LBL-12041 c.2
UC-32

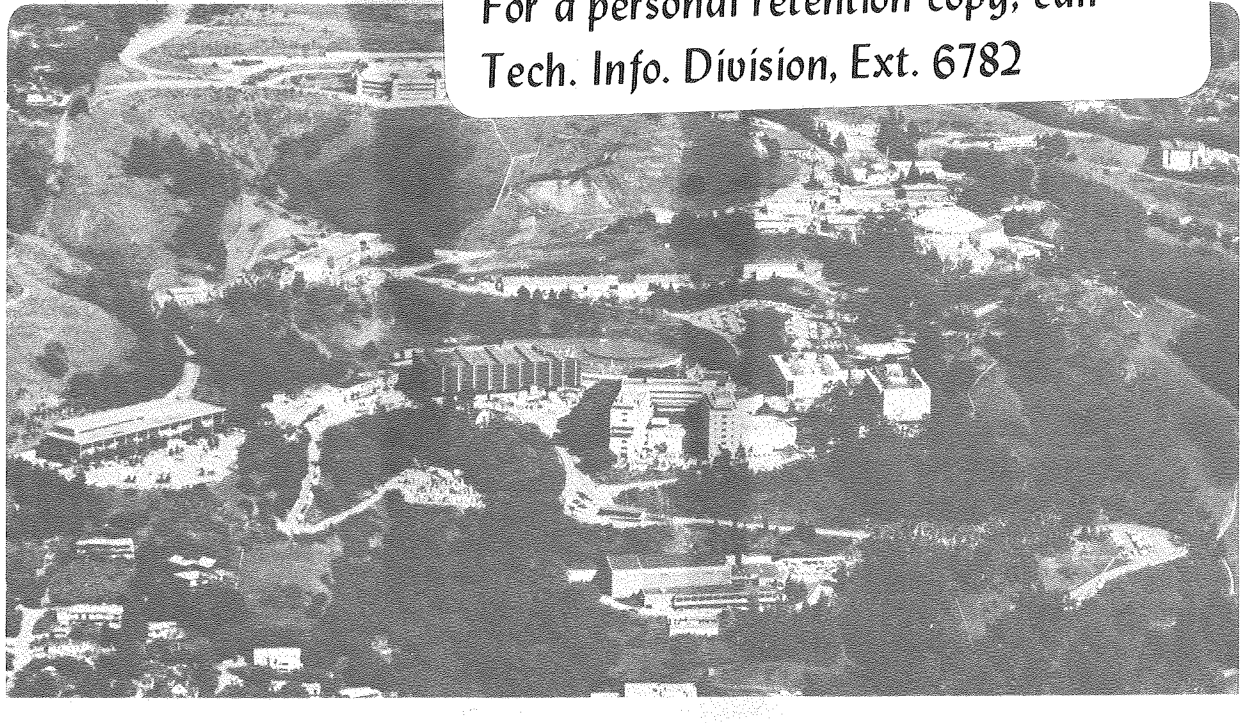
Lawrence Berkeley Laboratory
UNIVERSITY OF CALIFORNIA

Engineering & Technical
Services Division

# AN INTERFACE BETWEEN A PRINTED CIRCUIT BOARD COMPUTER AIDED DESIGN TOOL (Tektronix 4051 based) AND A NUMERICAL PAPER TAPE CONTROLLED DRILL PRESS (Slo-Syn 530 - 100 w/ Dumore Automatic Head Number 8391)

Bradford K. Heckman and Victor K. Chinn

January 1981



LBL-12041 c.2

Prepared for the U.S. Department of Energy under Contract W-7405-ENG-48

# AN INTERFACE BETWEEN A PRINTED CIRCUIT BOARD COMPUTER AIDED DESIGN TOOL (Tektronix 4051 based) AND A NUMERICAL PAPER TAPE CONTROLLED DRILL PRESS (Slo-Syn 530 - 100 w/ Dumore Automatic Head Number 8391*)

Bradford K. Heckman
and
Victor K. Chinn

Lawrence Berkeley Laboratory
University of California
Berkeley, CA 94720

January 1981

* This work was supported by the U.S. Department of Energy under contract No. W-7405-ENG-48.

# AN INTERFACE BETWEEN A PRINTED CIRCUIT BOARD COMPUTER AIDED DESIGN TOOL (Tektronix 4051 based) AND A NUMERICAL PAPER TAPE CONTROLLED DRILL PRESS (Slo-Syn 530 - 100 w/ Dumore Automatic Head Number 8391*)

Bradford K. Heckman
and
Victor K. Chinn

Lawrence Berkeley Laboratory
University of California
Berkeley, CA 94720

January 1981

## INTRODUCTION

This capability is a result of an ongoing project sponsored by the Special Projects Department of the Electronics Engineering Section of the Engineering and Technical Services Division. It was designed and implemented with the assistance of personnel of the Computer Applications Department of the Computing Section of the Engineering and Technical Services Division. The overall goal of the project is to provide a resource which will allow increased productivity of printed circuit boards (PCB's or PC's) at Lawrence Berkeley Laboratory.

## THE PROBLEM

In the Special Projects Department a PCB is currently designed using the Tektronix 4051, plotted on paper, and photographed. A clear overlay is produced from the photograph. The overlay is used to chemically create an "etched" board which needs holes drilled to be completed. The problem addressed in this report is the labor intensive and inaccurate process of a human operator drilling the holes by sight.

---

* This work was supported by the U.S. Department of Energy under contract No. W-7405-ENG-48.

## THE SOLUTION

The ability to control a drill press via paper tape already existed at LBL.

Three programs were written to provide the automated interface between the Tektronix 4051 and the drill press:

NC5 is a vendor supplied program (I Squared, Inc., of Berkeley) modified to place its output on a Tektronix 4051 file instead of directly onto a physical paper tape

NCTOLBL converts the NC5 file to a format compatible with the LBL drill press

USRSEND sends the converted file to the Computer Center CDC 6600B computer to punch a paper tape

These three programs are all contained on one cassette and designed to be executed one after another for the convenience of users. Users need only enter the name of program they want. Users may optionally have each program monitor their operation in a "verbose" fashion to assist in spotting suspected errors immediately. Users may flexibly allocate personal time to complete a process by running all three programs at once or at different times.

The Tektronix 4051 produces an "industry standard" format paper tape code which is not compatible with the LBL drill press in the following ways:

ASCII instead of EIA character code

Absolute X,Y coordinates instead of relative X,Y coordinates

Drill commands not at all compatible

The I Squared, Inc., NC5 program directly punches an "industry standard" paper tape. It was modified to place its output on a Tektronix 4051 "source" file instead. Program NCTOLBL reads the source file and converts it to a Tektronix 4051 "object" file, eliminating the three incompatibilities listed above. To punch the paper tape, a hardwire connection (Develcon) was created from the Tektronix 4051 to the Computer Center CDC 6600B machine and program USERSEND was written to transfer the object file by implementing a communications protocal called "ASDTP" (A Simple Data Transfer Protocol). ASDTP is a Computer Center product designed to optimize both data transfer and integrity. USERSEND also sends the transfered file to the physical paper tape punch. The paper tape may be picked up at the Computer Center and taken to the drilling press.

## OPERATION OF PROGRAMS

A familiarity of the current I Squared, Inc., Software for designing PCB's on the Tektronix 4051 is assumed. After the board is designed a file containing all the hole coordinates and the design of the board is produced and stored on a diskette. This diskette must be used as the input to the Sort 1A or Sort 1B sort which must be run before using the three programs designed for this project.

The following is a step by step operation of the complete process:

| | | |
|---|---|---|
| (1) | Design the PCB completely | Produces a "Hole" file diskette |
| (2) | Insert the NC50 cassette and press "Auto Load" | |
| (3) | Select either Sort 1A or Sort 1B (Requires diskette from step 1) (Be sure to place the write enable tab on flopy disc) (A more efficient "farmer's plow sort" may be run after the sort chosen above) | Creates a Hole Sorted diskette |
| (4) | Insert the NC to LBL cassette and press "break" to exit the NC50 program, press "return" and then press "Auto Load" (Baring user intervention, the following three steps should then occur automatically) | |
| (5) | "NC5" executes and creates a "Source" file (Requires diskette from step 3) | Produces a "Source" diskette |
| (6) | "NCTOLBL" executes and produces an "Object" file (Requires diskette from step 5) | Produces an "Object" diskette |

```
 (7)  "USRSEND" executes          !
      and asks if user            !
      wants to enter terminal     !
      mode.  Respond "Y" and then !
      press the button on the     !
      terminal support box (TSB). !
      Wait for the TSB green light!
      and then type "RS".         !
      Respond to "Request:" by    !
      typing "RECC" and "return". !
      Now log into the computer   !
      center "B machine" by enter-!
      ing these 2 lines:          !
      ">LB"                       !
      "CAD,12,999.494402,YOUR NAME,$<password>"!
      After the response          !
      "OK - SESAME" enter these   !
      2 lines:                    !
      "FETCHPS,NCLBL,CAD,CAD."    !
      "CALL,CAD1."                !
                                  !
 (8)  After response from program,!
      hit the UDK button # 5 on   !
      the Tektronix 4051. Then the!
      "USRSEND" program will con- !
      tinue and send data to the  !
      6600B computer.             !
                                  !
 (9)  When the program is finished! ** Ask the I/O operator to
      (response "OK - SESAME")    ! fill out the COS card.
      dial 6211 and have the      ! The paper tape is
      operator fill out a COS card! Stores Item # 7530 - 53345.
      by giving the Jobname, Acct !
      number, your name, and the  !
      approximate length of the   !
      punched tape.  Specify      !
      MYLAR tape when punching the!
      tape;      MYLAR tape must  !
      be user supplied beforehand.!
      (Enter ">" for the 7 char   !
      Jobname)                    !
      (Estimate 10 holes per foot !
      of punched tape)            !
                                  !
(10)  To actually punch the paper ! Produces a paper tape at the
      tape and terminate the job  ! Computer Center: Turnaround
      on the "B machine" enter:   ! time is approx 1 hour.
      "CALL,CAD2."                !
                                  !
(11)  After the response          !
      "Job ended - Disconnected"  !
      press the TSB box for the   !
      red light and press UDK     !
      button #5 on the Tekronix   !
      to return to the 4051.      !
==================================================================
```

## GENERAL COMMENTS

The production of paper tapes at the Computer Center via the hardware link and the software protocol takes up most of the time in the process of producing paper tapes. A time saving idea is to purchase a paper tape punch and directly have it connected to the Tektronix 4051. This would eliminate the time used sending data to the CDC 6600B computer, waiting for the operator to mount the paper tape to be punched. (It would also eliminate the cost of computer time used on the 6600B computer.) Finally, it would eliminate the time used picking up tapes at the Computer Center.

There is currently in existence a "Micro-Punch" station which could easily be used to effectively utilize the above option mentioned above.

## PROJECT FILES

| | |
|---|---|
| [CSAM-UNIX]/ua/brad/ArtTurney/Overview | Of the project |
| [CSAM-UNIX]/ua/brad/ArtTurney/Log | Brad Heckman's project log |
| [CSAM-UNIX]/ua/vic/ArtTurney/log | Vic Chinn's project log |
| [CSAM-UNIX]/ua/vic/ArtTurney/PCdoc | This report |
| [TEK 4051] /Master Cassette | INDEX |
| [TEK 4051] /Master Cassette | NC5 |
| [TEK 4051] /Master Cassette | NCTOLBL |
| [TEK 4051] /Master Cassette | USRSEND |

## PROJECT PERSONNEL

ETS/Electronics Engineering/Special Projects Department
- Charles Ostman - Project Staff x5571
- Art Turney - Project Coordinator x5571
- Allen Arthur - Special Projects Dept Head x5571

ETS/Computing Section/Computer Applications Department (CA)
- Vic Chinn - Project Staff x5354
- Brad Heckman - Project Coordinator x6068
- Ardith Kenney - CA Short Term Projects Coordinator x5867
- Ruth Hinkins - CA General Applications Coordinator x5402
- Leo Vardas - CA Head x5138

```
1000 REM : PROGRAM"INDEX"
1010 REM :
1020 REM : THIS PROGRAM SELECTS WHICH PROGRAM THE USER WISHES TO
1030 REM : ACCESS AND EXECUTE
1040 REM :
1050 REM : WRITTEN BY VICTOR K. CHINN
1060 REM : COMPUTER APPLICATIONS DEPARTMENT
1070 REM : LAWRENCE BERKELEY LABORATORIES
1080 REM : BLDG 50A  ROOM 6102
1090 REM
1100 PRINT "GGLJJNUMERICAL CONTROL DRILL PRESS [LBL] PROGRAM INDEXER"
1110 PRINT
1120 PRINT "SELECT NAME OF PROGRAM YOU WISH TO EXECUTE : "
1130 PRINT
1140 PRINT "[1] : NC5     : TO BEGIN NORMAL EXECUTION"
1150 PRINT "[2] : NCTOLBL : TO CONVERT 'SOURCE' TO 'OBJECT' FILE"
1160 PRINT "[3] : USRSEND : TO SEND 'OBJECT' FILE TO CDC6600B PUNCH"
1170 PRINT
1180 PRINT "ENTER SELECTION NUMBER :";
1190 INPUT A
1200 IF A=>1 AND A<=3 THEN 1220
1210 GO TO 1100
1220 FIND A+1
1230 OLD
1240 END
```

```
1000 REM THIS IS FILE 5 ************** TAPE PUNCH OUTPUT ***************
1010 REM : THIS PROGRAM WAS WRITTEN BY I SQUARED SOFTWARE CO.
1020 REM : AND WAS MODIFIED BY VICTOR K. CHINN TO BE COMPATIBLE WITH
1030 REM : LBL CODE.
1040 REM : COMPUTER APPLICATIONS DEPARTMENT
1050 REM : LAWRENCE BERKELEY LABORATORIES
1060 REM : BLDG 50A RM 6102
1070 REM :
1080 PRINT "GGLJJ"
1090 PRINT "NUMERICAL CONTROL DRILL PRESS [LBL] PROGRAM"
1100 PRINT
1110 PRINT
1120 PRINT "*NOTE: NC50 SORTING PROGRAM MUST BE RUN BEFORE"
1130 PRINT "       USING THIS PROGRAM"
1140 PRINT
1150 PRINT "     Place disk with sorted file in drive #1."
1160 PRI "Place a blank disk to contain the source file to be created "
1170 PRINT "into drive 2. Enter (return) when done ";
1180 INPUT E$
1190 UNIT 2
1200 CREATE "SRCDISK";0,0
1210 CREATE "SOURCE";1,1976
1220 GOSUB 2680
1230 PRINT "Do you want data searched for minimum hole"
1240 PRINT "coordinates (0=YES 1=NO)? ";
1250 INPUT B
1260 GO TO B<>0 AND B<>1 OF 1230
1270 GOSUB 1*(B=0) OF 1450
1280 GOSUB 1630
1290 GOSUB 1710
1300 GOSUB 1900
1310 GOSUB 2020
1320 PRINT "JJ_OUTPUT COMPLETE.GG SOURCE FILE CREATED ON DRIVE 2.GG"
1330 PRINT "NOW LINKING TO NCTOLBL CONVERSION PROGRAM"
```

```
1340 PRINT "ONE MOMENT PLEASE..."
1350 UNIT 1
1360 CALL "POS","0","9999",3,1
1370 CLOSE
1380 CMD "E",E$
1390 UNIT 2
1400 CLOSE
1410 CMD "E",E$
1420 FIND 2
1430 OLD
1440 END
1450 ON EOF (1) THEN 1570
1460 UNIT 1
1470 OPEN "HOLE";1,"",E$
1480 INPUT #1:X$,Y$
1490 IF K9=1 THEN 1620
1500 X$=SEG(X$,2,7)
1510 Y$=SEG(Y$,2,7)
1520 X=VAL(X$)
1530 Y=VAL(Y$)
1540 X5=X MIN X5
1550 Y5=Y MIN Y5
1560 GO TO 1480
1570 CLOSE
1580 PRINT "JThe minimum hole coordinates are:"
1590 PRINT "X COORDINATE = ";X5
1600 PRINT "Y COORDINATE = ";Y5
1610 K9=1
1620 RETURN
1630 PRINT "JEnter the offset to be subtracted from all X-coordinates"
1640 PRINT "to be entered onto the tape (not greater than ";X5;") ";
1650 INPUT X5
1660 PRINT "JEnter the offset to be subtracted from all Y-coordinates"
1670 PRINT "to be entered onto the tape (not greater than ";Y5;") ";
1680 INPUT Y5
```

```
1690 PAGE
1700 RETURN
1710 PRINT "Place disk with sorted file in drive #1."
1720 PRINT "ASCII code used in output to source file."
1730 B1=1
1740 PRINT "MODE SELECT : Word-Address (Variable Field) "
1750 B3=2
1760 GOSUB 2540
1770 GO TO 1820
1780 A$=REP("0101",83,4)
1790 PRINT "NOTE: RS-244 uses ODD parity and ""Word-Address"" output";
1800 PRINT " mode."
1810 B3=3
1820 I$=CHR(13*(B1=1))
1830 B$=CHR(55+33*(B1=1))
1840 C$=CHR(56+33*(B1=1))
1850 CALL "EXEC",I$,A$
1860 IF B1=2 THEN 1890
1870 CALL "EXEC",B$,A$
1880 CALL "EXEC",C$,A$
1890 RETURN
1900 UNIT 1
1910 OPEN "HOLE";3,"",V$
1920 INPUT #3:X$,Y$
1930 S$=SEG(X$,1,1)
1940 UNIT 2
1950 OPEN "SOURCE";4,"",V$
1960 PRINT #4:"LEADER"
1970 X$=SEG(X$,2,7)
1980 Y$=SEG(Y$,2,7)
1990 GOSUB 2180
2000 P$=S$
2010 RETURN
2020 ON EOF (3) THEN 2150
2030 GOSUB B3 OF 1900,2240,2240
```

```
2040 INPUT #3:X$,Y$
2050 IF K8=1 THEN 2170
2060 S$=SEG(X$,1,1)
2070 X$=SEG(X$,2,7)
2080 Y$=SEG(Y$,2,7)
2090 GOSUB 2180
2100 GO TO S$=P$ OF 2130
2110 P$=S$
2120 PRINT #4:"DRILLCHANGE"
2130 GOSUB B3 OF 1900,2240,2240
2140 GO TO 2040
2150 PRINT #4:"LEADEREND"
2160 K8=1
2170 RETURN
2180 REM ADJUST BY COORDINATE ORIGIN
2190 X6=VAL(X$)-X5
2200 X$=STR(X6)
2210 Y6=VAL(Y$)-Y5
2220 Y$=STR(Y6)
2230 RETURN
2240 REM ***** Variable Field Format (ASCII) **********************
2250 X=INT(VAL(X$)*10000+0.5)
2260 Y=INT(VAL(Y$)*10000+0.5)
2270 L$=STR(Y)
2280 L$=SEG(L$,2,LEN(L$)-1)
2290 IF LEN(L$)=6 THEN 2320
2300 L$="0"&L$
2310 GO TO 2290
2320 Q$=STR(X)
2330 Q$=SEG(Q$,2,LEN(Q$)-1)
2340 IF LEN(Q$)=6 THEN 2370
2350 Q$="0"&Q$
2360 GO TO 2340
2370 T$=SEG(L$,LEN(L$),1)
2380 IF T$<>"0" THEN 2410
```

```
2390 L$=SEG(L$,1,LEN(L$)-1)
2400 GO TO 2370
2410 IF LEN(L$)=0 THEN 2430
2420 CALL "EXEC",L$,A$
2430 T$=SEG(Q$,LEN(Q$),1)
2440 IF T$<>"0" THEN 2470
2450 Q$=SEG(Q$,1,LEN(Q$)-1)
2460 GO TO 2430
2470 IF LEN(Q$)=0 THEN 2490
2480 CALL "EXEC",Q$,A$
2490 PRINT #4:B$
2500 PRINT #4:Q$
2510 PRINT #4:C$
2520 PRINT #4:L$
2530 RETURN
2540 PRINT "PARITY SELECT : Off "
2550 PRINT
2560 PRINT "CREATING SOURCE FILE - THIS PROCESS MAY TAKE SOME TIME"
2570 PRINT
2580 P5=3
2590 GOSUB 2660
2600 RETURN
2610 RETURN
2620 A$=REP("6",84,1)
2630 RETURN
2640 A$=REP("0101",83,4)
2650 RETURN
2660 A$="39"
2670 RETURN
2680 REM***** DATA INITILIZATION ****************************************
2690 REM
2700 INIT
2710 SET DEGREES
2720 SET CASE
2730 FUZZ 4,1.0E-3
```

```
2740 DIM U$(9),T$(3),N$(4),S$(1),I$(1),A$(300),B$(1),G$(14),Q$(152)
2750 GOSUB 2860
2760 DIM D5(10),D6(5),D7(5),D8(10),D9(10),D$(100)
2770 DATA 1,0,0,0,0,0,0,0,0,0,0,1,1,1,1,0,0,0,99,99,0,0
2780 RESTORE 2770
2790 READ S9,S,W,W0,W1,X1,Y1,Z8,Z9,D5,X5,Y5,K9,K8
2800 D$=""
2810 E$=CHR(0)
2820 FOR I=1 TO 100
2830 D$=E$&D$
2840 NEXT I
2850 RETURN
2860 REM
2870 A$="30>>08:6079700:6089701>>0;;=::3>=?02"
2880 A$=A$&"4?<608=>020<690024074<850227014?"
2890 A$=A$&"0=5:26?269004="
2900 A$=A$&"27068680::00:70008=?02=>0009=?0026=739"
2910 RETURN
2920 REM
2930 G$=""
2940 FOR K=1 TO LEN(F$)
2950 K$=SEG(F$,K,1)
2960 K$=CHR(ASC(K$)-48+16*(POS("3569",K$,1)>0)+32*(K$="0"))
2970 G$=G$&K$
2980 NEXT K
2990 F$=G$
3000 RETURN
```

```
1000 REM : PROGRAM "NCTOLBL"
1010 REM :
1020 REM : THIS PROGRAM WILL CONVERT THE OUTPUT CREATED BY THE NC50
1030 REM : TAPE PUNCH PROGRAM TO AN LBL COMPATIBLE NC CODE.
1040 REM :
1050 REM : WRITTEN BY VICTOR K. CHINN
1060 REM : COMPUTER APPLICATIONS DEPARTMENT
1070 REM : LAWRENCE BERKELEY LABORATORIES
1080 REM : BLDG 50A  ROOM 6102
1090 REM :
1100 REM :
1110 REM : CLOSE ALL PREVIOUS FILES AND SET UP ENVIRONMENT SCREEN
1120 CLOSE
1130 PRINT @32,26:2
1140 PRINT "GGLJJNCTOLBL CONVERSION PROGRAMJJ"
1150 REM : GET NAMES OF FILES AND SET UP ACCESS TO THEM
1160 PRINT "PLACE THE SOURCE DISK CREATED FROM PROGRAM NC5 INTO"
1170 PRINT "DRIVE # 1.  THIS DISK SHOULD CURRENTLY BE IN DRIVE # 2"
1180 PRINT "PRESS RETURN WHEN READY ";
1190 INPUT C$
1200 UNIT 1
1210 OPEN "SOURCE";1,"",A$
1220 PRINT "PLACE THE DISK TO CONTAIN THE OBJECT CODE IN DRIVE 2"
1230 PRINT "PRESS RETURN WHEN READY :";
1240 INPUT B$
1250 UNIT 2
1260 PRINT "ENTER NAME OF OBJECT FILE : ";
1270 INPUT B$
1280 PRINT "INITIALIZE DISKETTE ON DRIVE 2 TO CONTAIN OBJECT CODE ?"
1290 PRINT "YOU MUST INITIALIZE DISKETTE IF IT IS NEW ";
1300 INPUT C$
1310 IF C$<>"Y" AND C$<>"N" THEN 1280
1320 IF C$="N" THEN 1350
1330 CREATE "OBJECT";0,0
```

```
1340 CREATE B$;1,1976
1350 OPEN B$;2,"",C$
1360 PRINT "CONVERTING SOURCE FILE TO OBJECT FILE"
1370 PRINT "NOTE : THIS PROCESS MAY  BE VERY TIME CONSUMING"
1380 PRINT "EACH '.' REPRESENTS ONE COORDINATE WRITTEN ONTO OBJECT FILE"
1390 REM : VARIABLE DEFINITIONS
1400 REM : X  ABS X COODDINATE       Y  ABS Y COORDINATE
1410 REM : T$ TAB                    R$ RWS
1420 REM : I$ '0' ZERO               Q$ '6'
1430 REM : E$ EOB                    L$ LEADER
1440 REM : V$ FEED TAPE              W$ REWIND CODE
1450 REM : Z$ BLANK
1460 X=0
1470 Y=0
1480 T$=CHR(62)
1490 R$=CHR(91)
1500 I$=CHR(32)
1510 Q$=CHR(22)
1520 E$="EOB"
1530 L$=CHR(0)
1540 H$="0123456789-"
1550 V$=CHR(127)
1560 W$=CHR(2)
1570 Z$=""
1580 REM : READ CHARS FROM SOURCE FILE AND CONVERT TO OBJECT FILE
1590 ON EOF (1) THEN 1830
1600 INPUT #1:D$
1610 REM : DRILLCHANGE CODE
1620 IF D$<>"DRILLCHANGE" THEN 1700
1630 PRINT #2:T$
1640 PRINT #2:T$
1650 PRINT #2:T$
1660 PRINT #2:I$
1670 PRINT #2:Q$
1680 PRINT #2:E$
```

```
1690 GO TO 1600
1700 IF D$="LEADER" THEN 1790
1710 IF D$<>"LEADEREND" THEN 2060
1720 REM : LEADEREND CODE
1730 PRINT #2:T$
1740 PRINT #2:T$
1750 PRINT #2:T$
1760 PRINT #2:I$
1770 PRINT #2:W$
1780 PRINT #2:E$
1790 FOR A=1 TO 50
1800 PRINT #2:L$
1810 NEXT A
1820 IF D$="LEADER" THEN 1980
1830 UNIT 1
1840 CALL "POS","0","9999",1,1
1850 CLOSE 1
1860 CMD "E",C$
1870 UNIT 2
1880 CALL "POS","0","9999",2,1
1890 CLOSE 2
1900 CMD "E",C$
1910 PRINT
1920 PRINT "GGGGJJCONVERSION TO FILE COMPLETED.  OBJECT FILE CREATED."
1930 PRINT "NOW LINKING TO 'USRSEND' TO SEND FILE."
1940 PRINT "ONE MOMENT PLEASE ..."
1950 FIND 3
1960 OLD
1970 END
1980 FOR A=1 TO 10
1990 PRINT #2:V$
2000 NEXT A
2010 PRINT #2:E$
2020 PRINT #2:R$
2030 PRINT #2:E$
```

```
2040 GO TO 1600
2050 REM : CODE READ FROM SOURCE IS A COORDINATE
2060 IF D$<>"X" AND D$<>"Y" THEN 2100
2070 PRINT #2:T$
2080 Z$=D$
2090 GO TO 1600
2100 G$=""
2110 G$=SEG(D$,2,LEN(D$)-1)
2120 IF LEN(G$)=4 THEN 2160
2130 FOR T=1 TO 5-LEN(D$)
2140 G$=G$&"0"
2150 NEXT T
2160 GOSUB 2530
2170 IF Z$<>"X" THEN 2210
2180 G=G-X
2190 X=X+G
2200 GO TO 2230
2210 G=G-Y
2220 Y=Y+G
2230 GOSUB 2720
2240 PRINT ".";
2250 IF G1<=-1000 OR G1=>1000 THEN 2290
2260 IF G1=>0 THEN 2290
2270 V$=SEG(G$,2,LEN(G$))
2280 G$="-"&V$
2290 IF G$="0" THEN 2490
2300 FOR B=1 TO LEN(G$)
2310 K$=SEG(G$,B,1)
2320 FOR C=1 TO 11
2330 J$=SEG(H$,C,1)
2340 IF K$=J$ THEN 2380
2350 NEXT C
2360 PRINT "ERROR...NO MATCH ERROR"
2370 STOP
2380 D=C-1
```

```
2390 IF D<>10 THEN 2420
2400 D=64
2410 GO TO 2460
2420 IF D<>0 THEN 2450
2430 D=32
2440 GO TO 2460
2450 D=D+16*(C=4)+16*(C=6)+16*(C=7)+16*(C=10)
2460 S$=CHR(D)
2470 PRINT #2:S$
2480 NEXT B
2490 IF Z$<>"Y" THEN 1600
2500 PRINT #2:E$
2510 GO TO 1600
2520 REM : SUBROUTINE TO CONVERT CODE READ IN TO VALUE IN VAR. G
2530 G=0
2540 FOR A=LEN(G$) TO 1 STEP -1
2550 K$=SEG(G$,A,1)
2560 FOR B=1 TO 11
2570 J$=SEG(H$,B,1)
2580 IF K$=J$ THEN 2620
2590 NEXT B
2600 PRINT "ERROR...NO MATCH ERROR"
2610 STOP
2620 IF B<>11 THEN 2660
2630 IF G<=0 THEN 2700
2640 G=G*-1
2650 GO TO 2700
2660 IF LEN(G$)-A<>0 THEN 2690
2670 G=G+(B-1)
2680 GO TO 2700
2690 G=G+(B-1)*10^(LEN(G$)-A)
2700 NEXT A
2710 RETURN
2720 REM SAVE G INTO VARIABLE G1
2730 G1=G
```

```
2740 M$=""
2750 G$=""
2760 IF G<>0 THEN 2790
2770 G$="0"
2780 RETURN
2790 IF G>0 THEN 2820
2800 G$=G$&"-"
2810 G=G*-1
2820 G=G/10
2830 K=G
2840 G=INT(G)
2850 IF K<1 THEN 2880
2860 K=K-1
2870 GO TO 2850
2880 K=K*10
2890 J$=SEG(H$,K+1,1)
2900 M$=M$&J$
2910 IF G=>1 THEN 2820
2920 FOR A=LEN(M$) TO 1 STEP -1
2930 J$=SEG(M$,A,1)
2940 G$=G$&J$
2950 NEXT A
2960 RETURN
2970 END
```

```
1000 REM : PROGRAM 'USRSEND'
1010 REM :
1020 REM : THIS PROGRAM WILL TRANSFER DATA FROM THE TEKTRONIX 4051
1030 REM : MICROCOMPUTER TO THE CDC 6000 COMPUTERS VIA THE 'ASDTP'
1040 REM : FILE TRANSFER PROTOCOL.
1050 REM :
1060 REM : WRITTEN BY VICTOR K. CHINN
1070 REM : COMPUTER APPLICATIONS DEPARTMENT
1080 REM : LAWRENCE BERKELEY LABORATORIES
1090 REM : BLDG 50A RM 6102
1100 REM :
1110 REM :
1120 REM : AN EXPLANATION OF THE 'ASDTP' FILE TRANSFER PROTOCOL IS
1130 REM : AVAILABLE FROM THE 'B' OR 'C' COMPUTER BY EXECUTING
1140 REM : THESE CONTROL CARDS :
1150 REM :
1160 REM : LIBCOPY,VKCHINN,DOC,ASDTPDC.
1170 REM : DISPOSE,DOC=PR,DT=1,PA=1F.
1180 REM :
1190 REM :
1200 REM : INITIALIZE
1210 REM : THIS COROUTINE INITIALIZES ALL VARIABLES TO BE USED IN
1220 REM : THE PROGRAM.  IT ALSO GETS THE FILE TO BE TRANSMITTED
1230 REM : AND 'OPENS' ACCESS TO IT.
1240 REM :
1250 REM : A$  = NAME OF FILE
1260 REM : B$  = EXTENSION IF ANY
1270 REM : C$  = RESPONSE FROM SERVER
1280 REM :
1290 REM :
1300 DIM A$(8),B$(8),C$(80),D$(80),E$(1),G$(2)
1310 DIM H$(16),I$(25),J$(6),K$(10),L$(10)
1320 H$="0123456789ABCDEF"
1330 J$=""
```

```
1340 M1=0
1350 REM : PROMPT USER
1360 PRINT "GGLJJUSRSEND SEND FILE TO PUNCH PROGRAMJJ"
1370 PRINT "*NOTE : THIS PROGRAM TAKES TIME IF IT IS RUN DURING"
1380 PRINT "          PEAK COMPUTING HOURS OF THE CDC 6600B"
1390 PRINT
1400 REM ENTER TERMINAL MODE IF REQUESTED
1410 PRINT "ENTER TERMINAL MODE ";
1420 INPUT L$
1430 IF L$="N" THEN 1520
1440 CALL "RATE",2400,4,2
1450 CALL "MARGIN",1,0,0
1460 PRINT @32,26:2
1470 PRINT "REMEMBER TO HIT THE BUTTON ON THE TSB-1A1 AND NOW"
1480 PRINT "LOGIN TO THE CDC 6400 B COMPUTER.  WHEN READY TO TRANSFER"
1490 PRINT "DATA HIT THE UDK-5 BUTTON TO RETURN TO THE MICRO 4051"
1500 PRINT "MODE AND THEN TYPE 'RUN' TO EXECUTE THIS PROGRAM AGAIN"
1510 CALL "TERMIN"
1520 PRINT "ENTER NAME OF FILE TO BE TRANSMITTED : ";
1530 INPUT A$
1540 PRINT "ENTER EXTENSION [IF NONE TYPE 'RETURN'] : ";
1550 INPUT B$
1560 PRINT "PLACE THE DISKETTE IN DRIVE NUMBER 1 THEN TYPE 'RETURN' : ";
1570 INPUT I$
1580 UNIT 1
1590 IF B$="" THEN 1620
1600 OPEN A$,B$;1,"PI",I$
1610 GO TO 1630
1620 OPEN A$;1,"PI",I$
1630 PRINT "DO YOU WISH TO MONITOR ALL DATA TRANSFERS ? [Y OR N] : ";
1640 INPUT L$
1650 IF L$<>"Y" AND L$<>"N" THEN 1630
1660 REM :
1670 REM :
1680 REM : STARTUP
```

```
1690 REM : THIS COROUTINE SENDS THE STARTUP CODE TO THE SERVER TO
1700 REM : INITIALIZE DATA FILE TRANSFER.  UPON TRANSMISSION IT WAITS
1710 REM : FOR THE CORRECT ACKNOWLEDGE RESPONSE BEFORE CONTINUING ELSE
1720 REM : IT ABORTS AND NOTIFIES THE USER.
1730 REM :
1740 REM :
1750 PRINT @40,30:
1760 PRINT @40:"$1040001FA"
1770 IF L$="N" THEN 1790
1780 PRINT "SENDING STARTUP CODE * $1040001FA *"
1790 INPUT @40:C$
1800 IF L$="N" THEN 1820
1810 PRINT "RESPONSE * ";C$;" *"
1820 I$=SEG(C$,3,10)
1830 IF I$="$0040004F7" THEN 2080
1840 PRINT "ERRORG.G.G.NO ACK CODE RECEIVED UPON TRANSMISSION OF START-"
1850 PRINT "UP CODE TO THE SERVER."
1860 PRINT "REPLY FROM SERVER WAS : ";C$
1870 STOP
1880 REM :
1890 REM :
1900 REM : CREATE LINE
1910 REM : THIS COROUTINE CREATES A LINE TO BE TRANSMITTED TO
1920 REM : THE SERVER.  LINES ARE CREATED BY THE FOLLOWING FORMAT :
1930 REM :
1940 REM :        $1BCMCMT----DATA----CS
1950 REM :
1960 REM : BC = BYTE COUNT    [STORED IN C1]
1970 REM : MC = MESSAGE COUNT [STORED IN M1]
1980 REM : MT = MESSAGE TYPE  [ALWAYS '05' FOR DATA]
1990 REM : CS = CHECK SUM     [STORED IN C2]
2000 REM :
2010 REM : D$ = LINE TO BE TRANSMITTED
2020 REM : E$ = CURRENT CHARACTER READ FROM FILE
2030 REM : T  = TEMPORARY VARIABLE
```

```
2040 REM : J$ = END OF FILE SET MARKER
2050 REM : G  = LINKAGE VARIABLE SENT TO THE SUBROUTINE TO BE CALLED
2060 REM :
2070 REM :
2080 C1=4
2090 C2=0
2100 D$="$1XXXXXX"
2110 ON EOF (1) THEN 2240
2120 INPUT #1:E$
2130 C1=C1+1
2140 IF E$<>"E" THEN 2180
2150 G$="80"
2160 G=128
2170 GO TO 2200
2180 G=ASC(E$)
2190 GOSUB 3030
2200 D$=D$&G$
2210 C2=C2+G
2220 IF C1<>20 THEN 2110
2230 GO TO 2250
2240 J$="EOFSET"
2250 M1=M1+1
2260 IF M1<=15 THEN 2280
2270 M1=0
2280 C2=C2+C1+M1+5
2290 T=INT(C2/256)
2300 C2=C2-T*256
2310 C2=255-C2
2320 D$=REP("05",7,2)
2330 G=C1
2340 GOSUB 3030
2350 D$=REP(G$,3,2)
2360 G=C2
2370 GOSUB 3030
2380 D$=D$&G$
```

```
2390 G=M1
2400 GOSUB 3030
2410 D$=REP(G$,5,2)
2420 REM :
2430 REM :
2440 REM : SEND LINE
2450 REM : THIS COROUTINE SENDS A LINE TO THE SERVER.  AFTER IT IS
2460 REM : SENT IT WAITS FOR AN 'ACK' CODE FROM THE SERVER BEFORE
2470 REM : CONTINUING.  IF IT RECEIVES AN 'NAK' [NO ACKNOWLEDGE]
2480 REM : CODE, THE LINE IS TRANSMITTED AGAIN AND THE PROCESS
2490 REM : IS REPEATED.
2500 REM :
2510 REM :
2520 PRINT @40,30:
2530 PRINT @40:D$
2540 IF L$="N" THEN 2560
2550 PRINT "SENDING   * ";D$;" *"
2560 INPUT @40:C$
2570 IF L$="N" THEN 2590
2580 PRINT "RESPONSE  * ";C$;" *"
2590 K$=SEG(C$,9,2)
2600 IF K$="07" THEN 2520
2610 IF J$<>"EOFSET" THEN 2080
2620 REM :
2630 REM :
2640 REM : END OF FILE SET
2650 REM : THIS COROUTINE RECGONIZES THE END OF FILE AND SENDS AN
2660 REM : END OF TRANSMISSION CODE TO THE SERVER.  AFTER IT IS SENT
2670 REM : AN ACKNOWLEDGEMENT CODE IS EXPECTED AND THEN THE PROGRAM
2680 REM : TERMINATES SUCCESSFULLY BY LEAVING THE USER IN TERMINAL
2690 REM : MODE COMMUNICATING TO THE 6000 MACHINES.  ELSE IF AN ERROR
2700 REM : OCCURS, THE PROGRAM ABORTS AND NOTIFIES THE USER.
2710 REM :
2720 REM :
2730 D$="$104XX02"
```

```
2740 G=M1
2750 GOSUB 3030
2760 D$=REP(G$,5,2)
2770 C2=255-(4+M1+2)
2780 G=C2
2790 GOSUB 3030
2800 D$=D$&G$
2810 PRINT @40,30:
2820 PRINT @40:D$
2830 IF L$="N" THEN 2850
2840 PRINT "SENDING   * ";D$;" *"
2850 INPUT @40:C$
2860 IF L$="N" THEN 2880
2870 PRINT "RESPONSE * ";C$;" *"
2880 K$=SEG(C$,9,2)
2890 IF K$="03" THEN 2920
2900 PRINT "ERRORG.G.G.EOT NOT ACKNOWLEDGED"
2910 STOP
2920 PRINT "EOT ACKNOWLEDGED...COMPLETED DATA TRANSFER"
2930 PRINT "ENTERING TERMINAL MODE : "
2940 CALL "TERMIN"
2950 END
2960 REM :
2970 REM :
2980 REM : SUBROUTINE CONVERSION
2990 REM : THIS SUBROUTINE CONVERTS THE LINKAGE VARIABLE G INTO A
3000 REM : HEX VALUE WHICH IS RETURNED IN VARIABLE G$
3010 REM :
3020 REM :
3030 G$=""
3040 Q=INT(G/16)
3050 R=G-Q*16
3060 K$=SEG(H$,Q+1,1)
3070 G$=G$&K$
3080 K$=SEG(H$,R+1,1)
```

```
3090 G$=G$&K$
3100 RETURN
```

```
 XXX    XXX   XXXX   XXXXX  XXXX
X   X  X   X  X   X    X    X   X
X   X  X      X   X    X    X   X
XXXXX   XXX   X   X    X    XXXX
X   X      X  X   X    X    X
X   X  X   X  X   X    X    X
X   X   XXX   XXXX     X    X
```

# A SIMPLE DATA TRANSFER PROTOCOL FOR THE LBL INTERACTIVE SYSTEM

20 DECEMBER 1978

TO OBTAIN A COPY OF THIS WRITEUP

```
LIBCOPY,AGUERCT,DOC,DASDTP.
DISPOSE,DOC=PR,PA=1F,DT=I.
```

TABLE OF CONTENTS 10/11/77

## ABSTRACT

THIS DOCUMENT DESCRIBES A NEW PROTOCOL WHICH IS BEING IMPLEMENTED AT LBL FOR USE BY USERS WISHING TO SEND AND/OR RECEIVE FILES THRU THE LBL INTERACTIVE SYSTEM ON INTERACTIVE TERMINAL LINES (RECC/MICROBUS).

IN ORDER TO KEEP THE FIELD LENGTH SMALL, THE PROTOCOL IS SIMPLE AND ONLY A FEW OPTIONS ARE AVAILABLE.
THE PROTOCOL SUPPORTS INPUT FROM THE USER PROCESS WITH THE DATA WRITTEN TO A DISK FILE IN EITHER ASCII OR DISPLAY CODE FORMAT
OUTOUT FROM THE SERVER PROCESS WITH THE DATA READ TO A DISK FILE IN EITHER ASCII OR DISPLAY CODE FORMAT

## 1.1 PROGRAM

### DESCRIPTION OF USE

THE USER WILL BE ABLE TO TRANSFER FILES FROM INTELLIGENT TERMINALS ON **RECC** OR **MICROBUS** LINES BY IMPLEMENTING **ASDTP**.

**ASDTP** IS IMPLEMENTED ON THE **BKY 6000** SYSTEM AS A MAIN PROGRAM RUNNING UNDER THE USERS INTERACTIVE JOB. THE PROGRAM CAN BE INVOKED AS A SIMPLE CONTROL CARD STATEMENT AS ALL BKY CONTROL CARDS ARE.

THE USERS ASDTP PROCESS IS IMPLEMENTED IN AN INTELLIGENT TERMINAL ON THE **TAPETTY** TERMINAL I/O LINK CONNECTED TO THE **RECC** OR **MICROBUS** SYSTEM. IT MAY BE INVOKED IN ANY MANNER THE USER WISHES.

FOR PURPOSES OF THIS DISCUSSION, THE PROGRAM WHICH RUNS IN THE 6000 **(ASDTP)**, WILL BE REFERED TO AS THE **SERVER PROCESS**. THE PROGRAM RUNNING IN THE USERS MACHINE WILL BE REFERED TO AS THE **USER PROCESS**.

### WHERE TO FIND ASDTP

THE ASDTP PROGRAM CAN BE USED BY EXECUTING THE FOLLOWING CONTROL CARDS.

```
LIBCOPY,AGUEROBJ,ASDTP/RR,ASDTP.
ASDTP.
```

## CALLING FORMAT

ASDTP CAN BE CALLED WITH ARGUMENTS.

```
ASDTP,F=<FILENAME>,LO=IODARB.
```

THIS WILL ENABLE THE SPECIFICATION OF PARAMETERS WITHOUT HAVING TO WRITE IT INTO THE USER PROCESS.

**FILENAME** THE LOCAL FILE ON WHICH THE DATA RESIDES IN THE SERVER PROCESS. DEFAULT=INFILE.

**LO=** LOAD OPTIONS
THE OPTIONS ARE SELECTED BY CONCATINATING THE FOLLOWING.

**A** ASCII MODE. SELECTION OF THIS MODE WILL CAUSE THE DATA TO BE INTERPRETED ON THE DISK AS ASCII IS, 5 CHARACTERS PER 60 BIT WORD, 8 BITS PER CHARACTER RIGHT JUSTIFIED IN A 12 BIT FIELD.

**D** DISPLAY CODE MODE. THIS MEANS THAT DATA ON THE DISK WILL BE INTERPRETED AS DISPLAY CODE IS, EACH 60 BIT WORD WILL CONTAIN 10 CHARACTERS OF 6 BITS EACH. THIS IS THE DEFAULT.

**I** INPUT MODE. THIS MEANS THAT THE PROGRAM WILL EXPECT TO RECEIVE DATA FROM THE LINK AND TO STORE IT ON A DISK FILE.

**O** THIS OPTION CAUSES THE DATA TO BE TAKEN FROM DISK AND SENT ON THE LINK TO THE USER PROCESS.

**R** THIS OPTION CAUSES THE DESIGNATED FILE TO BE REWOUND BEFORE USE.

**B** THIS WILL PERMIT READING OR WRITING OF THE DISK WITHOUT A REWIND. THIS IS THE DEFAULT.

AS CAN BE SEEN THESE OPTIONS CAN BE DIVIDED INTO THREE GROUPS. THE MEMBERS OF THE GROUPS ARE MUTUALLY EXCLUSIVE AND USING THEM TOGETHER SERVES NO FUNCTION EXCEPT TO CONFUSE THE USER. IF BOTH MEMBERS OF A GROUP ARE INCLUDED IN THE CALL TO THE PROGRAM, THE LAST ONE LISTED WILL TAKE PRECIDENCE. EXAMPLE - ASDTP,LO=RB. THERE WOULD BE NO REWIND OF THE FILE.

## GENERAL FORMAT

THE MESSAGE FORMAT IS TAKEN FROM THE LOADER FORMAT OF THE POPULAR MICROPROCESSORS. THE MESSAGE CONSISTS ENTIRELY OF ASCII CHARACTERS.

BECAUSE OF THE POSSIBILITY OF PLACING PART OF THE PROGRAM INTO **MICROBUS** WHEN IT ARRIVES, THE PROTOCOL WILL HAVE A TWO LAYER NATURE. THESE WILL BE REFERRED TO AS THE **LINK LEVEL** PORTION AND THE **HOST LEVEL** PORTION.

THE LINK LEVEL PROTOCOL REFERS TO THAT PORTION OF THE MESSAGE WHICH CONTAINS INFORMATION ABOUT FLOW CONTROL AND ERROR CORRECTION. THE HOST LEVEL PROTOCOL REFERS TO THE DATA AND THE STATUS MESSAGES WHICH FLOW BETWEEN THE TWO PROCESSES.

WHEN THIS PROTOCOL IS IMPLEMENTED IN A MICROPROCESSOR THE TWO LEVELS MAY EXIST TOGETHER IN THE SAME CODE, BUT THEY DONT HAVE TO.

THE MESSAGE FORMAT IS AS FOLLOWS.

**$NBCMCMT----DATA----CS**

| | | |
|---|---|---|
| **$** | **DOLLAR SIGN** | MUST BE THE FIRST CHARACTER IN THE MESSAGE. |
| N | **MESSAGE SOURCE** | CAN BE ZERO OR ONE. ZERO IS USED FOR ALL SERVER MESSAGES AND ONE IS USED FOR ALL USER MESSAGES. |
| BC | **BYTE COUNT** | CONTAINS THE NUMBER OF CHARACTERS IN THE MESSAGE. THE BYTE COUNT DOES NOT INCLUDE THE MESSAGE SOURCE CHARACTERS. |
| **MC** | **MESSAGE COUNT** | THIS BYTE CONTAINS TWO ROTATING COUNTS, MUDULO 16, ONE FOR EACH DIRECTION ON THE LINK. |
| **MT** | **MESSAGE TYPE** | CODE FOR THE TYPE OF MESSAGE THAT IS BEING TRANSMITTED |
| **DATA** | **DATA FIELD** | MAY BE NULL. (SEE FORMAT BELOW) |
| **CS** | **CHECKSUM** | A LINEAR SUM OF ALL FIELDS AFTER $N BEFORE CONVERSION, THEN COMPLIMENTED. |

## DATA REPRESENTATION

EACH ALPHANUMERIC CHARACTER FOLLOWING THE DOLLAR SIGN REPRESENTS FOUR BINARY BITS. THEREFORE TWO CHARACTERS REPRESENT AN 8 BIT BYTE. EACH CHARACTER PAIR IS FORMED BY MASKING THE APPROPRIATE BITS AND FORMING THE ASCII CHARACTER (1-9, A-F) FOR WHICH IT STANDS.

WHENEVER THE SERVER PROCESS FINDS A CARRIAGE RETURN/LINE FEED CHARACTER STRING IMBEDDED IN THE DATA THE SERVER WILL SUBSTITUTE ZEROS AND WILL GUARANTEE AN END OF LINE CONDITION. A SINGLE CARRIAGE RETURN OR LINE FEED WILL NOT BE SO TREATED.

## MESSAGE SOURCE

THE MESSAGE SOURCE IS SPECIFIED BY THE NUMBER FOLLOWING THE $ AND ARE DEFINED AS FOLLOWS.

| | | |
|---|---|---|
| 0 | **SERVER MESSAGE** | THIS CODE IS THE ONLY MESSAGE TYPE THAT IS LEGAL FOR THE SERVER PROCESS. |
| 1 | **USER MESSAGE** | |

NO OTHER VALUE IS VALID.

## 2.1 MESSAGE FIELDS

### BYTE COUNT

THE BYTE COUNT BYTE IS THE FIRST BYTE AFTER THE MESSAGE TYPE AND INDICATES THE NUMBER OF BYTES IN THE MESSAGE. THIS COUNT DOES NOT INCLUDE THE MESSAGE SOURCE BYTES BUT DOES INCLUDE THE CHECKSUM.

### MESSAGE COUNT BYTE

THE MESSAGE COUNT BYTE FOLLOWS THE BYTE COUNT AND IS DIVIDED INTO TWO FIELDS. THE UPPER 4 BITS CONTAIN THE CURRENT MESSAGE COUNT FOR THE SERVER PROCESS. THE LOWER 4 BITS CONTAIN THE CURRENT MESSAGE COUNTT FOR THE USER PROCESS.

THE COUNTS ARE INCREMENTED BY THE RESPECTIVE PROCESS WHENEVER THERE IS A NEW DATA MESSAGE FOR THAT PROCESS. THE MESSAGE COUNT IS NOT INCREMENTED ON RETRANSMISSIONS.

## CHECKSUM BYTE

THE CHECKSUM BYTE IS THE LAST CHARACTER OF THE MESSAGE, AND IS FORMED BY ADDING ALL THE CHARACTERS IN THE MESSAGE WITH THE EXCEPTION OF THE $ AND THE MESSAGE TYPE N. THE RESULT OF THIS ADDITION IS COMPLIMENTED AND USED AS THE CHECKSUM.

WHILE THE CHECKSUM IS SEPERATED FROM THE REST OF THE LINK CONTROL BYTES, IT SHOULD BE CONSIDERED AS PART OF THE LINK CONTROL.

## MESSAGE TYPE CODES

THIS FIELD IS USED TO DESIGNATE THE TYPE OF MESSAGE THAT IS BEING SENT. THE FOLLOWING IS THE LIST OF CODES WITH THE ACTION TAKEN BY EACH.

| | | |
|---|---|---|
| 0 | **ILLEGAL CODE** | |
| 1 | **STARTUP CODE** | THIS IS THE FIRST MESSAGE SENT BY USER PROCESS. WHEN THE SERVER PROCESS STARTS, IT WAITS FOR THIS CODE FROM THE USER. |
| 2 | **STOP CODE** | USED TO STOP THE SESSION. |
| 3 | **STOPREC CODE** | AN ACKNOWLEDGE CODE FOR THE STOP CODE. THIS IS THE LAST MESSAGE SENT. |
| 4 | **POLL CODE** | USED WHENEVER EITHER PROCESS NEEDS TO SEND OR REPLY BUT DATA CANNOT BE SENT. |
| 5 | **DATA CODE** | THE MESSAGE TYPE USED BY THE SENDING PROCESS WHENEVER IT HAS DATA TO SEND AND HAS PERMISSION FROM THE RECEIVING PROCESS TO DO SO. WHEN DATA IS SENT FROM THE USER, THIS MESSAGE MUST CONTAIN A MULTIPLE OF 10 CHARACTERS (BEFORE CONVERSION) OR THE LAST WORD WILL BE FILLED OUT WITH ZEROS CAUSING AN END OF LINE. |
| 6 | **EOR CODE** | THIS IS THE LAST MESSAGE TO BE SENT WHEN A TAPE MARK OR EOR IS FOUND. NO DATA IS SENT IN THIS MESSAGE BUT THE EFFECT IS TO FLUSH THE BUFFER AND WRITE AN EOR IN THE 6000, OR TO WRITE A TAPE MARK IF THE DATA COMES FROM THE 6000. |
| 7 | **NAK CODE** | INFORMS THE OTHER PROCESS THAT AN ERROR WAS DETECTED IN THE MESSAGE JUST SENT. THE EXPECTED RESULT IS THAT THE PREVIOUS MESSAGE SHOULD BE RESENT. |
| 8 | **ABORT CODE** | THIS MESSAGE IS USED TO INFORM THE OTHER PROCESS THAT THE FIRST |

| | | |
|---|---|---|
| | | PROCESS HAS GIVEN UP IN AN UNNATURAL WAY AND IS ENDING THE DIALOG. |
| 9 | **BUSY CODE** | THIS MESSAGE IS SENT WHENEVER THE RECEIVING PROCESS IS BUSY AND CANNOT RECEIVE DATA. DATA SHOULD NOT BE SENT ON THE LINK UNTIL THE BUSY PROCESS IS NOT BUSY. |
| 15 | **DATA+EBBR** | SAME AS 5 AN END OF RECORD CODED HAS BEEN FOUND<br>THE ZERO ARE NOT TRANSFERRED |

## DATA FIELD

DATA IS SENT AS TWO ASCII CHARACTERS (0-9,A-F), REPRESENTATING 4 BITS EACH. THE CHECKSUM IS COMPUTED BEFORE CONVERSION TO ASCII.

FOR PURPOSES OF THIS DISCUSSION CONSIDER THAT ANY DATA TRANSFER SESSION CONSISTS OF THREE PARTS.

1. STARTUP
2. DATA TRANSFER
3. STOP

THE STARTUP PORTION DEALS WITH HOW THE TWO PROCESSES BECOME AWARE OF EACH OTHER AND HOW THEY LOGICALLY MOVE ON TO THE DATA TRANSFER SECTION.

THE SERVER PROCESS, WHEN STARTED, WILL WAIT FOR THE USER PROCESS TO SEND A STARTUP MESSAGE. WHEN THIS MESSAGE IS RECEIVED, THE SERVER PROCESS WILL RETURN A STARTUP MESSAGE TO INDICATE THAT IT IS READY. THIS COMPLETES THE INITIALIZATION DIALOG. THE USER PROCESS MAY THEN SEND EITHER DATA OF STATUS WHICH BEGINS THE DATA TRANSFER PORTION OF THE DIALOG.

THE DATA TRANSFER PORTION START AS SOON AS THE INITIAL STARTUP DIALOG IS COMPLETE AND ENDS WHIN EITHER PROCESS SENDS A STOP MESSAGE.

THE STOP PORTION IS JUST A MEANS OF PROVIDING AN ORDERLY ENDING TO A SESSION. WHEN EITHER PROCESS SENDS A STOP MESSAGE, THE OTHER IS OBLIGED TO DO LIKEWISE. ONCE THE STOP MESSAGE HAS MADE THE ROUNDTRIP, THE SESSION IS DONE AND BOTH PROCESSES TERMINATE.

```
OUTPUT
1- THE USER PROCESS SHOULD FIRST SEND A START UP MESSAGE
                    $1040001FA
2- THE SERVER PROCESS SEND EITHER DATA OR STATUS
                    IF DATA THE MESSAGE TYPE WILL BE EITHER 5 OR 15
                    $00E9005373830204724302224317E  DATA
                    $00DA00F30343030303646352227C   DATA+EOR
                    IF STATUS THE MESSAGE TYPE WILL BE EITHER 7  CKSM ERROR
                                                          OR 8  ABORT
                                                          OR 3  STOP
3- THE USER PROCESS ANSWERS SENDING STATUS
                    THE MESSAGE TYPE WILL BE EITHER 4  OK
                                                    2  STOP
                                                    7  CHECKSUM ERROR
                                                    8  ABORT
   THE OTHERS MESSAGES TYPE ARE NOT ALLOWED
4- THE DATA TRANSFER PORTION STOPS
                     EITHER WHEN THE USER PROCESS MESSAGE TYPE=2
                     OR     WHEN THE SERVER PROCESS MESSAGE TYPE=3
```