Bell Laboratories

Cover Sheet for Technical Memorandum

Title- UNIX-INTELLEC MDS Interface

Date- September 1, 1976

TM- 76-3141-1
76-3142-1

Other Keywords- Intel 8080
Microcomputer Software Support

Author(s)	Location and Room	Extension
Joseph E. Lencoski	WH 3F-314	2921
Stuart A. Tartarone	HO 3G-615	4802

Charging Case- 39484-1

Filing Case- 36692-42

ABSTRACT

The recent introduction of the INTELLEC MDS as a debugging tool for Intel 8080 microcomputer systems coupled with the desirability of using the powerful Intel support software available on UNIX PDP-11 for program development required the design of a hardware/software interface. The software package designed permits programs which have been developed on the UNIX system using the SMAL2 compiler (sc), Intel 8080 assembler (as80), and the link-editor (ld80) to be transmitted over a dialed-up connection in pure binary and loaded into the INTELLEC for execution. This memorandum describes the interface, including a synopsis of the new UNIX commands to access these features, and provides program listings of the new software developed.

Pages Text 4 Other 21 Total 25

No. Figures 4 No. Tables 0 No. Refs. 7

Address Label

SEE REVERSE SIDE FOR DISTRIBUTION LIST

BELL TELEPHONE LABORATORIES, INC.

TM-76-3141-1
TM-76-3142-1

DISTRIBUTION
(REFER GEI 13.9-3)

COMPLETE MEMORANDUM TO

CORRESPONDENCE FILES

OFFICIAL FILE COPY
PLUS ONE COPY FOR
EACH ADDITIONAL FILING
CASE REFERENCED

DATE FILE COPY
(FORM E-1328)

10 REFERENCE COPIES

ARREDONDO,G A
+BLECHER,F H
+BROPHY,F J
+CERMAK,I A
+COPP,DAVID H
+COSTELLO,PETER E
+DAVIS,BERNARD P
DI PIAZZA,G
+DONKIN,A E
EHRLICH,N
+EVERHART,J R
+FERCH,R L
FISHER,R E
FRENKIEL,RICHARD H
+HOFMANN,BESSIE B
+JAMES,DENNIS B
JOHNSON,RALPH D
+JONES,DAVID F
+JONES,LARRY M
+JUDD,THOMAS H
+KARIM,M R
+LEMP,RONALD A
+MARANZANO,J F
MATTINGLY,R L
MILLER,J A
+MOLINELLI,JOHN J
NAHABEDIAN,CHARLES E
<O'BRIEN,JAMES A
+PANNONE,L V
+PEKARICH,S P
+PENNOTTI,RAYMOND J
PLITKINS,A
PORTER,PHILIP T
QUINN,T M
+RUSSELL,JESSE E
+SHILLING,STEVE A
+SHORTALL,W E
+SILVERSTEIN,STEVEN M
+SHOLIK,K F
+THOMAS,LEE C
TRIMBLE,DAVID C
+WAGNER,BRUCE D
WEISS,C D
+WILKES,J E,JR
WINGARD,T K
YOUNG,W RAE
ZYSMAN,G I
47 NAMES

COVER SHEET ONLY TO

COVER SHEET ONLY TO

CORRESPONDENCE FILES

4 COPIES PLUS ONE
COPY FOR EACH FILING
CASE

ACKERMAN,A FRANK
AHO,ALFRED V
AHRENS,RAINER B
AITKEN,GEORGE S
ALBERTS,BARBARA A
ALCALAY,D
ALLISON,C E,JR
AMITAY,N
AMRON,IRVING
ANDERSON,C R
ANDERSON,KATHRYN J
ANTOLICK,DAVID R
ARNOLD,S L
ARNOLD,THOMAS F
ARTIS,H P
ATAL,BISHNU S
AVIL,H J
BACCASH,JEANNE M
BACKUS,C F,SR
BALDWIN,GEORGE L
BARRALL,B R
BASEIL,RICHARD J
BATTAGLIA,FRANCES
BAUER,HELEN A
BAUER,STEPHEN M
BAUGH,C R
BAYER,D L
BECKER,RICHARD A
BECK,R P
BECK,ROBERT J
BEL BRUNO,KATHLEEN A
BERGLAND,G D
BERNSTEIN,DANIELLE R
BERNSTEIN,L
BERTH,R P
BIANCHI,M H
BILOWOS,R M
BIRCHALL,R H
BIREN,IRMA B
BLAZIER,S D
BLINN,J C
BODEN,F J
BONANNI,L E
BOROS,VICTOR B
BOURNE,STEPHEN R
BOWERS,JERRY L
BOWRA,JAMES W
BOYCE,W M
BOYLE,GERALD C
BRANDT,RICHARD B
BRANTLEY,JOAN T
BREILAND,JOHN R
BROWN,C W

COVER SHEET ONLY TO

BROWN,JAMES W
BUCHANAN,D N E
BURGESS,JOHN F,JR
BURGIEL,JOSEPH C
BURNETTE,W A
BUTLER,THOMAS T
BYRNE,EDWARD R
CALLAHAN,R L
CANADAY,RUDD H
CANNON,I B
CARDOZA,WAYNE M
CARRAN,JOHN H
CARR,DAVID C
CATO,H E
CAVINESS,JOHN D
CHAFFEE,N F
CHAMBERS,B C
CHAMBERS,J M
CHAMBERS,R P
CHEE,T
CHENG,CHENG-WEN
CHERRY,LORINDA L
CHIANG,T C
CHODROW,M M
CHRISTMAS,W P
CHRIST,C W,JR
CLAYTON,D P
CLOUTIER,J
CLYMER,J C
COBEN,ROBERT A
COHEN,HARVEY
COLE,LOUIS M
COLLINS,J P
CONNERS,RONALD R
COOK,T J
COOPER,J A,JR
COREY,D A
COSTON,W P
CRAGUN,DONALD W
CRANE,R P
CRUME,L L
CSASZAR,MARYANN
CUNNINGHAM,S J,JR
DAVIDSON,CHARLES LEWIS
DAVIS,D R
DAVIS,R DREW
DE GRAAF,D A
DE JAGER,D S
DEEM,G S
DENISON,JOANNA
DERMOND,FAITH L
DEUTSCH,DAVID N
DEVLIN,N V
DI PIETRO,R S
DIETZEL,K ROBERT
DIMMICK,JAMES O
DOLOTTA,T A
DOWDEN,DOUGLAS C
DOWD,P G
DRENNING,C R
DRUMMOND,R E
DSTEFAN,D J

COVER SHEET ONLY TO

DUDICK,ANTHONY L
DUNN,J C
DWORAK,F S
DWYER,T J
EDELSON,DAVID
EDMUNDS,T W
EHLINGER,JAMES C
EITELBACH,DAVID L
ELY,T C
EPLEY,ROBERT V
ERRICHIELLO,PHILIP M
ESSERMAN,ALAN R
ESTES,VIRGINIA DANIELLE
ESTOCK,RICHARD G
EVENSON,E K
FABISCH,M P
FASCIANO,V
FEDER,J
FEINBERG,HENRY R
+FEINER,A
FERRER,NANCY L
FITTERMAN,C H
FISCHER,HERBERT B
FLANDRENA,R
FLEISCHER,H I
FLYNN,MARY L
FOLEY,G
FONTENOT,SHARON M
FOO,YEOW PIN
FORTNEY,V J
FOUNTOUKIDIS,A
FRANKLIN,DANIEL L
FRANK,AMALIE J
FRANK,RUDOLPH J
FRANZ,ANN M
FRASER,A G
FREEDMAN,MARVIN I
FREEMAN,K G
FREEMAN,R DON
FREENY,STAN L
FULTON,A W
GALLANT,R J
GALLI,A T
GALVIN,MICHAEL F
GANNON,T P
GATES,G W
GAY,FRANCIS A
GEERS,T J,JR
GELBER,CHERON L
GERSOWITZ,E S
GEYLING,F T
GIBB,KENNETH R
GIBSON,H T,JR
GIBSON,T A
GILLON,ALEX C
GIMPEL,J F
GINSBERG,N
GITHENS,J A
GLASER,W A
GLASSER,ALAN L
GLUCK,F G
GLYNN,P

COVER SHEET ONLY TO

GOETZ,FRANK M
GOGUEN,N H
GOLABEK,RUTH T
GOLDSMITH,LAWRENCE D,JR
GOLDSTEIN,A JAY
GORMAN,J E
GORTON,D P
GRAHAM,R L
GRANT,GEORGE E
GRASON,JOHN
GRAVEMAN,R F
GREENBAUM,HOWARD J
GREENHALGH,H WAIN
GREENLAW,R L
GRESHAM,J P
GROSS,ARTHUR G
GUIDI,PIER V
GUTHERY,S B
GUTT,DONALD J
HABIB,N
HAGELBARGER,DAVID
HAHNER,MRS IRENE A
HAIGHT,R C
HALAS,WILLIAM P
HALE,A L
HALL,ANDREW D,JR
HALL,J T
HALPIN,T
HAMILTON,LINDA L
HAMILTON,PATRICIA A
HANCOCK,J DAVID
HARTMAN,W H
HARTWELL,WALTER T
HARUTA,K
+HAWKINS,RICHARD B
HAYDEN,DONALD F,JR
HEDRICK,ELLEN L
HELD,R W
HELLER,J ALLEN
HENIG,FRANCES H
HERGENHAN,C B
HESS,B E
HESS,MILTON S
HESTER,S DEAN
HOALST,B C
HONIG,W L
HOOKER,J W
HORLACHER,ROBERT L
HOUPT,WILLIAM H
HOWSON,ROBERT D
HOYT,WILLIAM F
HO,DON T
HO,TIEN-LIN
HUANG,V K L
HUNNICUTT,C F
HUNSBERGER,D J
IMAGNA,C P
IPPOLITI,O D
IRISH,D J
IRVINE,M M
IVIE,EVAN L
JACKOWSKI,D J
...
595 TOTAL

+ NAMED BY AUTHOR > CITED AS REFERENCE < REQUESTED BY READER (NAMES WITHOUT PREFIX WERE SELECTED USING THE AUTHOR'S SUBJECT OR ORGANIZATIONAL SPECIFICATION AS GIVEN BELOW)

MERCURY SPECIFICATION.....

COMPLETE MEMO TO:
3141-SUP 3142-SUP 3144-SUP 3231-SUP

COVER SHEET TO:
COHAMP = COMPUTER MICROPROCESSOR HARDWARE
UNSUP = UNIX/SERVICE,UTILITY PROGRAMS

HO CORRESPONDENCE FILES TM-76-3141-1
HO 5C101 TOTAL PAGES 20

TO GET A COMPLETE COPY:

1. BE SURE YOUR CORRECT ADDRESS IS GIVEN ON THE OTHER SIDE.
2. FOLD THIS SHEET IN HALF WITH THIS SIDE OUT AND STAPLE.
3. CIRCLE THE ADDRESS AT RIGHT. USE NO ENVELOPE.

PLEASE SEND A COMPLETE COPY TO THE ADDRESS SHOWN ON THE OTHER SIDE.

DISTRIBUTION CONTINUED
(REFER GEI 13.9-3)

COVER SHEET ONLY TO

JACKSON,J H
JACOBS,H S
JAKUBEK,R J
JANTZ,R C
JARVIS,JOHN F
JESSOP,W H
JOHNSON,KEITH W
JOHNSON,R L
JOHNSON,STEPHEN C
JOHNSTON,W E,JR
JONES,M B
JONES,S E
JONES,W MARTIN
JUDICE,C N
JUNNIER,R W
KAMINSKY,M L
KAPLAN,ALAN EDWARD
KAPLAN,FRANK
KAPLAN,J J
KAPLAN,MICHAEL M
KAUFELD,J C,JR
KAUFMAN,LARRY S
KAYEL,R G
KEATING,NANCY A
KEESE,W M
+KELLEY,R A
KELLY,L J
KEMP,CHARLES W
KENNEDY,ROBERT A
KERNIGHAN,BRIAN W
KERTZ,D R
KETTERER,BARBARA A
KEVORKIAN,D E
KILLMER,JOHN C,JR
KLAHN,R
KLAPMAN,RICHARD N
KLEIN,RUTH L
KNAKKERGAARD,S F
KNEUER,JOSEPH G
KNOX,BEVERLY G
KNUDSEN,DONALD B
KOBRAN,ALAN S
KOCH,H B
KOETHER,MARY E
KOLB,J
KOLETTIS,NICHOLAS J
KORN,DAVID G
KRIPALANI,ANIL T
LA CAVA,J L
LABONTE,L R
LAIRD,LINDA M
LAMBERT,CONSTANCE A
LANDOLINA,W C,JR
LANGSETH,R E
LANG,ALLEN L
LAYTON,R L
LAZAY,PAUL D
LERNER,EDWARD M
LESSER,P V
LESTER,T E
LEWIS,E A
LICHWINKO,J S
LIEBERT,T A
LINDERMAN,JOHN P
LIND,R O
LIST,W H S
LOGAN,JOHN
LOIKITS,E A
LONG,D W
LONG,P F
LOOP,J D
LORENC,A
LOYD,D G
LUDERER,GOTTFRIED W R
LUDSCHEIDT,INGRID

LUKACS,M E
LYCKLAMA A NYEHOLT,H
LYONS,TERRY G
MAC CASKILL,JANICE I
MAC WILLIAMS,WALTER H,JR
MACHOL,R E,JR
MACLENNAN,CAROL G
MADDEN,DOLORES M
MAINO,J R
MAJERNIK,JOHN F

COVER SHEET ONLY TO

MALIK,JOSEPH M
MANOCHIO,L J
MAROVICH,SCOTT B
MAROWITZ,JAY E
MARSH,MERRILL
MASHEY,J R
MASHEY,RENEE B
MAUNSELL,B I
MAUZEY,P
MAZIK,R F
MC DONALD,L J
MC EOWEN,J R
MC GILL,R
MC MILLAN,L G
MC MILLAN,W F
MC MULLEN,E C
MCILROY,M DOUGLAS
MEINKE,J M
MENIST,DAVID B
MENNINGER,R E
MENON,P R
MENZEL,FRANCES S
MERCADANTE,P F
METZ,R F
MIGLIORE,SANDRA A
MIHALOVIC,R P
MIKOLAJEK,J M
MILLER,NORMAN R
MILLER,RANDALL E
MILLER,STEVEN G
MILLSPAUGH,JOHN A,JR
MILLS,APLINE D
MILNE,D C
MITCHELL,OLGA M MRACEK
MOFFITT,L R
MORGAN,DENNIS J
MORGAN,SAMUEL P
MUEHTER,V A
MUI,LILY C
MULLER,R ALLAN
MYERS,FRANK H
NANTZ,T D
NAU,CARL DAVID
NEHRLICH,W R
NELSON,D R
NEMCHIK,JOSEPH M
NEY,LESLIE
NOWITZ,D A
O SHEA,W T
OLSEN,RONALD G
OPFERMAN,D C
O'BRYAN,H M,JR
O'NEILL,D M
O'SULLIVAN,JOHN A
OSAJIMA,Y R
OSOLINIK,F J,3RD
OSSANNA,J F,JR
OWENS,G J
OWENS,GENE L
PANIGRAHI,GODAVARISH
PARA,PAUL S
PARKER,JAMES C,JR
PASCHBURG,RICHARD H
PATEL,C K N
PEARLMAN,BERTRAM
PEARLMAN,MARY ELLEN
PECK,W DOUGLAS
PENNINO,THOMAS P
PERDUE,ROBERT J
PEREZ,CATHERINE D
PERNESKI,A J
PERRY,J L
PETERSON,RALPH W
PETERSON,T G
PETERS,KRISTINE J

PETROSKY,J S,JR
PETSCHENIK,N H
PETTIT,GEORGETTE
PFISTER,R G
PHILLIPS,S J
<PHIPPS,GARY W
PILLA,M A
PIRZ,FRANK C
PISKORIK,ELAINE M
POLLAK,H O

COVER SHEET ONLY TO

POLUCKI,S R
PRIM,R C
PRINS,G C
PRINTZ,KARLA J
PRYOR,ROGER W
PUCCI,M F
PUERLING,BRUCE W
QUINN,MARGARET E
RATTA,G A
RECCHIA,O
REED,S C
REHERT,ALLEN F
REINAGEL,DAVID J
REISINGER,CARL E
REMDE,J R
REMS,S B
REXING,GERALD LEE
RHODES,STEVE
RICHARDS,GARY F
RIDDLEBERGER,C O
<RIDDLE,GUY G
RINALDY,E A
RITACCO,J E
RIZZO,J F
ROBBINS,MARILYN F
ROBERTS,L W
ROBINSON,H E
ROBINSON,THOMAS W
<ROCHKIND,M M
ROCHKIND,MARC J
RODRIGUEZ,ERNESTO J
ROFRANO,ARTHUR J,JR
ROJEK,RICHARD C
ROMITO,LETITIA J
ROM,DOUGLAS B
ROOME,W D
ROSENBLUM,M J
ROSENFELD,PETER E
ROSIN,ROBERT FISHER
ROSLER,LAWRENCE
ROSZKOWSKI,R J
ROVEGNO,HELEN D
ROZENBLIT,M
RUGGIERO,G N
RUTT,T E
SABSEVITZ,A L
SALMON,RUTH L
SAND,DOUGLAS S
SATO,S M
SATZ,LAWRENCE R
SCARBOROUGH,P E,JR
SCAVEZZE,DANIEL C
SCHAEFER,EDWIN A
SCHLEGEL,C P
SCHNITZLER,PAUL
SCHRAMM,G W
SCRIBNER,N M
SEAGER,EDWARD M
SEARS,E R
SEEBACH,R J
SEKINO,LUCIA C
SEREN,R J
SHADLE,CHRISTINE H
SHANAHAN,J M,JR
SHANK,JERE C
SHANK,R A
SHELTON,JAMES C
SHIELS,J A
SHIRTZ,ANN L
SHOMO,ROBERT E
SHORTER,J W
SHUPE,C F
SIDOR,DONALD J
+SIMONE,C F
SINGH,R P

SINOWITZ,NORMAN R
SIX,F B
SMITH,B L
SMITH,DALE W
SMITH,KIME R,JR
SOBOTKA,L J
SOUTHERN,M J
SO,H C
SPIRES,R J
ST AMAND,PAUL G

COVER SHEET ONLY TO

STAMPFEL,J P
STANZIONE,DAN C
STARRICK,KATHLEEN K
STEFFER,V G
STOCKTON,LEE A
+STOKES,R R
STORM,A R,JR
STORTZ,NANETTE E
STROHECKER,CARY A
STUKAS,LORETTA I
STUK,GREGORY J
STUMPF,PETER W
STURMAN,JOEL N
SWANSON,G K
SWANSON,RONALD H
SWIFT,R E
SWITZER,K
TAGUE,BERKLEY A
TAIT,G RODNEY
TAMMARU,ENN
TANG,Y
TARTARONE,STUART A
TERRY,G S
TEST,JACK A
THAYER,D C
THOMAS,PHYLLIS J
THOMPSON,BERNARD E
THOMPSON,K
THOMPSON,MARYJANE
THOMPSON,MICHAEL J
TING,ANNSHENG CHIEN
TRIMBLE,L E
+TROE,J L
TRUXEL,G H
TURNER,J H I
TUTELMAN,DAVID M
USAS,ALAN M
VAN OKNUM,J H
VASSALLO,J C
VIGGIANO,F A
VOGEL,DENNIS R
VOGEL,GERALD C
VYSSOTSKY,V A
WALDHUTER,JEFFREY S
WALKER,ELIZABETH A
WALKER,J CALVIN
WALKER,J T
WALLACE,RICHARD E
WALTZ,JAMES J
WALVICK,EDWARD A
WANDZILAK,PHILIP D
WANG,T L
WARD,J C
WARNER,JACK L
WARNER,TERRY L
WATSON,D S
WATSON,R E,JR
WEBB,F J
WEHR,LARRY A
WEIGAND,W F
WEINBERGER,PETER J
WEISSMANN,GERD F
WEISS,ZVI ISRAEL
WENING,G J,JR
WEXELBLAT,R L
WEYTHMAN,JAMES E
WHIPPLE,JOHN H
WHITE,RALPH C,JR
WILLIAMS,JOHN G
WILLIAMS,R D
WILSON,D E
WILSON,GEOFFREY A
WINCKELMANN,W A
WOLFE,ROBERT M
WONSIEWICZ,B C

WOODRUFF,JOHN L
WOOD,JOSEPH L,III
WO,Y K
YAFFEY,C L
YAMIN,M
YANG,J N
YATES,G H
YEN,T C
ZDAN,D W
ZEBO,T J

COVER SHEET ONLY TO

ZECHER,B I
ZEPP,HILTON E
ZIMMITTI,WENDY W
ZINNES,A E
595 NAMES

Bell Laboratories

subject: UNIX-INTELLEC MDS Interface - Case 36692-42

date: September 1, 1976

from: Joseph E. Lencoski
Stuart A. Tartarone

TM76-3141-1
TM76-3142-1

MEMORANDUM FOR FILE

INTRODUCTION

The recent introduction of the INTELLEC MDS as a debugging tool for Intel 8080 microcomputer systems coupled with the desirability of using the powerful Intel support software available on UNIX[1] PDP-11 for program development required the design of a hardware/software interface. The software package designed permits programs which have been developed on the UNIX system using the SMAL2 compiler[2] (sc), Intel 8080 assembler[3] (as80), and the link editor (ld80) to be transmitted over a dialed-up connection in pure binary and loaded into the INTELLEC for execution. This memorandum describes the interface, including a synopsis of the new UNIX commands to access these features, and provides program listings of the new software developed.

This work was performed in conjunction with the development of a testing facility[4] for the mobile logic units to be used in the High-Capacity Mobile Telecommunications System (HCMTS)[5]. The logic unit is a microcomputer-based design using an Intel 8080 as the main controller.

SOFTWARE

UNIX Blocking Program

Executable object modules produced by the UNIX Intel support software are stored in a file called 80.out. The structure of the 80.out file, as illustrated in Figure 1, is not in a form which is readily acceptable for direct transmission to the INTELLEC. Checksums are not provided and unnecessary information (relocation information and symbol table) for program execution is appended. Thus, a reformatting of this file is necessary.

A program called "mds80" written in "C"[6] reads in the 80.out file and blocks it into a format which may be transmitted over a dialed-up connection and is compatible with a binary loading program (see below) developed for the INTELLEC. The format of the resulting file is illustrated in Figure 2. It consists of a dummy record of 16 ASCII "space" characters, followed by a sequence of blocked object records for the text segment, followed by a second dummy record of 16 ASCII "space" characters, concluding with a sequence of blocked object records for the data segment. If either program segment does not appear in 80.out, the corresponding object record in the blocked file is not produced. Each block holds a maximum of 256 bytes of executable code.

The format of the blocked object record appears in Figure 3. The first byte is a sync character which is represented by ASCII cntlq. This character is used by the binary loading program in the INTELLEC as a block delimiter. The second byte is the total byte count (reduced by 1) for the block (0 through 255). The next two bytes contain the hi and lo bytes, respectively, of the origin in the INTELLEC for the specific block. Then the object program follow coded in binary. Each byte represents one byte of the resulting 8080 program. Up to 256 bytes may be contained in each record. The final byte contains the checksum which is the 8-bit sum of the byte count, address, and each object byte.

The input to mds80 consists of a list of loaded object modules produced by the compile or link step. If no input file is submitted, 80.out is assumed. mds80 blocks each file independently and stores the output on a file suffixed .mds. mds80 (see Appendix A for a listing) reads the 80.out header to determine the origin and size of the text and data segments of the programs. For each segment of the program, it sets up the dummy block of 16 ASCII space characters; it then creates an object record for each 256-byte block until all bytes of each segment are exhausted. A checksum is computed and inserted at the end of each record. Note that mds80 assumes the structure of the header segment to be in Figure 1. If an earlier version of as80 is used, the structure fh should be altered.

UNIX Transmission Program

Once the blocked object module is produced, it must be transmitted to the INTELLEC. This is accomplished by executing a program called "send80" which sets up the proper machine modes to output data in pure binary using the UNIX stty command. The modes which must be set are hardware dependent and vary from machine to machine. A program which accomplishes this for the Laboratory 323 PDP-11/45 is illustrated in Appendix B. If used on another machine the local operations analyist should be consulted to determine the proper mode settings.

INTELLEC Binary Loading Program

A program called "bindld80" written in the Intel 8080 assembly language loads the object data transmitted by UNIX. The program whose source is contained in Appendix C must be called with DE set to the byte offset from the program origin contained in 80.out to the load point, specified in two's complement notation. This permits a program to be loaded into RAM for subsequent "burning" and execution in PROM. To load the program at the origin specified in 80.out, DE would be set to zero.

The loading program inspects data received from the CRT input port of the INTELLEC until the sync character (cntlq) appears. Once the sync character is received, the byte count, hi and lo addresses are received and stored with the proper offset added in. Then each data byte is received and stored in memory until the completion of a block. The checksum is computed and compared at the conclusion of a block. If an error is detected the program returns to the driving routine with the accumulator set to one. If valid, the incoming stream is inspected for a cntlq (for a second block) or for the UNIX prompt character (%) which indicates that transmission has completed. A successful load will return a zero to the accumulator and set the first load address in the HL register stored in standard Intel hi-lo format. Note that this program precludes loading data into memory locations beginning with byte 65,280 (the last 256 bytes of addressable memory). It is suggested that a driving program be written to call binld; the driving program could inspect load status upon return and display appropriate messages.

HARDWARE

Hardware is required to perform the necessary switching functions to permit a single terminal (keyboard/display) to interface with a modem (remote computer) and with the INTELLEC, and to permit the INTELLEC to directly interface with the modem. A switching circuit designed by Department 4391* suits this purpose; its block diagram is illustrated in Figure 4.

Generally switches 1 and 2 are activated which allow the display to monitor both the INTELLEC and the modem. To develop a program on UNIX, switch 3 would be activated. After an object module has been produced and blocked into the correct form using "mds80", switch 4 would be activated momentarily to initiate execution of the INTELLEC loading program (bindld). Switch 5 would then be activated to permit the modem output (the resulting object module) to

---

* Private communication with Dick Ferch.

be fed directly into the INTELLEC upon requesting UNIX to transmit the data using "send80". Switch 6 although not required in this application could be used to send data from the INTELLEC to a remote computer.

CONCLUSIONS

This memorandum described a hardware/software support package which establishes an interface between an INTELLEC MDS and a UNIX PDP-11 system permitting program development on UNIX and loading into the INTELLEC for execution. These programs will be available shortly as commands on the Laboratory 323 PDP-11 computer system.

Joseph E. Lencoski

Stuart A. Tartarone

WH-3142-JEL
HO-3141-SAT-cp

Att.
References
Appendices A through C
Figures 1 through 4

REFERENCES

[1] B. W. Kernighan, "UNIX for Beginners," TM74-1273-18, October 29, 1974.

[2] D. H. Copp, "Users' Guide to SMAL2 Language for the Intel 8080 Microprocessor," to be published.

[3] D. L. Bayer, "Assembler User's Manual," Memorandum For Record, Ocrober 2, 1974.

[4] S. A. Tartarone, "MOLOSEX - A Microcomputer Testing Facility for the HCMTS Mobile Logic Unit," Proceedings of the 1976 Bell Laboratories-Western Electric Microcomputer Symposium, to be published.

[5] P. T. Porter, "High Capacity Mobile Telecommunications System (HCMTS) 1975 Overview (Issue 2), Case 36692-42, Memorandum For File, February 17, 1976.

[6] D. M. Ritchie, "C Reference Manual," TM74-1273-1, January, 1974.

APPENDIX A

MANUAL PAGES AND PROGRAM LISTING FOR MDS80

NAME

mds80 - object file blocking program for INTELLEC MDS

SYNOPSIS

mds80 [file...]

DESCRIPTION

Mds80 is the UNIX blocking program which reformats the the object file produced by sc, as80, or ld80 into a format compatible for entry into the INTELLEC MDS using a companion binary loading program (see binld80 (I)).

Mds80 uses the list of files specified as input and leaves the output on the file whose name is that of the input with '.mds' replacing '.o', '.out', or ''. If no file is specified, 80.out is used.

The blocked object module may be read into the INTELLEC using send80.

FILES

80.out loaded output

SEE ALSO

as80(I), binld80(I), ld80(I), send80(I), sc(I)

BUGS

```
#

/*
 * mds80
 * functional name:              convert object file
 * functional description:       this program converts the object
                                 module generated by 80.out
                                 into a blocked format to be loaded
                                 by the intellec
 * calling parameters:           list of object file names
                                 if no file is specified, 80.out
                                 is assumed
 * returns:                      none

 * program redesigned:           8/24/76
 * programmer:                   Stu Tartarone

 */


/*

        Format of blocked object file

                Dummy Record
                Text Segment Object Record 0
                        .  .  .
                Text Segment Object Record n
                Dummy Record
                Data Segment Object Record 0
                        .  .  .
                Data Segment Object Record n

        Format of dummy record

                16 Ascii Space Characters

        Format of Object Record

                        byte
                        1       sync char (ctl-a)
                        2       data byte count (0-255)
                        3       hi-address of block origin
                        4       lo-address of block origin
                        5       object byte 1
                        6       object byte 2
                        .
                        .
                        .
                        n+4     data byte n     (0<=n<=255)
                        n+5     checksum (8 bit sum of byte count,
                                          address, and data)
```

```
 */



/*
 * Size of resulting blocked object records
 */

#define BLKSIZE 256
#define MAGIC   413



/*
 * format of the header for an 80.out file
 */

struct fh
{
        int     magic;
        char    *tsize;
        char    *dsize;
        char    *bsize;
        char    *ssize;
        char    *orgs[3];
        char    *rsize;
        int     flags;
        char    pad[12];
};

/*
 * Input Data Structure
 */

        struct findate
        {
                char data[BLKSIZE];
        } ;


/*
 * output object file structure
 */
        struct  foutdata
        {
                char sync;
                char blksize;
                char hiaddr;
                char loaddr;
                char outdata[BLKSIZE];
        } ;
```

```
main (argc,argv)
int argc;
char *argv[ ];
{
        char *defaul;
        defaul = "80.out";
        if (argc == 1)
                mds(defaul);
        else
                while (--argc != 0)
                {
                        ++argv;
                        mds(argv[0]);
                }
        exit();
}

mds(name)
char *name;
{
        int bytecnt,f1,f2,blk,i,j;
        int address[2];
        int size[2];
        char cksum;
        char *dummy;
        struct fh hdr;
        struct findata inrecr;
        struct foutdata outrecr;
        char outnm[16];

/*
 * input file opening and error handling
 */

        if( (f1=open (name,0))   < 0)
        {
                printf ("can't open %s\n",name);
                return;
        }

/*
 * initialization
 */
        outrecr.sync     = 021;                 /* initialize sync */
        read (f1,&hdr,sizeof hdr);              /* reads  header data */

/*
 * check magic number
 */

        if (hdr.magic != MAGIC)
        {
                printf("improper format %s\n",name);
                close(f1);
                return;
        }
```

```
/*
 * create name.mds as output file
 */

        for (i = 0; ((name[i] != '\0') && (name[i] != '.')); i++)
                outnm[i] = name[i];
        outnm[i] = '.';
        outnm[++i] = 'm';
        outnm[++i] = 'd';
        outnm[++i] = 's';
        outnm[++i] = '\0';
        if( (f2=creat (outnm,0666)) < 0)
        {
                printf ("can't create %s\n",outnm);
                close(f1);
                return;
        }



/*
 * Two separate convert operations will be performed.  The first convert will
 * block the text segment; the second convert will block the data segment
 */

        address[0] = hdr.orgs[0];
        address[1] = hdr.orgs[1];
        size[0] = hdr.tsize;
        size[1] = hdr.dsize;
        dummy = "                ";              /* set up 16 blanks

        for (j = 0; j < 2; j++)
        {
                bytecnt = 0;                    /* initialize byte count */


/*
 * output dummy blank string
 */

                write (f2,dummy,16);

/*
 * read and write data
 */

                while (size[j] > bytecnt)
                {
                        if ( (blk = size[j] - bytecnt) >= BLKSIZE)
                                blk = BLKSIZE;
                        outrecr.blksize = blk + 1;
                        outrecr.hiaddr = address[j] / 256;
                        outrecr.loaddr = address[j] % 256;
                        cksum = blk + outrecr.hiaddr + outrecr.loaddr;
```

```
                        read (f1,&inrecr,blk);
                        for (i = 0; i < blk; i++)
                        {
                                cksum =+ inrecr.data[i];
                                outrecr.outdata[i] = inrecr.data[i];
                                bytecnt++;
                        }
                        write (f2,&outrecr,blk + 4);
                        address[j] =+ blk;
                        write (f2,&cksum,1);
                }
        }
        close(f1);
        close(f2);
        return;

}
```

APPENDIX B

MANUAL PAGES AND PROGRAM LISTING FOR SEND80

NAME

send80 - sends the blocked 8080 object module to the INTELLEC MDS

SYNOPSIS

send80 name ...

DESCRIPTION

Send80 transmits a blocked 8080 object module (see mds80(I)) over a dialed-up link to the INTELLEC MDS. The INTELLEC MDS binary loading program (see binld80 (I)) must be invoked prior to executing this command. Name is the list of files to be transmitted.

FILES

SEE ALSO

as80(I), binld80(I), ld80(I), mds80(I), sc(I)

BUGS

Process cannot be interrupted once invoked.

```
#
/*
 * send80

 * functional name:              send 8080 object code

 * functional description:       this program transmits a file in pure
                                  binary to the standard output  for
                                  loading of object code into an
                                  INTELLEC system.

 * calling parameters:  list of files to be output

 * returns:                      none

 * program designed:    5/1/76
 * program redesigned:  8/24/76

 * programmer:          Stu Tartarone

 */


/*
 *      tty flags
 */
#define SPECIAL 01
#define XTABS   02
#define LCASE   04
#define ECHO    010
#define CRMOD   020
#define RAW     040
#define ODDP    0100
#define EVENP   0200
#define ANYP    0300
#define T8BIT   0100000
#define NLDELAY 01400
#define TBDELAY 06000
#define CRDELAY 030000
#define VTDELAY 040000

/*
 *      types
 */
#define ZAPPER  01

/*
 *      delay algorithms
 */
#define CR1     010000
#define CR2     020000
#define CR3     030000
#define NL1     0004000
#define NL2     001000
#define NL3     001400
#define TAB1    002000
```

```
#define TAB2    004000
#define TAB3    006000
#define FF1     040000

        speeds
 */
#define B110    3
#define B150    5
#define B300    7
#define B1200   9
#define B2400   11
#define B9600   13

main(argc,argv)
int argc;
char *argv[ ];
{
        int f1;
        if (argc == 1)
        {
                printf("\nno source file specified\n");
        }
        else
                while (--argc != 0)
                {
                        ++argv;
                        send(argv[0]);
                }
        exit();
}


send(name)
char *name;
{
        int n;                  /* input count                          */
        int f1;
        int v[3];               /* standard mode                        */
        int w[3];               /* pure binary mode                     */
        int buf[512];           /* buffer for output                    */

/*
* file opening
 */

        if ( (f1 = open(name,2)) < 0)
        {
                printf("\ncan't open %s\n",name);
                return;
        }

/*
 *      change modes, output data, change back to standard mode
 */
```

```
        gtty (0,v);                 /* read standard mode                */
        w[0] = v[0];
        w[1] = ZAPPER;
        w[2] = SPECIAL|ANYP|T8BIT|RAW;
        stty (0,w);                 /* set up new mode                   */
        while ( (n=read(f1,buf,512)) != 0 )
                write (1,buf,n);
        stty (0,v);                 /* set up old mode                   */
        close(f1);
        return;
}
```

APPENDIX C

MANUAL PAGES AND PROGRAM LISTING FOR BINLD

NAME

binld80 - lists the binary loading program for the INTELLEC MDS

SYNOPSIS

binld80

DESCRIPTION

Binld80 lists the binary loading program for the INTELLEC MDS which is compatible with the output produced by mds80. The output of this command should be directed to a file. The editor should then be invoked to specify an origin for the program and to possibly precede the routine with a driving program written by the user. The result should then be assembled using as80. The resulting object code should then be burned into PROM and installed in the INTELLEC MDS.

To load a program , binld80 should first be invoked in the INTELLEC; then send80 should be used to transmit the object module preprocessed by mds80 over a dialed-up connection into the INTELLEC.

FILES

/lib/binld80

SEE ALSO

as80(I), mds80(I), send80(I)

BUGS

```
////////////////////////////////////////////////////////////
/
/
/ binld
/ functional name:          binary loader
/ functional description:   this program received an object file
/                           generated by mds80 and stores it in
/                           the appropriate memory locations in the
/                           INTELLEC.
/ calling parameters:       DE set to offset from load address
/ returns:                  A = 0 => successful load
/                             = 1 => unsuccessful load
/                           HL = first load address
/ program designed:         5/1/76
/ programmer:               Joe Lencoski
/
/       An origin should be specified for the program
/       by modifying the textorg statement
/
crtin = 0xfcb9  / MDS crt input subroutine
/
/
/


        .textorg 0x0000
        .globl binld
binld:
        / binary load subroutine
        /
        /
        / look for cntl-a sync char
        push    b
        mvi     h,0xff  /first flg
        push    h
binld0:
        call    crtin
        cpi     0x11    /ctl-a
        jnz     binld0
        jmp     binld2
binld1:
        call    crtin
        ani     0x7f    / delete parity
        cpi     '%'
        jz      bend
        cpi     0x11    / ctl-a sync
        jnz     binld1
binld2:
```

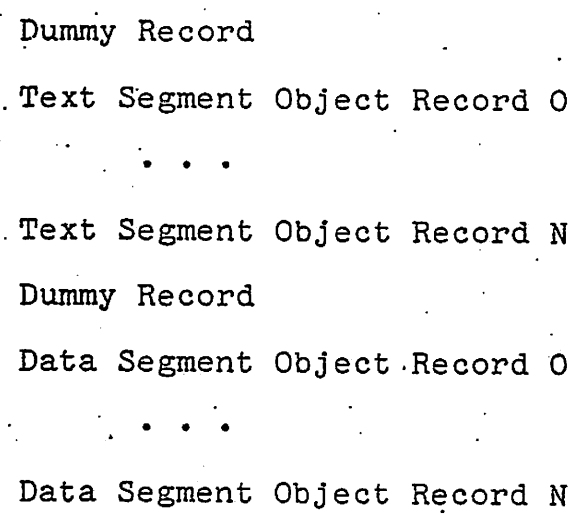
```
        mvi     c,0     / computed checksum reg
/ get byte count
        call    crtin
        mov     b,a
        add     c
        mov     c,a
/ get hi addr in h
        call    crtin
        mov     h,a
        add     c
        mov     c,a

/ get lo addr in l
        call    crtin
        mov     l,a
        add     c
        mov     c,a
        dad     d       / add offset addr
        xthl            / hl <=> *sp
        mvi     a,0377
        cmp     h
        jnz     binld3  / jump not first
        pop     h
        push    h
binld3: xthl
/ get data
bdata:
        call    crtin
        mov     m,a
        add     c
        mov     c,a
        inx     h
        mov     a,b
        dcr     b
        ana     a
        jnz     bdata
/ get checksum
        call    crtin
        inr     c       / adjust for byte ct =1
        cmp     c
        jz      binld1
        mvi     a,1     / error on checksum
        jmp     bendr
/ normal termination
bend:
        xra     a
bendr:
        pop     h       / first load addr
        pop     b
        ret
```

- Header (includes segment counts and origins)
  - magic no.
  - text count
  - data count
  - bss count
  - symbol table count
  - text origin
  - data origin
  - bss origin
  - relocation count
  - flags
  - 6 integers reserved
- Object Code for Text Segment
- Object Code for Data Segment
- Relocation Information
- Symbol Table

Format of 80.out

Figure 1



Format of Blocked Object Module

Figure 2

Text segment data record:

byte	
1	sync char (cntlq)
2	data byte count (0 through 255)
3	hi-address of block origin
4	lo-address of block origin
5	object byte 1
6	object byte 2
. . .	
n+4	object byte n (1 < = n < = 256)
n+5	check sum (8-bit sum of byte count, address, and data)

format of Blocked Object Record

Figure 3

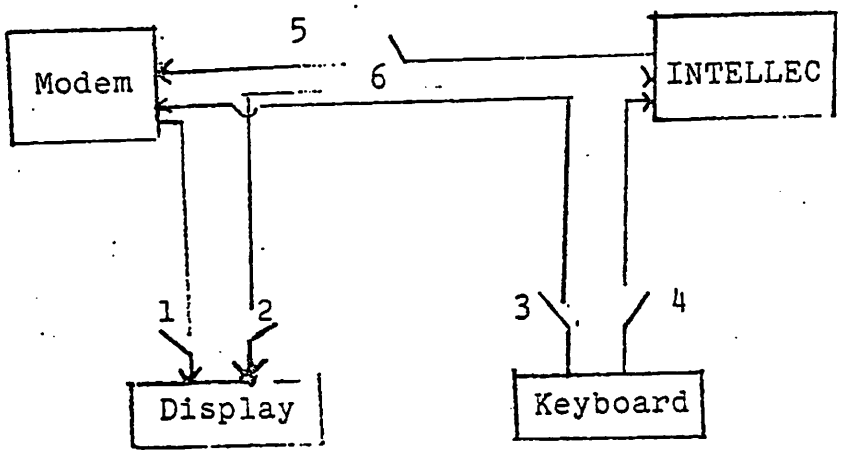


Figure 4