1  POP H  ; RESTORE DUMP POINTER
;
; PRINT A PAGE OF MEMORY

4-264
702D CDC83C   CALL CRLF ; EMPTY LINE
7030 0639   MVI B,57 ; 57 LINES TO PAGE BOTTOM

; PRINT A LINE OF MEMORY (16 BYTES)

7032 CD8A3D   DUMP5: CALL LADR ; PRINT THE ADDR
7035 E5   PUSH H ; REMEMBER STARTING BYTE

7036 CD4E3C   DUMP6: CALL BLK ; A BLANK SPACE
703E 7E   MOV A,M ; GET NEXT BYTE
703A CDC23D   CALL LBYTE ; PRINT AS HEX
703D 23   INX H ; BUMP POINTER
703E 3E0F   MVI A,0FH ; CHECK FOR MULTIPLE OF 16
7040 A5   ANA L ;
7041 C23670   JNZ DUMP6 ; JUMP NO
7044 CD4E3C   CALL BLK ; PRINT A BLANK
7047 CD4E3C   CALL BLK ; AND ANOTHER
704A 0E2A   MVI C, '*' ; GET AN ASTERISK
704C CD503C   CALL CO ; PRINT IT
704F E1   POP H ; GET BEGINNING ADDR AGAIN

; INTERPRET BYTES

7050 4E   DUMP7: MOV C,M ; GET NEXT BYTE
7051 3E1F   MVI A,1FH ; IF BLANK OR HIGHER
7053 B9   CMP C ; THEN OK
7054 DA5970   JC DUMP8 ; ELSE MAKE IT BLANK
7057 0E20   MVI C, '/' ; IF '/' OR LESS
7059 3E5C   DUMP8: MVI A, '\' ; IF '\' OR LESS
705B B9   CMP C ; THEN OK
705C D26170   JNC DUMP9 ; ELSE FOR A BLANK
705F 0E20   MVI C, '*' ; PRINT THE CHAR
7061 CD503C   DUMP9: CALL CO ; PRINT THE CHAR
7064 23   INX H ; POINT TO THE NEXT ONE
7065 3E0F   MVI A,0FH ; CHECK FOR END OF LINE
7067 A5   ANA L ;
7068 C25070   JNZ DUMP7 ; JUMP NOT YET
706B 0E2A   MVI C, '*' ; GET AN ASTERISK
706D CD503C   CALL CO ; PRINT IT
7070 CDC83C   CALL CRLF ; DECREMENT LINE COUNTER
7073 05   DCR B ;

; CHECK FOR 256 BYTE BREAK

7074 7D   MOV A,L ; GET LOW BYTE
7075 B7   ORA A ; SET THE FLAGS
7076 C22370   JNZ DUMP5 ; JUMP NO BREAK
7079 CDC83C   CALL CRLF ; ELSE A BLANK LINE
707C 05   DCR B ; DECREMENT LINE COUNTER
707D 2B     DCX H       ; ADJUST H
707E CD9F3D CALL HILO    ; CHECK FOR FINISH
7081 D28D70 JNC DUMPA   ; JUMP NOT YET
7084 CD9970 CALL PBOTM   ; GET TO BOTTOM OF PAGE
7087 CDC83C CALL CRLF    ; SKIP A LINE
708A C33E3C JMP LER      ; RETURN TO MONITOR
    ; CHECK FOR BOTTOM OF PAGE
708D 3E13 DUMPA: MVI A,19 ; NEED 19 MORE LINES
708F 90 SUB B            ; TO CONTINUE ON THIS PAGE
7090 DA3270 JC DUMP5     ; JUMP MORE THAN 19
7093 CD9970 CALL PBOTM   ; PUT A BOTTOM ON THE PAGE
7096 C31870 JMP DUMP2    ; GET NEXT PAGE STARTED
    ;
7099 CDA270 PBOTM: CALL LSKIP   ; PAGE BOTTOM ROUTINE
709C 0E2D MVI C, '-'        ; SKIPS TO LAST LINE
709E CD503C CALL CO        ; AND PRINTS A DASH
70A1 C9 RET
    ;
70A2 CDC83C LSKIP: CALL CRLF   ; SKIPS FORWARD
3A5 05 DCR B              ; NUMBER OF LINES
70A6 2C2A270 JNZ LSKIP     ; IN REG B
70A9 C9 RET
    ;
70AA 41444452 HEAD: DB 'ADDR 1 2 3 4 5 6 '
70AE 20202020
70B2 20312020
70B6 32202033
70BA 20203420
70BE 20352020
70C2 362020
70C5 37202038 DB '7 8 9 A B C D E F '
70C9 20203920
70CD 20412020
70D1 42202043
70D5 20204420
70D9 20452020
70DD 46202020
70E1 302E2E2E DB '0...5....A....F'
70E5 2E352E2E
70E9 2E2E412E
70ED 2E2E2E46
70F1 0D DB 0DH         ; CARRIAGE RETURN
70F2 0A DB 0AH         ; LINE FEED
    ;
~900 END
ASCII Display

This routine expects two or three hex parameters. The first and second (16 bits) specify the bounds of a memory area to be displayed on the console device. The contents of these memory locations are interpreted as ASCII characters (MSB ignored).

Console device

9 monitor routines

Two or three hex numbers from the console device

Display on the console device

Registers Modified:
A,B,C,D,E,H,L

RAM Required:
3 bytes

ROM Required:
421 bytes (or RAM)

Maximum Subroutine Nesting Level:

Assembler/Compiler Used:
B6700/8080 Cross Assembler V 2.3

Programmer:
Gerhard Moertel, S.A. Nilsson

Company:
Universitaet Karlsruhe
Institut fuer Informatik IV

Address:
Zirkel 2
7500 Karlsruhe 1, GERMANY
RAM CHECK

Writes alternating 1's and 0's into all RAM locations then reads them and writes the two character result on a model 32 Teletype. Modifiable to ASR33 or LA-36, Decwriter. Can also check ROM read.

8080A, 4k ROM, any amount of RAM, 8251 USART, selectable baud rate clock.

The above program.

Starting address of RAM (resident in ROM).

Hex address and two 6's followed by two L's if address will read and write both 1's and 0's.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>A, B, C, D, E, H, L</td>
<td>Scheer</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RAM Required:</th>
<th>Company:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Any amount</td>
<td>Analog Precision, INC.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ROM Required:</th>
<th>Address:</th>
</tr>
</thead>
<tbody>
<tr>
<td>4k</td>
<td>1620 N. Park</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Maximum Subroutine Nesting Level:</th>
<th>City:</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0-</td>
<td>Tucson</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Assembler/Compiler Used:</th>
<th>State:</th>
</tr>
</thead>
<tbody>
<tr>
<td>MDS V1.0</td>
<td>Arizona 85719</td>
</tr>
</tbody>
</table>
MEMORY TEST PROGRAM/KYBD ENTRY OF START AND END VALUES

Provides extensive bit pattern tests to RAM memory located in any memory space higher than 300H.

Intellic equipment with kybd and printer

MDS-800 Monitor

Start address as 4 hex digits and end address as 4 hex digits from console.

Printout of all error locations along with expected and found values.

Registers Modified:
ALL

Programmer:
Floyd L. Nordin

Company:
Nordin Enterprises

Address:
P.O. Box 1277

City:
Cupertino

State:
CA 95014

RAM Required:
About 3/4 K bytes

ROM Required:
Monitor Subroutines

Maximum Subroutine Nesting Level:
12

Assembler/Compiler Used:
8080 MDS Macro Assembler V. 1.0
Basic CPU state vector maintenance in a multiprogrammed environment.

These three subroutines can be used by an operating system to maintain CPU state vectors when transferring control from one program to another. The subroutines use RAM locations LPnx to store the x state vectors of program n. Round robin program sequencing is used.

Required Hardware
None

Required Software
None

Input Parameters
None

Output Results
Multiprogrammed state vector maintenance.

Registers Modified: All
Assembler/Compiler Used: MicroPac # 80/A Micro Assembler
RAM Required: 30 bytes
Programmer: Stanley J. Kaczynski
ROM Required: 83x bytes
Company: Varian/Extrion
Maximum Subroutine Nesting Level: Unlimited
Address: P.O. Box 1226/Blackburn Ind. Park
          Gloucester, MA 01930
; REF. NO. AB46
; PROGRAM TITLE BASIC CPU STATE VECTOR MAINTENANCE

0000 E3  LC12:  XTHL
0001 228400  SHLD  LP1P
0004 E1  POP  H
0005 228600  SHLD  LP1H
0008 EB  XCHG
0009 228800  SHLD  LP1D
000C C5  PUSH  B
000D E1  POP  H
000E 228A00  SHLD  LP1B
0011 F5  PUSH  PSW
0012 E1  POP  H
0013 228C00  SHLD  LP1A
0016 2A9600  LHLD  LP2A
0019 E5  PUSH  H
001A F1  POP  PSW
  31B 2A9400  LHLD  LP2B
001E E5  PUSH  H
001F C1  POP  B
0020 2A9200  LHLD  LP2D
0023 EB  XCHG
0024 2A8E00  LHLD  LP2P
0027 E5  PUSH  H
0028 2A9000  LHLD  LP2H
002B C9  RET
002C E3  LC23:  XTHL
002D 228E00  SHLD  LP2P
0030 E1  POP  H
0031 229000  SHLD  LP2H
0034 EB  XCHG
0035 229200  SHLD  LP2D
0038 C5  PUSH  B
0039 E1  POP  H
003A 229400  SHLD  LP2B
003D F5  PUSH  PSW
003E E1  POP  H
003F 229600  SHLD  LP2A
0042 2A9000  LHLD  LP3A
0045 E5  PUSH  H
0046 F1  POP  PSW
0047 2A9E00  LHLD  LP3B
  34A E5  PUSH  H
  04B C1  POP  B
004C 2A9C00  LHLD  LP3D
004F EB  XCHG
L  0050  2A9800  LHLD  LP3P
0053  E5      PUSH  H
0054  2A9A00  LHLD  LP3H
0057  C9      RET
0058  E3      XTHL
0059  229800  SHLD  LP3P
005C  E1      POP   H
005D  229A00  SHLD  LP3H
0060  EB      XCHG
0061  229C00  SHLD  LP3D
0064  C5      PUSH  B
0065  E1      POP   H
0066  229E00  SHLD  LP3B
0069  F5      PUSH  PSW
006A  E1      POP   H
006B  22A000  SHLD  LP3A
006E  2A8C00  LHLD  LP1A
0071  E5      PUSH  H
0072  F1      POP   PSW
0073  2A8A00  LHLD  LP1B
0076  E5      PUSH  H
0077  C1      POP   B
0078  2A8800  LHLD  LP1D
007B  EB      XCHG
007C  2A8400  LHLD  LP1P
007F  E5      PUSH  H
0080  2A8600  LHLD  LP1H
0083  C9      RET
0084  001E  LP1P:  DW    170000
0086  021E  LP1H:  DW    170020
0088  041E  LP1D:  DW    170040
008A  061E  LP1B:  DW    170060
008C  081E  LP1A:  DW    170100
008E  0A1E  LP2P:  DW    170120
0090  0C1E  LP2H:  DW    170140
0092  0E1E  LP2D:  DW    170160
0094  101E  LP2B:  DW    170200
0096  121E  LP2A:  DW    170220
0098  141E  LP3P:  DW    170240
009A  161E  LP3H:  DW    170260
009C  181E  LP3D:  DW    170300
009E  1A1E  LP3B:  DW    170320
00A0  1C1E  LP3A:  DW    170340
0000  END
MP8208 A/D Converter Routine

Program functions as an interface between user's PL/M program and the Burr-Brown MP8208 A/D converter on an Intellec 8/MOD 80.


PL/M 80 cross-compiler

A/D Channel number

Address-type variable containing 12 bit value from selected A/D channel.

Registers Modified:
A,B,H,L

Assembler/Compiler Used:
PL/M 80

RAM Required:
1 byte of RAM used for storage, 2 bytes for stack

Programmer:
Jeff Stewart

ROM Required:
Program occupies 17 bytes of ROM or RAM.

Company:
Parke, Davis

Maximum Subroutine Nesting Level:
1

Address:
2800 Plymouth Road
Ann Arbor, Mich. 48106
00001 1 /*REF. NO. AC14 */
00003 1 /*PROGRAM TITLE MP8208 A/D CONVERTER ROUTINE */
00005 1 DECLARE ORG LITERALLY '400H';
00007 1 DECLARE AIODASMM LITERALLY '406H';
00009 1 ORG:
00011 1 DECLARE AIODASMM DATA (26H,0FFH,79H,87H,6FH,7EH,2CH,46H,0C9H);
00013 1 DECLARE TEMP BYTE, VALUE ADDRESS;
00015 1 ATODREAD: PROCEDURE(NUMBER) ADDRESS;
00017 1 GO TO ATODREAD;
00019 1 END ATODREAD;
00021 1 START:
00023 1 TEMP=INPUT(0);
00025 1 VALUE=ATODREAD(TEMP AND 0FH);
00027 1 IF ((TEMP AND 80H)=0) THEN
00029 1 DO;
00031 1 OUTPUT(0FFH)=LOW(VALUE);
00033 1 END;
00035 1 ELSE
00037 1 DO;
00039 1 OUTPUT(0FFH)=HIGH(VALUE);
00041 1 END;
00043 1 GO TO START;
00045 1 EDF;

NO PROGRAM ERRORS
"PAGE"

To provide a page break for a Tektronic #4010 graphics terminal.

The required hardware is an Intel MDS 800 and a Tektronic #4010 CRT

The required software is MDS Monitor Ver 1.1. The following changes are needed: Insert a CALL to EE00H in FD59 & FD5B in the Monitor. These instructions are to be replaced by "CD,00,EE,00" IN ORDER

When the end of a page is reached depressing the letter "E" on the keyboard will erase the page and allow output to continue. Substituting 00 at F000 will disable this routine.

Not Applicable

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Assembler/Compiler Used:</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM Required:</td>
<td>ISIS 8080 Macro Assy. V1.0</td>
</tr>
<tr>
<td>6 locations</td>
<td>Programmer:</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>Steve Weisbrod</td>
</tr>
<tr>
<td>179 locations</td>
<td>Company:</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td>Medtronic, Inc.</td>
</tr>
<tr>
<td>2</td>
<td>Address:</td>
</tr>
<tr>
<td></td>
<td>6120 Earle Brown Dr.</td>
</tr>
<tr>
<td></td>
<td>Mpls. MN 55430</td>
</tr>
</tbody>
</table>
;REF. NO. AC15
;PROGRAM TITLE PAGE BREAK FOR TEKTRONIX 4010 I/O GRAPHICS TERMINAL
;
;
"PAGE"
;
THE FUNCTION OF THIS PROGRAM IS TO
;PROVIDE A PAGE BREAK FOR A TEKTRONIC #4010
;GRAPHICS TERMINAL.
;
THE REQUIRED HARDWARE IS A INTEL MDS 800
;AND A TEKTRONIC #4010 CRT
;
THE REQUIRED SOFTWARE IS MDS MONITOR
;VER 1.1. THE FOLLOWING CHANGES ARE
;NEEDED: INSERT A CALL TO EE00H IN
;PLACE OF THE TWO INSTRUCTIONS AT
;FD59 & FD5B IN THE MONITOR
;THESE INSTRUCTIONS ARE TO BE REPLACED
;BY "CD,00,EE,00" IN ORDER
;
WHEN THE END OF A PAGE IS REACHED DEPRESSING
;THE LETTER "E" ON THE KEYBOARD WILL
;ERASE THE PAGE AND ALLOW OUTPUT TO
;CONTINUE. SUBSTITUTING 00 AT
;F000 WILL DISABLE THIS ROUTINE
;
******************************************************************************
EE00  ORG 0EE00H
F809  CO EQU 0F809H
F803  CI EQU 0F803H
EE00 F5 PAGE: PUSH PSW
EE01 D5 PUSH D
EE02 E5 PUSH H
EE03 3A02F0 LDA 0F002H ;CHECK TO SEE IF THIS IS THE
EE06 FE53 CPI 53H ;FIRST TIME THROUGH.
EE06 C21EEE JNZ SETUP
EE0B 3A03F0 LDA 0F003H
EE0E FE50 CPI 50H
EE10 C21EEE JNZ SETUP
EE13 3A04F0 LDA 0F004H
EE16 FE7C CPI 7CH
EE18 C21EEE JNZ SETUP
EE1B CA35EE JZ TEST1
EE1E 2102F0 SETUP: LXI H, 0F002H; SETUP FIRST TIME
EE21 3653 MVI M, 53H ;THROUGH.
EE23 2103F0 LXI H, 0F003H
EE26 3650 MVI M, 50H
EE28 2104F0 LXI H, 0F004H
EE2B 367C   MVI M, 7CH  
EE2D 2100F0  LXI H, 0F000H  
EE30 3601   MVI M, 01H  
EE32 23    INX H  
EE33 3600   MVI M, 00H  
EE35 2100F0  TEST1: LXI H, 0F000H;TEST FOR GRAPHICS OR DISHBLE  
EE38 7E    MOV A, M  
EE39 FE01  CPI 01H  
EE3B CA42EE JZ TEST2  
EE3E C5   PUSH B  
EE3F C37DEE JMP DONE  
EE42 79  TEST2: MOV A, C;TEST FOR LF  
EE43 E67F  ANI 7FH  
EE45 FE0A  CPI 0AH  
EE47 C5   PUSH B  
EE48 C253EE JNZ INPUT  
EE4B 2101F0  LXI H, 0F001H  
EE4E 3601   MVI M, 01H  
EE50 C37DEE JMP DONE  
EE52 2101F0  INPUT: LXI H, 0F001H  
EE56 7E    MOV A, M  
EE57 FE00  CPI 00H  
EE59 CA7DEE JZ DONE  
EE5C 3600   MVI M, 00H  
EE5E 1E06  MVI E, 06H  
EE60 1D    LOOP4: DCR E  
EE61 CA6DEE JZ BREAK  
EE64 06FF  MVI B, 0FFH  
EE66 05    LOOP5: DCR B  
EE67 C266EE JNZ LOOP5  
EE6A C360EE JMP LOOP4  
EE6D DBF7 BREAK: IN 0F7H ;ANYTHING ON INPUT PORT?  
EE6F E602  ANI 02H  
EE71 CA7DEE JZ DONE  
EE74 DBF6  IN 0F6H ;IS IT A BREAK?  
EE76 E67F  ANI 7FH  
EE78 FE60  CPI 60H  
EE7A CC86EE CZ WAIT  
EE7D C1    DONE: POP B  
EE7F E1    POP H  
EE7F D1    POP D  
EE80 F1    POP PSW  
EE81 DBF7  IN 0F7H ;INSTR'N FM MON  
EE83 E601  ANI 01H  
EE85 C9    RET  
EE86 CD03F8  WAIT: CALL CI ;"E" DEPRESSED?  
EE89 E67F  ANI 7FH  
EE8B FE45  CPI ',E'  
EE8D C286EE JNZ WAIT  
EE90 C5    ERASE: PUSH B ;SAVE CHAR
EE91 0E1B MVI C,1BH ; ERASE CRT
EE93 CD09F8 CALL CO
EE96 0E0C MVI C,0CH ; RESET CRT
EE98 CD09F8 CALL CO
EE9B 1602 DELAY: MVI D,02H ; TIME DELAY FOR
; CRT ERASE RECOVERY
EE9D 7A LOOP1: MOV A,D
EE9E FE00 CPI 00H
EEA0 C9B3EE JZ RESTR
EEA3 15 DCR D
EEA4 1EFF MVI E,0FFH
EEA6 1D LOOP2: DCR E
EEA7 C93DEE JZ LOOP1
EEAA 06FF MVI B,0FFH
EEAC 05 LOOP3: DCR B
EEAD C2ACEE JNZ LOOP3
EEB0 C3A6EE JMP LOOP2
EEB3 C1 RESTR: POP B ; RESTORE
; CHAR FOR NEW PAGE
EEB4 C9 RET
00000 END
CRTBZ

Transmits an ENQ (EC) from the computer to CRT terminal and waits til an ACK (FC) is received. Permits handshaking between CRT & computer for timing synchronization.

MDS (Intellec), 2640 CRT terminal (Hewlett Packard) RS232C cable

I/O subroutines in Intellec MDS Monitor CI and CO

None

Permits smooth interfacing of 2640 CRT with MDS otherwise MDS dumps information at a faster rate (9600 baud) than the CRT can take (2400 baud maximum), consequently CRT will skip or delete some information.

NOTE: CRTBZ is ordered as one program with GET. Ref. No. AC16 refers to both routines.

Registers Modified: A, C and FLAGS

Assembler/Compiler Used: Intellec MDS Assembler Version 1.0

RAM Required: 14 bytes

Programmer: Ahmed Muneeruddin

ROM Required: None

Company: Management Business Machines

Maximum Subroutine Nesting Level: 

Address: 5867 Broadway Denver, Colorado 80216
GET

Read characters from CRT (2640 HP) in block mode strapped for line & format mode on; store the characters at a specified place. The number of unprotected fields to be read is selected by the user (i.e. the length of the block).

Intellec MDS system, 2640 CRT terminal (HP)
RS232C cable

MDS I/O routines: CI, CO
CRT Control Routines: CRTBZ, ENABL, TRIGR, RESET, CRLF, PRINT

B must contain number of unprotected fields to be read.
H&L should contain starting address in RAM where information can be read.

Information read from CRT memory is available in specified area of RAM with end of block identified with a record separator rs. Each unprotected field is separated with a carriage return. At the end of input process, a message appears for the CRT operator to turn off the block mode key on the CRT.

NOTE: GET is ordered as one program with CRTBZ. Ref. No. AC16 refers to both routines.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>A, C, B, H&amp;L, &amp; FLAGS</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM Required:</td>
<td>4C bytes + storage for other subroutines</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>NONE</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td></td>
</tr>
</tbody>
</table>

Assembler/Compiler Used: MDS 8
Macro Assembler Version 1.0

Programmer:
Ahmed Muneeruddin

Company:
Management Business Machines

Address:
5867 Broadway
Denver, Colorado 80216
Disassembler

This program transforms machine codes in memory to a listing of:
1. Addresses
2. Machine codes (1,2 or 3 bytes) depending on the type of instruction.
4. Addresses when applicable (JMP and CALL)

TTY on port 0 and 1

Machine line INTELLEC 8/MOD 8
MONITOR VER 3.0 (on PROMS)

Intellec 8 MOD/8 Text Editor VER4.0
Macro Assembler Ver 2.0
Monitor Ver 3.0

Start of this program:
G037ED
The TTY types DISASSEMBLER and expects two HEX parameters of the memory dump to be disassembled, typed on the TTY.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>A,B,C,D,E,H,L</th>
<th>Assembler/Compiler Used: Intellec 8 Macro Assembler 2.0 Monitor Ver 3-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM Required:</td>
<td>255 Bytes</td>
<td>Programmer: Manuel Puigbo</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>MONITOR VER3-0 1024 BYT</td>
<td>Company: Elecma</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td>2</td>
<td>Address: Ricardo Calvo, 13 Barcelona (6) Spain</td>
</tr>
</tbody>
</table>
ERLIST

Will search through a file assumed to be all 8080 macro assembler output list file and will copy all lines containing errors to the console.

MDS-800 and MDS-DOS with 32K RAM

ISIS (NOTE: will not run under V1.0)

File Name
(ERLIST FILENAME.EXT)

Lines which contain an error code to the console

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Assembler/Compiler Used:</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL</td>
<td>ASM80 - V1.1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RAM Required:</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>32K MDS</td>
<td>Barry Yarkoni</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ROM Required:</th>
<th>Company:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Maximum Subroutine Nesting Level:</th>
<th>Address:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>57 West Timonium Road</td>
</tr>
<tr>
<td></td>
<td>Timonium, Md. 21093</td>
</tr>
</tbody>
</table>
EXAMIN
Copies "N" lines at a time from a specified file to the MDS console. Hitting the space bar will type the next 10 lines. An "E" will terminate the program at any time.

MDS-800, MDS-DOS, console, and 32K RAM

32K ISIS, any version
MDS monitor, V1.2 or greater

FILE NAME
EXAMIN FILENAME.EXT (CR): program will request console input

Directly to console, 10 lines at a time

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Assembler/Compiler Used:</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>ASM80 - V1.1</td>
</tr>
<tr>
<td>RAM Required:</td>
<td>Programmer:</td>
</tr>
<tr>
<td>32K MDS</td>
<td>Barry Yarkoni</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>Company:</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td>Address: 57 West Timonium Road Timonium, Md. 21093</td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
Ice-80 Disassembler

A subroutine to be used as a "MDS call" in Ice-80 command language. Translates control block information to assembly statements that are output to selected list device.

Intellec MDS and Ice-80.

Ice-80 software driver & RAM based version 2.0 or 3.0 or disk based version 1.0.

Information stored in control block by Ice-80 when returning form emulation mode.

Assembly statements are output to selected list device as the instructions are executed by the Ice-80 CPU.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Assembler/Compiler Used:</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>MDS Assembler</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RAM Required:</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1121 bytes</td>
<td>Ove Andersson</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ROM Required:</th>
<th>Company:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Intel Scandinavia A/S</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Maximum Subroutine Nesting Level:</th>
<th>Address:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Lyngbyvej 32F,II</td>
</tr>
<tr>
<td></td>
<td>2100 Copenhagen Ø, Denmark</td>
</tr>
</tbody>
</table>
DELETE COMMENTS

Reads 8080 source tape from assigned reader (uses monitor RI routine) and deletes all comments. A reduced source tape (instructions only) is punched on the assigned punch device (uses monitor P O routine; can be co-resident with 8080 assembler, etc.)

MDS 800 with TTY assigned as punch and reader, is the minimum configuration. High speed punch and reader may be used if available.

MDS 800 System Monitor

Program calls RI ($F8$86M) 
Requires character read to be in REG C
Requires carry Bit set if reader has timed out.
(This is as per the monitor routine.)

Program calls P O ($F8$8CH)
Character to be punched is in REG C
At end of tape (reader times out) program returns control to the monitor.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Assembler/Compiler Used:</th>
</tr>
</thead>
<tbody>
<tr>
<td>A,B,C,FLAGS</td>
<td>8080 MDS Assembler V.1.0</td>
</tr>
<tr>
<td>RAM Required:</td>
<td>Programmer:</td>
</tr>
<tr>
<td>160 Bytes</td>
<td>I.E. Powers</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>Company:</td>
</tr>
<tr>
<td></td>
<td>Plessey Radar</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td>Address:</td>
</tr>
</tbody>
</table>

Cheapside, Liverpool
L2 2EA (England)
I.S 8080 MACRO ASSEMBLER, V1.0

; REF. NO. AB52
; PROGRAM TITLE DELETE COMMENTS
;
;
; DELETE COMMENTS PROGRAM I E POWERS

; PROGRAM READS 8080 SOURCE TAPE FROM ASSIGNED READER
; ALL COMMENTS PREFIXED WITH \ ARE DELETED AND ALL
; RETAB AND NULL CHARACTERS ARE IGNORED
; THE REMAINING CHARACTERS ARE OUTPUT ON THE ASSIGNED
; PUNCH DEVICE
; LINES BEGINNING WITH \ ARE DELETED COMPLETELY
; COMMENTS FOLLOWING INSTRUCTIONS ARE DELETED WITHOUT
; AFFECTING THE FOLLOWING CR LF SEQUENCE
;
; OPERATING PROCEDURE:-
; LOAD ME BY TYPING R0 ON THE CONSOLE DEVICE
; PLACE THE TAPE TO BE PROCESSED IN THE READER AND
; TYPE G3000 ON THE CONSOLE
; AFTER PUNCHING THE REDUCED TAPE I RE-ENTER THE MONITOR
;
; TEST PROCEDURE:-
; ASSEMBLE TEST TAPE TO OBTAIN AN OBJECT TAPE
; USE MY OBJECT TAPE TO PROCESS TEST SOURCE TAPE
; ASSEMBLE THE REDUCED SOURCE TAPE
; BOTH OBJECT TAPES SHOULD CORRESPOND EXACTLY

3000 ORG 3000H
3000 AF BEGIN: XRA A ; CLEAR ACC
3001 329E30 STA IGLIN ; CLEAR IGNORE LINE FLAG
3004 329F30 STA IGTXT ; CLEAR IGNORE TEXT FLAG
3007 32A030 STA NEWLI ; CLEAR NEWLINE FLAG
300A CD4830 CALL PUL ; PUNCH LEADER
300D CD06F8 READ: CALL 0F806H ; READ CHAR FROM RI
3010 DA9A30 JC EOJ ; IF READER TIMES OUT - GOTO END OF JOB
3013 0601 MVI B,1 ; SET OUTPUT CHARACTER COUNT TO 1
3015 4F MOV C,A ; MOVE INPUT CHAR TO A
3016 E67F ANI 7FH ; IGNORE PARITY BIT
3018 FE7F CPI 7FH ; CHECK FOR RETAB
301A CA0D30 JZ READ ; IGNORE RETAB - GET NEXT CHAR
301D FE00 CPI 0 ; CHECK FOR NULL
301F CA0D30 JZ READ ; IGNORE NULL
3022 FE0D CPI 0DH ; CHECK FOR CR
3024 CA5730 JZ CRET ; CR DETECTED
3027 FE0A CPI 0AH ; CHECK FOR LF
3029 CA6630 JZ LINF ; LF DETECTED
302C FE38 CPI 3BH ; CHECK FOR ;
302E CA7A30  JZ SEMCO ; detected
 ; assume char other than CR LF or ;
3031 AF XRA A
3032 32A030 STA NEWLII ; clear newline flag
3035 3A9E30 LDA IGLIN ; check ignore whole line flag
3038 FE00 CPI 0
303A C20D30 JNZ READ ; flag set - ignore char
303D 3A9F30 LDA IGTXT ; check ignore text flag
3040 FE00 CPI 0
3042 C20D30 JNZ READ ; flag set - ignore char
 ; no flags set - print char on punch
3045 CD4F30 PRI: CALL PO
3048 C30D30 JMP READ ; get next char
 ; punch leader routine
304B 0664 PUL: MVI B,100 ; set length of leader count
304D 0E00 MVI C,0 ; set a null char for output
 ; punch output routine
304F CD0CF8 PO: CALL OF80CH ; call punch output in monitor
3052 05 DCR B ; decrement char count
3053 C24F30 JNZ PO ; jump back if more to punch
3056 C9 RET
3057 3A9E30 CRET: LDA IGLIN ; check ignore whole line flag
305A FE00 CPI 0
305C C20D30 JNZ READ ; flag set - ignore CR
305F AF XRA A ; flag not set - zero A
3060 329F30 STA IGTXT ; clear the ignore text flag
3063 C34530 JMP PRI ; output the char
3066 3E01 LINF: MVI A,1 ; we are starting a new line
3068 32A030 STA NEWLII ; set newline flag
306B 3A9E30 LDA IGLIN ; check ignore whole line flag
306E FE00 CPI 0
3070 CA4530 JZ PRI ; flag not set - output LF
3073 AF XRA A ; flag set - ignore LF
3074 329E30 STA IGLIN ; reset the flag
3077 C30D30 JMP READ ; get next char
3079 3A9E30 SEMCO: LDA IGLIN ; check ignore whole line flag
307D FE00 CPI 0
307F C20D30 JNZ READ ; ignore comments within comments
3082 3A9030 LDA NEWLII ; check newline flag
3085 FE00 CPI 0
3087 CA9230 JZ MID ; not set - mid line comment
308A 3E01 MVI A,1 ; flag set whole line is comment
308C 329E30 STA IGLIN ; set ignore whole line flag
308F C30D30 JMP READ ; ignore ; and get next char
3092 3E01 MID: MVI A,1 ; handle mid line comment
3094 329F30 STA IGTXT ; set ignore text on this line flag
3097 C30D30 JMP READ ; get next char
 ; 09A CD4830 EOJ: CALL PUL ; end of job - punch trailer
309C 07 RST 0 ; back to monitor
309E 00 IGLIN: DB 0 ; ignore whole line flag
309F 00 IGTXT: DB 0 ; IGNORE TEXT ON THIS LINE FLAG
30A0 00 NEWLI: DB 0 ; NEW LINE FLAG
0000 END
TYPE

Types a source file with the correct number of spaces inserted for each tab. Display can be "frozen" by typing any keyboard character, and unfrozen by a similar action. The program is terminated when a C is typed.

Required Hardware
MDS with Diskette

Required Software
ISIS-I

Input Parameters
File Name

Output Results
File Listing

Registers Modified: Programmed: Norman H. Azadian
RAM Required: Company: System Development Corp.
ROM Required: Address: 2500 Colorado Avenue
Maximum Subroutine Nesting Level: City: Santa Monica
Assembler/Compiler Used: Intellec 8080 State: California 90406
Macro Assembler V1.0
SDK-80 keyboard monitor

basic operating system for the SDK-80 microcomputer kit. allows loading program and execution of user programs as well as breakpoint and register examination and modification

1. Intel SDK-80 microcomputer with I/O port installed at locations 0F4h to 0F7h.

2. Keyboard and keyboard control circuit per schematic
none additional

keystrokes

implementation of functions requested

| Registers Modified: | Assembler/Compiler Used: |
| all | MDS Isis-ASM80 V1.0 |
| | |
| RAM Required: | Programmer: |
| 19 decimal locations | J.F. Jankura |
| | |
| ROM Required: | Company: |
| 24A hexadecimal locations | General Electric |
| | |
| Maximum Subroutine Nesting Level: | Address: |
| 5 levels | Nela Park, Cleveland, Oh. |
Disc Dump Routine for ICOM FDOS-II/MOD 80 floppy disc operating system.

This routine provides a formatted dump of any portion of the floppy disc, in decoded ASCII, hexadecimal, or both. The user specifies the number of sectors to be dumped and the desired format.

Intellec 8/MOD 80, ICOM floppy disc drive.

Intellec monitor, ICOM FODS-ll/mod 80 resident module.

Unit no., first sector and track to be dumped, no. of sectors to be dumped, and format (alpha, hex, or both).

Formatted dump of disc. DD-marked sectors can not be dumped, so a message is printed to this effect. At the end of the dump, the user may rerun the program, or go to either the Intellec or FDOS monitor.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Assembler/Compiler Used:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PL/M vers 3.2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RAM Required:</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>700 Bytes hex</td>
<td>L.P.M. Payzant</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ROM Required:</th>
<th>Company:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Nova Scotia Technical College</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Maximum Subroutine Nesting Level:</th>
<th>Address:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P. O. Box 1000, Halifax, Nova Scotia, Canada.</td>
</tr>
</tbody>
</table>
100H% /* DISC DUMP PROGRAM */
/* P. PAYZANT, N. S. T. C., AUGUST, 1976 */
DECLARE (UNIT, TRACK, SECTOR, CODE, PRINT, J, K, CHAR) BYTE;
DECLARE (ACTTRACK, ACTSECTION, EXPSECTION, EXPTRACK) BYTE;
DECLARE (NSECT, I) ADDRESS;
DECLARE BUF (128) BYTE;
DECLARE TRUE LITERALLY 'OFFH', FALSE LITERALLY '0';

CIC% PROCEDURE BYTE; /* READ A CHARACTER */
GO TO 3803H;
END CIC;

COC% PROCEDURE (CHAR); /* PRINT A CHARACTER */
DECLARE CHAR BYTE;
GO TO 3803H;
END COC;

SPACE% PROCEDURE (NSPACES); /* PRINT N SPACES */
DECLARE (N, NSPACES) BYTE;
DO N=1 TO NSPACES;
CALL COC("");
END;
END SPACE;

RDISC% PROCEDURE BYTE; /* READ A BYTE FROM THE DISC */
GO TO 3105H;
END RDISC;

PRINTN% PROCEDURE (NUMBER); /* PRINT A BYTE IN HEX */
DECLARE (NUMBER, CHAR) BYTE;
CHAR=SHR(NUMBER, 4);
IF CHAR>=10 THEN CHAR=CHAR+'A'-10;
ELSE CHAR=CHAR+'0';
CALL COC(CHAR);
CHAR=NUMBER AND 0FH;
IF CHAR>=10 THEN CHAR=CHAR+'A'-10;
ELSE CHAR=CHAR+'0';
CALL COC(CHAR);
END PRINTN;

PRINTA% PROCEDURE (STRINGADDR, CRLFFLAG); /* PRINT A STRING */
DECLARE STRINGADDR ADDRESS, (J, CRLFFLAG) BYTE;
DECLARE (STRING BASED STRINGADDR) (70) BYTE;
IF CRLFFLAG=1 THEN DO;
CALL COC(0DH); /* CR */
CALL COC(0AH); /* LF */
END;
J=0;
DO WHILE STRING(J)<> '$';
call COC(STRING(J));
J=J+1;
END;
END PRINTA;

READHEX% PROCEDURE ADDRESS; /* READ A HEX NUMBER TYPED AT THE CONSOLE */
DECLARE VAL ADDRESS, CHAR BYTE;
VAL=0;
DO WHILE TRUE;

4-304
CHAR=CIC;
CALL COC(CHAR);
IF (CHAR AND 7FH)=0DH THEN DO;
   CALL COC(0AH);
   RETURN VAL;
END;
CHAR=(CHAR AND 7FH)-'0';
IF CHAR=0 AND CHAR <=9 THEN DO;
   VAL=(16*VAL)+CHAR;
END;
ELSE IF CHAR>=11H AND CHAR <=16H THEN DO;
   VAL=(16*VAL)+CHAR-7;
END;
ELSE CALL COC(07H); /* RING BELL FOR NON-HEX CHARACTER */
END;
END READHEX;

DECLARE MSG1 DATA ('DISC DUMP PROGRAM$'), MSG2 DATA ('UNIT NO.? (0-3) $'),
   MSG3 DATA ('STARTING TRACK? (00-4C) $'), MSG4 DATA ('STARTING SECTOR? (1-1A) $'),
   MSG5 DATA ('NO OF SECTORS? (1-7D2) $'), MSG6 DATA ('ALPHA(1), HEX(2), OR BOTH(3)? $');
DECLARE MSG7 DATA ('TRACK $'), MSG8 DATA ('SECTOR $'), MSG9 DATA ('BYTE $'),
   MSG10 DATA ('$'), MSG11 DATA ('ENTER A (AGAIN), D (DOS), OR M (MONITOR) $'),
   MSG12 DATA ('* THIS SECTOR DD MARKED* $');

/ * DIALOGUE WITH USER */

   CALL PRINTA (.MSG1,1);
   CALL PRINTA (.MSG2,1);
   IF (UNIT%=LOW(READHEX))<0 OR UNIT >3 THEN GO TO P1;
    P1%
   CALL PRINTA (.MSG3,1);
   IF (TRACK%=LOW(READHEX))<0 OR TRACK>4CH THEN GO TO P2;
    P2%
   CALL PRINTA (.MSG4,1);
   IF (SECTOR%=LOW(READHEX))<1 OR SECTOR>1AH THEN GO TO P3;
    P3%
   CALL PRINTA (.MSG5,1);
   IF (NSECT%=LOW(READHEX))<1 OR NSECT>7D2H THEN GO TO P4;
    P4%
   CALL PRINTA (.MSG6,1);
   IF (CODE%=LOW(READHEX))<1 OR CODE>3 THEN GO TO P5;
    P5%

/ * INITIALIZE FILE POINTER TABLE */

DECLARE TABLELOC ADDRESS, (TABLE BASED TABLELOC) (5) BYTE;
   TABLELOC=5H; /* ADDRESS OF INPUT FILE TABLE */
   NSECT=NSECT+1;
   TABLE(0)=LOW(NSECT);
   TABLE(1)=HIGH(NSECT);
   TABLE(2)=TRACK;
   TABLE(3)=(SECTOR-1) OR ROL(UNIT,6);
   TABLE(4)=0;

/ * MAIN DUMP LOOP */

   EXPTRACK=TRACK;
   EXPSECT=SECTOR;
   NSECT=NSECT-1;
   DO I=1 TO NSECT;

/ * READ 1 SECTOR (128 BYTES) INTO RAM. */

   DO J=0 TO 127;
      BUF(J)=RDISC;
      END;
/* GET SECTOR AND TRACK JUST READ FROM FILE POINTER TABLE, AND CHECK THAT IT MATCHES THE EXPECTED TRACK AND SECTOR. IF NOT, ISSUE DD MESSAGE, INCREMENT EXPECTED SECTOR, AND CHECK AGAIN UNTIL EXPECTED AND ACTUAL MATCH. */

CALL PRINTA(. MSG7, 1);  
CALL PRINTN(EXPSECT); 
CALL PRINTA(. MSG8, 0);  
CALL PRINTN(EXPSECT);   
ACTSECT=TABLE(3) AND 1FH;  
ACTTRACK=TABLE(2);      
DO WHILE ACTSECT<>EXPSECT OR ACTTRACK<>EXPTRACK;
  CALL PRINTA(. MSG12, 0); 
  EXPSECT=EXPSECT+1;       
  IF EXPSECT=1BH THEN DO;  
    EXPSECT=1;               
    EXPTRACK=EXPTRACK+1;    
    END;                    
  CALL PRINTA(. MSG7, 1);  
  CALL PRINTN(EXPTRACK);   
  CALL PRINTA(. MSG8, 0);  
  CALL PRINTN(EXPSECT);   
  END;                     
  EXPSECT=EXPSECT+1;       
  IF EXPSECT=1BH THEN DO;  
    EXPSECT=1;               
    EXPTRACK=EXPTRACK+1;    
    END;                    
  CALL PRINTA(. MSG9, 1);  

/* PRINT THE SECTOR IN LINES OF 10H BYTES */

DO J=0 TO 70H BY 10H;  
  CALL PRINTA(. MSG10, 1); 
  CALL SPACE(1);    
  CALL PRINTN(J);   
  CALL SPACE(1);   
  IF CODE AND $01H=1 THEN DO;  /* ALPHA DUMP */ 
    PRINT=FALSE;          
    DO K=0 TO 15;        /* SCAN FOR PRINTABLE CHARACTERS */ 
      CHAR=BUF(J+K); 
      IF CHAR>20H AND CHAR<7EH THEN PRINT=TRUE;  
      END;               
    IF PRINT THEN DO K=0 TO 15; 
      CALL SPACE(2);   
      CHAR=BUF(J+K) AND 7FH;  
      IF CHAR<20H OR CHAR>7EH THEN CHAR=' '; 
      CALL COC(CHAR);     
      END;               
    END;               
  IF CODE AND $02H=2 THEN DO;  /* HEX DUMP */ 
  IF (CODE AND $01H)=1 THEN DO;  
    CALL PRINTA(. MSG10, 1); 
    CALL SPACE(4);        
    END;                 
  DO K=0 TO 15;   
    CALL SPACE(1);   
    CALL PRINTN.BUF(J+K);  
    END;               
  END;
END;

P6%  CALL PRINTA('MSG11,1);
    CALL COC(CHAR%=CIC);
    CHAR=CHAR AND 7FH;
    IF CHAR='A' THEN GO TO P1;
    ELSE IF CHAR='M' THEN GO TO 3800H;
    ELSE IF CHAR='D' THEN GO TO 3000H;
    ELSE GO TO P6;
EUF
COPY COMPLETE.
LIST/PRINT/TYPE "LIST, SRC" on Diskette.

LIST a file on the line printer allowing space for tabs.

Intellec MDS and appropriate output device.

ISIS diskette Operating Systems.

Change output file name (symbol "LFILE") to
:TO: for TTY (PRINT PROG)
:CO: for CRT/TTY (LIST PROG)
:VO: for CRT (LIST PROG)
:LP: for Line Printer (LIST PROG)

Tab characters will be acknowledged.

Registers Modified:  
Assembler/Compiler Used:
ASM80

RAM Required:  
32K

ROM Required:

Maximum Subroutine Nesting Level:

Programmer:  
Brian L. Halla

Company:  
Intel Corp.

Address:  
3065 Bowers Ave.
Santa Clara, Ca. 95051
Interfacing the MDS and HP2644A

Acts as an interface between MDS and HP2644A mini data station. Lets user use tape for both reading and writing data.

MDS system, HP2644A mini data station optional floppy disc system.

For writing data of tape: C Reg should contain ASCII character.
For reading data from tape: A Reg contains the ASCII character.
For using the console as a list device: A Reg should contain ASCII character.

When reading data from tape, the program returns the ASCII character in the 'A' registers. When it detects end of file, it returns two null characters. When writing data to tape, the program writes out the character in the 'C' register to tape.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Assembler/Compiler Used:</th>
</tr>
</thead>
<tbody>
<tr>
<td>A, PSW</td>
<td>ASM80 Assembler</td>
</tr>
<tr>
<td>RAM Required:</td>
<td>Programmer:</td>
</tr>
<tr>
<td>240H Bytes</td>
<td>A. Aggarwal</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>Company:</td>
</tr>
<tr>
<td>0</td>
<td>BNR, Inc.</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td>Address:</td>
</tr>
<tr>
<td>2</td>
<td>3174 Porter Dr., Palo Alto, CA.</td>
</tr>
</tbody>
</table>
Video Driver

To drive a video system of 16 lines by 64 characters per line.

1K video ram, locations OFCOOH to OFFFH, as part of 8080 bus.
Each memory location represents 1 character. Each successive block
of 64 locations represents one line.

None

A register has 7 bit ASCII character.

FF (form feed) character sets all of 1K block to spaces (020H)
and sets the character pointer to the start of the memory.

LF (line feed) character adds 64 modulo 1024 to the character
pointer and then blanks line.

CR (carriage return) character sets memory locations from the
character pointer to the end of the current line to blanks and
sets the character pointer to the beginning of the current
line.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>None</th>
<th>Assembler/Compiler Used:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Intellec 8 Mod 80 Ver 3.0</td>
</tr>
<tr>
<td>RAM Required:</td>
<td></td>
<td>Programmer:</td>
</tr>
<tr>
<td>1K Video RAM at OFCOOH</td>
<td></td>
<td>S. Graf</td>
</tr>
<tr>
<td>2 bytes for character pointer</td>
<td></td>
<td>Company:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>B.C. Telephone Company</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>96 bytes</td>
<td>Address:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>768 Seymour STR. (4th WCT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Vancouver, B.C., Canada</td>
</tr>
<tr>
<td></td>
<td></td>
<td>V6B 3K9</td>
</tr>
</tbody>
</table>
; REF. NO. AC19
; PROGRAM TITLE VIDEO DRIVER

; RAM SPACE

0000 00FC  VIDCP:   DW  0FC00H ; 2 BYTES REQUIRED

; EQUATES

000C  FF  EOU  0CH ; FORM FEED
000A  LF  EOU  0AH ; LINE FEED
000D  CR  EOU  0DH ; CARRIAGE RETURN
0020  SPACE  EOU  020H ; SPACE

; PROGRAM SPACE

; ENTRY POINT

; WRITE:

0002  F5  PUSH  PSW ; SAVE
0003  E5  PUSH  H
0004  2A000A  LHLH  VIDCP ; GET CHARACTER POINTER
0007  FFAC  CPI  FF ; CHECK FOR FORM FEED
0009  CA1E00  JZ  VIDFF ; JUMP IF FORM FEED
000C  FE0D  CPI  CR ; CHECK FOR CARRIAGE RETURN
000E  CA2F00  JZ  VIDCR ; JUMP IF CARRIAGE RETURN
0011  FE0A  CPI  LF ; CHECK FOR LINE FEED
0013  CA3800  JZ  VIDLF ; JUMP IF LINE FEED
0016  77  MOV  M.A ; PUT CHARACTER IN VIDEO MEMORY
0017  23  INX  H ; BUMP POINTER

; COMMON EXIT POINT

; VIDRT:

0018  22000A  SHLD  VIDCP ; SAVE CHARACTER POINTER
001B  F1  POP  H ; RESTORE
001C  F1  POP  PSW ; RESTORE
001D  C9  RET ; EXIT
FORM FEED SERVICE

001E 2100FC  LXI  H,0FC00H; SET STARTOF VIDEO MEMORY
0021  220000  SHLD VIDCP; SET NEW START POSITION

VIDFC:

0024  3620  MVI  M,SPACE; PUT SPACE IN VIDEO MEMORY
0026  23    INX  H; BUMP POINTER
0027  AF    XRA  A; SET A TO 0
0028  BC    CMP  H; CHECK IF H IS 0
0029  C22400 JNZ  VIDFC; JUMP IF NOT
002C  E1  POP  H; DONE - RESTORE
002D  F1  POP  PSW; RESTORE
002E  C9  RET; EXIT

CARRIAGE RETURN SERVICE

VIDCR:

002F  E5    PUSH  H; SAVE PRESENT CARRACTER POSITION
0030  CD5800 CALL  VBLNK; BLANK FILL LINE
0033  E1    POP  H; RESTORE CHARACTER POSITION
0034  7D    MOV  A,L; GET LOW BYTE CHARACTER POINTER
0035  E6C0  ANI  0C0H; SET BACK TO START OF LINE
0037  6F    MOV  L,A; PUT BACK
0038  C31800 JMP  VIDRT; USE COMMON EXIT

LINE FEED SERVICE

VIDLF:

0038  7D    MOV  A,L; GET LOW BYTE CHARACTER POSITION
003C  C640  ADI  040H; INCREMENT LINE COUNT
003E  6F    MOV  L,A; PUT BACK
003F  7C    MOV  A,H; GET HIGH BYTE
0040  CE00  ACI  A; INCREMENT IF NECESSARY
0042  67    MOV  H,A; PUT BACK
0043  FA4C00 JM  VLFRL; BLANK LINE IF NO WRAP AROUND
0046  26FC  MVI  H,0FC8H; SET TO START IF OVER END
0048  7D    MOV  A,L; GET LOW ORDER LINE BITS
0049  E6C0  ANI  0C0H; CLEAR THEM
004B  6F    MOV  L,A; PUT BACK

VLFRL:

004C  E5    PUSH  H; SAVE CHARACTER POSITION
004D  7D    MOV  A,L; GET POSITION IN LINE
004E  E6C0  ANI  0C0H; SET TO START
0050  6F    MOV  L,A
0051  CD5800 CALL  VBLNK; BLANK LINE
0054  E1    POP  H; RESTORE CHARACTER POSITION
0055 C31A00   JMP VIDRT ;USE COMMON EXIT
          ;
          ; BLANK TO END OF LINE
          ;
0058 3620   MVI M.SPACE ;PUT IN SPACE
005A 2C   INR L ;RUMP COUNTER
0058 7D   MOV A.L ;GET POSITION IN LINE
005C F63F   ANI 03FH ;ONLY
005E C25A00   JNZ VBLNK ;JUMP IF NOT AT END OF LINE
0061 C9   RET ;DONE
          ;
0000   END
BASIC DIGITAL PANEL METER CALL

Reads BCD data from a 4 digit plus sign panel meter and sets a BASIC variable X equal to the data. Meter initiation is done in hardware.

8080 system with panel meter I/O interface.

8080 BASIC, floating point package, and octal debugging program.

BASIC call statement protocol:

CALL (1,X)

X Should Be Previously Defined.

The variable X equals the meter reading integer value.

I/O port numbers: 6 reads MS 8 bits, 7 reads LS 8 bits, 1 reads sign bit: D7.

Registers Modified: A, B, C, D, E, H, L

Assembler/Compiler Used: 8080 macro assem. ver. 2.2

Programmer: C. L. Pomernacki

Company: LLL

Address: Livermore, California 94550
; REF. NO. AC20
; PROGRAM TITLE BASIC DIGITAL PANEL METER CALL
;
;
; CALL ROUTINES

0000 30A0  FWAM:   DW 00A30H ; DEFINE FWAM POINTER

; ENTRIES TO SUBTABLE

0002 01  SUBS:   DB 1
0003 0600  DW BCDBIN
0005 FF  DB 3770 ; END OF TABLE

; SUB #1

0006 DB06  BCDBIN:  IN 6 ; MS8 TO A
0008 67  MOV H,A ; A TO H
0009 DB07  IN 7 ; LS8 TO A
000B 6F  MOV L,A ; A TO L
000C 1500  MOVI E,0 ; CLEAR E REG.
000E 7D  MOV A.L ; L TO A
000F CD2500  CALL MSKMS
0012 D5  PUSH D ; D,E TO STACK
0013 7D  MOV A.L ; L TO A AGAIN
0014 CD2900  CALL MSKLS
0017 D5  PUSH D ; D,E TO STACK
0018 7C  MOV A.H ; H TO A
0019 CD2500  CALL MSKMS
001C D5  PUSH D ; D,E TO STACK
001D 7C  MOV A.H ; H TO A AGAIN
001E CD2900  CALL MSKLS
0021 D5  PUSH D ; D,E TO STACK
0022 C3100  JMP BBIN
0025 E60F  MSKMS:  ANI 170 ; MASKOUT MS4
0027 57  MOV D,A ; A TO D
0028 C9  RET
0029 E60F  MSKLS:  ANI 360Q ; MASKOUT LS4
002B 0F  RRC ; SHIFT
002C 0F  RRC
002D 0F  RRC
002E 0F  RRC ; 4 RIGHT
002F 57  MOV D,A ; A TO D
0030 C9  RET
BBIN: MVI C, 4 ; # IS # OF BCD DIGITS
LXI H, 0 ; INITIALIZE H, L
XRA A ; AND A TO ZERO

; MULT EACH DIGIT BY 10

LOOP: PUSH H ; SAVE LS 16 BITS ON STK.
POP D ; RESTORE 16 BITS TO D, E
ADD D ; ADD 1
DAD D ; TO MAKE 5
ADC A ; MULT BY 2
DAD H ; TO MAKE 10
ADC A ; SAVE A IN B
MOV B, A ; GET NEXT DIGIT
MOV A, D ; INTO A REG.
ADD L ; ADD IN NEW DIGIT
MOV L, A
MOV A, H
ACI 0 ; ADD IN CARRY
MOV H, A
MOV A, B
ACI 0 ; ADD IN CARRY
DCR C ; BUMP DOWN COUNTER
JNZ LOOP ; DONE?

; NOTE: A, H, L CONTAIN FIXED PT. #
; DIGITS ARE RIGHT JUSTIFIED MS IN TOP OF STACK

LXI D, 7C10H ; GET REG. ADD.
XCHG ; D, E HAS # H, L HAS ADD.
MOV M, A ; MS TO FREG1
INX H
MOV M, D ; S TO FREG1+1
INX H
MOV M, E ; LS TO FREG1+2
INX H
IN 1 ; SIGN TO A
MOV M, A ; SIGN TO FREG1+3
DCX H
DCX H
DCX H
DCX H
DCX H
CALL @CA0FH ; FREG #
POP D ; GET VAR. ADD.
XCHG ; D, E HAS FREG1

0052 11307C
0055 EB
0056 77
0057 23
0058 72
0059 23
005A 73
005B 23
005C DB01
005E 77
005F 2B
0060 2B
0061 2B
0062 CD0FCA
0065 D1
0066 EB
0067 CD1B6    CALL 0B617H ; SET VAR. =FREG1
006A C9      RET    ; DONE
006C END
APL Graphic Display on a 5 x 7 Dot Matrix

The program decodes a standard APL keyboard and outputs the pressed key to a 5 x 7 dot matrix display.

Required Hardware

None

Required Software

Input port 80 is 7-bit code from an APL keyboard

Output Results

Output ports 0 through 4 drive the five columns on a MAN 2A dot matrix display. I/O latch stores the output data, which is input into the seven rows of the MAN 2A dot matrix display.

Program offered on diskette only.

---

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Assembler/Compiler Used:</th>
</tr>
</thead>
<tbody>
<tr>
<td>A, B, C, H, and L</td>
<td>8080 Assembler, Ver. 2.4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RAM Required:</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 byte</td>
<td>Helene Young Myers</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ROM Required:</th>
<th>Company:</th>
</tr>
</thead>
<tbody>
<tr>
<td>610 bytes</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Maximum Subroutine Nesting Level:</th>
<th>Address:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4313 Judith Street</td>
</tr>
<tr>
<td></td>
<td>Rockville, Maryland 20853</td>
</tr>
</tbody>
</table>
$BLPT

Buffered line printer driver for the Centronix 101A Line Printer, utilizing the Intellec 8-Mod 80

Intellec 8 Mod 80
Centronix 101A Line Printer

Intellec 8 Mod 80 Exec

AL = 1
...
(CHAR IN C)
CALL LO

The assembled line is printed on the line printer when a carriage return character is received.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Maximum Subroutine Nesting Level:</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RAM Required:</th>
<th>Assembler/Compiler Used:</th>
</tr>
</thead>
<tbody>
<tr>
<td>CALL LO</td>
<td>Intellec 8 Macro Assembler</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ROM Required:</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>256 bytes</td>
<td>George Woodley</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Company:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Woodley Associates</td>
</tr>
<tr>
<td>604 Indian Home Road</td>
</tr>
<tr>
<td>Danville, CA 94526</td>
</tr>
</tbody>
</table>

Dec. 1976
Ref. AC22
LIST

THIS ROUTINE IS TO BE USED AS AN MDSCALL IN CONJUNCTION WITH THE ICE80 SOFTWARE DRIVER. WHEN CALLED, IT WILL RETRIEVE THE PC FROM THE ICE80 CONTROL BLOCK, MATCH IT WITH AN ADDRESS IN THE USER'S LIST FILE GENERATED BY ASM80, AND PRINT THE CORRESPONDING SOURCE CODE ON THE CONSOLE. IN THE STEP MODE, THE ENTIRE PROGRAM MAY BE LISTED.

MDS-800 MDS-DOS
ICE-80 CONSOLE

ISIS MDS MONITOR V1.2
ICE-80 ASM80

THE PROGRAM WILL ASK FOR THE USER'S LIST FILE FILENAME ON THE FIRST SUBROUTINE CALL.

A LINE OF SOURCE CODE, STARTING WITH THE LABEL FIELD AND EXTENDING THROUGH THE COMMENT FIELD, IS LISTED ON THE CONSOLE. ISIS ERRORS CAUSE AN ERROR MESSAGE TO BE DISPLAYED AND HALT THE PROCESSOR. A LARGE BLOCK OF COMMENTS OR PSEUDO-OPS IN THE MIDDLE OF THE LIST FILE MAY CAUSE THE ROUTINE TO HANG UP IN A LOOP.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Assembler/Compiler Used:</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL</td>
<td>ISIS 8080 V1.1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RAM Required:</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>435 BYTES</td>
<td>BERNARD J. VERREAU</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ROM Required:</th>
<th>Company:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NATIONAL CASH REGISTER</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Maximum Subroutine Nesting Level:</th>
<th>Address:</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>P.O. BOX 607, MILLSBORO DEL. 19963</td>
</tr>
</tbody>
</table>
Trace Routine

This routine is a debugging aid, which when appended to a program allows
the user to run the program normally, or to single step the program. In
single step mode, he can display breakpoint, jump to the monitor, ISIS,
or proceed, by typing in a character from the system console. The list
of commands is easily expanded.

MDS

MDS System Monitor V 2.0
MAC 80 for assembly

"DEBUG" is one byte which is set to one if the user wishes to run his
program in trace mode; otherwise set to zero.

"SINGLESTEP" is one byte which, when not on will merely print user's
message on the console device. If set to one, the program will single
step and wait for the user to type a character on the console device,
execute the corresponding command before proceeding.

The user must also supply the text message desired.

The program will print out the desired text, singlestep if desired, and
respond to user commands by displaying the breakpoint, or jumping to the
system monitor, or jumping to ISIS, or simply proceeding. The possible
commands can be expanded.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Assembler/Compiler Used:</th>
</tr>
</thead>
<tbody>
<tr>
<td>None - all registers restored</td>
<td>MAC 80</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RAM Required:</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>115 bytes</td>
<td>Ed Klingman</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ROM Required:</th>
<th>Company:</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>Cybernetic Micro Systems</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Maximum Subroutine Nesting Level:</th>
<th>Address:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Three (counting monitor calls)</td>
<td>2460 Embarcadero Way</td>
</tr>
<tr>
<td></td>
<td>Palo Alto, CA. 94303</td>
</tr>
</tbody>
</table>
8080 Symbol Table Dump

This program prints the name and location of all symbols (including macro's), after the first or second pass of the assembler. The program can be assembled in PROM if desired.

Intellec 8/MOD 80 and ASR 33 TTY

Monitor V 3.0
Macro Assembler V 4.1

None

Test tape included to verify macro capability. This program is based on the (8008) "Symbol Table List Routine" (Ref. AB30) by Robert Uleski.

Registers Modified: All
RAM Required: 161 bytes
ROM Required: N/A
Maximum Subroutine Nesting Level: N/A
Assembler/Compiler Used: Macro Assembler V 4.1
Programmer: Earl G. Day
Company: DIT-MCO International Corp.
Address: 5612 Brighton Terr
          Kansas City, MO. 64130
; REF NO. AE3
; PROGRAM TITLE 8080 SYMBOL TABLE DUMP

2A00 ORG 2A00H
3CCD CRLF EQU 3CCDH
3C5D TTY EQU 3C5DH
1C8E SYMBOL EQU 1C8EH
386B START EQU 386BH
3C5C CONV EQU 3C5CH
2A00 CDDC3C CALL CRLF
2A03 0616 MVI B, 22
2A05 CD832A CALL BLK0 ; PRINT THE NUMBER OF BLANKS
          ; IN REGISTER B
2A08 21732A LXI H, MSG ; PRINT "SYMBOL TABLE"
2A0B 060E MVI B, 14
2A0D 4E SYM0: MOV C, M
2A0E 23 INX H
2A0F CD5D3C CALL TTY
A12 05 DCR B
2A13 C2D2A JNZ SYM0
2A16 118E1C LXI D, SYMBOL ; POINT TO BEGINNING OF
 ; SYMBOL TABLE
2A19 CD2B2A LOOP: CALL LOOP0 ; PRINT FOUR SYMBOLS PER LINE
2A1C CD2B2A CALL LOOP0
2A1F CD2B2A CALL LOOP0
2A22 CD2B2A CALL LOOP0
2A25 CDDC3C CALL CRLF
2A28 C3192A JMP LOOP
2A2B 62 LOOP0: MOV H, D ; PRINT A SYMBOL AND ADDRESS SR
2A2C 6B MOV L, E
2A2D 7E MOV A, M
2A2E 0608 MVI B, 8 ; SET TRIES COUNTER.
2A30 CD8D2A CALL ALFA1 ; FIND ALPHA CHAR.
2A33 6B MOV L, E
2A34 0605 MVI B, 5
2A36 4E LOOP1: MOV C, M ; LOAD & PRINT LABEL
2A37 23 INX H
2A38 CD5D3C CALL TTY
2A3B 05 DCR B
2A3C C2362A JNZ LOOP1
2A3F 0E20 MVI C, 20H ; PRINT TWO SPACES
2A41 23 INX H
2A42 CD5D3C CALL TTY
2A45 0E20 MVI C, 20H
2A47 CD5D3C CALL TTY
2A4A 23 INX H
2A4B CD592A CALL BYTE ; PRINT MSB OF ADDRESS
2A4E 2B  DCX  H
2A4F CD592A CALL  BYTE  ;PRINT LSB OF ADDRESS
2A52 CD6C2A CALL  NEX
2A55 CD812A CALL  BLANK  ;PRINT 6 SPACES
2A58 C9  RET
2A59 7E  BYTE:  MOV  A,M  ;PRINT A BYTE AS TWO ASCII CHARACTERS
2A5A 0F  RRC
2A5B 0F  RRC
2A5C 0F  RRC
2A5D 0F  RRC
2A5E EE0F  ANI  0FH
2A60 CD662A CALL  HXD
2A63 7E  MOV  A,M
2A64 EE0F  ANI  0FH
2A66 CD553C  HXD:  CALL  CONV  ;CALLS MONITOR CONVERT SB-RT.
2A69 CD35DC  JMP  TTY
2A6C 7B  MOV  A,E  ;SET D&F TO NEXT SYMBOL
2A6D C608  ADI  8
2A6F 5F  MOV  E,A
2A70 D0  RNC
2A71 14  INR  D
2A72 C9  RET
2A73 53594D42  MSG:  DB  'SYMBOL TABLE', 0DH, 0AH
2A77 4F4C2854
2A7B 41424C45
2A7F 0D0A
2A81 0606  BLANK:  MVI  B,6  ;PRINT 6 BLANKS
2A83 0E20  BLKO:  MVI  C,20H  ;PRINT THE NUMBER OF BLANKS
2A85 CD5D3C  CALL  TTY
2A88 05  DCR  B
2A89 C8  RZ
2A8A C3832A  JMP  BLKO
2A8D 7E  ALFA1:  MOV  A,M
2A8E FE5A  CPI  5AH  ;COMPARE TO 'Z'
2A90 D2992A  JNC  NOALFA  ;JMP IF ACC '<Z'
2A93 FE41  CPI  41H  ;COMPARE TO 'A'
2A95 DA992A  JC  NOALFA  ;JMP IF ACC '<A'
2A98 C9  RET
2A99 23  NOALFA:  INX  H
2A9A 13  INX  D
2A9B 05  DCR  B
2A9C C2802A  JNZ  ALFA1
2A9F C36338  JMP  START
0000  END
STATEMENT-COUNTER

All Assembler statements in a 8080 source file are counted.

Intellec MDS 800

MDS Monitor

The source file to be counted.

Output message on the TTY.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>J. Thommen</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RAM Required:</th>
<th>Company:</th>
</tr>
</thead>
<tbody>
<tr>
<td>640 bytes</td>
<td>MOSTEC, LIESTAL/SWITZERLAND</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ROM Required:</th>
<th>Address:</th>
</tr>
</thead>
<tbody>
<tr>
<td>--</td>
<td>Fraumattstr. 11</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Maximum Subroutine Nesting Level:</th>
<th>City:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CH-4410 Liestal</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Assembler/Compiler Used:</th>
<th>State:</th>
</tr>
</thead>
<tbody>
<tr>
<td>8080 MDS MACRO ASS. Vers. 1.0</td>
<td>Switzerland</td>
</tr>
</tbody>
</table>
Program Title

9600

Function

Re-initialize CRT UART for 9600 Baud

Required Hardware

Intellec MDS development system

Required Software

None

Input Parameters

Output Results

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Programmer: Evan Schaffer</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM Required:</td>
<td>Company: Interface Consultants</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>Address: P. O. Box 952</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td>City: Santa Cruz</td>
</tr>
<tr>
<td>Assembler/Compiler Used:</td>
<td>State: California 95060</td>
</tr>
</tbody>
</table>

© Intel Corporation, 1976
; REF. NO. ABS8
; PROGRAM TITLE 9600 RE-INITIALIZE CRT UART FOR 9600 BAUD
;
;
RE-INITIALIZE CRT TO 9600 BAUD
;

0001 MDS SET 1
0000 INTELLEC SET NOT MDS AND 1

3100 ORG 3100H
START:
3100 3E40 MVI A, 40H
3102 D3F7 OUT CRTC ; SEND RESET COMMAND
3104 3E4E MVI A, 04EH
3106 D3F7 OUT CRTC ; SEND 9600 BAUD COMMAND
3108 3E27 MVI A, 27H
310A D3F7 OUT CRTC ; ENABLE UART
    IF INTELLEC
        JMP *
    ENDIF
    IF MDS
    ENDIF
3110 C0E09 MVI C, EXIT
3110 E 111431 LXI D, EBLK
3111 CD4000 CALL ISIS
;
EBLK:
3114 1631 DW ESTAT
3116 DS 2
    ENDIF
;
00F7 CRTC EQU 0F7H
00F5 TTC EQU 0F5H
0009 EXIT EQU 9
0040 ISIS EQU 64
3100 END START


4-329
SNAP DUMP 8080

To provide register and memory dumps for software debug.

MDS-800 System

MDS Monitor subroutine for list output.

1) Calling Address
2) Options Flags
3) Start Memory Dump Address
4) End Memory Dump Address

Printed Snap Message with Requested Data Displayed.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Programmer:</th>
<th>Company:</th>
<th>Address:</th>
<th>City:</th>
<th>State:</th>
</tr>
</thead>
<tbody>
<tr>
<td>NONE, all are saved/restored</td>
<td>S. G. Thompson</td>
<td>Harris Controls</td>
<td>P. O. Box 430</td>
<td>Melbourne</td>
<td>Florida 32901</td>
</tr>
<tr>
<td>RAM Required: 17 bytes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ROM Required: 554 bytes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level: 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Assembler/Compiler Used: Microtek Version 3.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
CALLING SEQUENCE:

CALL SNAP
DB FLAGS+ID
DW START
DW END

CALLING PARAMETERS:

Flags/ID Byte:

*10NNNNNN* = No Memory Dumped
*01NNNNNN* = No Register Dumped
*11NNNNNN* = No Register or Memory Dumped
*00NNNNNN* = Both Register and Memory Dumped

N = ID# used to identify dump

NOTE: The flag option bits may be specified in any combination

Start Adr./End Adr. Define the start and end of the memory area to be dumped.

CHARACTER OUTPUT SUBROUTINE ASSIGNMENT

The program presently uses the MDS monitors character output subroutine with the reverse break (control "b"). This allows assignment of the output and interruption of the printing by the operator. The equate to L0 may be used to change the assignment.
SAMPLE OUTPUT

SNAP AT: AAAA ID=BB
REGISTERS: A=QQ B=RR C=SS D=TT E=UU H=VV L=WW CC=XX=SZ0AP1C SP=YYYY
CCCC DD DD DD DD DD DD DD DD DD DD DD DD DD DD DD *EEEEEEEEEEEEEEEEEEEEE*

WHERE:
AAAA = Address of Snap In Hex
BB = Snap Identification Number in Decimal
QQ = A Register Contents In Hex
RR = B Register Contents In Hex
SS = C Register Contents In Hex
TT = D Register Contents In Hex
UU = E Register Contents In Hex
VV = H Register Contents In Hex
WW = L Register Contents In Hex
XX = Condition Code Register Contents In Hex
YYYY = Stack Pointer Register Contents In Hex
CCCC = Hex Address of 16 Words of Storage Dumped
DD = Hex Memory Data
E = ASCII Interpretation of Memory Data if (A-Z, 0-9, blank) else a Period.
RTM (real time monitor).

RTM is a small real time operating system which is meant to supervise periodic execution of RT-programs in a fixed priority manner.

It has also a few standard output routines which must be used.

The monitor itself require the INTELLEC 8080 SYSTEM In addition comes a real time clock.

MCS-80 system monitor C0 (console output) is used, but can of course easily be replaced.

Each RT-program must have a description table located from 155H.

Depends on user programs.

Note: To start up RTM supply RST4 via console, start in address F6 and turn on the real time clock.
FORMAT INTEL DATA

Procedure FMINDA is part of a bootloader which loads object code from a cassette tape into the Intellec MDS memory. The object code, from either the MAC80 Assembler or PL/M Compiler (or both), is written onto cassette tape using a T.I. Silent 700. Procedure FMINDA takes each 90 word record containing the object code in ASCII bytes, reformats the data and stores the object code in the designated location in RAM.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Assembler/Compiler Used:</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL</td>
<td>PL/M COMPILER</td>
</tr>
<tr>
<td>RAM Required:</td>
<td>Programmer:</td>
</tr>
<tr>
<td>380 BYTES</td>
<td>VICTOR H. SAUCEDO</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>Company:</td>
</tr>
<tr>
<td></td>
<td>Honeywell Information Systems</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td>Address:</td>
</tr>
</tbody>
</table>

P.O. Box 6000
Phoenix, Arizona 85005
LIST VER 3.0

To display/print ISIS files or MDS Monitor listings on the console in pages with optional line-numbers, optional assembly listing error only display, and for ISIS files, optional listing between given line-numbers only. LIST includes all tabs.

UPDATED VERSION 2/28/77

MDS, Console

MDS Monitor
ISIS (Optional)

Monitor users set D = 0 for no line numbers, else numbers.
E = 0 for normal listing, else error only.
then go to start address.

ISIS users give a LIST as a console command. See the listing for details of syntax and options. e.g.
LIST FILE NE 143, 227 lists only lines in error between 143 and 227 inclusive, with each such line numbered.

MDS Monitor - Sets up the list assignment. All subsequent list output goes through LIST.
ISIS - Displays the required file.
After each page key no. of lines wanted (0 - 9) or escape to ISIS.

Registers Modified: 

Programmer:

RAM Required: 

Company:

ROM Required: 

Address:

Maximum Subroutine Nesting Level: 

City:

Assembler/Compiler Used: 

State:
8080 CPU Exerciser Routine

Designed as an on-line, periodic, exercising program. Executes almost all instruction in the 8080 to ensure proper functioning. A set of instructions is executed to which a predefined result is expected. If result does not match, program halts. See attached sheets.

N/A

None

None

N/A

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>All including SP</td>
<td>W. Iwamoto/R. Lonchar</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RAM Required:</th>
<th>Company:</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 Bytes</td>
<td>North Electric Company</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ROM Required:</th>
<th>Address:</th>
</tr>
</thead>
<tbody>
<tr>
<td>376 Bytes</td>
<td>P. O. Box 20345</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Maximum Subroutine Nesting Level:</th>
<th>City:</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Columbus</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Assembler/Compiler Used:</th>
<th>State:</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISIS 8080 Macro Assembler</td>
<td>Ohio 43220</td>
</tr>
</tbody>
</table>
Requirements and Notes:

A. The 3 Bytes at Locations 40FCH-40FEH are to be RAM locations.

B. Program is address dependent, occuring from location 4102H. The CPI instruction 4107H will have to be changed if RAM addresses do not correspond to the ones in this assembly. Adjustment may also have to be made to the ORI instruction at location 4133H. Accumulator should contain 0C3H after the operation.

C. Numbers at locations 401FH-4024H are random and may be changed. This is also true for the LVI instruction at location 4092H. This instruction requires a RAM address.

D. Instruction at location 4179H should be changed to a JMP or an RET instruction.

No results are generated and no input parameters are needed. Program either passes or halts.
BTP (Binary Tape Programme)

To read, check or punch binary paper tape directly to and from MDS memory using a rubout character as the tape index. Specified blocks or memory can be inverted (complemented). All peripherals are assigned by the resident monitor.

MDS 800
Console Device
Listing Device
Paper Tape Reader
Paper Tape Punch

MDS Monitor

Initial command R,V,P,I,M or D then MDS memory addresses specified in hexadecimal followed by carriage return.

M and D do not require addresses or CR.
The correct format for the other commands is shown in the listing.

Tape will be punched when using P command with blank leader and trailer.
Data is immediately preceded by a rubout character and the data from memory is addressed along sequential locations on the tape.
All commands except P, make software changes only.
An error signal will result if:
1. The memory doesn't write correctly.
2. The memory addresses are not specified correctly.
3. The tape doesn't agree with memory when verifying.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>Jim Arkell</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RAM Required:</th>
<th>Company:</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5K for 48H (37B0 to 37FF) Programme</td>
<td>RAPID RECALL LIMITED</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ROM Required:</th>
<th>Address:</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5K when ROM resident</td>
<td>9 Betterton Street</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Maximum Subroutine Nesting Level:</th>
<th>City:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>London</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Assembler/Compiler Used:</th>
<th>State:</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISIS MACRO 8080 V1.0</td>
<td>WC2H 9BS, England</td>
</tr>
</tbody>
</table>
LERR (List Assembly Errors)

Searches diskette list file for assembly error statements and prints all such statements on console device.

MDS-80, (32K RAM), TTY, Diskette System

ISIS V1.0, MDS Monitor CO routine

ISIS 8080 Macro Assembler Program
list file name, such as "FILE.LST".

Listing on console device of all error containing program statements.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Assembler/Compiler Used:</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>ISIS 8080 Macro Assembler</td>
</tr>
<tr>
<td>RAM Required:</td>
<td>Programmer:</td>
</tr>
<tr>
<td>See Listing</td>
<td>M. Polad</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>Company:</td>
</tr>
<tr>
<td>See Listing</td>
<td>Data Card Corporation</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td>Address: 7625 Parklawn Ave</td>
</tr>
<tr>
<td>See Listing</td>
<td>Minneapolis, MN 55435</td>
</tr>
</tbody>
</table>
; REF. NO. AB63
; PROGRAM TITLE LERR

TITLE 'PROGRAM LERR' 9/16/76

; COMMAND: <LERR FILENAME>

; PROGRAM SEARCHES THROUGH LIST FILE (OUTPUT FROM ASM80)
; PRINTS ALL STATEMENTS CONTAINING ASSEMBLY ERRORS ON THE
; DEVICE.

4000 ORG 4000H

0000 OPEN EQU 0
0003 READ EQU 3
000C ERROR EQU 12
0040 ISIS EQU 64
0009 EXIT EQU 9
000D CR EQU 0DH
30A LF EQU 0AH
F809 CO EQU 0F809H

4000 31A441 BEGIN: LXI SP, STACK+10
4003 0003 MVI C, READ
4005 119B40 LXI D, RBLK
4008 CD7240 CALL SYS
4008 0E00 MVI C, OPEN
400D 11A540 LXI D, OBLK
4010 CD7240 CALL SYS ; OPEN LIST FILE ; READ FILENAME FROM :CI:
4013 CD5840 CALL RD ; READ BLOCK FROM LIST FILE TO BUF
4016 CD5440 LOOP: CALL GNC ; GET A CHAR FROM BUF
4019 79 MOV A, C
401A FE0A CPI LF
401C C21640 JNZ LOOP ; LOOP UNTIL CHAR = LF
401F CD5440 LOOP1: CALL GNC ; GET NEXT CHAR (FIRST ON LINE)
4022 79 MOV A, C
4023 FE0D CPI 0DH
4025 CA1640 JZ LOOP ; IF CHAR = CR OR SPACE, LOOK FOR NEXT LF
4028 FE20 CPI 
402A CA1640 JZ LOOP
402D 79 CPHD: MOV A, C ; SAVE FIRST CHAR ON LINE
402E 32BD40 STA TEMP
4031 CD5440 CALL GNC ; GET 2ND CHAR ON LINE
4034 79 MOV A, C
325 FE20 CPI 
4037 C21640 JNZ LOOP ; IF 2ND CHAR NOT SPACE, IGNORE (PAGE HDG
403A 3ABD40 LDA TEMP ; GET FIRST CHAR
403D 4F MOV C, A

4-348
403E CD09F8 CALL CO ;OUTPUT FIRST CHAR
4041 4E MOV C,M ;GET 2ND CHAR
4042 CD09F8 WRT: CALL CO ;SEND CHAR TOS:CO: UNTIL NEXT LF
4045 CD5440 CALL GNC
4048 79 MOV A,C
4049 FE0A CPI LF
404B C24240 JNZ WRT
404E CD09F8 CALL CO
4051 C31F40 JMP LOOP1

4054 23 GNC: INX H ;GET CHAR FROM BUF
4055 4E MOV C,M
4056 05 DCR B
4057 C5B40 CZ RD ;IF LAST CHAR IN BUF, READ 128 CHAR FROM
405A C9 RET

405B C5 RD: PUSH B
405C 0E03 MVI C,READ
405E 11F40 LXI D,RBLK1
4061 CD7240 CALL SYS
4064 2A3E41 LHLH ACTUAL
4067 7C MOV A,H
4068 B5 ORA L
4069 CAE540 JZ DONE ;IF ACTUAL = 0, DONE
406C C1 POP B
406D 47 MOV B,A
406E 21BD40 LXI H, BUF-1
4071 C9 RET

4072 CD4000 SYS: CALL ISIS
4075 3ABB40 LDA STATUS
4078 B7 ORA A
4079 C27D40 JNZ ERR
407C C9 RET

407D 0E0C ERR: MVI C, ERROR
407F 11B940 LXI D, EBLK
4082 CD4000 CALL ISIS
4085 CD9040 DONE: CALL CRLF
4088 0E09 MVI C,EXIT
408A 11B940 LXI D, XBLK
408D CD4000 CALL ISIS

4090 0E0D CRLF: MVI C, CR
4092 CD09F8 CALL CO
4095 0E0A MVI C, LF
397 CD09F8 CALL CO
409A C9 RET

409B 0100 RBLK: DW 1
409D BE40          DW    BUF
409F 8000          DW    128
40A1 3E41          DW    ACTUAL
40A3 BB40          DW    STATUS

40A5 AF40  OBLK:  DW    AFTN
40A7 BE40          DW    BUF
40A9 0100          DW    1
40AB 0000          DW    0
40AD BB40          DW    STATUS

  RBLK1:

40AF    AFTN:    DS    2
40B1 BE40          DW    BUF
40B3 8000          DW    128
40B5 3E41          DW    ACTUAL
40B7 BB40          DW    STATUS

  EBLK:

40B9 BB40          DW    STATUS

  XBLK:  DW    STATUS

  STATUS:    DS    2
40BD    TEMP:    DS    1
40BE    BUF:    DS    128
413E    ACTUAL:    DS    2
4140    STACK:    DS    10
4000    END    BEGIN
TRACE VER 7.0

TRACE single-step through any user's program for a keyed number of instructions or until DI executed. It displays the user's registers flags, op-code mnemonic and data after each instruction. The monitor routines may be used at any time to control execution. The user's programme may be in RAM, ROM or both.

MDS, Console

MDS Monitor Ver 1

ISIS (Optional)

MDS Monitor users load own program and TRACE and go to TRACE with a breakpoint at their start address. ISIS users may command - TRACE FILENAME where FILENAME is a binary file. Escape to ISIS and to the Monitor is provided with .G (C.R.) to restart TRACE.DI (OF3H) may be put at any number of breakpoints and CLEAR is a run to breakpoint without printing command.

See example.

Note that interrupts below level 2 cannot be used.
CI cannot be used, CO works only on single shot
DI cannot be traced - it acts as a breakpoint
EI is not traced correctly.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>N/A</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>RAM Required:</th>
<th>MDS 1021 bytes ISIS 1107 bytes</th>
</tr>
</thead>
</table>

| ROM Required:       | None                            |

| Maximum Subroutine Nesting Level: | N/A. 4 bytes user's stock |

| Assembler/Compiler Used:          | ISIS 8080 MACRO ASSEMBLER |

| Programmer:                      | C. J. Lusby Taylor. |

| Company:                         | Satchwell Control Sys. Ltd. |

| Address:                         | P. O. Box 57 |

| City:                            | Farnham Road |

| State:                           | SLough, Berks. |
ADCCP Remainder Routine

Generates ADCCP remainder term R (X) as described in ADCCP document X3534/589 draft 5, pages 74-76. The ADCCP generator polynomial is P (X) = X^{16} + X^{12} + X^{9} + 1. See attached note for methodology.

Intel 8080 Microcomputer

This routine can run as a subroutine to a main program. The main program enters the routine with each message byte, most significant byte first. With the initial byte, the remainder, which is in the B and C registers must be preset, which for ADCCP is a value of all ones. Of course, the B and C register values generated by the routine must be returned to the routine along with subsequent message bytes. PUSH, POP, RET, etc. instructions are added according to users design.

A Register - Message data bytes
B Register - High order bits of R (X)
C Register - Low order bits of R (X)

For a string of bytes to be transmitted the R (X) generated by this routine, upon being inverted, can become the final two bytes of the transmitted sequence, i.e. the Frame Check Sequence. For a string of bytes that has been received, B and C will contain 1DH and 0FH respectively, if the transmitted sequence has been received without error. The foregoing assumes that both B and C registers are preset to all ones.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Assembler/Compiler Used:</th>
</tr>
</thead>
<tbody>
<tr>
<td>A, B, C, D, E</td>
<td>8080 Macro Assembler Version 1.0</td>
</tr>
<tr>
<td>RAM Required:</td>
<td>Programmer:</td>
</tr>
<tr>
<td>22H</td>
<td>George H. Mineah</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>Company:</td>
</tr>
<tr>
<td>0</td>
<td>Aeronutronic Ford Corporation</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td>Address: 2880 E. Fountain Blvd.</td>
</tr>
<tr>
<td>Does not alter current maximum</td>
<td>Colorado Springs, CO.  80910</td>
</tr>
</tbody>
</table>
; REF. NO. AB64
; PROGRAM TITLE ADCCP REMAINDER ROUTINE
; ;
; ;
; ; ADCCP REMAINDER ROUTINE
; IN:
; A=MESSAGE BYTE MOST SIGNIFICANT FIRST
; B=HIGH ORDER BITS OF R(X) PRESET TO FFH FOR FIRST BYTE
; C=LOW ORDER BITS OF R(X) PRESET TO FFH FOR FIRST BYTE
; OUT:
; B=HIGH ORDER BITS OF R(X)
; C=LOW ORDER BITS OF R(X)

3D000 ORG 3D00H
3D00 A8 ARR: XRA B
3D01 57 MOV D, A
3D02 0F RRC
3D03 0F RRC
3D04 0F RRC
3D05 0F RRC
3D06 E60F ANI 0FH
3D08 AA XRA D
3D09 5F MOV E, A
3D0A 07 RLC
3D0B 07 RLC
3D0C 07 RLC
3D0D 07 RLC
3D0E E6F0 ANI 0F0H
3D10 A9 XRA C
3D11 47 MOV B, A
3D12 7B MOV A, E
3D13 0F RRC
3D14 0F RRC
3D15 0F RRC
3D16 E61F ANI 1FH
3D18 A8 XRA B
3D19 47 MOV B, A
3D1A 7B MOV A, E
3D1B 0F RRC
3D1C 0F RRC
3D1D 0F RRC
3D1E E6E0 ANI 0E0H
3D20 A8 XRA E
3D21 4F MOV C, A
3D22 76 HLT
0000 END
WIPE

"Automates" the process of file deletion on ISIS diskettes. Only those files with attributes of W = 0 and F = 0 can be wiped.

MDS with floppy disks

ISIS

File name of file that contains replica of normal directory of the diskette to be "wiped". If this name is for example TEMP from - DIR to TEMP$1 then:

1. WIPE TEMP allows one to select files to be deleted.
2. WIPE TEMP$A deletes all files that are not format or write protected.

At console get displayed names of files up for deletion (if condition (1)). Type "Y" to query if you want to delete "N" if not.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>ALL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programmer:</td>
<td>Bob Glossman</td>
</tr>
<tr>
<td>RAM Required:</td>
<td>16K System Minimum</td>
</tr>
<tr>
<td>Company:</td>
<td>Stanford University</td>
</tr>
<tr>
<td>ROM Required:</td>
<td></td>
</tr>
<tr>
<td>Address:</td>
<td>c/o V. Grinich ERL 101</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td></td>
</tr>
<tr>
<td>City:</td>
<td>Stanford</td>
</tr>
<tr>
<td>Assembler/Compiler Used:</td>
<td>MDS MAC 80</td>
</tr>
<tr>
<td>State:</td>
<td>California 94305</td>
</tr>
</tbody>
</table>
TABS

Takes files that use control-I as tab character and expand tabs so that legible listings can be obtained.

MDS with diskette system.

ISIS

Source file and (optionally) destination file, i.e.
-- TABS PROG 1
or
-- TABS PROG 1 to PROG 2

If no destination file used (i.e. to etc. omitted) create a new file PROG1.TBS with tabbing done. Otherwise the file PRO62 will contain the tabbed result.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>--</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM Required:</td>
<td>--</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>--</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td>--</td>
</tr>
<tr>
<td>Assembler/Compiler Used:</td>
<td>MAC80</td>
</tr>
<tr>
<td>Programmer:</td>
<td>Bob Glossman</td>
</tr>
<tr>
<td>Company:</td>
<td>Stanford University</td>
</tr>
<tr>
<td>Address:</td>
<td>c/o V. Grinich IOIERL</td>
</tr>
<tr>
<td>City:</td>
<td>Stanford</td>
</tr>
<tr>
<td>State:</td>
<td>California 94305</td>
</tr>
</tbody>
</table>
INTELLEc MICROCOMPUTER DEVELOPMENT SYSTEM DIAGNOSTIC CONFIDENCE TEST VERSION 1.1

Exercise Intellec Microcomputer Development System to verify functionality.

Intellec MDS 800

System functionality

DISKETTE AVAILABLE IN OBJECT CODE ONLY. LISTING IS IN SOURCE CODE. DISKETTE IS PRICED AT $35.00. PAPER TAPE IS NOT AVAILABLE.

CAUTION: BEFORE USING THIS PROGRAM, READ INSTRUCTIONS THOROUGHLY.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>ALL</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM Required:</td>
<td>16 K</td>
<td>Company:</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>None</td>
<td>Address:</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td>--</td>
<td>City:</td>
</tr>
<tr>
<td>Assembler/Compiler Used:</td>
<td>--</td>
<td>State:</td>
</tr>
</tbody>
</table>
Intellec 8/Mod 80 Text Editor

Text Editor

Intellec 8/Mod 80 and TTY

Mod 80 Monitor

Listing available from Insite for a prepaid $15.00 handling fee. Paper tape is not available except when ordering system. Program Diskette not offered.

Registers Modified: All

RAM Required: prox 4K bytes

ROM Required:

Maximum Subroutine Nesting Level:

Assembler/Compiler Used: PL/M

Programmer:

Company:

Address:

City:

State:

© Intel Corporation, 1976
2708 Prom Programmer for Intellec 8/MOD80

To program, read, and verify a 2708 type PROM of up to 1024 Bytes. This cannot be done using the Prom logic of the Intellec 8/MOD80 or Intellec MDS Monitor because of the special programming algorithm required by the 2708.

Intellec 8 MOD80 or Intellec MDS and Universal Prom Programmer (UPP)

System Monitor

See Text of Program

Listing available form Insite for a prepaid $15.00 handling fee. Paper tape is not available except when ordering system. Program Diskette is not offered.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>Company:</td>
</tr>
<tr>
<td>RAM Required:</td>
<td>Address:</td>
</tr>
<tr>
<td>1.5K</td>
<td>City:</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>State:</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td></td>
</tr>
<tr>
<td>Assembler/Compiler Used:</td>
<td></td>
</tr>
<tr>
<td>8080 Macro Assem.</td>
<td>Ver. 2.3</td>
</tr>
</tbody>
</table>

© Intel Corporation, 1976
Intellec 8/MOD80 Monitor

Controls Intellec 8/MOD80

Routines for

Intellec 8/MOD80

--

See Listing

See Listing

Listing available form Insite for a prepaid $15.00 handling fee. Paper tape is not available except when ordering system. Program Diskette is not offered.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RAM Required:</th>
<th>Company:</th>
</tr>
</thead>
<tbody>
<tr>
<td>2K</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ROM Required:</th>
<th>Address:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Maximum Subroutine Nesting Level:</th>
<th>City:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Assembler/Compiler Used:</th>
<th>State:</th>
</tr>
</thead>
<tbody>
<tr>
<td>8080 Macro Assem. Ver. 2.2</td>
<td></td>
</tr>
</tbody>
</table>
Intellec/MDS Monitor Version 2

Intellec/MDS Interactive Program for Handling Six I/O Devices, and Utility Routines for Display/Modify 8080 Memory and Registers.

Intellec MDS

See Program Listing

Listing available form Insite for a prepaid $15.00 handling fee. Paper tape is not available except when ordering system. Program Diskette not offered.
Utility Macros for 8080

Save time in coding and debugging 8080 programs.

PDP-ll Cross Assembler for 8080 (e.g. Intel Library Ref no. C2)

If other assemblers are used, modify syntax accordingly.

Macro expansions

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Assembler/Compiler Used:</th>
</tr>
</thead>
<tbody>
<tr>
<td>n/a</td>
<td>PDP-ll Cross Assembler</td>
</tr>
<tr>
<td>RAM Required:</td>
<td>Programmer:</td>
</tr>
<tr>
<td>n/a</td>
<td>Alan C. Hui</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>Company:</td>
</tr>
<tr>
<td>n/a</td>
<td>STANDARD ENGINEERING CORP.</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td>Address:</td>
</tr>
<tr>
<td>n/a</td>
<td>44800 Industrial Drive, Fremont CA 94538</td>
</tr>
</tbody>
</table>
I/O Port Exerciser

The program allows the user to output patterns to PPI ports and to read and print input port values under keyboard control.

SBC 80/10, TTY or RSZ32-C compatible terminal

EXERCISER PROM

see write up

see write up

Registers Modified:

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>ALL</th>
</tr>
</thead>
</table>

Programmer:

<table>
<thead>
<tr>
<th>Programmer:</th>
<th>Jeffrey W. Scott</th>
</tr>
</thead>
</table>

RAM Required:

<table>
<thead>
<tr>
<th>RAM Required:</th>
<th>100 bytes</th>
</tr>
</thead>
</table>

Company:

<table>
<thead>
<tr>
<th>Company:</th>
<th>Computer Applications</th>
</tr>
</thead>
</table>

ROM Required:

<table>
<thead>
<tr>
<th>ROM Required:</th>
<th>1024 bytes</th>
</tr>
</thead>
</table>

Address:

<table>
<thead>
<tr>
<th>Address:</th>
<th>3030 Bridgeway</th>
</tr>
</thead>
</table>

Maximum Subroutine Nesting Level:

<table>
<thead>
<tr>
<th>City:</th>
<th>Sausalito</th>
</tr>
</thead>
</table>

Assembler/Compiler Used:

<table>
<thead>
<tr>
<th>State:</th>
<th>California 94965</th>
</tr>
</thead>
</table>

MDS MACRO Assembler
Fly Reader Driver

1. Sets up user IO entry pt. for monitor
2. Generates read CMD for reader
3. Inputs ASCII character and sets carry if no char. for 15 μs

Fly Reader 30 - Teleterminal Corp.
12 Cambridge Street
Burlington, Ma. 01803
(Plugs into TTY plug on MDS 800)

Returns ASCII character in a register; sets carry if no character read in 15 μs.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>William J. Casey</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RAM Required:</th>
<th>Company:</th>
</tr>
</thead>
<tbody>
<tr>
<td>025H</td>
<td>Logic Sciences</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ROM Required:</th>
<th>Address:</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>6420 Hillcroft, # 317</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Maximum Subroutine Nesting Level:</th>
<th>City:</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Houston</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Assembler/Compiler Used:</th>
<th>State:</th>
</tr>
</thead>
</table>
| 8080 MDS Assemb. V. 1.0           | Texas      | 77081
; REF. NO. AB75
; PROGRAM TITLE FLY READER DRIVER

; "FLY READER DRIVER"

3FEF ORG 3FEFH
3FEF 0038 DW 3800H ; READER I/O ENTRY POINT
3800 ORG 3800H
00F9 PTR EQU 0F9H
00FF RTC EQU 0FFH
3800 C5 PUSH B
3801 060F MVI B, 0FH ; INITIALIZE COUNTER
3803 3E08 MVI A, 8 ; GENERATE READ COMMAND PULSE
3805 D3F9 OUT PTR
3807 3E0C MVI A, 0CH
3809 D3F9 OUT PTR
380B DBF9 STAT: IN PTR ; INPUT PTR STATUS
380D E602 ANI 2
380F CA2138 JZ DATA
3812 DBFF IN RTC ; INPUT REAL TIME CLOCK STATUS
3814 DBFF IN RTC
3816 CA0B38 JZ STAT
3819 05 DCR B ; DECREMENT COUNTER
381A C20B38 JNZ STAT
381D AF XRA A
381E 37 STC ; SET CARRY FOR NO DATA IN 15MS FROM READ
381F C1 POP B
3820 C9 RET
3821 DBF8 DATA: IN 0F8H ; INPUT DATA
3823 2F CMA
3824 C1 POP B
3825 C9 RET
0000 END
NON-ENCODED KEYBOARD SUBROUTINE HANDLER

The subroutine functions as a handler for two independent, 16-button, non-encoded keyboards. The handler tests for any key closure, provides contact bounce elimination, allows only two-key rollover, and generates a unique code for the key closure.

(SEE COMMENTS ON ATTACHED SOURCE LISTING)

Pointers to keyboard processing routines (PKYB1, PKYB2) must be assigned by the calling program before entering the subroutine. The handler can be executed on a programmed scan cycle, or on an interrupt basis if the hardware is modified allowing any key depression to generate an external hardware interrupt.

B-REGISTER = key switch code
(Re-arranging the keyboard look-up table assigns a different key switch code to the switch matrix crosspoints.)

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.B,C,D,E,H,L</td>
<td>VITO A. TRUJILLO</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RAM Required:</th>
<th>Company:</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 bytes</td>
<td>COMPUTER CONSULTANT</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ROM Required:</th>
<th>Address:</th>
</tr>
</thead>
<tbody>
<tr>
<td>D4H bytes</td>
<td>2940 BRAUN COURT</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Maximum Subroutine Nesting Level:</th>
<th>City:</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Depends on calling program)</td>
<td>GOLDEN</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Assembler/Compiler Used:</th>
<th>State:</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICOM FDOS-II Macroassembler</td>
<td>COLORADO 80401</td>
</tr>
</tbody>
</table>
Program Title: LOAD

To allow loading of ISIS-II absolute object modules into the area of ROM occupied by ISIS-II. (i.e. below 3180H)

Required Hardware:
Intellic MDS-800
Single or Dual Drive Floppy Disk System
System Console

Required Software:
To Compile: PL/M-80 and ASM80
To Execute: ISIS-II

Input Parameters:
Name of file to be LOADED
e.g.
- LOAD :F1:TEST.PRC

Output Results:
The program will print out the load address and start address of LOADED program on the system console and then exit to the monitor. To execute the program just loaded, use the monitor "GO" command with the start address given by the "LOAD" program.

Available in Object Code with a source listing. Diskette available in source code.

Registers Modified: All

Programmer: Ron Williams

RAM Required:

Company:

ROM Required:

Address:

Maximum Subroutine Nesting Level:

City:

Assembler/Compiler Used:

State:

PLM-80, ASM80

© Intel Corporation, 1976
THIS PROGRAM ALLOWS THE LOADING OF ISIS-II OBJECT MODULES WHICH HAVE
A LOAD ADDRESS WITHIN THE ISIS-II PROGRAM AREA (I.E. BELOW 3180H).
The program to be loaded must be in absolute binary format. Loading
of relocatable modules is not allowed either by the 'LOAD' program
or by the ISIS-II binary loader.

TO EXECUTE THE 'LOAD' PROGRAM, SIMPLY TYPE THE WORD "LOAD" FOLLOWED
BY ONE OR MORE SPACES AND THE NAME OF THE FILE TO BE LOADED.

FOR EXAMPLE: LOAD :F1:COMP.ABS

AFTER THE FILE HAS BEEN LOADED, THE PROGRAM WILL PRINT THE LOAD ADD-
RESS AND START ADDRESS ON THE SYSTEM CONSOLE AND THEN EXIT TO THE
INTELLEC MONITOR. YOU CAN THEN EXECUTE YOUR PROGRAM BY TYPING "G"
FOLLOWED BY YOUR PROGRAM'S START ADDRESS.

FOR EXAMPLE: G100

*******************************************************************************

TO INSTALL THE 'LOAD' PROGRAM IN YOUR ISIS-II SYSTEM USE THE FOLLOWING
STEPS:

1) USING THE ISIS-II TEXT EDITOR, GENERATE THE "SUBMIT" FILE SHOWN
BELOW...

   PLM80 :F1:LOAD.SRC
   ^E (CONTROL/E)
   ASM80 :F1:MOVER.SRC
   LINK :F1:LOAD.OBJ,:F1:MOVER.OBJ,SYSTEM.LIB TO &
   :F1:LOAD.LNK
   LOCATE :F1:LOAD.LNK CODE (3200H)

2) THE CONTROL/E ABOVE ALLOWS YOU TO TAKE OUT THE PL/M-80 DISKETTE
   AND INSERT THE ISIS-II DISKETTE BEFORE CONTINUING WITH THE
   INSTALLATION. AFTER INSERTING THE ISIS-II DISKETTE, TYPE A
   CONTROL/E TO CONTINUE WITH THE ASSEMBLY.

3) THE "SUBMIT" FILE ALONG WITH THE TWO SOURCE FILES SHOULD ALL BE
   ON DRIVE 1. BY PUTTING THE "SUBMIT" FILE ON A FILE CALLED
   'LOAD.CSD' IN DRIVE 1 AND TYPING IN:
   SUBMIT :F1:LOAD
   YOU WILL HAVE THE 'LOAD' PROGRAM ON DRIVE 1.
SCAN

Scans a listing file held on a diskette and prints assembly error on the line printer

MDS 800, Diskette, Line Printer, Terminal

ISIS Operating System

Filename (from terminal)

Prints assembly errors on line printer.

Registers Modified: C. D. Saunders
RAM Required: Company:
955 Bytes + ISIS Marconi Instruments
ROM Required: Address:
Maximum Subroutine Nesting Level: City:
2 Longacres St. Albans
Assembler/Compiler Used: State:
(MDS 800) 8080 U.K.
Program Title: Driver for Tektronix 4010 Grafic Screen

Function: Procedures for controlling Tektronix 4010. These procedures make it possible to use the 4010 as a grafic output unit for 8080.

TTY on port 0 and 1.

Required Hardware:

Required Software:

Input Parameters:

Output Results:

Registers Modified: all
RAM Required: 3/4 K
ROM Required:
Maximum Subroutine Nesting Level: 3
Assembler/Compiler Used:

Programmer: Henning Nielsen
Company: Institut for Elektro-
niske Systemer
Address: Aalborg University Centre
City: P.O.Box 159
9100 Aalborg
State: Denmark

© Intel Corporation, 1976
T.I. SILENT 700 - SBC 80 MONITOR INTERFACE

Allows SBC 80 monitor to be used with T.I. Silent 700 terminal in place of teletype. Tape operations operate at 120 characters/second.

A. SBC, power supply, etc., set-up for 1200/300 baud operation
B. T.I. 733 ASR terminal equipped as follows:
   1) ADC (Automatic Device Control) or RDC (Remote Device Control)
   2) 1200 baud -- if 1200 baud option is not available, replace
      the statement CALL HIGH
      with three (3) NOP instruction wherever CALL HIGH
      appears (at RDRON + 3 and PCHON).

SBC monitor with modifications as described.

Accepts all monitor commands as specified in Intel documentation, with
two exceptions: an additional carriage return is required after the
issuance of and one is required at the completion of the R and W
commands to permit tape operations at 120 characters/sec. The ADC
or RDC option may be omitted if the tape playback/record mechanism
is operated manually.

Identical in operation with SBC 80 monitor, except changing
terminal speed (optional).

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>A, C, F/F's</td>
<td>William M. Seifert</td>
</tr>
<tr>
<td>RAM Required:</td>
<td>Company:</td>
</tr>
<tr>
<td>Same as SBC 80P monitor</td>
<td>Los Alamos Scientific Laboratory</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>Address:</td>
</tr>
<tr>
<td>192 BYTES</td>
<td>P.O. Box 1663, MS/317</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td>City:</td>
</tr>
<tr>
<td>3</td>
<td>Los Alamos</td>
</tr>
<tr>
<td>Assembler/Compiler Used:</td>
<td>State:</td>
</tr>
<tr>
<td>MAC80 Macro Assembler Ver. 2.4</td>
<td>New Mexico 87545</td>
</tr>
</tbody>
</table>
GLANCE

To provide an easy way to visually examine a lengthy diskette resident text file. The file is transferred, line by line, to the CRT console with tab characters replaced by spaces. The display can be frozen and the speed of output changed. Quick jumps of maximum 25600 characters can be specified, both forward and backward. After such a jump the CRT screen will be filled up with text and the display frozen.

NDS 800 with diskette and CRT.

ISIS or ISIS-II
Monitor routines: Console status test (CSTS)
Console input (CI)

Filename of the text file to be examined.
ISIS command: GLANCE filename.
Run time commands: 1. N (numeric) will change the speed of output
2. +N forward jump \( \cdot 50 \cdot 2^N \) characters
3. -N backward jump \( \cdot 50 \cdot 2^N \) characters
4. Control C to exit to ISIS

Text file output to console.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM Required:</td>
<td>Esko Lehtinen</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>Company: AB Bofors, dept. KCN</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td>Address: P.O. Box 500</td>
</tr>
<tr>
<td>Assembler/Compiler Used:</td>
<td>City: S-690 20 Bofors</td>
</tr>
<tr>
<td></td>
<td>State: Sweden</td>
</tr>
</tbody>
</table>

© Intel Corporation, 1976
Keyboard Scanner

Allows one to use a key matrix keyboard as the Console I/O device for a SDK 80 board, (converts key matrix to ASC II.)

SDK 80 board with standard I/O (8255 at 0F4H). Any switch closure key matrix keyboard IK RAM on SDK 80.

SDK 80 Monitor

See attached sheet

ASC II coded character in register A.

Registers Modified:
A, Flags

RAM Required:
IK

ROM Required:
256 bytes

Maximum Subroutine Nesting Level:
2

Assembler/Compiler Used:
Intellec MCS 80 Assembler

Programmer:
Lindsay Weaver

Company:
Mass. General Hospital Anesth.

Address:
Bio-Eng. Unit
Fruit St., Boston, Mass. 02114
Example of How to Interface
to an 80 Key Matrix
ORG 1000H

; WHEN THESE ROUTINES ARE USED TO REPLACE CI AND BREAK
; IN THE SDK MONITOR, THE INSTRUCTIONS:
; MVI A, 95H / OUT 0F7H SHOULD BE ADDED TO THE
; INITIALIZATION SECTION TO INITIALIZE THE 8255

1000 CD1510 CI: CALL SCAN ; SUBSTITUTE FOR THE SDK CONSOLE IN ROUTI
1003 D20010 JNC CI
1006 C9 RET

0218 FRET EQU 218H
0343 SRET EQU 343H

1007 CD1510 BREAK: CALL SCAN ; SUBSTITUTE FOR THE SDK BREAK ROUTINE
100A D20710 JNC BREAK
100D FE1B CPI 27 ; 27=ESC THE SDK BREAK CHAR.
100F CA4303 JZ SRET
1012 C31802 JMP FRET

SCAN:

; SCAN IS A SUBROUTINE THAT SCANS THE KEYS ONCE.
; IF A CHAR. IS TO BE RETURNED THE CARRY BIT IS SET
; AND THE CHAR. IS IN A ON RETURN. IF NO CHAR IS TO BE
; RETURNED THE CARRY IS ZERO ON RETURN
; DESTROYS: A, F/F'S
; Inputs: NONE
; Outputs: A, C-FLAG
; Calls: NOTHING

00F4 PA EQU 0F4H
00FE PC EQU 0F6H
00F7 PCN EQU 0F7H ; 8255 PORT ADDRESSES
1015 3E09 MVI A, 9
1017 D3F6 OUT PC ; SET OUTPUT FOR FIRST SCAN
1019 C5 PUSH B
101A D5 PUSH D
101B E5 PUSH H ; SAVE REGISTERS
101C 0E04 MVI C, 4 ; THIS IS A DELAY TO PREVENT KEY BOUNCE F
101E 06FF SC2: MVI B, 255 ; CAUSING MULTIPLE ENTRIES
1020 85 SC4: DCR B
1021 C2010 JNZ SC4
1024 9D DCR C
1025 C21E10 JNZ SC2
1028 4F MOV C, A ; INIT. C
1029 060A MVI B, 10 ; PUT INDEX IN B
102B 219F11    LXI H, KEY-1
102E 11A911    LXI D, KEYN-1          ;GET POINTERS TO SCAN TABLES
1031 0D      SC05:    DCR C
1032 23      INX H
1033 13      INX D
1034 DBF4    IN PA          ;GET RESULTS
1036 F5      PUSH PSW       ;STORE A
1037 B6      ORA M
1038 2F      CMA            ;(KEYS)*NOT(KEEP(J)+1)
1039 EB      XCHG
103A 77      MOV M, A        ;STORE KEYN(J)
103B 79      MOV A, C        ;STORE KEYS
103C D3F6    OUT PC          ;SELECT 8 KEYS
103E EB      XCHG
103F 1F      POP PSW
1040 2F      CMA
1041 77      MOV M, A        ;STORE KEY(J)
1042 05      DCR B
1043 C23110    JNZ SC05      ;DO THE LOOP 10 TIMES
1046 21A911    LXI H, KEYN
1049 060H    SC20:    MVI B, 10       ;LOAD INDEX IN B
104B 7E      MOV A, M        ;GET BYTE
104C B7      ORA A          ;SET FLAGS
104D C25A10    JNZ SC30      ;IF ONE OF THE BITS IS SET GO FIND ASCII
1050 23      INX H
1051 05      DCR B
1052 C24810    JNZ SC25      ;DO LOOP 10 TIMES
1055 37      STC
1056 3F      CMC           ;IF WE GET HERE NO KEY WAS PRESSED
1057 C37E10    JMP SC99      ;CLEAR CARRY AND RETURN
105A 4F      SC30:    MOV C, A        ;SAVE A
105B 78      MOV A, B
105C 3D      DCR A
105D 87      ADD A
105E 87      ADD A
105F 87      ADD A
1060 47      MOV B, A        ;B=(B-1)*8=J*8
1061 79      MOV A, C
1062 1F      SC35:    RAR
1063 DA6A10    JC SC40       ;GET OUT OF LOOP IF CARRY SET
1066 04      INR B
1067 C26210    JNZ SC35      ;OFFSET NOW IN B, NOW CHECK SHIFT
106A DBF6    SC40:    IN PC        ;OFFSET NOW IN B, NOW CHECK SHIFT
106C 2F      CMA
106D E620    ANI 20H
106F CA7610    JZ SC45
1072 3E50      MVI A, 80
1074 80      ADD B
1075 47      MOV B, A        ;ADD 80 TO B IF SHIFT KEY IS Pressed
1076 210011    SC45:    LXI H, ASCI
1079 7D       MOV    A, L
107A 80       ADD    B
107B 6F       MOV    L, A ; PUT ADDRESS OF ASCII CODE IN H, L
107C 7E       MOV    A, M ; GET ASCII
107D 37       SC50: STC       ; SET CARRY FOR TRUE RETURN
107E E1       SCl99: POP    H
107F D1       POP    D
1080 C1       POP    B
1081 C9       RET

; TEST1 ALLOWS ONE TO FIGURE OUT THE REQUIRED ENTRIES FOR
; THE TABLE ASCII FOR ANY KEYBOARD

1082 210011   TEST1: LXI    H, ASCII
1085 06A0     MVI    B, 160
1088 77       T2: MOV    M, A
1089 23       INX    H
108A 3C       INR    A
108B 05       DCR    B
108C C28810   JNZ    T2
108F 3E98     T3: MVI    A, 98H
1091 D3F7     OUT    PCN
1093 8F       XRA    A
1094 06A0     MVI    B, 10
1095 21A011   LXI    H, KEY
1098 77       T4: MOV    M, A
1099 23       INX    H
109B 05       DCR    B
109C C29910   JNZ    T4 ; INIT. TABLE KEY
109F CD1510   T5: CALL    SCAN
10A2 D29F10   JNC    T5
10A5 CDC032   CALL    02C3H ; CALL NMOUT
10A8 0E0D     MVI    C, 0DH ; PUT CR IN C
10AA CDF401   CALL    01F4H ; CALL ECHO
10AD C39F10   JMP    T5

; ONCE THE TABLE ASCII HAS BEEN FILLED TEST2 ALLOWS ONE
; TO TEST SCAN BEFORE INSTALLATION ON PROM

10B0 3E98     TEST2: MVI    A, 98H
10B2 D3F7     OUT    PCN
10B4 8F       XRA    A
10B5 06A0     MVI    B, 10
10B7 21A011   LXI    H, KEY
10BA 77       T24: MOV    M, H
10BB 23       INX    H
10BC 05       DCR    B
10BD C28A10   JNZ    T24
10C0 CD1510   T25: CALL    SCAN
10C3 D20010 JNC T25
10C6 4F MOV C,A
10C7 CDE301 CALL 01E3H ;CALL CO
10CA C30010 JMP T25

1100 ORG 1100H
1100 ASCII: DS 160 ; THIS IS A TABLE OF ASCII CODES FOR EACH
11A0 KEY: DS 10 ; KEY AND KEYN SHOULD BE IN RAM DURING BO
11AA KEYN: DS 10 ;AND DEBUGGING

0000 END
P2708  PROM Programming Routine

Programs 2708 PROMS with data from ISIS-11 Locater/Assembler. An 8K RAM area is loaded. Any data outside this area is ignored (not loaded). An offset parameter allows selection of user address to Load. Automatic test for erased PROM before programming. Range bell sounds at end of Programming. True/False data. Prints legal commands on request. Compare and Move (transfer). Filename set at initial start-up or can be changed in Load command.

MDS with 32K RAM, UPP, 2708 Personality Card

ISIS-11 V2.2 or Greater

Disk File with absolute object code--can be produced by locater or assembler (using asegl). (Will reject any file that is not Absolute.) See sample run.

REvised 8/8/77

Programmed and Verified 2708 PROMS. See sample run.

Paper tape not available.
Program offered on Diskette only.
Diskette is priced at $35.00.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>John F. Mahony</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RAM Required:</th>
<th>Company:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non Resident</td>
<td>Electra-Physics Labs Inc.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ROM Required:</th>
<th>Address:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non Resident</td>
<td>715 Sutter Street Mall</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Maximum Subroutine Nesting Level:</th>
<th>City:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non Resident</td>
<td>Folsom</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Assembler/Compiler Used:</th>
<th>State:</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISIS-11 8080/8085 Assembler V1.0</td>
<td>California 95630</td>
</tr>
</tbody>
</table>
CALL PROGRAM AND SPECIFY FILE
LOAD COMMAND
ATTENTION TO LOAD WITH "WRONG" OFFSET
CHANGE OF FILE NAME
FILE NON-EXISTANT (ISIS-11 ERROR)
ATTENTION TO LOAD RELOCATABLE FILE
LOAD ORIGINAL FILE
PROGRAM A PROM
TESTS AUTOMATICALLY FOR ERASED PROM
NEW PROM INSERTED; IT PROGRAMS & COMPARES OK
LOWER CASE COMMANDS NOT RECOGNIZED
EXIT CALL PROGRAM WITHOUT FILE NAME
198 PROM PROGRAMMER, V2.0
198 ENTER FILE NAME: P270B
198 NO DATA LOADED
198 INSERT PROM, TYPE CR TO CONTINUE
198 PROM NOT ERASED
198 INSERT PROM, TYPE CR TO CONTINUE
198
UNRECOGNIZED COMMAND
198
198
198
198
198
198
198
198
198
198
198
198
198
198
198

COMPARING PROM WITH RAM; CHANGES MADE ARE FOUND
PRINT OUT LEGAL COMMANDS
LEGAL COMMANDS ARE:

P (PROGRAM PROM)
L (LOAD DISK FILE INTO BK BUFFER)
E (IF PROMPT, RETURN TO ISIS)
T (SET TRUE DATA) (DEFAULT VALUE)
C (COMPARE PROM WITH BLOCK IN BK BUFFER)
M (MOVE PROM INTO BLOCK IN BK BUFFER)
X (FOR THIS LIST OF COMMANDS)

P, M, AND C REQUIRE A 1K BLOCK NUMBER FROM 0 TO 7.
L REQUIRES AN OFFSET PARAMETER THAT SETS THE LOWEST USER ADDRESS IN THE BK BUFFER.
DEFAULT VALUE=0, I.E. USER ADDRESSES FROM 0000H TO 1FFFH ARE LOADED.
WITH A VALUE OF 2000H, ADDRESSES 2000H TO 3FFFH ARE LOADED, ETC.
L ALSO REQUIRE A FILENAME IF THE CURRENT FILE IS TO BE CHANGED.
IF THE LAST NAME ENTERED WILL BE USED.
*** THE PROGRAM WILL REQUEST ANY PARAMETERS THAT IT REQUIRES.

EXIT
Data General to Intellec MDS Diskette Transport Package

RDOS/ISIS is a Data General computer resident diskette file maintenance package supporting Intel ISIS - format files on IBM compatible diskette drives.

Data General computer, TTY console, 32-K memory, IBM compatible diskette drive and controller

Data General RDOS, ALGOL, and assembler.
User-written diskette I/O driver

1. Initialized ISIS diskettes.
2. User files to be copied to diskette.
3. Operator console commands

1. Updated ISIS diskette.
2. User diskette files read from diskette
3. Diskette information:
   a. Sector Usage
   b. Directory Contents

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>N/A</td>
<td>James T. Reynolds</td>
</tr>
<tr>
<td>RAM Required:</td>
<td>Company: SCI Systems, Inc.</td>
</tr>
<tr>
<td>N/A</td>
<td>Address: 8600 South Memorial Parkway</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>City: Huntsville</td>
</tr>
<tr>
<td>N/A</td>
<td>State: ALABAMA 35802</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td></td>
</tr>
<tr>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>Assembler/Compiler Used:</td>
<td></td>
</tr>
<tr>
<td>D 6 ALGOL &amp; Assembler</td>
<td></td>
</tr>
</tbody>
</table>
Main routine DDUMP
Diskette DUMP routine

Subroutines SCAN, VDUMP, MULT

The DDUMP routine dumps the content of the MDS diskette on the LIST device in both Hex and ASCII representation, and displays the error message (if any) on the CONSOLE device. The DDUMP is entered to the CONSOLE device, as a system command of the format DDUMP (:Fx:) (start track), (start sector), (end track), (end sector). (Refer to attached documents for details).

MDS, MDS-DOS, CONSOLE device (CRT), LIST device (Line Printer)

MDS monitor
ISIS-II

DDUMP command parameters, (ISIS-II system call READ)

SCAN routine parameter block

DDUMP IOPB Block (10 bytes)

DDUMP Buffer for sector dump (128 bytes)

Disk dump displayed on LIST device in Hex and ASCII representation, 16 bytes across.

Error message (if any) on the CONSOLE device.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM Required:</td>
<td>Raymond Sin</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>Company: ITT Industries of Canada, Ltd. - SEL Division</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td>Address: P.O. Box 138, Toronto-Dominion Centre</td>
</tr>
<tr>
<td>Assembler/Compiler Used:</td>
<td>City: Toronto M5K 1H1</td>
</tr>
<tr>
<td></td>
<td>State: Ontario</td>
</tr>
</tbody>
</table>
Cross Reference for PAS8Ø PASCAL programs - XREF8Ø

XREF8Ø processes PAS8Ø source programs to give a cross reference listing of all identifiers and frequency tally of key words.

Intellec MDS system with 64K bytes RAM memory, dual floppy disks, and CRT/TTY console.

ISIS-II operating system and PAS8Ø resident PASCAL interpreter.

The input parameters are the MDS file name of a PASCAL source program to be cross referenced and the destination file name of the cross reference listing.

The output result is the specified destination file which has a line numbered listing of the PASCAL source program followed by the cross reference of identifiers by line number occurrence and the key word frequencies.

XREF8Ø is run by loading the PAS8Ø system and then executing the XREF program (XREF8Ø is written in PASCAL). The following is an example of a sample entry with all users inputs underlined.

-PAS8Ø
PAS8Ø VMØ1-ØØ
*XREF(SOURCE.PAS,OUTPUT.XRF)
TERMINATED AT LINE ØØ343
*

This sequence would create a file called OUTPUT.XRF which is the cross reference listing of the PASCAL source program SOURCE.PAS.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>ALL</th>
<th>Programmer: BOB EICHENLAUB</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM Required:</td>
<td>64K BYTES</td>
<td>Company: JOHN FLUKE MFG. CO. INC.</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>NONE</td>
<td>Address: P. O. BOX 43210</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td></td>
<td>City: MTNL. TERRACE</td>
</tr>
<tr>
<td>Assembler/Compiler Used:</td>
<td>PAS8Ø PASCAL COMPILER</td>
<td>State: WASHINGTON</td>
</tr>
</tbody>
</table>
SBC 80/10 INTERACTIVE MONITOR

Resident Interactive Monitor for the SBC 80/10, including the following operator commands: ABORT, DISPLAY, FILL, PROGRAM EXECUTE, INSERT, MOVE, READ PAPER TAPE, SUBSTITUTE, EXAMINE.

SBC 80/10 Singel Board Computer, ASR-33 Tetetype Machine (or equivalent).

(None)

Keyboard commands are described fully in the listing.

Some features of the Interactive Monitor are as follows: All commands are checked for validity before being executed. Paper tape input is buffered to allow checksum validation before being installed. The "Program Execute" command permits the setting and clearing of breakpoints. Provisions are made for a front-panel hardware interrupt switch. The entire Interactive Monitor occupies only 1024 bytes.

Program offered on diskette only.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>Wayne Stahnke</td>
</tr>
<tr>
<td>RAM Required:</td>
<td>Company: Wayne Stahnke Company</td>
</tr>
<tr>
<td>64 Bytes</td>
<td>Address: 2513 28th Street</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>City: Santa Monica</td>
</tr>
<tr>
<td>1024 Bytes</td>
<td>State: California 90405</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>

© Intel Corporation, 1976
I/O Routine for TI Silent 700 Terminal

Operate silent 700 on interrupt driven basis, and perform data transfer in blocks without supervision of main program or executive program

8251 Programmable communication interface using TxRDY and RxRDY as interrupt requests
8214 Priority interrupt control unit using INT output to strobe interrupt vector into input port 8212

Memory location 0038H reserved for RST-7 branching point

**INPUT ROUTINE**
Entry from KSR 733

**OUTPUT ROUTINE**
HL-Address of first output character
B- Number of output characters
Output buffer memory (ADR 1250-129FH)

Input Buffer memory (ADR 1200-124FH), containing input data block terminated with CR character

Character counter (ADR 12A0-12A1H)
Number of characters in input data block

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>Fred Lee</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Company:</th>
</tr>
</thead>
<tbody>
<tr>
<td>University of California</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Department of Pharmacology</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>City:</th>
</tr>
</thead>
<tbody>
<tr>
<td>School of Medicine</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>State:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Los Angeles, CA 90024</td>
</tr>
</tbody>
</table>

© Intel Corporation, 1976

6/8/77 98 034C 4-412
TTY DIAGNOSTIC

The TTY characters are output from the CPU to the TTY punch and console verifying correct operation of the CPU to TTY path. The paper tape which is punched can be read into the CPU, ("Echoing" characters back to the TTY console), to verify the TTY to CPU Data Path.

TTY on Port 0 & 1.

8080 V3.0 Monitor (ORG 3800)

Origin of Punch Routine is 2200 H.
Origin of Read Routine is 221C H.

The TTY Character set is output to the TTY Console and punch in the first portion of the program. Following a RESET and RESTART at 221C H the punched paper tape is read. The characters are output back to the TTY for comparison with the original data. The paper tape may also be verified in the local mode.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>A, B, C, D, Flags</th>
<th>Programmer:</th>
<th>W. E. Barkman</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM Required:</td>
<td>NONE</td>
<td>Company:</td>
<td>Union Carbide Corp.</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>4010 words</td>
<td>Address:</td>
<td>Box Y, Bldg. 9998, Stop 2</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td>1</td>
<td>City:</td>
<td>Oak Ridge</td>
</tr>
<tr>
<td>Assembler/Compiler Used:</td>
<td>8080 V4.1</td>
<td>State:</td>
<td>Tennessee 37830</td>
</tr>
</tbody>
</table>
; REF. NO. AA14
; PROGRAM TITLE TTY DIAGNOSTIC
; TTY INTERFACE DIAGNOSTIC
; CHARACTERS ARE OUTPUT FROM
; CPU AND PUNCH ON TAPE
; CONTINUOUSLY. READING THE
; TAPE ECHOS CHARACTERS BACK
; TO THE TTY.

2200                ORG 2200H
380C                PO EQU 380CH
3806                RI EQU 3806H
3809                CO EQU 3809H
2200 163A           START: MVI D, 58 ; SET LOOP COUNTER
2202 0E21           MVI C, 21H ; SET INITIAL CHARACTER
2204 CD0C38         NEXT: CALL PO
2207 0C             INR C
2208 15             DCR D
2209 C20422         JNZ NEXT
220C 0E0A           MVI C, 0AH ; LINE FEED
20E CD0C38          CALL PO
2211 0E0D           MVI C, 0DH ; CARRIAGE RETURN
2213 CD0C38         CALL PO
2216 C30022         JMP START ; PRESS RESET TO STOP
2219 00             NOP
221A 00             NOP
221B 00             NOP
221C CD0638         NULL: CALL RI ; READ CHARACTER
221F 4F             MOV C, A
2220 CD0938         CALL CO ; ECHO TO TTY
2223 C31C22         JMP NULL
0000                END
RECOVR

The function is to find and recover to a diskette file data accidentally lost while using the ISIS text editors

MDS 800 (32K RAM), Single density DOS, and CRT or TTY

Standard ISIS I or ISIS II system diskette containing
ISIS Text Editor V1.1 or V1.6

See page 2 of attached procedure.

A file on the diskette containing the text editor input buffer contents or text buffer contents as selected.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Programmer:</th>
</tr>
</thead>
</table>
| All                 | Ross E. Morgan

<table>
<thead>
<tr>
<th>RAM Required:</th>
<th>Company:</th>
</tr>
</thead>
<tbody>
<tr>
<td>32K</td>
<td>Intel Corporation (ext. 2031)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ROM Required:</th>
<th>Address:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2880 Northwestern Parkway</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Maximum Subroutine Nesting Level:</th>
<th>City:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stack usage 6 Bytes</td>
<td>Santa Clara</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Assembler/Compiler Used:</th>
<th>State:</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISIS 8080 Macro Assembler V1.0</td>
<td>California 95051</td>
</tr>
</tbody>
</table>
HOW TO USE THE RECOVERY PROGRAM

RECOVR is a program that will find and recover to a diskette file data accidentally lost while using the ISIS TEXT EDITORS.

The data can be lost in one of two places:

1. If you thought you were doing an insert (I) and you get a message:
   "" ILLEGAL IN THIS CONTEXT
   then the data is lost in the input buffer. To recover this data use the "I" switch in the command tail. It doesn't make any difference whether ISIS-1 or ISIS-11 is being used, when data is lost in the input buffer.

2. If you had data in the EDITOR and suddenly it is gone, or someone hits Interrupt 1, or you try to exit the EDITOR and get an ERROR 7 (full diskette); then the data is lost in the text buffer. Whether ISIS-1 or ISIS-11 is being used is important when recovering data from the text buffer. If ISIS-11 was being used, RECOVR defaults to ISIS-11 text buffer and no switch is necessary. If ISIS-1 is being used, then the "I" switch must be set.

An ISIS-1 based RECOVR can recover data from the ISIS-11 buffers and visa-versa.

COMMAND SYNTAX:   -RECOVR[filename [switch]]

"filename" is the name of a diskette file that is to contain the recovered data.

[ ] means optional. Omitting this portion will default RECOVER to the ISIS-11 text buffer.

"switch" is either the letter "I" for the Text Editor input buffer or the number "1" if ISIS-1 was used when the data was lost.

NOTE:

When recovering data from the input buffer the first twelve characters of the file will be permanently lost.
Program Title

MSAVE/MLOAD utilities for MDS-800 with DOS.

Function

MSAVE saves a block of RAM memory into an ISIS file. MLOAD loads RAM memory with an ISIS file.

Required Hardware

MDS -800, DOS, 32K RAM memory

Required Software

MDS monitor V2.0, ISIS I or II

Input Parameters

MSAVE Filename_s A
From? xxxx
To? yyyy
MLOAD Filename_L A
At? zzzz

Output Results

MSAVE creates Filename_s with length yyyyH - xxxxH + 1
MLOAD places in memory Filename_L at location zzzzH

Registers Modified:

ALL

RAM Required:

32K

ROM Required:

-

Maximum Subroutine Nesting Level:

8/4

Assembler/Compiler Used:

ISIS II 8080/8085 Assembler V1.0

Programmer:

(317) 353 3249

Carl Harcourt, B/834

Company:

NAVAL AVIONICS FACILITY, Indpls

Address: 6000 East 21st Street

Indianapolis, Indiana 46218
DISPLAY

A set of general purpose routines for using a Video-memory display device.

A Video-memory type of CRT display controller attached to the 8080 bus. The display should present one character for each byte of memory. It should display reverse video (black on white) for each character with bit 7 set.

None.

The variable CHR contains the character to be displayed by the routine DSWRT. All other routines require no parameters. The address and size of the video memory are Assembly-time variables.

The video-memory will display the desired characters performing scrolling when necessary and allowing full cursor movement. The cursor is simulated by inverting bit 7 of certain memory locations to display reverse video.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>Jeff Kravitz</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RAM Required:</th>
</tr>
</thead>
<tbody>
<tr>
<td>9 bytes + stack + video memory</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ROM Required:</th>
</tr>
</thead>
<tbody>
<tr>
<td>170 (hex) bytes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Maximum Subroutine Nesting Level:</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Assembler/Compiler Used:</th>
</tr>
</thead>
<tbody>
<tr>
<td>8080 Macro Assembler</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Company:</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Address:</th>
</tr>
</thead>
<tbody>
<tr>
<td>197 Fairhaven Blvd.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>City:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Woodbury</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>State:</th>
</tr>
</thead>
<tbody>
<tr>
<td>New York</td>
</tr>
</tbody>
</table>

| 11797 |

8/8/77 Intel Corporation, 1976
Hazeltine 2000 CRT Function Driver

Provides an interface for all ten special functions of the Hazeltine 2000 CRT. See enclosed sheet for details.

Hazeltine 2000 CRT

None

The user selects the following assembly parameters:
(a.) Output port
(b.) Register pair for (x,y) cursor addressing
(c.) The user may choose to assemble out any of the first seven functions on the enclosed sheet via an EQU to reduce memory needs.
(d.) Functions are invoked simply by CALL. The cursor addressing function expects a cursor position, (x,y), in the register pair selected in (b.) above.

The given function is performed on the CRT.

Hazeltine Corporation
Greenlawn, N.Y. 11740

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Programmer: Dan Lasley</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zero (excluding stack)</td>
<td>Company: Memphis State University</td>
</tr>
<tr>
<td>37 to 73 bytes</td>
<td>Address: Computer Services</td>
</tr>
<tr>
<td>One</td>
<td>City: Memphis</td>
</tr>
<tr>
<td>MAC 80 Vers 2.0, Xerox Sigma 9</td>
<td>State: TN 38111</td>
</tr>
</tbody>
</table>
; REF. NO. AB89
; PROGRAM TITLE 'HAZELTINE 2000 CRT FUNCTION DRIVER'

TITLE 'HAZELTINE 2000 CRT FUNCTION DRIVER'

FUNCTION DRIVER FOR HAZELTINE 2000 CRT

WRITTEN BY DAN LASLEY
COMPUTER SERVICES
MEMPHIS STATE UNIVERSITY
MEMPHIS, TN 38152

BEGIN IN Variant DATA...

0000  BPAIR  EQU    0 ; BC REGISTERS
0002  DPAIR  EQU    2 ; DE REGISTERS
0004  HPAIR  EQU    4 ; HL REGISTERS

BEGIN USER-SELECTABLE DATA:

(A) SELECT CRT OUTPUT PORT...

0000  HAZEL  EQU    0 ; ARBITRARILY ZERO

(B) SELECT REGISTER PAIR AS INPUT TO THE "ADCUR" (ADDR CURSOR) ROUTINE.

0004  REG  EQU    HPAIR ; ARBITRARILY "HL"

(C) SELECTIVELY ENABLE THE FOLLOWING FUNCTIONS BY SETT GVEN FLAG NON-ZERO:

(1) "ZXMIT" - TRANSMIT
(2) "ZDELN" - DELETE LINE
(3) "ZINLN" - INSERT LINE
(4) "ZSBAC" - SET BACKGROUND
(5) "ZCFOR" - CLEAR FOREGROUND
(6) "ZSFOR" - SET FOREGROUND
(7) "ZPRIN" - PRINT

0001  ZXMIT  EQU    1 ; INCLUDE ALL FUNCTIONS BY DE
301   ZDELN  EQU    1
0001  ZSBAC  EQU    1
0001  ZINLN  EQU    1
0001  ZCFOR  EQU    1
0001  ZSFOR   EQU   1
0001  ZPRIN   EQU   1

; SELECT A START ADDRESS FOR THE CODE.
1000  HERE   EQU   1000H  ; PUT IT ANY OLD WHERE

; BEGIN CODE...
1000  ORG   HERE

;******************************************************************************************
; * TRANSMIT FUNCTION....
;******************************************************************************************
1000  3E0E  XMIT:
302  C32A10  MVI   A,14  ; 14 IS CONTROL-N
            JMP   DAR  ;"DUMP AND RETURN"
            ENDF

;******************************************************************************************
; * DELETE LINE FUNCTION....
;******************************************************************************************
1005  3E13  DELN:
1007  C33310  MVI   A,19  ; CONTROL-S
            JMP   DAN  ;"DUMP AND WAIT"
            ENDF

;******************************************************************************************
; * SET BACKGROUND FUNCTION....
;******************************************************************************************
100A  3E19  SBACK:
100C  C32A10  MVI   A,25  ; CONTROL-Y
            JMP   DAR
            ENDF

;******************************************************************************************
; * INSERT LINE FUNCTION....
;******************************************************************************************
; IF ZINLN
100F 3E1A INLN: MVI A, 26 ;CONTROL-Z
1011 C3310 JMP DAW
ENDIF
;
;
;******************************************************************************
;* CLEAR FOREGROUND FUNCTION....
;******************************************************************************
;
1014 3E1D CFOR: MVI A, 29 ;CONTROL-SHIFT-M
1016 C3310 JMP DAW
ENDIF
;
;
;******************************************************************************
;* SET FOREGROUND FUNCTION....
;******************************************************************************
;
1019 3E1F SFOR: MVI A, 31 ;CONTROL-SHIFT-O
101B C32A10 JMP DAR
ENDIF
;
;
;******************************************************************************
;* PRINT FUNCTION....
;******************************************************************************
;
101E 3E1E PRINT: MVI A, 30 ;CONTROL-SHIFT-N
1020 C32A10 JMP DAR
ENDIF
;
;
;******************************************************************************
;* CLEAR SCREEN FUNCTION....
;******************************************************************************
;
323 3E1C CLEAR: MVI A, 28 ;CONTROL-SHIFT-L
1025 C3310 JMP DAW
;
; HOME CURSOR FUNCTION...
; FALL INTO "DAR".
;
;=============================================================================
; 1028 3E12 HOME:  MVI  A,18   ;CONTROL-R
; 102A F5  DAR:   PUSH PSW   ;SAVE CURRENT FUNCTION
; 102B 3E7E MVI  A,126   ;DUMP LEAD-IN (CONTROL-SHIFT
; 102D D300 OUT  HAZEL   ; PERIOD) TO CRT.
; 102F F1  POP  PSW   ;RESTORE FUNCTION
; 1030 D300 OUT  HAZEL   ;PERFORM FUNCTION AND RETURN
; 1032 C9  RET   ;CALLER WITHOUT DELAY.
;
;=============================================================================
; "DUMP AND WAIT" ROUTINE.
;=============================================================================
;
; 1033 CD2A10 DAW:  CALL DAR   ;OUTPUT LEAD-IN AND FUNCTION
; 1036 3E7F MVI  A,127   ;CERTAIN FUNCTIONS REQUIRE 0
; 1038 D300 OUT  HAZEL   ;MORE DELAY CHARACTERS AT
; 133A D300 OUT  HAZEL   ;BAUD RATES <= 1200 BAUD
; 103C C9  RET   
;
;=============================================================================
; ADDRESS CURSOR FUNCTION. LEFT REGISTER OF REGISTER PAI
; CONTAINS THE X (COLUMN) COORDINATE. RIGHT REGISTER CON
; THE Y (ROW) COORDINATE. NOTE THAT USER MAY SELECT ANY
; PAIRS BC, DE, OR HL VIA ASSEMBLY PARAMETERS.
;=============================================================================
;
; 103D 3E11 ADCUR:  MVI  A,17   ;CONTROL-Q
; 103F CD2A10 CALL DAR   ;OUTPUT LEAD-IN AND CONTROL-
; 1042 7C MOV  A,REG   ;MOVE X COORDINATE
; 1043 D300 OUT  HAZEL   ;TO OUTPUT PORT.
; 1045 7D MOV  A,REG+1   ;MOVE Y COORDINATE
; 1046 D300 OUT  HAZEL   ;TO OUTPUT PORT.
; 1048 C9  RET   
; 0000 END
Program Title

Intel Format Hex Data File Load/Read

Function

Loads Hex files from paper tape, etc., into memory.

Required Hardware

8080 System. Paper Tape Reader.

Required Software

None

Input Parameters

Status bit for read data available is expected on Port 2, bit 7. Data is read and echoed on Port 3.

Output Results

Data read is converted to binary and placed in memory beginning at address specified in beginning of record.

Registers Modified:

A, B, D, H, L, 'M'

Programmer:

Gary Tock/Doug Aamold

RAM Required:

144 Bytes

Company:

Micro Control Company

ROM Required:

None

Address:

7956 Main Street, North East

Maximum Subroutine Nesting Level:

2

City:

Minneapolis

Assembler/Compiler Used:

ISIS 8080 Macro Assembler

State:

Minnesota 55432
; REF. NO. AB90
; PROGRAM TITLE INTEL FORMAT HEX DATA FILE LOAD/READ
;
;
0000 ORG 0000H
;
; THIS IS A PROGRAM FOR READING INDUSTRY STANDARD
; (INTEL) FORMAT HEX DATA
;
; THE 'READ DATA AVAILABLE' SIGNAL IS EXPECTED
; ON PORT 2 BIT 7. THE ASCII DATA IS EXPECTED
; ON PORT 3. THE DATA READ IS ALSO ECHOED ON
; PORT 3.

0000 CD4200 LOOP1: CALL INCH ; GET AND ASCII CHAR. W/O PARITY.
0003 FE3A CPI ";" ; BEGINNING OF RECORD INDICATOR?
0005 C20000 JNZ LOOP1 ; KEEP LOOKING IF NOT
0008 CD2800 CALL GETBT ; GET A BINARY BYTE
                      ; LENGTH FOR THE RECORD.

000B B7 ORA A ; IF ZERO, WE'RE DONE.
000C CA2700 JZ EOF ; END OF FILE. HALT.
000F 47 MOV B, A ; SAVE LENGTH INDICATOR.
0010 CD2800 CALL GETBT ; GET 1ST ADDRESS FOR RECORD.
0013 67 MOV H, A ; MOST SIGNIFICANT BYTE
0014 CD2800 CALL GETBT ; ; LEAST SIGNIFICANT BYTE
0017 6F MOV L, A ; SKIP RECORD TYPE INDICATOR.
0018 CD2800 LOOP2: CALL GETBT ; GET A BINARY BYTE OF DATA
001B CD2800 MOV M, A ; STORE IN MEMORY.
001E 77 INX H ; NEXT ADDRESS.
0020 05 DCR B ; DECREMENT LENGTH INDICATOR.
0021 CD1800 JNZ LOOP2 ; MORE BYTE THIS RECORD?
0024 C30000 JMP LOOP1 ; GET NEXT RECORD.

0027 76 EOF: HLT ; END OF FILE STOP

0028 CD3500 GETBT: CALL INDIG ; GET A NUMBER
002B 87 ADD A ; MOVE LOW 4 BITS
002C 87 ADD A ; TO HIGH 4 BITS
002D 87 ADD A
002E 87 ADD A
02F 57 MOV D, A ; SAVE RESULT.
0030 CD3500 CALL INDIG ; GET ANOTHER NUMBER
0033 B2 ORA D ; 'ADD IN' TO FORM BINARY BYTE
0034 C9 RET
0035 CD4200 INDIG: CALL INCH ; GET A CHARACTER (NUMBER).
0038 FE3A CPI '9+1 ; 0-9 ?
003A FA3F00 JM IND1 ; STRIP OFF UPPER BITS.
003D C609 ADI 9
003F E60F IND1: ANI 0FH ; STRIP OFF UPPER 4 BITS.
0041 C9 RET
0042 DB02 INCH: IN 2 ; GET STATUS.
0044 E680 ANI 80H ; READ DATA AVAILABLE ?
0046 CA4200 JZ INCH ; KEEP LOOKING IF NOT READY?
0049 DB03 IN 3 ; GET A CHARACTER.
004B D303 OUT 3 ; ECHO CHAR. TO PRINT DEVICE
004D E67F ANI 7FH ; STRIP OFF PARITY
004F C9 RET
0000 END
"DATCON.B1" Analog to Digital Conversion Program

The A/D Conversion Program represents the minimum software of Burr Brown's MP 8416 analog I/O system. It works in dialogue mode. The data conversion process begins after selection of the desired range. Each voltage data is represented by a 12 bit two's complement binary number. The program converts this binary number into a real decimal number and outputs one value per line with sign and voltage specification. After acquisition of ten measurements the program ends or continues depending on user's input. During console input incorrect numbers are recognized and an error is announced. After input of "DATCON.B1" and carriage return the program starts under ISIS control.

TTY, MDS 800, Floppy Disc, Analog I/O System MP8416 (Burr Brown)

MDS 800 software and ISIS software (16K version is sufficient)

The only necessary input parameters are the four numbers "1,2,3,4" and the two characters "Y and N"

Output results are the commentary of the dialogue program and the converted voltage data.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>./</td>
<td>Helmut Klie</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RAM Required:</th>
<th>Company:</th>
</tr>
</thead>
<tbody>
<tr>
<td>850 bytes</td>
<td>Biomed. Technik, Medizinische Hochschule Hannover</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ROM Required:</th>
<th>Address:</th>
</tr>
</thead>
<tbody>
<tr>
<td>./</td>
<td>Nobeling 25</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Maximum Subroutine Nesting Level:</th>
<th>City:</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>3000 Hannover 61</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Assembler/Compiler Used:</th>
<th>State:</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISIS 8080 Assembler (16K version)</td>
<td>Federal Republic of Germany</td>
</tr>
</tbody>
</table>

* Intel Corporation, 1976
Program Title

FAST

Function

This program will reset the CRT USART to operate at four times the normal baud rate. This is done by using the divide by 4 rate instead of the divide by 16 which the monitor uses.

Required Hardware

MDS with 16K, Disk, CRT on ports 246 and 247

Required Software

MDS monitor, ISIS

Input Parameters

No input parameters. Called from ISIS

Eg.: -FAST

Output Results

A message will appear asking you to change the CRT Baud rate and type any character. At program completion, control is returned to ISIS.

Registers Modified:

N/A

Programmer:

Gerry de Koning

RAM Required:

108 BYTES

Company:

Toronto Transit Commission

ROM Required:

NONE

Address:

1900 Yonge Street

Maximum Subroutine Nesting Level:

1 call

City:

Toronto

Assembler/Compiler Used:

MDS Macro Assembler

State:

Ontario M4S 1Z2

8/8/77

Intel Corporation, 1976
;;; REF. NO. AB93
;;; PROGRAM TITLE FAST

PROGRAM: FAST

PROGRAMMER: GERRY DE KONING
C.I.S. PROJECT
TORONTO TRANSIT COMMISSION

ENVIRONMENT: HARDWARE: INTELLEC MDS WITH CRT CONSOLE A
DISK SYSTEM.
SOFTWARE: MDS MONITOR V2.0
ISIS V1.2

DESCRIPTION: THIS PROGRAM WILL SWITCH THE MDS INTERFACE
TO FOUR TIMES THE DEFAULT SPEED. A MESSAG
THE USER TO SWITCH TO CRT TO THE FAST BAUD
AFTER THIS IS DONE, THE USER ENTERS A CHAR
ON THE CONSOLE INPUT DEVICE. THE PROGRAM W
RETURN TO ISIS.
EQUATES:

00F7 CRTCONTROL EQU 0F7H
00F6 CRDATA EQU 0F6H
0040 ISIS EQU 40H
F809 CO EQU 0F809H
F803 CI EQU 0F803H

MAIN PROGRAM

3300 ORG 3300H
3300 0628 FAST: MVI B, MSGLEN ; OUTPUT MESSAGE
3302 213F33 LXI H, MSG
3305 4E FAST0: MOV C, H
3306 CD09F8 CALL CO
3309 23 INX H
330A 05 DCR B
308 C20533 JNZ FAST0

330E 21FFFF DELAY: LXI H, 0FFFFH ; DELAY SO THAT LF CAN PRINT
3311 110100 LXI D, 1
3314 2B DLY0: DCX D
3315 19 DAD D
3316 2B DCX H
3317 D21433 JNC DLY0

331A F3 DI ; CLEAR CRT USAR T
331B AF XRA A
331C D3F7 OUT CRTCONTROL ; RESET USAR T
331E D3F7 OUT CRTCONTROL ; USART MODE:
3320 D3F7 OUT CRTCONTROL ; ASYNC *16
3322 3E40 MVI A, 40H ; 8 BITS
3324 D3F7 OUT CRTCONTROL ; NO PARITY
3326 3E4E MVI A, 4EH ; ONE STOP BIT
3328 D3F7 OUT CRTCONTROL ; USART COMMAND:
332A 3F37 MVI A, 37H ; RTS, ERROR RESET, R*EN, DTR
332C D3F7 OUT CRTCONTROL ; 32E FB EI ; READ TO CLEAR DATA
332F DBF6 IN CRDATA ; WAIT TILL CHARACTER HIT.
3331 CD03F8 CALL CI
3334 0E09    MVI    C, 9     ;EXIT TO ISIS
3336 113733  LXI    D, $+1
3339 CD4000  CALL   ISIS
333C C30800  JMP    8     ;ISIS RESTART IF UNSUCCESSFUL EX
333F 53455420 MSG:   DB     'SET CRT RATE TO FAST AND HIT CHARACTER', 13, 10
3343 43525420
3347 52415445
334B 20544F20
334F 46415354
3353 20414E44
3357 20484954
335B 20434841
335F 52414354
3363 4552000A
0028 MSGLEN EQU    $-MSG
3300 END      FAST
SLOW

SLOW is the undoing of FAST. If the CRT baud rate has been changed, SLOW will restore the CRT USART to its default speed.

MDS with 16K, Disk, CRT on ports 246 and 247

MDS Monitor, ISIS

No input parameters; called from ISIS

Eg.: -SLOW

A message will appear asking you to change the CRT baud rate and type any character. At program completion, control is returned to ISIS.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>N/A</td>
<td>Gerry de Koning</td>
</tr>
<tr>
<td>RAM Required:</td>
<td>Company:</td>
</tr>
<tr>
<td>108 BYTES</td>
<td>Toronto Transit Commission</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>Address:</td>
</tr>
<tr>
<td>NONE</td>
<td>1900 Yonge Street</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td>City:</td>
</tr>
<tr>
<td>1 call</td>
<td>Toronto</td>
</tr>
<tr>
<td>Assembler/Compiler Used:</td>
<td>State:</td>
</tr>
<tr>
<td>MDS Macro Assembler</td>
<td>Ontario M4S 1Z2</td>
</tr>
</tbody>
</table>
PROGRAM: SLOW

PROGRAMMER: GERRY DE KONING
C.I.S. PROJECT
TORONTO TRANSIT COMMISSION

ENVIRONMENT: HARDWARE: INTELLEC MDS WITH CRT CONSOLE AND DISK SYSTEM.
SOFTWARE: MDS MONITOR V2.0
ISIS V1.2

DESCRIPTION: THIS PROGRAM WILL SWITCH THE MDS INTERFACE TO THE DEFAULT BAUD RATE. A MESSAGE WILL BE DISPLAYED UNTIL THE USER SET THE CRT TO THE SLOW BAUD RATE. WHEN THIS IS DONE, THE USER ENTERS A CHARACTER AND THE PROGRAM RETUNS TO THE ISIS.
EQUATES:

00F7 CRTCONTROL EQU 0F7H
00F6 CRTDATA EQU 0F6H
0040 ISIS EQU 40H
F803 CI EQU 0F803H
F809 CO EQU 0F809H

MAIN PROGRAM

3300 ORG 3300H
3300 0628 SLOW: MVI B, MSGLEN ; OUTPUT MESSAGE
3302 213F33 LXI H, MSG
3305 4E SLOW0: MOV C, M
3306 CD09F8 CALL CO
3309 23 INX H
330A 05 DCR B
330B C20533 JNZ SLOW0

330E 21FFFF DELAY: LXI H, 0FFFFH ; DELAY SO THAT LF CAN PRINT
3311 110100 LXI D, 1
3314 2B DLY0: DCX H
3315 19 DAD D
3316 2B DCX H
3317 D21433 JNC DLY0

331A F3 DI ; CLEAR CRT USART
331B AF XRA A ; CLEAR CUE Bits
331C D3F7 OUT CRTCONTROL ; CLEAR CUE Bits
331E D3F7 OUT CRTCONTROL ; CLEAR CUE Bits
3320 D3F7 OUT CRTCONTROL ; CLEAR CUE Bits
3322 3E40 MVI A, 40H ; RESET USART
3324 D3F7 OUT CRTCONTROL ; RESET USART MODE:
3326 3E4F MVI A, 4FH ; ASYNC *64
3328 D3F7 OUT CRTCONTROL ; 8 BITS
332A 3E37 MVI A, 37H ; NO PARITY
332C D3F7 OUT CRTCONTROL ; ONE STOP BIT
332E FB EI ; USART COMMAND:
332F D0F6 IN CRTDATA ; RTS, ERROR RESET, R*EN, DTR,
3331 CD03F8 CALL CI ; READ TO CLEAR DATA
3333 CD03F8 CALL CI ; WAIT TILL CHARACTER ENTERED.
3334    0E09       MVI       C,9       ;EXIT TO ISIS
3336    113733     LXI       D,$+1
3339    CD4000     CALL      ISIS
333C    C30800     JMP       8       ;ISIS RESTART IF UNSUCCESSFUL EX
333F    53455420   MSG:      DB       'SET CRT RATE TO SLOW AND HIT CHARACTER',13,10
3343    43525420
3347    52415445
334B    20544F20
334F    534C4F57
3353    20414E44
3357    20484954
335B    20434841
335F    52414354
3363    45520D0A
0028    MSGLEN    EQU      $-MSG
3300    END       SLOW
5 LEVEL (BARDOT) TO 8 LEVEL (ASCII) PAPER TAPE CONVERSION

CONVERTS 5 LEVEL BARDOT PAPER TAPES FROM TELEX, TWX, OR
AMATEUR RTTY TO ASCII.

INTELLEC 8/MOD 80 WITH ASR 33 TTY ON PORTS 0 AND 1.

SYSTEM MONITOR VERSION 3.0

BARDOT TAPE IS PLACED IN TTY READER AS SHOWN BELOW.
READER = 1 IS ASSIGNED WITH SYSTEM MONITOR.

WHEN READER INPUT ROUTINE IS CALLED THE PROPER ASCII
CHARACTER IS RETURNED IN THE ACCUMULATOR.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>A, F (HL RESTORED)</td>
<td>MARK D. HANSEN</td>
</tr>
<tr>
<td>RAM Required:</td>
<td>Company: INSTRUMENTATION SPECIALTIES</td>
</tr>
<tr>
<td>ROM Required: 90 HEX BYTES</td>
<td>Address: P.O. BOX 5347</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td>City: LINCOLN, NEBRASKA 68505</td>
</tr>
<tr>
<td>Assembler/Compiler Used:</td>
<td>State:</td>
</tr>
<tr>
<td>version 3.0</td>
<td></td>
</tr>
</tbody>
</table>
5 TO 8 LEVEL PAPER TAPE CONVERSION PROGRAM

MARK D. HANSEN  5/28/77

REQUIRES INTELLEC 8/MOD 80
SYSTEM MONITOR VERSION 3.0

PLACE 5 LEVEL (BARDOT) TAPE IN ASR 33
READER WITH THE TAPE COVERING THE LEFT
5 READER PINS.

READER ASSIGNMENT = 1 FOR CHARACTERS
                   = 2 FOR FIGURES

USE MONITOR COMMAND "AR=1" TO CHANGE TO
BARDOT MODE

READER JUMP VECTORS ARE AT 3706H (R=1)
    AND 3709H (R=2)

3706  ORG 3706H
3706 C31B37  JMP LTRIN
3709 C33437  JMP FIGIN

3718  ORG 3718H

LTRIN:
371B CD6137  CALL FAKE  ; TTY READER IN
371E D8  RC  ; RETURN IF NO TAPE
371F E61F  ANI 0001111B  ; MASK OFF LOWER 5 BITS
3721 FE1B  CPI FIGS  ; COMPARE WITH FIGURES CHARACTER
3723 CA4B37  JZ FLIP  ; IF FIGURES CHANGE TO READER=2
3726 FE1F  CPI LTRS  ; IF LETTERS CHARACTER
3728 CA1B37  JZ LTRIN  ; IGNORE AND GET ANOTHER CHARACTER
372B E5  PUSH H
372C 216537  LXI H, TABLETR  ; POINT TO START OF CONVERSION TABLE

FIN:
372F 85  ADD L  ; ADD INPUT CHARACTER TO POINT TO
3730 6F  MOV L, A  ; PROPER TABLE ENTRY
3731 7E  MOV A, M  ; GET PROPER ASCII CHARACTER
3732 E1  POP H  ; RESTORE H & L REGISTERS
3733 C9  RET  ; RETURN WITH PROPER ASCII CHARACTER IN A

FIGIN:
3734 CD6137 CALL FAKE ;GET READER INPUT FROM TTY
3737 D8 RC ;RETURN IF NO TAPE LEFT
3738 E61F ANI 00011111B ;MASK OFF LOWER 5 BITS
373A FE1F CPI LTRS ;COMPARE WITH LETTERS CODE AND
373C CA5637 JZ FLOP ;SWITCH TO LETTERS MODE (R=1)
373E FE1B CPI FIGS ;IF FIGURES CHARACTER IGNORE
3741 CA3437 JZ FIGIN ;AND GET ANOTHER
3744 E5 PUSH H
3745 218537 LXI H, TABFIG ;POINT TO FIGURES TABLE
3748 C32F37 JMP FIN ;GO FINISH LIKE LETTERS

;FLIP:
374B 3A0300 LDA 3 ;SWITCH FROM LETTERS TO FIGURES MODE
374E F604 ORI 00000100B ;BY CHANGING THE I/O STATUS BYTE
3750 320300 STA 3 ;FROM READER=1 TO READER=2
3753 C33437 JMP FIGIN

;FLOP:
3756 3A0300 LDA 3 ;SWITCH FROM FIGURES TO LETTERS MODE
3759 E6FB ANI 11111011B ;BY CHANGING THE I/O STATUS BYTE
375B 320300 STA 3 ;FROM READER=2 TO READER=1
375E C31B37 JMP LTRIN

;FAKE:
3761 E5 PUSH H ;PUSH DOWN ONE LEVEL OF STACK TO
3762 C3AF3E JMP 3EAFH ;COMPENSATE FOR POP AT END
3763 3EAFH ;TTY READER ROUTINE IN MONITOR

;LETTERS MODE LOOKUP TABLE

;TABLTR:
3765 00 DB 0 ;BLANK
3766 54 DB 54H ;T
3767 0D DB 0DH ;R
3768 4F DB 4FH ;O
3769 20 DB 20H ;SPACE
376A 48 DB 48H ;H
376B 4E DB 4EH ;N
376C 4D DB 4DH ;M
376D 0A DB 0AH ;LF
376E 4C DB 4CH ;L
376F 52 DB 52H ;R
3770 47 DB 47H ;G
3771 49 DB 49H ;I
3772 50 DB 50H ;P
3773 43 DB 43H ;C
3774 56 DB 56H ;V
3775 45 DB 45H ;E
3776 5A DB 5AH ;Z
3777 44 DB 44H ; D
3778 42 DB 42H ; E
3779 53 DB 53H ; S
377A 59 DB 59H ; Y
377B 46 DB 46H ; F
377C 58 DB 58H ; X
377D 41 DB 41H ; A
377E 57 DB 57H ; W
377F 4A DB 4AH ; J
3780 00 DB 00H ; FIGS
3781 55 DB 55H ; U
3782 51 DB 51H ; O
3783 4B DB 4BH ; K
3784 00 DB 00H ; LTRS

; FIGURES MODE LOOKUP TABLE

; TABFIG:
3785 00 DB 00H ; BLANK
3786 35 DB 35H ; 5
3787 0D DB 0DH ; CR
3788 39 DB 39H ; 9
3789 20 DB 20H ; SPACE
378A 23 DB 23H ; #
378B 2C DB 2CH ;
378C 2E DB 2EH ;
378D 0A DB 0AH ; LF
378E 29 DB 29H ;)
378F 34 DB 34H ; 4
3790 26 DB 26H ; &
3791 38 DB 38H ; 8
3792 30 DB 30H ; 0
3793 3A DB 3AH ; :;
3794 3B DB 3BH ; ;
3795 33 DB 33H ; 3
3796 22 DB 22H ; "
3797 24 DB 24H ; $
3798 3F DB 3FH ; ?
3799 07 DB 07H ; BELL
379A 36 DB 36H ; 6
379B 21 DB 21H ; !
379C 2F DB 2FH ; /
379D 2D DB 2DH ; -
379E 32 DB 32H ; 2
379F 27 DB 27H ; /
7A0 00 DB 00H ; FIG
7A1 37 DB 37H ; 7
7A2 31 DB 31H ; 1
7A3 28 DB 28H ; C
7A4 00 DB 00H ; LTRS
0000      END

001B      FIGS EQU 00011011B
001F      LTRS EQU 00011111B

Sample Automatic Test Equipment Program

Test program for a single board microcomputer-based Automatic Test Equipment to test digital circuits up to 24 I/O lines. When more memory is available the system can easily be expanded to test circuits up to 156 I/O lines. In that case, an MDS microcomputer system with diskett is advisable. Then the test data for large circuits can be stored on a diskett. (For more information refer to attached thesis).

TTY or CRT, special ATE interface circuitry which costs less than $2.00 per I/O line. This interface can be mounted on the unused space of the SDK-80 single board for 24 lines of I/O.

SKD-80 or SBC 80/10 monitor.

1. Tests to be applied to the circuit under test written in a special format. This data can be stored into the RAM via keyboard or can be stored on a ROM for the circuits more frequently tested. A high level language program is in progress to provide the test data in the required format.

2. Commands to the test program (test data address and repetition parameter). For more information refer to Chap. IV of my thesis (ATE user's instructions).

Test results will be written on console. If only the detection of faults requested the output message will indicate whether the circuit is all right or faulty (4000 tests/sec. less memory requirement for test data). If the location of the fault is then appropriate, diagnostic messages will be written on console.

Ideal to test SSI, MSI integrated circuits. Can be used to test IC's in digital labs and for educational purposes. Single board ATE can be constructed with approximate hardware cost of $300.00.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>ALL</th>
<th>Programmer: Hamid T. Hashemi</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM Required:</td>
<td>129 bytes minimum</td>
<td>Company: Computer Science Dept.</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>1K bytes, C2708 or equiv.</td>
<td>Address: SUNY at Potsdam, N.Y.</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td>3</td>
<td>City: Potsdam,</td>
</tr>
<tr>
<td>Assembler/Compiler Used:</td>
<td>None</td>
<td>State: New York 13676</td>
</tr>
</tbody>
</table>
SBC COMMUNICATOR

This routine runs in an Intel SBC 80/10(20) and provides the ability to load and execute programs from 1) the terminal (TTY) connected to the 80/10 board USART or 2) from any host CPU connected to the USART on an Intel SBC 104 board. It also allows the microcomputer board(s) to become transparent so that regular terminal communication with the host CPU is possible.

Intel SBC 80/10 and SBC 104

-------- 8251 --------- 8251 ---------

TTY 80/10 80/10 host
-------- ---------

None if program is allowed to reside at location OOH
SBC 80P MONITOR is useful.

Right bracket followed by L indicates load stream coming
(ASC11 5DH)
Right bracket followed by S indicates start address coming
any other characters get passed directly to the other serial
stream (from TTY to host or from host to TTY)

Host communication or
loaded file in the SBC memory or
started SBC program.

Registers Modified:
ALL

RAM Required:
2 loc. + stack

ROM Required:
189H locations + SBC Monitor if used

Maximum Subroutine Nesting Level:
?

Assembler/Compiler Used:
PDP11 Cross Assembler

Programmer:
Rex Tracy

Company:
Colorado State University

Address:
Elec. Engr. Dept. - CSU

City:
Ft. Collins

State:
Colorado 80523

© Intel Corporation, 1976
IBM SELECTRIC INPUT PROGRAM

THE PROGRAM ENABLES SELECTRIC TO BE USED AS AN INPUT DEVICE AND OPTIONALLY PRINT ON TELETYPewriter.

INTELLEC 8/MOD 80 SYSTEM, IBM SELECTRIC 735, SELECTRIC INPUT INTERFACE AS PER ATTACHED CIRCUIT DIAGRAM, TELETYPewriter ON PORT 0 AND 1 WITH TAPE PUNCH IF DESIRED.

8080 MONITOR VER. 1.0

SELECTRIC INPUT DATA READ FROM PORTS 20H AND 21H INTO ACCUMULATOR

CONVERTED ASCII CHARACTER IN REGISTER C AND PRINTED/PUNCHED ON TTY.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>A, B, C, H &amp; L</td>
<td>D. M. VAIDYA</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RAM Required:</th>
<th>Company:</th>
</tr>
</thead>
<tbody>
<tr>
<td>NONE</td>
<td>WESTFIELD COLLEGE</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ROM Required:</th>
<th>Address:</th>
</tr>
</thead>
<tbody>
<tr>
<td>328 BYTES</td>
<td>KIDDERPORE AVENUE</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Maximum Subroutine Nesting Level:</th>
<th>City:</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>LONDON, NW3 7ST</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Assembler/Compiler Used:</th>
<th>State:</th>
</tr>
</thead>
<tbody>
<tr>
<td>CROSS ASSEMBLER</td>
<td>ENGLAND</td>
</tr>
</tbody>
</table>
INSERT TAB CHARACTERS FOR SPACES

A tab (CTRL-I) is substituted for each group of 8 spaces or less. Single spaces are left as they are.

Intellie MDS and appropriate input and output devices.

ISIS diskette Operating system

Provide input file and output file when program is called.

ie: SPTAB File 1 to File 2

File 1 and File 2 can be any legal file (.TR:, .Cl:, .F1:.PCM, SRC, etc)

Only single spaces are left in the output file; all other groups of spaces are changed to an appropriate number of TAB characters.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM Required:</td>
<td>John Goode</td>
</tr>
<tr>
<td>64K (less with smaller buffers)</td>
<td>Company: Dow Chemical USA</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>Address: P.O. Drawer K Bldg. B-1219</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td>City: Freeport</td>
</tr>
<tr>
<td>Assembler/Compiler Used:</td>
<td>State: Texas 77541</td>
</tr>
</tbody>
</table>

© Intel Corporation, 1976
PROM PROGRAMMER FOR SDK

Programmes 2708 EPROM with data from RAM/ROM area on SDK board.
Number of programming pulses is set to 5 times needed number of
pulses to change the data in all addresses. Poulis 1ms with one
wait state.
INTEL MCS-80 SDK, ASR-3 TTY
Interface circuit for EPROM, see diagram.

SDK MONITOR PROM

Start address, end address (RAM/ROM area), start address in EPROM.

Instructions: 1) Write data into RAM area (I command) or place
   ROM to be duplicated into normal ROM socket on
   SDK board.
2) Type: G1200 (CR)
   Output: MON PROM SEE PROC ADAS C/R:
3) 26V and PDHN off, place EPROM into socket, PDHN on.
   Type addresses: xxxx, yyyy,zzzz (CR)
   Output: ALL ONE (EPROM erased in area)
   26V ON C/R
4) 26V on , type: (CR)
   Wait for programming min 5 sec max 256 sec
   Output: XX (number (HEX) of used progamming
   pulses)
   MON PROM SEE PROC ADAS C/R:

   cont. next page.

Registers Modified: All
RAM Required: 379 byte
ROM Required: SDK MONITOR PROM
Maximum Subroutine Nesting Level: NA
Assembler/Compiler Used: ISIS II Bo88/Bo85 ASSR+, V1.0

Programmer: Helge Lassen
Company: Soenderborg tekniske skole
Address: Skovvej 26
City: Soenderborg
State: Denmark

© Intel Corporation, 1976

4-464 98 034C

page 1
5) Repeat from step 3 (with new addresses or EPROM) or return to monitor type: not hex char.

CAUTION: It is only allowed to shift EPROM with kit RESET or when program type MON PROM and 26V and DPWR is off.

Error type out:
- NOT ALL ONE: EPROM not erased in area
- ADR OVERFLOW: RAM/ROM area greater than 1K or area not placed correct in EPROM.
- BAD ROM: More than 256 pulses needed for programming or not ok after last programming pulse.
SDK80 TRAP

Inserts a branch to SDK80 monitor from the user program and a return path to the original program without loss of program flow. Allows all monitor facilities to be used to debug a program in RAM or inspect program operation.

SDK80 with monitor routines.
Program uses trap buffer memory 1000 to 102F if only 3 traps are used, leaving locations 1030 to 11FF for user programs. If extra memory is available the trap buffer origin may be shifted by altering locations 1240, 128F and 12A5 in the listing. The number of traps able to be used, up to 16, is determined by available buffer memory, requiring 16 (decimal) locations per trap.

1. To insert a trap in a user program, return to monitor (RESET) and type .G12G0 (CR). Next type IJ,XXXX where I is the trap identifier, J is the range (either 3, 4 or 5) of the skipped code, and XXXX is the address where the trap is to be inserted.

   e.g. for the following program segment:

   LOC     CODE     MNEMONIC
   183D    3E3C     MVIA,3C
   183F    CDCDAB   CALL ABCD

   a) to insert trap number 2 at 183D type 25, 183D
   b) to insert trap number 1 at 183F type 13, 183F

   Note that J must be 3, 4 or 5 and must completely span a program statement (or statements).

2. To remove trap I, enter the trap routine as above and type I (CR). Where I is the trap identifier number to be removed.

3. To return to the user program from the monitor after entering a trap type G (CR) when the program will resume at the left of PC value (as for the SDK monitor) or G XXXX (CR) to resume from location XXXX.

When the trap point in the program is encountered TI is printed where I is the trap identifier, and control is passed to the SDK80 monitor.

Source Listing available for $15.00.
Paper Tape or Diskette Not Offered.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>All</th>
<th>Programmer:</th>
<th>C G Brickell</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM Required:</td>
<td>1K and Buffer</td>
<td>Company:</td>
<td>Fisher &amp; Paykel Ltd</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>SDK Monitor</td>
<td>Address:</td>
<td>Private Bag</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td></td>
<td>City:</td>
<td>Pannure</td>
</tr>
<tr>
<td>Assembler/Compiler Used:</td>
<td></td>
<td>State:</td>
<td>NEW ZEALAND</td>
</tr>
</tbody>
</table>

© Intel Corporation, 1976
TRACE AND REGISTER PRINT OUT.

TO PRINT OUT REGISTERS ON COMMAND OR TO STEP THROUGH A PROGRAM INSTRUCTION BY INSTRUCTION AND TO HAVE REGISTERS PRINTED OUT AFTER EVERY INSTRUCTION EXECUTED. IF TRACE PROGRAM IS ENDED LAST ADDRESS EXECUTED WILL BE DISPLAYED.

TTY --- STANDARD INTEL SETUP.

REGISTER HEADING AND CONTENTS.

Registers Modified:
NONE.

RAM Required:
275H RAM LOCATIONS.

ROM Required:
NONE.

Maximum Subroutine Nesting Level:
NONE.

Assembler/Compiler Used:
INTELDEC MDS 800

Programmer:
R. A. POYNER

Company:
NAVAL OCEAN SYSTEMS CENTER

Address:
PT. LOMA SAN DIEGO, CALIF. 92152
EXEC

To provide character oriented access to ISIS files, and a 'clean' return to ISIS.

MDS with at least 18K RAM, disk system, and console

MDS monitor and ISIS V1.2

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM Required:</td>
<td></td>
</tr>
<tr>
<td>initially to 4600H, then to 39C2</td>
<td></td>
</tr>
<tr>
<td>ROM Required:</td>
<td></td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td></td>
</tr>
<tr>
<td>Assembler/Compiler Used:</td>
<td></td>
</tr>
<tr>
<td>ISIS 8080 MACRO ASSEMBLER V1.1</td>
<td></td>
</tr>
<tr>
<td>Gerry de Koning</td>
<td></td>
</tr>
<tr>
<td>Company:</td>
<td></td>
</tr>
<tr>
<td>Toronto Transit Commission</td>
<td></td>
</tr>
<tr>
<td>Address:</td>
<td></td>
</tr>
<tr>
<td>1900 Yonge Street</td>
<td></td>
</tr>
<tr>
<td>City:</td>
<td></td>
</tr>
<tr>
<td>Toronto, Ontario</td>
<td></td>
</tr>
<tr>
<td>State:</td>
<td></td>
</tr>
<tr>
<td>CANADA M4S 1Z2</td>
<td></td>
</tr>
</tbody>
</table>

© Intel Corporation, 1976

12/6/77 4-470
MDS BACK TO BACK DATA TRANSFER

Enables data to be transmitted between two MDS systems using the high speed punch and reader interfaces

Two MDS systems. One 7474 D flip flop

PL/M 80

The sending MDS system invokes the attached PLM program. The receiving MDS system uses the copy command as follows:

COPY :HR: TO filename

A file is transferred

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL REGISTERS</td>
<td>Bill Holmes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RAM Required:</th>
<th>Company:</th>
</tr>
</thead>
<tbody>
<tr>
<td>8CH</td>
<td>NASA</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ROM Required:</th>
<th>Address:</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADH</td>
<td>Goddard Space Flight Center</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Maximum Subroutine Nesting Level:</th>
<th>City:</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Code 533 Greenbelt</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Assembler/Compiler Used:</th>
<th>State:</th>
</tr>
</thead>
<tbody>
<tr>
<td>PL/M 80</td>
<td>Maryland 20771</td>
</tr>
</tbody>
</table>
>DECLARE ROUTINE ADDRESS INITIAL(0FD5FH);
>DECLARE I BYTE;
>DECLARE NULL BYTE INITIAL(34H);
>DECLARE BUFFER(128) BYTE;
>DECLARE ACTUAL#COUNT ADDRESS;
>DECLARE STATUS ADDRESS;
>DECLARE AFT#IN ADDRESS;
>DECLARE READ#ACCESS LITERALLY '1';

OPEN:
    PROCEDURE(AFT, FILE, ACCESS, MODE, STATUS) EXTERNAL;
    DECLARE (AFT, FILE, ACCESS, MODE, STATUS) ADDRESS;
END OPEN;

CLOSE:
    PROCEDURE(AFT, STATUS) EXTERNAL;
    DECLARE (AFT, STATUS) ADDRESS;
END CLOSE;

READ:
    PROCEDURE(AFT, BUFFER, COUNT, ACTUAL, STATUS) EXTERNAL;
    DECLARE (AFT, BUFFER, COUNT, ACTUAL, STATUS) ADDRESS;
END READ;

EXIT:
    PROCEDURE EXTERNAL;
    DECLARE STATUS ADDRESS;
END EXIT;

CALL READ(1..BUFFER, 128..ACTUAL#COUNT,..STATUS);
CALL OPEN,AFT#IN..BUFFER,READ#ACCESS,0..STATUS);

CALL ROUTINE(NULL);

CALL ROUTINE(NULL);

CALL ROUTINE(NULL);

CALL ROUTINE(NULL);
12 2   END;
13 1   CALL CLOSE(AFT#IN,STATUS);
 4 1   CALL EXIT;
35 1   END DPU;

MODULE INFORMATION:

   CODE AREA SIZE  = 00ADH    173D
   VARIABLE AREA SIZE = 008CH    140D
   MAXIMUM STACK SIZE = 0008H     8D
   56 LINES READ
   0 PROGRAM ERROR(S)

END OF PL/M-80 COMPILATION
FDUMP

FDUMP dumps an ISIS-II diskette file to another ISIS-II file in printable form: Hexadecimal, Octal and ASCII representations are included.

Required Hardware
MDS, MDS-DOS, Console Device

Required Software
MDS Monitor
ISIS-II

Input Parameters
FDUMP is called via a system command:
-FDUMP file to file

Output Results
Error message (if any) to console.
The output file is formatted as 72 byte lines (each followed by <CR> <LF>) containing Hexadecimal, Octal and ASCII interpretations of 8 bytes of the input line.

Programmer:
Garth Eaglesfield

Company:
Micro Focus Ltd.

Address:
18, Vernons Yard

City:
London W.11.

State:
ENGLAND

© Intel Corporation, 1976
12/6/77
ENHANCED MDS TEXT EDITOR, X111

Adds Powerful features to Intel's V1.6 Text Editor

MDS-800 with disk operating system

ISIS-II

Revised 12/78

*Series-II - 220 Users! Please request special macro for use with integrated drive.

Program offered in Object Code on diskette only for $50.00.
The Enhanced Text Editor Maintains all the commands of Intel's current text editor (V1.6) with the addition of many new powerful features. Some of them are:

1) Auto append. Text is automatically read into the text buffer from the specified file.

2) Text rearranging. Blocks of text can be switched around.

3) Macros. Command strings can be assigned to single letter or single control key designations. These macros can be written to a file called EDIT.MAC which is automatically read in when the editor is called.

4) V-Markers. Eight different pointer positions can be designated to delineate the text area for various commands and block moves.

5) Value stack. This gives the editor certain math capabilities so that operations like "counting the number of times a string exists" can be done.
DISKETTE RECOVERY PROGRAM, RECOVERY 1

To permit recovery of files on an ISIS diskette whose directory file has been destroyed, but which is otherwise intact.

Disk based Intellec MDS-800

DISKETTE AVAILABLE IN LOCATED OBJECT CODE ONLY. PAPER TAPE AVAILABLE ON HEX ONLY. LISTING NOT AVAILABLE.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>ALL</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM Required:</td>
<td></td>
<td>Company:</td>
</tr>
<tr>
<td>ROM Required:</td>
<td></td>
<td>Address:</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td></td>
<td>City:</td>
</tr>
<tr>
<td>Assembler/Compiler Used:</td>
<td>PL/M 80 or CROSS</td>
<td>State:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PL/M Compiler</td>
</tr>
</tbody>
</table>
Program Title
PRINT

Function
Lists any ISIS disk file on the line printer up to 255 times. Each copy is paged automatically.

Required Hardware
INTEL MDS-DOS
Centronics 306C line printer
LS1 ADM-3 VDU

Required Software
ISIS Version 1.2
Monitor Version 2.0

Input Parameters
Enter Program name, space, and then file name.

Output Results
Listed File with tabulator characters implemented.

Registers Modified:
All

Programmer:
R. C. Taylor B.Sc.

RAM Required:
As for ISIS

Company:
McMichael Limited

ROM Required:
Monitor

Address:
Wexham Road

Maximum Subroutine Nesting Level:
3

City:
Slough

Assembler/Compiler Used:
ISIS Macro V 1.1

State:
BERKS, SL2 5EL

© Intel Corporation, 1976
2/6/78
Program Title

TYPE

Function

Lists any ISIS disk file on the system console (VDU) and prompts for a return key every page of information.

Required Hardware

INTEL MDS-DOS
Centronics 306C line printer
LS1 ADM-3 VDU

Required Software

ISIS Version 1.2
Monitor Version 2.0

Input Parameters

Enter Program name, space, and then file name.

Output Results

Listed File with tabulator characters implemented.

Registers Modified:

All

Programmer:

R. C. Taylor B.Sc.

RAM Required:

As for ISIS

Company:

McMichael Limited

ROM Required:

Monitor

Address:

Wexham Road

Maximum Subroutine Nesting Level:

3

City:

Slough

Assembler/Compiler Used:

ISIS Macro V 1.1

State:

BERKS, SL2 5EL
JOIN

Merges 2 hex files from separate assemblies into a single hex file for HEXBIN. Transfer address of primary file is retained.

MDS

ISIS

ISIS command syntax:

-JOIN <primary file>, <secondary file>

Secondary file is appended to primary file; transfer address of secondary file is deleted; transfer address of primary file is applied to the total file.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM Required:</td>
<td>Richard Kucia</td>
</tr>
<tr>
<td>From 3200H through 353CH</td>
<td>Company: Realistic Controls</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>Address: 3530 Warrensville Ctr. Rd.</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td>City: Shaker Heights</td>
</tr>
<tr>
<td>Assembler/Compiler Used:</td>
<td>State: Ohio</td>
</tr>
<tr>
<td>ISIS Assembler 1.1</td>
<td></td>
</tr>
</tbody>
</table>
**PRINT PROGRAM FOR G.E. TERMINET - 1200 PRINTER**

Writes a diskette file to :TO:, inserting correct delays for CR LF characters for a G.E. Terminet - 1200 so the printer doesn't overprint short lines.

Intel disk drive, G.E. Terminet-1200 Printer on :TO: port, :TO: port strapped for 1200 Baud

**ISIS-II DOS**

When ISIS command prompt shows on console, press "ON LINE" button on Terminet, enter "PRINT to <filename> CR" on console, where <filename> is an ISIS diskette file.

Prints file on Terminet Printer with correct CR LF delays. If error or when done, displays a message on the console.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>ALL</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM Required:</td>
<td>617 BYTES DECIMAL</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>--</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td>--</td>
</tr>
<tr>
<td>Assembler/Compiler Used:</td>
<td>PL/M Resident Compiler</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Programmer:</th>
<th>John S. Santic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company:</td>
<td>Western Union Data Services</td>
</tr>
<tr>
<td>Address:</td>
<td>70 McKee Drive</td>
</tr>
<tr>
<td>City:</td>
<td>Mahwah</td>
</tr>
<tr>
<td>State:</td>
<td>New Jersey 07430</td>
</tr>
</tbody>
</table>
PRINT OUT SOURCE FILE OF FLOPPY DISK

The function of this program is to print out source files stored on the floppy disk to the designated list device. This program recognizes the tab character. The tab in this program is set at every 8 spaces.

Required Hardware
MDS 800
Console
Floppy Disk

Required Software
ISIS-II, V2.2
Monitor Program

Input Parameters
Go to monitor and assign the list device to console.
Go to ISIS II, V2.2
Enter P followed by a space and the filename to be printed.

Output Results
The program will read the designated file and output this file to the console using the monitor output routine. The tab character will be recognized by this program and set at every 8 spaces.

Registers Modified:

ALL

RAM Required:
Ø172H Bytes

ROM Required:

Maximum Subroutine Nesting Level:

Assembler/Compiler Used:

INTELHC MDS MACRO ASSEMBLER

Programmer:
Bill Uhlarik

Company:
ITT Barton

Address:
900 So. Turbull Cyn Rd.

City:
City of Industry

State:
California

© Intel Corporation, 1976
LOC OBJ  SEQ SOURCE STATEMENT

0 ;REF. NO. AB110
1 PROGRAM TITLE P

2 3 4 5 6 7 8 9

10 -------------------------------
11 THE FOLLOWING PROGRAM PRINTS OUT A SPECIFIED FILE ON THE CONSOLE
12 DEVICE AND IS INVOKED BY THE COMAND
13
14 P FILENAME
15
16 BY BILL UHLARIK SEPT 21,1977
17
18
19

0000
0001 0002 0003 0004 0005 0006 0007 0008 0009 000A 000B 000C 000D 000E 000F 0010 0011 0012 0013 0014 0015 0016 0017 0018 0019 001A 001B 001C 001D 001E 001F 0020 0021 0022 0023 0024 0025 0026 0027 0028 0029 002A 002B 002C 002D 002E 002F

21 OPEN EQU 0
22 CLOSE EQU 1
23 READ EQU 3
24 EXIT EQU 9
25 ERROR EQU 12
26 ENDL EQU 0AH ;LINE FEED CHARACTER
27 ICOUNT EQU 08H ;TAB COUNT = 8 SPACES
28 TAB EQU 09H ;TAB CHARACTER
29 LO EQU 0F80FH ;ADDRESS OF OUTPUT ROUTINE IN MONITOR PGM

30 ■
31 EXTRN ISIS
32 ■
33 ■
34 BEGIN: CALL ADV ;CLEAR :LP: BUFFER
35 ■
36 ■
37 ■
38 ■
39 ■
40 ■
41 ■
42 ■
43 ■
44 ■
45 ■
46 ■
47 ■
48 ■
49 ■
50 ■
51 ■
52 ■
53 ■
54 ■
55 ■
56 ■
57 ■
58 ■

2/6/78 4-498
<table>
<thead>
<tr>
<th>Line</th>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A03B</td>
<td>B7</td>
<td>59 ORA A</td>
</tr>
<tr>
<td>A03C</td>
<td>C2COA0</td>
<td>60 JNZ ERR</td>
</tr>
<tr>
<td>A03D</td>
<td>2AE8A0</td>
<td>61 LHLD ACTUAL</td>
</tr>
<tr>
<td>A042</td>
<td>7C</td>
<td>62 MOV A,H</td>
</tr>
<tr>
<td>A043</td>
<td>B5</td>
<td>63 ORA L</td>
</tr>
<tr>
<td>A044</td>
<td>CAADA0</td>
<td>64 JZ DONE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>65</td>
</tr>
<tr>
<td>A047</td>
<td>21F1A0</td>
<td>66 LXI H,BUFFER</td>
</tr>
<tr>
<td>A04A</td>
<td>2271A1</td>
<td>67 SHLD BP</td>
</tr>
<tr>
<td>A04D</td>
<td>2AEBA0</td>
<td>68 LHLD ACTUAL</td>
</tr>
<tr>
<td>A050</td>
<td>2B</td>
<td>69 DCX H</td>
</tr>
<tr>
<td>A051</td>
<td>22EBA0</td>
<td>70 SHLD ACTUAL</td>
</tr>
<tr>
<td>A054</td>
<td>2A71A1</td>
<td>72 PRNT: LHLD BP</td>
</tr>
<tr>
<td>A057</td>
<td>7E</td>
<td>73 MOV A,M</td>
</tr>
<tr>
<td>A05B</td>
<td>FE09</td>
<td>74 CP I TAB</td>
</tr>
<tr>
<td>A05A</td>
<td>CA3BA0</td>
<td>75 JZ SPACE</td>
</tr>
<tr>
<td>A05D</td>
<td>FE0A</td>
<td>76 CP I ENBL</td>
</tr>
<tr>
<td>A05F</td>
<td>CA9A0</td>
<td>77 JZ RESET1</td>
</tr>
<tr>
<td>A062</td>
<td>4E</td>
<td>78 CONT1: MOV C,M</td>
</tr>
<tr>
<td>A063</td>
<td>CD0FFB</td>
<td>79 CALL LO</td>
</tr>
<tr>
<td>A066</td>
<td>3F0A0</td>
<td>80 LDA COUNT</td>
</tr>
<tr>
<td>A069</td>
<td>3D</td>
<td>81 DCR A</td>
</tr>
<tr>
<td>A06A</td>
<td>32F0A0</td>
<td>82 STA COUNT</td>
</tr>
<tr>
<td>A06D</td>
<td>CA92A0</td>
<td>83 JZ RESET</td>
</tr>
<tr>
<td>A070</td>
<td>23</td>
<td>84 CONT: INX H</td>
</tr>
<tr>
<td>A071</td>
<td>2271A1</td>
<td>85 SHLD BP</td>
</tr>
<tr>
<td>A074</td>
<td>2AEBA0</td>
<td>86 LHLD ACTUAL</td>
</tr>
<tr>
<td>A077</td>
<td>7C</td>
<td>87 MOV A,H</td>
</tr>
<tr>
<td>A07B</td>
<td>B5</td>
<td>88 ORA L</td>
</tr>
<tr>
<td>A079</td>
<td>CA3BA0</td>
<td>89 JZ LOOP</td>
</tr>
<tr>
<td>A07C</td>
<td>2B</td>
<td>90 DCX H</td>
</tr>
<tr>
<td>A07D</td>
<td>22EBA0</td>
<td>91 SHLD ACTUAL</td>
</tr>
<tr>
<td>A060</td>
<td>C354A0</td>
<td>92 JMP PRINT</td>
</tr>
<tr>
<td>A06G</td>
<td>OE28</td>
<td>93 SPACE: MVI C.20H</td>
</tr>
<tr>
<td>A085</td>
<td>CD0FFB</td>
<td>94 CALL LO</td>
</tr>
<tr>
<td>A088</td>
<td>3F0A0</td>
<td>95 LDA COUNT</td>
</tr>
<tr>
<td>A08B</td>
<td>3D</td>
<td>96 DCR A</td>
</tr>
<tr>
<td>A08C</td>
<td>32F0A0</td>
<td>97 STA COUNT</td>
</tr>
<tr>
<td>A08F</td>
<td>C254A0</td>
<td>98 JNZ PRINT</td>
</tr>
<tr>
<td>A092</td>
<td>3E0B</td>
<td>99 RESET: MVI A,ICOUNT</td>
</tr>
<tr>
<td>A094</td>
<td>3F0A0</td>
<td>100 STA COUNT</td>
</tr>
<tr>
<td>A097</td>
<td>C370A0</td>
<td>101 JMP CONT</td>
</tr>
<tr>
<td>A09A</td>
<td>3E09</td>
<td>102 RESET1: MVI A,ICOUNT+1</td>
</tr>
<tr>
<td>A09C</td>
<td>3F0A0</td>
<td>103 STA COUNT</td>
</tr>
<tr>
<td>A09F</td>
<td>C362A0</td>
<td>104 JMP CONT</td>
</tr>
<tr>
<td>A0A2</td>
<td>OEO0</td>
<td>105 ; ADV: MVI C.0DH</td>
</tr>
<tr>
<td>A0A4</td>
<td>CD0FFB</td>
<td>106 CALL LO</td>
</tr>
<tr>
<td>A0A7</td>
<td>OEOA</td>
<td>107 CALL LO</td>
</tr>
<tr>
<td>A0A9</td>
<td>CD0FFB</td>
<td>108 CALL LO</td>
</tr>
<tr>
<td>A0AC</td>
<td>C9</td>
<td>109 RET</td>
</tr>
<tr>
<td>A0AD</td>
<td>CD2AA0</td>
<td>111 ; DONE: CALL ADV</td>
</tr>
<tr>
<td>A0AG</td>
<td>OEO1</td>
<td>112 CALL C,CLOSE</td>
</tr>
<tr>
<td>A0BB</td>
<td>11DA0</td>
<td>113 MVI L,XI D,CBLK</td>
</tr>
<tr>
<td>A0BF</td>
<td>CD0000</td>
<td>114 CALL ISIS</td>
</tr>
<tr>
<td>A0BB</td>
<td>0E09</td>
<td>115 MVI C,EXIT</td>
</tr>
<tr>
<td>A0BB</td>
<td>11EA0</td>
<td>116 MVI L,XI D,XBLK</td>
</tr>
<tr>
<td>A0BD</td>
<td>CD6000</td>
<td>117 CALL ISIS</td>
</tr>
</tbody>
</table>

; START OF PRINT
; INITIALIZE MEMORY POINTER
; INITIALIZE BUFFER COUNTER FOR PRINT LOOP
; ACTUAL = ACTUAL - 1
; CHECK FOR TAB
; CHECK FOR END OF LINE
; OUTPUT CHARACTER
; UPDATE TAB COUNT
; UPDATE BUFFER POINTER
; CHECK FOR END OF BUFFER
; UPDATE BUFFER COUNTER
; ACTUAL = ACTUAL - 1
; OUTPUT SPACE
; CHECK IF TAB IS REACHED
; RE-INITIALIZE TAB COUNT
; RE-INITIALIZE TAB COUNT
; CARRIAGE RETURN & LINE FEED ROUTINE
; CLOSE THE INPUT FILE
; NORMAL EXIT
LOC   OBJ   SEQ   SOURCE STATEMENT

A0C0  0E8C  120   ERR:   MVI   C,ERROR
A0C2  1IEA0  121   LXI   D,EBLK
A0C5  CD0000  122   CALL   ISIS
A0CB  0E99  123   MVI   C,EXIT
A0CA  1IEA0  124   LXI   D,XBLK
A0CD  CD0000  125   CALL   ISIS

126  ;
A0D0  0EA0  127   OBLK:   DW   AFT
A0D2  F1A0  128   DW   BUFFER
A0D4  0100  129   DW   1
A0D6  0000  130   DW   0
A0D8  ECA0  131   DW   STATUS

132  ;
A0DA  134   CAFT:   DS   2
A0DC  ECA0  135   DW   STATUS

136  ;
A0DE  0100  138   AFT:   DW   1
A0E0  F1A0  139   DW   BUFFER
A0E2  0000  140   DW   128
A0E4  EBA0  141   DW   ACTUAL
A0E6  ECA0  142   DW   STATUS

143  ;
A0EB  144   ACTUAL:   DS   2
145  ;
A0FA  ECA0  146   XBLK:   DW   STATUS

147  ;
A0EC  148   EBLK:
A0EE  ECA0  149   STATUS:   DS   2
150   DW   STATUS

151  ;
A0F0  152   COUNT:   DS   1
A0F1  153   BUFFER:   DS   128

154  ;
A171  156   BP:   DS   2
157  ;
A000  158   END   BEGIN

PUBLIC SYMBOLS

EXTERNAL SYMBOLS
ISIS  E  0000

USER SYMBOLS
ACTUAL A A0E5  ADV A A0A2  AFT A A0DE  BEGIN A A000  BP A A171  BUFFER A A0F1
CAFT A A0DA
CBLK A A0DA
CLOSE A 0001  CONT A A070  CONT1 A A062  COUNT A A0F0  DONE A A0AD
EBLK A A0EC
EMBL A 000A  ERR A A0C0  ERROR A 000C  EXIT A 0009  ICOUNT A 0008  ISIS E 0000
END A 0009  LOOP A A030  OBLK A A0D0  OPEN A 0000  PRNT A A054  RBLK A A0DE  READ A 0003
RESET A A092  SPACE A A083  STATUS A A0EC  TAB A 0009  XBLK A A0E9
RESET1 A A09A

ASSEMBLY COMPLETE, NO ERROR(S)
ONLINE, UPLOAD, DOWNLOAD

To use the Intellec MDS-800 as a terminal or to transfer files to/from any timesharing system. (Specifically PDP-10)

A Serial Port
(Attached documentation describes addition of an RS232 Port)

User must write a driver on the host computer to communicate with upload and download.
User must also modify programs to compile on the compiler chosen and to run on the user's system configuration.

NOTE: ONLINE is available in Assembly Language only, UPLOAD and DOWNLOAD are available in PL/M only.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RAM Required:</th>
<th>Company:</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>ROM Required:</th>
<th>Address:</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Maximum Subroutine Nesting Level:</th>
<th>City:</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Assembler/Compiler Used:</th>
<th>State:</th>
</tr>
</thead>
</table>

See Note
ONLINE, UPLOAD, DNLD

To use the MDS as a terminal, or to transfer files to/from the PDP-10, the user needs three MDS programs: ONLINE, UPLOAD, DNLD. The user may need to make changes to the hardware configuration as specified in the attached spec. Once the hardware changes are made, the general algorithm is:

. Use program ONLINE to call up the PDP-10.
. Hit carriage return twice (should get period'.' prompt)
. Log on to the PDP-10 (or your host computer)
. When ready to transfer files, hit the 'break' key, which will return you to ISIS.
. Type what needs done:
   UPLOAD MDS-filename to PDP-filename
   DNLD PDP-filename to MDS-filename
. To get back to the PDP-10, re-type ONLINE (no need to log in again)
. Be sure to log off.
MDS Modifications for PDP-10 Communication

When choosing a serial port, there are two alternatives on the Intellec MDS. The first you may use the teletype port, which is a current loop serial line. Otherwise, a second monitor board may be added to the system, in parallel to the original monitor board, giving you an extra RS232 port.

Modifications are made to this extra monitor board so that its CRT port can be used as the PDP-10 port:

First, the base address of the "CRT" port on the new monitor board must be changed from F00 to D00. This entails cutting the trace to pin 15 of chip A34, and jumping pin 14 of the same chip to the feed-through adjacent to pin 16 (which is where pin 15 used to be connected).

Second, the new monitor board needs a special harness which connects to the CRT connector (PDP-10 line). There are four wires in the harness and one jumper.

On the board connector, 2 wires from pin A15 go to cable connector pin 7. One wire from board connector pin 14 to cable connector pin 3. One wire from board connector pin 15 to cable connector pin 2. One jumper on board connector from pin A4 to A10.

Software
Be sure your software contains the correct port numbers for the serial line you chose to use.

Baud Rate Changes
If a baud rate of 300 is desired for the following on the second monitor card: Change jumper from 2400 baud (standard 19 to 20) to 300 baud (13 to 14).
STEP

Single-steps an assembly language program and halts in the monitor after each instruction. The registers may then be examined or changed, or the program continued. The STEP program can be used in conjunction with the Monitor's "GO" command to execute subroutines, then continue the single-step mode. May be resident completely in ROM.

8080 CPU console

Monitor version 3.0 (can be changed)

User program via the Monitor "GO" command using at least 1 breakpoint. After executing the breakpoint, the program may be single stepped in one of two ways (assuming STEP is in ROM):

1) Place a JMP to STEP in an unused restart location, and interrupt the CPU with the console switches set to the proper value.
2) Modify the monitor to have another command (may be "O") and type that letter.

The STEP program will execute the next instruction and then print the new PC, all registers are saved. This program will not single step in ROM, as it uses the restart logic.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM Required:</td>
<td>Monitor exit template</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>246</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Assembler/Compiler Used:</th>
<th>Macro V.4.0</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Programmer:</th>
<th>Gary Saxer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company:</td>
<td>ATE ASSOCIATES, INC.</td>
</tr>
<tr>
<td>Address:</td>
<td>8448 Reseda Blvd., #201 Northridge, CA 91324</td>
</tr>
</tbody>
</table>
8080 V3.0

D100, 10F (Little program from 100-10A)
0100 21 00 02 00 3E 0F 23 3C 06 01 00 00 00 26
'S10 00-C3 00- 00-35 (Place JMP to STEP at RST 2 location, and set console switches to D7H
'G100,100 (Start Program)
*0100 (Starts at beginning)
.X (Register contents)
A=00 B=00 C=00 D=00 E=00 F=02 H=00 L=00 M=0000 P=0100 S=0100
.*0103 Interrupt switch depressed, causes RST 2 to be executed
.X
A=00 B=00 C=00 D=00 E=00 F=02 H=02 L=00 M=0200 P=0103 S=0100
.*0104
.X
A=00 B=00 C=00 D=00 E=00 F=02 H=02 L=00 M=0200 P=0104 S=0100
.*0105
.X
A=00 B=00 C=00 D=00 E=00 F=02 H=02 L=00 M=0200 P=0105 S=0100
.*0106
.X
A=00 B=00 C=00 D=00 E=00 F=02 H=02 L=00 M=0200 P=0106 S=0100
.*0107
.X
A=00 B=00 C=00 D=00 E=00 F=02 H=02 L=00 M=0200 P=0107 S=0100
.*0108
.X
A=10 B=00 C=00 D=00 E=00 F=12 H=02 L=01 M=0201 P=0108 S=0100
.*0109
.X
A=10 B=00 C=00 D=00 E=00 F=12 H=02 L=01 M=0201 P=0109 S=0100
.*0110
.X
A=10 B=00 C=00 D=00 E=00 F=12 H=02 L=02 M=0202 P=0110 S=0100
.*0111
.X
A=11 B=00 C=00 D=00 E=00 F=06 H=02 L=02 M=0202 P=0111 S=0100
.*0112
.G,106 "GO" command break points at 108
.*0113
.X
A=12 B=00 C=00 D=00 E=00 F=06 H=02 L=03 M=0203 P=0112 S=0100
.*0114
.RST 2 "Step past 108"
.*0115
.G,108 Then do instructions and STOP at 108 again
.*0116
.X
A=13 B=00 C=00 D=00 E=00 F=02 H=02 L=04 M=0204 P=0116 S=0100

2/6/78
0039 C3000
  JMP INC RF ;RESUME PROG
  ; OP CODES <= 03E

003C CD9300  INCRB: CALL SI XE ;8B IMMEDIATE? (YES=NO RETURN)
003F FE01  CPI 01H ;16B IMM ED?
0041 CA2100  JZ THREE ;YES
0044 7A  MOV A,D ;GET OP CODE

0045 FE20  CPI 02H ;< LXI H?
0047 DA2D00  JC INCRE ;YES E=1
004A E607  ANI 07H ;CLEAR UPPER
004C FE02  CPI 02H ;ADR?
004E C2D00  JNZ INCRE ;NO E=1
0051 1C THREE: INR E
0052 C3000  JMP INCRD ;E=3
  ; OP CODES > C1

0055 CD9300  INCR C: CALL SI XE ;8B IMM E D?
0058 FE08  CPI 00H ;RET?
005A CAB400  JZ TRP RT ;YES
005D B7  ORA A ;RET?
005E CAB400  JZ TRPR T ;YES
0061 IF  RAR ;EVEN OP CODE?
0062 DA7100  JC IOST? ;NO
0065 CDE700  CALL LD B1 ;GET LOC TO TRAP
0066 CDA200  CALL TRAPA ;TRAP (BC)
006B CDF000  CALL FIX
006E C35100  JMP THREE ;TRAP NEXT LOC
0071 7A IOST?: MOV A,D ;CHECK EXCEPTIONS
0072 FE03  CPI 0D3H ;OUT?
0074 CA2C00  JZ INCRD ;YES E=2
0077 FED8  CPI 0DBH ;IN?
0079 CA2C00  JZ INCRD ;YES
007C FE09  CPI 039H ;REAL RET? (UNCOND)
007E CAB400  JZ TRP RT ;YES
0081 FEE9  CPI 0F9H ;PCHL?
0083 C2D00  JNZ INCRE ;NO E=1

0086 211100 LXI H,0011H ;YES GET HL INTO BC
0089 39  DAD SP ;POINT TO HL LOC
008A 4E  MOV C.N
008B 23  INX H
008C 46  MOV B,N
008D CDA200  CALL TRAPA
0090 C3000  JMP INCRF ;OUT OF HERE
  ; CHECK FOR IMMEDIATE INST

0093 57  SI XE: MOV D,A ;SAVE OP CODE
0094 E607  ANI 07H ;CLEAR UPPER 5
0096 FE06   CP1 06H   ; NOT 8B INMED
009B C9   RNZ   ; IGNORE OUR CALL
009E E1   SIXXT: POP H   ; RET ADDR IN HL
00A0 C32C00   JMP   INCRD   ; E=2
00A3   TRAP:   INX B   ; BC+E
00A5 1D   DCR E   ; NO TRAP AT ZERO
00A6 CACB00   JZ   STOP   ; SAVE TRAP LOC
00A9 71   MOV H.C   ; SAVE ITS
00AB 23   INX H   ; GET SP POINTER
00AD 0A   LDA X B   ; OLD SP LOC
00B4 210900   TRAPRT:   LAX B,0009H   ; OLD SP LOC
00B7 39   DAD SP   ; CALL TRAP
00B8 56   MOV D,M   ; CALL TRAP
00B9 2B   DCX H   ; CALL TRAP
00BB EB   XCHG   ; CALL TRAP
00BE DED0C00   CALL   LDB12   ; CALL TRAP
00CF 0600   CALL FIX   ; CALL TRAP
00C5 C32000   JMP   INCRE   ; AFTER RET INST
00CB 21D000   STOP:   LXI H,MSG
00C8 060E   MVI B,000H
00CD 4E   MOV C,M
00CE C633B   CALL CO
00D1 05   DCR B
00D2 23   INX H
00D3 C2CD00   JNZ S=6
00D5 C363B   JMP START
00DE 54524150   MSG:   DB 'TRAP TO ZERO',0DH,0AH

2/6/78
ISIS 8080 MACRO ASSEMBLER, V1.1

00DD 20544F20
00E1 5A45524F
00E5 000A

; TRAP A LOC POINTED TO IN MEMORY

LDB1: PUSH B ; PUT BC IN HL
POP H
INX H ; POINT PAST OP CODE

LDB2: PUSH B ; SAVE BC IN DE
POP D
MOV C, M ; GET NEW BC
INX H
MOV B, M ; TRAP NEXT (TLOC ON TOP CANT CALL HERE)

; RESTORE LDB1 SAVED STUFF

FIX: XTHL ; GET RET ADDR
PUSH H ; PUT IT BACK (TLOC PUT BACK TOO)

PUSH D
POP B ; RESTORE BC

PUSH D
MV I E, I ; RESET E

RET

END
Program Test-Loader

Load/Dump/Test ROMSIM (8048 Support Software)

Intel ROM Simulator, Intellec MDS DOS System

See attached documentation

Output Results

See attached documentation

Available on diskette only.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>N/A</th>
<th>Programmer:</th>
<th>John Kovach</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM Required:</td>
<td>N/A</td>
<td>Company:</td>
<td>Magnavox</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>N/A</td>
<td>Address:</td>
<td>1700 Magnavox Way</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td>N/A</td>
<td>City:</td>
<td>Fort Wayne</td>
</tr>
<tr>
<td>Assembler/Compiler Used:</td>
<td>ISIS I ASM 80 V1.1</td>
<td>State:</td>
<td>Indiana</td>
</tr>
</tbody>
</table>

* Intel Corporation, 1976

12/78
1.0 INTRODUCTION

The Program Test Loader is an Intel 8080 program designed to test, load, and
dump the Intel ROM Simulator from the Intellec MDS system operating under
either ISIS-I or ISIS-II. It was written to provide the 8048 developer with
a means to load and dump 8048 object files to and from locations D000H to
DFFFH of the ROM SIMULATOR. It also provides a programmable feature that
tests the RAM located in the ROM SIM.

2.1 INVOKING THE PROGRAM

The Program Test-Loader is invoked from either ISIS-I or ISIS-II by typing
PTL and a carriage return in response to the system dash prompt (-). The
program must exist in binary format on diskette before it can be invoked.

2.2 OPTIONS

The Program Test Loader provides the user with the options tabulated in
Table 2A. Each option is invoked by its name or by the short hand notation
given for that option (in parenthesis). The format for specifying options
is as follows: OPTION, SPACE, LIST, CARRIAGE RETURN.

OPTIONS may be entered after the program has given the question mark prompt ("?").
Each time an option has been completed, the program returns a message to
verify the option performed and then issues the question mark prompt ("?")
to indicate it is ready to receive another option request. Table 2-B tabulates
the responses given to each option request.
Symbol Table Program for 8080/8085 *Version 1.2

To print a symbol table of a previously located program sorted alphanumerically or by address, and printed in columns. Publics, Local Symbols, and PL/M Line numbers are included.

Intellec MDS with disk drive

ISIS-I1

See reverse page 4-516

*Revised 12/78
- 20% faster
- now accepts more than 255 symbols per module.

Available in object code on diskette or HEX code on paper tape.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>All</th>
<th>Programmer: Gary Carleton</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM Required:</td>
<td>32K</td>
<td>Company: Intel Corp.</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>--</td>
<td>Address:</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td>--</td>
<td>City:</td>
</tr>
<tr>
<td>Assembler/Compiler Used:</td>
<td>PL/M-80 V3.0</td>
<td>State:</td>
</tr>
</tbody>
</table>
The program is invoked by entering:

```
SYMBOL FILE1 [TO FILE2] [PAGEWIDTH(n)] [ADDR] [NOTRUNC]
```

(Brackets indicate optional controls)

**FILE1:** Previously located object file program.

**FILE2:** ISIS file where symbol table is to be written. Can be :LP:, :CO:, a disk file, etc.

**PAGEWIDTH(n):** Number of characters per line similar to PL/M and assembler controls. n must be less than 200.

**ADDR:** Specifies sort by address instead of sort by name.

**NOTRUNC:** Symbol names are normally truncated to the first ten characters. NOTRUNC causes no truncation.

**DEFAULT VALUES**

These are the values used when the optional controls are not specified:

**FILE2** - FILE1.SYM

**PAGEWIDTH(n)** - n=80

**ADDR** - Sort by symbol name (or line number)

**NOTRUNC** - Truncate at tenth character

As with the LOCATE symbol table, the original assemblies or compilations must have used the DEBUG control for the symbol table to be included in the object module. The normal sequence of program execution when using SYMBOL is:

1. Assemblies or Compilations (with DEBUG)
2. LINK (If necessary)
3. LOCATE (without PURGE)
4. SYMBOL

If you use the PURGE control, the following step should be added:

5. LOCATE (with PURGE)

For convenience, steps 3, 4, and 5 may be included in a submit file.

**Notes:**

The sort algorithm used is not the most efficient. Because of this, large modules may cause the intellect to pause for a minute or more.

Also, SYMBOL should be on a system diskette for ISIS to properly regain control after execution.
<table>
<thead>
<tr>
<th>PUBLICS:</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3B5FH</td>
<td>@P0014</td>
<td>3B40H</td>
<td>@P0015</td>
</tr>
<tr>
<td>3B44H</td>
<td>@P0017</td>
<td>3B4CH</td>
<td>@P0029</td>
</tr>
<tr>
<td>3B6BH</td>
<td>@P0031</td>
<td>3B6EH</td>
<td>@P0032</td>
</tr>
<tr>
<td>3B76H</td>
<td>@P0034</td>
<td>3B78H</td>
<td>@P0035</td>
</tr>
<tr>
<td>3B8DH</td>
<td>@P0049</td>
<td>3B94H</td>
<td>@P0096</td>
</tr>
<tr>
<td>3B9EH</td>
<td>@P0101</td>
<td>3BA1H</td>
<td>@P0102</td>
</tr>
<tr>
<td>3872H EINTOBBCD</td>
<td></td>
<td>368BH</td>
<td>BLANKS</td>
</tr>
<tr>
<td>3BA9H ERROR</td>
<td></td>
<td>3BBEH</td>
<td>EXIT</td>
</tr>
<tr>
<td>37EDH HEAXADDRTA</td>
<td></td>
<td>3C36H</td>
<td>INAFTN</td>
</tr>
<tr>
<td>37EH ISISSRR</td>
<td></td>
<td>3AD2H</td>
<td>LINECONV</td>
</tr>
<tr>
<td>3C38H OUTAFTN</td>
<td></td>
<td>3C3DH</td>
<td>OUTFILE</td>
</tr>
<tr>
<td>3BCDH READ</td>
<td></td>
<td>398FH</td>
<td>READREC</td>
</tr>
<tr>
<td>3BF1H WRITE</td>
<td></td>
<td>39BFH</td>
<td>WRITEREC</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MODULE: MAINPROGRAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYMBOLS:</td>
</tr>
<tr>
<td>3C36H INAFTN</td>
</tr>
<tr>
<td>3C36H OUTAFTN</td>
</tr>
<tr>
<td>3C3EH SORTBYADDR</td>
</tr>
</tbody>
</table>

| LINE NUMBERS:                  |
| 3680H 7                        | 3686H 8         |
| 3689H 9                         |              |

<table>
<thead>
<tr>
<th>MODULE: UTILS</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYMBOLS:</td>
</tr>
<tr>
<td>3C57H ASCIIINDEX</td>
</tr>
<tr>
<td>3C78H BCDPTR</td>
</tr>
<tr>
<td>3872H EINTOBBCD</td>
</tr>
<tr>
<td>3A10H CHAREQL</td>
</tr>
<tr>
<td>3753H ERRMSG1</td>
</tr>
<tr>
<td>3901H HEADING</td>
</tr>
<tr>
<td>3C56H HEXINDEX</td>
</tr>
<tr>
<td>3C60H I</td>
</tr>
<tr>
<td>3C84H INDEX</td>
</tr>
<tr>
<td>3C76H LEN</td>
</tr>
<tr>
<td>3C9DH MEMORY</td>
</tr>
<tr>
<td>3C69H READCNT</td>
</tr>
<tr>
<td>3C5EH REMAINDER</td>
</tr>
<tr>
<td>3C61H STATUS</td>
</tr>
<tr>
<td>3C72H STRG1PTR</td>
</tr>
<tr>
<td>39BFH WRITEREC</td>
</tr>
</tbody>
</table>

| LINE NUMBERS: |
| 3789H 17       | 378FH 19       |
| 37A6H 21       | 37B9H 22       |
| 37EDH 24       | 37C7H 26       |

2/6/78
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>37D6h</td>
<td>28</td>
</tr>
<tr>
<td>3604h</td>
<td>31</td>
</tr>
<tr>
<td>3826h</td>
<td>34</td>
</tr>
<tr>
<td>385Ch</td>
<td>37</td>
</tr>
<tr>
<td>3871h</td>
<td>40</td>
</tr>
<tr>
<td>368Ch</td>
<td>44</td>
</tr>
<tr>
<td>38b4h</td>
<td>47</td>
</tr>
<tr>
<td>38DAh</td>
<td>50</td>
</tr>
<tr>
<td>3901h</td>
<td>53</td>
</tr>
<tr>
<td>3916h</td>
<td>57</td>
</tr>
<tr>
<td>393 Eh</td>
<td>60</td>
</tr>
<tr>
<td>395Ch</td>
<td>63</td>
</tr>
<tr>
<td>3981h</td>
<td>66</td>
</tr>
<tr>
<td>398Fh</td>
<td>69</td>
</tr>
<tr>
<td>39E6h</td>
<td>73</td>
</tr>
<tr>
<td>39CEh</td>
<td>77</td>
</tr>
<tr>
<td>39E0h</td>
<td>81</td>
</tr>
<tr>
<td>39FBh</td>
<td>84</td>
</tr>
<tr>
<td>3A0h</td>
<td>87</td>
</tr>
<tr>
<td>3A4Fh</td>
<td>91</td>
</tr>
<tr>
<td>3A60h</td>
<td>94</td>
</tr>
<tr>
<td>3A61h</td>
<td>97</td>
</tr>
<tr>
<td>3A93h</td>
<td>101</td>
</tr>
<tr>
<td>3AA7h</td>
<td>104</td>
</tr>
<tr>
<td>3AD1h</td>
<td>107</td>
</tr>
<tr>
<td>3AEPFh</td>
<td>111</td>
</tr>
<tr>
<td>3B08h</td>
<td>114</td>
</tr>
<tr>
<td>3B34h</td>
<td>117</td>
</tr>
</tbody>
</table>

**Starting Address:** 3680h

2/6/78
8048 - Seven Segment Display Interface Subroutines -- SCAN
Collection of utility subroutines which may be used with the 8048 family to scan a keyboard matrix, debounce and encode key depressions, and drive a multiplexed seven segment display. Also included are utilities to translate Hexidecimal data into segment patterns, right or left entry to the display registers, and clearing or writing character sequences to the display.

Simple X-Y matrix of up to 64 switches, seven-segment display up to eight digits, and high current segment and digit drivers. Matrix & display size may be increased arbitrarily with the addition of external decoders.

User-written background program (game, calculator, telephone-dialer, etc.) requiring low-cost keyboard and display capability.

Determined by user's program and utilities selected.

Live keyboard, continuous flicker-free multi-character display.

Program offered on diskette only.

Registers Modified: Pointers & one in Bank0 and four in Bank1

| RAM Required: | 12 bytes |
| ROM Required: | 25010 bytes |
| Maximum Subroutine Nesting Level: | 3 |
| Assembler/Compiler Used: | ASM48 V2.0 |

Programmer:
J. Wharton

Company:

Address:

City:

State:
AP29 "USING THE 8085 SERIAL I/O LINES"

Two software packages using the 8085 SID and SOD pins for serial I/O. The first set of subroutines may be used to interface an 8085 to a CRT at a wide range of baud rates, with automatic rate identification. The second set includes two low-level utilities for recording and reloading bytes of data using an inexpensive audio cassette recorder and simple interface.

This code was used in the appendix of Intel Application Note AP-29, "USING THE INTEL 8085 SERIAL I/O LINES". The software and hardware required is fully described in the note.

The Application note AP-29 will be included with the program.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>J. Wharton</td>
</tr>
<tr>
<td>RAM Required:</td>
<td>Company:</td>
</tr>
<tr>
<td>4 Bytes &amp; Stack</td>
<td></td>
</tr>
<tr>
<td>ROM Required:</td>
<td>Address:</td>
</tr>
<tr>
<td>326 Bytes</td>
<td></td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level: 3</td>
<td>City:</td>
</tr>
<tr>
<td>Assembler/Compiler Used:</td>
<td>State:</td>
</tr>
<tr>
<td>ASM80, V1.0</td>
<td></td>
</tr>
</tbody>
</table>

© Intel Corporation, 1976
4/78

98-0340
4-522
8048 TUNE GENERATOR

Generates fixed sequence of tones and spaces (i.e. Music)

See attached diagram.

None

This program will run repetitively through the sequence of notes and spaces starting address TAB, ending when OFFH character is encountered. The pitch and tempo will vary with basic clock frequency but can be altered in the program (this program was run using ALE = 5 m.sec.)

Programmed tune

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>R0, R1, R2, R3 + Flags F0 + F1</td>
<td>T. Harvey</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RAM Required:</th>
<th>Company:</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 Bytes for registers</td>
<td>Avery-Hardoll Ltd.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ROM Required:</th>
<th>Address:</th>
</tr>
</thead>
<tbody>
<tr>
<td>57 + Tune</td>
<td>Downley Road</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Maximum Subroutine Nesting Level:</th>
<th>City:</th>
</tr>
</thead>
<tbody>
<tr>
<td>--</td>
<td>HAVANT, Hants.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Assembler/Compiler Used:</th>
<th>State:</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISIS-II 8048 Assembler V1.2</td>
<td>ENGLAND</td>
</tr>
<tr>
<td>LOC</td>
<td>OBJ</td>
</tr>
<tr>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>0010</td>
<td>1414</td>
</tr>
<tr>
<td>0012</td>
<td>0410</td>
</tr>
<tr>
<td>0014 234F</td>
<td>A8</td>
</tr>
<tr>
<td>0016</td>
<td>A3</td>
</tr>
<tr>
<td>0017</td>
<td>A9</td>
</tr>
<tr>
<td>0018</td>
<td>62</td>
</tr>
<tr>
<td>0019</td>
<td>27</td>
</tr>
<tr>
<td>001A</td>
<td>BB16</td>
</tr>
<tr>
<td>001E</td>
<td>45</td>
</tr>
<tr>
<td>001F</td>
<td>042D</td>
</tr>
<tr>
<td>0021 7625</td>
<td>30</td>
</tr>
<tr>
<td>0023 163E</td>
<td>31</td>
</tr>
<tr>
<td>0025 EA21</td>
<td>32</td>
</tr>
<tr>
<td>0027 EB21</td>
<td>33</td>
</tr>
<tr>
<td>0029 BB16</td>
<td>34</td>
</tr>
<tr>
<td>002E 18</td>
<td>35</td>
</tr>
<tr>
<td>002C A5</td>
<td>36</td>
</tr>
<tr>
<td>002D F8</td>
<td>37</td>
</tr>
<tr>
<td>002E A3</td>
<td>38</td>
</tr>
<tr>
<td>002F A9</td>
<td>39</td>
</tr>
<tr>
<td>0030 C637</td>
<td>40</td>
</tr>
<tr>
<td>0032 37</td>
<td>41</td>
</tr>
<tr>
<td>0033 C63A</td>
<td>42</td>
</tr>
<tr>
<td>0035 0421</td>
<td>43</td>
</tr>
<tr>
<td>0037 B5</td>
<td>44</td>
</tr>
<tr>
<td>0038 0425</td>
<td>45</td>
</tr>
</tbody>
</table>

Ref.# AB114
<table>
<thead>
<tr>
<th>LOC</th>
<th>OBJ</th>
<th>SEQ</th>
<th>SOURCE STATEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>003A</td>
<td>99FE</td>
<td>53</td>
<td>FINISH: ANL P1,#0FEH; RESET O/P BIT (PORT 1 BIT 0)</td>
</tr>
<tr>
<td>003C</td>
<td>65</td>
<td>55</td>
<td>STOP TCNT</td>
</tr>
<tr>
<td>003D</td>
<td>93</td>
<td>56</td>
<td>RETR; ** END OF ROUTINE **</td>
</tr>
<tr>
<td>003E</td>
<td>65</td>
<td>57</td>
<td>TIMER OVERFLOW ROUTINE:</td>
</tr>
<tr>
<td>003F</td>
<td>37</td>
<td>58</td>
<td>STOP TCNT</td>
</tr>
<tr>
<td>0040</td>
<td>95</td>
<td>59</td>
<td>FLAG SET HI &amp; LO ON</td>
</tr>
<tr>
<td>0041</td>
<td>B647</td>
<td>60</td>
<td>ALTERNATE CYCLES</td>
</tr>
<tr>
<td>0043</td>
<td>8901</td>
<td>61</td>
<td>JF0 OPRST</td>
</tr>
<tr>
<td>0045</td>
<td>0449</td>
<td>62</td>
<td>ORL P1,#01H; SET OUTPUT BIT (PORT 1 BIT 0)</td>
</tr>
<tr>
<td>0047</td>
<td>99FE</td>
<td>63</td>
<td>JMP CONT</td>
</tr>
<tr>
<td>004F</td>
<td>F9</td>
<td>64</td>
<td>ANL P1,#0FEH; RESET OUTPUT BIT</td>
</tr>
<tr>
<td>004A</td>
<td>37</td>
<td>65</td>
<td>MOV A,R1; NOTE DIVISION FACTOR</td>
</tr>
<tr>
<td>004B</td>
<td>62</td>
<td>66</td>
<td>CPL A</td>
</tr>
<tr>
<td>004C</td>
<td>45</td>
<td>67</td>
<td>ALTERNATE CYCLES</td>
</tr>
<tr>
<td>004D</td>
<td>0421</td>
<td>68</td>
<td>JMP LOOP1; RETURN TO &quot;NOTE DURATION&quot; LOOP</td>
</tr>
<tr>
<td>0016</td>
<td></td>
<td>69</td>
<td>TEMPO EQU 16H; DEFINES LENGTH OF NOTES</td>
</tr>
<tr>
<td>003C</td>
<td></td>
<td>70</td>
<td>TC EQU 60D; TOP C</td>
</tr>
<tr>
<td>0040</td>
<td></td>
<td>71</td>
<td>B EQU 64D; B</td>
</tr>
<tr>
<td>0043</td>
<td></td>
<td>72</td>
<td>A? EQU 67D; A#</td>
</tr>
<tr>
<td>0047</td>
<td></td>
<td>73</td>
<td>AA EQU 71D; A</td>
</tr>
<tr>
<td>004B</td>
<td></td>
<td>74</td>
<td>G? EQU 75D; G#</td>
</tr>
<tr>
<td>0050</td>
<td></td>
<td>75</td>
<td>G EQU 80D; G</td>
</tr>
<tr>
<td>0055</td>
<td></td>
<td>76</td>
<td>F? EQU 85D; F#</td>
</tr>
<tr>
<td>0059</td>
<td></td>
<td>77</td>
<td>F EQU 89D; F</td>
</tr>
<tr>
<td>005F</td>
<td></td>
<td>78</td>
<td>E EQU 95D; E</td>
</tr>
<tr>
<td>0065</td>
<td></td>
<td>79</td>
<td>D? EQU 101D; D#</td>
</tr>
<tr>
<td>0068</td>
<td></td>
<td>80</td>
<td>D EQU 107D; D</td>
</tr>
<tr>
<td>0071</td>
<td></td>
<td>81</td>
<td>C? EQU 113D; C#</td>
</tr>
<tr>
<td>0078</td>
<td></td>
<td>82</td>
<td>BC EQU 120D; BC</td>
</tr>
<tr>
<td>0000</td>
<td></td>
<td>83</td>
<td>P EQU 00H; PAUSE- NO TONE FOR 1 CYCLE</td>
</tr>
<tr>
<td>0050</td>
<td>65</td>
<td>85</td>
<td>DB P, D, D, P, D?, P, F, F</td>
</tr>
<tr>
<td>0052</td>
<td>65</td>
<td>86</td>
<td>4/78</td>
</tr>
<tr>
<td>0054</td>
<td>00</td>
<td>87</td>
<td></td>
</tr>
<tr>
<td>0055</td>
<td>59</td>
<td>88</td>
<td></td>
</tr>
<tr>
<td>0056</td>
<td>59</td>
<td>89</td>
<td></td>
</tr>
<tr>
<td>0057</td>
<td>00</td>
<td>90</td>
<td></td>
</tr>
<tr>
<td>0058</td>
<td>6B</td>
<td>91</td>
<td>4-528</td>
</tr>
<tr>
<td>LOC</td>
<td>OBJ</td>
<td>SEQ</td>
<td>SOURCE STATEMENT</td>
</tr>
<tr>
<td>------</td>
<td>-----</td>
<td>-----</td>
<td>------------------</td>
</tr>
<tr>
<td>0059</td>
<td>6B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>005A</td>
<td>6B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>005B</td>
<td>00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>005C</td>
<td>6B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>005D</td>
<td>00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>005E</td>
<td>59</td>
<td></td>
<td></td>
</tr>
<tr>
<td>005F</td>
<td>59</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0060</td>
<td>00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0061</td>
<td>50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0062</td>
<td>00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0063</td>
<td>50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0064</td>
<td>50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0065</td>
<td>50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0066</td>
<td>00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0067</td>
<td>4B</td>
<td>100</td>
<td>DB F, P, G, G, P</td>
</tr>
<tr>
<td>0068</td>
<td>4B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0069</td>
<td>00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>006A</td>
<td>50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>006B</td>
<td>50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>006C</td>
<td>50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>006D</td>
<td>00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>006E</td>
<td>59</td>
<td></td>
<td></td>
</tr>
<tr>
<td>006F</td>
<td>00</td>
<td>102</td>
<td>DB P, D?, D?, P, F</td>
</tr>
<tr>
<td>0070</td>
<td>65</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0071</td>
<td>65</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0072</td>
<td>00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0073</td>
<td>59</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0074</td>
<td>59</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0075</td>
<td>00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0076</td>
<td>65</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0077</td>
<td>65</td>
<td>103</td>
<td>DB D?, P, D, D, P, D?, D?, D?</td>
</tr>
<tr>
<td>0078</td>
<td>00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0079</td>
<td>6B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>007A</td>
<td>6B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>007B</td>
<td>00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>007C</td>
<td>65</td>
<td></td>
<td></td>
</tr>
<tr>
<td>007D</td>
<td>65</td>
<td></td>
<td></td>
</tr>
<tr>
<td>007E</td>
<td>65</td>
<td></td>
<td></td>
</tr>
<tr>
<td>007F</td>
<td>65</td>
<td>104</td>
<td>DB D?, P, D?, P, F, P, G</td>
</tr>
<tr>
<td>0080</td>
<td>00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0081</td>
<td>65</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0082</td>
<td>00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0083</td>
<td>59</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0084</td>
<td>00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0085</td>
<td>50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0086</td>
<td>00</td>
<td>105</td>
<td>DB P, G?, P, A?</td>
</tr>
<tr>
<td>0087</td>
<td>4B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0088</td>
<td>00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0089</td>
<td>43</td>
<td></td>
<td></td>
</tr>
<tr>
<td>008B</td>
<td>00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>008C</td>
<td>43</td>
<td></td>
<td></td>
</tr>
<tr>
<td>008D</td>
<td>43</td>
<td></td>
<td></td>
</tr>
<tr>
<td>008E</td>
<td>00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>008F</td>
<td>43</td>
<td>4/78</td>
<td></td>
</tr>
</tbody>
</table>

4-529
<table>
<thead>
<tr>
<th>LOC</th>
<th>OBJ</th>
<th>SEQ</th>
<th>SOURCE STATEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0090</td>
<td>43</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0091</td>
<td>00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0092</td>
<td>43</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0094</td>
<td>43</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0095</td>
<td>00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0096</td>
<td>4B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0097</td>
<td>00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0098</td>
<td>50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0099</td>
<td>50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>009B</td>
<td>4B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>009C</td>
<td>4B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>009D</td>
<td>00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>009E</td>
<td>4B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>009F</td>
<td>4B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00A0</td>
<td>00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00A1</td>
<td>4B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00A3</td>
<td>00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00A4</td>
<td>4B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00A5</td>
<td>4B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00A6</td>
<td>4B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00A7</td>
<td>00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00A8</td>
<td>50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00A9</td>
<td>00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00AB</td>
<td>59</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00AC</td>
<td>00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00AD</td>
<td>50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00AE</td>
<td>50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00AF</td>
<td>00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00B0</td>
<td>4B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00B1</td>
<td>00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00B3</td>
<td>00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00B4</td>
<td>59</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00B5</td>
<td>00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00B6</td>
<td>65</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00B7</td>
<td>00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00B8</td>
<td>50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00B9</td>
<td>50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00BB</td>
<td>00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00BC</td>
<td>4B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00BD</td>
<td>00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00BE</td>
<td>43</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00BF</td>
<td>43</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00C0</td>
<td>00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00C1</td>
<td>3C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00C2</td>
<td>00</td>
<td>113</td>
<td>DB P, G?, P, G, G, P, F, F</td>
</tr>
<tr>
<td>00C3</td>
<td>4B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00C4</td>
<td>00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00C5</td>
<td>50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00C6</td>
<td>50</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4/78

4-530
<table>
<thead>
<tr>
<th>LOC</th>
<th>OBJ</th>
<th>SEQ</th>
<th>SOURCE STATEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>00C7</td>
<td>00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00C8</td>
<td>59</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00C9</td>
<td>59</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00CA</td>
<td>00</td>
<td>114</td>
<td>DB P, D?, D?, D?, D?, D?, D?, P</td>
</tr>
<tr>
<td>00CB</td>
<td>65</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00CC</td>
<td>65</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00CD</td>
<td>65</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00CE</td>
<td>65</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00CF</td>
<td>65</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00D0</td>
<td>65</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00D1</td>
<td>00</td>
<td>115</td>
<td>DB P, P, P, P, P, P, 0FFH</td>
</tr>
<tr>
<td>00D2</td>
<td>00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00D3</td>
<td>00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00D4</td>
<td>00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00D5</td>
<td>00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00D6</td>
<td>00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00D7</td>
<td>00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00D8</td>
<td>00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00D9</td>
<td>FF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>116</td>
<td></td>
<td>END</td>
<td></td>
</tr>
<tr>
<td>117</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**USER SYMBOLS**

A? 0043 AA 0047 B 0040 BC 0078 C? 0071 CONT
E 005F F 0059 F? 0055 FINISH 003A G 0050 G?
LOOP2 0025 NOTONE 0037 OPRST 0047 P 0000 START 0010 TAB
TIMER 003E TUNE 0014

ASSEMBLY COMPLETE, NO ERRORS
Card Reader Driver, Hollerith to ASCII Conversion

Punched card input utility.
Reads from Document M-200 Card Reader,
performs Hollerith to ASCII translation,
writes to user-specified diskette file on Intellec MDS
using ISIS-II file management.

1. Intellec MDS, any console, diskette drives.
2. Documation M-200 Card Reader.
3. CompuScan-designed card reader interface.

1. ISIS-II Intel Systems Implementation Supervisor.
2. Executable utility file.

Standard program invocation format is utilized; e.g. if the
utility is filed under the name CDRDRV, then the program is
called with the name of the destination diskette file as an
argument: CDRDRV filename where filename follows ISIS-II
file naming syntax, and may include a diskette drive specifier.

Diskette file is created with user-specified name;
data is ASCII coded; card records are trimmed of trailing
spaces, and a carriage return and line feed are appended
to each record.

---

Notes:
A. Console break is provided: CTRL-A from the console causes
reading to stop prior to the next card pick; the disk file
is closed, and control returns to ISIS.
B. Normal file termination via end-of-file card consisting of
a multi-punch of at least all eight punches 2 through 9 in
column 1.
*C. Practicality of implementation depends upon card reader
interface, but Hollerith to ASCII translation will be of
general interest.

| Registers Modified: | not applicable | Programmer: | George Cotsonas
| RAM Required: | code seg length: 2D6H | Alex Schapira |
| ROM Required: | data seg length: A3H | Company: | CompuScan, Inc. |
| | stack seg length: 64H | Address: | 900 Huyler Street |
| Maximum Subroutine Nesting Level: | unknown but minimal | City: | Teterboro |
| Assembler/Compiler Used: | ISO-II 8080/8085 Assembler v.1.0 | State: | New Jersey |
| | | | 07608 |
COMPARE files

Compares two files, and indicates whether they are identical or not.

MDS-800

ISIS-I or II

Execute by the command

-COMPAR filename1 filename2

(same or comma between filenames)

If files identical, a message to that effect. If not, the differences are listed on the console. If more than 8 bytes differ, further messages are not output, but the total number of errors is stated at the end. If the files are of different lengths, this is also stated.

Registers Modified:

ALL

Programmer:

D. W. Wright

RAM Required:

32K

Company: Standard Telecommunication Laboratories Limited

ROM Required:

--

Address:

London Road

Maximum Subroutine Nesting Level:

City:

Harlow, Essex.

Assembler/Compiler Used:

PL/M-80 and ASM80

State:

UK

© Intel Corporation, 1976

Ref. # AB116

98-034D

4-535
SBC 80/10 8255 Test

Test SBC 80/10 8255 I/O Ports as Mode 0 outputs

SBC 80/10, TTY/CRT console

SBC 80 P Monitor

To initiate test type G3D0 on console.

To Console;
- #; on completion of test
- Port #, error bit pattern
  ; if an error occurs during test

Registers Modified:  
Programmer:  
P. N. Mark

RAM Required:  
Ø0D9H

Company:  
Durand Machine Co. Ltd.

ROM Required:  

Address:  
101-11th Street

City:  
New Westminster

Maximum Subroutine Nesting Level:

State:  
British Columbia, CANADA

Assembler/Compiler Used:

© Intel Corporation, 1976

4/78
LOC OBJ SEQ SOURCE STATEMENT

FFFF
  1  SBC80 SET 0FFFFFFH
  3
  4 ;TITLE=SBC80/10 8255 MODE 0 OUTPUT TEST PBC1V102077`
  5
  6 ;PERFORMS INCREASING & DECREASING BIT TEST ON SBC80/10
  7 ; 8255 PORTS (PROGRAMMED AS MODE 0 OUTPUTS)
  8 ; IF ANY ERROR IS ENCOUNTERED,
  9 ; THE PORT # & THE ERROR PATTERN IS OUTPUTED
 10 ; TO THE SYSTEM'S CONSOLE
 11 ;
 12 ; PROGRAM EQUATES
 13
 00E7
  14 CWR1 EQU 0E7H ; GROUP 1 CNTL REG
 00EB
  15 CWR2 EQU 0EBH ; " 2 "
 00E8
  16 MCH1 EQU 0E8H ; " 1 MODE 0 OUTPUT
 00E9
  17 MCH2 EQU 0E9H ; " 2 "
 00EA
  18 POT0 EQU 0E4H ; PORT 0 GROUP 1
 00E5
  19 POT1 EQU 0E5H ; " 1
 00EE
  20 POT2 EQU 0E6H ; " 2
 00E8
  21 POT3 EQU 0E8H ; PORT 3 GROUP 2
 00E9
  22 POT4 EQU 0E9H ; " 4
 00EA
  23 POT5 EQU 0EAH ; " 5
 24 ;
 25
 0600
  26 SETFW EQU 0600H ; FORWARD TEST CONSTANT
 0605
  27 SETBK EQU 0605H ; REVERSE " "
 01E8
  28 CO SET 01E8H ; SBC80P OUTPUT TO CONSOLE SUB
 0212
  29 EXIT SET 0212H ; " ERROR RECOVERY ROUTINE
 02C2
  30 HEX SET 02C2H ; " HEX OUTPUT SUB
 31
 32 IF NOT SBC80
 33 CO SET 0381H ; IF NOT SBC80P MONITOR
 34 EXIT SET 0150H
 35 ENDF
 36
 37
 2D00
  38 ORG 2D00H
 39
 40
 2D00 3E80
  41 MVI A, MCH1 ; INIT 8255'S
 2D02 D3E7
  42 OUT CWR1 ; GROUP 1
 2D04 3E80
  43 MVI A, MCH2
 2D06 D3EB
  44 OUT CWR2 ; " 2
 2D08 31FF3F
  45 LXI SP, 3FFFH ; INIT STACK
 2D0B 3E02
  46 MVI A, 2 ; SET TEST PASS CTR
 2D0F 05
  47 PUSH PSW ; &SAVE
 48
 49 GO:
 3D0E 110006
  50 LXI D, SETFW ; (D=PORT CTR E=PORT ID) FORWARD TEST
 3D11 0E00
  51 MVI B, 0 ; 0=FWD 1=REVERSE TEST DIRECTION
 3D13 21D13D
  52 LXI H, LOWN ; FWD TEST PATTERN PTR
<table>
<thead>
<tr>
<th>LOC</th>
<th>OBJ</th>
<th>SEQ</th>
<th>SOURCE STATEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>3D58</td>
<td>3E04</td>
<td>108</td>
<td>PORT4: ;SEE PORT 0 ROUTINE FOR DESCRIPTION</td>
</tr>
<tr>
<td>3D5A</td>
<td>92</td>
<td>109</td>
<td>MVI A, 4</td>
</tr>
<tr>
<td>3D5B</td>
<td>C2673D</td>
<td>110</td>
<td>SUB E</td>
</tr>
<tr>
<td>3D5E</td>
<td>7E</td>
<td>111</td>
<td>JNZ PORT5</td>
</tr>
<tr>
<td>3D5F</td>
<td>D3E9</td>
<td>112</td>
<td>MOV A, M</td>
</tr>
<tr>
<td>3D61</td>
<td>CDC23D</td>
<td>113</td>
<td>OUT POT4</td>
</tr>
<tr>
<td>3D64</td>
<td>DBE9</td>
<td>114</td>
<td>CALLDELAY</td>
</tr>
<tr>
<td>3D66</td>
<td>C9</td>
<td>115</td>
<td>IN POT4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>116</td>
<td>RET</td>
</tr>
<tr>
<td></td>
<td></td>
<td>117</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>118</td>
<td>PORT5: ;DEFAULT TO TEST PORT 5</td>
</tr>
<tr>
<td>3D67</td>
<td>7E</td>
<td>119</td>
<td></td>
</tr>
<tr>
<td>3D69</td>
<td>D3EA</td>
<td>120</td>
<td>MOV A, M</td>
</tr>
<tr>
<td>3D6A</td>
<td>CDC23D</td>
<td>121</td>
<td>OUT POT5</td>
</tr>
<tr>
<td>3D6D</td>
<td>DBE9</td>
<td>122</td>
<td>CALLDELAY</td>
</tr>
<tr>
<td>3D6F</td>
<td>C9</td>
<td>123</td>
<td>IN POT5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>124</td>
<td>RET</td>
</tr>
<tr>
<td></td>
<td></td>
<td>125</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>126</td>
<td>INDEX:</td>
</tr>
<tr>
<td>3D70</td>
<td>C1</td>
<td>127</td>
<td>POP B ;RESTORE TEST DIR</td>
</tr>
<tr>
<td>3D71</td>
<td>0D</td>
<td>128</td>
<td>DCR C ;DEC PORT BIT CTR</td>
</tr>
<tr>
<td>3D72</td>
<td>CA823D</td>
<td>129</td>
<td>JZ NEXT ;INDEX PORT CTR?</td>
</tr>
<tr>
<td>3D75</td>
<td>78</td>
<td>130</td>
<td>MOV A, B ;DIRECTION?</td>
</tr>
<tr>
<td>3D76</td>
<td>A7</td>
<td>131</td>
<td>ANA A</td>
</tr>
<tr>
<td>3D77</td>
<td>C27E3D</td>
<td>132</td>
<td>JNZ BACK ;FORWARD?</td>
</tr>
<tr>
<td>3D7A</td>
<td>23</td>
<td>133</td>
<td>INX H ;YES, INDEX PATTERN PTR</td>
</tr>
<tr>
<td>3D7B</td>
<td>C3183D</td>
<td>134</td>
<td>JMP TEST ;CONT FWD TEST</td>
</tr>
<tr>
<td></td>
<td></td>
<td>135</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>136</td>
<td>BACK:</td>
</tr>
<tr>
<td>3D7E</td>
<td>2B</td>
<td>137</td>
<td>DCX H ;BACKWARD TEST DEC PATTERN PTR</td>
</tr>
<tr>
<td>3D7F</td>
<td>C31D2D</td>
<td>138</td>
<td>JMP PORT0 ;CONT TEST</td>
</tr>
<tr>
<td></td>
<td></td>
<td>139</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>140</td>
<td>NEXT:</td>
</tr>
<tr>
<td>3D82</td>
<td>78</td>
<td>141</td>
<td>MOV A, B ;DIRECTION?</td>
</tr>
<tr>
<td>3D83</td>
<td>A7</td>
<td>142</td>
<td>ANA A</td>
</tr>
<tr>
<td>3D84</td>
<td>CA9B3D</td>
<td>143</td>
<td>JZ INC ;IF FWD INDEX TO HIGHER PORT</td>
</tr>
<tr>
<td>3D87</td>
<td>1D</td>
<td>144</td>
<td>DCR E ;ELSE DEC TO LOWER PORT</td>
</tr>
<tr>
<td>3D88</td>
<td>C38C3D</td>
<td>145</td>
<td>JMP INC+1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>146</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>147</td>
<td>INC:</td>
</tr>
<tr>
<td>3D8B</td>
<td>1C</td>
<td>148</td>
<td>INR E ;INDEX TO NEXT PORT</td>
</tr>
<tr>
<td>3D8C</td>
<td>15</td>
<td>149</td>
<td>DCR D ;DEC PORT CTR</td>
</tr>
<tr>
<td>3D8D</td>
<td>CA9B3D</td>
<td>150</td>
<td>JZ DONE? ;IF ALL PORT TESTED CHANGE DIRECTION</td>
</tr>
<tr>
<td>3D99</td>
<td>78</td>
<td>151</td>
<td>MOV A, B ;DIRECTION?</td>
</tr>
<tr>
<td>3D9A</td>
<td>A7</td>
<td>152</td>
<td>ANA A</td>
</tr>
<tr>
<td>3D9C</td>
<td>C2133D</td>
<td>153</td>
<td>JNZ GO+5 ;FWD, RESET FWD PATTERN PTR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>154</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>155</td>
<td>BKWD:</td>
</tr>
<tr>
<td>3D95</td>
<td>21D2D</td>
<td>156</td>
<td>LXI H, HII ;REVERSE PATTERN TEST PTR</td>
</tr>
<tr>
<td>3D98</td>
<td>C3163D</td>
<td>157</td>
<td>JMP PASS ;GO RESET PORT BIT CTR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>158</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>159</td>
<td>DONE?:</td>
</tr>
</tbody>
</table>

4-541
<table>
<thead>
<tr>
<th>LOC</th>
<th>OBJ</th>
<th>SEQ</th>
<th>SOURCE STATEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>3D98 F1</td>
<td>163</td>
<td>POP PSW ; RESTORE TEST PASS CTR</td>
<td></td>
</tr>
<tr>
<td>3D9C 8D</td>
<td>164</td>
<td>DCR A, ;2 TEST PASSES?</td>
<td></td>
</tr>
<tr>
<td>3D9D F5</td>
<td>165</td>
<td>PUSH PSW ; SAVE TEST PASS CTR FOR NEXT PASS</td>
<td></td>
</tr>
<tr>
<td>3D9E CA1202</td>
<td>166</td>
<td>JZ EXIT ; RETURN TO MONITOR IF DONE</td>
<td></td>
</tr>
<tr>
<td>167</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3DA1 78</td>
<td>168</td>
<td>MOV A, B ; DIRECTION?</td>
<td></td>
</tr>
<tr>
<td>3DA2 A7</td>
<td>169</td>
<td>ANA A</td>
<td></td>
</tr>
<tr>
<td>3DA3 C20E2D</td>
<td>170</td>
<td>JNZ GO ; BRANCH TO GO IF REVERSE</td>
<td></td>
</tr>
<tr>
<td>3DA6 43</td>
<td>171</td>
<td>MOV B, E ; PREV WAS FWD</td>
<td></td>
</tr>
<tr>
<td>3DA7 110506</td>
<td>172</td>
<td>LXI D, SETBK ; REVERSE TEST PATTERN</td>
<td></td>
</tr>
<tr>
<td>3DA9 C3953D</td>
<td>173</td>
<td>JMP BKWD ; GO TO REVERSE TEST</td>
<td></td>
</tr>
<tr>
<td>174</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3DAD BE</td>
<td>175</td>
<td>; CHECK FOR PATTERN MATCH</td>
<td></td>
</tr>
<tr>
<td>3DAE CA703D</td>
<td>176</td>
<td>; IF ERROR ENCOUNTERED, PRINT OUT PORT#</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>177</td>
<td>; ERROR BIT PATTERN, &amp; RETURN CONTROL TO MONITOR</td>
</tr>
<tr>
<td></td>
<td>178</td>
<td>;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>179</td>
<td>CHECK:</td>
<td></td>
</tr>
<tr>
<td>3DB1 F5</td>
<td>180</td>
<td>CMP M ; MATCH?</td>
<td></td>
</tr>
<tr>
<td>3DB2 7B</td>
<td>181</td>
<td>JZ INDEX ; INDEX TO NEXT BIT IF MATCH</td>
<td></td>
</tr>
<tr>
<td>182</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>183</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>184</td>
<td></td>
<td>IF SBC80</td>
<td></td>
</tr>
<tr>
<td>3DB4 F5</td>
<td>185</td>
<td>PUSH PSW ; ERROR, SAVE ERROR PATTERN</td>
<td></td>
</tr>
<tr>
<td>186</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>187</td>
<td></td>
<td>CALL HEX ; PRINT PORT#</td>
<td></td>
</tr>
<tr>
<td>188</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>189</td>
<td></td>
<td>CALL CO ; &quot; SPACE</td>
<td></td>
</tr>
<tr>
<td>190</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>201</td>
<td></td>
<td>CALL HEX ; &quot; ERROR PATTERN</td>
<td></td>
</tr>
<tr>
<td>192</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>203</td>
<td></td>
<td>JMP EXIT ; RETURN CONTROL TO MONITOR</td>
<td></td>
</tr>
<tr>
<td>204</td>
<td></td>
<td>ENDIF</td>
<td></td>
</tr>
<tr>
<td>194</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>195</td>
<td></td>
<td>IF NOT SBC80</td>
<td></td>
</tr>
<tr>
<td>196</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>197</td>
<td></td>
<td>PUSH PSW</td>
<td></td>
</tr>
<tr>
<td>198</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>199</td>
<td></td>
<td>MOV A, E</td>
<td></td>
</tr>
<tr>
<td>200</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>201</td>
<td></td>
<td>RST 4</td>
<td></td>
</tr>
<tr>
<td>202</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>203</td>
<td></td>
<td>POP PSW</td>
<td></td>
</tr>
<tr>
<td>204</td>
<td></td>
<td>ENDIF</td>
<td></td>
</tr>
<tr>
<td>205</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>206</td>
<td></td>
<td>;</td>
<td></td>
</tr>
<tr>
<td>207</td>
<td></td>
<td>5 SEC DELAY SUBROUTINE</td>
<td></td>
</tr>
<tr>
<td>208</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>209</td>
<td></td>
<td>DELAY:</td>
<td></td>
</tr>
<tr>
<td>210</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>211</td>
<td></td>
<td>PUSH B</td>
<td></td>
</tr>
<tr>
<td>212</td>
<td></td>
<td>MVI C, 0FFH ; SET 500MS CONSTANT</td>
<td></td>
</tr>
<tr>
<td>213</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>214</td>
<td></td>
<td>DLY1:</td>
<td></td>
</tr>
<tr>
<td>215</td>
<td></td>
<td>MVI B, 0FFH ; &quot; 2MS &quot;</td>
<td></td>
</tr>
<tr>
<td>216</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| 217 | | DLY2:
<table>
<thead>
<tr>
<th>LOC</th>
<th>OBJ</th>
<th>SEQ</th>
<th>SOURCE STATEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>3DC7</td>
<td>05</td>
<td>218</td>
<td>DCR B</td>
</tr>
<tr>
<td>3DC8</td>
<td>C2C7</td>
<td>219</td>
<td>JNZ DLY2 ; 2 MS?</td>
</tr>
<tr>
<td>3DCB</td>
<td>0D</td>
<td>220</td>
<td>DCR C</td>
</tr>
<tr>
<td>3DCC</td>
<td>C2C5</td>
<td>221</td>
<td>JNZ DLY1 ; 500 MS?</td>
</tr>
<tr>
<td>3DCF</td>
<td>C1</td>
<td>222</td>
<td>POP B</td>
</tr>
<tr>
<td>3DD0</td>
<td>C9</td>
<td>223</td>
<td>RET</td>
</tr>
<tr>
<td>224</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>225</td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>226</td>
<td></td>
<td></td>
<td>;TEST PATTERNS</td>
</tr>
<tr>
<td>227</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3DD1</td>
<td>0001</td>
<td>228</td>
<td>LOWW: DW 0100H</td>
</tr>
<tr>
<td>3DD3</td>
<td>0307</td>
<td>229</td>
<td>DW 0703H</td>
</tr>
<tr>
<td>3DD5</td>
<td>0F1F</td>
<td>230</td>
<td>DW 1F0FH</td>
</tr>
<tr>
<td>3DD7</td>
<td>3F</td>
<td>231</td>
<td>DB 3FH</td>
</tr>
<tr>
<td>3DD8</td>
<td>7FFF</td>
<td>232</td>
<td>HII: DW 0FF7FH</td>
</tr>
<tr>
<td>233</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>234</td>
<td></td>
<td></td>
<td>END</td>
</tr>
</tbody>
</table>

**PUBLIC SYMBOLS**

**EXTERNAL SYMBOLS**

**USER SYMBOLS**

<table>
<thead>
<tr>
<th>BACK</th>
<th>A 3D7E</th>
<th>BKWD</th>
<th>A 3D95</th>
<th>CHECK</th>
<th>A 3DAD</th>
<th>CO</th>
<th>A 01E8</th>
<th>CWR1</th>
<th>A 00E</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLY1</td>
<td>A 3DC5</td>
<td>DLY2</td>
<td>A 3DC7</td>
<td>DONE?</td>
<td>A 3D9B</td>
<td>EXIT</td>
<td>A 0212</td>
<td>GO</td>
<td>A 3D0</td>
</tr>
<tr>
<td>INC</td>
<td>A 3D8B</td>
<td>INDEX</td>
<td>A 3D70</td>
<td>LOWW</td>
<td>A 3DD1</td>
<td>MCW1</td>
<td>A 0080</td>
<td>MCW2</td>
<td>A 008</td>
</tr>
<tr>
<td>PORT0</td>
<td>A 3D1D</td>
<td>PORT1</td>
<td>A 3D2B</td>
<td>PORT2</td>
<td>A 3D3A</td>
<td>PORT3</td>
<td>A 3D49</td>
<td>PORT4</td>
<td>A 3D5</td>
</tr>
<tr>
<td>POT1</td>
<td>A 00E5</td>
<td>POT2</td>
<td>A 00E6</td>
<td>POT3</td>
<td>A 00E8</td>
<td>POT4</td>
<td>A 00E9</td>
<td>POT5</td>
<td>A 00E</td>
</tr>
<tr>
<td>SETFH</td>
<td>A 0600</td>
<td>TEST</td>
<td>A 3D18</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
μScope™ 820 Test Instrument, iSBC™ 80/10 Diagnostic Program

Performs GO/NO GO test and diagnostics on all iSBC™ 80/10 functional areas: CPU, RAM, ROM, SERIAL I/O, PARALLEL I/O OFF-BOARD RAM, & SELF-CHECK.

iSBC™ 80/10, μScope™ 820 Microprocessor System Console, 8080A μScope Probe

2716 EPROM with Diagnostic Program

None

See attached page

Available on non-system diskette only--$35.00. Source object and list file included The Applications Note AP-42 also included with program.
μSCOPE™ 820 TEST INSTRUMENT,
iSBC™ 80/10 DIAGNOSTIC PROGRAM

This program used in conjunction with Intel's μScope™ 820 Test Instrument dramatically reduces the costs to support microprocessor based systems in the field. For example, using this program a field service technician can walk up to a down iSBC™ 80/10 based computer system and,

1st with less than 10 key strokes uniquely configure the test program to the System's Options,
2nd run a complete CPU exercise with diagnostics to make sure the heart of the system is fine,
3rd test each of the major areas of the system and provide diagnostics easily understood by the technicians, and
4th provide advanced capabilities for finding intermittents and providing fault isolation to the component level.

The tests performed include:

- CPU at System Clock Rates
- RAM and ROM up to 64K
- Serial, Parallel, and Multibus™ I/O
- Self-Check on the Test Program itself

The program allows the user to create a 2716 EPROM which plugs into the socket on the μScope™ 820 Console front panel. The iSBC™ 80/10 Diagnostic Program is written to allow the user to easily modify sections to customize the program for his unique requirements. The program is heavily commented to aid the user. A more detailed discussion of the program is presented in Intel's Applications Note #42,"Writing Diagnostics with the μScope™ 820 Microprocessor System Console".
IOTEST - I/O Test Program for the SBC 80/20

To exercise all output lines of the 8255's and the 8251 USART for a teletype interface on an SBC 80/20 board.

An SBC 80/20 board and a teletype adapter such as the SBC 530. The program can be easily configured for a CRT at a higher baud rate.

NONE

NONE

Each 8 bit output port will operate as an 8 bit synchronous up counter allowing all I/O lines and drivers to be checked as well as testing the interface to the logic external to the board. The USART will transmit an ASCII 0 continuously until another character is received after which it will then repeat the new character. The tape reader control line is also toggled.

Registers Modified: A, B, C, H, L

Programmer: Lee Mandell

RAM Required: NONE

Company: LJM Associates

ROM Required: 77 Bytes

Address: 6331 Glade Ave., Suite 318

Maximum Subroutine Nesting Level: 0

City: Woodland Hills

Assembler/Compiler Used: ISIS II 8080/8085 MACRO V2.0

State: California 91367
ASM80: F1: IOTEST

ISIS-II 8080/8085 MACRO ASSEMBLER, V2.0

IOTEST PAGE 1

LOC OBJ SEQ SOURCE STATEMENT

1 ; I/O TEST PROGRAM FOR THE SBC 80/20
2 NAME IOTEST
3800 3  RAM EQU 3800H
00ED 4  TTYRP2 EQU 0EDH
0000 3ECE 6  START: MVI A, 0CEH
0002 3ED 7  OUT 0EDH
0004 3E27 8  MVI A, 27H
0006 3ED 9  OUT 0EDH
10 ; 8251 INITIALIZATION - 1.76 KHZ FOR TIMER #2
0008 3EB6 11  MVI A, 0B6H
000A 3DF 12  OUT 0DFH
000C 3E3 13  MVI A, 63H
000E 3DE 14  OUT 0DEH
0010 3E02 15  MVI A, 02H
0012 3DE 16  OUT 0DEH
17 ; OUTPUT PORTS INITIALIZATION
0014 3E80 18  MVI A, 80H
0016 3E7 19  OUT 0E7H
0018 3EB 20  OUT 0E6H
21 ;
22 ; OUTPUT 0 FROM TELETYPYE CONTINUOUSLY
23 ; UNLESS CHARACTER FROM TELETYPYE ENTERED
24 ; THEN TRANSMIT NEW CHARACTER CONTINUOUSLY
25 ;
26 ; TOGGLE READER CONTROL AT 10 HZ
27 ; TOGGLE ALL OTHER I/O AT LOOP RATE
28 ;
001A 013027 29 LOOP0: LXI B, 2730H
001B 210000 30  LXI H, 0
0020 23 31 LOOP: INX H
0021 7D 32  MOV A, L
0022 D3E4 33  OUT 0E4H
0024 D3E5 34  OUT 0E5H
0026 D3E6 35  OUT 0E6H
0028 D3E8 36  OUT 0E8H
002A D3E9 37  OUT 0E9H
002C D3EA 38  OUT 0EAH
002E DBED 39  IN TTYRP2
0030 E601 40  ANI 1
0032 CA3E00 41  JZ CONT1
0035 78 42  MOV A, B
0036 EE02 43  XRI 2
0038 47 44  MOV B, A
0039 D3ED 45  OUT TTYRP2
003B 79 46  MOV A, C
003C D3EC 47  OUT TTYRP2-1
003E DBED 48 CONT1: IN TTYRP2
0040 E602 49  ANI 2
0042 CA2000 50  JZ LOOP
0045 D8EC 51  IN TTYRP2-1
0047 E67F 52  ANI 7FH

6/78 4-551
<table>
<thead>
<tr>
<th>LOC</th>
<th>OBJ</th>
</tr>
</thead>
<tbody>
<tr>
<td>0049</td>
<td>4F</td>
</tr>
<tr>
<td>004A</td>
<td>C32000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SEQ</th>
<th>SOURCE STATEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>53</td>
<td>MOV C,A</td>
</tr>
<tr>
<td>54</td>
<td>JMP LOOP</td>
</tr>
<tr>
<td>55</td>
<td>END</td>
</tr>
</tbody>
</table>

PUBLIC SYMBOLS

EXTERNAL SYMBOLS

USER SYMBOLS
CONT1 A 003E LOOP A 0020 LOOP0 A 001A RAM A 3800 START A 000

ASSEMBLY COMPLETE, NO ERRORS
CUT-AND-PASTE EDITOR (PL/M)

This program provides cut-and-paste capability to augment the MDS-800 ISIS II editor and is nearly as fast as a disk to disk copy.

MDS-800 Development System with floppy disk and full 64K RAM. (Can redefine buffer dimension for smaller RAM memory.)

ISIS II Operating System (SYSTEM LIB and PLM 80 LIB also required for linking).

1. Input file name.
2. Input text file with insert and block delineators edited in.

1. Output file with blocks of data moved around and/or deleted (has original input file name.)
2. Original input file renamed with .BAK extension.

For further description see detailed comments in source listing which include sample paragraph before and after CUTPAS is executed.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>All</th>
<th>Programmer: Dr. A. J. Spuria</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM Required:</td>
<td>or 32K</td>
<td>Company: Bendix Corporation Communications Division</td>
</tr>
<tr>
<td>64K [Can configure for 48K]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ROM Required:</td>
<td>None</td>
<td>Address: E. Joppa Road</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td>N/A</td>
<td>City: Towson</td>
</tr>
<tr>
<td>Assembler/Compiler Used:</td>
<td>PL/M</td>
<td>State: Maryland 21204</td>
</tr>
</tbody>
</table>
/*
  CUT AND PASTE CAPABILITY TO AUGMENT THE NDS-880 EDITOR.
  THE EDITOR IS USED IN CONJUNCTION WITH THIS PROGRAM TO ALLOW BLOCKS
  OF ASCII DATA TO BE MOVED FROM ONE LOCATION AND INSERTED AT ONE OR
  MORE DIFFERENT LOCATIONS. THE ORIGINAL DATA WILL BE DELETED FROM THE
  ORIGINAL LOCATION UNLESS AN INSERT COMMAND IS PLACED THERE. IF NO INSERT
  COMMANDS ARE GIVEN FOR A BLOCK, IT IS MERELY DELETED AND NOT INSERTED
  ANYWHERE. UP TO 9 LEVELS OF NESTING ARE ALLOWED.

  IN ORDER TO USE CUTPAS IT IS FIRST NECESSARY TO USE THE EDITOR
  TO PLACE AN INSERT-HERE DELIMITER CHARACTER AND A BLOCK DELIMITER
  CHARACTER AS THE FIRST TWO CHARACTERS IN THE FILE TO BE CUT AND PASTED.
  IN THE FOLLOWING THESE WILL BE ASSUMED TO BE $ AND \ RESPECTIVELY FOR
  PURPOSES OF ILLUSTRATION. AFTER DEFINING THE DELIMITERS, IT IS NECESSARY
  TO PUT $N BEFORE AND AFTER EACH BLOCK TO BE MOVED AND \N ON EACH PLACE IT
  IS TO BE INSERTED, WHERE N IS ANY ASCII CHARACTER. AFTER EXITING THE
  EDITOR, INVOKE THE CUT AND PASTE CAPABILITY BY:

  CUTPAS :F1:FILENAME.EXT

  NOTE: THERE MUST BE EXACTLY 1 BLANK AFTER CUTPAS. THE :F1: MUST BE OMITTED
  IF THE FILE IS ON DRIVE 0. THE FILE EXTENSION EXT CANNOT BE TMP OR BAK. THE FILE
  FILENAME.EXT WILL BE RENAMED FILENAME.BAK UPON COMPLETION AND THE CUT-AND-PASTED
  FILE WILL BE NAMED FILENAME.EXT. THIS PROGRAM REQUIRE A
  65,536 NDS MEMORY COMPLEMENT SINCE THE WHOLE INPUT FILE IS HANDLED IN RAM
  TO MINIMIZE DISK ACCESS. THE LARGEST FILE THAT CAN BE CUT AND PASTED IS
  48,000 CHARACTERS INCLUDING $N AND \N MARKERS.

  THE FOLLOWING PARAGRAPH ILLUSTRATES THE USE OF THE MARKERS
  IN A FILE TO BE CUT AND PASTED. IT ALSO ILLUSTRATES THE USE OF NESTED BLOCKS
  ALTHOUGH THIS CAN BECOME QUITE COMPLICATED:

  PARAGRAPH BEFORE USING CUTPAS:

  @2:THIS SENTENCE WILL BE MOVED TO THE MIDDLE OF THE PARAGRAPH. @3:THIS SENTENCE WILL BE DELETED.
  \8:PART OF THIS SENTENCE \WILL BE DELETED. PART \WILL BE MOVED TO @3
  \3:THE END OF THE PARAGRAPH. @3:AND THE WHOLE SENTENCE EXCEPT FOR THE DELETED
  PART WILL APPEAR AT THE START OF THE PARAGRAPH. @2
  @1:NOTE THAT THE USER MUST KEEP TRACK OF CARRIAGE RETURNS AND LINE FEEDS
  AND PLACE HIS BLOCK DELIMITERS RESPECTIVELY. @1:

  PARAGRAPH AFTER USING CUTPAS:

  PART OF THIS SENTENCE WILL BE MOVED TO THE END OF THE PARAGRAPH
  AND THE WHOLE SENTENCE EXCEPT FOR THE DELETED
  PART WILL APPEAR AT THE START OF THE PARAGRAPH.
  THIS SENTENCE WILL BE MOVED TO THE MIDDLE OF THE PARAGRAPH.
  NOTE THAT THE USER MUST KEEP TRACK OF CARRIAGE RETURNS AND LINE FEEDS
AND PLACE HIS BLOCK DELINEATORS RESPECTIVELY.
THE END OF THE PARAGRAPH.

ERROR NUMBERS REPORTED ARE AS FOLLOWS:
101=ERROR IN READING .CI TO GET FILE NAME
102=ERROR IN OPENING INPUT FILE
103=ERROR IN OPENING OUTPUT FILE
104=ERROR IN READING INPUT FILE
105=ERROR IN WRITING OUTPUT FILE
106=ERROR IN CLOSING INPUT FILE
107=ERROR IN CLOSING OUTPUT FILE
108=ERROR IN RENAMING :F1:FILENAME EXT TO :F1:FILENAME.BAK
109=ERROR IN RENAMING :F1:FILENAME.TMP TO :F1:FILENAME.EXT
126=FOUND END OF FILE MARK WHILE SEARCHING FOR SEGMENT TO INSERT
127=FOUND END OF FILE MARK WHILE SEARCHING FOR END OF A BLOCK
128=NESTING DEEPER THAN 9 LEVELS
193=INPUT FILE LARGER THAN 15,568 CHARACTERS

ERROR NUMBERS 101 TO 109 ARE FOLLOWED BY THE CORRESPONDING ISIS II ERROR NUMBER
*/

1 OUTAS: DO:
$INCLUDE(:F1:ABBREV)

2 1 = DECLARE AS LITERALLY 'LITERALLY';
3 1 = DECLARE DCL AS 'DECLARE';
4 1 = DCL PROC AS 'PROCEDURE';
5 1 = DCL PUB AS 'PUBLIC';
6 1 = DCL EXT AS 'EXTERNAL';
7 1 = DCL STR AS 'STRUCTURE';
8 1 = DCL ADJ AS 'ADDRESS';
9 1 = DCL CSR AS '20H';
10 1 = DCL CSR AS '20H';
11 1 = DCL RELAYS AS '21H';
12 1 = DCL PASS AS '22H';
13 1 = DCL RESULT AS '23H';
14 1 = DCL T2ON AS '20H';
15 1 = DCL T2HIGH AS '20H';
16 1 = DCL IN AS 'INPUT';
17 1 = DCL OUT AS 'OUTPUT';
18 1 = DCL RET AS 'RETURN';
19 1 = DCL EN AS 'ENABLE';
20 1 = DCL DIS AS 'DISABLE';
21 1 = DCL POP AS 'STACKPTR=STACKPTR+2';
22 1 = DCL RIT AS 'DO:ENABLE;RETURN;END();'
/*BUFLEN MUST BE MODIFIED FOR SMALLER MEMORY CONFIGURATIONS*/
23 1 DCL BUFLEN AS '48000';
24 1 DCL BUFFER(BUFLEN) BYTE;
25 1 DCL (ACTUAL, STATUS, AFTRIN, AFTROUT, N, BUFPTR, START, COUNT) ADR;
26 1 DCL BUFSIZE BASED BUFPTR BYTE;
27 1 DCL (LEVEL, N, ATSIGN, BACKSLASH, SLASH) BYTE;
28 1 DCL ADR;
29 1 DCL STK(11) STR(START ADR, N BYTE);
30 1 DCL ORIGIN(15) BYTE;
31 1 DCL NEWNAME(15) BYTE;
32 1 DCL BACKNAME(15) BYTE;
33 1 DCL BLANK AS '20H';
34 1 DCL OR AS '13';
35 1 DCL PERIOD AS '2EH';
36 1 DCL EOFMARK AS '4';
DCL CHECK#STATUS AS 'IF STATUS<>0 THEN DO: CALL ERROR(M); CALL ERROR(STATUS); CALL EXIT; END';
OPEN: PROC(AFT,F,ACC,MODE,S)EXT;
   DCL (AFT,F,ACC,MODE,M)ADR;
END OPEN;
CLOSE: PROC(AFT,S)EXT;
   DCL (AFT,S)ADR;
END CLOSE;
READ: PROC(AFT,BUF,CNT,ACT,S)EXT;
   DCL (AFT,BUF,CNT,ACT,S)ADR;
END READ;
WRITE: PROC(AFT,BUF,CNT,S)EXT;
   DCL (AFT,BUF,CNT,S)ADR;
END WRITE;
RENAME: PROC(OLD,NEW,S)EXT;
   DCL (OLD,NEW,S)ADR;
END RENAME;
DELETE: PROC(NAME,S)EXT;
   DCL (NAME,S)ADR;
END DELETE;
EXIT: PROC EXT;
END EXIT;
ERROR: PROC(ERRORNUM)EXT;
   DCL ERRORNUM ADR;
END ERROR;

/*READ :CI: TO GET FILE NAME*/
CALL READ(1, BUFFER,128, actual, status);
M=188; CHECK#STATUS;

/*OPEN :F1:FILENAME, EXT FOR INPUT*/
CALL OPEN(AFT#IN, BUFFER, 1,0, status);
M=182; CHECK#STATUS;

/*SAVE ORIGINAL FILE NAME*/
CALL MOVE(15, BUFFER+1, ORIGNAME);

/*SETUP OUTPUT FILE NAME IN NEWNAME AND BACKUP FILE NAME IN BACKNAME*/
DO I=2 TO 11;
   IF (BUFFER(I)=BLANK) OR (BUFFER(I)=CR) THEN BUFFER(I)=PERIOD;
   IF BUFFER(I)=PERIOD THEN GO TO OPEN#OUT;
END;
OPEN#OUT: BUFFER(I+1)='.'; BUFFER(I+2)=':'; BUFFER(I+3)=';'; BUFFER(I+4)=BLANK;
   CALL MOVE(15, BUFFER+1, NEWNAME);

/*SETUP BACKUP FILE NAME IN BACKNAME*/
BUFFER(I+1)=';'; BUFFER(I+2)=';'; BUFFER(I+3)=';';
   CALL MOVE(15, BUFFER+1, BACKNAME);

/*OPEN FILE .F1:FILENAME, TMP FOR OUTPUT*/
CALL OPEN(AFTOUT, NAME_FILE, 2, 0, STATUS);
M=103; CHECK$STATUS;

/* READ WHOLE INPUT FILE INTO BUFFER */

BUFPRTR=BUFFER;
READ$IN CALL READ(AFT#IN,BUFPRTR,128, ACTUAL, STATUS);
M=104; CHECK$STATUS;
IF ACTUAL<128 THEN GO TO START$SEARCH;
BUFPRTR=BUFPRTR+128;

/* CHECK FOR FILE LARGER THAN BUFFER SIZE */

IF BUFPRTR>BUFFER(BUFLEN-132) THEN
DO;
CALL ERROR(199);
CALL EXIT;
END;
GO TO READ$IN;

/* SETUP SEARCH PARAMETERS AND PUT END OF FILE MARK AT END OF BUFFER */

START$SEARCH: START= BUFFER+2;
AT#SIGN=BUFFER(0);
BACK#SLASH=BUFFER(1);
BUFPRTR=BUFPRTR+ACTUAL;
BUFBYT=EOF$MARK;
LEVEL=0;

/* START SEARCH FOR @, \ OR END OF FILE MARK */

SEARCH: BUFPRTR=START;
DO I=0 TO BUFLEN;
IF (BUFBYT=AT#SIGN) OR (BUFBYT=BACK#SLASH) OR (BUFBYT=EOF$MARK) THEN GO TO WRITE$OUT;
BUFPRTR=BUFPRTR+1;
END;

/* WRITE STUFF OUT UP TO MARKER FOUND */

WRITE$OUT: COUNT=BUFPRTR-START;
CALL WRITE(AFT#OUT, START, COUNT, STATUS);
M=105; CHECK$STATUS;

/* NOW PROCESS INTERVAL DEPENDING ON MARKER FOUND */

IF BUFBYT=EOF$MARK THEN GO TO RENAME$FILES;
IF BUFBYT=AT#SIGN THEN GO TO INSERT$STUFF;

/* IF BUFBYT=BACK#SLASH THEN PROCESS AS FOLLOWS */

BUFPRTR=BUFPRTR+1;
/* IF AT LEVEL 0 THEN SKIP \n... \n BLOCK */

IF LEVEL=0 THEN GO TO SKIP$BLOCK;
/* AT LEVEL 0: SKIP BLOCK \n... \n IF AND ONLY IF N DOES NOT CORRESPOND
WITH ON FOR THE PREVIOUS LEVEL*/

147 1 IF BUFBYTOSTAK(LEVEL-1).N THEN GO TO SKIP#BLOCK;

/*/FOUND ENDING \\N. SO DECREMENT LEVEL CTR AND START SEARCHING
AFTER ON*/

149 1 LEVEL=LEVEL-1;
150 1 START=STAK(LEVEL).START;
151 1 GO TO SEARCH;

/*/TO SKIP BLOCK: POINT TO CHAR FOLLOWING ENDING \\N*/

152 1 SKIP#BLOCK: N=BUFBY;
153 1 DO I=1 TO BUFLEN;
154 2 BUFPTR=BUFPTR+1;
155 2 IF BUFBY=EOF#MARK THEN
156 2 DO CALL ERROR(197); CALL EXIT; END;
158 2 IF BUFBY=BACK#SLASH THEN GO TO END#LOOP;
159 2 BUFPTR=BUFPTR+1;
160 2 IF BUFBY=N THEN GO TO GON#;
162 2 END#LOOP: END;

/*/NOW GO START SEARCHING FOR NEXT MARKER*/

166 1 GON#N: START=BUFPTR+1;
167 1 GO TO SEARCH;

/*/INSERT STUFF FROM \\N...\\N INTO ON POSITION*/
/*/SAVE START AFTER ON AND SAVE N ON STAK AND INCR LEVEL CTR*/

168 1 INSERT#STUFF: BUFPTR=BUFPTR+1;
169 1 STAK(LEVEL).START=BUFPTR+1;
170 1 STAK(LEVEL).N.N=BUFBY;
171 1 LEVEL=LEVEL+1;
172 1 IF LEVEL=10 THEN DO CALL ERROR(198); CALL EXIT; END;

/*/FIND SEGMENT \N...\N*/

177 1 BUFPTR= BUFFER+2;
178 1 DO I=0 TO BUFLEN;
179 2 IF BUFBY=EOF#MARK THEN
180 2 DO CALL ERROR(196); CALL EXIT; END;
184 2 IF BUFBY=BACK#SLASH THEN GO TO END#LOOP2;
186 2 BUFPTR=BUFPTR+1;
187 2 IF BUFBY=N THEN GO TO GOS#SEG;
189 2 END#LOOP2: BUFPTR=BUFPTR+1;
190 2 END;

/*/FOUND SEGMENT... NOW GO SEARCH FOR NEXT MARKER*/

191 1 GOS#SEG: START=BUFPTR+1;
192 1 GO TO SEARCH;

/*/REACHED END OF FILE... CLOSE OUT FILE AND RENAME THEM*/

193 1 RENAMEFILES: CALL close(AFT#IN., STATUS);
194 1 M=106; CHECK#STATUS;
201 1 CALL CLOSE(AFIT#OUT, STATUS);
202 1 M=107; CHECK#STATUS;
209 1 CALL DELETE(BACKNAME, STATUS);
210 1 CALL RENAME(ORIGNAME, BACKNAME, STATUS);
211 1 M=108; CHECK#STATUS;
218 1 CALL RENAME(NEWNAME, ORIGNAME, STATUS);
219 1 M=109; CHECK#STATUS;
226 1 CALL EXIT;
227 1 END OUTPAS;

MODULE INFORMATION:

CODE AREA SIZE   = 0505H  12850
VARIABLE AREA SIZE = BBEEH  48100
MAXIMUM STACK SIZE = 0000H  80
289 LINES READ
0 PROGRAM ERROR(S)

END OF PL/M-80 COMPILATION
Hewlett Packard Calculator to MDS800 I/O Control Program—HPIO

This program inputs and outputs data and instructions between Hewlett Packard 9815 programmable calculator and MDS 800 memory.

RS232 interface for HP I/O.
TTY with current loop interface for program control.

MDS Monitor

To output to the HP calculator, data and instructions must be in memory starting at 7000H (relocatable if desired).

When inputting from HP calculator, memory will be load with data and instructions starting at 7000H (relocatable if desired).

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>A,C,D,E,H,L</td>
<td>John E. Kiesling</td>
</tr>
<tr>
<td>RAM Required:</td>
<td>Company:</td>
</tr>
<tr>
<td>100D - Program +</td>
<td>Quality Measurement Systems</td>
</tr>
<tr>
<td>Data Storage Space</td>
<td>Address:</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>2555 Baird Road</td>
</tr>
<tr>
<td>MDS-Monitor</td>
<td>City:</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td>Penfield,</td>
</tr>
<tr>
<td>1</td>
<td>State:</td>
</tr>
<tr>
<td>Assembler/Compiler Used:</td>
<td>New York 14526</td>
</tr>
<tr>
<td>LOC</td>
<td>OBJ</td>
</tr>
<tr>
<td>------</td>
<td>------</td>
</tr>
<tr>
<td>5000</td>
<td></td>
</tr>
<tr>
<td>00F7</td>
<td></td>
</tr>
<tr>
<td>F803</td>
<td></td>
</tr>
<tr>
<td>F809</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>5000</td>
<td></td>
</tr>
<tr>
<td>210070</td>
<td></td>
</tr>
<tr>
<td>5003</td>
<td>16FF</td>
</tr>
<tr>
<td>5005</td>
<td>3E81</td>
</tr>
<tr>
<td>5007</td>
<td>320300</td>
</tr>
<tr>
<td>500A</td>
<td>3E05</td>
</tr>
<tr>
<td>500C</td>
<td>D3F7</td>
</tr>
<tr>
<td>500E</td>
<td>CD03F8</td>
</tr>
<tr>
<td>5011</td>
<td>4F</td>
</tr>
<tr>
<td>5012</td>
<td>3E07</td>
</tr>
<tr>
<td>5014</td>
<td>3E80</td>
</tr>
<tr>
<td>5018</td>
<td>320300</td>
</tr>
<tr>
<td>501B</td>
<td>71</td>
</tr>
<tr>
<td>501C</td>
<td>3E81</td>
</tr>
<tr>
<td>501E</td>
<td>B9</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>5020</td>
<td>1E02</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>6/78</td>
<td></td>
</tr>
<tr>
<td>LOC</td>
<td>OBJ</td>
</tr>
<tr>
<td>------</td>
<td>-----</td>
</tr>
<tr>
<td>5030</td>
<td>23</td>
</tr>
<tr>
<td>5031</td>
<td>C30550</td>
</tr>
<tr>
<td>5034</td>
<td>1D</td>
</tr>
<tr>
<td>5035</td>
<td>C20550</td>
</tr>
<tr>
<td>5038</td>
<td>00</td>
</tr>
<tr>
<td>5039</td>
<td>C7</td>
</tr>
</tbody>
</table>

6000 67 ORG 6000H
6000 210070 68 START2: LXI H, 7600H
6000 70 SET H AND L TO THE
6003 16FF 69 BEGINNING OF THE
6000 71 DATA TO BE OUTPUT
6005 3E00 72
6007 320300 73
6009 4E 74 START3: MVI A, 00H
600B 23 75 STA 03H
600C 3E01 76 TRANSFER IO CONTROL TO
600E 320300 77
6011 66F7 78 BCK: IN CRTC
6013 6E81 79 ANI 81H
6015 C21160 80 WAIT FOR DEVICE TO BE
6018 CO09F8 81 NOT BUSY
601B 3E01 82
601D 89 83 MVI A, 01H
601F 84 84 STA 03H
6021 3E00 86 TRANSFER IO CONTROL TO
6023 0A 87
6024 3A3160 88 CALL 00H
6027 C30560 89 IF DATA = B1H (CODE
602A 1E02 90 FOR END) GO TO
602C 1600 91 ENDF1
602E C30560 92 JZ ENDF1
6031 1D 93 MVI A, 00H
6033 80 94 CMP D
6034 89 95 JZ CK1
6037 C30560 96 JMP START3
603A 1E02 97 ENDF1: MVI E, 02H
603C 1600 98 SET REG E = 2
603E C30560 99 THIS COUNTER WILL MAKE
6041 1D 100 SURE THE LAST TWO BYTES
6044 89 101 AFTER THE END
6046 89 102 ARE PICK UP
6048 89 103 MVI D, 00H
604A 89 104 SET REG D = 00 TO FLAG
604C 89 105 THAT THE NEXT BYTES ARE
604E 89 106 CHECK SUM BYTES
6050 89 107 CMP E
6052 1D 108 JZ START3
6053 1E02 109 ENDF1: MVI E, 02H
6055 1600 110 SET REG E = 2
6057 1600 111 THIS COUNTER WILL MAKE
6059 1600 112 SURE THE LAST TWO BYTES
605B 1600 113 AFTER THE END
605D 1600 114 ARE PICK UP
605F 1600 115 MVI D, 00H
6061 1600 116 SET REG D = 00 TO FLAG
6063 1600 117 THAT THE NEXT BYTES ARE
6065 1600 118 CHECK SUM BYTES
6067 1600 119 IF REG E = 0 ALL
LOC OBJ  SEQ SOURCE STATEMENT

6032 C20560  108 ; DATA HAS BEEN OUTPUT
6035 3E01     109 JNZ START3
6037 320300    110 DONE1: MVI A, 01H
603A 00        111 STA 03H
603B C7        112 NOP

113 RST 0

114

115 ;********************************************************************************
116 ; END
117 ;********************************************************************************
118 ;********************************************************************************

5000 119 END 5000H

PUBLIC SYMBOLS

EXTERNAL SYMBOLS

USER SYMBOLS

BCK A 6011 CI A F803 CK A 5034 CK1 A 6031 CO A F80
DONE1 A 6035 ENDF A 502C ENDF1 A 602A START A 5005 START1 A 500

ASSEMBLY COMPLETE, NO ERRORS
HP - Controller for Hewlett-Packard 9871A Printer

To interface a Hewlett-Packard 9871A Printer to an Intel MDS-800.

Intel MDS-800
Intel SBC-508
Hewlett-Packard 9871A

MDS Monitor V2.0
MDS Boot V2.0
ISIS-II V2.2

C-Register contains character for output by printer
Printer status on port ØDFH

Character to be printed appears on port ØDFH
Printer control on port ØDEH

This program outputs characters to the HP 9871A printer using the handshake routine described in Chapter Two of the 'Hewlett-Packard 9871A Printer Operating and Service Manual'. This manual is Hewlett-Packard Part Number Ø9871-90030.

Available on diskette only.

Registers Modified:
All

Programmer:
Carl Owenby, Jr.

RAM Required:
ØB3H for Loader; 69H for Driver

Company:
Gibson-Owenby, Inc.

ROM Required:
None

Address:
Post Office Box 973

Maximum Subroutine Nesting Level:
Two

City:
Quincy,

Assembler/Compiler Used:
ISIS-II PL/M 80 V3.0

State:
Florida 32351

© Intel Corporation, 1976

6/78
Five files are required to compile, link, and locate this program. They are HPLOD.SRC, HPDRV.SRC, HPLIDR.SRC, 9871A.DAT, and HP9871.CSD.

The HP9871.CSD file must be edited according to the amount of memory in the MDS-800 system. For a 16K system, the code parameter of the locate command should be changed to 3E00H. For a 32K system, the parameter should be 7E00H; for a 48K system it should be 0BE00H; and should be left at 0F600H for a system with 64K of RAM.

After making any required modification in the Command Sequence Definition file as described above, the program may be compiled by entering the command 'SUBMIT :F1:HP9871.CSD'. This should be done with the PL/M 80 compiler disk in Drive 0 and the program source files on the disk in Drive 1.

After compilation the program may be loaded and executed by entering the command 'HP'.

This interface has been operated successfully operated using a twenty foot length of thirty-four conductor flat cable connected in accordance with the run list in file 9871A.DAT.
ISIS-II PL/M-80 V3.0 COMPILATION OF MODULE HP9871ALOADER
OBJECT MODULE PLACED IN :F1;HPLOD.OBJ
COMPILER INVOKED BY: PLM80 :F1;HPLOD.SRC

$DATE(30JUL77)
$TITLE('HPLOD - HP 9871A LOADER')
$PAGewidth(72)
$PAGELENGTH(57)

HP$9871ALOADER:
  DO;

/*
THIS PROGRAM EXPECTS THE HEWLETT-PACKARD 9871A PRINTER
TO BE INTERFACED IN THE FOLLOWING FASHION TO THE MDS-800.
*/

$INCLUDE(:F1;9871A.DAT)

= 9871A DESCRIPTION SBC-508 DESCRIPTION
= PIN # PIN #
= 3 S1 (NOT USED) 91 I3DAT3/ (DATA BIT
= 4 S0 (NOT USED) 92 I3DAT2/ (DATA BIT
= 5 IO3/ (COVER ON) 89 I3DAT1/ (DATA BIT
- 3 TO INPUT PORT 3) 82 I3DAT0/ (DATA BIT
= 6 IO2/ (ABORT) 79 I3DAT0/ (DATA BIT
- 2 TO INPUT PORT 3) 79 I3DAT0/ (DATA BIT
= 7 IO1/ (READY) 79 I3DAT0/ (DATA BIT
- 1 TO INPUT PORT 3) 79 I3DAT0/ (DATA BIT
= 8 IO0/ (BUFFER SPACE) 90 I3DAT5/ (DATA BIT
- 0 TO INPUT PORT 3) 79 I3DAT0/ (DATA BIT
= 9 CMD/ (CONTROL) 51 O3DAT7/ (DATA BIT
- 0 FROM OUTPUT PORT 2) 51 O3DAT7/ (DATA BIT
= 10 HLT/ (STOP) 26 O2DAT1/ (DATA BIT
- 1 FROM OUTPUT PORT 2) 26 O2DAT1/ (DATA BIT
= 11 FLG/ (FLAG) 98 I3DAT5/ (DATA BIT
- 5 TO INPUT PORT 3) 98 I3DAT5/ (DATA BIT
= 12 CALC7/ 51 O3DAT7/ (DATA BIT
= 7 FROM OUTPUT PORT 3) 36 O3DAT6/ (DATA BIT
= 13 CALC6/ 53 O3DAT5/ (DATA BIT
- 6 FROM OUTPUT PORT 3) 53 O3DAT5/ (DATA BIT
= 14 CALC5/ 97 I3DAT4/ (DATA BIT
- 5 FROM OUTPUT PORT 3) 97 I3DAT4/ (DATA BIT
= 24 POP/ (POWER-ON PRESET) 51 O3DAT7/ (DATA BIT
- 4 TO INPUT PORT 3) 51 O3DAT7/ (DATA BIT
= 25 CHASSIS GROUND 39 O3DAT0/ (DATA BIT
= 26 LOGIC GROUND 39 O3DAT0/ (DATA BIT
= 27 LOGIC GROUND 39 O3DAT0/ (DATA Bit
= 28 CALC0/ 39 O3DAT0/ (DATA BIT
- 0 FROM OUTPUT PORT 3)
= 29  CALC1/
-  1 FROM OUTPUT PORT 3)  38  O3DAT1/ (DATA BIT
= 30  CALC2/
-  2 FROM OUTPUT PORT 3)  50  O3DAT2/ (DATA BIT
= 31  CALC3/
-  3 FROM OUTPUT PORT 3)  37  O3DAT3/ (DATA BIT
= 32  CALC4/
-  4 FROM OUTPUT PORT 3)  52  O3DAT4/ (DATA BIT
= 33  LOGIC GROUND
= 34  LOGIC GROUND
= 35  +5V@500MA

---

<table>
<thead>
<tr>
<th>PORT NUMBER</th>
<th>ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00DCH</td>
</tr>
<tr>
<td>1</td>
<td>00DDH</td>
</tr>
<tr>
<td>2</td>
<td>00DEH</td>
</tr>
<tr>
<td>3</td>
<td>00DFH</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>FLAT CABLE</th>
<th>9871A</th>
<th>SBC-508</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>WIRE#</td>
<td>PIN#</td>
<td>PIN#</td>
<td>GND</td>
</tr>
<tr>
<td>1</td>
<td>28</td>
<td>39</td>
<td>GND</td>
</tr>
<tr>
<td>2</td>
<td>29</td>
<td>38</td>
<td>CALC0/</td>
</tr>
<tr>
<td>3</td>
<td>30</td>
<td>50</td>
<td>GND</td>
</tr>
<tr>
<td>4</td>
<td>31</td>
<td>37</td>
<td>CALC1/</td>
</tr>
<tr>
<td>5</td>
<td>32</td>
<td>52</td>
<td>GND</td>
</tr>
<tr>
<td>6</td>
<td>14</td>
<td>53</td>
<td>CALC2/</td>
</tr>
<tr>
<td>7</td>
<td>13</td>
<td>36</td>
<td>GND</td>
</tr>
<tr>
<td>8</td>
<td>12</td>
<td>51</td>
<td>CALC3/</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>27</td>
<td>GND</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>26</td>
<td>CALC4/</td>
</tr>
<tr>
<td>11</td>
<td>8</td>
<td>90</td>
<td>GND</td>
</tr>
<tr>
<td>12</td>
<td>7</td>
<td>89</td>
<td>CALC5/</td>
</tr>
<tr>
<td>13</td>
<td>6</td>
<td>92</td>
<td>GND</td>
</tr>
<tr>
<td>14</td>
<td>5</td>
<td>91</td>
<td>CACL6/</td>
</tr>
<tr>
<td>15</td>
<td>11</td>
<td>98</td>
<td>GND</td>
</tr>
<tr>
<td>16</td>
<td>17</td>
<td>97</td>
<td>IO0/</td>
</tr>
<tr>
<td>17</td>
<td>18</td>
<td>87</td>
<td>GND</td>
</tr>
<tr>
<td>18</td>
<td>19</td>
<td>25</td>
<td>IO1/</td>
</tr>
<tr>
<td>19</td>
<td>20</td>
<td>22</td>
<td>GND</td>
</tr>
<tr>
<td>20</td>
<td>21</td>
<td>10</td>
<td>IO2/</td>
</tr>
<tr>
<td>21</td>
<td>22</td>
<td>8</td>
<td>GND</td>
</tr>
<tr>
<td>22</td>
<td>23</td>
<td>7</td>
<td>IO3/</td>
</tr>
<tr>
<td>23</td>
<td>24</td>
<td>6</td>
<td>GND</td>
</tr>
<tr>
<td>24</td>
<td>25</td>
<td>5</td>
<td>FLG/</td>
</tr>
<tr>
<td>25</td>
<td>26</td>
<td>11</td>
<td>GND</td>
</tr>
<tr>
<td>26</td>
<td>27</td>
<td>24</td>
<td>POP/</td>
</tr>
<tr>
<td>27</td>
<td>28</td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>29</td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>30</td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>31</td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>32</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
DECLARE DCL LITERALLY 'DECLARE';
DCL AS LITERALLY 'LITERALLY';
DCL PROC AS 'PROCEDURE';

DCL MEMTOP BYTE AT (0005H),
   FILE$NAME (*) BYTE DATA ('HPRES '),
   ENTRY ADDRESS,
   STATUS ADDRESS;

LOAD:
   PROC (FILE,BIAS,RETSW,ENTRY,STATUS) EXTERNAL;
   DCL (FILE,BIAS,RETSW,ENTRY,STATUS) ADDRESS;
END LOAD;

MEMCK:
   PROC ADDRESS EXTERNAL;
END MEMCK;

CALL MOVE (319,MEMCK+1,MEMCK-511);

MEMTOP = HIGH (MEMCK - 193);

CALL LOAD (.FILENAME,0,1,.ENTRY,.STATUS);
END HP$9871A$LOADER;

MODULE INFORMATION:
CODE AREA SIZE = 004DH    77D
VARIABLE AREA SIZE = 0004H    4D
MAXIMUM STACK SIZE = 0008H    8D
114 LINES READ
0 PROGRAM ERROR(S)

END OF PL/M-80 COMPILATION
$DATE (30JUL77)
$TITLE ('HPDRV - HP 9871A DRIVER')
$PAGEWIDTH (72)
$PAGELENGTH (57)

1
HP$9871A$RESIDENT$DRIVER:
   DO;

2   1
   DECLARE DCL LITERALLY 'DECLARE';
3   1
   DCL AS LITERALLY 'LITERALLY';
4   1
   DCL PROC AS 'PROCEDURE';

5   1
   DCL PRINTERTL CONTROL AS 'ODEH',
      CONTROL$LINE$HIGH AS '0';

6   1
   EXIT:
      PROC EXTERNAL;
7   2
      END EXIT;

8   1
   IOCHK:
      PROC BYTE EXTERNAL;
9   2
      END IOCHK;

10  1
   ISET:
      PROC (STATUS) EXTERNAL;
11  2
      DCL STATUS BYTE;
12  2
      END ISET;

13  1
   IODEF:
      PROC (DEVICE,LOCATION) EXTERNAL;
14  2
      DCL DEVICE BYTE,
15  2
         LOCATION ADDRESS;
16  1
      END IODEF;

17  1
   LIST$DRIVER:
      PROC (CHARACTER) EXTERNAL;
18  2
      DECLARE CHARACTER BYTE;
19  2
      END LIST$DRIVER;

20  1
   OUTPUT (PRINTERTL CONTROL) =
      CONTROL$LINE$HIGH;

21  1
   CALL IODEF (6,.LIST$DRIVER);
22  1
   CALL IOSET (IOCHK AND 0011$1111B OR 1100$0000B);
23  1
   CALL EXIT;
END HP$9871A$RESIDENT$DRIVER;

MODULE INFORMATION:

  CODE AREA SIZE  = 001FH       31D
  VARIABLE AREA SIZE = 0000H     0D
  MAXIMUM STACK SIZE = 0002H     2D
  49 LINES READ
  0 PROGRAM ERROR(S)

END OF PL/M-80 COMPILATION
ISIS-II PL/M-80 V3.0 COMPILATION OF MODULE HP9871ALISTDRIVER
OBJECT MODULE PLACED IN :F1:HPLIDR.OBJ
COMPILED BY: PLM80 :F1:HPLIDR.SRC

$DATE(30JUL77)
$TITLE('HPLIDR - HP 9871A LIST DRIVER')
$PAGewidth(72)
$PAGELength(57)

1
HP$9871A$LIST$DRIVER:
   DO;

   2   1
   DECLARE DCL LITERALLY 'DECLARE';
       DCL AS LITERALLY 'LITERALLY';
       DCL PROC AS 'PROCEDURE';

   5   1
       DCL PRINTER$STATUS AS '0DFH',
          CHARACTER$OUTPUT$PORT AS '0DFH',
          PRINTER$CONTROL AS '0DEH',
          CONTROL$LINE$LOW AS '01H',
          CONTROL$LINE$HIGH AS '0',
          PRINTER$NOT$READY AS 'NOT ROR (INPUT (PRINTER$STAT
          US),1)',
          FLAG$IS$LOW AS 'NOT ROL (INPUT (PRINTER$STATUS),3)
          ',';

   6   1
      LIST$DRIVER:
      PROC (CHARACTER) PUBLIC;
      DCL CHARACTER BYTE;

      8   2
      DO WHILE PRINTER$NOT$READY;
      END;

      10  2
      OUTPUT (CHARACTER$OUTPUT$PORT) = CHARACTER;

      11  2
      OUTPUT (PRINTER$CONTROL) = CONTROL$LINE$LOW;

      12  2
      DO WHILE FLAG$IS$LOW;
      END;

      14  2
      OUTPUT (PRINTER$CONTROL) = CONTROL$LINE$HIGH;

      15  2
      END LIST$DRIVER;

      16  1
      END HP9871A$LIST$DRIVER;

MODULE INFORMATION:

       CODE AREA SIZE      = 0028H   40D

6/78 4-577
VARIABLE AREA SIZE = 0001H
MAXIMUM STACK SIZE = 0000H
40 LINES READ
0 PROGRAM ERROR(S)

END OF PL/M-80 COMPILATION
Print text for SBC 80/10

To print text and numerical data on a teletype with spacing inserted as desired.

SBC 8080 (8010)

SBC 80P Monitor

Text (ASCII), Numeric data (FPU format) and output buffer address and length.

The text will be output with spacing inserted as desired followed by optional output of numeric data. A check is made to insure that the buffer is not overflowed.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>ALL</th>
<th>Programmer: Doug Heere</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM Required:</td>
<td>61 bytes</td>
<td>Company: PPG Industries</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>None</td>
<td>Address: Box 400</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td>3</td>
<td>City: Wichita Falls</td>
</tr>
<tr>
<td>Assembler/Compiler Used:</td>
<td>PDP-11</td>
<td>State: Texas 76307</td>
</tr>
</tbody>
</table>
1; PROGRAM PRDAT
2; THIS PROGRAM IS DESIGNED TO PRINT DATA ON A
3; TELETYPE. THE DATA IS A MIXTURE OF TEXT AND NUMERIC
4; DATA. VARYING LENGTH MESSAGES ARE LOADED FROM PROM
5; OR RAM AREAS INTO AN OUTPUT BUFFER IN RAM. THE
6; NUMERIC MUST PREVIOUSLY HAVE BEEN LOADED INTO A
7; BUFFER IMMEDIATELY FOLLOWING THE OUTPUT BUFFER.
8; THE NUMERIC DATA IS IN FLOATING POINT UNIT FORMAT.
9; THE PRINT PROGRAM ALLOWS THE INSERTION OF SPACES
10; IN THE TEXT AND CONTROL OF PRINTING BOTH TEXT AND
11; NUMERIC DATA, OR TEXT ONLY. AS THE TEXT IS LOADED
12; BY THE PROGRAM, IT IS CHECKED FOR SPACING CHARACTERS
13; (OEH - OCH) AND PRINT CONTROL CHARACTERS (OFEH, OFFH).
14; WHEN A SPACING CHARACTER IS FOUND, THE EQUIVALENT
15; NUMBER OF SPACES IS INSERTED INTO THE OUTPUT BUFFER.
16; THE PRINT CONTROL CHARACTER IS THE LAST CHARACTER IN
17; THE TEXT. A VALUE OF OFFH CAUSES ONLY THE TEXT TO BE
18; PRINTED, WHILE A VALUE OF OFEH CAUSES OUTPUT OF THE
19; NUMERIC DATA FOLLOWING THE TEXT.
20; THE FOLLOWING EXAMPLE SHOWS THE BUFFER RELATIONSHIP
21; AND TEXT STRING SETUP. MESS1 WILL OUTPUT ONLY THE
22; TEXT ' TEXT1 ', WHILE MESS2 WILL CAUSE OUTPUT
23; OF THE TEXT ' TEXT2 ' FOLLOWED BY NUMERIC
24; DATA PREVIOUSLY LOADED IN THE ANSWER BUFFER.
25;
26; OUTBF SET $       ; OUTPUT BUFFER
27; DS 30             ; 30 BYTES
28; ANSWR SET $       ; NUMERIC DATA
29; DS 17             ; 17 BYTES
30;
31; MESS1: DB 00H     ; CARRIAGE RETURN
32; DB 04H            ; 4 SPACES
33; DB ' TEXT1 '      ; TEXT
34; DB 03H            ; 3 SPACES
35; DB 0FEH           ; PRINT CONTROL, TEXT ONLY
36; MESS2: DB ' TEXT2 ' ; TEXT
37; DB 05H            ; 5 SPACES
38; DB 0FFH           ; PRINT CONTROL, TEXT AND
39;
40; THE LENGTH OF THE BUFFER IS INPUT TO THE ROUTINE IF
41; OVERFLOW CHECKING. IF THE TEXT AS EXPANDED OVERFLOWS TO
42; BUFFER, IT IS TRUNCATED.
43;
44; INPUT ARGUMENTS:
45; REG C   - LENGTH OF OUTPUT BUFFER (NO
46; INCL ANSWER BUFFER)
47; REG D&E  - ADDRESS OF OUTPUT BUFFER
48; REG H&L  - ADDRESS OF TEXT
49;
50; REGISTERS DESTROYED: ALL
51;
LOC OBJ  SEQ  SOURCE STATEMENT

53
54  ;  MONITOR ROUTINES USED: ECHO
55
56  ORG 4400H
57
58  ECHO EQU 01F9H  ;  MONITOR ROUTINE
59
60  ;  MAIN PROGRAM ENTRY POINT
61

62  PRDAT SET $  ;  SAVE OUTPUT BUFFER ADDRESS
63  D5
64  CALL LDTXT  ;  LOAD FIXED TEXT TO BUFFER
65  CD1C44
66  POP H  ;  GET OUTPUT BUFFER ADDRESS
67
68  ;  LAST CHARACTER IN FIXED TEXT (PRINT CONTROL) IS RETURN
69  ;  IN ACCUMULATOR BY LDTXT. SAVE IT FOR LATER
70
71  MOV B,A  ;  PUT LAST CHAR IN REG B
72
73  MOV B,C  ;  SAVE B AND C
74
75
76  ;  REGISTER C CONTAINS NUMBER OF SLOTS IN BUFFER NOT FILE
77  ;  BY FIXED TEXT. THIS IS USED TO LOCATE ANSWER BUFFER
78
79
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107

6/78

4-583
108 ; INPUT -
109 ; C - BUFFER SIZE
110 ; D&E - OUTPUT BUFFER ADDRESS
111 ; H&L - TEXT ADDRESS
112 ;
113 ; OUTPUT -
114 ; C - UNUSED PORTION OF OUTPUT BUFFER
115 ;
116 LDTXT SET $ ; ENTRY POINT
117 DCR C ; ADJUST BUFFER COUNTER
118 GETCH: MOV A,M ; GET TEXT CHARACTER
119 ;
120 TEST FOR SPACING CHARACTER <0DH
121 ;
122 CPI 0DH ; IS IT SPACING CHARACTER?
123 JNC LODCH ; NO, LOAD IT
124 ;
125 SPACING CHARACTER FOUND. INSERT SPACES IN OUTPUT BUFFER
126 ;
127 MOV B,A ; SET UP COUNTER
128 DCR B ; AND ADJUST IT
129 XCHG ; POINT H&L TO OUTPUT BUFFER
130 SPACE: MVI M,20H ; INSERT SPACE
131 INX H ; ADJUST BUFFER POINTER
132 ;
133 CHECK FOR BUFFER OVERFLOW
134 ;
135 DCR C ; DECREMENT BUFFER SPACE AVAILABLE
136 JZ TRUNC ; IF FULL, TRUNCATE
137 ;
138 LOOP UNTIL REQUIRED NUMBER OF SPACES INSERTED
139 ;
140 DCR B ; DECREMENT SPACE COUNTER
141 JP SPACE ; LOOP FOR DESIRED NUMBER
142 ;
143 ADJUST BUFFER COUNTER
144 ;
145 INR C ; ADJUST ROOM IN BUFFER
146 ;
147 CHECK FOR END OF BUFFER AND LOAD NEXT CHARACTER
148 ;
149 NXTCH: DCR C ; DECREMENT ROOM IN BUFFER
150 JZ TRUNC ; IF FULL TRUNCATE
151 XCHG ; POINT H&L TO TEXT
152 INX H ; NEXT CHARACTER
153 JMP GETCH ; LOAD NEXT CHARACTER
154 ;
155 NON SPACING CHARACTER FOUND. CHECK FOR PRINT CONTROL
156 ;
157 LODCH: CPI 0FDH ; IS IT PRINT CONTROL?
158 JC NOTPC ; NO, GO LOAD IT
159 ;
160 PRINT CONTROL CHARACTER FOUND, END LOADING
161 ;
162 ;
LOC OBJ SEQ SOURCE STATEMENT

163 ; ENTRY FOR END OF TEXT OR TRUNCATION
164 ; PUT ZERO IN LAST POSITION, RETURN REMAINING BUFFER ROOM
165 ; REG C
166 ;
4440 EB 167 ENDCN: XCHG ; POINT H & L TO OUTPUT BUFFER
4441 3600 168 MVI M, 00H ; PUT ZERO IN LAST POSITION
4443 EB 169 XCHG ; POINT H & L TO TEXT
4444 C9 170 RET ; EXIT
171 ;
172 ; MESSAGE TRUNCATED, FIND PRINT CONTROL CHARACTER
173 ;
4445 EB 174 TRUNC: XCHG ; POINT TO TEXT
4446 7E 175 LOOPA: MOV A, M ; GET TEXT CHARACTER
4447 FEFD 176 CPI 0FH ; IS IT PRINT CNTRL
4449 D24044 177 JNC ENDCN* ; YES, FINISH
444C 23 178 INX H ; NO, POINT TO NEXT CHARACTER
444D C34644 179 JMP LOOPA ; AND CHECK IT
180 ;
181 ; TEXT CHARACTER, LOAD TO BUFFER
182 ;
4450 EB 182 NOTPC: XCHG ; POINT TO OUTPUT BUFFER
4451 77 184 MOV M, A ; LOAD CHARACTER TO BUFFER
4452 23 185 INX H ; INCREMENT BUFFER POINTER
4453 C33244 186 JMP NXTCN ; GET NEXT CHARACTER
187 ;
188;
189 ;********************************************************************************
190 ;
191 ; SUBROUTINE TO PRINT TEXT OR NUMERIC DATA
192 ;
193 ; INPUT-
194 ; D- CHARACTER TO COMPARE TO TERMINATE PRINT
195 ; H & L- OUTPUT BUFFER
196 ;
4456 197 OUTPT SET $ ;
4456 7E 198 MOV A, M ; GET CHARACTER
4457 BA 199 CMP D ; IS IT END CHARACTER?
4458 C8 200 RZ ; YES, FINISHED
201 ;
202 ; OUTPUT CHARACTER TO TELETYPING USING MONITOR ROUTINE
203 ;
4459 4F 204 MOV C, A ; PREPARE FOR OUTPUT
445A CDF901 205 CALL ECHO ; OUTPUT CHARACTER
445D 23 206 INX H ; POINT TO NEXT CHARACTER
445E C35644 207 JMP OUTPT ; GET NEXT CHARACTER
208 ;
209 ;********************************************************************************
210 ;
211 ;
212 CNASI EQU 0ADFH
213 BINOC EQU 15E5H
214 ;
215 ; TEST NORMAL MESSAGE WITH ANSWER
216 ;
4461 217 START SET $.
L0C OBJ SEQ SOURCE STATEMENT
4461 3E1E 218 MVI A, 30 ; SET BUFFER SIZE
4463 11D144 219 LXI D, OUTB1 ; POINT TO BUFFER
4466 213845 220 LXI H, HEAD1 ; HEADER MESSAGE
4469 CD0044 221 CALL PRDAT ; PRINT IT
446C 3E1E 222 MVI A, 30 ; RESET BUFFER SIZE
446E 11D144 223 LXI D, OUTB1 ; POINT TO BUFFER
4471 215045 224 LXI H, STEP1 ; STEP MESSAGE
4474 CD0044 225 CALL PRDAT ; PRINT IT
4477 3E1E 226 MVI A, 30 ; SET BUFFER SIZE AT 30 BYTE
4479 CD9E45 227 CALL DOIT ; LOAD COUNT IN ANSWER BUFFER
447C 11D144 228 LXI D, OUTB1 ; POINT TO BUFFER
447F 210045 229 LXI H, MESS1 ; POINT TO MESSAGE
4482 CD0044 230 CALL PRDAT ; PRINT MESSAGE AND ANSWER
231 ; TEST TRUNCATION OF MESSAGE
232 ;
4485 3E1E 234 MVI A, 30 ; SET BUFFER SIZE
4487 11D144 235 LXI D, OUTB1 ; POINT TO BUFFER
448A 216345 236 LXI H, STEP2 ; POINT TO MESSAGE
448D CD0044 237 CALL PRDAT ; PRINT IT
4490 3E0A 238 MVI A, 10 ; SET BUFFER LENGTH AT 10 BY
4492 CD9E45 239 CALL DOIT ; PUT COUNT IN ANSWER
4495 11E544 240 LXI D, OUTB2 ; POINT TO SHORT BUFFER
4498 210045 241 LXI H, MESS1 ; POINT TO MESSAGE
449B CD0044 242 CALL PRDAT ; PRINT MESSAGE
243 ; TEST MESSAGE WITHOUT ANSWER
244 ;
449E 3E1E 246 MVI A, 30 ; SET BUFFER SIZE
44A0 11D144 247 LXI D, OUTB1 ; POINT TO BUFFER
44A3 217445 248 LXI H, STEP3 ; POINT TO MESSAGE
44A6 CD0044 249 CALL PRDAT ; PRINT IT
44A9 3E1E 250 MVI A, 30 ; SET BUFFER AT 30 BYTES
44AB CD9E45 251 CALL DOIT ; LOAD COUNT IN ANSWER
44AE 11D144 252 LXI D, OUTB1 ; POINT TO LONG BUFFER
44B1 211345 253 LXI H, MESS2 ; POINT TO MESSAGE
44B4 CD0044 254 CALL PRDAT ; PRINT IT
255 ; TEST TRUNCATING EXPANSION
256 ;
44B7 3E1E 258 MVI A, 30 ; SET BUFFER SIZE
44BA 11D144 259 LXI D, OUTB1 ; POINT TO BUFFER
44BC 213945 260 LXI H, STEP4 ; POINT TO MESSAGE
44BF CD0044 261 CALL PRDAT ; PRINT IT
44C2 3E1E 262 MVI A, 30 ; SET BUFFER TO 30 BYTES
44C4 CD9E45 263 CALL DOIT ; LOAD COUNT IN ANSWER
44C7 11D144 264 LXI D, OUTB1 ; POINT TO LONG BUFFER
44CA 212445 265 LXI H, MESS3 ; POINT TO MESSAGE
44CD CD0044 266 CALL PRDAT ; PRINT IT
267 ;
268 ; END OF TEST
269 ;
44D0 CF 270 RST 1
271 ;
6/78 272 ; BUFFER AREAS

4-586
LOC OBJ  SEQ SOURCE STATEMENT

273 ;
44D1 0014 274 OUTB1 SET $  
44E5 000A 275 DS 20  
44EF 0011 276 OUTB2 SET $  
280 ;
281 ; TEXTS  
282 ;
4500 0D 283 MESS1: DB 00H ; CARRIAGE RETURN  
4501 02 284 DB 02H ; 2 SPACES  
4502 42554646 285 DB 'BUFFER LENGTH ='  
4506 4552204C  
450A 454E4754  
450E 48203D  
4511 02 286 DB 02H ; 2 SPACES  
4512 FF 287 DB 0FFH ; PRINT TEXT AND ANSWER  
4513 0D 288 MESS2: DB 00H ; CARRIAGE RETURN  
4514 04 289 DB 04H ; 4 SPACES  
4515 57495448 290 DB 'WITHOUT ANSWER'  
4519 4F555420  
451D 414E5357  
4521 4552  
4523 FE 291 DB 0FEH ; TEXT ONLY  
4524 0D 292 MESS3: DB 00H ; CARRIAGE RETURN  
4525 02 293 DB 02H ; 2 SPACES  
4526 5452554E 294 DB 'TRUNCATE SPACES'  
452A 43415445  
452E 20535041  
4532 434553 295 DB 0AH ; 10 SPACES  
4536 0C 296 DB 0CH ; 12 SPACES  
4537 FF 297 DB 0FFH ; TEXT AND ANSWER  
4538 0D 298 HEAD1: DB 00H ; CARRIAGE RETURN  
4539 06 299 DB 06H ; 6 SPACES  
453A 54455354 300 DB 'TEST OF PRDAT ROUTINE'  
453E 204F4620  
4542 50524441  
4546 5420524F  
454A 5554494E  
454E 45  
454F FE 301 DB 0FEH ; TEXT ONLY  
4550 0D 302 STEP1: DB 0DH ; CARRIAGE RETURN  
4551 20544558 303 DB 'TEXT PLUS ANSWER'  
4555 54205040  
4559 55532041  
455D 4E535745  
4561 52  
4562 FE 304 DB 0FEH ; TEXT ONLY  
4563 0D 305 STEP2: DB 0DH ; CARRIAGE RETURN  
4564 20545255 306 DB 'TRUNCATED TEXT'  
4568 4E434154  
456C 45442054  
4570 455854  
4-587
LOC OBJ SE0 SOURCE STATEMENT
4573 FE 307 DB 0FEH ; TEXT ONLY
4574 0D 308 STEP3: DB 0DH ; CARRIAGE RETURN
4575 20544558 309 DB ' TEXT WITHOUT ANSWER'
4579 54205749
457D 54484F55
4581 5420414E
4585 53574552
4589 FE 310 DB 0FEH ; TEXT ONLY
458A 0D 311 STEP4: DB 0DH ; CARRIAGE RETURN
458B 20545255 312 DB ' TRUNCATED SPACING'
458F 4E344154
4593 45442053
4597 50414349
459B 4E47
459D FE 313 DB 0FEH ; TEXT ONLY
314 ; 315 ; ROUTINE TO LOAD COUNT INTO ANSWER BUFFER
316 ;
459E 55 317 DOIT: PUSH PSW ; SAVE ACCUM
459F 5F 318 MOV E,A ; BINARY COUNT
45A0 1600 319 MVI D,00H ; SET FOR CONVERSION
45A2 CDE515 320 CALL BINBC ; CONVERT TO BCD
45A5 7D 321 MOV A,L ; GET BCD COUNT
45A6 21EF44 322 LXI H,ANSW ; POINT TO ANSWER BUFFER
45A9 CDF0A 323 CALL CNASI ; CONVERT COUNT TO ASCII IN B&C
45AC 70 324 MOV M,B ; STORE FIRST DIGIT
45AD 23 325 INX H ; MOVE POINTER
45AE 71 326 MOV M,C ; STORE SECOND DIGIT
45AF 23 327 INX H ; MOVE POINTER
45B0 3620 328 MVI M,20H ; SET SPACE FOR END CHARACTER
45B2 F1 329 POP PSW ; RETRIEVE COUNT
45B3 4F 330 MOV C,A ; PUT COUNT INTO REG C
45B4 C9 331 RET
332 ;
333 ; END OF PROGRAM
334 ;
335 ; END

PUBLIC SYMBOLS

EXTERNAL SYMBOLS

USER SYMBOLS
ANSWR A 44EF BINBC A 15E5 CNASI A 0ADF DOIT A 459E ECHO A 01F
GETCH A 441D HEAD1 A 4538 LDXT A 441C LODCH A 443B LOOPA A 444
MESS3 A 4524 NOTPC A 4450 NXTCH A 4432 OUTB1 A 44D1 OUTB2 A 44E
SPACE A 4426 START A 4461 STEP1 A 4550 STEP2 A 4563 STEP3 A 457

ASSEMBLY COMPLETE, NO ERRORS

6/78

4-588
MON805 -- Monitor for iSBC 80/05 or 80/04

2K byte debug monitor for Intel 80/05 and 80/04. Provides simple memory and register display and modification commands as well as program execution with breakpoints. In addition, the monitor supports paper tape I/O if a TTY unit is connected as the console device.

Intel iSBC 80/05 or 80/04 with console CRT or TTY (TTY required for paper tape I/O).

None

Monitor includes rudimentary command line interpreter for operator interface. See SBC 80P05 User's Guide (Manual #9800508A) for further details.

Available on non-system diskette only for $35.00. (source & object code included)

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM Required:</td>
<td>31 bytes &amp; stack</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>1714 bytes</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td>8</td>
</tr>
<tr>
<td>Assembler/Compiler Used:</td>
<td>8080 macroassembler</td>
</tr>
</tbody>
</table>

Programmer: This monitor is supplied in ROM form to customers purchasing Company: the iSBC 80P05. Address: City: State:
MON810 -- Monitor for iSBC 80/10 or 80/10A

2K byte debug monitor for Intel iSBC 80/10 and 80/10A. Provides simple memory and register display and modification commands as well as program execution with breakpoints. In addition, the monitor supports paper tape I/O if a TTY unit is connected as the console device.

Intel iSBC 80/10 or 80/10A with console CRT or TTY (TTY required for paper tape I/O).

none

Monitor includes rudimentary command line interpreter for operator interface. See SBC 80P and 80P10 Prototyping Package User's Guide (manual order #98002230) for further details.

Available on non-system diskette only for $35.00. (source & object code included)

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM Required:</td>
<td>16 + stack usage</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>1374 bytes</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td>~8</td>
</tr>
<tr>
<td>Assembler/Compiler Used:</td>
<td>8080 macroassembler</td>
</tr>
</tbody>
</table>

Programmer: This monitor is supplied in ROM form to customers purchasing the iSBC 80P10 Prototype Package and System 80/10.

Company: Intel Corporation, 1976

© Intel Corporation, 1976
MON820 -- Monitor for iSBC 80/20 or 80/20-4

2K byte debug monitor for Intel iSBC 80/20 and 80/20-4. Provides simple memory and register display and modification commands as well as program execution with breakpoints and single stepping. In addition, the monitor supports paper tape I/O if a TTY unit is connected as the console device.

Intel iSBC 80/20 or 80/20-4 with console CRT or TTY (TTY required for paper tape I/O)

none


Available on non-system diskette only for $35.00. (source & object code included)

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM Required:</td>
<td>45 + stack</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>1708</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td>~ 8</td>
</tr>
<tr>
<td>Assembler/Compiler Used:</td>
<td>8080 macroassembler</td>
</tr>
<tr>
<td>Programmer:</td>
<td>This monitor is provided in ROM form to customers purchasing Company: the iSBC 80P20 prototype package and system 80/20-4.</td>
</tr>
<tr>
<td>Address:</td>
<td></td>
</tr>
<tr>
<td>City:</td>
<td></td>
</tr>
<tr>
<td>State:</td>
<td></td>
</tr>
</tbody>
</table>
INPUT/OUTPUT DIAGNOSTIC 8080

Allows interactive testing of any MDS I/O port(s). Also allows saving and reloading the test program.

MDS System with Disc

MDS ISIS System

See Command Summary

Program available on diskette only.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL</td>
<td>S. G. Thompson</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RAM Required:</th>
<th>Company:</th>
</tr>
</thead>
<tbody>
<tr>
<td>2340 Decimal</td>
<td>Harris Controls</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ROM Required:</th>
<th>Address:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P.O. Box 430</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Maximum Subroutine Nesting Level:</th>
<th>City:</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>Melbourne</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Assembler/Compiler Used:</th>
<th>State:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel V1.1</td>
<td>Florida 32901</td>
</tr>
</tbody>
</table>
The following commands are valid when the diagnostic is loaded and has typed the sign-on message: 'I/O test V1.1' and prompted the operator with 'M:'. Abbreviations for each command are in parentheses.

CLR (CL)
- Clears the Input/Output buffer of all commands.

EDIT N (ED)
- Edits the Input/Output buffer starting at line number 'N'.
  Valid inputs are:
  IN YY  input data from port 'YY'
  (I)
  OUT XX, YY  output data 'XX' to port 'YY'
  (O)
  DELT  delete line entry
  (D)
  X  exit from editor
  CR  (carriage return) steps to next line, the present line is not altered

EXIT (EX)
- Exit to the MDS-800 monitor

INIT
- Initialize the one bit flag variables and 16 bit variables as follows:
  FS = 1  format switch on
  RS = 0  repeat switch off
  HS = 0  home cursor switch off
  IC = 32_{16}  inter character time of 50_{10} ms
  IM = 0  inter message time of 0_{10} ms

IN YY
- Inputs and displays the data received from port 'YY'.

LIST XX (LI)
- List the Input/Output buffer starting with line 'XX'. If the line # is missing then the listing starts at line number 0.

LOAD NAME (LO)
- Load the previously saved diagnostic Input/Output buffer saved under the disk name 'NAME'. 'NAME' may have the following formats:
  :F1:STEVE.TST  as :drive#:Filename.extension
  STEVE  as Filename (assumes drive 0)
  DIAG.00X  as Filename.extension
  DIAG  as Filename (assumes drive 0)

OUT XX, YY
- The data specified by 'XX' is output to port 'YY'.

SAVE NAME
- Save the present diagnostic input/output buffer to disk under the name 'NAME'. 'NAME' may have the formats previously listed for the load command.

SV NAME ZZZZ
- Set the Variables named 'NAME' to the value 'ZZZZ'.

SEND XX, YY (SE)
- Send the I/O commands in the I/O buffer starting at line number XX through line number YY.

TA
- Type out All the values of the variables.
The following variables are used in conjunction with the 'SEND' command:

FS - Format Switch if 'one' causes the data received from any IN commands to be printed on the operator interaction device.

HS - Home Switch if 'one' causes the control codes for clear screen and home to be executed by the output formatter so that one CRT page of data will be repetitively displayed.

RS - Repeat Switch if 'one' causes the I/O buffer to be sent repeatedly until a break character (STX = control B) is entered by the operator.

IC - Inter Character timing value. The time value is used between each I/O command. The units are 1/1000 of a second.

IM - Inter Message timing value. The time value used between each repetition of the I/O buffer (assumes RS=1). The units are 1/1000 of a second.
.EXTRCT

Extracts selected lines from an ISIS source or print file

MDS, Floppy Disk

ISIS

1) Command Line:
   EXTRCT source file TO destination file

2) Range of Lines:
   XXXX,YYYY

3) End of Operation:
   END

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>N/A</th>
<th>Programmer:</th>
<th>Ken Norris</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM Required:</td>
<td>37AH</td>
<td>Company:</td>
<td>Daniel Systems</td>
</tr>
<tr>
<td>ROM Required:</td>
<td></td>
<td>Address:</td>
<td>9525 Katy Frwy.</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td>N/A</td>
<td>City:</td>
<td>Houston</td>
</tr>
<tr>
<td>Assembler/Compiler Used:</td>
<td>8080/8085 Macro Assembler V2.0</td>
<td>State:</td>
<td>Texas 77024</td>
</tr>
</tbody>
</table>
NAME: EXTRACT

STRLN 40H

; THIS PROGRAM EXTRACTS LINES FROM A SOURCE FILE

; COMMAND SEQUENCE:
; EXTRACT SOURCE FILE TO DESTINATION FILE

; PROGRAM WILL PROMPT CONSOLE WITH '*'.
; RESPOND WITH: LINA_LINEB(CR).
; LINES FROM LINEA TO LINEB WILL BE MOVED TO DESTINATION FILE
; REPEAT UNTIL ALL DESIRED SEGMENTS HAVE BEEN MOVED, THEN
; RESPOND WITH: END(CR).
; (DO NOT ATTEMPT TO BACK UP, RESULTS ARE UNDEFINED.)
; PROGRAM WILL RETURN TO ISIS.

OPEN EQU 0
READ EQU 3
WRITE EQU 4
EXIT EQU 9
ERROR EQU 12
EXTRAN ISIS

; MACRO DEFS
SYSTEM MACRO @TYPE,@BLOCK
HVI C,@TYPE
LXI D,@BLOCK
CALL ISIS
LOA STATUS
ORA A
JNZ ERR
ENDM

MOVE MACRO @TO
LXI D,@TO
CALL MOVEH
ENDM

TEST MACRO @HL,@H2
LXI H,@HL
LXI D,@H2
CALL TESTH
ENDM

CHECK MACRO @dest
LXI H,@dest
CALL CONMD
ENDM

CONFR MACRO @L,@2
<table>
<thead>
<tr>
<th>LOC</th>
<th>OBJ</th>
<th>SEQ</th>
<th>SOURCE STATEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>0000</td>
<td>S</td>
<td>BEGIN: LXI SP,STACK ; INITIALIZE POINTERS</td>
</tr>
<tr>
<td>0003</td>
<td>0000</td>
<td>62</td>
<td>LXI H,0</td>
</tr>
<tr>
<td>0006</td>
<td>0000</td>
<td>D</td>
<td>SHLD IPR</td>
</tr>
<tr>
<td>0018</td>
<td>216800</td>
<td>D</td>
<td>LXI H,CBUFF</td>
</tr>
<tr>
<td>0018</td>
<td>216800</td>
<td>D</td>
<td>MOVE IFILE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>D</td>
<td>MOVE DUMMY</td>
</tr>
<tr>
<td></td>
<td></td>
<td>D</td>
<td>MOVE OFILE</td>
</tr>
<tr>
<td>0036</td>
<td>C2E000</td>
<td>C</td>
<td>TEST DUMMY,ATOT ; MIDDLE WORD 'TO'?</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0084</td>
<td>3A0000</td>
<td>D</td>
<td>LDR ACTUAL ; EOT?</td>
</tr>
<tr>
<td>0087</td>
<td>B7</td>
<td>123</td>
<td>ORR A</td>
</tr>
<tr>
<td>0088</td>
<td>CAD100</td>
<td>C</td>
<td>JZ DONE</td>
</tr>
<tr>
<td>0094</td>
<td>DAR100</td>
<td>C</td>
<td>TEST OFEND</td>
</tr>
<tr>
<td>0097</td>
<td>116800</td>
<td>D</td>
<td>LXI D,CBUFF ; NO, TEST &amp; CONVERT TO DECIMAL</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CHECK PARML ; START OF RANGE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>JNC SERR ; BAD</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CHECK PARML2 ; END OF RANGE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>JNC SERR ; COMPARE PARML2,PARML1 ; TEST PARML1&lt;PARML2</td>
</tr>
<tr>
<td>0086</td>
<td>FAD000</td>
<td>C</td>
<td>JMI SERR ; NO</td>
</tr>
<tr>
<td>0089</td>
<td>CD6801</td>
<td>C</td>
<td>CALL SKIPE ; YES, SKIP TO START OF RANGE ON INPUT FILE</td>
</tr>
<tr>
<td>008C</td>
<td>CD6801</td>
<td>C</td>
<td>CALL SAVE ; COPY FROM HERE TO END OF RANGE</td>
</tr>
<tr>
<td>008F</td>
<td>C36600</td>
<td>C</td>
<td>JMP LOOP ; CONTINUE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ERR : SYSTEM ERROR.EBLK ; SYSTEM ERROR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DONE : SYSTEM EXIT.EBLK ; RETURN TO SYSTEM</td>
</tr>
<tr>
<td>00E0</td>
<td>3ECA</td>
<td></td>
<td>SERR : MVI R.202 ; SYNTAX ERROR</td>
</tr>
<tr>
<td>00E2</td>
<td>320102</td>
<td>C</td>
<td>STA STATUS</td>
</tr>
<tr>
<td>00E5</td>
<td>3C2000</td>
<td>C</td>
<td>JMP ERR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00E8</td>
<td>3E00</td>
<td></td>
<td>MOVEM : MVI R.00H ; MOVES WORD POINTED TO BY HLU TO BUFFER</td>
</tr>
<tr>
<td>00ER</td>
<td>12</td>
<td></td>
<td>STAX D ; POINTED TO BY DATE</td>
</tr>
<tr>
<td>00EB</td>
<td>7E</td>
<td></td>
<td>MOVMOV ; MOV R.M ; TERMINATES ON TRAILING COMMA, SPACE</td>
</tr>
<tr>
<td>00EC</td>
<td>23</td>
<td></td>
<td>INK H ; SLASH OR CR. IGNORES LEADING COMMA</td>
</tr>
<tr>
<td>00ED</td>
<td>FE0D</td>
<td></td>
<td>CPI 00H ; SPACE OR SLASH TERMINATOR RETURNED</td>
</tr>
<tr>
<td>00EF</td>
<td>C2F600</td>
<td>C</td>
<td>JNZ MOV10 ; IN A-REG</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MVI R.0FH</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ORA A ; A-REG MINUS ON RET =&gt; NO PARML</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RET</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CALL VALDL ; LEADING TERM?</td>
</tr>
</tbody>
</table>
0145 09  231  DAD    B                  ;*10
0147 4F  232  MOV    C,A
0148 0500 233  MVI    B,0
0149 09  234  DAD    B                  ;+ NEW DIGIT
0148 C3201 C 255  JMP    CON05
014E 11  256  CON10,  POP    B
014F 7D  257  MOV    A,L
0150 02  258  STAX    B
0151 03  259  INX    B
0152 7C  240  MOV    A,H
0153 02  241  STAX    B
0154 C32501 C 242  JMP    GOOD                  ;SUCCESS
0157 E1  243  CON15,  POP    H                  ;ERROR
0158 C32701 C 244  JMP    BAD
0159 DE30 246  VALDG, SUI '0'                  ;VALID DIGIT TEST(CONVERT TO BIN)
015A FA701 C 247  JM    BAD
015B F080 248  CPI    10
015C F22701 C 249  JP    BAD                  ;ERROR
015E C32501 C 250  JMP    GOOD                  ;OK NOW
0162 245  251  SKIP:   COMPR    IPTR,PARM1                ;READ TO START OF RANGE
0172 D8  257  RC    OK
0173 2A0200 D 258  SYSTEM READ,IBLK                  ;NOT YET
017C 200200 D 265  LHL    IPTR
0180 23  266  INX    H
0184 C36001 C 267  SHLD    IPTR
0188 C36001 C 268  JMP    SKIP
0189 269  270  SAVE:   SYSTEM WRITE,OBLK                  ;SAVE THIS RECORD
0195 D8  277  COMPR    IPTR,PARM2                ;SAVED TO END OF RANGE YET?
0199 290200 D 282  RC    YES
019D 29A200 D 283  SYSTEM READ,IBLK                  ;NO, GET NEXT
019F 222200 D 290  LHL    IPTR
01B8 23  291  INX    H
01BC C30001 C 292  SHLD    IPTR
01C4 293  294  JMP    SAVE
01C6 295  296  :CONTROL BLOCKS
01CF 18000 D 296  CBBLK:   DW    1,CBUFF,128,ACTION,STATUS                  ;CONSOLE INPUT BLOCK
01C1 68000 D
01C3 0000
01C5 0000 D
01C7 0002 C
01C9 F101 C 297  00BLK:   DW    OAF,OF,IFILE,2,0,STATUS                  ;OPEN OUTPUT BLOCK
01CB 48000 D
01CD 0000
01CF 0000
01D1 0002 C
01D3 E701 C 298  01BLK:   DW    IAF,IFILE,1                  ;OPEN INPUT BLOCK
01D5 0000 D
01D7 0100
01D9 0000 299  BAFT:   DW    0 STATUS
01DB 0002 C
01DD 0901 C 300  BBLK:   DW    BAFT,OB,OF,IFILE,2,0,STATUS                  ;OPEN :BB: BLOCK
MOV15 C 00FE MOVE + 0001 MOVEN C 00E8 OAFT C 01F1 OBLK C 01F1 OFILE D 0048 O1BLK C 01D3
O0BLK C 01C9 OPEN A 0000 Parm1 D 0004 Parm2 D 0006 READ A 0003 SAVE C 018C SBLK C 01F9
SERR C 00E8 SKIP C 0168 STATUS C 0201 SYSTEM + 0000 TEST + 0002 TESTW C 0115 VALDG C 015B
VALDL C 0109 WRITE A 0004 XBLK C 0203

ASSEMBLY COMPLETE, NO ERRORS
**Program Title**: BINARY PUNCH TAPE FOR PROM PROGRAMMER - PUNCH

**Function**: MDS-800, INTELLEC DISKETTE OPERATING SYSTEM, TTY
OPTIONAL: HIGH SPEED PUNCH

**Required Software**: ISIS-II

**Input Parameters**: SEE ATTACHMENT

**Output Results**: SEE ATTACHMENT

<table>
<thead>
<tr>
<th>Registers Modified</th>
<th>Programmer</th>
<th>Company</th>
<th>Address</th>
<th>City</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL</td>
<td>JOHN P. NAGEL</td>
<td>SINGER-KEARFOTT</td>
<td>150 TOTOWA ROAD</td>
<td>WAYNE</td>
<td>NEW JERSEY, 07470</td>
</tr>
<tr>
<td>RAM Required:</td>
<td>32K</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ROM Required:</td>
<td>MDS MONITOR</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td>3 LEVELS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Assembler/Compiler Used:</td>
<td>ASM-80</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
BINAR Y PUNCH PROGRAM

Example of Use:

An 8K program starting at location 0000H has been written and debugged and resides on the non-system disk, and is named SAMPLE.OBJ. A User wants to punch 8, 1K paper tapes to be used for programming 8, 1K PROMS. The Binary Punch Program is named PUNCH and resides on the system disk.

The paper tape is punched as follows:

1) In ISIS-II mode, the User types the following command:

   - PUNCH :F1: SAMPLE.OBJ (CR)

2) The Program will respond with the following question:

   Data Length in HEX 100, 200, 400, 800?

   The User should type in 400 (CR)

3) The Program will respond with the following question:

   Start Address in HEX?

   The User should type in Ø (CR)

4) The Program will respond with the following question:

   Number of tapes?

   The User should type in 8 (CR)

5) The Paper Tape Punch will punch 8 tapes with a leader and a trailer and with 256 nulls inbetween each tape. Then the ISIS-II prompt will appear on the console.
HEXSYM

Converts absolute object code in Hex-file-format to an object module in OBJ-format with symbol table.

MDS-800 (with DOS)

ISIS-II, Console commands HEXOBJ, COPY, DELETE

The program HEXSYM must be used with the following sequence of commands:

```
HEXOBJ filename1 TO HHH START(addr)
HEXSYM filename1
COPY H1,H2,H3 TO filename2
DELETE H? , HHH
```

filename1 - file with object code in hex-file format
filename2 - converted file in absolute OBJ-file-format with a symbol table
H1,H2,H3 and HHH are auxiliary files

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL</td>
<td>Macha Erich (E52 E32)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RAM Required:</th>
<th>Company:</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISIS-dependent</td>
<td>AEG Telefunken</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ROM Required:</th>
<th>Address:</th>
</tr>
</thead>
<tbody>
<tr>
<td>--</td>
<td>Steinheimer Str. 117</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Maximum Subroutine Nesting Level:</th>
<th>City:</th>
</tr>
</thead>
<tbody>
<tr>
<td>--</td>
<td>D-6453 Seligenstadt</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Assembler/Compiler Used:</th>
<th>State:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intellec MDS Macro Assembler V. 2.0</td>
<td>WESTERN GERMANY</td>
</tr>
</tbody>
</table>
LOC OBJ  SEQ  SOURCE STATEMENT

1: TITLE 'RETRIEVE'
2;
3: A.D. SHORT  5TH JANUARY 1978
4;
5: WHEN USING THE MDS TEXT EDITOR (NO DOS) IT IS POSSIBLE FOR
6: TEXT TO BE LOST IF
7: A) THE INTERRUPT 0 SWITCH IS OPERATED INADVERTENTLY OR
8: B) AN 'E' IS ENTERED IN A COMMAND WHEN THE DESIGNATED
9: SYSTEM PUNCH IS NOT AT THE TIME AVAILABLE IN WHICH
10: CASE THE INTERRUPT 0 SWITCH MUST BE OPERATED
11: DELIBERATELY TO ESCAPE.
12: TEXT IS NORMALLY LOST FOR GOOD.
13: THIS PROGRAM EXTRACTS TEXT FROM THE TEXT BUFFER MEMORY AREA
14: AND OUTPUTS IT VIA THE DESIGNATED SYSTEM PUNCH IN THE CORRECT
15: FORMAT FOR RE-ENTERING IT TO THE TEXT EDITOR USING THE 'A'
16: COMMAND.
17: IF TEXT HAS BEEN OVERWRITTEN THE PROGRAM WILL EXIT WITH
18: 'ALL LOST' DISPLAYED ON THE CONSOLE DEVICE IF OK THEN
19: 'ALL RECORDED' WILL BE DISPLAYED. A CHECK IS NEVERTHLESS
20: ADVISABLE SINCE PART OF THE RECORDED TEXT MAY BE CORRUPT
21: IF OTHER PROCEDURES ARE TRIED BEFORE THIS PROGRAM IS RUN.
22:

0000 23  BEGND  EQU  0FFH  ;START WORD OF TEXT BUFFER
0026 24  BUFLS  EQU  0096H  ;BUFFER LENGTH STORE
F000 25  CO  EQU  0F803H  ;MONITOR CONSOLE O/P ROUTINE
0000 26  CR  EQU  00H  ;ASCII CARRIAGE RETURN CHAR
0000 27  EOU  EQU  1AH  ;END OF FILE MARK/CONTROL Z
0004 28  LF  EQU  00H  ;ASCII LINE FEED CHAR
F00C 29  PO  EQU  0F80CH  ;MONITOR PUNCH O/P ROUTINE
0000 30  START  EQU  0000H  ;BEGINNING OF TEXT BUFFER

31;
0020 32  ORG  28H
33;
0020 34 318400 34  LXI  SP,STAK  ;LOAD STACK POINTER
0023 35 219000 35  LXI  H:BUFLS  ;LOAD ADDRESS OF BUFFER LENGTH STORE
0026 36 5EH 36  MOV  E,M  ;LOAD LS BYTE OF BUFFER LENGTH WORD
0027 37 56 37  MOV  D,M  ;LOAD MS BYTE OF BUFFER LENGTH WORD
0029 38 21000E 38  LXI  H:START  ;LOAD ADDR OF BEGINNING OF TEXT BUFFER
002C 39 7E 39  MOV  A,M  ;GET FIRST WORD
002D 41 4EFF 41  CPI  BEGND  ;IS IT THE START WORD?
002F 42 C24700 42  INZ  ERROR  ;NO-ALL LOST 50 BRANCH FOR FAILURE
0032 43 23 43  AGAIN:  INX  H  ;ELSE ADDRESS NEXT WORD
0033 44 1B 44  DCH  D  ;AND DECR BUFFER LENGTH
0034 45 7A 45  MOV  A,D
0035 46 0E00 46  CPI  0
0037 47 C24000 47  JNZ  READ  ;IF MS BYTE >0 BRANCH FOR READ
003A 48 78 48  MOV  A,E
003B 49 FE00 49  CPI  0
003D 4A C06000 4A  JZ  FINIS  ;IF LS BYTE=0 THEN BRANCH TO END
0040 4E 5E 51  READ  MOV  C,M  ;GET NEXT CHARACTER
0041 52 C00CF8 52  CALL  PO  ;OUTPUT IT TO SYSTEM PUNCH

8/78
LOC OBJ SEQ SOURCE STATEMENT

0044 C33200 53 JMP AGAIN ; AND SEE ABOUT ANOTHER CHARACTER
0047 216800 54 ERROR: LXI H,MES1 ; LOAD ADDRESS OF ERROR MESSAGE
004A 000A 55 MVI B,LMES1 ; LOAD MESSAGE LENGTH
004C D0E00 56 CALL MSLG ; OUTPUT MESSAGE
004F C7 57 RST 0 ; AND RETURN TO MONITOR
0050 0E1A 58 FINIS: MVI C, EOF ; LOAD END OF FILE CHARACTER
0052 CD0CF8 59 CALL PO ; AND OUTPUT IT TO SYSTEM PUNCH
0055 217200 60 LXI H,MES2 ; LOAD ADDRESS OF SUCCESS MESSAGE
0058 060E 61 MVI B,LMES2 ; LOAD MESSAGE LENGTH
005A D0500 62 CALL MSLG ; OUTPUT MESSAGE
005D C7 63 RST 0 ; AND RETURN TO MONITOR
0060 4E 64 MSLG: MOV C, M ; LOAD MESSAGE CHARACTER
0063 DF9F8 65 CALL CO ; AND OUTPUT IT TO CONSOLE
0062 05 66 DCR B ; DECREMENT CHARACTER COUNT
0063 C8 67 RZ ; RETURN IF ZERO=ALL DONE
0064 23 68 INX H ; ELSE SET FOR NEXT CHARACTER
0065 C5E00 69 JMP MSLG ; AND LOOP
0068 414C4C20 70 MES1: DB 'ALL LOST', CR, LF
006C 4C4F5354
0070 00
0071 0A
0074 71 LMES1 EQU -$MES1
0072 414C4C20 72 MES2: DB 'ALL RECORDED', CR, LF
0076 5245434F
0079 52444544
007C 0D
007F 0A
0080 73 LMES2 EQU -$MES2
0084 76 STAK EQU $

PUBLIC SYMBOLS

EXTERNAL SYMBOLS

8/78
USER SYMBOLS

AGAIN A 0032 BEGND A 00FF BLANK A 0088 BURLS A 0096 CO A F899 CR A 0000 EDF A 001A
ERROR A 0047 FINIS A 0050 LF A 000A LMES1 A 000A LMES2 A 000E MES1 A 0068 MES2 A 0072
MSGL A 005E PO A F80C READ A 0040 STACK A 0004 START A 0000

ASSEMBLY COMPLETE, NO ERRORS
CALIBRATION OF AN EXTERNAL INPUT DEVICE (EID) FOR MAPPING WITH THE SURFACE OF A CRT TERMINAL

This program has two independent parts. The first part initializes the CRT screen for the calibration procedure by displaying the active EID-area and marks where to calibrate the screen (EIDCAL-IN). The second part (EIDCALMA) of the calibration procedure collects the EID data of calibration points. The received data are transformed to the screen coordinate system and converted from BCD to binary. Now each coordinate pair is checked to determine if its X-coordinate is in the approximate area of the calibration spots. After the valid data of the last spot is accepted, the following refinements and computations are done: (1) left and right boundary of area (2) bottom and top boundary of area (3) area increments in X and Y axis (4) screen character per TSD digit in X and Y axis.

Fixed point package.
BCD - binary conversion

EID - digits from the EID-service-routine.

Computations:
Left and right boundary of device area.
Bottom and top boundary of device area.
Screen characters per EID-digit.

Flags: Device calibrated.

AB128A is offered as one program with AB128B.
TRANSFORMATION ROUTINE FOR COORDINATE DIGITS FROM AN EXTERNAL INPUT DEVICE (EID) INTO X- Y- COORDINATES OF A CRT TERMINAL

This program supports a hardware arrangement of an external input device, which produces coordinate digits. The coordinates are displayed on a CRT terminal. The significant steps are: Coordinate origin transformation, BCD to binary conversion. Limit checking. Computations. Preparation for cursor positioning.

Required Hardware

Fixed point package
BCD to binary conversion

Required Software

X and Y digits from the service routine of the EID.
Output of the calibration procedure (see separate program).

Input Parameters

Output Results

X and Y coordinates of the CRT terminal screen
Flag, which indicates that the input device was activated.
Flag, which requests to position the cursor at the calculated spot.
Data to position the cursor.

AB128B is offered as one program with AB128A.

Registers Modified:

Programmer: Dr. Theodor Luetzeler

RAM Required: 5A H

Company: Siemens Corporation

ROM Required: 364 H

Address: 3 Computer Drive

Maximum Subroutine Nesting Level: 6 H

City: Cherry Hill

Assembler/Compiler Used: PLM 80

State: New Jersey

© Intel Corporation, 1976
CHECK SUM

This program calculates two verification digits for data string until 1K bytes, and types them out on the console output device. It uses the ISIS-II system calls and it's invoked by the command CHKSUM.ABS <FILENAME>, where <FILENAME> is the name of the hexadecimal file that has the data string input. The calculation way is a simple summation of all input bytes, with overflow from the low checksum to the high checksum. The two output digits could be used for ROM check purposes.

Intellec MDS with 32K RAM

ISIS-II operating system

Diskette with CHKSUM.ABS program and with data string input file. CHKSUM.ABS <FILENAME> from the console.

Two hexadecimal digits on the console.

Registers Modified: 

Programmer: DIEGO SÁNCHEZ HERNANDEZ

RAM Required: 

Company: G.E.E. - Electromedicina

ROM Required: 

Address: Doctor Esquerdo, 187

Maximum Subroutine Nesting Level: 

City: MADRID.- 7

Assembler/Compiler Used: 

Assembler: ISIS-II 8080/8085 Macro Assembler V2.0

State: SPAIN

© Intel Corporation, 1976 8/78
LOC 0EJ    SEQ    SOURCE STATEMENT

1:  CHECK SUM
2:  THIS PROGRAM CALCULATES TWO VERIFICATION DIGITS FOR DATA
3:  STRING UNTIL 1K BYTES, AND TYPES OUT THEM ON THE CONSOLE
4:  OUTPUT DEVICE.
5:  IT USES THE ISIS-II SYSTEM CALLS AND IT IS INVOKED BY THE
6:  COMMAND:
7:     CHKSUM ABS <FILENAME>
8:  WHERE <FILENAME> IS THE NAME OF THE HEXADECIMAL FILE THAT
9:  HAS THE DATA STRING INPUT.
10: THE CALCULATION WAY IS A SIMPLE SUMMATION OF ALL INPUT
11: BYTES, WITH OVERFLOW FROM THE LOW CHECKSUM TO THE HIGH
12: CHECKSUM.
13: THE TWO OUTPUT DIGITS COULD BE USED FOR ROM CHECK PURPOSES.
14: MARCH-1978. DIEGO SANCHEZ.
15:
16:
17:
18 #TITLE( "SYMBOLS")
19 #EJECT
<table>
<thead>
<tr>
<th>LOC</th>
<th>OBJ</th>
<th>SCO</th>
<th>SOURCE STATEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>20</td>
<td>EQU</td>
<td>OPEN EQU 0</td>
</tr>
<tr>
<td>0001</td>
<td>21</td>
<td>EQU</td>
<td>CLOSE EQU 1</td>
</tr>
<tr>
<td>0003</td>
<td>22</td>
<td>EQU</td>
<td>READ EQU 3</td>
</tr>
<tr>
<td>0004</td>
<td>23</td>
<td>EQU</td>
<td>WRITE EQU 4</td>
</tr>
<tr>
<td>0009</td>
<td>24</td>
<td>EQU</td>
<td>EXIT EQU 9</td>
</tr>
<tr>
<td>000C</td>
<td>25</td>
<td>EQU</td>
<td>ERROR EQU 12</td>
</tr>
<tr>
<td></td>
<td>26</td>
<td></td>
<td>;</td>
</tr>
<tr>
<td></td>
<td>27</td>
<td></td>
<td>;</td>
</tr>
<tr>
<td></td>
<td>28</td>
<td></td>
<td>;</td>
</tr>
<tr>
<td></td>
<td>29</td>
<td></td>
<td>$TITLE('MAIN PROGRAM')</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td></td>
<td>$EBLECT</td>
</tr>
<tr>
<td>LOC</td>
<td>OBJ</td>
<td>SEG</td>
<td>SOURCE STATEMENT</td>
</tr>
<tr>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>------------------</td>
</tr>
<tr>
<td>31</td>
<td></td>
<td></td>
<td>EXTRN ISIS</td>
</tr>
<tr>
<td>32</td>
<td></td>
<td></td>
<td>; BEGINING OF CODE SEGMENT</td>
</tr>
<tr>
<td>33</td>
<td></td>
<td></td>
<td>GSEG</td>
</tr>
<tr>
<td>34</td>
<td></td>
<td></td>
<td>; STACK INITIALIZATION</td>
</tr>
<tr>
<td>35</td>
<td></td>
<td></td>
<td>CHKSUM: LXI SP, STACK</td>
</tr>
<tr>
<td>36</td>
<td></td>
<td></td>
<td>; READ FILENAME FROM THE :CO:</td>
</tr>
<tr>
<td>37</td>
<td></td>
<td></td>
<td>MOV C, READ</td>
</tr>
<tr>
<td>38</td>
<td></td>
<td></td>
<td>LXI D, RBLK</td>
</tr>
<tr>
<td>39</td>
<td></td>
<td></td>
<td>CALL ISIS</td>
</tr>
<tr>
<td>40</td>
<td></td>
<td></td>
<td>LDA STATUS</td>
</tr>
<tr>
<td>41</td>
<td></td>
<td></td>
<td>ORA A</td>
</tr>
<tr>
<td>42</td>
<td></td>
<td></td>
<td>JNZ ERR</td>
</tr>
<tr>
<td>43</td>
<td></td>
<td></td>
<td>; OPEN THE INPUT FILE</td>
</tr>
<tr>
<td>44</td>
<td></td>
<td></td>
<td>MOV C, OPEN</td>
</tr>
<tr>
<td>45</td>
<td></td>
<td></td>
<td>LXI D, OBLK</td>
</tr>
<tr>
<td>46</td>
<td></td>
<td></td>
<td>CALL ISIS</td>
</tr>
<tr>
<td>47</td>
<td></td>
<td></td>
<td>LDA STATUS</td>
</tr>
<tr>
<td>48</td>
<td></td>
<td></td>
<td>ORA A</td>
</tr>
<tr>
<td>49</td>
<td></td>
<td></td>
<td>JNZ ERR</td>
</tr>
<tr>
<td>50</td>
<td></td>
<td></td>
<td>LHLD AFT</td>
</tr>
<tr>
<td>51</td>
<td></td>
<td></td>
<td>SHLD CRAFT</td>
</tr>
<tr>
<td>52</td>
<td></td>
<td></td>
<td>; READ DE INPUT FILE</td>
</tr>
<tr>
<td>53</td>
<td></td>
<td></td>
<td>MOV C, READ</td>
</tr>
<tr>
<td>54</td>
<td></td>
<td></td>
<td>LXI D, RBLK</td>
</tr>
<tr>
<td>55</td>
<td></td>
<td></td>
<td>CALL ISIS</td>
</tr>
<tr>
<td>56</td>
<td></td>
<td></td>
<td>LDA STATUS</td>
</tr>
<tr>
<td>57</td>
<td></td>
<td></td>
<td>ORA A</td>
</tr>
<tr>
<td>58</td>
<td></td>
<td></td>
<td>JNZ ERR</td>
</tr>
<tr>
<td>59</td>
<td></td>
<td></td>
<td>; BEGINING OF THE PROCESS</td>
</tr>
<tr>
<td>60</td>
<td></td>
<td></td>
<td>MOV D, 0</td>
</tr>
<tr>
<td>61</td>
<td></td>
<td></td>
<td>MOV E, 0</td>
</tr>
<tr>
<td>62</td>
<td></td>
<td></td>
<td>D.E = HIGH+LOW CHKSUM</td>
</tr>
<tr>
<td>63</td>
<td></td>
<td></td>
<td>; D.E = HIGH+LOW CHKSUM</td>
</tr>
<tr>
<td>64</td>
<td></td>
<td></td>
<td>LOOP1: MOV A, A</td>
</tr>
<tr>
<td>65</td>
<td></td>
<td></td>
<td>A = :</td>
</tr>
<tr>
<td>66</td>
<td></td>
<td></td>
<td>LOOP2: CMP M</td>
</tr>
<tr>
<td>67</td>
<td></td>
<td></td>
<td>INX H</td>
</tr>
<tr>
<td>68</td>
<td></td>
<td></td>
<td>JNZ LOOP2</td>
</tr>
<tr>
<td>69</td>
<td></td>
<td></td>
<td>; JUMP 6 BYTES</td>
</tr>
<tr>
<td>70</td>
<td></td>
<td></td>
<td>CALL PACKET</td>
</tr>
<tr>
<td>71</td>
<td></td>
<td></td>
<td>JZ DONE</td>
</tr>
<tr>
<td>72</td>
<td></td>
<td></td>
<td>LXI B, 7</td>
</tr>
<tr>
<td>73</td>
<td></td>
<td></td>
<td>DAD B</td>
</tr>
<tr>
<td>74</td>
<td></td>
<td></td>
<td>MOV B, A</td>
</tr>
<tr>
<td>75</td>
<td></td>
<td></td>
<td>B = COUNTER</td>
</tr>
<tr>
<td>76</td>
<td></td>
<td></td>
<td>SUM: MOV A, A</td>
</tr>
<tr>
<td>77</td>
<td></td>
<td></td>
<td>A = :</td>
</tr>
<tr>
<td>78</td>
<td></td>
<td></td>
<td>ADD E</td>
</tr>
<tr>
<td>79</td>
<td></td>
<td></td>
<td>JNC SUM1</td>
</tr>
<tr>
<td>80</td>
<td></td>
<td></td>
<td>INR D</td>
</tr>
<tr>
<td>81</td>
<td></td>
<td></td>
<td>SUM1: MOV E, A</td>
</tr>
<tr>
<td>82</td>
<td></td>
<td></td>
<td>INX H</td>
</tr>
<tr>
<td>83</td>
<td></td>
<td></td>
<td>DCR B</td>
</tr>
<tr>
<td>84</td>
<td></td>
<td></td>
<td>JNZ SUM</td>
</tr>
<tr>
<td>85</td>
<td></td>
<td></td>
<td>JMP LOOP1</td>
</tr>
</tbody>
</table>
LOC   OBJ     SED     SOURCE STATEMENT
        86 ;
0062 212800 D  87 DONE: LXI H, CHECKN ; STORE CHECK SUM
0065  72     88 MOV M,D ; HIGH CHECK SUM
0066 C06900 C  89 CALL UNPACK
0069  23     90 INX H
006A  73     91 MOV M,E ; LOW CHECK SUM
006B C08900 C  92 CALL UNPACK
        93 ;
006E 0E04    94 MVI C, WRITE ; WRITE TO THE CONSOLE
0070 111400 D  95 LXI D, HBLK
0073 C00000 E  96 CALL ISIS
0076 3A2200 D  97 LDA STATUS
0079  B7     98 ORA A
007A C20000 C  99 JNZ ERR
        100
007E 0E01    101 MVI C, CLOSE ; CLOSE THE INPUT FILE
007F 111000 D  102 LXI D, CBLK
0082 C00000 E  103 CALL ISIS
0085 0E09    104 MVI C, EXIT ; NORMAL EXIT
0087 112000 D  105 LXI D, EBLK
008A C00000 E  106 CALL ISIS
        107 ;
008D 0E0C    108 ERR: MVI C, ERROR
008F 112200 D  109 LXI D, EBLK
0092 C00000 E  110 CALL ISIS
0095 0E09    111 MVI C, EXIT ; ERROR EXIT
0097 112000 D  112 LXI D, EBLK
009A C00000 E  113 CALL ISIS
        114 ;
        115 ;
        116 ;
117 TITLE('PACK')
118 ZEJECT
LOC OBJ SEQ SOURCE STATEMENT

119 ; SUBROUTINE THAT PACKS TWO HEXADECIMAL DIGITS INTO ONE
120 ; BYTE AND PUTS IT IN THE ACCUMULATOR.
121 ; LIKE INPUT PARAMETERS HAL REGISTERS MUST POINT AT THE
122 ; TWO DIGITS, AND THE ACCUMULATOR MUST BE EQUAL TO 3AH.
123 ;
0060 BE 124 PACKET: CM P M ; A = : 0
006E 7E 125 MOV A, H
009F D8400 C 126 JNC NUMBER
00A2 D637 127 SUI 37H ; LETTER
00A4 E60F 128 NUMBER: ANI 0FH
00A6 07 129 RLC
00A7 07 130 RLC
00A8 07 131 RLC
00A9 07 132 RLC
00AA 4F 133 MOV C, A ; C = 4 MSB
00AB 330A 134 MVI A, 3AH ; A = : 0
00AD 23 135 INX H
00AE BE 136 CMP M
00AF 7E 137 MOV A, M
00B0 D2500 C 138 JNC NUMBER2
00B3 D637 140 SUI 37H ; LETTER
00B5 E60F 141 NUMBER2: ANI 0FH
00B7 B1 142 ORA C ; A = PACKET NUMBER
00B8 C9 143 RET
144 ;
145 ;
146 ;
147 $TITLE('UNPACK')$
148 $EJECT$
UNPACK

LOC OBJ    SEQ      SOURCE STATEMENT

149 ; SUBROUTINE THAT UNPACKS ONE BYTE INTO TWO HEXADECIMAL
150 ; DIGITS.
151 ; HIL REGISTERS MUST POINT AT THE INPUT BYTE AND THE
152 ; TWO OUTPUT DIGITS WILL BE IN THE SAME MEMORY POSITION
153 ; AND IN THE NEXT.
154 ;
0089 3EF0 155 UNPACK: MVI 8.0FH
008A A6    156 ANA  M
008B 0F    157 RRC
008C 0F    158 RRC
008D 0F    159 RRC
008E 0F    160 RRC
008F 0F    161 CPI  0AH
00C2 DAD700 C 162 JC  NMBR
00C5 6607 163 ADI  7 ; LETTER
00C7 0630 164 NMBR: ADI 30H
00C9 47    165 MOV  B,A
00CA 3E8F 166 MVI 8.0FH
00CC A6    167 ANA  M
00CD F69A 168 CPI  0AH
00CF DAD400 C 169 JC  NMBR1
00D2 6607 170 ADI  7 ; LETTER
00D4 0630 171 NMBR1: ADI 30H
00D6 70    172 MOV  M,B
00D7 23    173 INX  H
00D8 77    174 MOV  M,A
00D9 C9    175 RET

176 ;
177 ;
178 ;
179 $TITLE('BUFFERS AND DATA')
180 $EJECT
<table>
<thead>
<tr>
<th>LOC</th>
<th>OBJ</th>
<th>SEQ</th>
<th>SOURCE STATEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>181</td>
<td></td>
<td></td>
<td>; BEGINING OF DATA SEGMENT</td>
</tr>
<tr>
<td>182</td>
<td></td>
<td></td>
<td>183 RBLK:</td>
</tr>
<tr>
<td>184</td>
<td></td>
<td></td>
<td>0100 184 AFT: DW 1</td>
</tr>
<tr>
<td>185</td>
<td></td>
<td></td>
<td>2000 D 185 DW BUFFER</td>
</tr>
<tr>
<td>186</td>
<td></td>
<td></td>
<td>0004 000C DW 3072</td>
</tr>
<tr>
<td>187</td>
<td></td>
<td></td>
<td>0006 2000 DW ACTUAL</td>
</tr>
<tr>
<td>188</td>
<td></td>
<td></td>
<td>0008 2000 DW STATUS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>189</td>
<td></td>
</tr>
<tr>
<td>190</td>
<td></td>
<td></td>
<td>000A 0000 DW AFT</td>
</tr>
<tr>
<td>191</td>
<td></td>
<td></td>
<td>000C 2000 DW BUFFER</td>
</tr>
<tr>
<td>192</td>
<td></td>
<td></td>
<td>000E 0300 DW 3</td>
</tr>
<tr>
<td>193</td>
<td></td>
<td></td>
<td>0010 0000 DW 0</td>
</tr>
<tr>
<td>194</td>
<td></td>
<td></td>
<td>0012 2200 DW STATUS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>195</td>
<td></td>
</tr>
<tr>
<td>196</td>
<td></td>
<td></td>
<td>0014 0000 DW 0</td>
</tr>
<tr>
<td>197</td>
<td></td>
<td></td>
<td>0016 200C DW CHECKN</td>
</tr>
<tr>
<td>198</td>
<td></td>
<td></td>
<td>0018 0000 DW 4</td>
</tr>
<tr>
<td>199</td>
<td></td>
<td></td>
<td>001A 2200 DW STATUS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>200</td>
<td></td>
</tr>
<tr>
<td>201</td>
<td></td>
<td></td>
<td>0002 202 CAFT: DS 2</td>
</tr>
<tr>
<td>202</td>
<td></td>
<td></td>
<td>001E 2200 DW 203</td>
</tr>
<tr>
<td></td>
<td></td>
<td>204</td>
<td></td>
</tr>
<tr>
<td>205</td>
<td></td>
<td></td>
<td>0020 2200 DW STATUS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>206</td>
<td></td>
</tr>
<tr>
<td>207</td>
<td></td>
<td></td>
<td>0002 208 EBLK: DW STATUS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>208</td>
<td></td>
</tr>
<tr>
<td>209</td>
<td></td>
<td></td>
<td>0024 2200 DW STATUS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>210</td>
<td></td>
</tr>
<tr>
<td>211</td>
<td></td>
<td></td>
<td>0002 211 ACTUAL: DS 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>212</td>
<td></td>
</tr>
<tr>
<td>213</td>
<td></td>
<td></td>
<td>0000 213 BUFFER: DS 3072</td>
</tr>
<tr>
<td></td>
<td></td>
<td>214</td>
<td></td>
</tr>
<tr>
<td>215</td>
<td></td>
<td></td>
<td>0004 215 CHECKN: DS 4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>216</td>
<td></td>
</tr>
<tr>
<td>217</td>
<td></td>
<td></td>
<td>0000 C 217 END CHKSUM</td>
</tr>
</tbody>
</table>

PUBLIC SYMBOLS

EXTERNAL SYMBOLS
ISIS E 0000

USER SYMBOLS
ACTUAL D 0026  AFT D 0000  BUFFER D 0028  CAFT D 001C  CBLK D 001C  CHECKN D 0C28  CHKSUM C 0000
CLOSE A 0001  DONE C 0062  EBLK D 0022  ERR C 008D  ERROR A 000C  EXIT A 0009  ISIS E 0000
LOOP1 C 0030  LOOP2 C 003F  NMBR C 00C7  NMBR1 C 00D4  NUMBER C 0044  NUMBR2 C 00B5  OBLK D 000A
OPEN A 0000  PACKET C 009D  RBLK D 0000  READ A 0003  STATUS D 0022  SUM C 004F  SUM1 C 0059
UNPACK C 0089  WBLK D 0014  WRITE A 0004  XBLK D 0020

ASSEMBLY COMPLETE, NO ERRORS

8/78
WRITEP - Output Procedures for PLM80

Contains several procedures to call by PLM80 programs for formatted output of ADDRESS or BYTE values, or output of characters and strings.

MDS Intellec 800

PLM80
The program uses the ISIS-call WRITE for doing its output, but may easily be changed in order to use another output procedure, and thus become independent of ISIS.

See Attached.

See Attached.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL</td>
<td>Karl Pentzlin</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RAM Required:</th>
<th>Company:</th>
</tr>
</thead>
<tbody>
<tr>
<td>00CEH</td>
<td>Informatik-Forum GmbH</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ROM Required:</th>
<th>Address:</th>
</tr>
</thead>
<tbody>
<tr>
<td>0AF4H</td>
<td>Fasanenstraße 16</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Maximum Subroutine Nesting Level:</th>
<th>City:</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>D-8000 München 60</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Assembler/Compiler Used:</th>
<th>State:</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLM80 Compiler V.3.0</td>
<td>WEST GERMANY</td>
</tr>
</tbody>
</table>
WRITEP contains procedures as listed below:

1. Formatted Output Procedures:

   CALL WRITEA (address$value, format)
   CALL WRITEB (byte$value, format)

   where "format" is to be declared as ADDRESS parameter and has to be a string of one or two characters. The first character specifies the wanted format, as listed below. The second character, if given, specifies the number of digits to be printed (not at formats L and Y).

   B  Binary format. Default length: 16 for WRITEA, 8 for WRITEB
   O or Q: Octal format. Default length: 6 for WRITEA, 3 for WRITEB
   D  Decimal format. Default length: 5 for WRITEA, 3 for WRITEB
   H  Hex format. Default length: 4 for WRITEA, 2 for WRITEB
   I  Integer format. Decimal digits, leading zeroes are replaced by spaces. Default length: Number of digits to be printed without leading zeroes, depending on specified value.
   S  Signed integer format. Like Integer format, but, if most significant bit of value is set, it is interpreted as negative sign.
   L  Logical format. The second character of the format string (must not be a format character. Default is "+" ) is printed, if the specified value is "true". If "false", a space is printed.
   N  Name format. Only for WRITEB. The name of the specified value interpreted as an ASCII character is printed (i.e., "LF" for OAH). Default length: Length of the name. For all names check source file.
   Y  Two bytes octal format. Only for WRITEA.

   High and low byte of specified value are printed by 3 octal digits each, and are separated by the second character of the format string (which must not be a format character. Default is ".") Therefore, the format length is 7.

   Moreover, there is an error format, which is used if the format string does not begin with a format character. It is like format H, with additional printing of the erroneous format string.

2. Other Output procedures:

   CALL WRITEM
   Print the end of a line (that is, CR and LF).

   CALL WRITEC (character, count)
   Write the specified character <count> times. "character" is to be declared as BYTE parameter, "count" as ADDRESS parameter.

   CALL WRITES (stringpointer)
   Write the string based on "stringpointer" (which is to be declared as ADDRESS parameter). The string may contain any ASCII characters and is to be terminated with an ETX (End of Text, 03H) character.
CALL WRITEN (stringpointer, count)
Write the string based by "stringpointer", and the length of which is
<count> (both parameters are to be declared ADDRESS).

CALL WRITEV (stringpointer, count)
Like WRITEN, but invisible ASCII characters (like CR, LF) are replaced
by visible ones (CR and LF by "/", HT (Horizontal Tabulation) by the
inverse slash, all others by "!").

3. Output Control procedure:

   CALL WRITED (aftn)
   Change output device. "aftn" (which is to be declared as ADDRESS
   parameter) specifies the output device by its AFTN, as it is given
   by the ISIS-call OPEN. Console output (:CO:) is specified by zero.
   At beginning of a program run, console output is output device for
   all WRITEP procedures.

WRITEP uses an internal buffer for all output done by WRITEP procedures
except when output device is console output (console output is always done
immediately during each WRITEP procedure call).
Therefore, regard this notice:

Except if you output on console, you have to call WRITED before you want to
output on the same device as used by WRITEP by other than WRITEP procedures
(e.g. by ISIS-call WRITE), and you have to call WRITED at the end of your
program run, due to the fact that WRITED empties the internal buffer before
regarding the new "aftn" value (which, in fact, may be the same as the old one).
**Program Title**  
ERRORP/MESSGP - Printing Subroutine for ISIS-II Error Messages and User Messages with Filename

**Function**  
ERRORP prints an ISIS-II error message, together with a filename, if one is specified.  
MESSGP does the same with a user defined message.

**Required Hardware**  
MDS 800 Intellec

**Required Software**  
ISIS-II  
The system calls WRITE and ERROR are used.

**Input Parameters**  
See Attached.

**Output Results**  
See Attached.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASSUME ALL</td>
<td>Karl Pentzlin</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RAM Required:</th>
<th>Company:</th>
</tr>
</thead>
<tbody>
<tr>
<td>0020H</td>
<td>Informatik-Forum GmbH</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ROM Required:</th>
<th>Address:</th>
</tr>
</thead>
<tbody>
<tr>
<td>061BH</td>
<td>Fasanenstraße 16</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Maximum Subroutine Nesting Level:</th>
<th>City:</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>D-8000 München 60</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Assembler/Compiler Used:</th>
<th>State:</th>
</tr>
</thead>
<tbody>
<tr>
<td>PL/M-80 Compiler V.3.0</td>
<td>WEST GERMANY</td>
</tr>
</tbody>
</table>
ERRORP is to be declared as follows:

```
ERRORP: PROCEDURE (ERRORNUMBER, FILENAME) EXTERNAL;
DECLARE (ERRORNUMBER, FILENAME) ADDRESS; END;
```

ERRORNUMBER contains the error number as given by any ISIS-II system call by the STATUS parameter. If ERRORNUMBER = 0, (that is, the ISIS-II system call was done without an error) the call of ERRORP will have no effect. Otherwise, an error message will be output as readable text, e.g. "NO SUCH FILE" if ERRORNUMBER = 13.

FILENAME either contains 0 (then only the error message will be printed), or is a pointer to the beginning of an ISIS-II filename or device specification. Then this name will be printed before the error message, separated by a comma (without leading spaces, if any).

MESSGP is to be declared as follows:

```
MESSGP: PROCEDURE (TEXTPOINTER, FILENAME) EXTERNAL,
DECLARE (TEXTPOINTER, FILENAME) ADDRESS; END;
```

TEXTPOINTER is a pointer to the beginning of an ASCII string representing the user defined message. The text has to be concluded by an ETX (End of Text, O3H) byte.

FILENAME has the same function as for ERRORP.

All printing will be done on :CO: (Console Output). Before printing a message, a Carriage Return/Line Feed will be output.
FORMFD-FORMFEED TO LINEFEED CONVERSION PROGRAM

This program will convert all formfeeds in a program listing, except the first one, to line feeds. The program was created for use with the Intel MDS-770 lineprinter. The program will add a paging feature to the 770 printer.

MDS-800 or SERIES-II Development System

ISIS-II Operating System
MDS System Monitor

Syntax as follows:

FORMFD filename.LST To :LP: (To Lineprinter)
To :TO: (To Teletype List)
To filename.EXT (To Store On Disk)

The output device will list your program with the appropriate number of Linefeeds replacing all but the first formfeed.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM Required:</td>
<td>Mary Jane Elmore</td>
</tr>
<tr>
<td>32K</td>
<td>Company:</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>Intel Corp.</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td>Address:</td>
</tr>
<tr>
<td>Assembler/Compiler Used:</td>
<td>3065 Bowers Ave.</td>
</tr>
<tr>
<td>PL/M 80</td>
<td>City:</td>
</tr>
<tr>
<td></td>
<td>Santa Clara</td>
</tr>
<tr>
<td></td>
<td>State:</td>
</tr>
<tr>
<td></td>
<td>Calif. 95051</td>
</tr>
</tbody>
</table>
ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE ERRBLOCK
OBJECT MODULE PLACED IN :F4:UTILS.OBJ
COMPILER INVOKED BY: PLM80 :F4:UTILS.PLM WORKFILES(:F4:, :F4:)

$DEBUG PAGEWIDTH(80)

/*PROCEDURE TO SEND ERROR MESSAGE TO CONSOLE */

1 ERRBLOCK: DO;

2   1 EXIT:
     PROCEDURE EXTERNAL;
3    2 END EXIT;

4   1 ERROR:
     PROCEDURE(ERRNUM) EXTERNAL;
5    2 DECLARE ERRNUM ADDRESS;
6    2 END ERROR;

7   1 ERRPRT: PROCEDURE(ERRNO) PUBLIC;
8    2 DECLARE ERRNO ADDRESS;
9    2 CALL ERROR(ERRNO);

10   2 CALL EXIT;
11    2 END ERRPRT;
12   1 END ERRBLOCK;

MODULE INFORMATION:

  CODE AREA SIZE   = 0012H   18D
  VARIABLE AREA SIZE = 0002H   2D
  MAXIMUM STACK SIZE = 0002H   2D
  30 LINES READ
  0 PROGRAM ERROR(S)

END OF PL/M-80-compilation
DECLARE BUFFER(128) BYTE;
DECLARE OUTBUF(2000) BYTE;
DECLARE ACTUAL#COUNT ADDRESS;
DECLARE STATUS ADDRESS;
DECLARE AFT#IN ADDRESS;
DECLARE AFT#OUT ADDRESS;
DECLARE READ#ACCESS LITERALLY '1';
DECLARE WRITE#ACCESS LITERALLY '2';
DECLARE NOT#END#OF#FILE LITERALLY 'ACTUAL#COUNT <> 0';
DECLARE ERRMSG(*) BYTE DATA ('ERROR IN COMMAND TAIL',0DH,0AH);
DECLARE OUT#BUF#INDEX BYTE;
DECLARE READ#BUF#INDEX ADDRESS;
DECLARE WRITE#BUF#INDEX ADDRESS;
DECLARE LINE#COUNT BYTE;
DECLARE I BYTE;
DECLARE LINE#FEED LITERALLY '0AH';
DECLARE FORM#FEED LITERALLY '0CH';
DECLARE FIRST#READ BYTE;
DECLARE TRUE LITERALLY '1';
DECLARE FALSE LITERALLY '0';

OPEN:
PROCEDURE(AFT, FILE, ACCESS, MODE, STATUS) EXTERNAL;
DECLARE(AFT, FILE, ACCESS, MODE, STATUS) ADDRESS;
END OPEN;

CLOSE:
PROCEDURE(AFT, STATUS) EXTERNAL;
DECLARE(AFT, STATUS) ADDRESS;
END CLOSE;

READ:
PROCEDURE(AFT, BUFFER, COUNT, ACTUAL, STATUS) EXTERNAL;
DECLARE(AFT, BUFFER, COUNT, ACTUAL, STATUS) ADDRESS;
END READ;

WRITE:
PROCEDURE(AFT, BUFFER, COUNT, STATUS) EXTERNAL;
DECLARE(AFT, BUFFER, COUNT, STATUS) ADDRESS;
END WRITE;
34 1  EXIT:
   PROCEDURE EXTERNAL;
35 2  END EXIT;
36 1  ERROR:
   PROCEDURE (ERRNUM) EXTERNAL;
37 2  DECLARE ERRNUM ADDRESS;
38 2  END ERROR;
39 1  ERRPRT:
   PROCEDURE (ERRNO) EXTERNAL;
40 2  DECLARE ERRNO ADDRESS;
41 2  END ERRPRT;

/* READ THE CONSOLE FILE TO GET THE PARAMETER STRING */
42 1  CALL READ(1, BUFFER, 128, ACTUAL$COUNT, STATUS);
43 1  IF STATUS > 0 THEN CALL ERRPRT(STATUS);
44 1  CALL OPEN(TEXTOUT$, BUFFER, READACCESS, 0, STATUS);
45 1  IF STATUS > 0 THEN CALL ERRPRT(STATUS);

/* OUTPUT BUFFER INDEX LOOP */
48 1  OUT$BUF$INDEX = 0;
49 1  I = 0;
50 1  DO WHILE OUT$BUF$INDEX = 0 AND I <= ACTUAL$COUNT;
51 2  IF BUFFER(I) = ' ' THEN
52 2  IF BUFFER(I+1) = 'T' THEN
53 2  IF BUFFER(I+2) = '0' THEN
54 2  IF BUFFER(I+3) = ' ' THEN
55 2  OUT$BUF$INDEX = I+4;
56 2  END;
57 2  I = I+1;
58 2  END;
59 1  IF OUT$BUF$INDEX = 0 THEN
60 1  DO:
61 2  CALL WRITE(0$, ERRMSG, 23, STATUS);
62 2  CALL EXIT;
63 2  END;
64 1  CALL OPEN(TEXTOUT$, BUFFER(OUT$BUF$INDEX), WRITEACCESS, 0, STATUS);
65 1  IF STATUS > 0 THEN CALL ERRPRT(STATUS);
/* CHANGE FORMFEEDS TO LINEFEEDS */

FIRST$READ = TRUE;
WRITE$BUF$INDEX = 0;
READ$BUF$INDEX = 0;
LINE$COUNT = 66;
ACTUAL$COUNT = 1;

CALL READ(AFT$IN, BUFFER, 128, ACTUAL$COUNT, STATUS);

DO WHILE NOT$END$OF$FILE;
    IF STATUS > 0 THEN CALL ERRPRT(STATUS);
    IF FIRST$READ = TRUE THEN
        DO;
            OUTBUF(0) = BUFFER(0);
            WRITE$BUF$INDEX = WRITE$BUF$INDEX + 1;
            DO READ$BUF$INDEX = 1 TO ACTUAL$COUNT - 1;
            IF BUFFER(READ$BUF$INDEX) = FORM$FEED THEN
                DO I = 0 TO LINE$COUNT - 1;
                    OUTBUF(WRITE$BUF$INDEX) = LINE$FEED;
                    WRITE$BUF$INDEX = WRITE$BUF$INDEX + 1;
                END;
                LINE$COUNT = 66;
            END;
        ELSE DO;
            OUTBUF(WRITE$BUF$INDEX) = BUFFER(READ$BUF$INDEX);
            WRITE$BUF$INDEX = WRITE$BUF$INDEX + 1;
        END;
        IF BUFFER(READ$BUF$INDEX) = LINE$FEED THEN
            LINE$COUNT = LINE$COUNT - 1;
        IF LINE$COUNT = 0 THEN
            LINE$COUNT = 66;

        END;
        FIRST$READ = FALSE;
    END;
END;
99 2 ELSE DO;

100 3 DO READ$BUF$INDEX = 0 TO ACTUAL$COUNT - 1;
101 4 IF BUFFER<READ$BUF$INDEX> = FORM$FEED THEN
102 4 DO;
103 5 DO I = 0 TO LINE$COUNT - 1;
104 6 OUTBUF<WRITE$BUF$INDEX> = LINE$FEED;
105 6 WRITE$BUF$INDEX = WRITE$BUF$INDEX + 1;
106 6 END;
107 5 LINE$COUNT = 66;
108 5 END;
109 4 ELSE DO;
110 5 OUTBUF<WRITE$BUF$INDEX> = BUFFER<READ$BUF$INDEX>;
111 5 WRITE$BUF$INDEX = WRITE$BUF$INDEX + 1;
112 5 END;
113 4 IF BUFFER<READ$BUF$INDEX> = LINE$FEED THEN
114 4 LINE$COUNT = LINE$COUNT - 1;
115 4 IF LINE$COUNT = 0 THEN
116 4 LINE$COUNT = 66;
117 4 END;
118 3 END;
119 2 CALL WRITE(AFT$OUT, OUTBUF, WRITE$BUF$INDEX, STATUS);
120 2 IF STATUS > 0 THEN CALL ERRPRT(STATUS);
122 2 WRITE$BUF$INDEX = 0;
123 2 CALL READ(AFT$IN, BUFFER, 128, ACTUAL$COUNT, STATUS);
124 2 END;
125 1 CALL CLOSE(AFT$IN, STATUS);
126 1 IF STATUS > 0 THEN CALL ERRPRT(STATUS);
128 1 CALL CLOSE(AFT$OUT, STATUS);
129 1 IF STATUS > 0 THEN CALL ERRPRT(STATUS);
131 1 CALL EXIT;
132 1 END FORMFD;
PL/M-80 COMPILER

MODULE INFORMATION:

CODE AREA SIZE = 0339H 827D
VARIABLE AREA SIZE = 0860H 2144D
MAXIMUM STACK SIZE = 0008H 8D
220 LINES READ
0 PROGRAM ERROR(S)

END OF PL/M-80 COMPILATION
MONITOR ROUTINES FOR A 3M DCD1 CASSETTE TAPE DRIVE

To record and play-back records from a SBC system to a 3M DCD1 tape cassette. Intended as an extension of the SBC-monitor.

TTY or equiv. serial terminal. One 24 bits parallel port (8255) One 3M DCD1 tape cassette drive incl. electronics. One 74123 TTL circuit or equiv. Six 220Ω pull up resistors.

SBC 80P monitor or equiv. routines. The tape commands can easily be incorporated into the SBC monitor.

1) \texttt{CW} on the TTY to print the catalogue of the tape.

2) \texttt{OW} on the TTY to dump the content of a memory area onto the tape and get the catalogue updated. The start address, end address and record number will be asked for by the routine and must be given by the user.

3) \texttt{PW} on the TTY to play back a record from the tape into the memory. The start address, address limit and record number will be asked for by the routine and must be given by the user.

The program will continuously check the status of the tape drive and give a message if anything is wrong. The output routine will check read the written record and then update the catalogue of the tape. In case of read error the program pushes the addresses of the incorrect bytes during the read operation, and then lists these addresses on the TTY.

For further information, see the demonstration run or contact the programmer.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>Per Nylen</td>
</tr>
<tr>
<td>RAM Required:</td>
<td>Company:</td>
</tr>
<tr>
<td>1 k bytes</td>
<td>University of Stockholm</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>Address:</td>
</tr>
<tr>
<td>10750 bytes</td>
<td>Vanadisvägen 9</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td>City:</td>
</tr>
<tr>
<td></td>
<td>S-113 46 Stockholm</td>
</tr>
<tr>
<td>Assembler/Compiler Used:</td>
<td>State:</td>
</tr>
<tr>
<td>ASM/20</td>
<td>Sweden</td>
</tr>
</tbody>
</table>
ALPHA: AN ALPHABETIZED LISTING OF THE DISK DIRECTORY OUTPUT TO THE LINE PRINTER

1) Stack pointer set, 2) system list device set to line printer, 3) file :FO: ALPHA DIR opened and read, 4) first two lines of directory printer, 5) table built of addresses of entries in the directory in alphabetical order, 6) the lines of the directory printed as they are listed in the table, 7) the last two lines of the directory printed, 8) the system list device reset, 9) program exited and control returned to ISIS-II.

MDS
disk drive: FO:
line printer
console input (CRT or TTY)
ISIS-II (Tested on V2.2D, V3.4D)
Intellect Monitor

:FO: ALPHA. DIR Input to buffer in RAM (IBUF)

Directory is output character by character from Register C by calling the LO routine.

I/O operations are performed by the ISIS-II subroutine calls and driver routines.

Registers Modified:
A,F,B,C,D,E,H,I,SP,PC

Programmer:
S. Bann

RAM Required: Module is located 3400 to 56BAH 2298H + BUFFERS

Company:
Xerox

ROM Required:
MDS Monitor loaded in ROM

Address:
701 S. Aviation Blvd. A2-43

Maximum Subroutine Nesting Level:
One level

City:
El Sigundo

Assembler/Compiler Used:
ISIS-II 8080/8085 Macro Assembler V2.0

State:
CA 90245
LOC OBJ

SEQ SOURCE STATEMENT

1 # TITLE("PRINT ALPHABETIZED DIRECTORY")
2
3
4 ; PROGRAM NAME: ALPHA
5 ; WRITTEN BY: SHELLI LANN BANN
6 ; DATE: JUNE 30, 1978
7 ; REVISION: A.00
8
9 ; ALPHA IS EXECUTED UNDER THE CONTROL OF ISIS-2
10
11 ; THIS PROGRAM TAKES AS INPUT :FILE: ALPHA DIR, A FILE CONTAINING THE
12 ; UNALPHABETIZED DIRECTORY LISTING CREATED WITH THE ISIS-2 DIR COMMAND.
13
14 ; IT CREATES AS AN OUTPUT BUFFER A TABLE OF POINTERS(ADRESSES) WHICH
15 ; REFER TO THE FIRST BYTES OF FILENAMES IN THE INPUT FILE ALPHABETICALLY.
16 ; I.E. THE FIRST POINTER AT THE TOP OF THE OUTPUT FILE REFERS TO THE FIRST
17 ; FILENAME TO BE OUTPUT IN ALPHABETICAL ORDER
18
19 ; IT OUTPUTS THE ALPHABETIZED DIRECTORY TO THE LINE PRINTER
20
21 ; ALPHA ASSUMES: 1) THE DIRECTORY LISTING CONSISTS OF 2 LINES OF HEADER.
22 ; A VARIABLE NUMBER OF LINES OF FILENAMES AND THEIR LENGTHS AND ATTRIBUTES.
23 ; AND 2 LINES OF ENDING; 2) EACH LINE IS SEPARATED BY A LINE FEED CHARACTER;
24 ; 3) THE SECOND TO THE LAST LINE IS UNIQUE BECAUSE IT STARTS WITH A SPACE;
25 ; 4) EACH FILENAME IS UNIQUE.
26
27 ; THESE ARE JUSTIFIED ASSUMPTIONS UNDER V2.2D AND V3.4D ISIS-2 SOFTWARE
28
29 ; SYSTEM CALLS
30
31 ; EXTRN IOST  ; SET I/O CONFIGURATION
32 ; EXTRN IOCR ; GET I/O CONFIGURATION
33 ; EXTRN LIO  ; LIST OUTPUT
34 ; EXTRN ISIS
35
36 ; DSEG
37
38 0000 OPEN EQU 0
39 0003 READ EQU 3
40 0009 EXIT  EQU  9
41
42 ; OPEN CALL PARAMETERS
43 0000 1A00 D  43 OBLK: DW  PBLK
44 0002 0C08 D  44 DW  OFILE
45 0004 0100 D  45 DW  1
46 0006 0000 D  46 DW  0
47 0008 0000 D  47 DW  OSTAT
48 0002 0800 D  48 OSTAT: DS  2
49 0008 3A6303A OFILE: DB '.F0:ALPHA DIR '
50 0010 41C5848
50 0014 412E4449
50 0018 5220

10/78
LOC OBJ SEE SOURCE STATEMENT

50
51 ;READ CALL PARAMETERS
0002 52 RBLK: DS 2
0010 53 DW IBUF
001E 54 DW 1900H
0020 55 DW ACTUAL
0022 56 DW RSTAT
0024 57 ACTUAL: DS 2
0026 58 RSTAT: DS 2
59
60 ;EXIT CALL PARAMETERS
0028 61 EBLK: DW ESTAT
0030 62 ESTAT: DS 2
63
64 ;SYSTEM CONFIGURATION STATUS BYTE STORAGE
0031 65 IOSTOR: DS 1
66
67 ;INPUT BUFFER FOR ALPHA DIR
2000 68 IBUF: DS 2000H
69
70 ;OUTPUT BUFFER VARIABLES AND ADDRESSES
71 ;THE OUTPUT BUFFER IS BUILT UPWARDS WITH NEW ENTRIES PUSHING THE TOP
72 ;OF THE BUFFER UPWARDS (TOWARDS LOWER MEMORY ADDRESSES) WHILE THE BOTTOM
73 ;OF THE BUFFER REMAINS FIXED.
74 ;EACH ENTRY TAKES TWO BYTES OF STORAGE BECAUSE THEY ARE ADDRESSES
75 ;THE LOW BYTE PREcedes THE HIGH BYTE FOR EACH ADDRESS
76
2020 FF 77 XX: DB 0FFH ;EOF DELIMITER FOR BUFFER
0190 78 BUFFER: DS 4000 ;LIST OF ADDRESSES OF ALPHABETIZED
79 ;FILE NAMES
21BE 2020 80 ENDBUF: DW XX ;ADDRESS OF EOF BUFFER DELIMITER
21C0 B21 81 TPBUFF: DW ENDBUF
82
83
84 CSEG
85
86 ;SET STACK POINTER
87 START: STKLN 200 ;SET THE MAXIMUM STACK LENGTH
0000 88 LXI SP, STACK ;LOAD STACK POINTER
89 WITH VALUE DETERMINED BY
90 THE LOCATE COMMAND
91
92 ;SET LIST DEVICE TO LINE PRINTER
0003 CD0000 93 CALL I0CHK ;GET I/O DEVICE CONFIGURATION
0006 322C00 94 STA I0STOR ;STORE STATUS WORD
0009 E63F 95 ANDI 3FH ;CLEAR LIST DEVICE
000B F600 96 ORI 80H ;SET LINE PRINTER TO LIST DEVICE
0000 4F 97 MOV C, A
98
99
100 OPEN DIRECTORY FILE
101
102 0B00 102 MVI C, OPEN
103 110000 D 103 LXI D, OBLK
104 CD0000 E 104 CALL ISIS
LOC OBJ SED SOURCE STATEMENT

105
106 ;READ DIRECTORY FILE
107
0019 0083 108 MVI C, READ
001B 111000 D 109 LXI D, RBLK
001E C00000 E 110 CALL ISIS
111
112
113 ;PRINT OUT 2 LINE HEADER
0021 008C 114 MVI C, 0CH
0023 CD0000 E 115 CALL LO
0026 008A 116 MVI C, 0AH
0028 CD0000 E 117 CALL LO
002B 212000 D 118 LXI H, IBUF
002E 4E 119 ;SET HIL TO TOP OF INPUT BUFFER
120
121 ;FIRST LINE
0032 7E 122 MOV A, M
0033 EE0A 123 XRI 0AH
0035 23 124 INX H
0036 C22E00 C 125 JNZ PANTHD
0039 4E 126 ;GET CHARACTER FROM INPUT BUFFER
003A CD0000 E 127 CALL LO
003D 7E 128 MOV A, M
003E EE0A 129 XRI 0AH
0044 23 130 INX H
0041 C23900 C 131 JNZ PANTHD2
0044 7E 132 ;GET CHAR FROM INPUT BUFFER
0045 EE20 133 MOV A, M
0047 C9B400 C 134 XRI 20H
004A 44 135 JZ LLINES
004B 4D 136 ;ALPHABETIZE ROUTINE
004C 29021 D 137 ;RESET TOP OF BUFFER
004D 4D 138 ;LOOK : MOV B, H
004E 4D 139 ;POINT B&G TO INPUT CHARACTER
004F 2B 140 MOV C, L
0050 2B 141 LHLD TBUFF
0051 29021 D 142 ;POINT HIL TO TOP OF OUT BUFFER
0052 2B 143 DCH H
0053 2B 144 ;MOVE TOP OF OUT BUFFER
0054 2B 145 DCH H
0055 2B 146 ;BACK TWO SPACES
0056 2B 147 SHLD TBUFF
0057 C23021 D 148 ;STORE NEW TOP OF OUT BUFFER
0058 23 149 INX H
0059 23 150 SET HIL BACK TO OLD
005A 23 151 INX H
005B 23 152 ;TOP OF BUFFER
005C 23 153 ;NEW WORD REFERS TO FILENAME TO BE ALPHABETIZED
005D 23 154 ;OLD WORD REFERS TO THE FILENAME ALREADY ALPHABETIZED TO WHICH THE NEW WORD
005E 23 155 IS CURRENTLY BEING COMPAR
005F 23 156 ;COMPARE FIRST LETTER OF NEW FILENAME TO ALREADY ALPHABETIZED FILENAMES
0060 23 157
0061 23 158
0062 23 159
0063 23 160
0064 23 161
0065 23 162
0066 23 163
0067 23 164
0068 23 165
0069 23 166
006A 23 167
006B 23 168
006C 23 169
006D 23 170
006E 23 171
006F 23 172
0070 23 173
0071 23 174
0072 23 175
0073 23 176
0074 23 177
0075 23 178
0076 23 179
0077 23 180
0078 23 181
0079 23 182
007A 23 183
007B 23 184
007C 23 185
007D 23 186
007E 23 187
007F 23 188
0080 23 189
0081 23 190
0082 23 191
0083 23 192
0084 23 193
0085 23 194
0086 23 195
0087 23 196
0088 23 197
0089 23 198
008A 23 199
008B 23 200
008C 23 201
008D 23 202
008E 23 203
008F 23 204
0090 23 205
0091 23 206
0092 23 207
0093 23 208
0094 23 209
0095 23 210
0096 23 211
0097 23 212
0098 23 213
0099 23 214
009A 23 215
009B 23 216
009C 23 217
009D 23 218
009E 23 219
009F 23 220
00A0 23 221
00A1 23 222
00A2 23 223
00A3 23 224
00A4 23 225
00A5 23 226
00A6 23 227
00A7 23 228
00A8 23 229
00A9 23 230
00AA 23 231
00AB 23 232
00AC 23 233
00AD 23 234
00AE 23 235
00AF 23 236
00B0 23 237
00B1 23 238
00B2 23 239
00B3 23 240
00B4 23 241
00B5 23 242
00B6 23 243
00B7 23 244
00B8 23 245
00B9 23 246
00BA 23 247
00BB 23 248
00BC 23 249
00BD 23 250
00BE 23 251
00BF 23 252
00C0 23 253
00C1 23 254
00C2 23 255
00C3 23 256
00C4 23 257
00C5 23 258
00C6 23 259
00C7 23 260
00C8 23 261
00C9 23 262
00CA 23 263
00CB 23 264
00CC 23 265
00CD 23 266
00CE 23 267
00CF 23 268
00D0 23 269
00D1 23 270
00D2 23 271
00D3 23 272
00D4 23 273
00D5 23 274
00D6 23 275
00D7 23 276
00D8 23 277
00D9 23 278
00DA 23 279
00DB 23 280
00DC 23 281
00DD 23 282
00DE 23 283
00DF 23 284
00E0 23 285
00E1 23 286
00E2 23 287
00E3 23 288
00E4 23 289
00E5 23 290
00E6 23 291
00E7 23 292
00E8 23 293
00E9 23 294
00EA 23 295
00EB 23 296
00EC 23 297
00ED 23 298
00EE 23 299
00EF 23 300
00F0 23 301
00F1 23 302
00F2 23 303
00F3 23 304
00F4 23 305
00F5 23 306
00F6 23 307
00F7 23 308
00F8 23 309
00F9 23 310
00FA 23 311
00FB 23 312
00FC 23 313
00FD 23 314
00FE 23 315
00FF 23 316

10/78
<table>
<thead>
<tr>
<th>LOC</th>
<th>OBJ</th>
<th>SEG</th>
<th>SOURCE STATEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0056 E5</td>
<td>160</td>
<td>SEARCH</td>
<td>PUSH H ; STORE POINTER OF ADDRESS OF OLD WORD IN STACK</td>
</tr>
<tr>
<td>0057 5E</td>
<td>162</td>
<td>MOV</td>
<td>E.M ; POINT TO FIRST CHARACTER</td>
</tr>
<tr>
<td>0058 23</td>
<td>163</td>
<td>INX</td>
<td>H ; OF FILENAME IN OUT BUFFER</td>
</tr>
<tr>
<td>0059 56</td>
<td>164</td>
<td>MOV</td>
<td>D.M ; OF FILENAME IN INPUT BUFFER</td>
</tr>
<tr>
<td>005A 60</td>
<td>165</td>
<td>MOV</td>
<td>H.B ; OF FILENAME IN INPUT BUFFER</td>
</tr>
<tr>
<td>005B 69</td>
<td>167</td>
<td>MOV</td>
<td>L.C</td>
</tr>
<tr>
<td>005C 7E</td>
<td>168</td>
<td>MOV</td>
<td>A.M ; GET CHARACTER FROM IN BUFFER</td>
</tr>
<tr>
<td>005D EB</td>
<td>169</td>
<td>XORH</td>
<td>; POINT HAL TO OLD WORD</td>
</tr>
<tr>
<td>005E BE</td>
<td>170</td>
<td>CMP</td>
<td>M ; AND POINT DUE TO NEW WORD</td>
</tr>
<tr>
<td>005F CA7000 C</td>
<td>172</td>
<td>JZ</td>
<td>EQUAL ; COMPARE NEW WORD WITH OLD WORD</td>
</tr>
<tr>
<td>0062 D28800 C</td>
<td>173</td>
<td>JNC</td>
<td>SORT ; NEW WORD &gt; OLD WORD</td>
</tr>
<tr>
<td>174</td>
<td>ADD ADDRESS OF NEW FILENAME TO TOP OF ALPHABETIZED LIST</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0065 E1</td>
<td>176</td>
<td>FOUND</td>
<td>POP H ; POINT HAL TO ADDRESS OF OLD WORD</td>
</tr>
<tr>
<td>0066 28</td>
<td>178</td>
<td>DCX</td>
<td>H ; IN OUTPUT BUFFER</td>
</tr>
<tr>
<td>0067 70</td>
<td>179</td>
<td>MOV</td>
<td>M.B ; SET HAL TWO SPACES BACK</td>
</tr>
<tr>
<td>0068 28</td>
<td>180</td>
<td>DCX</td>
<td>H ; SET ADDRESS OF NEW WORD</td>
</tr>
<tr>
<td>0069 71</td>
<td>181</td>
<td>MOV</td>
<td>M.C</td>
</tr>
<tr>
<td>182</td>
<td>GET THE NEXT WORD TO ALPHABETIZE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>006A 60</td>
<td>186</td>
<td>NXTWD</td>
<td>MOV H.B ; POINT HAL TO IN BUFFER AT START</td>
</tr>
<tr>
<td>006B 69</td>
<td>187</td>
<td>MOV</td>
<td>L.C ; OF FILENAME THAT WAS JUST ALPHABETIZED</td>
</tr>
<tr>
<td>006C 7E</td>
<td>188</td>
<td>NXTLR</td>
<td>MOV A.M ; GET CHARACTER FROM IN BUFFER</td>
</tr>
<tr>
<td>006D EEB8</td>
<td>189</td>
<td>XRI</td>
<td>0AH</td>
</tr>
<tr>
<td>006E 23</td>
<td>190</td>
<td>INX</td>
<td>H ; IS CHAR A LINE FEED?</td>
</tr>
<tr>
<td>0070 C26C00 C</td>
<td>191</td>
<td>JNZ</td>
<td>NXTLR ; POINT HAL TO NEXT CHAR IN IN BUFFER</td>
</tr>
<tr>
<td>0073 7E</td>
<td>192</td>
<td>MOV</td>
<td>A.M ; GET CHAR FROM IN BUFFER</td>
</tr>
<tr>
<td>0074 EE20</td>
<td>193</td>
<td>XRI</td>
<td>20H ; IS CHAR A SPACE?</td>
</tr>
<tr>
<td>0076 CA9500 C</td>
<td>194</td>
<td>JZ</td>
<td>PTDIR ; JUMP IF CHAR IS SPACE TO OUTPUT</td>
</tr>
<tr>
<td>0079 C34A00 C</td>
<td>196</td>
<td>JMP</td>
<td>LOOK ; THE DIRECTORY</td>
</tr>
<tr>
<td>007C 23</td>
<td>197</td>
<td>EQUAL</td>
<td>; ELSE JUMP TO ALPHABETIZE THIS NEW FILENAME</td>
</tr>
<tr>
<td>007D 13</td>
<td>199</td>
<td>INK</td>
<td>D ; FIRST LETTERS ARE SAME, COMPARE NEXT LETTERS OF SAME WORDS</td>
</tr>
<tr>
<td>007E 1A</td>
<td>200</td>
<td>LDAX</td>
<td>D ; POINT HAL TO NEXT CHAR OF OLD WORD</td>
</tr>
<tr>
<td>201</td>
<td>POINT D TO NEXT CHAR OF NEW WORD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>202</td>
<td>MOVE CHARACTER OF NEW WORD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>203</td>
<td>INTO ACCUMULATOR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>007F BE</td>
<td>204</td>
<td>CMP</td>
<td>M ; COMPARE NEW WORD TO OLD WORD</td>
</tr>
<tr>
<td>0080 CA7000 C</td>
<td>205</td>
<td>JZ</td>
<td>EQUAL ; JUMP IF LETTERS ARE THE SAME</td>
</tr>
<tr>
<td>0083 D46500 C</td>
<td>206</td>
<td>JC</td>
<td>FOUND ; JUMP IF NEW WORD &gt; OLD WORD</td>
</tr>
<tr>
<td>207</td>
<td>MOVES OLD WORD BACK IN BUFFER</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0086 E1</td>
<td>209</td>
<td>SORT</td>
<td>POP H ; HAL POINT TO THE ADDRESS OF THE OLD WORD</td>
</tr>
<tr>
<td>0087 54</td>
<td>210</td>
<td>MOV</td>
<td>D.H</td>
</tr>
<tr>
<td>0088 50</td>
<td>211</td>
<td>MOV</td>
<td>E.L</td>
</tr>
<tr>
<td>0089 1B</td>
<td>212</td>
<td>DCX</td>
<td>D ; ONE POINT TO THE DESTINATION</td>
</tr>
<tr>
<td>008A 1B</td>
<td>213</td>
<td>DCX</td>
<td>D ; OF THE ADDRESS OF THE OLD WORD</td>
</tr>
</tbody>
</table>

10/78
<table>
<thead>
<tr>
<th>LOC</th>
<th>OBJ</th>
<th>SEQ</th>
<th>Source Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>0088 7E</td>
<td>215</td>
<td>MOV A, M</td>
<td>; MOVE BACK LOW BYTE</td>
</tr>
<tr>
<td>008C 12</td>
<td>216</td>
<td>STAX D</td>
<td></td>
</tr>
<tr>
<td>008D 23</td>
<td>217</td>
<td>INX H</td>
<td>; MOVE BACK HIGH BYTE</td>
</tr>
<tr>
<td>008E 13</td>
<td>218</td>
<td>INX D</td>
<td>; THE NEXT ENTRY IN THE OUTPUT BUFFER NOW BECOMES THE OLD WORD</td>
</tr>
<tr>
<td>008F 7E</td>
<td>219</td>
<td>MOV A, M</td>
<td>; THE NEXT ENTRY IN THE OUTPUT BUFFER NOW BECOMES THE OLD WORD</td>
</tr>
<tr>
<td>0090 12</td>
<td>220</td>
<td>STAX D</td>
<td>; THE NEXT ENTRY IN THE OUTPUT BUFFER NOW BECOMES THE OLD WORD</td>
</tr>
<tr>
<td>0091 23</td>
<td>221</td>
<td>INX H</td>
<td>; THE NEXT ENTRY IN THE OUTPUT BUFFER NOW BECOMES THE OLD WORD</td>
</tr>
<tr>
<td>0092 C35600 C</td>
<td>224</td>
<td>JMP SEARCH</td>
<td></td>
</tr>
<tr>
<td>0095 E5</td>
<td>230</td>
<td>PTDIR: PUSH H</td>
<td>; SET STACK TO POINT TO CHARACTER IN INPUT BUFFER</td>
</tr>
<tr>
<td>0096 2AC021 D</td>
<td>232</td>
<td>LHLD TPBUFF</td>
<td>; POINT HAL TO THE TOP OF THE OUTPUT BUFFER</td>
</tr>
<tr>
<td>0099 5E</td>
<td>234</td>
<td>PRINT: MOV E, M</td>
<td>; POINT DUE TO THE FILENAME</td>
</tr>
<tr>
<td>009A 23</td>
<td>235</td>
<td>INX H</td>
<td>; POINT HAL TO THE NEXT ENTRY IN THE OUT BUFFER</td>
</tr>
<tr>
<td>009B 56</td>
<td>236</td>
<td>MOV D, M</td>
<td>; POINT DUE TO NEXT ENTRY OF OUT BUFFER</td>
</tr>
<tr>
<td>009C 23</td>
<td>237</td>
<td>INX H</td>
<td>; GET CHARACTER TO BE OUTPUT</td>
</tr>
<tr>
<td>009D EB</td>
<td>239</td>
<td>XCHG</td>
<td>; IS CHAR END OF BUFFER?</td>
</tr>
<tr>
<td>009E 7E</td>
<td>241</td>
<td>MOV A, M</td>
<td>; JUMP IF CHAR IS END OF BUFFER DELIMITER</td>
</tr>
<tr>
<td>009F EEFF</td>
<td>242</td>
<td>XRI 0FFH</td>
<td>; JUMP IF CHAR IS NOT A LINE FEED</td>
</tr>
<tr>
<td>00A1 C9B300 C</td>
<td>243</td>
<td>JZ ENDING</td>
<td>; POINT HAL TO NEXT ENTRY IN OUT BUFFER</td>
</tr>
<tr>
<td>00A4 4E</td>
<td>245</td>
<td>PLTTR: MOV C, M</td>
<td>; PRINT CHARACTER</td>
</tr>
<tr>
<td>00A5 C00000 E</td>
<td>246</td>
<td>CALL LO</td>
<td>; PRINT CHARACTER</td>
</tr>
<tr>
<td>00A7 7E</td>
<td>247</td>
<td>MOV A, M</td>
<td>; GET SAME CHARACTER</td>
</tr>
<tr>
<td>00A9 EE0A</td>
<td>248</td>
<td>XRI 0AH</td>
<td>; IS CHAR A LINE FEED?</td>
</tr>
<tr>
<td>00B2 23</td>
<td>249</td>
<td>INX H</td>
<td>; POINT HAL TO NEXT CHARACTER</td>
</tr>
<tr>
<td>00BC 2B400 C</td>
<td>250</td>
<td>JNZ PLTTR</td>
<td>; PRINT CHARACTER</td>
</tr>
<tr>
<td>00BF EB</td>
<td>251</td>
<td>XCHG</td>
<td>; PRINT CHARACTER</td>
</tr>
<tr>
<td>00C0 C39900 C</td>
<td>253</td>
<td>JMP PRINT</td>
<td>; PRINT CHARACTER</td>
</tr>
<tr>
<td>00C3 E1</td>
<td>256</td>
<td>ENDING: POP H</td>
<td>; PRINT CHARACTER</td>
</tr>
<tr>
<td>00C4 4E</td>
<td>257</td>
<td>LLINES: MOV C, M</td>
<td>; PRINT CHARACTER</td>
</tr>
<tr>
<td>00C5 C00000 E</td>
<td>258</td>
<td>CALL LO</td>
<td>; PRINT CHARACTER</td>
</tr>
<tr>
<td>00C7 7E</td>
<td>259</td>
<td>MOV A, M</td>
<td>; GET SAME CHARACTER</td>
</tr>
<tr>
<td>00C9 EE0A</td>
<td>260</td>
<td>XRI 0AH</td>
<td>; IS CHAR A LINE FEED?</td>
</tr>
<tr>
<td>00CB 23</td>
<td>261</td>
<td>INX H</td>
<td>; POINT HAL TO NEXT CHARACTER</td>
</tr>
<tr>
<td>00CC C2B400 C</td>
<td>262</td>
<td>JNZ LLINES</td>
<td>; PRINT CHARACTER</td>
</tr>
<tr>
<td>00CF 4E</td>
<td>263</td>
<td>LLINE: MOV C, M</td>
<td>; PRINT CHARACTER</td>
</tr>
<tr>
<td>00D0 C00000 E</td>
<td>266</td>
<td>CALL LO</td>
<td>; PRINT CHARACTER</td>
</tr>
<tr>
<td>00D3 7E</td>
<td>267</td>
<td>MOV A, M</td>
<td>; GET SAME CHARACTER</td>
</tr>
<tr>
<td>00D4 EE0A</td>
<td>268</td>
<td>XRI 0AH</td>
<td>; IS CHARACTER A LINE FEED?</td>
</tr>
<tr>
<td>00D6 23</td>
<td>269</td>
<td>INX H</td>
<td>; POINT HAL TO NEXT CHARACTER</td>
</tr>
<tr>
<td>LOC</td>
<td>OBJ</td>
<td>SEQ</td>
<td>SOURCE STATEMENT</td>
</tr>
<tr>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>------------------</td>
</tr>
<tr>
<td>00C7</td>
<td>C20F00</td>
<td>C 270</td>
<td>JNZ LLINE ; JUMP IF CHARACTER IS NOT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>271</td>
<td>FOR A LINE FEED</td>
</tr>
<tr>
<td></td>
<td></td>
<td>272</td>
<td>; SET LIST DEVICE BACK AND EXIT TO ISIS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>273</td>
<td>LL</td>
</tr>
<tr>
<td>00CA</td>
<td>3B2000</td>
<td>D 275</td>
<td>XIT: LOA IOSTOR ; STORED SYSTEM I/O STATUS</td>
</tr>
<tr>
<td>00CD</td>
<td>4F</td>
<td>276</td>
<td>MOV C.A</td>
</tr>
<tr>
<td>00CE</td>
<td>C00000</td>
<td>E 277</td>
<td>CALL IOSET</td>
</tr>
<tr>
<td></td>
<td></td>
<td>278</td>
<td>; EXIT BACK TO ISIS-II</td>
</tr>
<tr>
<td>00D1</td>
<td>0E09</td>
<td>279</td>
<td>MVI C.EXIT ; ALL OPEN FILES ARE CLOSED</td>
</tr>
<tr>
<td>00D3</td>
<td>112000</td>
<td>D 280</td>
<td>LXi D.EBLK ; EXCEPT .CI. AND .CO. AND</td>
</tr>
<tr>
<td>00D6</td>
<td>C00000</td>
<td>E 281</td>
<td>CALL ISIS ; CONTROL PASSES BACK TO ISIS</td>
</tr>
<tr>
<td>0000</td>
<td>C</td>
<td>282</td>
<td>END START</td>
</tr>
</tbody>
</table>

PUBLIC SYMBOLS

EXTERNAL SYMBOLS

IOCHK E 0000 IOSET E 0000 ISIS E 0000 LO E 0000

USER SYMBOLS

ACTUAL D 0024 BUFFER D 202E EBLK D 0028 ENDBUF D 21BE ENDING C 00B3 EQUAL C 007C ESTAT D 002A
EXIT A 0009 FOUND C 0065 IBUF D 002D IOCHK E 0000 IOSET E 0000 IOSTOR D 002C ISIS E 0000
LLINE C 00BF LLINES C 00E4 LO E 0000 LOOK C 004A NXLTR C 006C NXTHD C 006A OBLK D 0000
OFILE D 000C OPEN A 0008 OSTAT D 000A PLTR C 00A4 PRINT C 0099 PRNTH2 C 0039 PRNTHD C 002E
PDIR C 0095 RBLK D 001A READ A 0003 RSTAT D 0026 SEARCH C 0056 SORT C 0086 START C 0000
TPBUFF D 21C8 XIT C 00CA XX D 202D

ASSEMBLY COMPLETE. NO ERRORS
Program Title: LARGE HEX FILE INTO PROM LENGTH BLOCKS CONVERTER

This program is used to prevent a "system error" which could be caused by a hex file larger than the usable memory slot available while operating the UPM. This program facilitates programming of the 2708 I Prom by breaking up a large hex file into prom length blocks.

Required Hardware: MDS 800 with 32K Ram, MDS 2DS, Console Device, UPP 102

Required Software: 32K ISIS-II

Input Parameters: This program should be located at absolute memory address of 3A00H. The Dseg portion should be located at 3B00H. I placed this program on the system floppy. The program to be worked on should be a hex file and can be in either drive 0 or 1. To invoke this program type "Block (:Fl:) hex File."

Output Results: The output results will be newly created files bearing the name of the hex file, with the extension of numerical sequence starting at 00. The extension is derived from the first address of each block.

After the hex file is blocked, then the UPM program my be called, and each block file may be read in in two's. After programming two proms, then use the offset function to offset the starting address of the next two blocks.

| Registers Modified: | Programmer:  
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM Required:</td>
<td>Steven L. Mulkey</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>Company: Zenith Electronics Corp. of MO.</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td>Address: 2500 East Kearney</td>
</tr>
<tr>
<td>Assembler/Compiler Used:</td>
<td>City: Springfield</td>
</tr>
<tr>
<td>Intellic MDS Macro Assembler</td>
<td>State: Missouri 65803</td>
</tr>
</tbody>
</table>
LOC  OBJ  SEQ  SOURCE STATEMENT

1   ;BLOCK PROGRAM WRITTEN BY
2     ;STEVEN MULKEY
3     ;ZENITH RADIO CORP OF MO
4     ;2500 EAST KARNEY
5     ;SPRINGFIELD, MO

0FE8E  6   CONV  EQU  0FE8E

0000  7   OPEN  EQU  0
0001  8   CLOSE  EQU  1
0003  9   READ  EQU  3
0004 10   WRITE  EQU  4
0009 11   EXIT  EQU  9
000C 12   ERROR  EQU  12
13   EXTRAN  ISIS

0000 310000  S  16 BEGIN: LXI SP, STACK+8 ;SET STACK
0003 3FEC  17 MVI A, BFCH
0005 320000  D  18 STA PROM ;STORE FILE EXTENSION NUMBER
0008 0E03  19 MVI C, READ
000A 116000  D  20 LXI D, RBK
000C 0D0000  E  21 CALL ISIS ;READ KEYBOARD FOR FILE TO BE BLOCKED
0010 382900  D  22 LDA STATUS
0013  B7  23 ORA A
0014 2C4900  C  24 JNZ ERR ;CHECK FOR ERROR
0017 CD0000  C  25 CALL STORE ;STORE FILE NAME OF FILE TO BE OPENED
001A B800  26 MVI C, OPEN
001C 112000  D  27 LXI D, RBK
001F C00000  E  28 CALL ISIS ;OPEN HEX FILE
0022 382A00  D  29 LDA STATUS
0025 B7  30 ORA A
0026 2C4900  C  31 JNZ ERR ;CHECK FOR ERROR
0029 C05900  C  32 CALL REED ;READ 2800H BYTES FROM HEX FILE
002C 290000  D  33 LHLD ACTUAL ;LOAD HL WITH ACTUAL BYTES READ
002F 7C  34 MVI A, H
0033 B5  35 ORA L
0034 C05100  C  36 JZ EXI ;IF HLH THEN HEX FILE HAS BEEN READ
0034 C06000  C  37 CALL LOAD ;LOAD STORED FILE NAME AND EXTENSION
0037 CD9900  C  38 CALL OPEN ;OPEN NEW FILE
003A 290000  D  39 LHLD DFT
003C 221200  D  40 SHLT CBLK ;STORE NEW FILE AFTN IN CLOSE BLOCK
0040 CD9900  C  41 CALL WRIT ;WRITE 2800H BYTES IN NEW FILE
0043 CD8900  C  42 CALL CLOS ;CLOSE NEW FILE
0046 C32900  C  43 JMP PA
0049 B000  44 ERR: MVI C, ERROR
004B 112A00  D  45 LXI D, EBLK
004E 000000  E  46 CALL ISIS ;PRINT WHAT TYPE OF ERROR
0051 0E09  47 EXI: MVI C, EXIT
0052 112E00  D  48 LXI D, EBLK
0055 000000  E  49 CALL ISIS ;EXIT BACK TO ISIS
50   ;SUBROUTINES
51
52

10/78
LOC  OBJ  SEQ  SOURCE STATEMENT

0059 0E03  53  REED:  MVI  C,READ
005B 11600  D  54  LXI  D,RBLK
005E C0000  E  55  CALL  ISIS:READ HEX FILE OR KEYBOARD
0061 3A2000  D  56  LDA  STATUS
0064 B7  57  ORA  A
0065 C26000  C  58  JNZ  RERR:JUMP IF ERROR OCCURED
0068 C9  59  RET
0069 D1  60  RERR:POP D:POP RETURN ADDRESS OFF OF STACK
006A C34000  C  61  JMP  ERR:AND JUMP TO ERROR ROUTINE
006D 218000  D  62  LOAD: LXI  H,OPA:FILE NAME STORAGE
0070 117000  D  63  LXI  D,OFILE:BUFFER FOR NEW FILE TO BE OPENED
0073 7E  64  CCC:MOV  A,M
0074 12  65  STRX  D
0075 23  66  INX  H
0076 F3  67  INY  D
0077 FFEE  68  CPI  '':TRANSFER FROM OPA TO OFILE
       69  UNTIL ' '
0079 C27000  C  70  JNZ  CCC
007C 3A200  D  71  LDA  PROM:LOAD A WITH STORED EXTENSION
007F 6804  72  ADI  04:ADD 4 TO A
0081 3A2000  D  73  STA  PROM:STORE RESULTS
0084 F5  74  PUSH  PSW:SAVE A
0085 1F  75  RAR
0086 1F  76  RAR
0087 1F  77  RAR
0088 1F  78  RAR:ROTATE A RIGHT 4 TIMES
0089 CDEEFFE  79  CALL  CONV:CONVERT A RIGHT 4 BITS TO ASCIA
008C 79  80  MOV  A,C
008D 12  81  STAX  D:STORE IN OFILE
008E 13  82  INX  D
008F F1  83  POP  PSW:RETRIEVE A
0090 CDEEFFE  84  CALL  CONV:CONVERT A RIGHT 4 BITS TO ASCIA
0093 79  85  MOV  A,C
0094 12  86  STAX  D:STORE IN OFILE
0095 13  87  INX  D
0096 3E20  88  MVI  A,’/’
0099 12  89  STAX  D:STORE TERMINATING CHAR
0099 C9  90  RET
009A 0E00  91  OPN:  MVI  C,OPEN
009C 110000  D  92  LXI  D,OBLK
009F CDE000  E  93  CALL  ISIS:OPEN FILE
00A2 3A200  D  94  LDA  STATUS
00A5 B7  95  ORA  A
00A6 C26000  C  96  JNZ  RERR:JUMP IF ERROR OCCURED
00A9 C9  97  RET
00AA 0E04  98  WRIT:  MVI  C,WRITE
00AC 110000  D  99  LXI  D,WRBLK
00AF CDE000  E  100  CALL  ISIS:WRITE INTO NEW FILE
00B2 3A200  D  101  LDA  STATUS
00B5 B7  102  ORA  A
00B6 C26000  C  103  JNZ  RERR:JUMP IF ERROR OCCURED

10/78
<table>
<thead>
<tr>
<th>LOC</th>
<th>OBJ</th>
<th>SEQ</th>
<th>SOURCE STATEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAA</td>
<td>009</td>
<td>180</td>
<td>RET</td>
</tr>
<tr>
<td>BAA</td>
<td>AERL</td>
<td>110</td>
<td>CLOSI, MVI C,CLOSE</td>
</tr>
<tr>
<td>BCA</td>
<td>111200</td>
<td>111</td>
<td>LXI D,CBLK</td>
</tr>
<tr>
<td>BCF</td>
<td>000000</td>
<td>112</td>
<td>CALL ISIS ;CLOSE FILE</td>
</tr>
<tr>
<td>BCC</td>
<td>329000</td>
<td>113</td>
<td>LDA STATUS</td>
</tr>
<tr>
<td>BCS</td>
<td>B7</td>
<td>114</td>
<td>ORA A</td>
</tr>
<tr>
<td>BCC</td>
<td>C26300</td>
<td>115</td>
<td>JNZ RERR, ;JUMP IF ERROR OCCURRED</td>
</tr>
<tr>
<td>BCD</td>
<td>009</td>
<td>116</td>
<td>RET</td>
</tr>
<tr>
<td>BBD</td>
<td>213000</td>
<td>118</td>
<td>STOREI, H,BUFFER;POINT HL TO INPUT FROM KEYBO.</td>
</tr>
<tr>
<td>BCD</td>
<td>118000</td>
<td>119</td>
<td>LXI D,OPA;SET D AT PERM. FILE NAME STO.</td>
</tr>
<tr>
<td>BDE</td>
<td>23</td>
<td>120</td>
<td>INX H</td>
</tr>
<tr>
<td>BDD</td>
<td>7E</td>
<td>121</td>
<td>MOVI A,M;TRANSFERS FROM M INTO A</td>
</tr>
<tr>
<td>BDF</td>
<td>2EE</td>
<td>122</td>
<td>CPI ',';AND CHECK FOR ','</td>
</tr>
<tr>
<td>BEE</td>
<td>C8EE00</td>
<td>123</td>
<td>JZ BBB</td>
</tr>
<tr>
<td>BDD</td>
<td>0E7</td>
<td>124</td>
<td>CPI ',';AND SPACE, IF NEITHER OCCURED</td>
</tr>
<tr>
<td>BDF</td>
<td>C8EE00</td>
<td>125</td>
<td>JZ BBB</td>
</tr>
<tr>
<td>BDD</td>
<td>92</td>
<td>126</td>
<td>STAX D;STO CHAR AND JMP TO BB</td>
</tr>
<tr>
<td>BDF</td>
<td>23</td>
<td>127</td>
<td>INX H</td>
</tr>
<tr>
<td>BDD</td>
<td>13</td>
<td>128</td>
<td>INX D</td>
</tr>
<tr>
<td>BDF</td>
<td>C31000</td>
<td>129</td>
<td>JMP BB</td>
</tr>
<tr>
<td>BDF</td>
<td>3EE00</td>
<td>130</td>
<td>BBB, MVI A,',';STORE ',' IN PERM STOR</td>
</tr>
<tr>
<td>BDF</td>
<td>12</td>
<td>131</td>
<td>STAX D</td>
</tr>
<tr>
<td>BDF</td>
<td>005</td>
<td>132</td>
<td>RET</td>
</tr>
<tr>
<td></td>
<td></td>
<td>133</td>
<td>DSEG</td>
</tr>
<tr>
<td>BDE</td>
<td>0000</td>
<td>135</td>
<td>OBLK: DW OAFI;POINTER FOR OPEN AFTN</td>
</tr>
<tr>
<td>BDE</td>
<td>0000</td>
<td>136</td>
<td>DW OFILE;POINTER FOR FILE TO BE OPENED</td>
</tr>
<tr>
<td>BDE</td>
<td>0000</td>
<td>137</td>
<td>DW 2;OPEN FILE FOR WRITE</td>
</tr>
<tr>
<td>BDE</td>
<td>0000</td>
<td>138</td>
<td>DW 0;NO ECHO</td>
</tr>
<tr>
<td>BDE</td>
<td>2000</td>
<td>139</td>
<td>DW STATUS;POINTER FOR STATUS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>140</td>
<td>OAFI</td>
</tr>
<tr>
<td>BDE</td>
<td>0000</td>
<td>141</td>
<td>OBLK: DS 2;STORAGE FOR OPENED FILE AFTN</td>
</tr>
<tr>
<td>BDE</td>
<td>0000</td>
<td>142</td>
<td>DW BUFFER;POINTER FOR STORAGE AREA</td>
</tr>
<tr>
<td>BDE</td>
<td>0000</td>
<td>143</td>
<td>DS 2;STORAGE FOR THE NO. OF BYTE TO BE WRITTEN</td>
</tr>
<tr>
<td>BDE</td>
<td>2000</td>
<td>144</td>
<td>DW STATUS;POINTER FOR STATUS</td>
</tr>
<tr>
<td>BDE</td>
<td>0000</td>
<td>145</td>
<td>DS 2;AFTN OF FILE TO BE CLOSED</td>
</tr>
<tr>
<td>BDE</td>
<td>2000</td>
<td>146</td>
<td>DW STATUS;POINTER FOR STATUS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>147</td>
<td>RAFT</td>
</tr>
<tr>
<td>BDE</td>
<td>0100</td>
<td>148</td>
<td>RBLK: DW 1;READ KEYBOARD</td>
</tr>
<tr>
<td>BDE</td>
<td>0000</td>
<td>149</td>
<td>DW BUFFER;POINTER TO STORAGE AREA</td>
</tr>
<tr>
<td>BDE</td>
<td>0000</td>
<td>150</td>
<td>DW 2880;BYTES TO BE READ</td>
</tr>
<tr>
<td>BDE</td>
<td>0000</td>
<td>151</td>
<td>DW Actual;HOW MANY WERE ACTUALLY READ</td>
</tr>
<tr>
<td>BDE</td>
<td>2000</td>
<td>152</td>
<td>DW STATUS;POINTER FOR STATUS</td>
</tr>
<tr>
<td>BDE</td>
<td>1600</td>
<td>153</td>
<td>PBLK: DW RAFT;AFTN OF HEX FILE</td>
</tr>
<tr>
<td>BDE</td>
<td>0000</td>
<td>154</td>
<td>DW BUFFER;POINTER TO STORAGE AREA</td>
</tr>
<tr>
<td>BDE</td>
<td>0100</td>
<td>155</td>
<td>DW 1;OPEN FOR READ OPERATION</td>
</tr>
<tr>
<td>BDE</td>
<td>0000</td>
<td>156</td>
<td>DW 0;NO ECHO</td>
</tr>
<tr>
<td>BDE</td>
<td>2000</td>
<td>157</td>
<td>DW STATUS;POINTER TO STATUS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>158</td>
<td>EBLK</td>
</tr>
<tr>
<td>BDE</td>
<td>0000</td>
<td>159</td>
<td>DS 2;STORAGE FOR STATUS</td>
</tr>
<tr>
<td>BDE</td>
<td>2000</td>
<td>160</td>
<td>DW STATUS;POINTER FOR STATUS</td>
</tr>
<tr>
<td>BDE</td>
<td>2000</td>
<td>161</td>
<td>XBLK: DW STATUS;POINTER FOR STATUS</td>
</tr>
<tr>
<td>BDE</td>
<td>0480</td>
<td>162</td>
<td>DS 2880;DEFINED STORAGE OF 2880H BYTES</td>
</tr>
</tbody>
</table>

10/78
PUBLIC SYMBOLS

EXTERNAL SYMBOLS
ISIS E 0000

USER SYMBOLS
AA C 0029 ACTUAL D 000E BB C 0001 BBB C 00E2 BEGIN C 0000 BUFFER D 0030 CBLK D 0012
cccc C 0073 CLOS C 008A CLOSE A 0001 CONV A F00E EBLK D 002A ERR C 0049 ERROR A 000C
EXIT C 0051 EXIT A 0089 ISIS E 0000 LOAD C 005D OAFD D 000A OBLK D 0000 OFILE D 0070
OPA D 0089 OPEN R 0000 OPA C 009A PSLK D 0020 PROM D 0082 RAFT D 0016 RBLK D 0016
READ A 0003 REED C 0059 RERR C 0069 STATUS D 002A STORE C 00CA WBK D 000A WRIT C 00AA
WRITE A 0004 XBLK D 002E

ASSEMBLY COMPLETE, NO ERRORS
PROM BUS MAPPER

To Relocate Hex data Bits, prior to programming a PROM which is to be used in a system where the address and data lines between µP and PROM are not directly related.

MDS 800 System

Program + console input and console output routines

Hex data at address 1000H upwards
Address and data interconnect information input on console

Relocated Hex data at address 2000H upwards

Registers Modified:
- All

RAM Required:
- 16K

ROM Required:

Maximum Subroutine Nesting Level:
- 2

Assembler/Compiler Used:
- 8080 Macro Ass. V.2.2

Programmer:
- R. A. Stevenson

Company:
- G.E.T. Telecoms. Ltd.

Address:
- Wiminhank Road

City:
- Aycliffe Ind. Estate

State:
- Durham England
FLEXIBLE NAME LIST MANAGER

To store and retrieve variable length names in a symbol table, together with associated attributes.

8080/8085 + memory

None

HL = (BLOCK)

BLOCK: DB INDEX
       DB ATTRIBUTE
       DB NO, CHARS
       DB 'STRING'

INDEX & ATTRIBUTE given 'STRING' & NO. CHARs name present
INDEX given 'STRING' & NO. CHARs name not present
'STRING' & ATTRIBUTE given INDEX name present
*NAME NOT FOUND* ERROR
*TABLE FULL* ERROR

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>A &amp; PSW</td>
<td>M. G. Dineley</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RAM Required:</th>
<th>Company:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dependant on No. of names &amp; size</td>
<td>Dept. E. E. &amp; E. UMIST</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ROM Required:</th>
<th>Address:</th>
</tr>
</thead>
<tbody>
<tr>
<td>256K</td>
<td>Sackville St.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Maximum Subroutine Nesting Level:</th>
<th>City:</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Manchester England</td>
</tr>
</tbody>
</table>

Assembler/Compiler Used:

ISIS-II Macro Assembler

© Intel Corporation, 1976
IDENT1 - FRONT PAGE IDENTIFIER PRINT PROGRAM

The program prompts the operator to furnish the date, disk identification, filename and programmer's name. The program then composes and prints an identification front page as a cover page for the program listing which follows it.

INTELLEC MDS with floppy disk (dual) drive and line printer.

ISIS-II

The program does not interact with other programs.

Registers Modified:

All

Programmer:

Phil Greenberg

RAM Required:

3950 = 0F7EH

Company:

Conrac Corp. - Plant 3

ROM Required:

None

Address:

32 Fairfield Place

Maximum Subroutine Nesting Level:

N/A

City:

West Caldwell

Assembler/Compiler Used:

INTELLEC MDS Macro Assembler

State:

New Jersey 07006

© Intel Corporation, 1976

10/78
CERROR - PL/M-80 COMPILER ERROR DISPLAY PROGRAM

Program displays PL/M-80 Compiler Errors
Assembled program must be linked with System.LIB and located above
ISIS-II. Program resides on system diskette.

MDS 800, DOS-2DS, TTY or CRT

ISIS-II

Command CERROR :FX:Name.LST

All PL/M Compiler errors are copied on :CO:

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Programmer:</th>
<th>Company:</th>
</tr>
</thead>
<tbody>
<tr>
<td>All registers</td>
<td>Prof. Ing. Dalibor Nemec</td>
<td>VSE- Dept. of Technology</td>
</tr>
<tr>
<td>RAM Required:</td>
<td></td>
<td>Address:</td>
</tr>
<tr>
<td>13B H</td>
<td></td>
<td>Pelhrimovska 9, Praha 4,</td>
</tr>
<tr>
<td>ROM Required:</td>
<td></td>
<td>City:</td>
</tr>
<tr>
<td>None</td>
<td></td>
<td>Ceschoslov</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>State:</td>
</tr>
<tr>
<td>Assembler/Compiler Used:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8080/8085 Macro Assembler</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

© Intel Corporation, 1976

10/78
LOC   OBJ   SEQ   SOURCE STATEMENT

1 ;
2 $  TITLE('PL/M-88 COMPILER ERRORS')
3   NAME CERROR
4 ;
0000 5  OPEN EOU 0
0005 6  READ EOU 3
0009 7  EXIT EOU 9
000C 8  ERROR EOU 12
000A 9  LF EOU 00H
000D 10  OR EOU 00H
0809 11  CI EOU 0F8004 ; MONITOR ROUTINE ADDRESS
12 ;
13   EXTRN ISIS
14 ;
15   CSEG
0000 310000 S 16 BEGIN: LXI SP,STACK ; ADJUST STACK POINTER
0003 0B03 17  MVI C,READ ; READ NAME OF LIST
0005 110000 D 18  LXI D,RLBK ; FILE FROM .CI:
0008 CD400 C 19  CALL SYSTEM ; INTO BUFFER
000B 0E00 20  MVI C,OPEN ; OPEN LIST FILE
000D 110000 D 21  LXI D,OBK ; FOR READING
0010 CD400 C 22  CALL SYSTEM
0013 C05F00 C 23  CALL READBL ; READ BLOCK INTO BUFFER
0016 CD400 C 24  LOOP: CALL CHAR ; PICK UP CHAR FROM BUFFER
0019 79 25  MOV A,C
001A FE0A 26  CPI LF ; (LF) LAST CHAR ON LINE, SEQUENCE: (OR)-(LF)
001C C21680 C 27  JNZ LOOP ; LOOP UNTIL LAST CHAR
001F CD400 C 28  LOOP1: CALL CHAR ; PICK UP NEXT CHAR
0022 79 29  MOV A,C
0023 FE0A 30  CPI '*' ; BEGINNING OF COMPILER ERROR MESSAGE
0025 C0300 C 31  JZ DISPATCH ; DISPLAY ERR MESSAGE
0028 FE0D 32  CPI CR ; LINE ENDS (OR) SEQUENCE (CLF)(CR)
002A C21680 C 33  JNZ LOOP ; LOOP UNTIL LAST CHAR
002C C14F00 C 34  JMP LOOP1 ; TEST NEXT LINE
002E CD400 C 35  DISPATCH: CALL CO ; DISPLAY FIRST CHAR OF MESSAGE
0033 CD400 C 36  CALL CHAR ; PICK UP SECOND CHAR
0036 79 37  MOV A,C
0037 FE0A 38  CPI LF ; TEST IF END OF LINE
0039 C23000 C 39  JNZ DISPATCH ; LOOP UNTIL END OF COMPILER MESSAGE
003C CD400 C 40  CALL CO ; DISPLAY LAST CHAR ON LINE
003F C14F00 C 41  JMP LOOP1 ; TEST NEW LINE
42 ;
0042 23 43  CHA E: INK H ; ADDRESS NEXT CHAR
0043 4E 44  MOV C, M ; PICK UP CHAR FROM BUFFER
0044 85 45  DCR B ; IF BUFFER EMPTY READ
0045 C05F00 C 46  CZ READBL ; NEW BLOCK
0048 C9 47  RET
48 ;
0049 CD8000 E 49  SYSTEM: CALL ISIS
004C 32200 D 50  LDA STATUS ; TEST ERROR STATUS
004F 87 51  ORA A
0053 C27600 C 52  JNZ ERR ; BRANCH TO ERROR ROUTINE

10/78
<table>
<thead>
<tr>
<th>LOC</th>
<th>OBJ</th>
<th>SEQ</th>
<th>SOURCE STATEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0053 C9</td>
<td>53</td>
<td></td>
<td>RET</td>
</tr>
<tr>
<td>0054 CD</td>
<td>55</td>
<td>HLNE:</td>
<td>MVI C,CR ; INSERT CR INTO DISPLAY BUFF</td>
</tr>
<tr>
<td>0056 CD8F8</td>
<td>56</td>
<td></td>
<td>CALL CO ; DISPLAY CHAR</td>
</tr>
<tr>
<td>0059 BE8A</td>
<td>57</td>
<td></td>
<td>MVI C,LF ; INSERT LF INTO DISPLAY BUFF</td>
</tr>
<tr>
<td>005B CD8F8</td>
<td>58</td>
<td></td>
<td>CALL CO ; DISPLAY CHAR</td>
</tr>
<tr>
<td>005E C9</td>
<td>59</td>
<td></td>
<td>RET</td>
</tr>
<tr>
<td>005F CD</td>
<td>61</td>
<td>READBL:</td>
<td>PUSH B ; SAVE LAST CHAR FROM LAST BLOCK IN C REG</td>
</tr>
<tr>
<td>0060 BE83</td>
<td>62</td>
<td></td>
<td>MVI C,READ ; READ NEW BLOCK FROM</td>
</tr>
<tr>
<td>0062 111400 D</td>
<td>63</td>
<td></td>
<td>LXI D,RBLK1 ; LIST FILE INTO BUFFER</td>
</tr>
<tr>
<td>0065 CD4900 C</td>
<td>64</td>
<td></td>
<td>CALL SYSTEM</td>
</tr>
<tr>
<td>0068 290C00 D</td>
<td>65</td>
<td></td>
<td>LHLD ACTUAL ; IF ALL BLOCKS OF LIST</td>
</tr>
<tr>
<td>006B 7C</td>
<td>66</td>
<td></td>
<td>MOV A,H ; FILE ALREADY READ</td>
</tr>
<tr>
<td>006C 85</td>
<td>67</td>
<td></td>
<td>ORA L ; THEN ACTUAL=0</td>
</tr>
<tr>
<td>006D CAE000 C</td>
<td>68</td>
<td></td>
<td>JZ DONE</td>
</tr>
<tr>
<td>0070 C1</td>
<td>69</td>
<td></td>
<td>POP B ; RESTORE C REG</td>
</tr>
<tr>
<td>0071 47</td>
<td>70</td>
<td></td>
<td>MOV B,A ; NUMBER OF CHAR READ INTO BUFFER</td>
</tr>
<tr>
<td>0072 212100 D</td>
<td>71</td>
<td></td>
<td>LXI H,BUFFER-1 ; BUFFER ADDRESSING</td>
</tr>
<tr>
<td>0075 C9</td>
<td>72</td>
<td></td>
<td>RET</td>
</tr>
<tr>
<td>0076 BE8C</td>
<td>74</td>
<td>ERR:</td>
<td>MVI C,ERROR ; DISPLAY SYSTEM ERROR NUMBER</td>
</tr>
<tr>
<td>0078 111E80 D</td>
<td>75</td>
<td></td>
<td>LXI D,EBLK</td>
</tr>
<tr>
<td>007B CD0000 E</td>
<td>76</td>
<td></td>
<td>CALL ISIS</td>
</tr>
<tr>
<td>007E CD5400 C</td>
<td>78</td>
<td>DONE:</td>
<td>CALL HLNE ; SEPARATE WITH ONE LINE</td>
</tr>
<tr>
<td>0081 BE89</td>
<td>79</td>
<td></td>
<td>MVI C,EXIT ; CLOSE LIST FILE</td>
</tr>
<tr>
<td>0083 111E80 D</td>
<td>80</td>
<td></td>
<td>LXI D,DLBK ; AND RETURN TO SUPERVISOR</td>
</tr>
<tr>
<td>0086 CD0000 E</td>
<td>81</td>
<td></td>
<td>CALL ISIS</td>
</tr>
<tr>
<td>0087</td>
<td>83</td>
<td></td>
<td>DSEG</td>
</tr>
<tr>
<td>0090 0100</td>
<td>84</td>
<td>RBLK:</td>
<td>DW 1 ; READ CONSOLE</td>
</tr>
<tr>
<td>0092 2200 D</td>
<td>85</td>
<td></td>
<td>DW BUFFER</td>
</tr>
<tr>
<td>0094 7A00</td>
<td>86</td>
<td></td>
<td>DW 122</td>
</tr>
<tr>
<td>0096 9C00 D</td>
<td>87</td>
<td></td>
<td>DW ACTUAL</td>
</tr>
<tr>
<td>0098 2000 D</td>
<td>88</td>
<td></td>
<td>DW STATUS</td>
</tr>
<tr>
<td>0099</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>009A 1400 D</td>
<td>90</td>
<td>OBLK:</td>
<td>DW AFTN</td>
</tr>
<tr>
<td>009C 2200 D</td>
<td>91</td>
<td></td>
<td>DW BUFFER ; FILE TO BE OPEN</td>
</tr>
<tr>
<td>009E 0100</td>
<td>92</td>
<td></td>
<td>DW 1 ; FOR READING</td>
</tr>
<tr>
<td>00A0 0000</td>
<td>93</td>
<td></td>
<td>DW 0</td>
</tr>
<tr>
<td>00A2 2000 D</td>
<td>94</td>
<td></td>
<td>DW STATUS</td>
</tr>
<tr>
<td>00A3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00A4</td>
<td>96</td>
<td>RBLK1:</td>
<td></td>
</tr>
<tr>
<td>00A6 2200 D</td>
<td>97</td>
<td></td>
<td>AFTN:</td>
</tr>
<tr>
<td>00A8 7A00</td>
<td>99</td>
<td></td>
<td>DW 122</td>
</tr>
<tr>
<td>00A9 9C00 D</td>
<td>100</td>
<td></td>
<td>DW ACTUAL</td>
</tr>
<tr>
<td>00AC 2000 D</td>
<td>101</td>
<td></td>
<td>DW STATUS</td>
</tr>
<tr>
<td>00A5</td>
<td>102</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00A6</td>
<td>103</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00A8 2000 D</td>
<td>104</td>
<td>EBLK:</td>
<td>DW STATUS</td>
</tr>
<tr>
<td>00A9</td>
<td>105</td>
<td></td>
<td>STATUS:</td>
</tr>
<tr>
<td>00AA</td>
<td>106</td>
<td>BUFFER:</td>
<td>DS 122 ; LENGTH OF BUFFER</td>
</tr>
<tr>
<td>00AB</td>
<td>107</td>
<td></td>
<td>ACTUAL:</td>
</tr>
</tbody>
</table>
LOC OBJ SEQ SOURCE STATEMENT

100 STKLN 10 ; DEPTH OF STACK
109;
0000 C 110 END BEGIN

PUBLIC SYMBOLS

EXTERNAL SYMBOLS
ISIS E 0000

USER SYMBOLS
ACTUAL D 009D AFTN D 0014 BEGIN C 0000 BUFFER D 0022 CHAR C 0042 CO A F809 CR A 0000
DBLK D 001E DISCH C 0030 DONE C 007E EBLK D 001E ERR C 0076 ERROR A 000C EXIT A 0009
ISIS E 0000 LF A 000A LOOP C 0016 LOOP1 C 001F NLINE C 0054 OBLK D 000A OPEN A 0000
RBLK D 0000 RBLK2 D 0014 READ A 0003 READBL C 005F STATUS D 0020 SYSTEM C 0049

ASSEMBLY COMPLETE, NO ERRORS
IDENT1 - FRONT PAGE IDENTIFIER PRINT PROGRAM

The program prompts the operator to furnish the date, disk identification, filename and programmer's name. The program then composes and prints an identification front page as a cover page for the program listing which follows it.

INTELLEC MDS with floppy disk (dual) drive and line printer.

ISIS-II

The program does not interact with other programs.

| Registers Modified: | All |
| RAM Required: | 3950 = 0F7EH |
| ROM Required: | None |
| Maximum Subroutine Nesting Level: | N/A |
| Assembler/Compiler Used: | INTELLEC MDS Macro Assembler |
| Programmer: | Phil Greenberg |
| Company: | Conrac Corp. - Plant 3 |
| Address: | 32 Fairfield Place |
| City: | West Caldwell |
| State: | New Jersey 07006 |
MON830 - Monitor for ISBC 80/30

2K Debug monitor for ISBC 80/30. Provides simple memory and register display and modification commands as well as program execution with breakpoints. The monitor also provides the user with routines for performing console I/O and paper tape I/O.

ISBC 80/30 plus console device

None

None. At system power up, monitor will wait for a string of ASCII 'u's to be typed on the console device, to determine the baud rate of the terminal. After this, a banner is output and command mode is entered. Commands are implemented for memory/ register display/ modification, execution with and without single stepping and breakpoints, and paper tape program load/save.

Available on non-system diskette only for $35.00 (source & object code included)

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RAM Required:</th>
<th>96</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROM Required:</td>
<td>2040</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Maximum Subroutine Nesting Level:</th>
<th>8</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Assembler/Compiler Used:</th>
<th>MDS Macro Assembler</th>
</tr>
</thead>
</table>

© Intel Corporation, 1978
12/78

98-034E
4-679
UPI-41 8-DIGIT LED DISPLAY CONTROLLER

This program uses the UPI-41 as a LED Display Controller which scans and refreshes eight multiplexed seven-segment LED Displays. The characters are defined by input from the master microprocessor in the form of an eight bit word per digit-character selection. A total of 32 different alphanumeric characters are available for display. Applications; clock or temperature readout, various message displays.

All information pertaining to required hardware, software, input parameters and output results are fully documented in the Intel application note AP 41 ("INTRODUCTION TO THE UPI-41A") and the listing for this program.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
</tr>
</thead>
<tbody>
<tr>
<td>A, RB1, R0, R2, R3, R7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RAM Required:</th>
</tr>
</thead>
<tbody>
<tr>
<td>(within UPI-41)</td>
</tr>
<tr>
<td>EH (see AP 41)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ROM Required:</th>
</tr>
</thead>
<tbody>
<tr>
<td>(within UPI-41)</td>
</tr>
<tr>
<td>73H (see AP 41)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Maximum Subroutine Nesting Level:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Assembler/Compiler Used:</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASM48 MOD41</td>
</tr>
</tbody>
</table>

Programmer: Robin J. Jigour
Company: Intel
Address: 
City: 
State: 

© Intel Corporation, 1976
12/78

Ref. #AB138
98-034D
4-680
LOC OBJ      SEQ      SOURCE STATEMENT
1 ;                      ********************************************
2 ;    *  UPI-41 8-DIGIT LED DISPLAY CONTROLLER  *
3 ;                      ********************************************
4 ;
5 ;
6 ;
7 ; THIS PROGRAM USES THE UPI-41 AS A LED DISPLAY CONTROLLER
8 ; WHICH SCANS AND REFRESSES EIGHT SEVEN-SEGMENT LED DISPLAYS.
9 ; THE CHARACTERS ARE DEFINED BY INPUT FROM A MASTER CPU IN THE
10 ; FORM OF ONE EIGHT BIT WORD PER DIGIT-CHARACTER SELECTION.
11 ;
12 ;
13 ;
14 ;********************************************
15 ;
16 ; REGISTER DEFINITIONS:
17 ;  REGISTER      RB1               RB0
18 ;  --------      -----              ----
19 ;  R0        DISPLAY MAP POINTER   NOT USED
20 ;  R1        NOT USED              NOT USED
21 ;  R2        DATA WORD AND CHARACTER STORAGE NOT USED
22 ;  R3        DIGIT COUNTER         NOT USED
23 ;  R4        NOT USED              NOT USED
24 ;  R5        NOT USED              NOT USED
25 ;  R6        NOT USED              NOT USED
26 ;  R7        ACCUMULATOR STORAGE   NOT USED
27 ;********************************************
28 ;
29 ; PORT PIN DEFINITIONS:
30 ;  PIN      PORT 1 FUNCTION      PORT 2 FUNCTION
31 ;  ----      ------------          -----------
32 ;  PA0-7    SEGMENT DRIVER CONTROL DIGIT DRIVER CONTROL
33 ;
34 #EJECT
<table>
<thead>
<tr>
<th>LOC</th>
<th>OBJ</th>
<th>SEQ</th>
<th>SOURCE STATEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>35</td>
<td></td>
<td></td>
<td>;*****************************************************************</td>
</tr>
<tr>
<td>36</td>
<td></td>
<td></td>
<td>; DISPLAY DATA WORD BIT DEFINITION:</td>
</tr>
<tr>
<td>37</td>
<td></td>
<td></td>
<td>; BIT FUNCTION</td>
</tr>
<tr>
<td>38</td>
<td></td>
<td></td>
<td>; --------</td>
</tr>
<tr>
<td>39</td>
<td></td>
<td></td>
<td>; 0-4 CHARACTER SELECT</td>
</tr>
<tr>
<td>40</td>
<td></td>
<td></td>
<td>; 5-7 DIGIT SELECT</td>
</tr>
<tr>
<td>41</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>42</td>
<td></td>
<td></td>
<td>; CHARACTER SELECT:</td>
</tr>
<tr>
<td>43</td>
<td></td>
<td></td>
<td>; D4 D3 D2 D1 D0 CHARACTER</td>
</tr>
<tr>
<td>44</td>
<td></td>
<td></td>
<td>; 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>45</td>
<td></td>
<td></td>
<td>; 0 0 0 0 0 1 1</td>
</tr>
<tr>
<td>46</td>
<td></td>
<td></td>
<td>; 0 0 0 1 0 2</td>
</tr>
<tr>
<td>47</td>
<td></td>
<td></td>
<td>; 0 0 0 1 1 3</td>
</tr>
<tr>
<td>48</td>
<td></td>
<td></td>
<td>; 0 0 1 0 0 4</td>
</tr>
<tr>
<td>49</td>
<td></td>
<td></td>
<td>; 0 0 1 0 1 5</td>
</tr>
<tr>
<td>50</td>
<td></td>
<td></td>
<td>; 0 0 1 1 0 6</td>
</tr>
<tr>
<td>51</td>
<td></td>
<td></td>
<td>; 0 0 1 1 1 7</td>
</tr>
<tr>
<td>52</td>
<td></td>
<td></td>
<td>; 0 1 0 0 0 8</td>
</tr>
<tr>
<td>53</td>
<td></td>
<td></td>
<td>; 0 1 0 0 1 9</td>
</tr>
<tr>
<td>54</td>
<td></td>
<td></td>
<td>; 0 1 0 1 0 A</td>
</tr>
<tr>
<td>55</td>
<td></td>
<td></td>
<td>; 0 1 0 1 1 B</td>
</tr>
<tr>
<td>56</td>
<td></td>
<td></td>
<td>; 0 1 1 0 0 C</td>
</tr>
<tr>
<td>57</td>
<td></td>
<td></td>
<td>; 0 1 1 0 1 D</td>
</tr>
<tr>
<td>58</td>
<td></td>
<td></td>
<td>; 0 1 1 1 0 E</td>
</tr>
<tr>
<td>59</td>
<td></td>
<td></td>
<td>; 0 1 1 1 1 F</td>
</tr>
<tr>
<td>60</td>
<td></td>
<td></td>
<td>; 1 0 0 0 0 G</td>
</tr>
<tr>
<td>61</td>
<td></td>
<td></td>
<td>; 1 0 0 0 1 H</td>
</tr>
<tr>
<td>62</td>
<td></td>
<td></td>
<td>; 1 0 0 1 0 I</td>
</tr>
<tr>
<td>63</td>
<td></td>
<td></td>
<td>; 1 0 1 0 0 J</td>
</tr>
<tr>
<td>64</td>
<td></td>
<td></td>
<td>; 1 0 1 0 1 L</td>
</tr>
<tr>
<td>65</td>
<td></td>
<td></td>
<td>; 1 0 1 1 0 N</td>
</tr>
<tr>
<td>66</td>
<td></td>
<td></td>
<td>; 1 0 1 1 1 O</td>
</tr>
<tr>
<td>67</td>
<td></td>
<td></td>
<td>; 1 1 0 0 0 P</td>
</tr>
<tr>
<td>68</td>
<td></td>
<td></td>
<td>; 1 1 0 0 1 Q</td>
</tr>
<tr>
<td>69</td>
<td></td>
<td></td>
<td>; 1 1 0 1 0 T</td>
</tr>
<tr>
<td>70</td>
<td></td>
<td></td>
<td>; 1 1 0 1 1 U</td>
</tr>
<tr>
<td>71</td>
<td></td>
<td></td>
<td>; 1 1 1 0 0 V</td>
</tr>
<tr>
<td>72</td>
<td></td>
<td></td>
<td>; 1 1 1 0 1 W</td>
</tr>
<tr>
<td>73</td>
<td></td>
<td></td>
<td>; 1 1 1 1 0 X</td>
</tr>
<tr>
<td>74</td>
<td></td>
<td></td>
<td>; 1 1 1 1 1 &quot;BLANK&quot;</td>
</tr>
<tr>
<td>75</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>76</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>77</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>78</td>
<td></td>
<td></td>
<td>; DIGIT SELECT:</td>
</tr>
<tr>
<td>79</td>
<td></td>
<td></td>
<td>; D7 D6 D5 DIGIT NUMBER</td>
</tr>
<tr>
<td>80</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>81</td>
<td></td>
<td></td>
<td>; 0 0 0 1</td>
</tr>
<tr>
<td>82</td>
<td></td>
<td></td>
<td>; 0 0 1 2</td>
</tr>
<tr>
<td>83</td>
<td></td>
<td></td>
<td>; 0 1 0 3</td>
</tr>
<tr>
<td>84</td>
<td></td>
<td></td>
<td>; 0 1 1 4</td>
</tr>
<tr>
<td>85</td>
<td></td>
<td></td>
<td>; 1 0 0 5</td>
</tr>
<tr>
<td>86</td>
<td></td>
<td></td>
<td>; 1 0 1 6</td>
</tr>
<tr>
<td>87</td>
<td></td>
<td></td>
<td>; 1 1 0 7</td>
</tr>
<tr>
<td>88</td>
<td></td>
<td></td>
<td>; 1 1 1 8</td>
</tr>
<tr>
<td>89</td>
<td></td>
<td></td>
<td>;***************************************************************************</td>
</tr>
<tr>
<td>90</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
LOC OBJ SEQ SOURCE STATEMENT

90 ;*************************************************************************
91 EQUATES
92 ;THE FOLLOWING CODE DESIGNATES "TIME" AS A VARIABLE THIS
93 ;ADJUSTS THE AMOUNT OF CYCLES THE TIMER COUNTS BEFORE
94 ;A TIMER INTERRUPT OCCURS AND REFRESH THE DISPLAY APPROXIMATELY
95 ;90 TIMES PER SECOND.
96 FFF1
97 ;*************************************************************************
98 INTERRUPT BRANCHING
99 ;THIS PORTION OF MEMORY IS DEDICATED FOR USE OF RESET AND
100 INTERRUPT BRANCHING WHEN THE INTERRUPTS ARE ENABLED THE
101 CODE AT THE FOLLOWING DESIGNATED SPOTS ARE EXECUTED WHEN A
102 ;RESET OR A INTERRUPT OCCURS.
103 0000
104 0000 0009
105 0002 00
106 0003 0430
107 0005 00
108 0006 00
109 0007 041F
110 ;*************************************************************************
111 ;INITIALIZATION
112 ;THE FOLLOWING CODE SETS UP THE UPI-41 AND DISPLAY HARDWARE
113 INTO OPERATIONAL FORMAT. THE DISPLAY IS TURNED OFF. THE DISPLAY
114 ;MAP IS FILLED WITH "BLANK" CHARACTERS, THE TIMER SET AND THE
115 ;INTERRUPTS ARE ENABLED
116 0000 05
117 0000 0B08
118 0000 8838
119 0000 E2FF
120 0000 00
121 0000 08
122 0000 18
123 0000 28
124 0000 20E
125 0000 B000
126 0000 23F1
127 0000 00
128 0000 00
129 0010 05
130 0010 041C
131 ;*************************************************************************
132 ;USER PROGRAM
133 ;A USERS PROGRAM WOULD INITIALIZE AT THIS POINT. THE FOLLOWING
134 ;CODE IS USED TO TAKE THE PLACE OF A POSSIBLE USER PROGRAM.
135 ;
136 0010 041C
137 LOOP JMF LOOP ;WAIT FOR INTERRUPT
138 ;*************************************************************************
LOC OBJ

140 ;************************************************************************************
141 ; DISPLAY ROUTINE
142 ; THIS PORTION OF THE PROGRAM IS AN INTERRUPT ROUTINE WHICH IS
143 ; ACTED UPON WHEN THE TIMER COUNT IS COMPLETED. THE ROUTINE UPDATES
144 ; ONE DISPLAY DIGIT FROM THE DISPLAY MAP PER INTERRUPT SEQUENTIALLY.
145 ; THEREFORE, TIMER INTERRUPTS WILL HAVE REFRESHED THE ENTIRE DISPLAY.
146 ; REGISTER BANK 1 IS SELECTED AND THE ACCUMULATOR IS SAVED UPON
147 ; ENTERING THE ROUTINE. ONCE THE DISPLAY HAS BEEN REFRESHED THE TIMER
148 ; IS RESET AND THE ACCUMULATOR AND PRE-INTERRUPT REGISTER BANK 1 IS RESTORED.
149 ;

001F D5 150 DISPLA SEL R81 ; REGISTER BANK 1
0020 AF 151 MOV R7.A ; SAVE ACCUMULATOR
0021 9A88 152 ORL P2.080H ; TURN DIGIT DRIVERS OFF
0023 FB 153 MOV R.R3 ; DIGIT COUNTER TO ACCUMULATOR
0024 4338 154 ORL A.038H ; "OR" TO GET DISPLAY MAP ADDRESS
0026 FB 155 MOV R.R8.A ; DISPLAY MAP POINTER
0027 FB 156 MOV A.000H ; GET CHARACTER FROM DISPLAY MAP
0028 39 157 OUTL P1.A ; OUTPUT CHARACTER TO SEGMENT DRIVERS
0029 FB 158 MOV A.R3 ; DIGIT COUNTER VALUE TO ACCUMULATOR
002A 3A 159 OUTL P2.A ; OUTPUT TO DIGIT DRIVERS
002B 1B 160 INC R3 ; INCREMENT DIGIT COUNTER
002C D97 161 XRL A.007H ; CHECK IF AT LAST DIGIT
002E 9632 162 JNZ SETIME ; RESET TIMER IN NOT LAST DIGIT
0030 8000 163 MOV R3.000H ; RESET DIGIT COUNTER
0032 23F1 164 SETIME MOV A.0TIME ; TIMER VALUE
0034 62 165 MOV T.A ; LOAD TIMER
0035 55 166 STRT T ; START TIMER
0036 FF 167 MOV A.R7 ; RESTORE ACCUMULATOR
0037 93 168 RETR ; RETURN

169 ;************************************************************************************
170 #REJECT
LOC OBJ SER SOURCE STATEMENT

171 ;
172 ;*******************************************************************************************
173 ; INPUT CHARACTER AND DIGIT ROUTINE
174 ; THIS PORTION OF THE PROGRAM IS AN INTERRUPT ROUTINE WHICH
175 ; IS ACTED UPON WHEN THE IBF BIT IS SET. THE ROUTINE GETS THE
176 ; DISPLAY DATA WORD FROM THE DDB AND DEFINES BOTH THE DIGIT AND
177 ; THE CHARACTER TO BE DISPLAYED. THIS IS DONE BY MEANS OF A
178 ; CHARACTER LOOK-UP TABLE AND A DISPLAY MAP FOR DIGIT AND CHARACTER
179 ; LOCATION. SPECIAL CONSIDERATION IS TAKEN FOR A DECIMAL POINT WHICH IS
180 ; SIMPLY ADDED TO THE EXISTING CHARACTER IN THE DISPLAY MAP. REGISTER
181 ; BANK 1 IS SELECTED AND THE ACCUMULATOR IS SAVED UPON ENTERING
182 ; THE ROUTINE. ONCE THE DATA WORD HAS BEEN FULLY DEFINED THE ACCUMULATOR
183 ; AND THE PRE-INTERUPT REGISTER BANK IS RESTORED.
184 ;
0038 D5 185 INPUT: SEL RB1 ;REGISTER BANK 1
0039 AF 186 MOV R7,A ;SAVE ACCUMULATOR
003A 22 187 IN A,DDB ;GET DATA
0038 AA 188 MOV R2.A ;SAVE DATA WORD
003C 47 189 SWAP A ;DEFINE DIGIT LOCATION
003D 77 190 RR A ;
003E 5387 191 ANL A,087H ;
0040 4338 192 ORL A,130H ;
0042 A8 193 MOV R0.A ;DIGIT LOCATION IN DIGIT POINTER
0043 FA 194 MOV R.A2 ;SAVE DATA WORD TO ACCUMULATOR
0044 531F 195 ANL A,01FH ;DEFINE CHARACTER LOOK-UP-TABLE LOC.
0046 E3 196 MOV R3,A ;GET CHARACTER
0047 AA 197 MOV R2,A ;SAVE CHARACTER
0048 D37F 198 XRL A,07FH ;IS CHARACTER DECIMAL POINT
004A 0650 199 JZ DPOINT ;
004C FA 200 MOV R.A2 ;SAVE CHARACTER TO ACCUMULATOR
004D A0 201 MOV OR8,A ;CHARACTER TO DISPLAY MAP
004E 0453 202 JMP RETURN ;
0050 FA 203 DPOINT: MOV A,R2 ;SAVED CHARACTER TO ACCUMULATOR
0051 50 204 ANL A,8R0 ;"AND" WITH OLD CHARACTER
0052 A0 205 MOV OR8,A ;BACK TO DISPLAY MAP
0053 FF 206 RETURN: MOV A,R7 ;RESTORE ACCUMULATOR
0054 93 207 RETR ;
208 ;*******************************************************************************************
209 REJECT

12/78
LOC OBJ SEQ SOURCE STATEMENT

218 ;******************************************************************************
219 ; LOOK-UP TABLE
220 ; THIS LOOK-UP TABLE ORIGIANATES IN PAGE 3 OF THE UPI-41 PROGRAM
221 ; AND DECIMAL POINT FOR A SELECTED CHARACTER FROM THE INPUT ROUTINE.
222 ; INVERSE LOGIC IS USED BECAUSE OF THE SPECIFIC DRIVER CIRCUITRY; THIS
223 ; IS ON A GIVEN SEGMENT MEANS IT IS OFF AND A 0 MEANS IT IS ON.
224 ;
225 ; Segment 0
226 ; Segment 1
227 ; Segment 2
228 ; Segment 3
229 ; Segment 4
230 ; Segment 5
231 ; Segment 6
232 ; Segment 7
233 ; Segment 8
234 ; Segment 9
235 ; Segment A
236 ; Segment B
237 ; Segment C
238 ; Segment D
239 ; Segment E
240 ; Segment F
241 ; Segment G
242 ; Segment H
243 ; Segment I
244 ; Segment J
245 ; Segment K
246 ; Segment L
247 ; Segment M
248 ; Segment N
249 ; Segment O
250 ; Segment P
251 ; Segment Q
252 ;******************************************************************************

USER SYMBOLS

BLANK 031F
BLKMAP 088E
CH0 0300
CH1 0301
CH2 0302
CH3 0303
CH4 0304
CH5 0305
CH6 0306
CH7 0307
CH8 0308
CH9 0309
CHA 030A
CHAFOS 031E
CHB 030B
CHC 030C
CHD 030D
CHE 030E
CHF 030F
CHG 0310
CHH 0311
CHI 0312
CHJ 0313
CHK 0314
CHL 0315
CHM 0316
CHN 0317
CHO 0318
CHR 0319
CHU 031A
CHV 031B
CHW 031C
CHX 031D
CHY 031E
CHZ 031F
DPSPLA 000F
DPNCNT 0050
INPUT 0038
LOOP 001D
RETURN 0053
SETTIME 0032
START 0009

ASSEMBLY COMPLETE, NO ERRORS
UPI-41A SENSOR MATRIX CONTROLLER

This program uses the UPI-41A as a Sensor Matrix Controller. It has monitoring capabilities of up to 128 sensors. The coordinate and sensor status of each detected change is available to the master microprocessor in a single byte. A 4x8 FIFO queue is provided for data buffering. Both hardware or polled interrupt methods can be used to notify the master of a detected sensor change. Applications; alarm systems, control panels, various keyboards.

All information pertaining to required hardware, software, input parameters and output results are fully documented in the Intel application note AP41 ("INTRODUCTION TO THE UPI-41A") and the listing for this program.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>A_STS, RB0 R0-R7</td>
<td>Robin J. Jigour</td>
</tr>
<tr>
<td>RAM Required:</td>
<td>Company:</td>
</tr>
<tr>
<td>(within UPI-41A)</td>
<td>Intel</td>
</tr>
<tr>
<td>3FH(see AP41)</td>
<td>Address:</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>*(Internal Use Only)</td>
</tr>
<tr>
<td>(within UPI-41A)</td>
<td>City:</td>
</tr>
<tr>
<td>9EH(see AP41)</td>
<td>*(Internal Use Only)</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td>State:</td>
</tr>
<tr>
<td>None</td>
<td></td>
</tr>
<tr>
<td>Assembler/Compiler Used:</td>
<td></td>
</tr>
<tr>
<td>ASM48 MOD42</td>
<td></td>
</tr>
</tbody>
</table>
**SOURCE STATEMENT**

1;  
2;  *** NOT USED  
3;  ********************  
4;  
5;  THIS PROGRAM USES THE UPI-41A AS A SENSOR MATRIX CONTROLLER.  
6; IT HAS MONITORING CAPABILITIES OF UP TO 128 SENSORS. THE COORDINATE  
7; AND SENSOR STATUS OF EACH DETECTED CHANGE IS AVAILABLE TO THE MASTER  
8; MICROPROCESSOR IN A SINGLE BYTE. A 40X8 FIFO QUEUE IS PROVIDED FOR  
9; DATA BUFFERING. BOTH HARDWARE OR POLLED INTERRUPT METHODS CAN BE USED  
10; TO NOTIFY THE MASTER OF A DETECTED SENSOR CHANGE.  
11;  
12; ********************  
13;  
14; REGISTER DEFINITIONS:  
15;  
16;  ---------  --------  --------  
17;  R0  MATRIX MAP POINTER  NOT USED  
18;  R1  FIFO POINTER  NOT USED  
19;  R2  SCAN ROW SELECT  NOT USED  
20;  R3  COLUMN COUNTER  NOT USED  
21;  R4  FIFO-IN  NOT USED  
22;  R5  FIFO-OUT  NOT USED  
23;  R6  CHANGE WORD  NOT USED  
24;  R7  COMPARE  NOT USED  
25;  
26; ********************  
27;  
28; PORT PIN DEFINITIONS:  
29;  
30; PIN  PORT 1 FUNCTION  PIN  PORT 2 FUNCTION  
31;  ---------  --------  --------  
32; P0-7  COLUMN LINE INPUTS  P0-3  ROW SELECT OUTPUTS  
33;  P4  FIFO NOT EMPTY INTERRUPT  
34;  P5  OBF INTERRUPT  
35;  P6-7  NOT USED  
36;  
37; ********************  
38;  
39; $EJECT
LOC OBJ SEQ SOURCE

48 ;-------------------------------------------------------------------------------------
49 ; CHANGE WORD BIT DEFINITION:
50 ;
51 ; BIT   FUNCTION
52 ;
53 ; ----   ------
54 ;
55 ; D0     OBF
56 ; D1-3   IBF, FB, F1 (NOT USED)
57 ; D4     FIFO NOT EMPTY
58 ; D5-7   USED DEFINED (NOT USED)
59 ;
60 ;-------------------------------------------------------------------------------------
61 ; EQUATES
62 ;
63 ; THE FOLLOWING CODE DESIGNATES THREE VARIABLES: SCANTM, FIFOB
64 ; AND FIFOTA. SCANTM ADJUSTS THE LENGTH OF A DELAY BETWEEN
65 ; SCANNING SWITCH. THIS SIMULATES DEBOUNCE FUNCTIONS. FIFOB
66 ; IS THE BOTTOM ADDRESS OF THE FIFO. FIFOTA IS THE TOP ADDRESS
67 ; OF THE FIFO. THIS MAKES IT POSSIBLE TO HAVE A FIFO 3 TO 40
68 ; BYTES IN LENGTH.
69 ;
70 ;-------------------------------------------------------------------------------------
71 ;
72 ; 000F 73 SCANTM EQU 0FH ; SCAN TIME ADJUST
73 ; 0008 74 FIFOB EQU 08H ; FIFO BOTTOM ADDRESS
74 ; 002F 75 FIFOTA EQU 2FH ; FIFO TOP ADDRESS
75 ;
76 ;
77 ; EJECT
LOC OBJ SEO SOURCE STATEMENT

78 ;******************************************************************************
79 ;
80 ;                      INITIALIZATION
81 ;
82 ;THE PROGRAM STARTS AT THE FOLLOWING CODE UPON RESET. WITHIN
83 ;THIS INITIALIZATION SECTION THE REGISTERS THAT MAINTAIN THE MATRIX
84 ;MAP, FIFO AND ROW SCANNING ARE SET UP. PORT 1 IS SET HIGH FOR USE
85 ;AS AN INPUT PORT FOR THE COLUMN STATUS. BIT 4 OF STATUS REGISTER IS
86 ;WRITTEN TO CONvey A FIFO EMPTY CONDITION. THE INITIAL COLUMN STATUS
87 ;OF ALL THE ROWS IN THE SENSOR MATRIX IS THEN READ INTO THE MATRIX
88 ;MAP. ONCE THE MATRIX MAP IS FILLED THE OBF INTERRUPT (PORT 2-4) IS
89 ;ENABLED.
90 ;
91 ;******************************************************************************
92 ;

0000  99 ORG 0
0000 B03F  94 INITHX: MOV R0, #3FH     ;MATRIX MAP POINTER REGISTER TOP ADDRESS
0002 BAOF  95 MOV R2, #0FH      ;SCAN ROW SELECT REGISTER TOP ROW
0004 BC08  96 MOV R4, #FIFOBA     ;FIFO INPUT ADDRESS REGISTER BOTTOM OF FIFO
0006 BD2F  97 MOV R5, #FIOTRA     ;FIFO OUTPUT ADDRESS REGISTER TOP OF FIFO
0008 B9FF  98 ORL P1, #0FFH     ;INITIALIZE PORT 1 HIGH FOR INPUTS
000A 2300  99 MOV A, #00H       ;INITIALIZE STATUS REGISTER: FIFO EMPTY
000C 90  100 MOV STS, A       ;WRITE TO STATUS REGISTER, BITS 4-7
000D FA  101 FILLMX: MOV A, R2     ;SCAN ROW SELECT TO ACCUMULATOR
000E 3A  102 OUTL P2, A         ;OUTPUT SCAN ROW SELECT TO PORT 2
000F 99  103 IN A, P1          ;INPUT COLUMN STATUS PORT 1
0010 AB  104 MOV @R0.A       ;LOAD MATRIX MAP WITH COLUMN STATUS
0011 FA  105 MOV A, R2       ;CHECK SCAN ROW SELECT REGISTER VALUE FOR 0
0012 C618 106 JZ OBFINT       ;IF 0 ENABLE OBF INTERRUPT
0014 C8  107 DEC R0        ;DECREMENT TO NEXT MATRIX MAP ADDRESS
0015 CA  108 DEC R2            ;DECREMENT TO SCAN NEXT ROW
0016 84DD 109 JMP FILLMX      ;FILL NEXT MATRIX MAP ADDRESS
0018 BA10 110 OBFINT: MOV R2, #10H    ;BIT 4 HIGH IN ROW SCAN SELECT REGISTER
001A FA  111 MOV A, R2     ;ROW SCAN SELECT VALUE TO ACCUMULATOR
001B 3A  112 OUTL P2, A     ;INITIALIZE PORT 2. BIT 4 FOR "EN FLAGS"
001C F5  113 EN FLAGS       ;ENABLE OBF INTERRUPT PORT 2. BIT 4
114 ;
115 #EJECT

12/78

4-692
LOC OBJ SEQ SOURCE STATEMENT

;*****************************************************************************
;Scan and Compare

;The following code is the scan and compare section of the program.
;Upon entering this section a check is made to see if the entire matrix
;has been scanned.  If so the registers that maintain the matrix map and row
;scanning are reset to the beginning of the sensor matrix.  If the entire
;matrix hasn't been scanned the registers increment to scan the next row.
;From this point on the row scan select register is used for two functions.
;Bits 0-3 for scanning and bits 4 and 5 for the external interrupts.  Thusly
;all usage of the registers is done by logically masking it so as to only
;affect the function desired. Once the registers are reset, one row of the
;sensor matrix is scanned.  A delay is executed to adjust for scan time
;(debounce).  A byte of column status is then read into the matrix map.
;At the time the new column status is compared to the old.  The result is
;stored in the compare register.  The program is then routed according to
;whether or not a change was detected.

;*****************************************************************************

081D FA 137 ADJREG: MOV A,R2 ;Scan row select to accumulator
081E 530F 138 ANL A,0FH ;Check for 0 scan value only, not interrupt
0820 C626 139 JZ RSETRG ;If 0 reset registers
0822 C8 140 DEC R0 ;Decrement matrix map pointer
0823 CA 141 DEC R2 ;Decrement scan row select
0824 042C 142 JMP SCANMX ;Scan matrix
0826 B83F 143 RSETRG: MOV R8,#3FH ;Reset matrix map pointer register, top address
0828 FA 144 MOV A,R2 ;Scan row select to accumulator
0829 430F 145 ORL A,0FH ;Reset scan row select, no interrupt change
082B A0 146 MOV R2,A ;Scan row select register
082C FA 147 SCANMX: MOV A,R2 ;Scan row select to accumulator
082D 3A 148 OUTL P2,A ;Output scan row select 10 port 2
082E B80F 149 MOV R3,#SCANTM ;Set delay for output scan time
0830 EB30 150 DELAY2: DJNZ R3,DELAY2 ;Delay
0832 09 151 IN A,P1 ;Input column status from port 1 to accumulator
0833 20 152 XCH A,R0 ;Store new column status save old in accumulator
0834 D8 153 XRL A,R0 ;Compare old with new column status
0835 AF 154 MOV R7,A ;Save compare result in compare register
0836 C669 155 JZ CHIFFUL ;If the same, check if fifo is full

;EJECT

12/78 4-693
LOC OBJ SEQ SOURCE STATEMENT

158 ;**************************************************************
159 ;
160 ; CHANGE WORD ENCODING
161 ;
162 ; THE FOLLOWING CODE IS THE CHANGE WORD ENCODING SECTION. THIS
163 ; SECTION IS ONLY EXECUTED IF A CHANGE WAS DETECTED. THE COLUMN COUNTER
164 ; IS SET AND DECREMENTED TO DESIGNATE EACH OF THE 8 COLUMNS. THE COMPARE
165 ; REGISTER IS LOOKED AT ONE BIT AT A TIME TO FIND THE EXACT LOCATION OF
166 ; THE CHANGE(S). WHEN A CHANGE IS FOUND IT IS ENCODED BY GIVING IT A
167 ; COORDINATE FOR ITS LOCATION. THIS IS DONE BY COMBINING THE PRESENT VALUE
168 ; IN THE ROW SCAN SELECT REGISTER AND THE COLUMN COUNTER. THE ACTUAL STATUS
169 ; OF THAT SENSOR IS ESTABLISHED BY LOOKING AT THE CORRESPONDING BYTE IN
170 ; THE MATRIX MAP. THIS STATUS IS COMBINED WITH THE COORDINATE TO ESTABLISH
171 ; THE CHANGE WORD. THE CHANGE WORD IS THEN STORED IN THE CHANGE WORD REGISTER.
172 ;
173 ;**************************************************************
174 ;
0038 B808 175 MOV R3, #08H ; SET COLUMN COUNTER REGISTER TO 8
003A CB 176 RR R3, R3 ; DECREMENT COLUMN COUNTER
003B F0 177 MOV A, R0 ; COLUMN STATUS TO ACCUMULATOR
003C 77 178 RR A ; ROTATE COLUMN STATUS RIGHT
003D A0 179 MOV @R0, A ; ROTATED COLUMN STATUS BACK TO MATRIX MAP
003E FF 180 MOV A, R7 ; COMPARE REGISTER VALUE TO ACCUMULATOR
003F 77 181 RR A ; ROTATE COMPARE VALUE RIGHT
0040 AF 182 MOV R7, A ; ROTATED COMPARE VALUE TO COMPARE REGISTER
0041 F245 183 JB7 ENCODE ; TEST BIT 7 IF CHANGE DETECTED ENCODE CHANGE WORD
0043 0469 184 JMP CCHFULL ; IF NO CHANGE IS DETECTED CHECK FOR FIFO FULL
0045 FA 185 ENCODE ; MOV A, R2 ; SCAN ROW SELECT TO ACCUMULATOR 00000000
0046 530F 186 ANL A, #0FH ; ROTATE ONLY SCAN VALUE
0048 E7 187 RL A ; ROTATE LEFT 00000000
0049 E7 188 RL A ; ROTATE LEFT 00000000
004A E7 189 RL A ; ROTATE LEFT 00000000
004B 4B 190 ORL A, R3 ; ESTABLISH MATRIX COORDINATE 0000000X
191 ; (OR) COLUMN COUNTER VALUE WITH ACCUMULATOR
004C AE 192 MOV R6, A ; SAVE COORDINATE IN CHANGE WORD REGISTER
004D F0 193 MOV A, @R0 ; COLUMN STATUS FROM MATRIX MAP TO ACCUMULATOR
004E 5380 194 ANL A, #08H ; 0 ALL BITS BUT BIT 7
0050 4E 195 ORL A, R6 ; (OR) SENSOR STATUS WITH COORDINATE FOR COMPLETED CHANGE WORD
0051 AE 196 MOV R6, A ; SAVE CHANGE WORD 0000000X
197 ;
198 #EJECT
FIFO-DBOUT MANAGEMENT

THE FOLLOWING CODE IS THE FIFO-DBOUT MANAGEMENT SECTION OF THE PROGRAM. THIS SECTION TAKES AN ENCODED CHANGE WORD AND LOADS IT INTO FIFO. THE FIFO NOT EMPTY INTERRUPT IS THEN SET AND THE FIFO-IN-POINTER GETS UPDATED. IF A FIFO FULL CONDITION IS THEN CHECKED FOR AND Routed Accordingly. IF BOTH THE FIFO AND OBF HAVE CHANGED WORDS THE PROGRAM LOCKS UP UNTIL THIS HAS CHANGED. IF THE FIFO ISN'T FULL COLUMN COUNTER = 0. FIFO EMPTY AND OBF CONDITIONS ARE CHECKED. THE FIFO-OUT-POINTER IS SET AND OBFOUT IS LOADED IF THE FIFO ISN'T EMPTY AND OBF ISN'T SET. IF THIS ISN'T THE SITUATION, PROGRAM FLOW IS ROUTED BACK TO THE SCAN AND COMPARE SECTION TO SCAN THE NEXT ROW.

0052 FC 216 LDBOFF: MOV A,R4 ;FIFO INPUT ADDRESS TO ACCUMULATOR
0053 FF 217 MOV R1,A ;FIFO POINTER USED FOR INPUT
0054 FF 218 MOV R6,A ;CHANGE WORD TO ACCUMULATOR
0055 FF 219 MOV @R1,A ;LOAD FIFO AT FIFO INPUT ADDRESS
0056 2218 220 STATN: MOV A,#10H ;BIT 4 FOR FIFO NOT EMPTY
0057 221 221 MOV R5,A ;WRITE TO STATUS REGISTER, FIFO NOT EMPTY
0058 222 222 INTAHl: ORL R2,#10H ;FIFO NOT EMPTY INTERRUPT PORT 2-5 HIGH
0059 223 223 MOV R2,A ;ROW SCAN SELECT TO ACCUMULATOR
005A 224 224 ORL A,#10H ;SAVE INTERRUPT, NO CHANGE TO SCAN VALUE
005B 225 225 MOV R2,A ;ROW SCAN SELECT REGISTER
005C 226 226 ADJFIN: MOV A,#IFOTA ;FIFO TOP ADDRESS TO ACCUMULATOR
005D 227 227 XRL A,R4 ;COMPARE WITH CURRENT FIFO INPUT ADDRESS
005E 228 228 JZ RSFFIN ;IF THE SAME RESET FIFO INPUT REGISTER
005F 229 229 INC R4 ;NEXT FIFO INPUT ADDRESS
0060 FF 230 JMP CHIFFUL ;CHECK FIFO FULL
0061 231 231 RSFFIN: MOV R4,#IFOBA ;RESET FIFO INPUT REGISTER, BOTTOM OF FIFO
0062 FF 232 CHIFFUL: MOV A,R4 ;FIFO INPUT ADDRESS TO ACCUMULATOR
0063 FF 233 MOV R5,A ;COMPARE INPUT WITH OUTPUT FIFO ADDRESS
0064 967D 234 JNZ CHCOUNT ;IF NOT SAME CHECK COLUMN COUNTER VALUE
0065 866D 235 CHOBFL: JDBF CHOBF1 ;IF OBF IS 1 THEN CHECK OBF
0066 236 236 ADJFOT: MOV A,#IFOTA ;FIFO TOP ADDRESS TO ACCUMULATOR
0067 FF 237 XRL A,R5 ;COMPARE TOP TO OUTPUT FIFO ADDRESS
0068 FF 238 JZ RSFFOT ;IF THE SAME RESET FIFO OUTPUT REGISTER
0069 1D 239 INC R5 ;NEXT FIFO OUTPUT ADDRESS
006A FF 240 JMP LORDDB ;LOAD DBOUT
006B 241 241 RSFFOT: MOV R5,#IFOBA ;RESET FIFO OUTPUT ADDRESS TO BOTTOM OF FIFO
006C FF 242 LORDDB: MOV A,R5 ;OUTPUT FIFO ADDRESS TO ACCUMULATOR
006D FF 243 MOV R1,A ;FIFO POINTER USED FOR OUTPUT
006E FF 244 MOV A,#10H ;CHANGE WORD TO ACCUMULATOR
006F FF 245 OUT DBH,A ;CHANGE WORD TO DBOUT
0070 FB 246 CHCOUNT: MOV A,R3 ;COLUMN COUNTER TO ACCUMULATOR
0071 FF 247 JNZ RRLOOK ;IF NOT 0 FINISH CHANGE WORD ENCODING
0072 963A 248 CHFFEN: MOV A,#IFOBA ;FIFO BOTTOM ADDRESS TO ACCUMULATOR
0073 FF 249 XRL A,R4 ;COMPARE FIFO INPUT ADDRESS WITH FIFO BOTTOM ADDRESS
0074 FF 250 JZ AdjFEM ;IF THE SAME, ADJUST TO CHECK FOR FIFO EMPTY
0075 FF 251 MOV A,R4 ;FIFO INPUT ADDRESS TO ACCUMULATOR
0076 87 252 DEC R4 ;DECREMENT FIFO INPUT ADDRESS IN ACCUMULATOR
0077 FF 253 XRL A,R5 ;COMPARE INPUT TO OUTPUT FIFO ADDRESSES

12/78
LOC   OBJ     SEQ      SOURCE STATEMENT

0088 0691   254      JZ      STATMT; IF SAME, WRITE STATUS REGISTER FOR FIFO EMPTY
0088 049C   255      JMP     CHOBF2; CHECK OBF
0088 232F   256      MOV     A, #FIFOTA; FIFO TOP ADDRESS TO ACCUMULATOR
0088 DD     257      XRL     A, R5; COMPARE TOP TO OUTPUT FIFO ADDRESS
0088 969C   258      JNZ     CHOBF2; IF NOT SAME THEN FIFO IS NOT EMPTY, CHECK OBF
0089 2300   259      MOV     A, #00H; CLEAR BIT 0 FOR FIFO EMPTY
0089 90     260      MOV     STS, A; WRITE TO STATUS REGISTER
0089 9A0F   261      INTRL0: ANL     R2, #0FCH; FIFO EMPTY, INTERRUPT PORT 2-5 LOW
0089 FA     262      MOV     A, R2; SCAN ROW SELECT TO ACCUMULATOR
0089 530F   263      ANL     A, #0DFH; SAVE INTERRUPT, NO CHANGE TO SCAN VALUE
0089 AA     264      MOV     R2, A; SCAN ROW SELECT REGISTER
0089 841D   265      JMP     ADJREG; ADJUST REGISTERS
0089 861D   266      CHOBF2: JZOF     ADJREG; IF OBF=1 THEN ADJUST REGISTERS
0089 846F   267      JMP     ADJFOT; ADJUST FIFO OUT ADDRESS TO LOAD DBBOUT
268 ;
269      END

USER SYMBOLS

ADJFEM 008C   ADJFIN 005F   ADJFOT 006F   ADJREG 001D   CHCTR 007D   CHFFEM 0000   CHFULL 0069   CHOBF1 0060
CHOBF2 009C   DELHY2 0030   ENCODE 0045   FIFOTA 0086   FIFOTA 002F   FILLX 0000   INITMX 0000   INITRL 0059
INTRL0 0094   LOAD6 0079   LOADFF 0052   OBINT 0018   RRL00K 003A   RSETRG 0026   RSFIN 0067   RSSFOT 0077
SCINRX 002C   SCINTM 000F   STATMT 0091   STATME 0056

ASSEMBLY COMPLETE, NO ERRORS
AP-27 LRC Printer Controller

This program uses the UPI-41 or UPI-41A as a dot matrix printer controller for the 40-column LRC 7040 dot matrix printer. The UPI buffers up to 40 ASCII characters, formats and prints the buffer whenever 40 characters or a CRLF is received whichever occurs first.

All information pertaining to required hardware, software, input parameters, and output results are documented in the Intel Application Note AP-27 ("Printer Control with the UPI-41").

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>See Listing</th>
<th>Programmer:</th>
<th>J. Beaston</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM Required:</td>
<td>8 bytes</td>
<td>Company:</td>
<td>Intel</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>766 bytes</td>
<td>Address:</td>
<td></td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td>--</td>
<td>City:</td>
<td></td>
</tr>
<tr>
<td>Assembler/Compiler Used:</td>
<td>ASM48 MOD41</td>
<td>State:</td>
<td></td>
</tr>
</tbody>
</table>
Combination I/O Device (UNIOD)

This program uses the UPI-41A as a combination serial and parallel I/O device. The serial part is full-duplex asynchronous with programmable baud rates of 1200, 600, 300, or 110 baud. The transmitter and receiver are double buffered. The receiver checks for framing and overrun errors.

The 8-bit parallel part is programmable for input or output.

All information pertaining to required hardware, software input and output parameters is fully documented in the Intel Application Note AP-41 ("Introduction to the UPI-41A") and the listing for this program.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>See listing</th>
<th>Programmer: J. Beaston</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM Required:</td>
<td>12 bytes</td>
<td>Company: Intel</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>363 bytes</td>
<td>Address:</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td>1</td>
<td>City:</td>
</tr>
<tr>
<td>Assembler/Compiler Used:</td>
<td>ASM48 MOD42</td>
<td>State:</td>
</tr>
</tbody>
</table>
SDK-86 Serial Monitor, V1.1

Interactive monitor with commands for examining/modifying registers and memory, controlling program execution using breakpoints or single step, moving memory blocks, inputting from or outputting to I/O ports, and reading and writing HEX/Object files on paper tape.

SDK-86
ASR-33 teletype or CRT

See SDK-86 MCS-86 System Design Kit User's Guide, Order #98000698

Available on non-system diskette only for $35.00
(source & object code included)

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>N/A</th>
<th>Programmer: Janet Takami</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM Required:</td>
<td>256 bytes</td>
<td>Company: Intel</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>4K bytes</td>
<td>Address:</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td></td>
<td>City:</td>
</tr>
<tr>
<td>Assembler/Compiler Used:</td>
<td>PL/M-86 Compiler V1.0</td>
<td>State:</td>
</tr>
</tbody>
</table>
SDK-86 Keypad Monitor, V.1.1

Interactive monitor with commands for examining/modifying registers and memory, controlling program execution using breakpoints or single step, moving memory blocks, and inputting from or outputting to I/O ports.

SDK-86

See SDK-86 MCS-86 System Design Kit User's Guide, Order #98000698

See above User's Guide

Available on non-system diskette only for $35.00 (source & object code included)

Registers Modified: N/A  Programmer: Janet Takami
RAM Required: 256 bytes  Company: Intel
ROM Required: 4K bytes  Address:
Maximum Subroutine Nesting Level:  City:
Assembler/Compiler Used: PL/M-86  State: Compiler V.1.0
**Program Title**

TAPE - Audio Tape Interface

**Function**

This routine outputs RAM data to an audio cassette recorder which is paralleled to a CRT terminal. The data can be read back by means of the monitor `l-command` so that one needs not enter any bootstrap loader.

**Required Hardware**

Intel SDK-80 Kit or other 8080 computer; CRT terminal (or TTY); cassette interface (see diagram included with source listing)

**Required Software**

SDK-80 monitor PROM

**Input Parameters**

Data area start, end; start program: .G1300 (CR)
Enter start/end-addresses: XXXX, YYYY (CR)
After the CR turn on cassette recorder immediately.
Leader and trailer of approx. 5 sec are produced by the program itself.
After data recording (visible on the terminal) the cassette recorder should be stopped while the trailer runs to avoid the sign on message being recorded.

**Output Results**

Data on cassette in 'l-format'

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>N/A</th>
<th>Programmer: Guenter Ruschitzka</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM Required:</td>
<td>70 bytes</td>
<td>Company:</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>SDK-80 monitor PROM</td>
<td>Address: Im Bruehl 1</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td>N/A</td>
<td>City: D-6921 Zuzenhausen</td>
</tr>
<tr>
<td>Assembler/Compiler Used:</td>
<td>8080</td>
<td>State: Western Germany</td>
</tr>
</tbody>
</table>

© Intel Corporation, 1978

12/78
TEXT PROCESSOR

Processes the textual material into intended format by using the format command language. The command languages are interspersed within the source text. The user can specify margins, case headings and footings, paragraphs, center text, right justify, page footnote, underline, create tables and more.

Intel MDS, Disk drive, Video terminal, Printer

ISIS-II file system

TEXT <input file> [<output file>]
where
<input file> is the source textual material
<output file> is the output file name
:LP: is for line printer
:CO: is for CRT, and :TO: is for teletype
If not specified, the default file name is used (see the user's Manual)

The output is generated as specified.
The error message is displayed onto the CRT

Program available on diskette only for $70.00

<table>
<thead>
<tr>
<th>Registers Modified: None</th>
<th>Programmers: Triyono</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM Required:</td>
<td>Company: Naval Postgraduate School</td>
</tr>
<tr>
<td>ROM Required: 8K</td>
<td>Address: Code 52Bg</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level: 2</td>
<td>City: Monterey</td>
</tr>
<tr>
<td>Assembler/Compiler Used: PL/M-80</td>
<td>State: CA 93940</td>
</tr>
</tbody>
</table>
8278 Keyboard/Display Controller

This program is the source code for the UPI-41A based 8278 Keyboard/Display Controller. Features of the 8278 are:

- 128 key scanning logic.
- 16 digit LED display multiplexing.
- Interface for either contact or capacitively coupled keyboards.
- N-key rollover.
- 8-character Keyboard FIFO.
- Right or left entry display.

All information related to required hardware, software, input parameters, and output results is documented in the 8278 Data sheet and program listing.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>See Listing</td>
<td>J. Beaston</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RAM Required:</th>
<th>Company:</th>
</tr>
</thead>
<tbody>
<tr>
<td>64 bytes</td>
<td>Intel</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ROM Required:</th>
<th>Address:</th>
</tr>
</thead>
<tbody>
<tr>
<td>96 bytes</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Maximum Subroutine Nesting Level:</th>
<th>City:</th>
</tr>
</thead>
<tbody>
<tr>
<td>See Listing</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Assembler/Compiler Used:</th>
<th>State:</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASM48 MOD42</td>
<td></td>
</tr>
</tbody>
</table>
8295 - Dot Matrix Printer Controller

This program is the source for the UPP-41A based 8295 Dot Matrix Printer Controller. It is intended to interface LRC 7040 series printers. The 8295 features are:

--On-chip 40 character buffer.
--Parallel or serial communication to master processor.
--Programmable DMA interface.
--10/12 chr/in print density.
--Single or double width printing.
--Programmable print intensity.
--Programmable line feeds.
--3 programmable tabulations.
--2 general-purpose output pins.

All information related to required hardware, software, input parameters, and output results is documented in the 8295 Data Sheet and program listing.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>See Listing</th>
<th>Programmer:</th>
<th>J. Beaston</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM Required:</td>
<td>48 bytes</td>
<td>Company:</td>
<td>Intel</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>1021 bytes</td>
<td>Address:</td>
<td></td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td>See Listing</td>
<td>City:</td>
<td></td>
</tr>
<tr>
<td>Assembler/Compiler Used:</td>
<td>ASM48 M0042</td>
<td>State:</td>
<td></td>
</tr>
</tbody>
</table>
OLIVETTI 20-Column Printer Controller

This program is for a UPI-41 controlling an Olivetti 20-column printer. The UPI interface to the master processor may be parallel via the standard peripheral interface or serial (2400 baud). The UPI accepts and buffers up to 20 ASCII characters before printing. Complete control and timing of the printer motor and solenoids is provided by the UPI.

All information pertaining to required hardware, software, input parameters, and output results are included in the attached document and program listing.

<table>
<thead>
<tr>
<th>Registers Modified</th>
<th>See Listing</th>
<th>Programmer: J. Beaston</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM Required</td>
<td>See Listing</td>
<td>Company: Intel</td>
</tr>
<tr>
<td>ROM Required</td>
<td>885 Bytes</td>
<td>Address:</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level</td>
<td>See Listing</td>
<td>City:</td>
</tr>
<tr>
<td>Assembler/Compiler Used</td>
<td>ASM48 MOD41</td>
<td>State:</td>
</tr>
</tbody>
</table>
8292 (GPIB Controller) implementation on 8741A.

This software implements the IEEE 488 controller function (8292) on the 8741A.

Refer to 8292 Data Sheet for required hardware.

Refer to listing and 8292 Data Sheet for required software, input parameters, output results, and other software details.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Programmer: T. Voll</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM Required: N/A</td>
<td>Company: Intel</td>
</tr>
<tr>
<td>ROM Required: 1000 10</td>
<td>Address:</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td>City:</td>
</tr>
<tr>
<td>Assembler/Compiler Used: MCS-48/UP1-41</td>
<td>State:</td>
</tr>
</tbody>
</table>
SEND48 - Download to PROMPT 48 for Series II.

Allows user to download a HEX file from a Series II Development System (with disk) to the PROMPT 48.

(1) Series II Development System with Serial Channel 2 available and unmodified.
(2) Prompt 48 strapped for RS232C, 2400 baud.
(3) Interconnecting cables.
(See program listing for details.)
ISIS-II in Development System.
Monitor firmware in Prompt 48.

See program listing for operating instructions.

None, except for standard ISIS error messages, if appropriate.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>All</th>
<th>Programmer:</th>
<th>P. Bushell</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM Required:</td>
<td>1206H bytes including a 4K buffer whose size may be altered.</td>
<td>Company:</td>
<td>MicroGenics</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>-</td>
<td>Address:</td>
<td>22, Willows Road,</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td>(Stack= 2AH)</td>
<td>Bourne End, Bucks.</td>
<td></td>
</tr>
<tr>
<td>Assembler/Compiler Used:</td>
<td>PLM80 ver. 3.1</td>
<td>SL8 5HG. ENGLAND.</td>
<td></td>
</tr>
</tbody>
</table>
OUTIN - PROMPT 48 or PROMPT 80 INTERFACE

"OUTIN" sends (OUTBYT) and receives (INBYT) one byte to
PROMPT48 or PROMPT80 via PROMPT-SPP-Cable (see source-listing)

Intellec-MDS, PROMPT48 or PROMPT80, PROMPT-SPP-Cable (see attached
PROMPT-SPP-Schematic)

ISIS II (for testing: see attached PROTES)

OUTBYT  Register C = transmit-value
INBYT  none

OUTBYT  non (but transmit-value is transmitted to PROMPT)
INBYT  Register A = received value form PROMPT (or 0, if
no startbit was detected during time-out-loop)

----for more details see OUTIN-source-listing----

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>A, B, C, D, E</td>
<td>Glasmacher</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RAM Required:</th>
<th>Company:</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td>Peter Glasmacher</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ROM Required:</th>
<th>Address:</th>
</tr>
</thead>
<tbody>
<tr>
<td>98 byte</td>
<td>Boxberger Str. 11, D-8000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Maximum Subroutine Nesting Level:</th>
<th>City:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Munich 45,</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ISIS-II 8080/8085 Macro Ass. V2.0</th>
<th>State:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assembler/Compiler Used:</td>
<td>West Germany</td>
</tr>
</tbody>
</table>
REMOTE48 - INTERACTIVE CONTROLLER OF PROMPT48

Remote, interactive control of PROMPT48 through MDS and CRT
(for detailed description see additional sheet)

MDS with 32K-RAM, CRT or TTY, PROMPT48, PROMPT-SPP

for use: ISIS-II
for generation: PLMB0 V2.0 or V3.0 (for REMOTE.SRC)
        ASM80 V2.0 (for QUIIN.SRC)

type commands on consol

Depends on typed commands:
PROMPT48-action (e.g. single step, go with break, etc.)
Dump of PROMPT48-data (e.g. registers, program-memory, etc.)
to consol and - if required- to an ISIS-file

The PROMPT48-data on consol are cleaned and tabulated.
an address-header is inserted after every 8 lines, to
get good readability.
The PROMPT48-data, that is sent to an ISIS-file, is in
Intel-Hex-Format.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>All</th>
<th>Programmer:</th>
<th>Peter Glasmacher</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM Required:</td>
<td>32K-MDS-RAM</td>
<td>Company:</td>
<td>Ingenie. Glasmacher</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>Nothing</td>
<td>Address:</td>
<td>Boxberger Strabe II</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td>Less 10</td>
<td>City:</td>
<td>D-8000 Munchen 45</td>
</tr>
<tr>
<td>PLMB0 V3.0 and ASM80 V2.0</td>
<td>Assmber/Compiler Used:</td>
<td>State:</td>
<td>West-Germany</td>
</tr>
</tbody>
</table>

© Intel Corporation, 1978
APPENDIX to REMOTE48

REMOTE48 accepts the following commands, which correspond to one or more keys of the PROMPT48.

f.e. the single-letter-command "S" performs the following PROMPT48-keystrokes or PROMPT48-commands

(GO) (SINGLE STEP) (.)(D) (REGISTER) (0) (,) (4) (8) (.)

and prints or displays it to the consol in this form

ADDR +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F
0000 04 67 94 D4 FF 00 00 00 00 30 FE 98 55 A5 6D 3E
0010 00 FF ......and so on........

**REMOTE48-Commands = PROMPT48-Keystrokes or function**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXIT</td>
<td>exits or returns from REMOTE48 to ISIS</td>
</tr>
<tr>
<td>PROG</td>
<td>(PROGRAM MEMORY)</td>
</tr>
<tr>
<td>DAT</td>
<td>(DATA MEMORY)</td>
</tr>
<tr>
<td>REG</td>
<td>(REGISTER)</td>
</tr>
<tr>
<td>DUP</td>
<td>(D) (PROGRAM MEMORY)</td>
</tr>
<tr>
<td>DUD</td>
<td>(D) (DATA MEMORY)</td>
</tr>
<tr>
<td>DUR</td>
<td>(D) (REGISTER)</td>
</tr>
<tr>
<td>C/P</td>
<td>(CLEAR ENTRY/PREVIOUS)</td>
</tr>
<tr>
<td>EX</td>
<td>(EXAMINE)</td>
</tr>
<tr>
<td>GB</td>
<td>(GO) (BREAK)</td>
</tr>
<tr>
<td>GN</td>
<td>(GO) (NO BREAK)</td>
</tr>
<tr>
<td>GS</td>
<td>(GO) (SINGLE STEP)</td>
</tr>
<tr>
<td>TO&lt;isis-file&gt;</td>
<td>write the content of the input-buffer,</td>
</tr>
<tr>
<td></td>
<td>f.e. dumped program-memory, to the</td>
</tr>
<tr>
<td></td>
<td>specified ISIS-file</td>
</tr>
<tr>
<td>R</td>
<td>(D) (REGISTERS) (0) (,) (4) (8) (.)</td>
</tr>
<tr>
<td>S</td>
<td>(GO) (SINGLE STEP) (.) (D) (REGISTERS) (0)</td>
</tr>
<tr>
<td></td>
<td>(,) (4) (8) (.).</td>
</tr>
<tr>
<td>,</td>
<td>(,) or (NEXT)</td>
</tr>
<tr>
<td>.</td>
<td>(,) or (EXECUTE/END)</td>
</tr>
<tr>
<td>0</td>
<td>(0)</td>
</tr>
<tr>
<td>1</td>
<td>(1)</td>
</tr>
<tr>
<td>etc.</td>
<td>etc.</td>
</tr>
<tr>
<td>9</td>
<td>(9)</td>
</tr>
<tr>
<td>A</td>
<td>(A)</td>
</tr>
<tr>
<td>etc.</td>
<td>etc.</td>
</tr>
<tr>
<td>E</td>
<td>(E)</td>
</tr>
<tr>
<td>F</td>
<td>(F)</td>
</tr>
</tbody>
</table>

hexadecimal numbers

It is allowed to insert spaces between commands or hexadecimal numbers, but not within commands. It is possible to correct a line with ISIS-line-editing-commands like "rub out" etc. The PROMPT48-monitor-echo-signs like "?" or "-" are displayed on the consol. REMOTE48 asks for new command or commands with TYPE COMMAND:_

4-724
**Error-reporting:**

*****NO CONNECTION BETWEEN MDS AND PROMPT (PRESS (MON INT) OR (SYS RST))!

This message is printed to the consol, if MDS and PROMPT don't communicate, f.e. PROMPT48 is switched off, the PROMPT48-key-monitor is active or the PROMPT 48 is running after "Go no break" or "Go with break".

^*****HERE OCCURED ERROR OR CONSL-BREAK

The first sign of this message ("^") points to the first sign of the unrecognized command, that is reprinted first.

**Example:**

TYPE COMMAND: __________ NONSENSE

TYPE COMMAND: NONSENSE

^*****HERE OCCURED ERROR OR CONSL-BREAK

TYPE COMMAND:

The underlined signs were typed in by the user.

The same message appears, if the PROMPT48 was running when one hit any consol-key.

**Using REMOTE48 under ISIS** (a short example of a REMOTE48-session)

-REMOTE (that is the call to REMOTE48 from ISIS-level)

PROMPT48-REMOTE VERSION 1.0

TYPE COMMAND: GS 0. (that is one single step begining at address 0)

???- (that is the echo of PROMPT48)

TYPE COMMAND: EXIT (that means: return to ISIS)

- (that is the ISIS-prompt)
SYMBOL TABLE INSERIER FOR AB22

Inserts labels into a disassembled object tape.

MDS System

Program AB22

This program accepts:

1. a symbol table from the console operator
2. a disassembled object tape of the format produced by program AB22.

Source tape with labels inserted, ready for reassembly.

<table>
<thead>
<tr>
<th>Registers Modified</th>
<th>Programmer</th>
</tr>
</thead>
<tbody>
<tr>
<td>A11</td>
<td>B.A. Robinson</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RAM Required</th>
<th>Company</th>
</tr>
</thead>
<tbody>
<tr>
<td>2837</td>
<td>Du Pont of Canada Limited</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ROM Required</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P.O. Box 5000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Maximum Subroutine Nesting Level</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Kingston</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Assembler/Compiler Used</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>8080 Macro Assem. V2</td>
<td>Ontario, Canada, K7L 5A5</td>
</tr>
</tbody>
</table>
PSEUDO DISASSEMBLER FOR SDK-80

The program lists a machine program one instruction a line. An instruction consists of one, two or three bytes. Each line is preceded by the instruction address.

SDK-80Kit for other 8080 computer

SDK-80 Monitor PROM

Start the program with G1300 (CR), then type in the begin address followed by (CR). ESC terminates the listing, any other character triggers another line.

Pseudo-disassembled machine code on console

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Programmer: Guenter Ruschitzka</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM Required: 8AH Bytes</td>
<td>Company:</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>Address: Im Bruehl 1</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td>City: 6921 Zuzenhausen</td>
</tr>
<tr>
<td>Assembler/Compiler Used: 8080</td>
<td>State: Western Germany</td>
</tr>
</tbody>
</table>
SCAN - SCANNER AND SELECTIVE FILE LINE PRINTER

Program allows rapid scanning of a disk text file using the console CRT. Any segment may also be directed to the Line Printer. Historical printing is used to insure all of a selected segment is printed.

MDS, Floppy Disk Drive, Line Printer, CRT, 32K RAM

ISIS-II, MDS Monitor

Program implemented by entering:
SCAN FILENAME

The selected file is output to the CRT.
Controls may be entered at any time.

CTRL/S   - Freeze Output
CTRL/Q   - Unfreeze Output
SPACE    - Starts or stops printer. When started printer will print historically up to 22 lines. If more than this number have been output without printing, then a line of asterisks is printed to indicate the skipped data.
CTRL/C   - Abort and Return to ISIS

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>All</th>
<th>Programmer:</th>
<th>R. Ryan</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM Required:</td>
<td>32K</td>
<td>Company:</td>
<td>TOTCO</td>
</tr>
<tr>
<td>ROM Required:</td>
<td></td>
<td>Address:</td>
<td>P. O. Box 1307</td>
</tr>
<tr>
<td></td>
<td></td>
<td>W. Rock Creek Road</td>
<td></td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td></td>
<td>City:</td>
<td>Norman</td>
</tr>
<tr>
<td>Assembler/Compiler Used:</td>
<td>ISIS-II 8080/8085</td>
<td>State:</td>
<td>Oklahoma</td>
</tr>
<tr>
<td></td>
<td>ASSEM. V2.0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
SDK85 -- Monitor for the 8085 System Design Kit

Monitor for the Intel 8085 System Design Kit (SDK-85). Provides a minimum level of utility functions for the user as well as such functions as memory and register manipulation, program loading and execution, and single step capability.

SDK-85 (also supports TTY)

None


---

Available on non-system diskette only for $35.00 (source & object code included)

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM Required:</td>
<td>38 bytes + stack</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>2K bytes</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td>4</td>
</tr>
<tr>
<td>Assembler/Compiler Used:</td>
<td>8080/8085 Macro Assembler</td>
</tr>
</tbody>
</table>

Programmers: This monitor is provided in ROM form to customers purchasing the SDK-85 Kit.

Company:  
Address:  
City:  
State:  

© Intel Corporation, 1978  
2/79  
4-731
DKDUMP - ISIS DISK FILE DUMP

Dumps ISIS disk files in HEX and ASCII to an output file
(typically :CO: or :LP:)

MDS DISK SYSTEM

ISIS-II

FROM KEYBOARD:
DKDUMP: :Fn:filename to :00:
Where: (:Fn: is the drive number); (filename is the fully
qualified file name); and (:00: is the output
device).

i.e. DKDUMP DKDUMP TO :LP:

A hexadecimal interpretation of the file with a separate
column for the ASCII equivalent code. 16 Bytes are dis-
played on each line, with a hexadecimal relative - to byte
0 of the file address displayed in the left margin.

Registers Modified:        Programmer:  Stu Adler
RAM Required:               Company:     Litton Energy Control
ROM Required:               Address:     8944 Mason Ave
Maximum Subroutine Nesting Level:  City:     Chatsworth
Assembler/Compiler Used:        State:      CA 91311
DDUMP - DISKETE DUMP

Dumps diskette on a block basis to :CO: or :LP: in hexadecimal and ASCII format

MDS, MDS-DOS, console device

MDS Monitor, ISIS-II

DDUMP Program prompts for output device, beginning track and sector, ending track and sector

† S - suspends display
† Q - continues display
† R - restarts DDUMP
† E - reboots ISIS

Outputs to selected device contents of specified block (or blocks) in hexadecimal and ASCII format

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>A11</th>
<th>Programmer:</th>
<th>Carl Harcourt</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM Required:</td>
<td>32K</td>
<td>Company:</td>
<td>Naval Avionics Center</td>
</tr>
<tr>
<td>ROM Required:</td>
<td></td>
<td>Address:</td>
<td>6000 E. 21st Street</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td></td>
<td>City:</td>
<td>Indianapolis</td>
</tr>
<tr>
<td>Assembler/Compiler Used:</td>
<td>ISIS-II Macro-Assembler</td>
<td>State:</td>
<td>Indiana 46218</td>
</tr>
</tbody>
</table>
RATE - BAUD RATE SELECTION FOR MDS 220 AND 230 SYSTEMS

Initializes serial ports 1 and 2 for Intel 220 and 230 Development Systems

Required Hardware

Intel 220 or 230 Development System

Required Software

ISIS-II System Programs

Input Parameters

Main console

Output Results

Serial port's baud rate, stop bits, parity, word length are selected by operator for both ports

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>8080-all</th>
<th>Programmer:</th>
<th>Tom Wrenn</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM Required:</td>
<td>2K bytes</td>
<td>Company:</td>
<td>Dayton Scientific Inc</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>2K bytes</td>
<td>Address:</td>
<td>121 West Park</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td></td>
<td>City:</td>
<td>Dayton</td>
</tr>
<tr>
<td>Assembler/Compiler Used:</td>
<td>ASM80, V2.0</td>
<td>State:</td>
<td>Ohio 45459</td>
</tr>
</tbody>
</table>
SEARCH - KEYWORD FILE SEARCH FOR ISIS-II ENVIRONMENT

Searches specified source file for lines containing specified keywords. Source lines containing all the keywords (maximum of 10) are printed to the specified list file.

MDS800 (with at least 32K RAM)
system console
floppy disk drive

ISIS-II System Files - program contains linkages to ISIS-II system library routines for file input/output

Command syntax is (assuming file :F:SEARCH is the compiled and located object module):

```
SEARCH FILENAME <PRINT LISTFILE> KEY KEYWORD1 KEYWORD2 KEYWORD3... WHERE:
-FILENAME IS THE FILE TO BE SEARCHED
-LISTFILE IS FILE TO WHICH LINES CONTAINING SPECIFIED KEYWORDS ARE TO BE PRINTED
(DEFAULTS TO :CO:)
-KEYWORD(N) ARE KEYWORDS FOR WHICH THE PROGRAM SEARCHES
-A MAXIMUM OF TEN KEYWORDS (DELIMITED BY BLANKS)
-TABS IN FILENAME ARE CONVERTED TO SPACES IN LISTFILE
```

Error Messages (in addition to ISIS-II error messages):

Error #

101 Reserved word "KEY" not found in command buffer
102 No keywords found in command buffer
103 (CRLF) (LINE SEPARATOR) was not found in source file
104 (CR) not found in command buffer

<table>
<thead>
<tr>
<th>Registers Modified</th>
<th>ALL</th>
<th>Programmer: Richard D. Heslip</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM Required:</td>
<td>Program length about 2K</td>
<td>Company: Gandalf Data Communications</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>None</td>
<td>Address: 9 Slack Road</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td>10</td>
<td>City: Ottawa</td>
</tr>
<tr>
<td>Assembler/Compiler Used:</td>
<td>PLM80</td>
<td>State: Ontario CANADA</td>
</tr>
</tbody>
</table>

© Intel Corporation, 1978
TIMER - MEASURES EXECUTION TIMES OF USER PROGRAMS

Determines the time taken by any user program to execute

RANGE: 1 Milli second to 4½ hours

MDS 800 with Real Time Clock on front panel control module

ISIS-II operating system V2.2
MDS Monitor V2.0

Starting address of the user program to be timed. The program must exit with a 'RETURN' statement as timer treats it as a subroutine

Time in milliseconds displayed on the system console

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>ALL</th>
<th>Programmer: M. Master</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM Required:</td>
<td>16 bytes</td>
<td>Company: Univ of Ottawa</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Electrical Engrg. Dept</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>181 bytes</td>
<td>Address: 770 King Edward Ave</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td>User program +2</td>
<td>City: Ottawa, Ontario</td>
</tr>
<tr>
<td>Assembler/Compiler Used:</td>
<td>ASM80, V2.0</td>
<td>State: CANADA K1N 6N5</td>
</tr>
</tbody>
</table>

© Intel Corporation, 1978
**KAPIAR, V1.2 - A GENERAL PURPOSE MACROPROCESSOR**

Processes user-defined MACROS. Built-in MACROS are:
- DEFINED, DELETE, SAVE, IFELSE, SUBSTR, APPEND, LENGTH, TYPE
- INDEX, INCR, DECR, REG, LOAD, ADD, SUB, TOKEN, LINE, and BUFFER.
Includes KAPIAR.SKL, a stripped-down version for skeleton of
dedicated Macroprocessors—extensive debugging facilities.

40 Kb or RAM, Diskette files, keyboard, video monitor

**ISIS-II**

**DOES NOT APPLY**

**DOES NOT APPLY**

**AVAILABLE ON DISKETTE ONLY**

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>ALL</th>
<th>Programmer:</th>
<th>Steve Newberry</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM Required:</td>
<td>40Kb</td>
<td>Company:</td>
<td>Stanford University</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>None</td>
<td>Address:</td>
<td></td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td></td>
<td>City:</td>
<td>Stanford</td>
</tr>
<tr>
<td>Assembler/Compiler Used:</td>
<td>PL/M80, V3.0</td>
<td>State:</td>
<td>California 94305</td>
</tr>
</tbody>
</table>
ECCO PAPER TAPE READER


ECCO Paper Tape Reader Model 2001-2 or equivalent
SBC 80/10 with SBC 116 or SBC 80/20

SBC 80P Monitor or
SBC 80P20 Monitor

Memory area as specified on Intel Hexadecimal Object File Formatted Paper Tape.

<table>
<thead>
<tr>
<th>Registers Modified</th>
<th>Programmer</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL</td>
<td>H.R. Pinnich, Jr.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RAM Required</th>
<th>Company</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Southeast Missouri State Univ.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ROM Required</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>OEEH</td>
<td>Department of Chemistry</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Maximum Subroutine Nesting Level</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>MACB0 Cross</td>
<td>Cape Girardeau</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Assembler/Compiler Used</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assembler V2.4</td>
<td>MO 63701</td>
</tr>
</tbody>
</table>
TRACE.ICE

Symbolic disassembly and register dump in ICE80.

Intellec MDS, ICE80

ISIS-II, ICE80

Trace is called as an MDSCALL after GO or STEP emulation. The user types the number of emulations and TRACE halts and waits for command when they have been completed. Exit to ice command is provided by esc.

Trace displays the current timer, all flags as symbols (e.g. +ZC) all registers in hex, P.C. in hex and symbolic, mnemonic, operand in hex and symbolic. All display is on one line. Symbols are taken from the ice symbol tables and H/M line no. tables. In addition in GO mode Trace displays the 44 cycle history, by symbolic disassembly.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>N/A</th>
<th>Programmer:</th>
<th>C.J. Lusby Taylor</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM Required:</td>
<td>4C5H</td>
<td>Company:</td>
<td>Intel International</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>None</td>
<td>Address:</td>
<td>Rue du Moulin a Papier 51</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td>N/A</td>
<td>City:</td>
<td>B-1160 Brussels</td>
</tr>
<tr>
<td>Assembler/Compiler Used:</td>
<td>ISIS ASM80 V2.0</td>
<td>State:</td>
<td>Belgium</td>
</tr>
</tbody>
</table>
Program Title: DOWN 80

Function:
To download an Intel hexadecimal formatted program from a Series II to PROMPT-80/85

Required Hardware:
PROMPT-SER (RS232C Cable)
User build (MDS Series II, PROMPT 80) male-to-male connector

Required Software:
ISIS-II V3.4

Input Parameters:
Program must be linked and located to ISIS-II system files.

Output Results:
Program loaded into PROMPT RAM

Registers Modified: 

Programmer: Conrad Weiderhold

RAM Required:
Company: Intel Corporation

ROM Required:
Address: 900 Jorie Blvd.
Suite 220

Maximum Subroutine Nesting Level:
City: Oakbrook

Assembler/Compiler Used: PL/M80, V3.1
State: Illinois 60521
### IBM BI-SYNC CRC16 GENERATION SUBROUTINE

Generates IBM CRC 16 Check Bytes using polynomial $x^{16} + x^{15} + x^2 + 1$

**Required Hardware**

8048 CPU

**Required Software**

User Drive Program

**Input Parameters**

Transmission data byte in accumulator

**Output Results**

Reg Bank 1 R5 & R6 Contain CRC

N.B. STX Character (02H) Resets CRC

PAD Character (FFH) Ignored (IBM Convention)

---

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>R81 - R4, - 7</td>
<td>Andy Belton</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RAM Required:</th>
<th>Company:</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Defined</td>
<td>Tech-Nel Data Products LTD</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ROM Required:</th>
<th>Address:</th>
</tr>
</thead>
<tbody>
<tr>
<td>37H Bytes</td>
<td>PO Box 7</td>
</tr>
<tr>
<td></td>
<td>75 High Street</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Maximum Subroutine Nesting Level:</th>
<th>City:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Just entry call</td>
<td>Brackley Northants</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Assembler/Compiler Used:</th>
<th>State:</th>
</tr>
</thead>
<tbody>
<tr>
<td>8048/41 Macro Assembler</td>
<td>NN13 5NN England</td>
</tr>
</tbody>
</table>
Program Title: HXEDIT - HEXADECIMAL DISK FILE EDITOR

Function: To modify hexadecimal disk files.

Required Hardware: MDS

Required Software: ISIS-II

Input Parameters: HXEDIT file
Program must be linked and located to ISIS-II system files.

Output Results: Arbitrary changes may be made to any located object disk file (not write-protected). Patches in machine language may be made to located objects, thereby avoiding the necessity for re-assembling and re-locating.

Registers Modified:

<table>
<thead>
<tr>
<th>RAM Required:</th>
<th>1070 Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROM Required:</td>
<td></td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td>4</td>
</tr>
<tr>
<td>Assembler/Compiler Used:</td>
<td>ASM80</td>
</tr>
</tbody>
</table>

Programmer: Ben A. Harris

Company: Techtran Industries

Address: 200 Commerce Drive

City: Rochester

State: New York 14623
ALPHANUMERIC INPUT FROM NUMERIC KEYBOARD

Enter a set of 44 alphanumeric characters from a 16-key keyboard into a 8080 system.

Required Hardware
8080 System, 16 key keyboard, 16-character display, console

Required Software
None

Input Parameters
EXAMPLE: keyboard input sequence
   " 3, *5, *7, *2, 4, 9, 8, Ø, 8, Ø, 1"
   will produce text "INTEL 8080A" on the display

Output Results
Keyboard "LF" will output this text to the console.

Registers Modified: All
RAM Required: 400 bytes
ROM Required: 20 bytes
Maximum Subroutine Nesting Level: ISIS-II
Assembler/Compiler Used: 8080/8085

Programmer: B. Hauert, R. Farkas
Company: Battelle-Geneva
Address: 7, route de Drize
City: CH-1227 Carouge/Geneve
State: Switzerland
8089 -- BREAK.89

8089 Break Point routine for saving all registers and displaying to CRT.

8086, 8089 MDS System, PL/M 86
(must include 8251 resident on 86BUS, channel attention decode with select=A0.)

8089 CP, CC, TF, channel 1 or 2, Breakpoint (program requested)
*This program must be compiled (large) mode (see line 14 of list code).

Display all registers to CRT, Restore code.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>Programmer:</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM Required:</td>
<td>Company: Intel Corporation</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>Address: 3065 Bowers Avenue</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td>City: Santa Clara</td>
</tr>
<tr>
<td>Assembler/Compiler Used: PL/M-86</td>
<td>State: California 95051</td>
</tr>
</tbody>
</table>
Modified SDK-80 Restart Routine

The original SDK-80 monitor restart routine destroys the carry bit because of the use of the DAD SP instruction before the flags are saved; the DAD SP instruction affects the carry bit.

This routine does not destroy the carry bit.

SDK-80 Kit or any 8080 computer

SDK-80 monitor

This routine replaces the original SDK-80 monitor restart routine. To make use of it the SDK-80 owner should place a JMP 1300 instruction in memory location 13FD (user branch location). Also, he should use RST 7 instead of RST1 to re-enter the monitor. The SDK-80 monitor contains code to branch to location 13FD when a RST 7 instruction is executed.

The JMP 1300 instruction stored in location 13FD transfers control to the modified SDK-80 restart routine that stores the register contents like the SDK-80 restart routine does but the carry bit will not be destroyed. The modified restart routine transfers control back to the SDK-80 monitor (GETCM).

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>ALL</th>
<th>Programmer:</th>
<th>Guenter Ruschitzka</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM Required:</td>
<td>27 Bytes</td>
<td>Company:</td>
<td></td>
</tr>
<tr>
<td>ROM Required:</td>
<td></td>
<td>Address:</td>
<td>Im Bruehl 1</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td></td>
<td>City:</td>
<td>6921 Zuzenhausen</td>
</tr>
<tr>
<td>Assembler/Compiler Used:</td>
<td>8080/8085 Macro Assembler V2.0</td>
<td>State:</td>
<td>Western Germany</td>
</tr>
</tbody>
</table>

8/79
© Intel Corporation, 1978
MDS SERIES II - DUMB TERMINAL

Allows MDS Series II keyboard and CRT to be used as a "dumb" terminal using Serial Port 2 on back panel.

MDS Series II (Model 220 or 230)

ISIS-II Operating System
SYSTEM.LIB library for CO, CSTS routines.

Characters typed on keyboard of MDS.
Characters sent into serial port by modem, etc.
Control-Z terminates program, and returns to ISIS.

Characters typed on MDS keyboard are sent out to serial port and characters received at serial port are sent to CRT on MDS.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>A, C, D, E, SP, H, L</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM Required:</td>
<td>128 Bytes</td>
</tr>
<tr>
<td>ROM Required:</td>
<td>None</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td>2</td>
</tr>
<tr>
<td>Assembler/Compiler Used:</td>
<td>8080/8085 Macro Assembler V2.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Programmer:</th>
<th>Dave Mabry</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company:</td>
<td>Chrysler Corporation</td>
</tr>
<tr>
<td>Address:</td>
<td>CIMS-418-32-03 P.O. Box 1118</td>
</tr>
<tr>
<td>City:</td>
<td>Detroit</td>
</tr>
<tr>
<td>State:</td>
<td>Michigan 48288</td>
</tr>
</tbody>
</table>
FILE GENERATOR

To create and load a source file from an off-line terminal into an ISIS file.

MDS, Bulk storage device - such as Techtran's 800 and 900 Series Disc and cassette terminals.

ISIS-II Systems files

FLGEN File

Flgen opens file for writing and sends a Control-Q to the terminal; thereby starting reading, which continues until reading device supplies a Control-S, at which time "Continue?" is typed. Keying "Y" causes another block (to Control-S) to be read in. Codes which cause assembler errors are eliminated. Keying anything but "Y" causes Flgen to exit to ISIS.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>ALL</th>
<th>Programmer:</th>
<th>Ben Harris</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM Required:</td>
<td>495 Bytes</td>
<td>Company:</td>
<td>Techtran Industries</td>
</tr>
<tr>
<td>ROM Required:</td>
<td></td>
<td>Address:</td>
<td>200 Commerce Drive</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td>4</td>
<td>City:</td>
<td>Rochester</td>
</tr>
<tr>
<td>Assembler/Compiler Used:</td>
<td>8080/8085 Macro Assembler V2.0</td>
<td>State:</td>
<td>New York 14623</td>
</tr>
</tbody>
</table>
PROGRAMMABLE SOFTWARE TIMERS

This program allows a user to set a software timer by specifying the time (# of counts) and vector (Address of subroutine) to be executed when the timer expires. This program allows for 24 timers.

Timer or other device to provide real time clock interrupts.

None

Timeon: Turns on the first available timer. The timer # is returned in the C Reg. This subroutine must be called with the time and vector on the stack.

Timeoff: Turns off the timer who's number is in the C Reg.

Timer: Is the actual interrupt routine that counts the counters and calls for the execution of the subroutines at the associated vector.

After expiration, that timer is turned off.

<table>
<thead>
<tr>
<th>Registers Modified:</th>
<th>None on Interrupt All on setup</th>
<th>Programmer:</th>
<th>Gary Tebbett</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM Required:</td>
<td>Enough for counters</td>
<td>Company:</td>
<td>Consulting Engineer</td>
</tr>
<tr>
<td>ROM Required:</td>
<td></td>
<td>Address:</td>
<td>39 Hill Street</td>
</tr>
<tr>
<td>Maximum Subroutine Nesting Level:</td>
<td>2</td>
<td>City:</td>
<td>Pittsburg</td>
</tr>
<tr>
<td>Assembler/Compiler Used:</td>
<td>8080/8085 Macro Assembler V2.0</td>
<td>State:</td>
<td>California 94565</td>
</tr>
</tbody>
</table>

8/79
© Intel Corporation, 1978

4-765