HUGHES

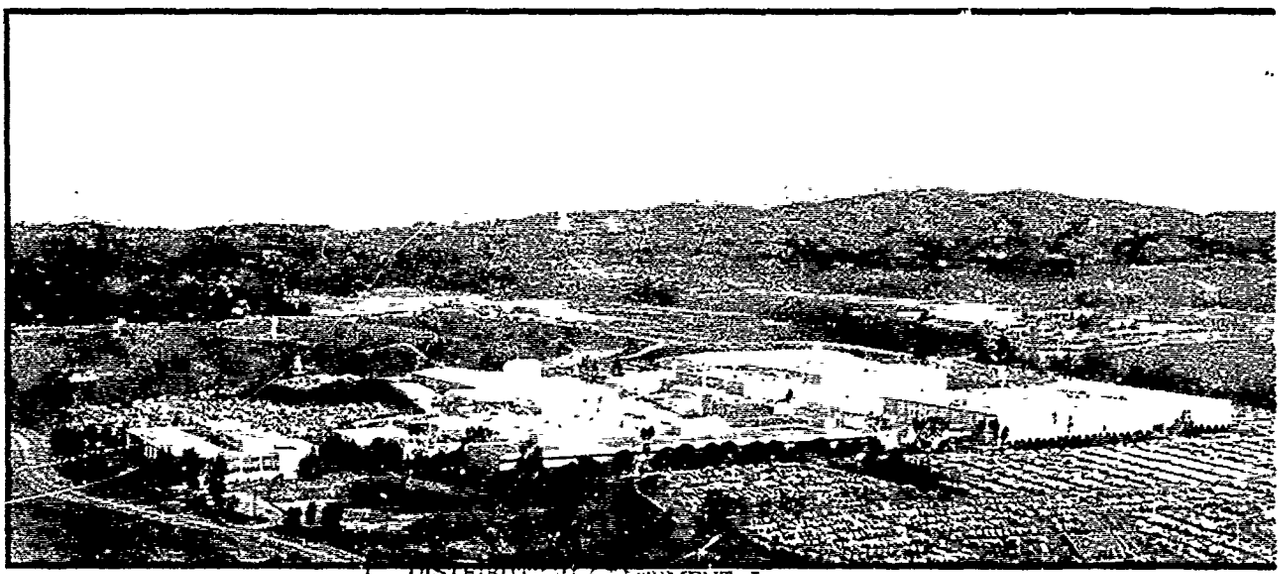
HUGHES AIRCRAFT COMPANY
GROUND SYSTEMS GROUP

Final Report
Appendix

Manufacturing Methods And Technology For Digital Fault Isolation Of Printed Circuit Boards

Project No. R783242

15 NOVEMBER 1980
CONTRACT NO. DAAK 40-78-C-0290



(9) FINAL REPORT. APPENDIX

(6) Manufacturing Methods and Technology for
Digital Fault Isolation of Printed Circuit Boards. Appendix I
Project No. R783242

Prepared for
U.S. Army Missile Command
Redstone Arsenal, Alabama 35809

Project Officer: G.D. Little
DRSMI-ETE
(205)-876-3848

(15) Contract No. DAAK 40-78-C-0290
CDRL A004

Prepared by

(12) 100

Hughes Aircraft Company
Ground Systems Group
Fullerton, California 92634

(11)
(14) Report Date: 15 November 1980
HAC - FR-80-12-975-APP

172370

RE: Distribution Statement
Basic report is distribution unlimited. Use same statement for appendix to basic per Mr. G. D. Little, Project Officer, DRSMI/ETE

FOREWORD

This report presents the final recommendations and conclusions, with supporting data, resulting from the contract option phase of contract DAAK 40-78-C-0290. It describes the manufacturing technology and test system that will enable detection, identification, and location of digital faults in the advanced missile electronic systems that will be used in the 1980's. Emphasis is placed on the fault diagnosis of large printed circuit boards containing complex hybrid digital microelectronic circuits. The Hughes-enhanced, state-of-the-art, DTS-70 automatic test system installed at Redstone Arsenal as a result of this contract provides the capability to isolate digital faults in such circuit boards to the component level with a test comprehensiveness of 90% or better.

The contract option phase of this project involved the purchase and installation of the DTS-70 system, the selection of the PN-1635972 and the PN-1646178 D/PCBs for testing, the development of generalized test software and the development of the specific hardware and software needed to test these worst-case boards. It also included a successful demonstration of the project's results for interested Department of Defense and industry personnel.

The report concludes with recommendations for the improvement of the DTS-70 System to increase its utility, with recommendations for improving the testability of digital printed circuit boards, and with recommendations for future digital fault isolation studies.

APPENDICES

CONTENTS

APPENDIX A – SOFTWARE

SECTION A.1

SIGNATURE ANALYSIS SOFTWARE

APPENDIX A – Software
Section A.1 – Signature Analysis Software

A.1.1 SIGNAL TRANSFER FILE TESTING

```
:*
:*      RUN THE INITIALIZING PROGRAM...
:*
: RU, INIT
:*
:*      ASSIGN THE 5004A S.A. AN LU IN SYSTEM...
:*
: SL,35,117
:*
:*      RUN SIGNATURE ANALYZER PROGRAM...
:*
: RU, SCAMPR
:*
:*      EXIT
: TR
```

APPENDIX A – Software
Section A.1 – Signature Analysis Software

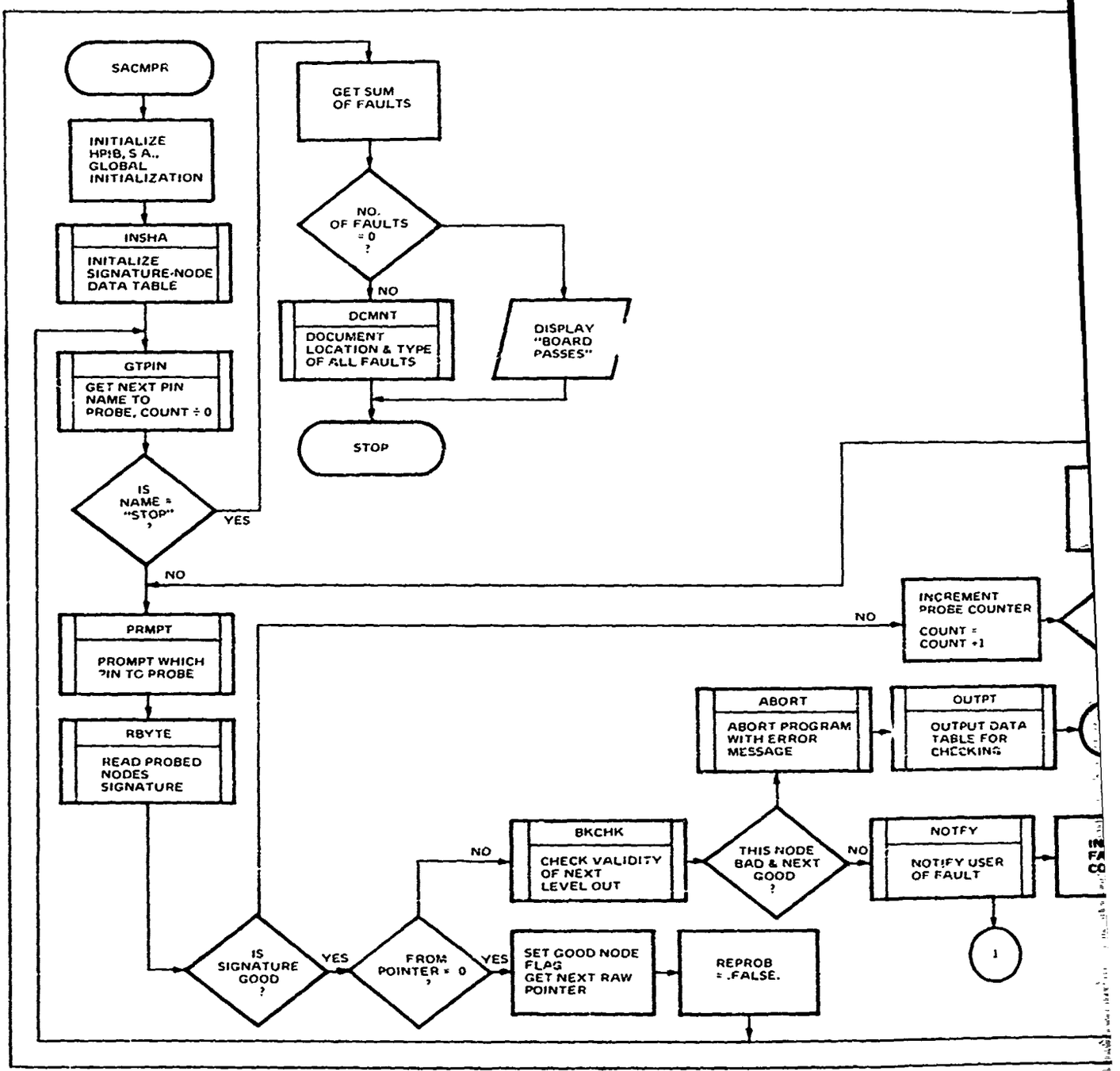
A.1.2 SAFILS SAMPLE LISTING

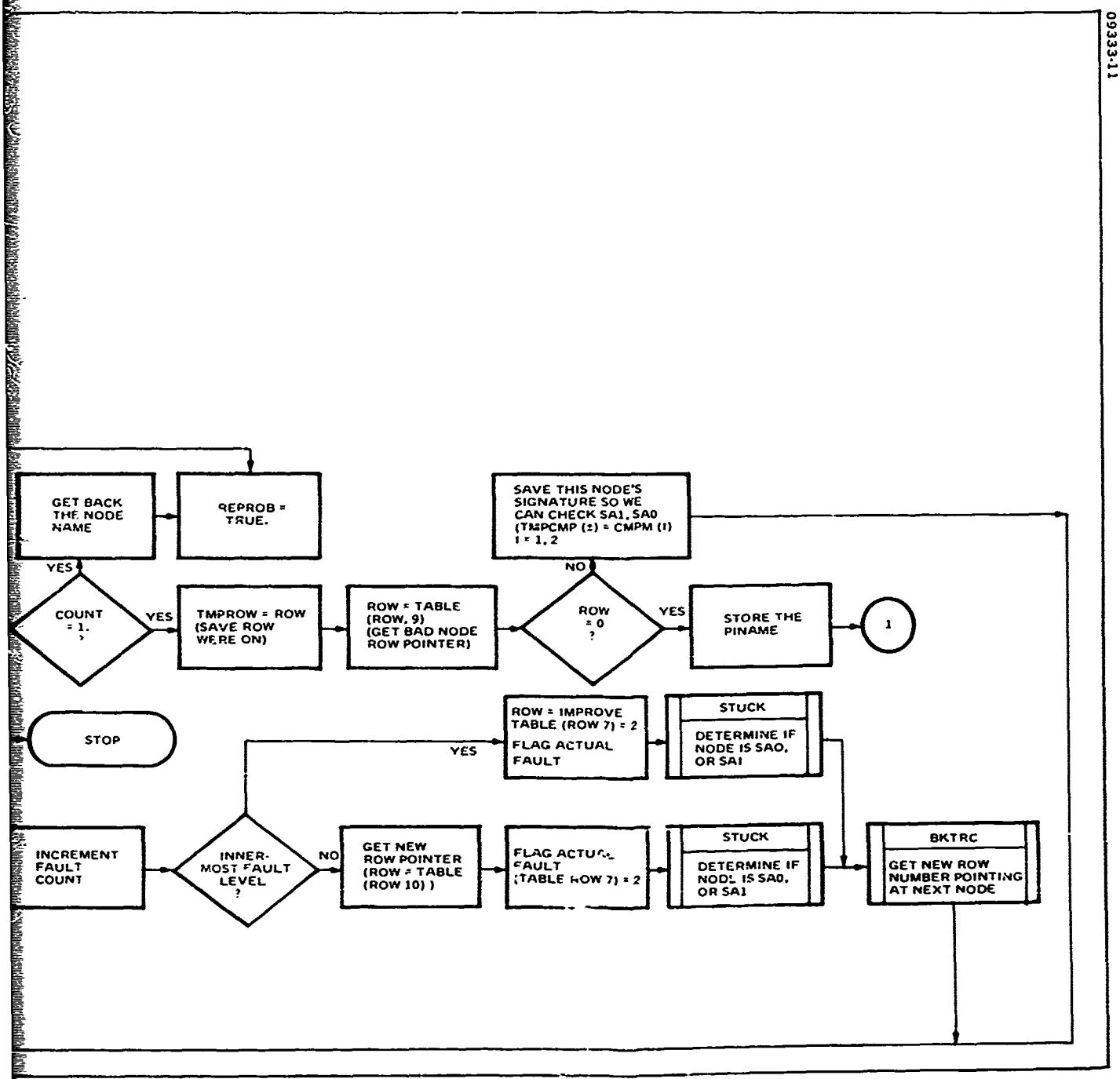
FILE NAME (COL 1-6)	CR (COL 7-8)	DESCRIPTION OF CONTENTS (COL 9-58)
6 chars	2 chars	50 chars
SIG972	19	SIGNATURE ANALYSIS OF ALL ADDRESS&DATA 927 BOARD
CPU972	19	SIGNATURE ANALYSIS OF 8080A CPU CHIP 972 BOARD
END OF FILE		

NOTE: The message "END OF FILE" must be placed in the data file starting in the 5th column to show the actual limit of the size of the file.

APPENDIX A – Software
Section A.1 – Signature Analysis Software

A.1.3 – SACMPR Flow Chart




09333-11
GET BACK THE NODE NAME
REPROB = TRUE.
SAVE THIS NODE'S SIGNATURE SO WE CAN CHECK SA1, SA0 (TMPCMP (I) = CMPM (I) I = 1, 2
YES
NO
COUNT = 1. ?
YES
TMPROW = ROW (SAVE ROW WE RE ON)
ROW = TABLE (ROW, 9) (GET BAD NODE ROW POINTER)
ROW = 0 ?
YES
STORE THE PINAME
1
STOP
ROW = IMPROVE TABLE (ROW 7) = 2
FLAG ACTUAL FAULT
STUCK
DETERMINE IF NODE IS SA0, OR SA1
YES
INCREMENT FAULT COUNT
INNER-MOST FAULT LEVEL ?
NO
GET NEW ROW POINTER (ROW = TABLE (ROW 10))
FLAG ACTUAL FAULT (TABLE ROW 7) = 2
STUCK
DETERMINE IF NODE IS SA0, OR SA1
BKTRC
GET NEW ROW NUMBER POINTING AT NEXT NODE

2

A.1.4 – SACMPR LISTINGS

SACMPR T=00004 IS ON CR00018 USING 00148 BLKS R=0000

```
FTN4,L
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
C        PROGRAM:   SACMPR
C
C        PURPOSE:   TO DIRECT THE TEST OPERATOR IN THE SIGNATURE ANALYSIS
C                   OF A DIGITAL PCB.
C
C        PROGRAMMER: DAVID S. WAGNER    -    BLDG. 688/T-125
C                    7030 E POTAWATAMI       HUGHES/(213) 802-4190
C                    TUCSON, AZ 85715        FULLERTON, CA 92634
C                    602-296-2760
C
C        DATE:      02 - JUL - 80
C
C        DATA FILE(S):  SAFILS::-18  contains list of available file
C                       names and description of contents
C                       User is queried to enter a data file name from
C                       this list
C        SUBROUTINES USED:
C
C         1. INSHA          - TO INITIALIZE TABLE FOR PROGRAM USE
C         2. GTPIN          - TO GET NEXT PIN TO PBOBE FROM TABLE
C         3. PRMPT          - TO PROMTP TEST OPERATOR TO PROBE PIN
C         4. RBYTE          - TO GET SIGNATURE OF PIN PROBED
C         5. CHECK          - TO COMPARE SIGNATURE WITH CORRECT ONE
C         6. BKCHK          - CHECKS VALIDITY OF PREVIOUS NODE
C         7. NOTFY          - NOTIFY THE OPERATOR OF A FAULT AND LOCATION
C         8. BKTRC          - BACKTRACE TO LAST GOOD PIN LOCATION
C         9. ABORT          - ERROR EXIT FROM THE PROGRAM
C        10. CNVRT          - ROUTINE TO CONVERT PACKED STRING TO UNPACKED
C        11. RCVRT          - ROUTINE TO CONVERT UNPACKED STRING TO PACKED
C        12. NUM            - FUNCTION TO CONVERT HOLLERITH TO INTEGER
C        13. UNPAK          - ROUTINE TO UNPACK AN A2 INTO 2 A1'S
C        14. ZEROR          - ROUTINE TO BLANK OUT ARRAYS
C        15. LENTH          - FUNCTION TO FIND NO. OF CHARACTERS IN STRING
C        16. PACK           - ROUTINE TO PACK 2 A1'S INTO AN A2
C        17. DCMNT          - TO DOCUMENT TEST RESULTS
C        18. STUCK          - TO CHECK IF SA0 OR SA1 FAULT
C
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
C     HIERARCHY OF SUBROUTINES AND FUNCTIONS USED IN SACMPR
C
C
C
C     MAIN
C
C           INSHA
C                 OPEN
C                 READF
C                 CNVRT
C                       NUM
C                       UNPAK
C           GTPIN
```

```
C              PRMPT
C              RBYTE
C              CHECK
C                     RCVRT
C                            LENTH
C                            FACK
C              B'CHK
C              NOTFY
C              STUCK
C              BKTRC
C              ABORT
C              DCMNT
C
C
C
C   FAULT FLAGGING CONVENTION USED IN DATA TABLE
C
C
C   0 -- NO FAULT
C   1 -- A NODE FAULT (NOT NECESSARILY BACKTRACED)
C   2 -- ACTUAL, BACKTRACED FAULT LOCATION
C   3 -- A STUCK AT 0 FAULT (SA0)
C   4 -- A STUCK AT 1 FAULT (SA1)
C
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
C
C   ARRAYS USED IN THIS PROGRAM:
C
C   ARRAY (200, 10)      Array used as temporary hold for blanking purposes
C   CHARS (2)            Array used in unpacking of input data
C   CMPAR (2)            Used in COMPARison of read signal and valid signal
C   DUMMY (3)            Array used to hold node name currently under test.
C   FILCHC(35)           Array used to hold data file names and descriotions
C   GND   (2)            Holds the Ground Characteristic Signature
C   HOLD  (4)            Array used to hold probed signature in 4A1 format
C   IBUF  (4)            Temporary storage array
C   IDCB  (144)          Array for Data Control Block for input from file
C   INBUF (400)          Storage array of input data (80 characters packed)
C   INTXT (10)           Packed input text array for reading from &SIGNA
C   IPBUF (10)           Temp. version of above (INTXT)
C   IRBUF (2)            Used for reading from Signature Analyzer
C   IPRMPT(12)           Holds a Logical Unit No. prompt message
C   LINE  (26)           An unpacked array of input text from &SIGNA
C   NAME  (3)            Array to hold the file name (&SIGNA)
C   POSCHR(16)           Array holding 16 possible signature values
C   TABLE (200,10)       The data table holding nodes, signatures, pointers
C   TEXT  (13)           Packed array of input text from &SIGNA
C   TMP   (3)            Temporary array for holding of re-probe node name
C   VCC   (2)            Holds the VCC Characteristic Signature
C
C
C   VARIABLES USED IN THIS PROGRAM:
C
C   COUNT                A flag used to check for necessity of a re-probe
C   FAULTS               A variable used to hold count of faults on board
C   GOOD                 Logical variable to flag good or bad signature
C   I                    Index for any and all DO loops
```

```
C  LUCRT                      Logical unit number for the CRT
C  LUSA                       Logical unit number for the Signature Analyzer
C  NUMREC                     Number of records in the Data file of Nodes
C  PRBCNT                     Counts the number of calls to GTPIN (# of probes)
C  REPROB                     Logical variable set TRUE if reprobe is needed
C  ROW                        Pointer into the current row in the TABLE
C  SUM                        The sum total of faults on the board
C  TMPROW                     A temporary holding of the ROW pointer
C
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
C
C   GLOBAL INITIALIZATION
C
C
      PROGRAM SACMP
      IMPLICIT INTEGER (A-Z)
      DIMENSION TABLE (200, 10), ARRAY (200, 10), HOLD (4), POSCHR (16)
      DIMENSION PINAME (3), CMPAR (2), DUMMY (3), INBUF (40), IPBUF (10)
      DIMENSION IBUF (4), IRBUF (2), IDCB (144), CHARS (2), TMPHLD (2)
      DIMENSION INTXT (10),  NAME (3), TEXT (13), LINE (26), TMP (3)
      DIMENSION TARRAY (15), VCC (2), GND (2), TMPCMP (2), CMPHLD (2)
      DIMENSION FILCHC (35)
      LOGICAL GOOD, VALU, REPROB
      PRBCNT = 0
      ROW    = 1
      LUCRT  = 1
      FAULTS = 0
      NUMREC = 0
      LUSA   = 35
      BELL   = 3400B
      GOOD   = .FALSE.
      VALU   = .FALSE.
      REPROB = .FALSE.
C
C
CC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CC
C
C
CC
CCC CHECK FOR HPIB LU AND INITIALIZE SA ETC...
CC
C
      CALL EXEC (13, LUSA, IEQT5, IEQT4, ISTAT)
      ISCD = 0
      IDVR = 0
      IDWN = 0
C
CC
CCC IF SELECT CODE IS 0  THEN LU IS NOT DEFINED
CC
C
      IF (IAND (IEQT4,77B) .EQ. 0) ISCD = 1
C
CC
CCC DRIVER MUST BE DVR37 FOR HP1B SO CHECK AND SEE...
```

```
CC
C
      IF (IAND (IEQT5/256, 77B) .NE. 37B) IDVR = 1

C
CC
CCC THE LU MUST BE UP SO CHECK IT....
CC
C
      IF (IAND (ISTAT, 100000B) .NE. 0) IDWN = 1
C
CC
CCC CHECK FOR ALL OF THE ABOVE TESTS SUCCESS... (AND HOPE!)
CC
C
      IF (ISCD.EQ.0 .AND. IDVR.EQ.0 .AND. IDWN.EQ.0) GO TO 9
C
CC
CCC IF WE GET HERE, WE UNFORTUNATELY HAVE AN ERROR (OR TWO...)
CC
C
      IF (ISCD .EQ. 1) WRITE (1, 10001)BELL, BELL, BELL
      IF (ISCD .EQ. 0 .AND. IDVR .EQ. 1)WRITE (1,10002)BELL,BELL,BELL
      IF (IDWN .EQ. 1) WRITE (1,10003)BELL, BELL, BELL
      STOP
C
CC
CCC CALCULATE THE HP-IB INTERFACE LU (59310A CARD IN THE COMPUTER)
CC
C
9     LUIB = LUSA - IAND (ISTAT, 77B)
C
CC
CCC SET THE REMOTE ENABLE LINE VIA AN EXEC CALL (VERY IMPORTANT)
CC
C
      CALL EXEC (3, 1600B + LUSA)
C
CC
CCC AND NOW EVERYTHING SHOULD BE READY TO GO... (LET'S HOPE!!!)
CC
C
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
CC
CCC
CCCC Initialize the data table of known good signatures by reading them in
CCC  from the &SIGNA file...
CC
C
      CALL INSNA  (TABLE  INT(1), IDCB, NAME, TEXT, LINE, NUMREC, FILCHC)
C
CC
CCC
CCCC Find out the next pin that should be probed and get the name of it
CCC
CC
C
```

```
10    CONTINUE
      CALL GTPIN (TABLE, ROW, COLUMN, DUMMY)
      PRBCNT = PRBCNT + 1
C
CC
CCC
CCCC If it's pin name is STOP then we are all done with the whole table
CCC
CC
C
      IF ((DUMMY (1) .EQ. 2HST).AND.(DUMMY (2) .EQ. 2HOP))GO TO 60
      COUNT = 0
C
CC
CCC
CCCC Prompt the operator to prompt it (the pin name)
CCC
CC
C
15    CONTINUE
      CALL PRMPT (DUMMY, REPROB)
      DO 16 I = 1,3
         TMP (I) = DUMMY (I)
16    CONTINUE
C
CC
CCC
CCCC Now get the actual signature probed from the HP-IB interface and the
CCC  Signature analyzer
CC
C
C     CALL INSIG (HOLD)
C     IF (HOLD (1) .EQ. 2HAB) GO TO 60
      CALL RBYTE (LUSH, HOLD, IPBUF)
C
CC
CCC
CCCC Now check to see if it is a good signature or not (GOOD will hold the
CCC  answer either .TRUE. or  FALSE.
CC
C
      CALL CHECK  (TABLE,ROW,GOOD,HOLD,CMPAR,CHARS,VCC,GND,PRBCNT)
      WRITE (1, 10004) (HOLD (I), I = 1, 4)
C
CC
CCC
CCCC If it is bad go to re-probe section of code (GO TO 40)
CCC
CC
C
      IF (.NOT. (GOOD)) GO TO 40
C
CC
CCC
CCCC If it is good with initial probe at board outer edge, then do
CCC  following lines of code, else we must BackTRaCe
CC
C
         IF (TABLE (ROW, 10) .NE. 0) GO TO 20
C
```

```
0299  CC
0300  CCC Now flag this node as a good node
0301  CC
0302  C
0303                TABLE (ROW, 6) = 1
0304  C
0305  CC
0306  CCC get the good goto pointer
0307  CC
0308  C
0309                ROW = TABLE (ROW, 8)
0310  C
0311  CC
0312  CCC We don't want to reprobe
0313  CC
0314  C
0315                REPROB = .FALSE.
0316  C
0317  CC
0318  CCC loop back and get the next pin that we have to probe
0319  CC
0320  C
0321                GO TO 10
0322  20          CONTINUE
0323  C
0324  CC
0325  CCC
0326  CCCC  Now see if node level below current was bad by checking previous
0327  CCC   table entry
0328  CC
0329  C
0330          CALL BKCHK (TABLE, ROW, VALU, PINAME)
0331          IF (VALU) GO TO 30
0332  C
0333  CC
0334  CCC
0335  CCCC Notify them of the fault and go back through the TABLE to see where
0336  CCC  the next probe should be
0337  CC
0338  C
0339  25          CONTINUE
0340              CALL NOTF/ (PINAME)
0341              WRITE (1, 10005)BELL
0342              FAULTS = FAULTS + 1
0343  C
0344  CC
0345  CCC   Just incremented the counter of faults, now flag the fault in the
0346  CCCC  TABLE in one of two types:  it is the furthest in-level node, or
0347  CCC   it is an ordinary fault at any other node:
0348  CC
0349  C
0350              IF (ROW .EQ. 0) GO TO 28
0351  C
0352  CC
0353  CCC  We get here for a normal node fault (not innermost)
0354  CC
0355  C
0356  C
0357  CC
0358  CCC find out where last node probed was and get it s row number
```

```
0359  CC
0360  C
0361              ROW = TABLE (ROW, 10)
0362  C
0363  CC
0364  CCC flag an actual fault here
0365  CC
0366  C
0367              TABLE (ROW, 7) = 2
0368              CALL STUCK (ROW, TMPCMP, VCC, GND, TABLE)
0369              GO TO 29
0370  C
0371  CC
0372  CCC  we get here for a fault at the innermost node level
0373  CC
0374  C
0375  28        ROW = TMPROW
0376            TABLE (ROW, 7) = 2
0377            CALL STUCK (ROW, CMPAR, VCC, GND, TABLE)
0378  C
0379  CC
0380  CCC  now BackTRaCe through the TABLE to find next node to probe
0381  CC
0382  C
0383  29        CALL BKTRC (TABLE, ROW)
0384            GO TO 10
0385  30     CONTINUE
0386  C
0387  CC
0388  CCC   NOTE: We should not ever get here as this section deals with the
0389  CCCC        case where a node is good with a previous node being bad. So
0390  CCC         flag an error and abort.
0391  CC
0392  C
0393        CALL ABORT (TABLE, ROW)
0394        CALL OUTPT (TABLE, NUMREC)
0395        STOP
0396  40    CONTINUE
0397  C
0398  CC
0399  CCC
0400  CCCC  Here after a bad probe, so probe again, and then if is still bad
0401  CCC   then follow bad-goto path in list table, else was a misprobe...
0402  CC
0403  C
0404        COUNT = COUNT + 1
0405        IF (COUNT .EQ. 1) GO TO 50
0406  C
0407  CC
0408  CCC notify them we are entering backcheck...
0409  CC
0410  C
0411            WRITE (1, 10006)BELL, BELL
0412  C
0413  CC
0414  CCC save ROW in case new row is 0
0415  CC
0416  C
0417            TMPROW = ROW
0418  C
```

```
0419  CC
0420  CCC get the new row pointer from the table
0421  CC
0422  C
0423            ROW = TABLE (ROW, 9)
0424  C
0425  CC
0426  CCC  If ROW is zero then this node is the fault as we cannot backtrace any
0427  CC   further.
0428  C
0429            IF (ROW .NE. 0) GO TO 46
0430  C
0431  CC
0432  CCC get here if this node is the fault, so store the node name and go
0433  CC  to notification section of the program
0434  C
0435               DO 45 J = 1, 3
0436                  PINAME (J) = TABLE (TMPROW, J)
0437  45           CONTINUE
0438               GO TO 25
0439  C
0440  CC
0441  CCC save this bad node's signature case the next node level probe is
0442  CC  good, we have to be able to compare it with VCC and GND characteristic
0443  C   signatures
0444  46        DO 47 I = 1, 2
0445               TMPCMP (I) = CMPAR (I)
0446  47        CONTINUE
0447            GO TO 10
0448  C
0449  CC
0450  CCC get node name back in preparation for a Re-Probe
0451  CC
0452  C
0453  50     DO 55 I = 1, 3
0454            DUMMY (I) = TMP (I)
0455  55     CONTINUE
0456         REPPOB = .TRUE.
0457         GO TO 15
0458  60     CONTINUE
0459  C
0460  CC
0461  CCC   We get here if we made it thru the table, so now do a checksum on
0462  CCCC  the bad table to see if board (or chip) fails the go/nogo test
0463  CCC
0464  CC
0465  C
0466         IF (FAULTS .EQ. 0) GO TO 80
0467            WRITE (LUCRT, 10007)FAULTS
0468            CALL DCMNT (TABLE, NUMREC, TARRAY)
0469            STOP
0470  80     WRITE (LUCRT, 10008)BELL, BELL, BELL
0471         STOP
0472  10001 FORMAT(" !!! LU # GIVEN IS NOT DEFINED",3A1)
0473  10002 FORMAT(" !!! LU # GIVEN IS NOT HP - IB",3A1)
0474  10003 FORMAT(" LU # GIVEN IS DOWN",3A1)
0475  10004 FORMAT(" SIG. IS:",1X,4A1)
0476  10005 FORMAT(" <<< EXITING BackCheck >>>",A1)
0477  10006 FORMAT(" >>> ENTERING BackCheck <<<",2A1)
0478  10007 FORMAT(" BOARD FAILS... THERE WERE ",I5," FAULTS")
```

```
10008 FORMAT(" BOARD PASSES",3A1)
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE ABORT (TABLE, ROW)
C
CC
CCC  THIS SUBROUTINE IS AN ERROR REPORTING EXIT FROM THE PROGRAM
CC
C
      IMPLICIT INTEGER (A-Z)
      DIMENSION TABLE (200,10)
      LUCRT = 32
C
CC
CCC  Write the error message to the output device
CC
C
      WRITE (LUCRT,10) ROW, (TABLE (I), I=1,3)
10    FORMAT(//," ERROR AT ROW=",I4,/," PIN AND CHIP=",3A2)
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE BKTRC (TABLE, ROW)
C
CC
CCC   THIS SUBROUTINE TRACES BACK THRU THE TABLE UNTIL IF FINDS THE LAST
CCCC  NON-ZERO GOOD-GOTO ENTRY, SETS ROW TO THAT VALUE AND GETS THE NEXT
CCC   PIN.
CC
C
      IMPLICIT INTEGER (A-Z)
      DIMENSION TABLE (200, 10)
C
CC
CCC Check to see if we now have a non-zero Good Goto pointer
CC
C
10    IF (TABLE (ROW, 8) .NE. 0) GO TO 20
C
CC
CCC No, we don t so decrement the row pointer and check again...
CC
C
         ROW = ROW - 1
         GO TO 10
C
CC
CCC Yes, we do, so set the new ROW to the value of the Good Goto pointer...
CC
C
20    ROW = TABLE (ROW , 8)
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE NOTFY (PINAME)
C
CC
CCC   THIS SUBROUTINE NOTIFIES THE TEST OPERATOR OF THE NODE LOCATION OF
CCCC  THE ACTUAL FAULT DETECTED BY THIS SIGNATURE ANALYSIS PROGRAM.
CCC
```

```
0539  CC
0540  C
0541        IMPLICIT INTEGER (A-Z)
0542        DIMENSION PINAME (3)
0543        LUCRT = 1
0544        WRITE (LUCRT ,5)(PINAME (I), I = 1,3)
0545  5     FORMAT(" LOCATION OF FAULT IS : ".3A2)
0546  C
0547  CC
0548  CCC Check to see if the fault is a VCC or GND fault (special case for each
0549  CC  board)
0550  C
0551        IF (PINAME (1) .EQ. 2HU4 .AND. PINAME (2) .EQ. 2H5. .AND.
0552       1    PINAME (3) .EQ. 2H20) GO TO 20
0553        IF (PINAME (1) .EQ. 2H U .AND. PINAME (2) .EQ. 2H45 .AND.
0554       1    PINAME (3) .EQ. 2H.2) GO TO 10
0555           RETURN
0556  10    WRITE (1, 15)
0557  15    FORMAT (" GROUND ERROR ON CPU CHIP, CHECK ORIENTATION")
0558        RETURN
0559  20    WRITE (1, 25)
0560  25    FORMAT (" VCC EPROR ON CPU CHIP, CHECK ORIENTATION")
0561        RETURN
0562        END
0563  CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0564        SUBROUTINE BKCHK (TABLE, ROW, VALU, PINAME)
0565  C
0566  CC
0567  CCC   THIS SUBROUTINE CHECKS TO SEE IF THE LAST NODE (ACTUALLY NEXT NODE
0568  CCCC  IN CIRCUIT TEST PROGRESSION) WAS BAD, AND THIS ONE IS GOOD, IF THIS
0569  CCC   IS THE CASE THEN IT IS THE NODE IN ERROR SO LOAD IT & SET VALU TRUE
0570  CC
0571  C
0572        IMPLICIT INTEGER (A-Z)
0573        LOGICAL VALU
0574        DIMENSION TABLE (200, 10), PINAME (3)
0575  C
0576  CC
0577  CCC Get the FPOM pointer and put it into POINTR
0578  CC
0579  C
0580        POINTR = TABLE (ROW, 10)
0581  C
0582  CC
0583  CCC Now check node (N-1) to see if BAD flag was set, if so then we're ok
0584  CC
0585  C
0586        IF (TABLE (POINTR, 7) .NE. 0) GO TO 5
0587  C
0588  CC
0589  CCC If we get here, it means that node N was ok, but N-1 was not, which is
0590  CC  an impossible occurrence, so flag it and return
0591  C
0592           VALU = .TRUE.
0593           RETURN
0594  5     VALU = .FALSE.
0595  C
0596  CC
0597  CCC Store the PIN nAME for notifying purposes
0598  CC
```

```
0599  C
0600        DO 10 J = 1, 3
0601           PINAME (J) = TABLE (POINTR, J)
0602  10    CONTINUE
0603        RETURN
0604        END
0605  CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0606        SUBROUTINE CHECK (TABLE,ROW,GOOD,HOLD.CMPAR,CHARS,VCC,GND,PRBCNT)
0607  C
0608  CC
0609  CCC   THIS SUBROUTINE CHECKS THE RESULT OF THE SIGNATURE JUST PROBED AND
0610  CCCC  COMPARES IT WITH THE CORRECT SIGNATURE STORED IN THE TABLE. IF IT
0611  CCC   IS GOOD IT SETS LOGICAL GOOD TO .TRUE.
0612  CC
0613  C
0614        IMPLICIT INTEGER (A-Z)
0615        LOGICAL GOOD
0616        DIMENSION TABLE (200, 10), HOLD (4), CMPAR (2), CHARS (2)
0617        DIMENSION VCC (2), GND (2)
0618        PAKLEN = 2
0619        NUMCHR = 4
0620  C
0621  CC
0622  CCC If PRBCNT is 1 or 2, then this is a VCC or GND characteristic signature
0623  CC  so check it, and if it is, then store it in either VCC or GND
0624  C
0625        IF (PRBCNT .NE. 1 .AND. PRBCNT .NE. 2) GO TO 19
0626           IF (PRBCNT .EQ. 2) GO TO 13
0627              DO 12 J = 1, 2
0628                 VCC (J) = TABLE (ROW, (J + 3) )
0629  12          CONTINUE
0630              GO TO 19
0631  13        DO 14 J = 1, 2
0632              GND (J) = TABLE (ROW, (J + 3) )
0633  14        CONTINUE
0634  C
0635  CC
0636  CCC Subroutine RCVRT converts the signature in HOLD (4A1) into 2A2 in
0637  CC  CMPAR so it can compare it with the table
0638  C
0639  19    CALL RCVRT (HOLD, CMPAR, NUMCHR, PAKLEN, CHARS)
0640  C
0641  CC
0642  CCC   NOW WE HAVE THE PROBED SIGNATURE IN CMPAR IN A 2A2 FORMAT (SAME AS IN
0643  CC    TABLE), SO WE CAN NOW COMPARE THEM.
0644  C
0645        IF((CMPAR(1).EQ.TABLE(ROW,4)).AND.
0646       +   (CMPAR(2).EQ.TABLE(ROW,5))) GO TO 20
0647           GOOD = .FALSE.
0648           TABLE (ROW, 7) = 1 .
0649           RETURN
0650  20    GOOD = .TRUE.
0651        TABLE (ROW, 6) = 1
0652        RETURN
0653        END
0654  CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0655        SUBROUTINE RBYTE (CUSH, HOLD, IRBUF)
0656  C
0657  CC
0658  CCC   THIS SUBROUTINE IS THE INTERFACE BETWEEN THE SIGNATURE ANALYZER
```

```
0659  CCCC  AND THE MAIN FORTRAN PROGRAM.  IT GETS THE SIGNATURE AND STORES IT
0660  CCC   IN THE VARIABLE HOLD IN A 4A1 FORMAT.
0661  CC
0662  C
0663        IMPLICIT INTEGER (A-Z)
0664        DIMENSION POSCHR (16), HOLD (4), IRBUF (2)
0665        DATA POSCHR /1H0,1H1,1H2,1H3,1H4,1H5,1H6,1H7,1H8,1H9,1HA,1HC,
0666       +            1HF,1HH,1HP,1HU/
0667  C
0668  CC
0669  CCC NOW PAUSE TO GIVE THE OPERATOR A CHANCE TO PROBE. WHEN HE(SHE) DOES
0670  CC  THEY THEN ENTER A CARRIAGE RETURN (<CR>) AND IT READS THE SIGNATURE
0671  C
0672        WRITE (1,1)
0673  1     FORMAT(" ENTER <CR> AFTER PROBE IS SET")
0674        READ  (1,2)IDUM
0675  2     FORMAT(I2)
0676        IF (IDUM .EQ. 01 .OR. IDUM .EQ. 10) STOP
0677        CALL EXEC (1, 2100B+LUSA, IRBUF, -4)
0678        HOLD (1) = POSCHR (IAND (IRBUF (1), 7400B) /256 + 1)
0679        HOLD (2) = POSCHR (IAND (IRBUF (1), 17B) + 1)
0680        HOLD (3) = POSCHR (IAND (IRBUF (2), 7400B) /256 + 1)
0681        HOLD (4) = POSCHR (IAND (IRBUF (2), 17B) + 1)
0682        RETURN
0683        END
0684  CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0685        SUBROUTINE PRMPT (DUMMY, REPROB)
0686  C
0687  CC
0688  CCC   THIS SUBROUTINE PROMPTS THE TEST OPERATOR TO PROBE THE PIN
0689  CC    SPECIFIED BY ALPHANUMERIC VARIABLE DUMMY
0690  C
0691        IMPLICIT INTEGER (A-Z)
0692        DIMENSION DUMMY (3)
0693        LOGICAL REPROB
0694        LUCRT = 1
0695        IF (REPROB) GO TO 15
0696           WRITE (LUCRT,10) DUMMY (I), I= 1,3)
0697  10       FORMAT(1X," PLEASE PROBE PIN : ",3A2)
0698           RETURN
0699  15    WRITE (LUCRT, 20)(DUMMY (I), I = 1, 3)
0700  20    FORMAT(" POSSIBLE MISPROBE",/," PLEASE REPROBE: ",3A2)
0701        REPROB = .FALSE.
0702        RETURN
0703        END
0704  CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0705        SUBROUTINE GTPIN (TABLE, ROW, COLUMN, DUMMY)
0706  C
0707  CC
0708  CCC   THIS SUBROUTINE GETS THE NEXT ALPHANUMERIC CHARACTER FOR THE
0709  CCCC  PIN NUMBER TO BE PROBED NEXT, AND PLACES IT IN DUMMY
0710  CCC
0711  CC
0712  C
0713        IMPLICIT INTEGER (A-Z)
0714        DIMENSION TABLE (200, 10), DUMMY (3)
0715        DO 10 I = 1,3
0716           DUMMY (I) = TABLE (ROW ,I)
0717  10    CONTINUE
0718        RETURN
```

```
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE INSHA(TABLE,INTXT,IDCB,NAME,TEXT,LINE,NUMREC,FILCHC)
C
CC
CCC
CCCC  This subroutine reads in the data from the Signature Analysis
CCC   Data file created for each unique PCB, and creates a table in
CC    memory for the rest of SACMPR to utilize.
C
C
C
C
C
C   ARRAYS AND VARIABLES USED IN THIS SUBPROGRAM:
C
C   FILCHC(35)     Array to hold data file name and description of contents
C   INTXT (13)     The data Input Text Buffer
C   IDCB (144)     Input Data Control Block buffer
C   LINE (26)      Holds unpacked version of an input record
C   NAME (3)       Holds the file name of SA data file (in 3A2 format)
C   TABLE (200,10) Table to hold all the input data
C   TEXT (13)      Holding array used by CNVRT to unpack INTXT
C
C   CR             Cartridge Reference Number of Data file
C   TXTLEN         the length of the TEXT array
C   LINLEN         the length of the LINE array
C   IL             # of words to read in a CALL READF record
C   IOPTN          file type parameter to CALL OPEN (not used, hence 0)
C   SECODE         the security code of the SA data file
C   IDCBS          size of the Data Control Block in words
C   IERR           Error parameter on all FMP calls
C   ROW            Row pointer into TABLE
C   NUMREC         Number of Records in the Data file of nodes
C
C
C   EXTERNAL SUBROUTINES AND FUNCTIONS CALLED:
C
C   CNVRT          converts packed text (A2) into unpacked text (A1)
C   NUM            converts hollerith characters into numbers (integer)
C   UNPAK          unpacks an A2 variable into 2-A1 variables
C
C
CC
CCC  INITIALIZATION
CC
C
      IMPLICIT INTEGER (A-Z)
      DIMENSION TABLE (200,10), INTXT (13), IDCB (144), NAME (3)
      DIMENSION TEXT (13), LINE (26), FILCHC (35)
      TXTLEN = 13
      LINLEN = 26
      IOPTN = 0
      SECODE = 2HRT
      IDCBS = 144
      INTXT (3) = 2H
C
CC
CCC Now open up the data-file-name data file
CC
```

```
0779  C
0780        NAME (1) = 2HSA
0781        NAME (2) = 2HFI
0782        NAME (3) = 2HLS
0783        IL       = 35
0784        CR       = 18
0785  C
0786  CC
0787  CCC Now actually open up the file
0788  CC
0789  C
0790        CALL OPEN (IDCB, IERR, NAME)
0791  101   WRITE (1,1)
0792  1     FORMAT(//," FOLLOWING IS LIST OF AVAILABLE FILES, ENTER YOUR",/,
0793       1" CHOICE FROM THIS LIST.  TO RE-PRINT THE LIST ENTER ?? IN ",/,
0794       2" RESPONSE TO REQUEST FOR FILE NAME",/,
0795       3//," FILE NAME",2X,"CR",2X,"DESCRIPTION OF CONTENTS",/,
0796       4" ---------  --  ----------------------",//,
0797       5" WHEN ASKED TO HIT <CR> AFTER A PROBE, SHOULD YOU WISH TO ",/,
0798       6" EXIT THE PROGRAM, TYPE A 1")
0799        CALL RWNDF (IDCB)
0800  2     CONTINUE
0801  C
0802  CC
0803  CCC Now read one entry from the data-file name data file
0804  CC
0805  C
0806        CALL READF (IDCB, IERR, FILCHC, IL, LEN)
0807        IF (IERR .LT. 0) GO TO 9000
0808        IF (FILCHC (1) .EQ. 2H  ) GO TO 4
0809           WRITE (1, 3) FILCHC
0810  3        FORMAT(1X,3A2.5X,A2,2X,31A2)
0811           GO TO 2
0812  4     WRITE (1, 5)
0813  5     FORMAT(//," ENTER FILENAME:")
0814        READ (1, 6) (NAME (I), I = 1, 3)
0815  6     FORMAT(3A2)
0816        IF (NAME (1) .EQ. 2H??) GO TO 101
0817        WRITE (1, 7)
0818  7     FORMAT(" ENTER CR:")
0819        READ (1, 8) CR
0820  8     FORMAT(I2)
0821        IF (CR .LT. 15 .OR. CR. GT. 20) GO TO 4
0822           CALL OPEN (IDCB, IERR. NAME)
0823           IF (IERR .LT. 0) GO TO 4
0824  C
0825  CC
0826  CCC we get here if we successfully opened their data file
0827  CC
0828  C
0829        ROW = 1
0830  C
0831  CC
0832  CCC  Now read the ROWth record...
0833  CC
0834  C
0835  9        CALL READF(IDCB,IERR,INTXT,IL,LEN)
0836  C
0837  CC
0838  CCC  If there is a reading error, go to the error handler
```

```
0839   CC
0840   C
0841             IF (IERR .LT. 0) GO TO 9000
0842   C
0843   CC
0844   CCC  Check for the word END, indicative of EOF
0845   CC
0846   C
0847             IF (INTXT (3) .EQ. 2HND) GO TO 40
0848   C
0849   CC
0850   CCC  Copy the Pin, Chip, and Signature into the data table
0851   CC
0852   C
0853             DO 10 I = 1,5
0854                TABLE (ROW, I) =  INTXT (I)
0855   10        CONTINUE
0856   C
0857   CC
0858   CCC  Now unpack the record to facilitate conversion into integers...
0859   CC
0860   C
0861             CALL CNVRT (INTXT, LINE, TXTLEN, LINLEN)
0862   C
0863   CC
0864   CCC  put integer value of 6th & 7th column of data file into TABLE
0865   CC
0866   C
0867             TABLE (ROW, 6) = 10 * NUM (LINE (11)) + NUM (LINE (12))
0868             TABLE (ROW, 7) = 10 * NUM (LINE (13)) + NUM (LINE (14))
0869   C
0870   CC
0871   CCC  Now convert the three pointers (GGTO, BGTO, FROM) into integers
0872   CC   and store them in their proper location in the TABLE
0873   C
0874             TABLE (ROW, 8) = 1000*NUM (LINE (15)) +100*NUM(LINE(16))
0875             TABLE(ROW,8)=TABLE(ROW,8)+10*NUM(LINE(17))+NUM(LINE(18))
0876             TABLE(ROW,9)=1000*NUM(LINE(19))+100*NUM(LINE(20))
0877             TABLE(ROW,9)=TABLE(ROW,9)+10*NUM(LINE(21))+NUM(LINE(22))
0878             TABLE(ROW,10)=1000*NUM(LINE(23))+100*NUM(LINE(24))
0879             TABLE(ROW,10)=TABLE(ROW,10)+10*NUM(LINE(25))+NUM(LINE(26))
0880   C
0881   CC
0882   CCC  Increment the ROW pointer...
0883   CC
0884   C
0885             ROW = ROW + 1
0886   C
0887   CC
0888   CCC  and go back to read another record
0889   CC
0890   C
0891             GO TO 9
0892   40     CONTINUE
0893   C
0894   CC
0895   CCC  we get here upon an EOF
0896   CC
0897   C
0898          NUMREC = ROW - 1
```

```
      RETURN
9000  WRITE (6,9010)IERR,NAME,CR
9010  FORMAT(" ERROR #",I4,"ON FILE:",3A2,"::",I3)
      STOP
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE CNVRT (TEXT, LINE, TXTLEN, LINLEN)
C
CC
CCC  This subroutine converts packed line in TEXT into an unpacked string
CC   twice as long, in LINE
C
C
C
C VARIABLES AND ARRAYS USED IN THIS SUBROUTINE
C
C
C  CHARS(2)      holds two unpacked characters used in call unpack
C  TEXT(TXTLEN) holds the packed text that needs to be unpacked
C  LINE(LINLEN) holds the unpacked version of TEXT
C  PTR           a horizontal column pointer into LINE
C
C
C EXTERNAL SUBROUTINES CALLED
C
C UNPAK          unpacks two packed characters into two unpacked CHARS
C
      IMPLICIT INTEGER (A - Z)
      DIMENSION CHARS (2), TEXT (TXTLEN), LINE (LINLEN), TMP (2)
      PTR = 0
C
CC
CCC  loop from the beginning of the TEXT to the end
CC
C
      DO 10 I = 1, TXTLEN
         PTR = PTR + 1
C
CC
CCC  unpacking as we go... using routine UNPAK
CC
C
         CALL UNPAK (TEXT (I), TMP)
C
CC
CCC  put the unpacked version into line
CC
C
         LINE (PTR) = TMP (1)
         PTR = PTR + 1
         LINE (PTR) = TMP (2)
10    CONTINUE
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      FUNCTION NUM (CHAR)
C
CC
CCC  This function converts a hollerith represented character in CHAR
CC   into an integer number in NUM
```

```
0959  C
0960  C
0961  C
0962  C  VARIABLES AND ARRAYS USED IN THIS FUNCTION
0963  C
0964  C
0965  C  CHAR       the actual character upon which conversion is to be done
0966  C
0967  C
0968  C  SUBROUTINES AND FUNCTIONS USED BY THIS FUNCTION
0969  C
0970  C
0971  C  CODE       system subroutine needed to do type conversion
0972  C
0973  C
0974        IMPLICIT INTEGER (A - Z)
0975  C
0976  CC
0977  CCC  we treat a blank as a leading zero, so handle accordingly
0978  CC
0979  C
0980        IF (CHAR .EQ. 1H ) GO TO 10
0981  C
0982  CC
0983  CCC  call system CODE routine to do type conversion
0984  CC
0985  C
0986           CALL CODE
0987  C
0988  CC
0989  CCC  "Read" the character out of CHAR into NUM
0990  CC
0991  C
0992           READ (CHAR, 5) NUM
0993  C
0994  CC
0995  CCC  by the following format
0996  CC
0997  C
0998  5        FORMAT(I1)
0999           RETURN
1000  10    NUM = 0
1001        RETURN
1002        END
1003  CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
1004        SUBROUTINE UNPAK (PAK, CHARS)
1005  C
1006  CC
1007  CCC
1008  CCCC This subroutine unpacks characters stored in variable PAK in an A2
1009  CCC  format, into 2 chars, at A1 each in the array CHARS(1), CHARS(2)
1010  CC
1011  C
1012        IMPLICIT INTEGER (A - Z)
1013        DIMENSION CHARS (2)
1014        CALL CODE
1015        READ(PAK, 5) CHARS
1016  5     FORMAT(2A1)
1017        RETURN
1018        END
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE ZEROR (ARRAY, LENGTH, WIDTH)
C
CC
CCC This subroutine sets ARRAY (LENGTH, WIDTH) to zero
CC
C
      IMPLICIT INTEGER (A-Z)
      DIMENSION ARRAY (LENGTH, WIDTH)
      DO 20 I = 1,LENGTH
         DO 10 J = 1,WIDTH
            ARRAY (I, J) = 0
10       CONTINUE
20    CONTINUE
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE OUTPT (TABLE, NUMREC)
C
CC
CCC This subroutine outputs the TABLE full of nodes and signatures
CC
C
      IMPLICIT INTEGER (A - Z)
      DIMENSION TABLE (200 ,10)
      DO 10 I = 1, NUMREC
         WRITE (6,5)I,(TABLE (I,J),J = 1,10)
5        FORMAT(1X,I3,"!",3A2,"!",2A2,"!",I1,"!",I1,3("!",I4))
10    CONTINUE
      WRITE (6,15)
15    FORMAT(1H1)
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE RCVRT (HOLD, CMPAR, NUMCHR, PAKLEN, CHARS)
C
CC
CCC  This subroutine takes the text in HOLD and using subroutine PACK
CC   packs it into CMPAR
C
C
C
C VARIABLES AND ARRAYS USED IN THIS SUBROUTINE
C
C
C  HOLD(NUMCHR)  unpacked HOLD of characters that need packing
C  CMPAR(PAKLEN)  packed CMPAR of characters that came from HOLD
C  NUMCHR          the number of unpacked characters in HOLD
C  PAKLEN          the number of packed characters in CMPAR
C  FLAG            flag to tell if we need to add an extra trailing blank
C  CHARS(2)        array of two unpacked characters from which to pack
C  J               horizontal pointer into the HOLD array
C
C
C
C EXTERNAL SUBROUTINES CALLED FROM THIS SUBROUTINE
C
C
C  PACK          packs two characters in CHAR(1)(2) into PAK
C  LENTH         function to return the number of characters in STRING
```

```
1079  C
1080        IMPLICIT INTEGER (A - Z)
1081        DIMENSION HOLD (NUMCHR), CMPAR (PAKLEN), CHARS (2)
1082        FLAG = 0
1083  C
1084  CC
1085  CCC   get the number of characters in HOLD
1086  CC
1087  C
1088        NUMCHR = LENTH (HOLD)
1089  C
1090  CC
1091  CCC   find out if it needs a trailing blank before it gets packed
1092  CC
1093  C
1094        IF (MOD (NUMCHR, 2) .EQ. 0) GO TO 5
1095  C
1096  CC
1097  CCC   here if it needs an extra blank, so set FLAG accordingly
1098  CC
1099  C
1100           FLAG = 1
1101  5     CONTINUE
1102        J = 0
1103  C
1104  CC
1105  CCC   loop for the number of characters in HOLD
1106  CC
1107  C
1108        DO 20 I = 1, NUMCHR
1109  C
1110  CC
1111  CCC   increment horizontal column pointer into HOLD to get next character
1112  CC
1113  C
1114           J = J + 1
1115  C
1116  CC
1117  CCC   put the first character to be packed into CHARS
1118  CC
1119  C
1120           CHARS (1) = HOLD (J)
1121  C
1122  CC
1123  CCC   see if we need an extra blank to be put in CHARS (2)
1124  CC
1125  C
1126           IF (.NOT. ((I .EQ. NUMCHR) .AND. (FLAG .EQ. 1))) GO TO 10
1127  C
1128  CC
1129  CCC   we get here if the trailing blank is needed
1130  CC
1131  C
1132              CHARS (2) = 1H
1133              GO TO 15
1134  C
1135  CC
1136  CCC   increment column pointer to get the next character
1137  CC
1138  C
```

```
1139  10        J = J + 1
1140  C
1141  CC
1142  CCC  put the second character into CHARS
1143  CC
1144  C
1145  15        CHARS (2) = HOLD (J)
1146  C
1147  CC
1148  CCC  and pack the two of them into PAK
1149  CC
1150  C
1151            CALL PACK (PAK, CHARS)
1152  C
1153  CC
1154  CCC  and then put the packed characters into the array CMPAR
1155  CC
1156  C
1157            CMPAR (I) = PAK
1158  20     CONTINUE
1159         RETURN
1160         END
1161  CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
1162         FUNCTION LENTH (STRING)
1163  C
1164  CC
1165  CCC This Function determines the number of characters in the array
1166  CC  STRING.  Maximum length is 80 and if the STRING is blank 0 is returned
1167  C
1168         IMPLICIT INTEGER (H - Z)
1169         DIMENSION STRING (80)
1170  C
1171  CC
1172  CCC Set string pointer at rightmost end of array
1173  CC
1174  C
1175         I = 80
1176  C
1177  CC
1178  CCC now check to see if we are at left edge of array
1179  CC
1180  C
1181  5      IF (I .LT. 1) GO TO 20
1182  C
1183  CC
1184  CCC if were not, then check to see if this character is non-blank
1185  CC
1186  C
1187            IF (STRING (I) .EQ. 1H ) GO TO 10
1188  C
1189  CC
1190  CCC if it is non-blank, then I is the length of this array
1191  CC
1192  C
1193               LENTH = I
1194               RETURN
1195  C
1196  CC
1197  CCC if it is a blank, however, then shift the pointer to the left and
1198  CC  loop again.
```

```
C
10        I = I - 1
          GO TO 5
20     LENTH = 0
       RETURN
       END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
       SUBROUTINE PACK (PAK, CHARS)
C
CC
CCCC This subroutine packs characters stored in array CHAR(1), CHAR(2) in
CCC  an A1 format each, into the variable PAK in an A2 format
CC
C
       IMPLICIT INTEGER (A - Z)
       DIMENSION CHARS (2)
       PAK = 2H
       CALL CODE
       WRITE (PAK, 5) CHARS
5      FORMAT(2A1)
       RETURN
       END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
       SUBROUTINE DCMNT (TABLE, NUMREC, TARRAY)
C
CC
CCC
CCCC This subroutine Documents all of the faults found in the UUT
CCC
CC
C
C
C
C   VARIABLES AND ARRAYS USED IN THIS SUBROUTINE
C
C   LOOP          Pointer used to loop through the TABLE
C   NUMREC        The number of records in TABLE
C   TABLE         The data Table holding all nodes and signatures, etc.
C   TARRY         The array used to hold the system time and date
C
C   EXTERNAL SUBROUTINES CALLED
C
C   FTIME         System routine to get time and date
C
C
       IMPLICIT INTEGER (A - Z)
       DIMENSION TABLE (200, 10), TARRAY (15)
       CALL FTIME (TARRAY)
       CALL OUTPT (TABLE, NUMREC)
C
CC
CCC Write out heading for error(s) on strip printer (LU = 32)
CC
C
C
CC
CCC LOOP through the TABLE scanning for fault flagged nodes...
CC
C
       DO 20 LOOP = 1, NUMREC
```

```
1259            IF (TABLE (LOOP, 7) .LE. 1) GO TO 20
1260               IF (TABLE (LOOP, 7) .NE. 2) GO TO 10
1261                  WRITE (32, 9) (TABLE (LOOP, J), J = 1, 3)
1262  9               FORMAT(" FAULT: ",3A2,"    ")
1263                  GO TO 20
1264  10              IF (TABLE (LOOP, 7) .EQ. 4) GO TO 12
1265                  WRITE (32, 11) (TABLE (LOOP, J), J = 1, 3)
1266  11              FORMAT(" SA0 -  ",3A2,"    ")
1267                  GO TO 20
1268  12              WRITE (32, 13) (TABLE (LOOP, J), J = 1, 3)
1269  13              FORMAT(" SA1 -  ",3A2,"    ")
1270  20     CONTINUE
1271         WRITE (32,30)
1272  30     FORMAT(" BAD SIGNATURE(S):  ",/," --------------------")
1273         WRITE (32,35) (TARRAY (I), I = 1, 15)
1274  35     FORMAT(" TIME:    ",1X,5A2,/,1X,10A2,/," --------------------",/,
1275        +4X,"TEST REPORT    ")
1276         RETURN
1277         END
1278  CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
1279         SUBROUTINE STUCK (ROW, TMPHLD, VCC, GND, TABLE)
1280  C
1281  CC
1282  CCC
1283  CCCC This subroutine checks to see if the fault found was a Stuck at 0, or
1284  CCC  Stuck at 1. and if so flags a SA0 with 3 in column 7 of table, or 4 if
1285  CC   it is a SA1
1286  C
1287         IMPLICIT INTEGER (A - Z)
1288         DIMENSION TABLE (200, 10), VCC (2), GND (2), TMPHLD (2)
1289  C      WRITE (1,1)TMPHLD,GND,VCC
1290  1      FORMAT(" TMPHLD ",2A2,2X,"GND ",2A2,2X,"VCC ",2A2)
1291  C
1292  CC
1293  CCC Check to see if it is a SA1, SA0
1294  CC
1295  C
1296         IF (TMPHLD(1).EQ.VCC(1) .AND. TMPHLD(2).EQ.VCC(2)) GO TO 20
1297            IF(.NOT.(TMPHLD(1).EQ.GND(1).AND.TMPHLD(2).EQ.GND(2)))GO TO 30
1298  C
1299  CC
1300  CCC Get here if it is a GND (SA0) fault
1301  CC
1302  C
1303               TABLE (ROW, 7) = 3
1304               RETURN
1305  20        CONTINUE
1306  C
1307  CC
1308  CCC Get here if it is a VCC (SA1) fault
1309  CC
1310  C
1311            TABLE (ROW, 7) = 4
1312            RETURN
1313  30     RETURN
1314         END
1315         END$
```

APPENDIX A – Software
Section A.1 – Signature Analysis Software

A.1.5 – SACMPR 8080 A/B MICROPROCESSOR DATA TABLE FORMAT

Row #	Node Name			Good Signature		Good Flag	Bad Flag	Good Go To	Bad Go To	"From" Go To	Element Name
	1	2	3	4	5	6	7	8	9	10	Word #
	1 2	3 4	5 6	7 8	9 10	11 12	13 14	15 16	17 18	19 20	Byte #
1	U 4	5 .	20					0 2	0 0	0 0	
2	U 4	5 .	2					0 3	0 0	0 0	
3	U 4	5 .	1					0 9	0 4	0 0	
4	U 4	5 .	4					0 0	0 5	0 3	
5	U 4	5 .	6					0 0	0 6	0 4	
6	U 4	5 .	9					0 0	0 7	0 5	
7	U 4	5 .	14					0 0	0 8	0 6	
8	U 4	5 .	15					0 0	0 0	0 7	
9	U 4	5 .	17					1 5	1 0	0 0	
10	U 4	5 .	18					0 0	1 1	0 9	
11	U 4	5 .	19					0 0	1 1	1 0	
12	U 4	5 .	25					0 0	1 3	1 1	
13	U 4	5 .	26					0 0	1 4	1 2	
14	U 4	5 .	27					0 0	0 0	1 3	
15	U 4	5 .	28					2 3	1 6	0 0	
16	U 4	5 .	29					0 0	1 7	1 5	
17	U 4	5 .	30					0 0	1 8	1 6	
18	U 4	5 .	31					0 0	1 9	1 7	
19	U 4	5 .	32					0 0	2 0	1 8	
20	U 4	5 .	33					0 0	2 1	1 9	
21	U 4	5 .	34					0 0	2 2	2 0	
22	U 4	5 .	35					0 0	0 0	2 1	

A.1.6 – SAMPLE SIGNATURE FILE, 8080 A/B MICROPROCESSOR USING NOOP

Row #	Node Name						Correct Signature				Good Flag		Bad Flag		Good Go To		Bad Go To		From Go To	
	1		2		3		4		5		6		7		8		9		10	
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	U	4	5	.	2	0	7	5	5	U						2		0		0
2		U	4	5	.	2	0	0	0	0						3		0		0
3		U	4	5	.	1	H	H	8	6						4		0		0
4		U	4	5	.	4	7	5	5	U						5		0		0
5		U	4	5	.	6	7	5	5	U						6		0		0
6		U	4	5	.	9	7	5	5	U						7		0		0
7	U	4	5	.	1	4	7	5	5	U						8		0		0
8	U	4	5	.	1	5	7	5	5	U						9		0		0
9	U	4	5	.	1	7	0	0	0	0					1	0		0		0
10	U	4	5	.	1	8	7	5	5	U					1	1		0		0
11	U	4	5	.	1	9	7	5	5	U					1	2		0		0
12	U	4	5	.	2	5	H	3	3	5					1	3		0		0
13	U	4	5	.	2	6	C	1	1	3					1	4		0		0
14	U	4	5	.	2	7	7	0	5	0					1	5		0		0
15	U	4	5	.	2	8	7	5	5	U					1	6		0		0
16	U	4	5	.	2	9	0	7	7	2					1	7		0		0
17	U	4	5	.	3	0	C	4	C	3					1	8		0		0
18	U	4	5	.	3	1	A	A	0	8					1	9		0		0
19	U	4	5.	.	3	2	7	2	1	1					1	0		0		0
20	U	4	5	.	3	3	A	3	C	1					2	1		0		0
21	U	4	5	.	3	4	7	7	0	7					2	2		0		0
22	U	4	5	.	3	5	5	7	7	A					2	3		0		0
23	U	4	5	.	3	6	0	0	0	0					2	4		0		0
24	U	4	5	.	3	7	A	C	9	9					2	5		0		0
25	U	4	5	.	3	8	P	C	F	3					2	6		0		0
26	U	4	5	.	3	9	1	1	8	0					2	7		0		0
27	U	4	5	.	4	0	8	9	F	L					2	8		0		0
28	S	T	O	P	S	T														
29	E	N	D	E	N	D														

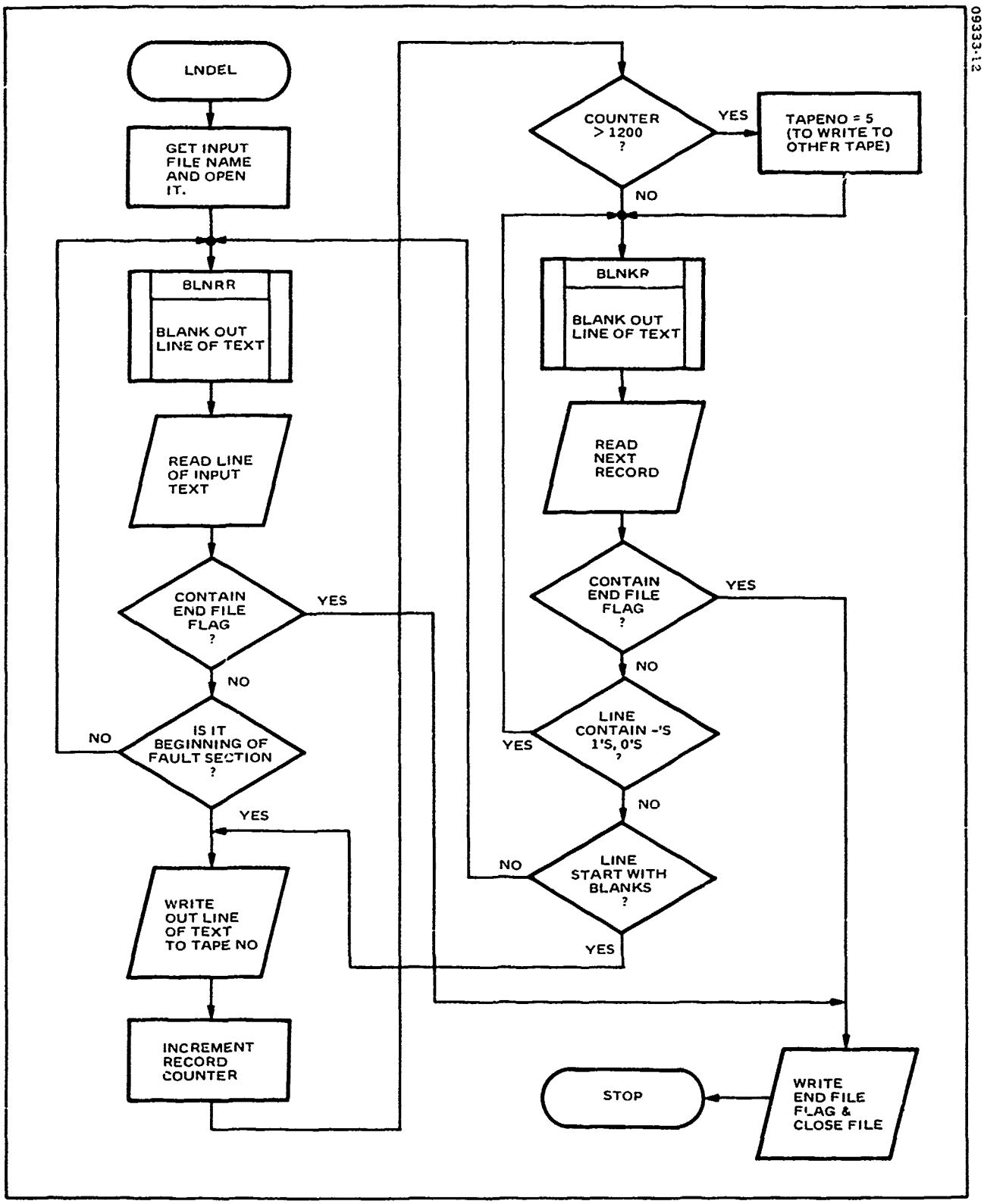
NOTE: The data are taken for NOOP Program with the START and CLOCK switches in the IN, and the STOP switch in the OUT position at the HP5004A Signature Analyzer front panel.

SECTION A.2

LNDEL

APPENDIX A – Software
Section A.2 – LNDEL

A.2.1 – LNDEL FLOW CHART



```
0059          DIMENSION INTXT (18), NAME (3), IDCB (144)
0060          DATA EOF/2HEF/
0061          TAPENO = 4
0062          IL = 18
0063          COUNT = 0
0064    C
0065    CC
0066    CCC Prompt to get the input file name
0067    CC
0068    C
0069          WRITE (1,1)
0070    1     FORMAT (" ENTER INPUT FILE NAME:")
0071          READ (1,2)NAME
0072    2     FORMAT(3A2)
0073    C
0074    CC
0075    CCC Open the input file name, if an open error, flag it and stop
0076    CC
0077    C
0078          CALL OPEN (IDCB, IERR, NAME)
0079          IF (IERR .LT. 0) GO TO 9000
0080    C
0081    CC
0082    CCC Blank out the current line of text
0083    CC
0084    C
0085    60    CALL BLNKR (INTXT)
0086    C
0087    CC
0088    CCC Read the next line of input text
0089    CC
0090    C
0091          CALL READF (IDCB, IERR, INTXT, IL)
0092          IF (IERR .LT. 0) GO TO 9000
0093    C
0094    CC
0095    CCC Check to see it the line has "TOTAL..." in it, indicative of our EOF
0096    CC
0097    C
0098          IF (INTXT (2) .EQ. 2HOT) GO TO 200
0099    C
0100    CC
0101    CCC While the word "FAULT..." does NOT appear, loop at 60 reading lines of
0102    CC  input
0103    C
0104              IF (INTXT (2)  NE. 2HAU) GO TO 60
0105    C
0106    CC
0107    CCC Until it does..., then write out all subsequent lines, excluding any and
0108    CC  all lines containing --- s until we get EOF or non blank first character
0109    C
0110    70            WRITE (TAPENO,75)INTXT
0111    75            FORMAT(18A2)
0112    C
0113    CC
0114    CCC Increment record counter (to see if we need to write to second tape
0115    CC
0116    C
0117                  COUNT = COUNT + 1
0118                  IF (COUNT .GE. 1200) TAPENO = 5
```

```
0119  C
0120  CC
0121  CCC Blank out line of text in preparation to read the next one
0122  CC
0123  C
0124  80          CALL BLNKR (INTXT)
0125  C
0126  CC
0127  CCC Read the next line of input text
0128  CC
0129  C
0130              CALL READF (IDCB, IERR, INTXT, IL)
0131  C
0132  CC
0133  CCC Check for EOF
0134  CC
0135  C
0136              IF (INTXT (2) .EQ. 2HOT) GO TO 200
0137  C
0138  CC
0139  CCC If chars are any comb. of -,1,0 then skip this line
0140  CC
0141  C
0142                  IF (INTXT (5) .LE. 020061B) GO TO 80
0143  C
0144  CC
0145  CCC If the first few characters are non-blank, then we have reached the end
0146  CC  of the current fault section, write no more lines, go back and read more
0147  C
0148                      IF (INTXT (2) .NE. 2H  ) GO TO 60
0149  C
0150  CC
0151  CCC Otherwise, go back and continue reading and writing...
0152  CC
0153  C
0154                          GO TO 70
0155  200   CONTINUE
0156  C
0157  CC
0158  CCC When we reach EOF, write out the EF for a flag
0159  CC
0160  C
0161        WRITE (1HPEN0,205)EOF
0162  205   FORMAT (A2)
0163        CALL CLOSE (IDCB)
0164        STOP
0165  9000  WRITE (1,9010)
0166  9010  FORMAT(" DATA FILE ERROR")
0167        STOP
0168        END
0169        SUBROUTINE BLNKR (INTXT)
0170  C
0171  CC
0172  CCC This subroutine blanks out the text it is passed as input
0173  CC
0174  C
0175        IMPLICIT INTEGER (A-Z)
0176        DIMENSION INTXT (18)
0177        DO 10 I = 1, 18
0178           INTXT (I) = 2H
```

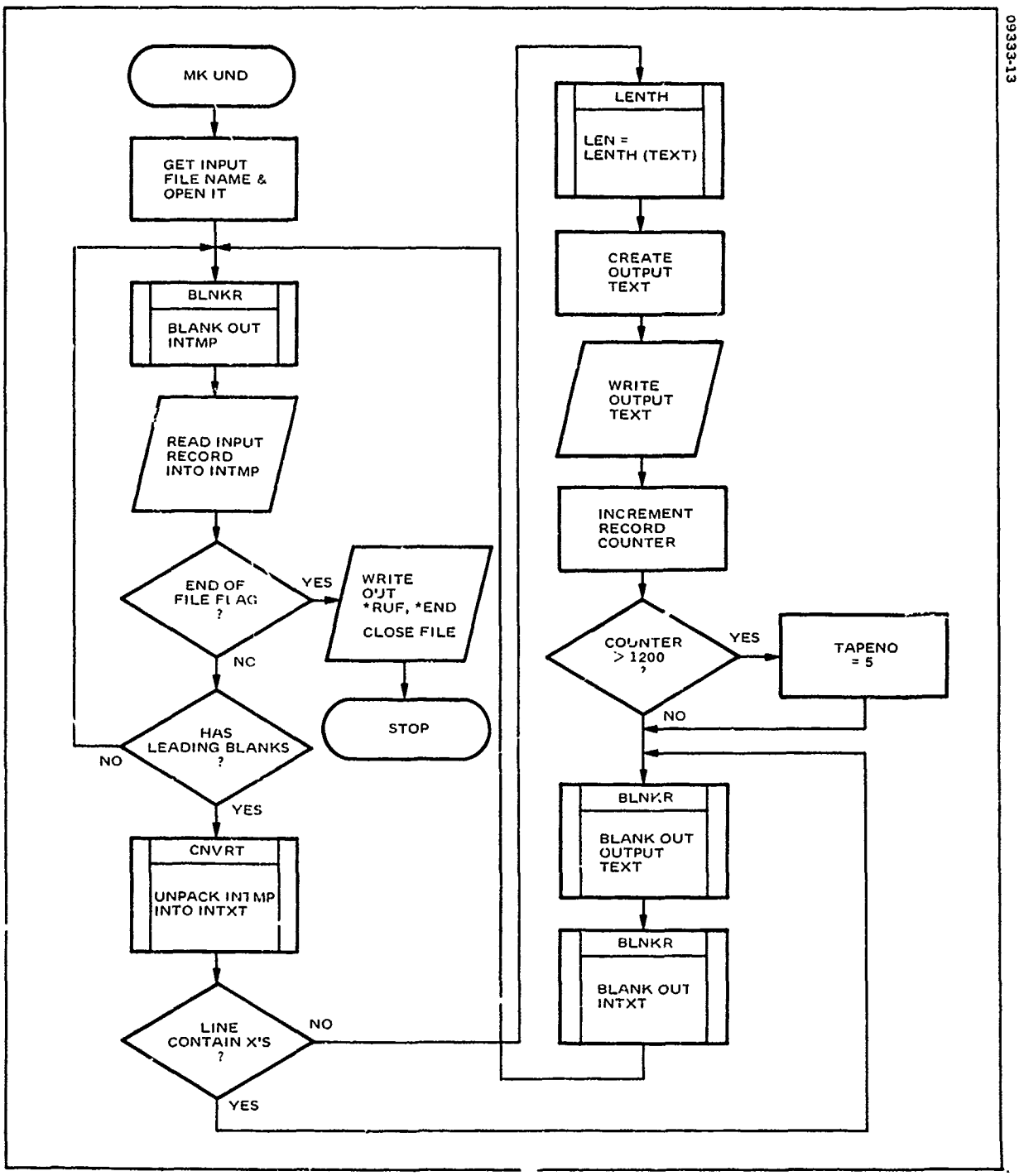
```
0179   10     CONTINUE
0180          RETURN
0181          END
0182          END$
```

SECTION A.3

MKUND

APPENDIX A – Software
Section A.3 – MKUND

A.3.1 – MKUND FLOW CHART



A.3.2 – MKUND LISTING

```
MKUND  T=00004 IS ON CP00018 USING 00010 BLKS R=0000

FTN4.L
      PROGRAM MKUND
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
C
C     PROGRAM:       MKUND
C
C     PURPOSE:       To take the faults detected output of LNDEL and make
C                    it into a +UNDected fault listing in a format which
C                    can be used as an input for further runs of SIMUL,
C                    enabling SIMUL to run faster and more efficiently.
C
C     PROGRAMMER:    DAVID S. WAGNER    -    BLDG. 688/T125
C                    7030 E POTAWATAMI       HUGHES/(213) 802-4190
C                    TUCSON, AZ 85715        FULLERTON, CA 92634
C                    602-296-2760
C
C     DATE:          23 - JUL - 80
C
C
C     DATA FILES:
C                    INPUT:    Any name desired, user is queried.  This file
C                              will be the output of the LNDEL program.
C
C                    OUTPUT:   Output will be directed to tape 4 for the
C                              first 1200 faults, thereafter to tape 5
C                              until 2400 are reached (the maximum)
C
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
C SUBROUTINE   TYPE     DESCRIPTION
C ----------   ------   --------------------------------------------------
C
C BLNKP        User     This routine blanks out the array of text it's passec
C
C CODE         System   This routine reformats data in memory
C
C CNVRT        User     This subroutine is used to convert a packed line of
C                       text to an unpacked form for text processing
C
C LENTH        User     This function returns the length, in characters, of
C                       the array of text it is passed
C
C UNPAK        User     This routine unpacks one word in an A2 format into
C                       2 words each in an A1 format
C
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
C ARRAYS USED IN THIS PROGRAM:
C
C ARRAY  (36)   Array used for blanking out purposes
C CHARS  (2)    Array into which an A2 word is unpacked
C END    (4)    Array holding "+END" characters for END of FILE
```

```
C NAME   (3)     Array to hold the input file name
C RUF    (4)     Array holding "*RUF" characters for Removing undetecable
C                faults prior to *END
C
C VARIABLES USED IN THIS PROGRAM:
C
C COUNT          Holds the count of number of records written to output tapes
C LINLEN         Holds the length in characters of the unpacked input record
C TAPENO         Holds the current tape drive no. for output
C TXTLEN         Holds the length in characters/2 (packed) of input records
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
C GLOBAL INITIALIZATION:
C
      IMPLICIT INTEGER (A - Z)
      DIMENSION IDCB (144), INTMP (18), INTXT (36), OUTXT (36)
      DIMENSION CHARS (2), ARRAY (36), NAME (3), END (4), RUF (4)
      DATA END/1H*,1HE,1HN,1HD/
      DATA RUF/1H*,1HR,1HU,1HF/
      TXTLEN = 18
      LINLEN = 36
      COUNT = 0
      TAPENO = 4
C
CC
CCC Prompt to get the input file name from the user
CC
C
      WRITE (1,1)
1     FORMAT(" ENTER INPUT FILE NAME:")
      READ (1,2)NAME
2     FORMAT(3A2)
C
CC
CCC Open the input file name. if an open error. flag it and stop
CC
C
      CALL OPEN (IDCB, IERR. NAME)
      IF (IERR .LT. 0) GO TO 9000
C
CC
CCC Blank out the current line of text
CC
C
10    CALL BLNKP (INTMP, TXTLEN)
C
CC
CCC Read the next line of input text
CC
C
      CALL READF (IDCB  IERR. INTMP)
      IF (IERR .LT. 0) GO TO 9000
C
CC
CCC Check to see if the line has "EF" in it, indicative of EOF
CC
C
      IF (INTMP (1) .EQ. 2HEF) GO TO 120
```

```
C
CC
CCC Now, while the line doesn't start with blanks, repeat reading lines of
CC  input until it does (lines that don't start with blanks cant have faults
C
      IF (INTMP (1) .NE. 2H  ) GO TO 10
C
CC
CCC Now unpack the line so we can   i.t character positions more easily
CC
C
          CALL CNVRT (INTMP, IN XT, TXTLEN, LINLEN)
C
CC
CCC If the line has an X (or X s) discard it
CC
C
          IF (INTXT (10) .EQ. 1HX) GO TO 115
C
CC
CCC Get the length of input text in characters...
CC
C
          LEN = LENTH (INTXT)
C
CC
CCC Now create the output text record
CC
C
          OUTXT (1) = 1H*
          OUTXT (2) = 1HD
          OUTXT (3) = 1HN
          OUTXT (4) = 1HD
          OUTXT (5) = 1H
          DO 100 J = 6, (LEN - 14) + 7
             OUTXT (J) = INTXT (J + 8)
100       CONTINUE
          OUTXT ( (LEN - 14) + 7) = 1H$
C
CC
CCC Write out the output text record to the current output tape drive
CC
C
          WRITE (TAPENO,110) OUTXT
110       FORMAT(36A1)
C
CC
CCC Increment record counter, and if greater than 1200, switch to drive 5
CC
C
          COUNT = COUNT + 1
          IF (COUNT .GE. 1200) TAPENO = 5
C
CC
CCC Now blank out input and output records preparatory to reading in another
CC
C
115       CALL BLNKR (OUTXT, LINLEN)
          CALL BLNKR (INTXT, LINLEN)
          GO TO 10
```

```
0179  C
0180  CC
0181  CCC Now, write out the *RUF, and the *END at the end of the output file
0182  CC
0183  C
0184  120    WRITE (TAPENO,125) RUF,END
0185  125    FORMAT(4A1,/,4A1)
0186         CALL CLOSE (IDCB)
0187         STOP
0188  9000   WRITE (1,9010)IERR
0189  9010   FORMAT(" ERROR ON DATA FILE ERROR IS # ",I6)
0190         STOP
0191         END
0192         SUBROUTINE BLNKP (ARRAY, LENGTH)
0193  C
0194  CC
0195  CCC This subroutine blanks out ARRAY dimensioned to LENGTH that it's passed
0196  CC
0197  C
0198         IMPLICIT INTEGER (A - Z)
0199         DIMENSION ARRAY (LENGTH)
0200         DO 10 I = 1, LENGTH
0201            IF (LENGTH .EQ. 18) ARRAY (I) = 2H
0202            IF (LENGTH .EQ. 36) ARRAY (I) = 1H
0203  10     CONTINUE
0204         RETURN
0205         END
0206         FUNCTION LENTH (INTXT)
0207  C
0208  CC
0209  CCC This function returns the number of characters in the string INTXT
0210  CC
0211  C
0212         IMPLICIT INTEGER (A - Z)
0213         DIMENSION INTXT (36)
0214         I = 36
0215  5      IF (INTXT (I) .NE. 1H ) GO TO 20
0216            I = I - 1
0217            IF (I .NE. 1) GO TO 5
0218               LENTH = 1
0219               RETURN
0220  20     LENTH = I
0221         RETURN
0222         END
0223         SUBROUTINE UNPAK (PAK, CHARS)
0224  C
0225  CC
0226  CCC This subroutine Unpacks 2 characters in PAK (A2) into 2A1's in array CHAR
0227  CC
0228  C
0229         IMPLICIT INTEGER (A - Z)
0230         DIMENSION CHARS (2)
0231         CALL CODE
0232         READ (PAK, 5) CHARS
0233  5      FORMAT(2A1)
0234         RETURN
0235         END
0236         SUBROUTINE CNVRT (INTMP  INTXT, TXTLEN, LINLEN)
0237  C
0238  CC
```

```
0239  CCC This subroutine converts packed text in INTMP(TXTLEN) into unpacked
0240  CC  text in INTXT(LINLEN)
0241  C
0242        IMPLICIT INTEGER (A - Z)
0243        DIMENSION CHARS (2), TMP (2)
0244        DIMENSION INTMP (TXTLEN), INTXT (LINLEN)
0245        PTR = 0
0246        DO 10 I = 1, TXTLEN
0247           PTR = PTR + 1
0248           CALL UNPAK (INTMP (I), TMP)
0249           INTXT (PTR) = TMP (1)
0250           PTR = PTR + 1
0251           INTXT (PTR) = TMP (2)
0252  10    CONTINUE
0253        RETURN
0254        END
0255        END$
```

SECTION A.4

INIT/NOOP

APPENDIX A – Software
Section A.4 – INIT/NOOP

A.4.1 – INIT LISTING

```
INIT   T=00004 IS ON CR00018 USING 00017 BLKS R=0000

0001  FTN4,L
0002        PROGRAM INIT
0003        IMPLICIT INTEGER (A-Z)
0004        REAL TIME, VDH,VCH,VDL,VCL
0005        DIMENSION IDRIV4 (3), IDRIV7 (2), IDRIV8 (4), IDRTOG (2)
0006        DIMENSION IDRV12 (2), IHI7 (2), IHITOG (2), ILOTOG (2)
0007        DIMENSION IHI8 (2), IHI4 (2), ILO7 (2), ILO4 (2), ILO8 (3)
0008        DIMENSION ILO12 (2), IPINS (150)  IERR (4), IBUF (5), INAM (3)
0009        DATA IDRIV4/2,47,46/
0010        DATA IDRIV7/1,104/
0011        DATA IDRTOG/1,105/
0012        DATA IDRIV8/3,106,109,107/
0013        DATA IDRV12/1,177/
0014        DATA IHI4/1,46/
0015        DATA ILO4/1,47/
0016        DATA ILO7/1,104/
0017        DATA IHI8/1,106/
0018        DATA ILO8/3,109,107/
0019        DATA ILO12/1,177/
0020        DATA IHITOG/1,105/
0021        DATA ILOTOG/1,105/
0022        LDTU = 30
0023        MODE = 1
0024        ICODE = 23
0025        IBUF (1) = 2H:T
0026        IBUF (2) = 2HR,
0027        IBUF (3) = 2HP0
0028        IBUF (4) = 2HW0
0029        IBUF (5) = 2HH
0030        INAM (1) = 2HFM
0031        INAM (2) = 2HGR
0032        INAM (3) = 2H
0033        CALL EXEC (ICODE, INAM  0,0,0,0,0,IBUF,5)
0034        CALL XINIT (LDTU, IERR, MODE, IPINS)
0035        IF (IERR .NE. 0) GO TO 9000
0036        TIME = 1E-2
0037        VDH = 5E0
0038        VDL = 0E0
0039        VCH = 4E0
0040        VCL = 1E0
0041        NSET = 1
0042        CALL XTREF (LDTU, IERR, NSET, VDH, VDL, VCH, VCL)
0043        IF (IERR .NE. 0) GO TO 9000
0044        NCRD = 4
0045        CALL XWSET (LDTU, IERR, NSET, NCRD)
0046        IF (IERR .NE. 0) GO TO 9000
0047        CALL XTDRV (IERR, MODE, IDRIV4, IPINS)
0048        IF (IERR .NE. 0) GO TO 9000
0049        CALL XETHI (IERR, IHI4, IPINS)
0050        IF (IERR .NE. 0) GO TO 9000
0051        CALL XETLO (IERR, ILO4, IPINS)
0052        IF (IERR .NE. 0) GO TO 9000
0053        CALL XTEST (IERR, ISTAT, MODE, IPINS)
0054        IF (IERR .NE. 0) GO TO 9000
0055  C      CALL PAUSR
0056        NCRD = 7
0057        CALL XWSET (LDTU, IERR, NSET, NCRD)
0058        IF (IERR .NE. 0) GO TO 9000
```

```
0059          CALL XTDRV (IERR, MODE, IDRIV7, IPINS)
0060          IF (IERR .NE. 0) GO TO 9000
0061          CALL XTDRV (IERR, MODE, IDRTOG, IPINS)
0062          IF (IERR .NE. 0) GO TO 9000
0063          CALL XETLO (IERR, IL07, IPINS)
0064          IF (IERR .NE. 0) GO TO 9000
0065          CALL XETHI (IERR, IHITOG, IPINS)
0066          IF (IERR .NE. 0) GO TO 9000
0067          CALL XTEST (IERR, ISTAT, MODE, IPINS)
0068          IF (IERR .NE. 0) GO TO 9000
0069    C      CALL PAUSR
0070          NCRD = 8
0071          CALL XWSET (LDTU, IERR, NSET, NCRD)
0072          IF (IERR .NE. 0) GO TO 9000
0073          CALL XTDRV (IERR, MODE, IDRIV8, IPINS)
0074          IF (IERR .NE. 0) GO TO 9000
0075          CALL XETHI (IERR, IHI8, IPINS)
0076          IF (IERR .NE. 0) GO TO 9000
0077          CALL XETLO (IERR, IL08, IPINS)
0078          IF (IERR .NE. 0) GO TO 9000
0079          CALL XTEST (IERR, ISTAT, MODE, IPINS)
0080          IF (IERR .NE. 0) GO TO 9000
0081    C      CALL PAUSR
0082          NCRD = 12
0083          CALL XWSET (LDTU, IERR, NSET, NCRD)
0084          IF (IERR .NE. 0) GO TO 9000
0085          CALL XTDRV (IERR, MODE, IDRV12, IPINS)
0086          IF (IERR .NE. 0) GO TO 9000
0087          CALL XETLO (IERR, IL012, IPINS)
0088          IF (IERR .NE. 0) GO TO 9000
0089          CALL XTEST (IERR, ISTAT, MODE, IPINS)
0090          IF (IERR .NE. 0) GO TO 9000
0091    C
0092    CC
0093    CCC NOW SET ADAPTER NO. 105 LOW
0094    CC
0095    C
0096    C      CALL PAUSR
0097          NCRD = 7
0098          CALL XWSET (LDTU, IERR, NSET, NCRD)
0099          IF (IERR .NE. 0) GO TO 9000
0100          CALL XTDRV (IERR, MODE, IDRTOG, IPINS)
0101          IF (IERR .NE. 0) GO TO 9000
0102          CALL XETLO (IERR, ILOTOG, IPINS)
0103          IF (IERR .NE. 0) GO TO 9000
0104          CALL XTEST (IERR, ISTAT, MODE, IPINS)
0105          IF (IERR .NE. 0) GO TO 9000
0106    C
0107    CC
0108    CCC NOW SET IT HIGH AGAIN
0109    CC
0110    C
0111    C      CALL PAUSR
0112          CALL XETHI (IERR, IHITOG, IPINS)
0113          IF (IERR .NE. 0) GO TO 9000
0114          CALL XTEST (IERR, ISTAT, MODE, IPINS)
0115          IF (IERR .NE. 0) GO TO 9000
0116          STOP
0117    9000  CONTINUE
0118          WRITE (1, 9010)IERR
```

```
0119  9010  FORMAT(" IERR IS:",I2,1X,3A2)
0120        STOP
0121        END
0122        SUBROUTINE PAUSR
0123        IMPLICIT INTEGER (A-Z)
0124        WRITE (1,10)
0125  10    FORMAT(" PAUSE <CR>")
0126        READ (1,20)IDUM
0127  20    FORMAT(I3)
0128        RETURN
0129        END
0130        END$
```

A.4.2 – NOOP LISTING

```
NOOP   T=00004 IS ON CR00018 USING 00014 BLKS R=0000

FTN4,L
      PROGRAM NOOP
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
C     PROGRAM:         NOOP
C
C     PURPOSE:         To initialize the hardware on the 1646178 board
C                      prior to testing.
C
C     PROGRAMMER:      DAVID S. WAGNER     - BLDG. 688/T125 (213) 802-4190
C
C     DATE:            JULY 2, 1980
C
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C SUBROUTINE  TYPE    DESCRIPTION
C -----------------------------------------------------------------------
C
C  EXEC        System  Used to execute a transfer filed prior to
C                      executing this program
C
C  XINIT       System  Used to initialize the pin programming array. See
C                      also DTS-70 Programmers Reference Manual pg 7-26
C  XDLY        System  Used to set Delay time interval for Driver/Comparator
C                      relays.  See DTS-70 P.R.M pg. 7-5
C  XTREF       System  Used to set the voltage reference levels for D/C
C                      See DTS-70 P.R.M. pg. 7-6
C  XWSET       System  Used to switch reference levels to D/C cards. See
C                      DTS-70 P.R.M. pg. 7-7
C  XTDRV       System  Used to enable the Driver on a specified list of pins
C                      See P.R.M. pg. 7-12
C  XTCMP       System  Used to enable the Comparator on specified pin list
C                      See P.R.M. pg. 7-13
C  XETHI       System  Used to define pins desired to be set hi. See DTS-70
C                      P.R.M. pg. 7-14
C  XETLO       System  Used to define pins desired to be set lo. See DTS-70
C                      P.R.M. pg. 7-15
C  XTEST       System  Used to actually do the digital test based on pin
C                      state array   See P.R.M. pg. 7-16
C  XSERN       System  Used to report any and all subroutine errors. See
C                      DTS-70  Prog. Ref. Man
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
C ARRAYS USED IN THIS PROGRAM:
C
C IDRIV4  (10)    Array to hold pin number(s) on D/C card no. 4
C IDRIV5  (8)     Array to hold pin number(s) on D/C card no. 5
C IERR    (4)     Error parameter holding array
C IDRIV6  (9)     Array to hold pin number(s) on D/C card no. 6
C IDRIV7  (4)     Array to hold pin number(s) on D/C card no. 7
C IDRIV8  (4)     Array to hold pin number(s) on D/C card no. 8
C IDRIVT  (2)     Array to hold pin number that is to be toggled
C IPINS   (150)   Pin state array needed to do digital tests
C IHI4    (4)     Array to hold pin numbers on D/C card 4 to set hi
```

```
0059  C IHI5     (7)     Array to hold pin numbers on D/C card 5 to set hi
0060  C ILO5     (2)     Array to ho d pin number on D/C card 5 to set lo
0061  C IHI6     (9)     Array to hold pin numbers on D/C card 6 to set hi
0062  C IHI7     (3)     Array to hold pin numbers on D/C card 7 to set hi
0063  C ILO7     (2)     Array to hold pin numbers on D/C card 7 to set lo
0064  C IHI8     (3)     Array to hold pin numbers on D/C card 8 to set hi
0065  C ILO8     (2)     Array to hold pin numbers on D/C card 8 to set lo
0066  C IHITOG   (2)     Array to hold pin number to toggle hi
0067  C ILOTOG   (2)     Array to hold pin number to toggle lo
0068  C IBUF     (5)     Array holds name of transfer file to execute
0069  C INAM     (3)     Array holds name of system prog. to execute (FMGR)
0070  C
0071  C
0072  C VARIABLES USED IN THIS PROGRAM:
0073  C
0074  C LDTU             Holds the logical unit no. for the Digital Test Unit
0075  C MODE             Defines the mode of current test (See DTS-70 PRM sec. 7)
0076  C NCRD             Holds the D/C card no. of card currently being set
0077  C NSET             Holds the reference set number of current set
0078  C TIME             Holds the D/C relay delay time in milliseconds
0079  C VCH              Defines the voltage comparator High
0080  C VCL              Defines the voltage comparator Lo
0081  C VDH              Defines the voltage driver High
0082  C VDL              Defines the voltage driver Lo
0083  C
0084  C GLOBAL INITIALIZATION:
0085  C
0086  C
0087        DIMENSION IDRIV4 (10), IDRIV5 (8), IDRIV6 (9), IDRIV7 (4)
0088        DIMENSION IDRIV8 (4), IDRIVT (2), IHITOG (2), ILOTOG (2)
0089        DIMENSION IPINS (150), IERR (4), IHI4 (2), IHI5 (7), IHI6 (9)
0090        DIMENSION IHI7 (3), IHI8 (3), IL04 (9), IL05 (2), IL08 (2)
0091        DIMENSION IBUF (5), INAM (3), IL07 (2)
0092        DATA IDRIV4/9,46,47,48,50,52,54,56,58,60/
0093        DATA IDRIV5/7,64,66,68,70,72,74,62/
0094        DATA IDRIV6/8,76,78,80,82,84,86,88,90/
0095        DATA IDRIV7/3,92,94,104/
0096        DATA IDRIV8/3,106,107,108/
0097        DATA IDRIVT/1,105/
0098        DATA IHITOG/1,105/
0099        DATA ILOTOG/1,105/
0100        DATA IHI4/1,46/
0101        DATA IHI5/6,64,66,68,70,72,74/
0102        DATA IHI6/8,76,78,80,82,84,86,88,90/
0103        DATA IHI7/2,92,94/
0104        DATA IHI8/2,106,108/
0105        DATA IL04/8,47,48,50,52,54,56,58,60/
0106        DATA IL05/1,62/
0107        DATA IL07/1,104/
0108        DATA IL08/1,107/
0109        LDTU = 30
0110        MODE = 1
0111        ICODE = 23
0112  C
0113  CC
0114  CCC The following code executes FMGR which runs the transfer file POWON
0115  CC
0116  C
0117        IBUF (1) = 2H:T
0118        IBUF (2) = 2HR,
```

```
0119          IBUF (3) = 2HPO
0120          IBUF (4) = 2HWO
0121          IBUF (5) = 2HN
0122          INAM (1) = 2HFM
0123          INAM (2) = 2HGR
0124          INAM (3) = 2H
0125          CALL EXEC (ICODE,INAM,0,0,0,0,0,IBUF,5)
0126    C
0127    CC
0128    CCC Now initialize the DTU software
0129    CC
0130    C
0131          CALL XINIT (LDTU, IERR, MODE, IPINS)
0132          IF (IERR .NE. 0) GO TO 9000
0133    C
0134    CC
0135    CCC Now define our delay time as 10 mSec
0136    CC
0137    C
0138          TIME = 1E-2
0139          CALL XDLY (LDTU, IERR, TIME)
0140          IF (IERR .NE. 0) GO TO 9000
0141    C
0142    CC
0143    CCC Now initialize our voltage references
0144    CC
0145    C
0146          VDH = 5E0
0147          VDL = 0E0
0148          VCH = 4E0
0149          VCL = 1E0
0150          NSET = 1
0151          CALL XTREF (LDTU, IERR, NSET, VDH, VDL, VCH, VCL)
0152          IF (IERR .NE. 0) GO TO 9000
0153          NCRD = 4
0154    C
0155    CC
0156    CCC And then switch them to our current reference set
0157    CC
0158    C
0159          CALL XWSET (LDTU, IERR, NSET, NCRD)
0160          IF (IERR .NE. 0) GO TO 9000
0161    C
0162    CC
0163    CCC Now enable the driver on card 4
0164    CC
0165    C
0166          CALL XTDRV (IERR, MODE, IDRIV4, IPINS)
0167          IF (IERR .NE. 0) GO TO 9000
0168    C
0169    CC
0170    CCC and set the pins on that card to their proper states
0171    CC
0172    C
0173          CALL XETHI (IERR, IHI4, IPINS)
0174          IF (IERR .NE. 0) GO TO 9000
0175          CALL XETLO (IERR, ILO4, IPINS)
0176          IF (IERR .NE. 0) GO TO 9000
0177    C
0178    CC
```

```
0179  CCC and now perform the actual setting of those pins
0180  CC
0181  C
0182        CALL XTEST (IERR, ISTAT, MODE, IPINS)
0183        IF (IERR .NE. 0) GO TO 9000
0184  C
0185  CC
0186  CCC now switch to card no. 7 and repeat the process
0187  CC
0188  C
0189        NCRD = 7
0190        CALL XTREF (LDTU, IERR, NSET, VDH, VDL, VCH, VCL)
0191        IF (IERR .NE. 0) GO TO 9000
0192        CALL XWSET (LDTU, IERR, NSET, NCRD)
0193        IF (IERR .NE. 0) GO TO 9000
0194        CALL XTDRV (IERR, MODE, IDRIVT, IPINS)
0195        IF (IERR .NE. 0) GO TO 9000
0196        CALL XETLO (IERR, ILOTOG, IPINS)
0197        IF (IERR .NE. 0) GO TO 9000
0198        CALL XTEST (IERR, ISTAT, MODE, IPINS)
0199        IF (IERR .NE. 0) GO TO 9000
0200  C
0201  CC
0202  CCC now card 5 is up to bat ..
0203  CC
0204  C
0205        NCRD = 5
0206        CALL XTREF (LDTU, IERR, NSET, VDH, VDL, VCH, VCL)
0207        IF (IERR .NE. 0) GO TO 9000
0208        CALL XWSET (LDTU, IERR, NSET, NCRD)
0209        IF (IERR .NE. 0) GO TO 9000
0210        CALL XTDRV (IERR, MODE  IDRIV5, IPINS)
0211        IF (IERR .NE. 0) GO TO 9000
0212        CALL XETHI (IERR, IHI5, IPINS)
0213        IF (IERR .NE. 0) GO TO 9000
0214        CALL XETLO (IERR, ILO5, IPINS)
0215        IF (IERR  NE  0) GO TO 9000
0216        CALL XTEST (IERR, ISTAT, MODE, IPINS)
0217        IF (IERR .NE. 0) GO TO 9000
0218  C
0219  CC
0220  CCC followed by card 6
0221  CC
0222  C
0223        NCRD = 6
0224        CALL XTREF (LDTU, IERR, NSET, VDH, VDL, VCH, VCL)
0225        IF (IERR  NE. 0) GO TO 9000
0226        CALL XWSET (LDTU, IERR, NSET, NCRD)
0227        IF (IERR .NE. 0) GO TO 9000
0228        CALL XTDRV (IERR, MODE, IDRIV6, IPINS)
0229        IF (IERR .NE. 0) GO TO 9000
0230        CALL XETHI (IERR, IHI6, IPINS)
0231        IF (IERR .NE. 0) GO TO 9000
0232        CALL XTEST (IERR, ISTAT, MODE, IPINS)
0233        IF (IERR .NE. 0) GO TO 9000
0234  C
0235  CC
0236  CCC then it is card 7 & turn again
0237  CC
0238  C
```

```
0239          NCRD = 7
0240          CALL XTREF (LDTU, IERR, NSET, VDH, VDL, VCH, VCL)
0241          IF (IERR .NE. 0) GO TO 9000
0242          CALL XWSET (LDTU, IERR, NSET, NCRD)
0243          IF (IERR .NE. 0) GO TO 9000
0244          CALL XTDRV (IERR, MODE, IDRIV7, IPINS)
0245          IF (IERR .NE. 0) GO TO 9000
0246          CALL XETHI (IERR, IHI7, IPINS)
0247          IF (IERR .NE. 0) GO TO 9000
0248          CALL XTEST (IERR, ISTAT, MODE, IPINS)
0249          IF (IERR .NE. 0) GO TO 9000
0250    C
0251    CC
0252    CCC then 8 comes...
0253    CC
0254    C
0255          NCRD = 8
0256          CALL XTREF (LDTU, IERR, NSET, VDH, VDL, VCH, VCL)
0257          IF (IERR .NE. 0) GO TO 9000
0258          CALL XWSET (LDTU, IERR, NSET, NCRD)
0259          IF (IERR .NE. 0) GO TO 9000
0260          CALL XTDRV (IERR, MODE, IDRIV8, IPINS)
0261          IF (IERR .NE. 0) GO TO 9000
0262          CALL XETHI (IERR, IHI8, IPINS)
0263          IF (IERR .NE. 0) GO TO 9000
0264          CALL XETLO (IERR, ILO8, IPINS)
0265          IF (IERR .NE. 0) GO TO 9000
0266          CALL XTEST (IERR, ISTAT, MODE, IPINS)
0267          IF (IERR .NE. 0) GO TO 9000
0268    C
0269    CC
0270    CCC NOW TOGGLE PIN 105
0271    CC
0272    C
0273          NCRD = 7
0274          CALL XTREF (LDTU, IERR, NSET, VDH, VDL, VCH, VCL)
0275          IF (IERR .NE. 0) GO TO 9000
0276          CALL XWSET (LDTU, IERR, NSET, NCRD)
0277          IF (IERR .NE. 0) GO TO 9000
0278          CALL XTDRV (IERR, MODE, IDRIVT, IPINS)
0279          IF (IERR .NE. 0) GO TO 9000
0280          CALL XETLO (IERR, ILOTOG, IPINS)
0281          IF (IERR .NE. 0) GO TO 9000
0282          CALL XTEST (IERR, ISTAT, MODE, IPINS)
0283          IF (IERR .NE. 0) GO TO 9000
0284    C
0285    CC
0286    CCC NOW SET IT HIGH AGAIN
0287    CC
0288    C
0289          CALL XTREF (LDTU  IERR, NSET, VDH, VDL, VCH, VCL)
0290          IF (IERR .NE. 0) GO TO 9000
0291          CALL XWSET (LDTU, IERR, NSET, NCRD)
0292          IF (IERR .NE. 0) GO TO 9000
0293          CALL XTDRV (IERR  MODE, IDRIVT, IPINS)
0294          IF (IERR .NE. 0) GO TO 9000
0295          CALL XETHI (IERR, IHITOG, IPINS)
0296          IF (IERR .NE. 0) GO TO 9000
0297          CALL XTEST (IERR, ISTAT, MODE, IPINS)
0298          IF (IERR .NE. 0) GO TO 9000
```

```
0299  C     WRITE (1,10)
0300  10    FORMAT(" FINISHED TOGGLING PIN 105",/," PAUSING HERE UNTIL <CR>")
0301  C     READ (1,20)IDUMMY
0302  20    FORMAT(I3)
0303        STOP
0304  9000  CONTINUE
0305  C
0306  CC
0307  CCC THIS SECTION HANDLES ERRORS ON DTS70 SUBROUTINE CALLS...
0308  CC
0309  C
0310        WRITE (1, 9010)IERR
0311  9010  FORMAT(" IERR IS: ",I2,1X,3A2)
0312        CALL XSERN (LDTU, IERR(1))
0313        STOP
0314        END
0315        END$
```

SECTION A.5

DFISML

A.5.1 – SUPPORT MAINTENANCE SCHEMA

```
DFI          DTS-70          SUPPORT MAINTENANCE
             SCHEMA
$ CONTROL:TABLE,FIELD;
  Begin Data Base:DFISML:10:19;        ((Image Data Base NAME))
      ((Security Information))
LEVELS:
      1      Leader                    ((Lowest Level and Code Pass Word))
      3      Senior
      5      Admin                     (( Highest Level and Code Pass Word))
      (( Item Definition))
ITEMS:
      PNUMB, x12 (1,3);                ((Part Number))
      SANUM, x12 (1,3);                ((Serial or Assembly Number))
      SACTV, x20 (1,3);                ((Support Activity))
                                       ((Software-Update; Generation))
                                       ((Hardware-Replace; Repair))
      STIME, R2 (1,3);                 (( Support Start Time))
      FTIME, R2 (1,3);                 (( Support Finish Time))
      ELTIME, R2 (3,5);                ((Support Elapsed Time))
      LCHARG, x10 (3,5);               ((Labor $ Charge))
      MCHARG, x10 (3,5);               ((Material $ Charge))
      DATE    , x12 (1,3);             ((Support-Maintenance Date))
    ((Set Definition))
SETS:
NAME: PART: : 19, A; ((Part Number File,Automatic Master))
ENTRY: PNUMB  (1);
CAPACITY: 23;
NAME: DATEF: : 19, A;                  ((Service Date File, Automatic Master))
ENTRY: DATE (1);
CAPACITY: 67 ;
NAME: SMFILE : : 19, D;                ((Support-Maint. Detail File))
ENTRY: PNUMB (PART), SANUM, SACTV, STIME, FTIME, ELTIME, LCHARG,
       MCHARG, DATE (DATEF);
CAPACITY: 161 ;
END.
```

A.5.2 – SUPPORT MAINTENANCE REPORT SMRPT

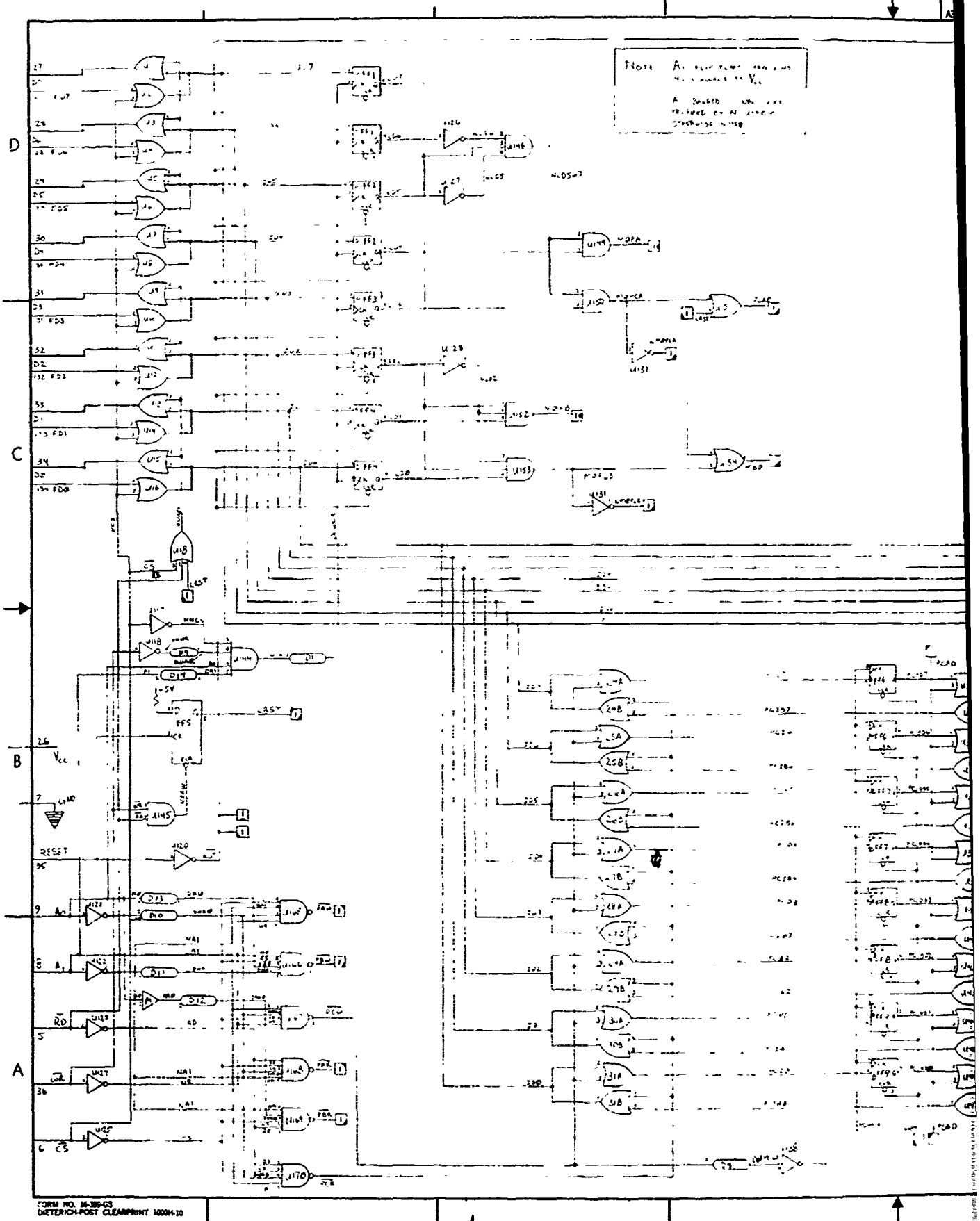
```
DFI       DTS-70          SUPPORT MAINTENANCE
QUERY-REPORT
        DFISML:10:19,ADMIN;
        Report NAME = SMRPT          (( SM Report Print Procedure File))
        H1, "DFI Support Maintenance Report", 81;
        H1, "Page", 107;
        H1, Page No, 111;
        H2, "DTS-70 System", 73;
        H3, "HAC---- Org-12-42-50----", 82, SPACE A2, E1;
        H4, "---- Part ------------ Serial/Asmby -------, Support-Activity-----Suppo rt
             ----Labor--------------Material---------Date---", 119;
        H5, "----Number--------------Number-------------------Time-Hr.
             ----$-----------------$-----------------", 119, Space A2
        S2, Date;
        S1, PNUMB;
        D1, Part, 26:
        D1, SANUM, 42;
        D1, SACTV, 66;
        D1, ELTIME, 79;
        D1, LCHARG, 93;
        D1, MCHARG, 107;
        D1, DATE, 119;
        G2, DATE, 26, E1;
        G1, PNUMB, 26;
        T2, "Date _ Sub-Totals", 66;
        T2, ELTIME, 79; Add;
        T2, LCHARG, 93, Add;
        T2, MCHARG, 107, Space B2, Add;
        TF, "Report Totals", 66;
        TF, ELTIME, 79, Add;
        TF, LCHARG, 93, Add;
        TF, MCHARG, 107, Space B2, Add;
        E1, "XX/XX/XX";
        END; "XX_XXX_XXXX_";
```

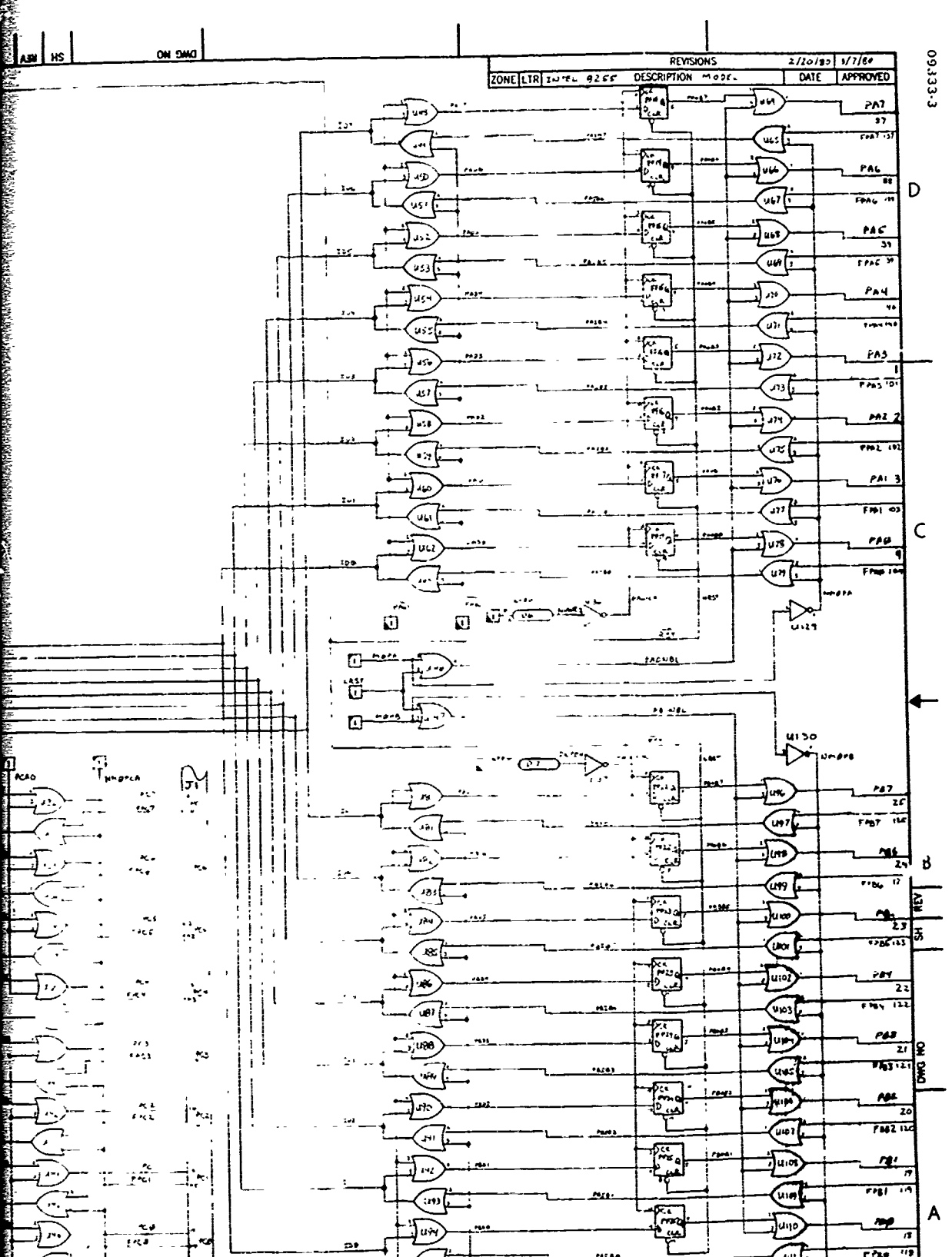
APPENDIX B – SCHEMATICS

SECTION B.1

PN 1635972 CIRCUIT BOARD MODEL

APPENDIX B – Schematics
Section B. 1 – PN 1635972 Circuit Board Model
B. 1. 1 – 8 CHANNEL, 4 PORT PROGRAMMABLE PERIPHERAL INTERFACE, 8255




REVISIONS
ZONE LTR INTEL 8255 DESCRIPTION MODEL
DATE APPROVED
2/20/80 1/7/80
09.333-3
INTEL 8255
PERIPHERAL PROCESSOR
CONTRACT: DFI - MMT
SIZE D
FSCM NO
Rev. C
DWG NO. DESIGN
PAUL CHEY
DAVID WAGNER
SCALE None
SHEET 1 of 1

APPENDIX B – Schematics
Section B.1 – PN 1635972 Circuit Board Model

B.1.2 – 4 BIT BI-DIRECTIONAL BUS DRIVER, 8216

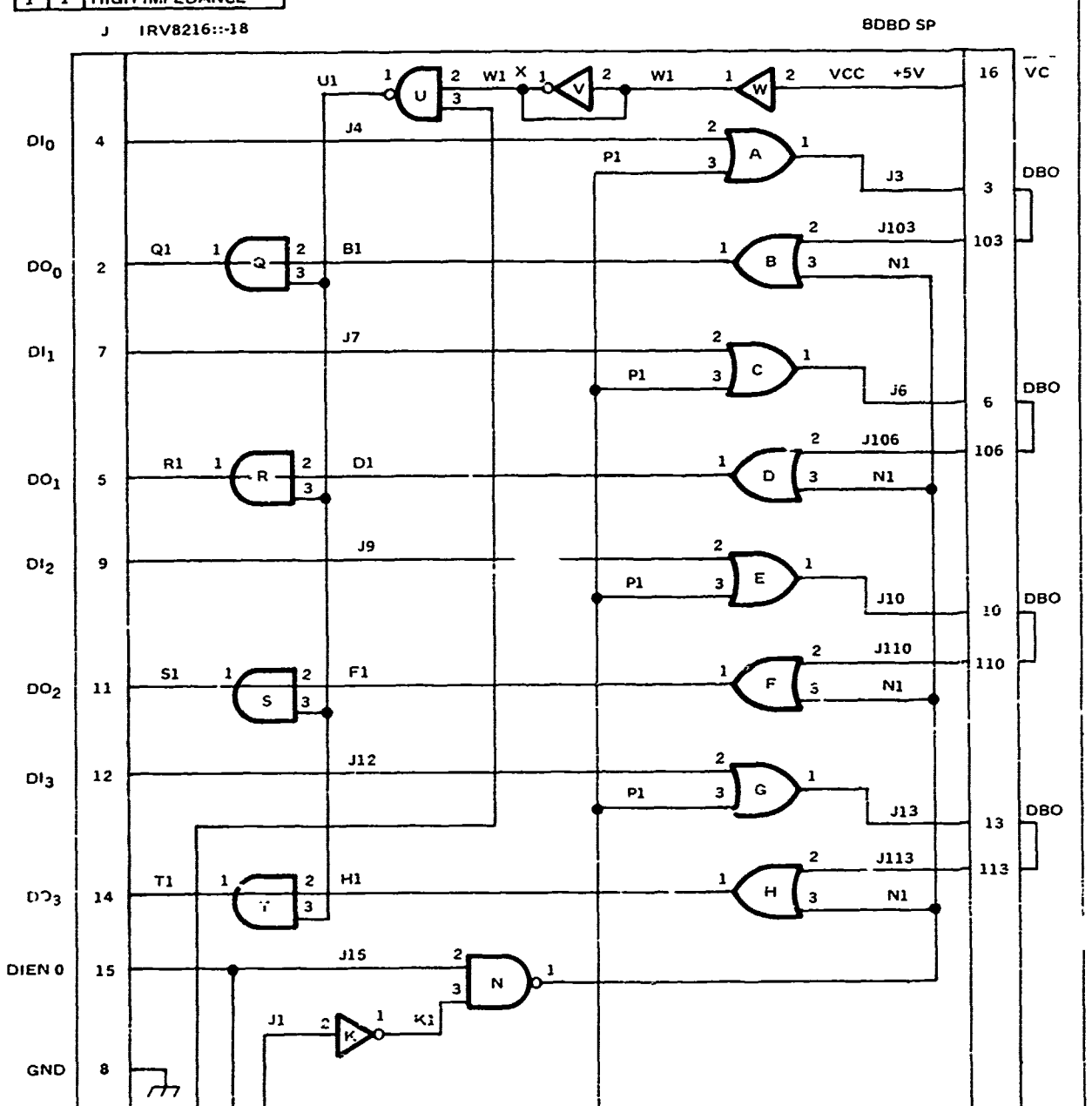
0	0	DI, DB
1	0	DB → DO
0		HIGH IMPEDANCE
1	1	HIGH IMPEDANCE

ONLY U14, U15, U10, U9

4 BIT BI-DIRECTIONAL BUS DRIVER (BDBD) 8216

J IRV8216::-18

BDBD SP



09333-1

B.1.3 – 4 BIT BI-DIRECTIONAL BUS DRIVER, 8216A

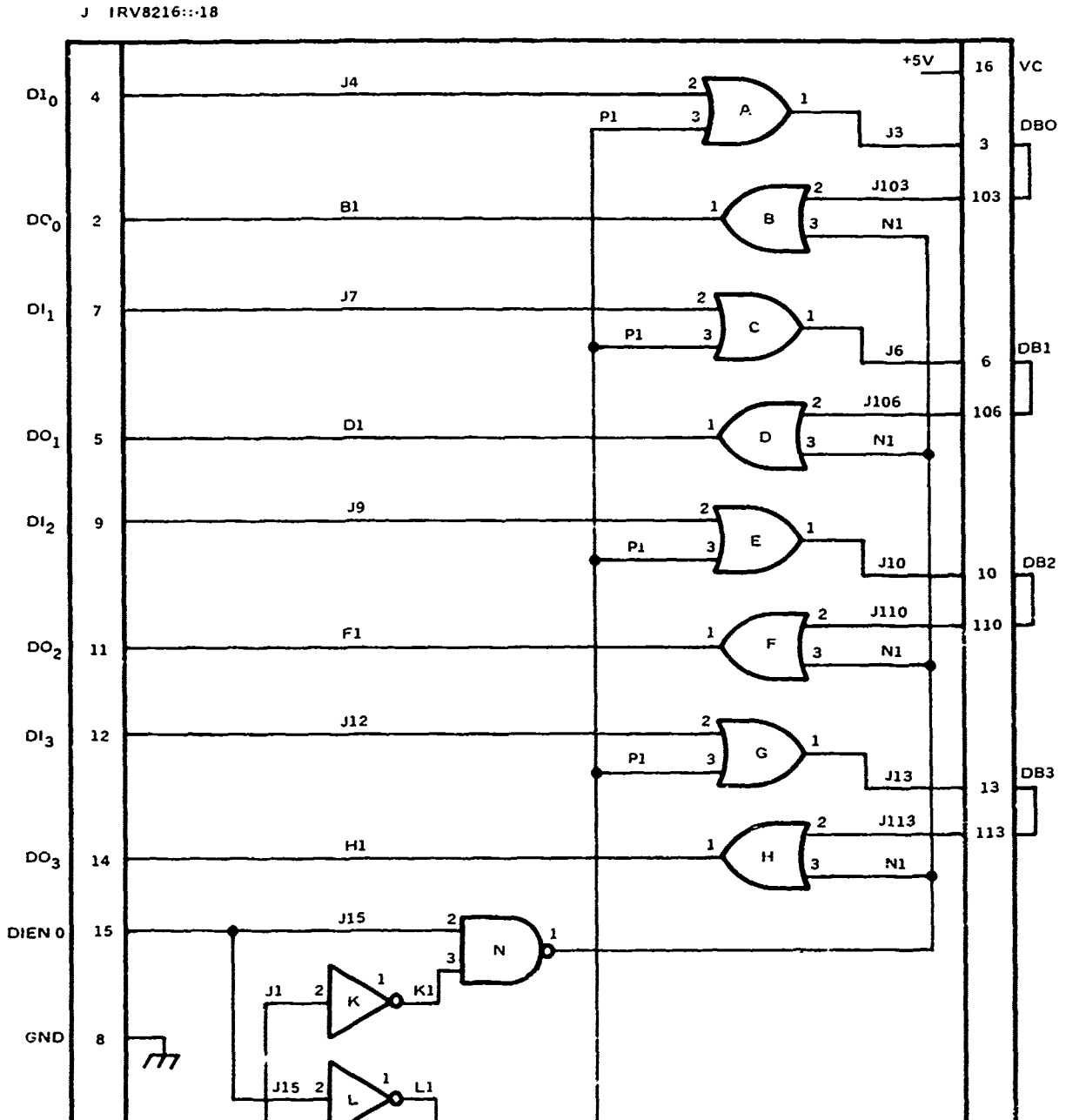
0	0	DI, DB
1	0	DB → DO
0		HIGH IMPEDANCE
1	1	HIGH IMPEDANCE

J IRV8216:::-18

ONLY U14, U15, &10, U9

4 BIT BI-Directional Bus Driver (BDBD) 8216A

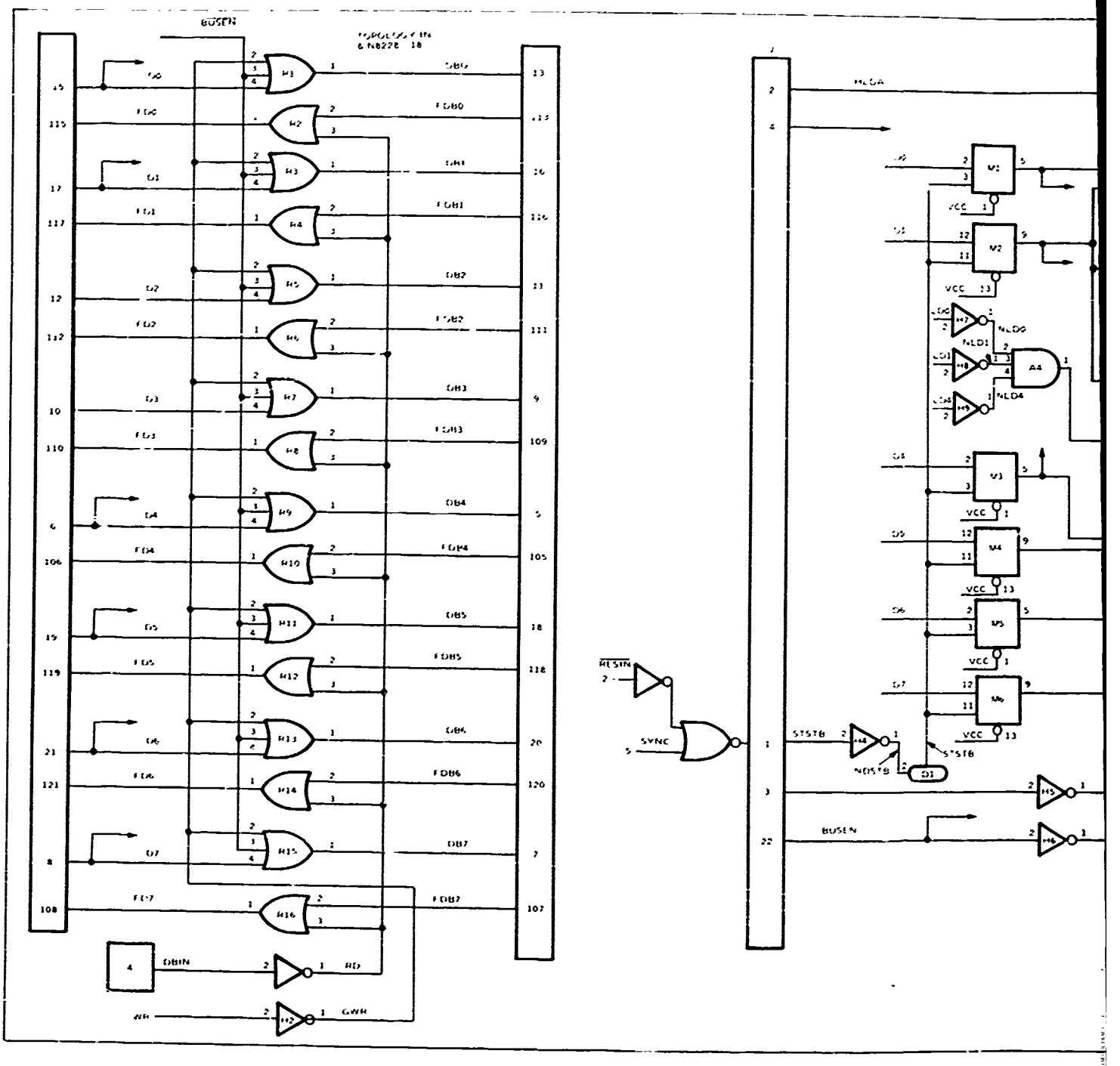
09333-25

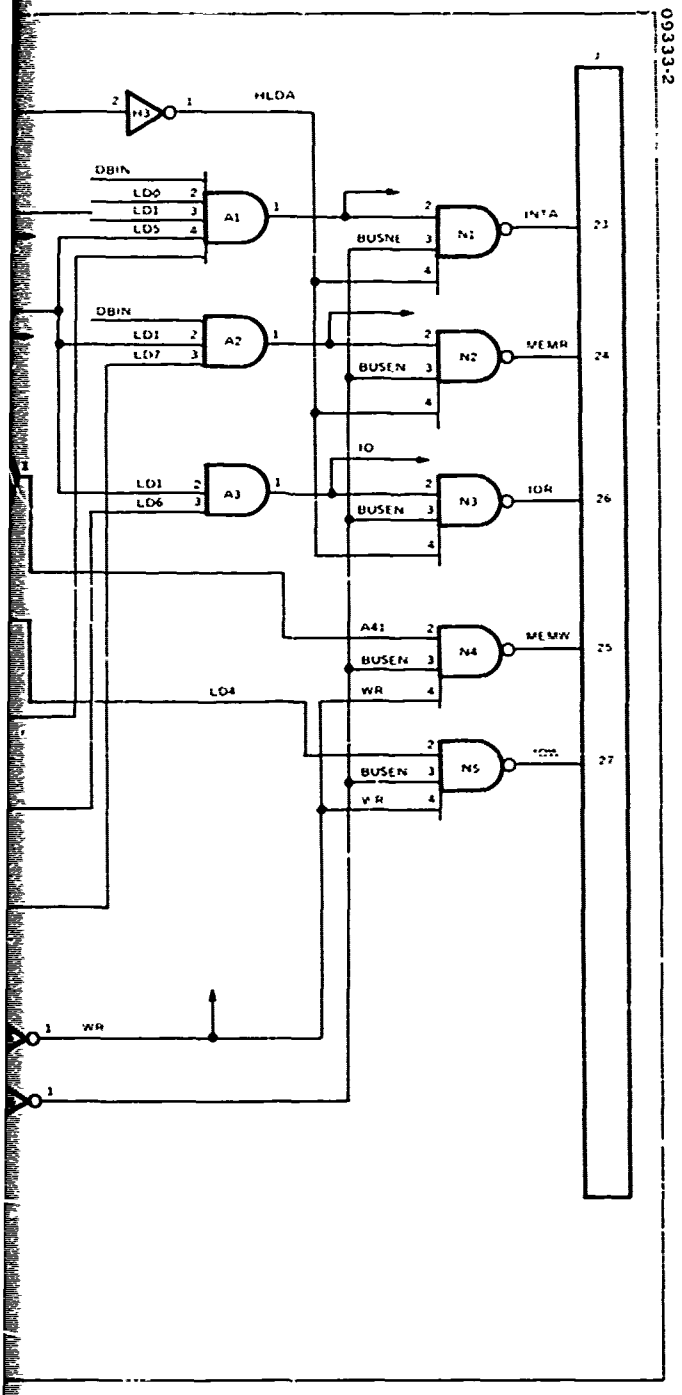


FICTICIOUS PINS: 103, 106, 110, 113

APPENDIX B – Schematics
Section B.1 – PN 1635972 Circuit Board Model

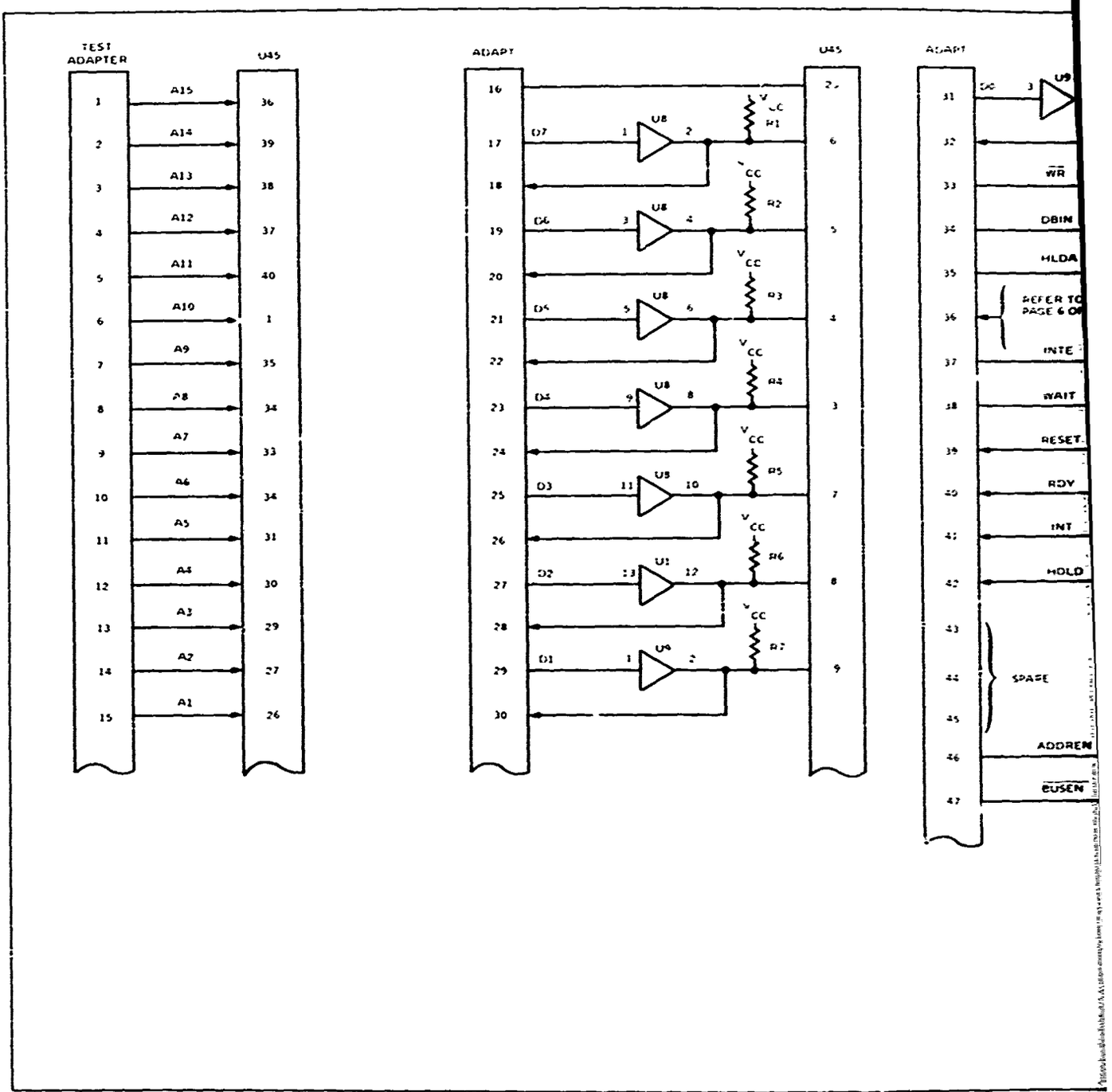
B.1.4 – SYSTEM CONTROLLER AND BUS DRIVER, 8228

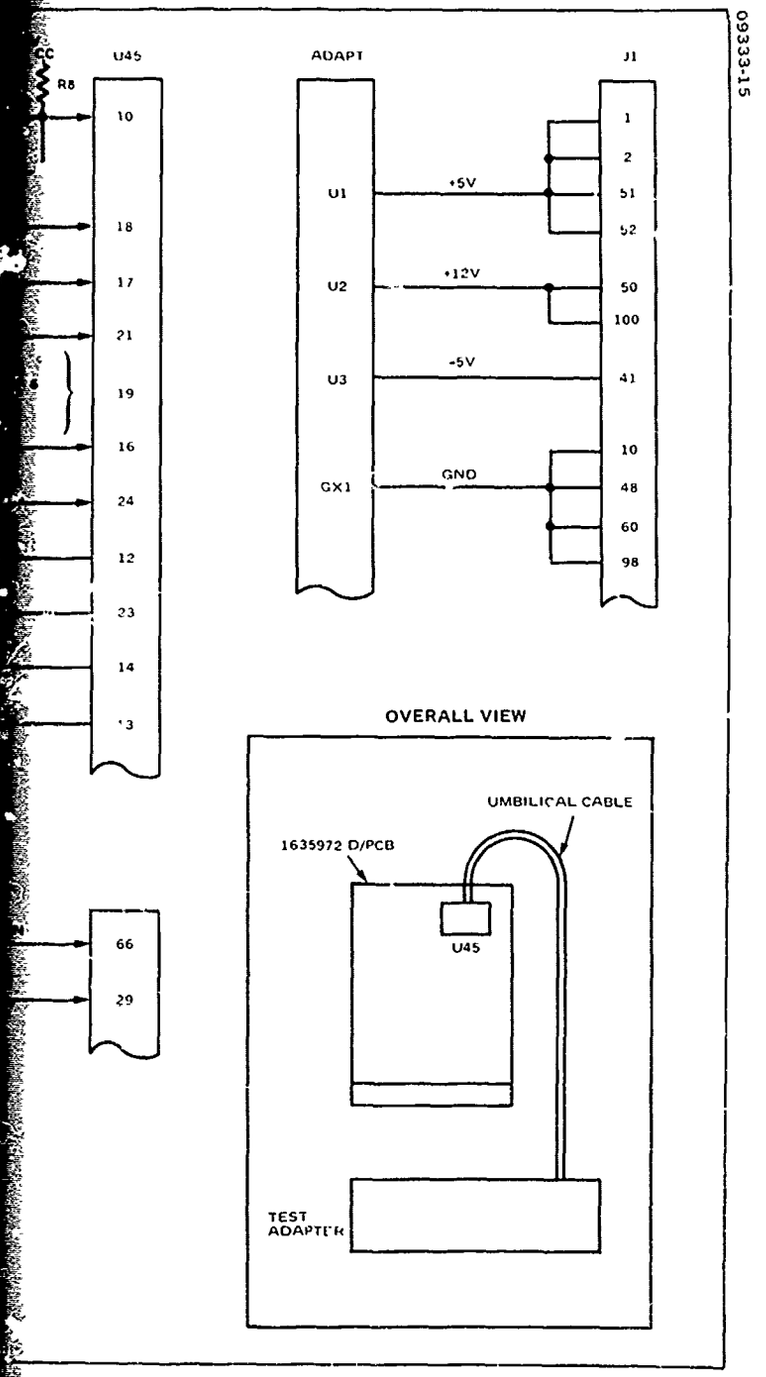



09333-2
HLDA
DBIN
LD0
LD1
LD5
A1
BUSNE
N1
INTA
23
DBIN
LD1
LD7
A2
BUSEN
N2
MEMR
24
IO
LD1
LD6
A3
BUSEN
N3
IOR
26
A41
BUSEN
WR
N4
MEMW
25
LD4
BUSEN
N5
27
WR

APPENDIX B – Schematics
Section B. 2 – PN 1635972 Circuit Board Test Adapter

B. 2. 1 – 8080 A/B UMBILICAL CABLE; PART 1 OF 6

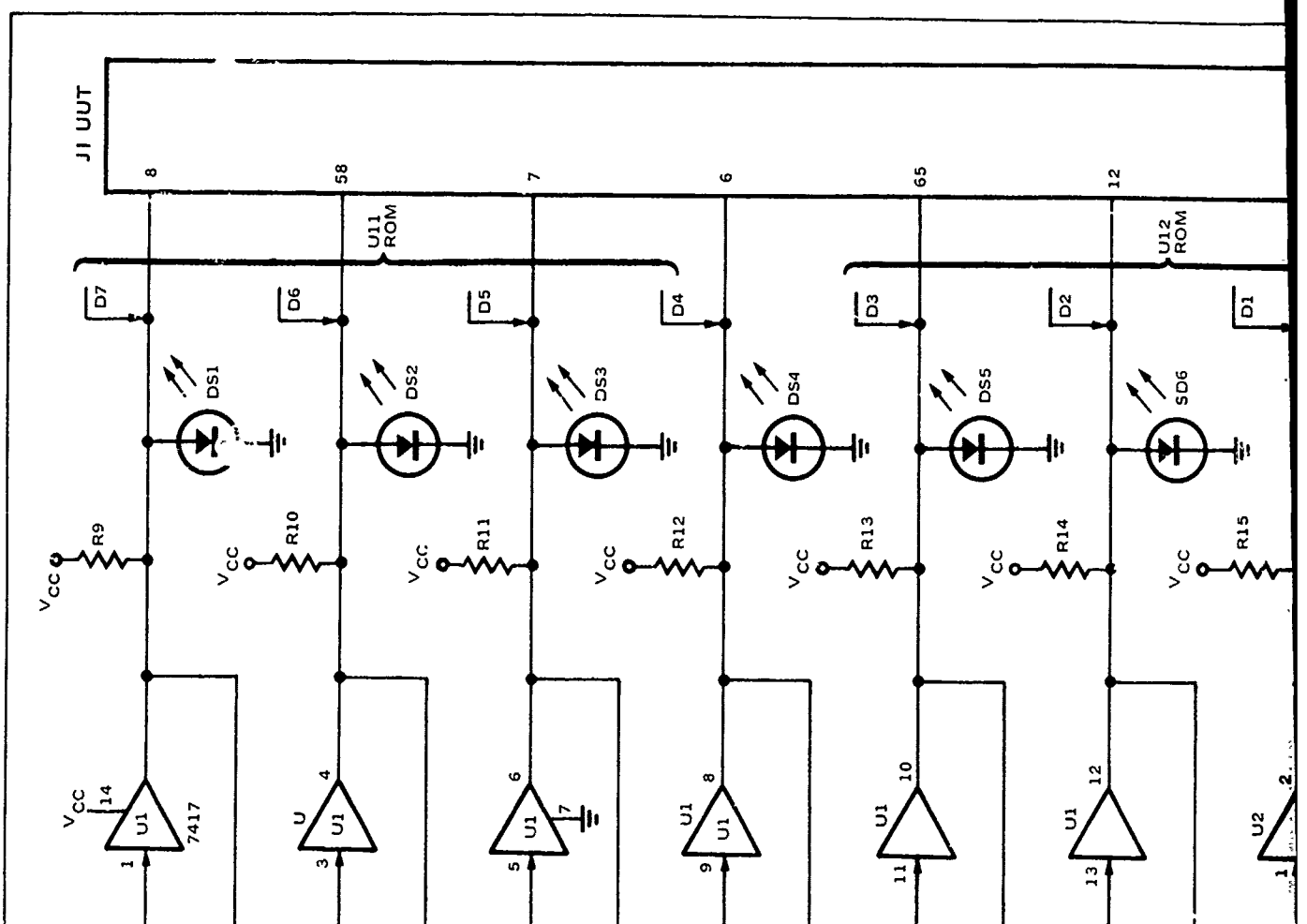


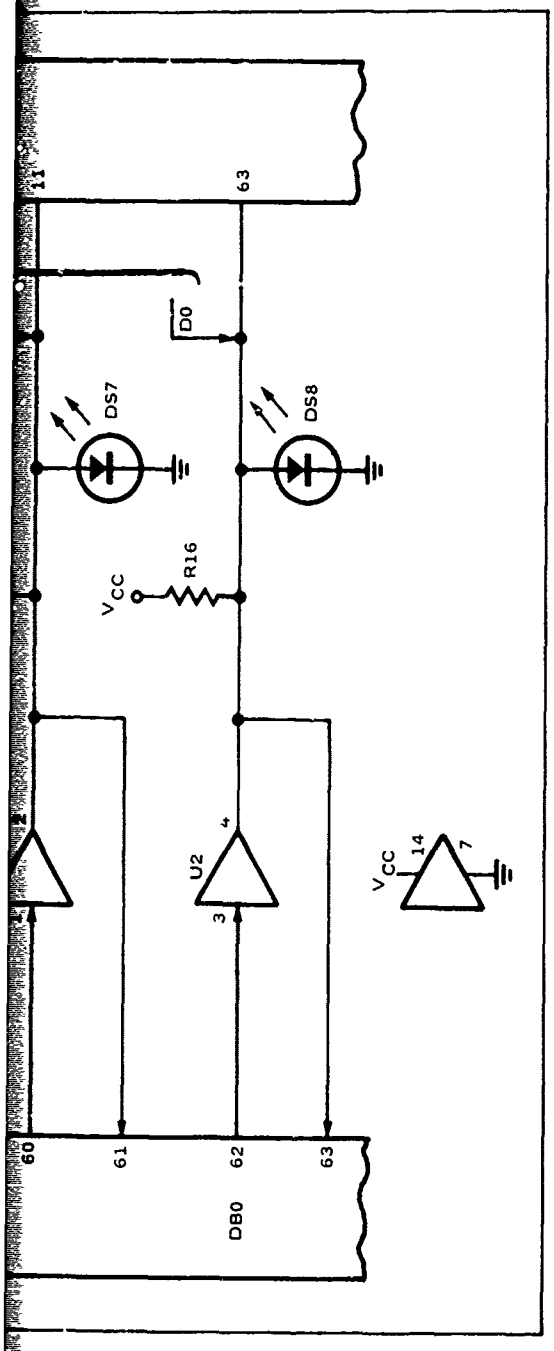

09333-15
U45
ADAPT
J1
CC
R8
10
18
17
21
19
16
24
12
23
14
13
66
29
U1
U2
GX1
+5V
+12V
-5V
GND
1
2
51
52
50
100
41
10
48
60
98
OVERALL VIEW
UMBILICAL CABLE
1635972 D/PCB
U45
TEST
ADAPTER

APPENDIX B – Schematics
Section B.2 – PN1635972 Circuit Board Test Adapter

B.2.2 – 8 Bit Data Bus, Buffers, Pull Up and LED Indicators; Part 2 of 6

09333-16

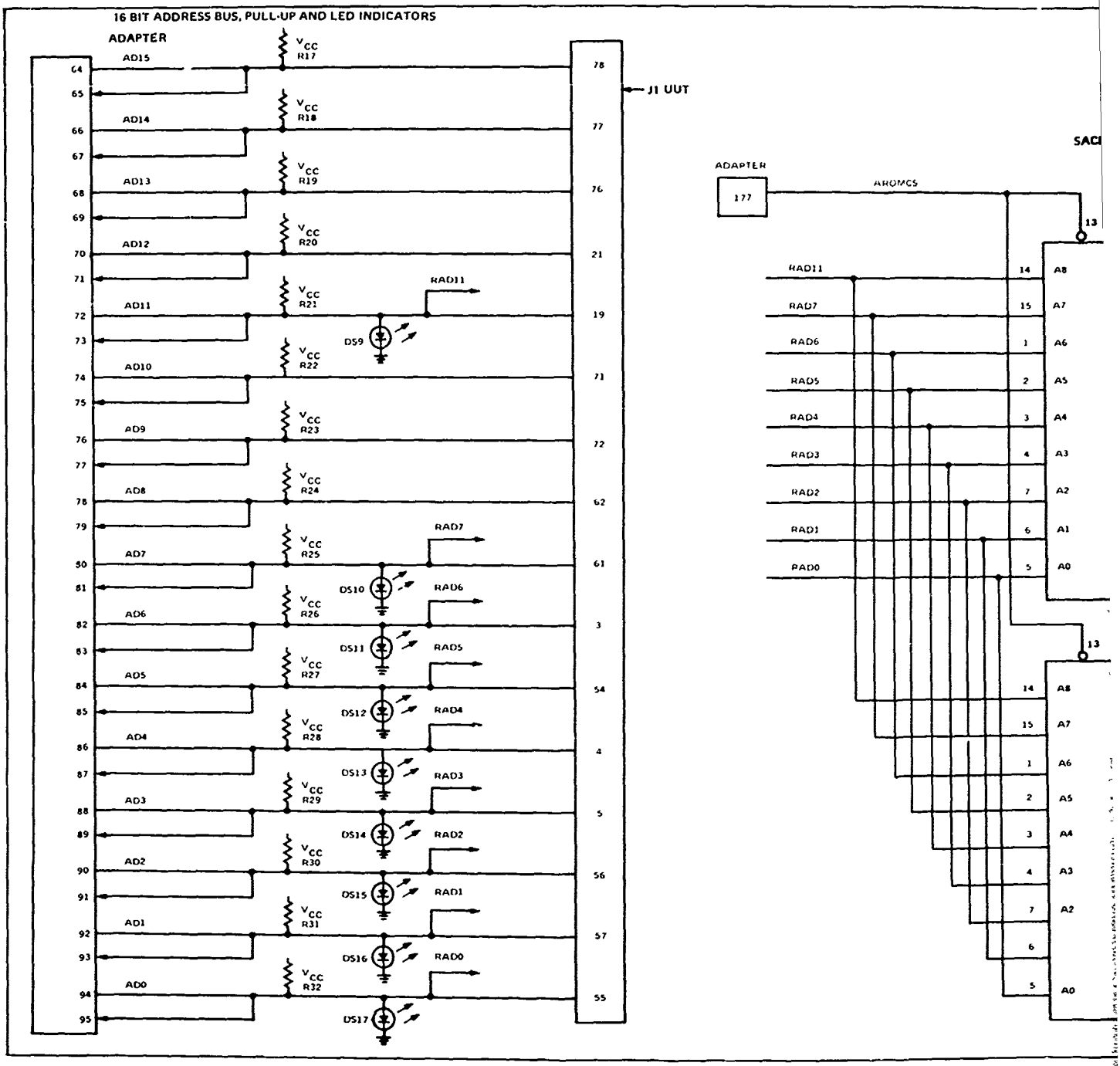


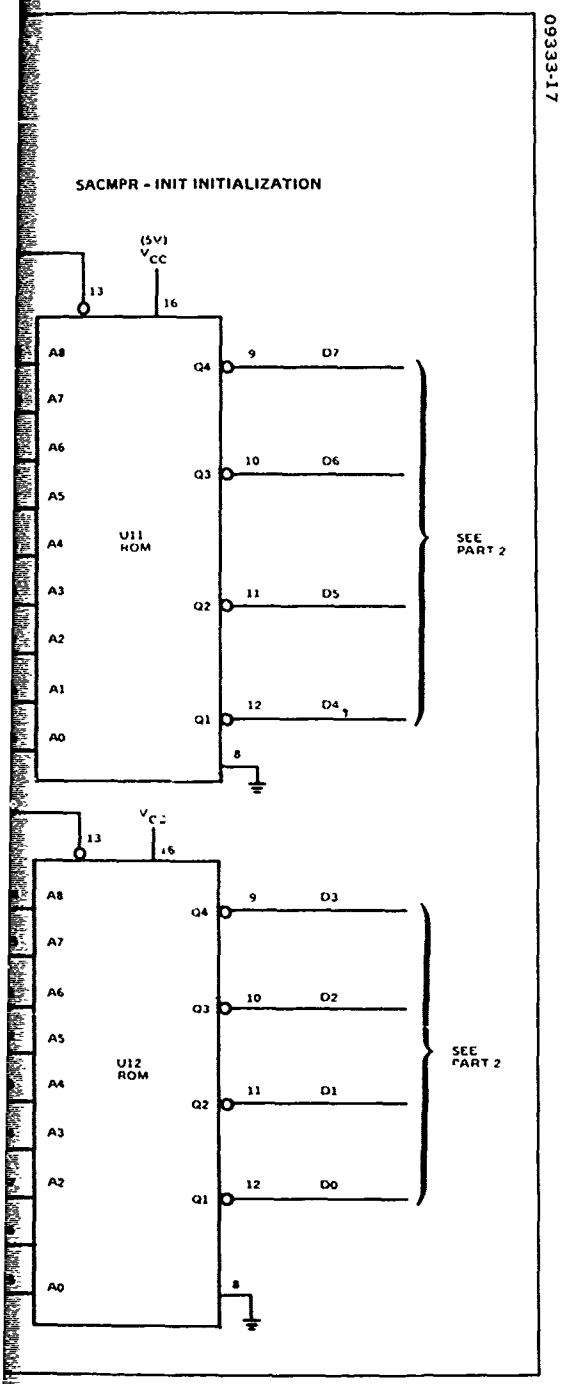


2

APPENDIX B – Schematics
Section B.2 – PN 1635972 Circuit Board Test Adapter

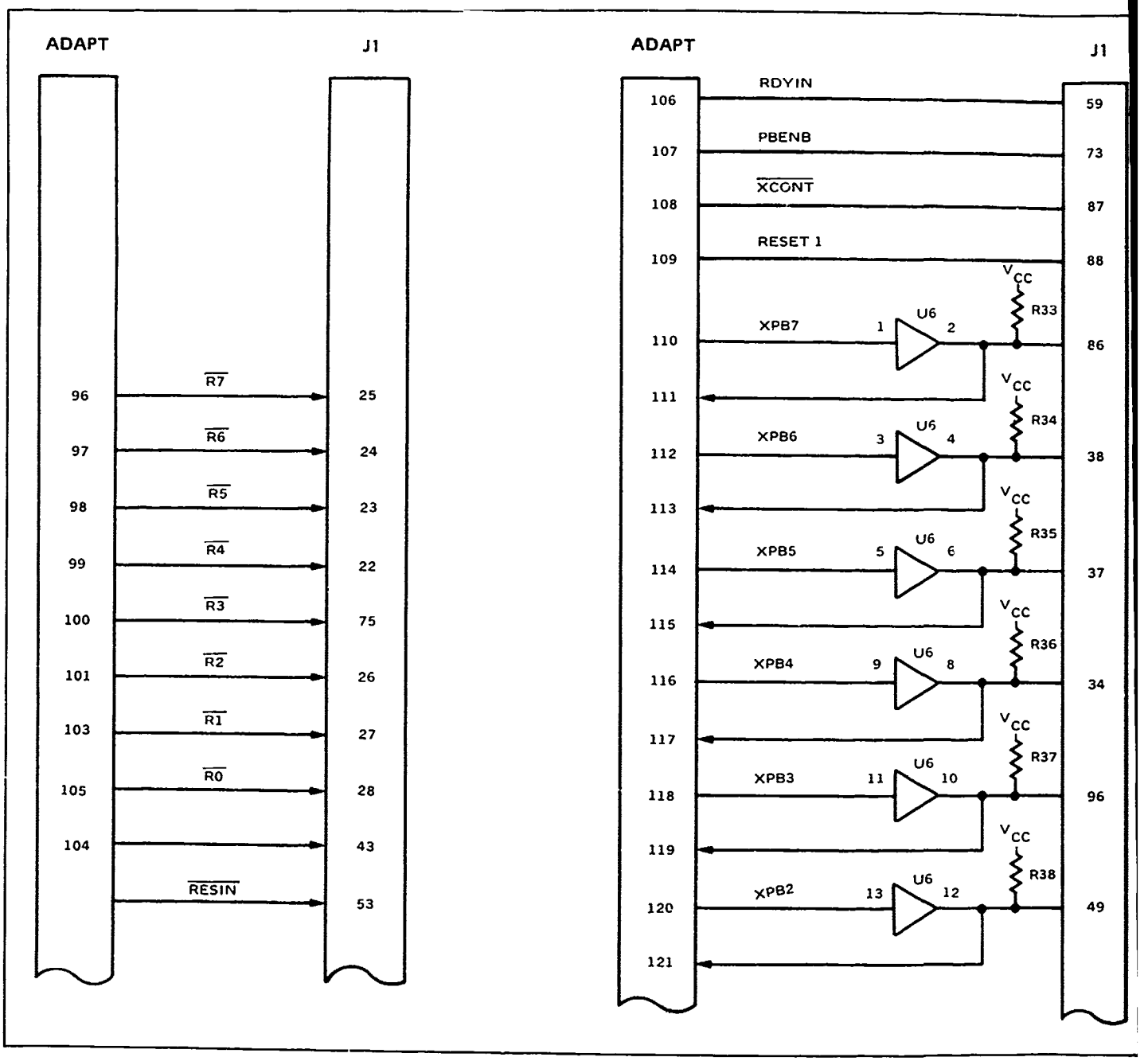
B.2.3 – 16 Bit Address Bus, SACMPR-INIT Initialization; Part 3 of 6

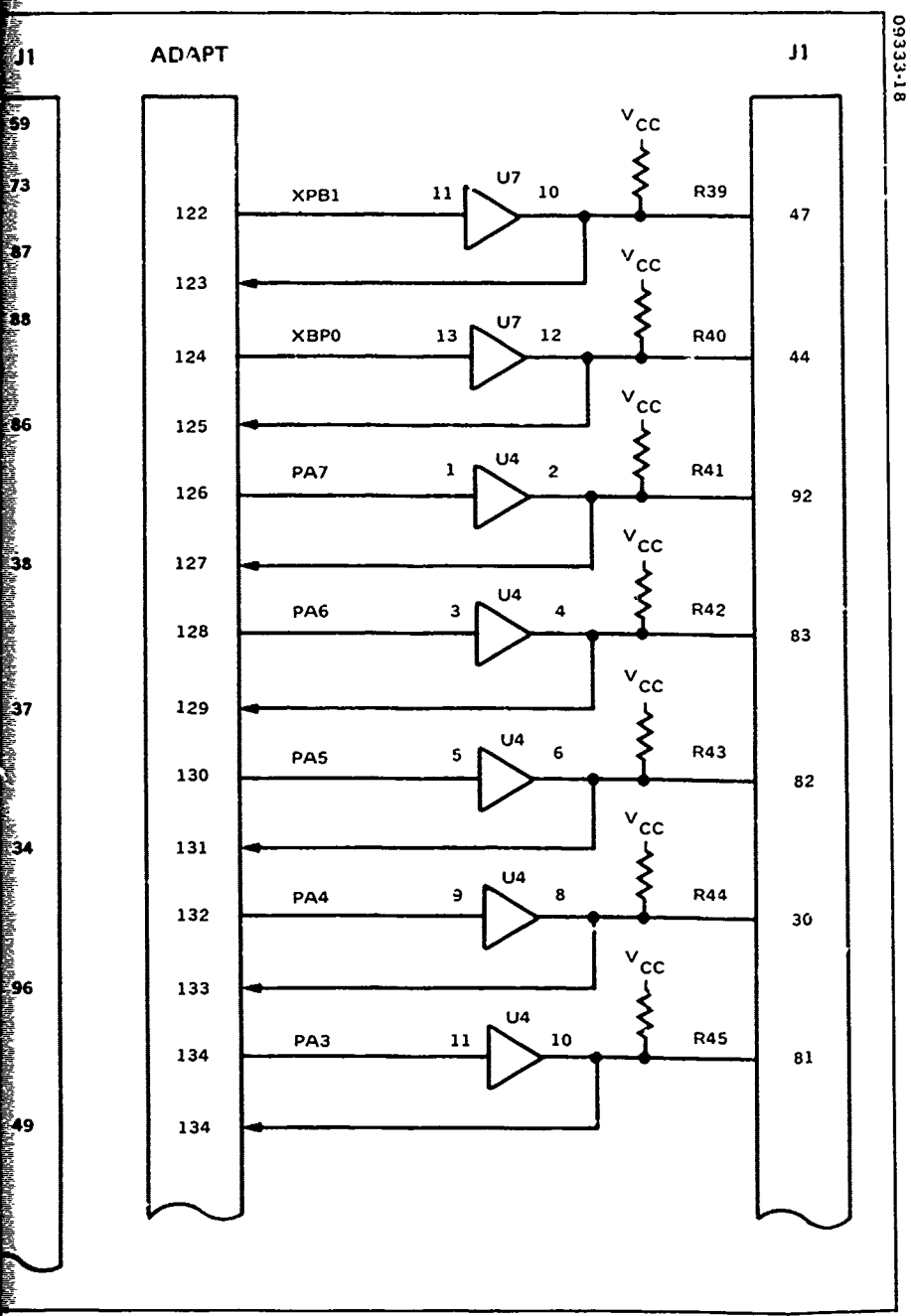



09333-17
SACMPR - INIT INITIALIZATION
(5V)
V_{CC}
13
16
A8
A7
A6
A5
A4
A3
A2
A1
A0
U11
ROM
Q4
Q3
Q2
Q1
9
10
11
12
D7
D6
D5
D4
8
SEE
PART 2
U12
ROM
D3
D2
D1
D0

APPENDIX B – Schematics
Section B. 2 – PN 1635972 Circuit Board Test Adapter

B. 2. 4 – 8255 PPI I/O BUFFER AND PULL UP; PART 4 OF 6

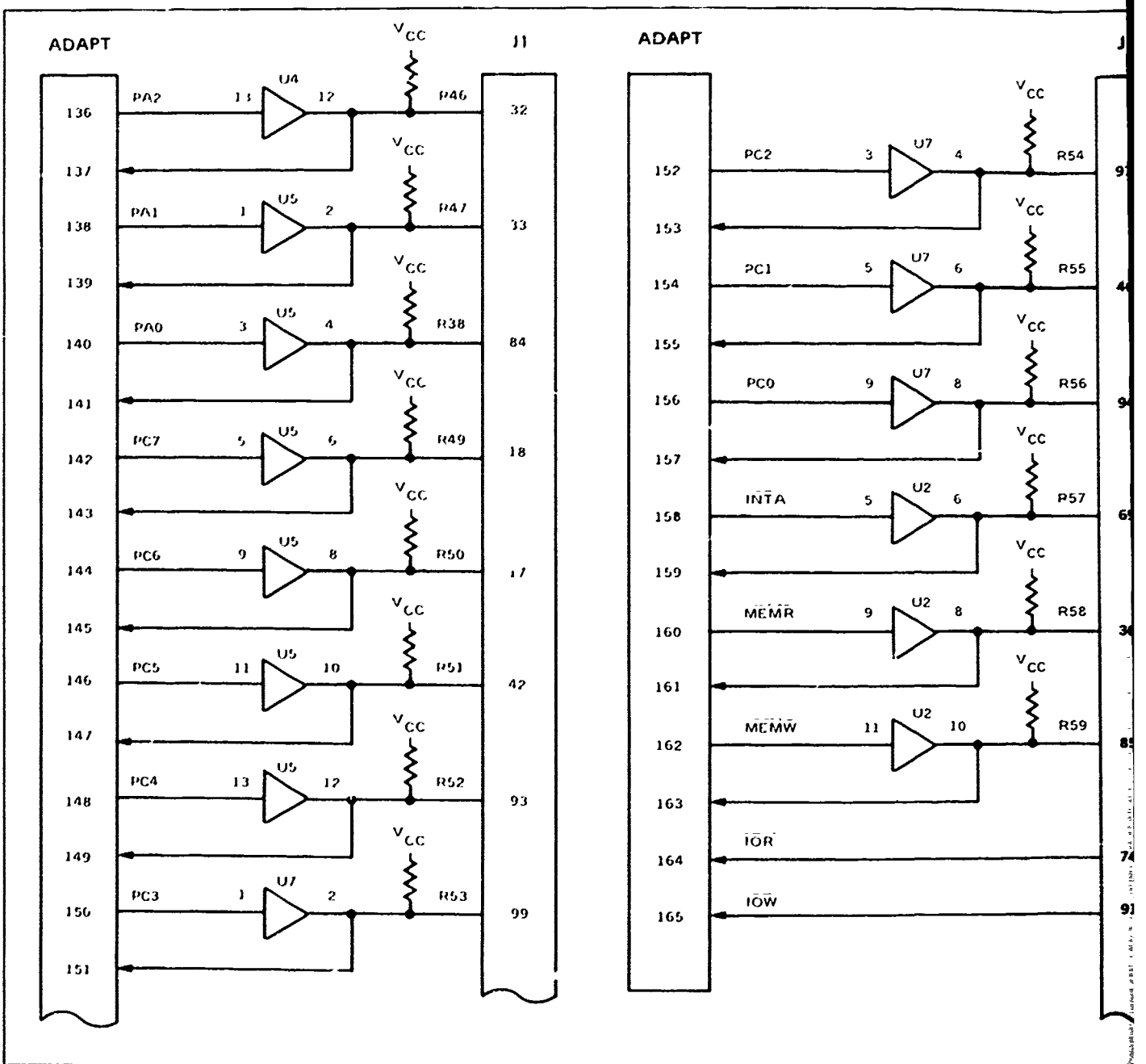


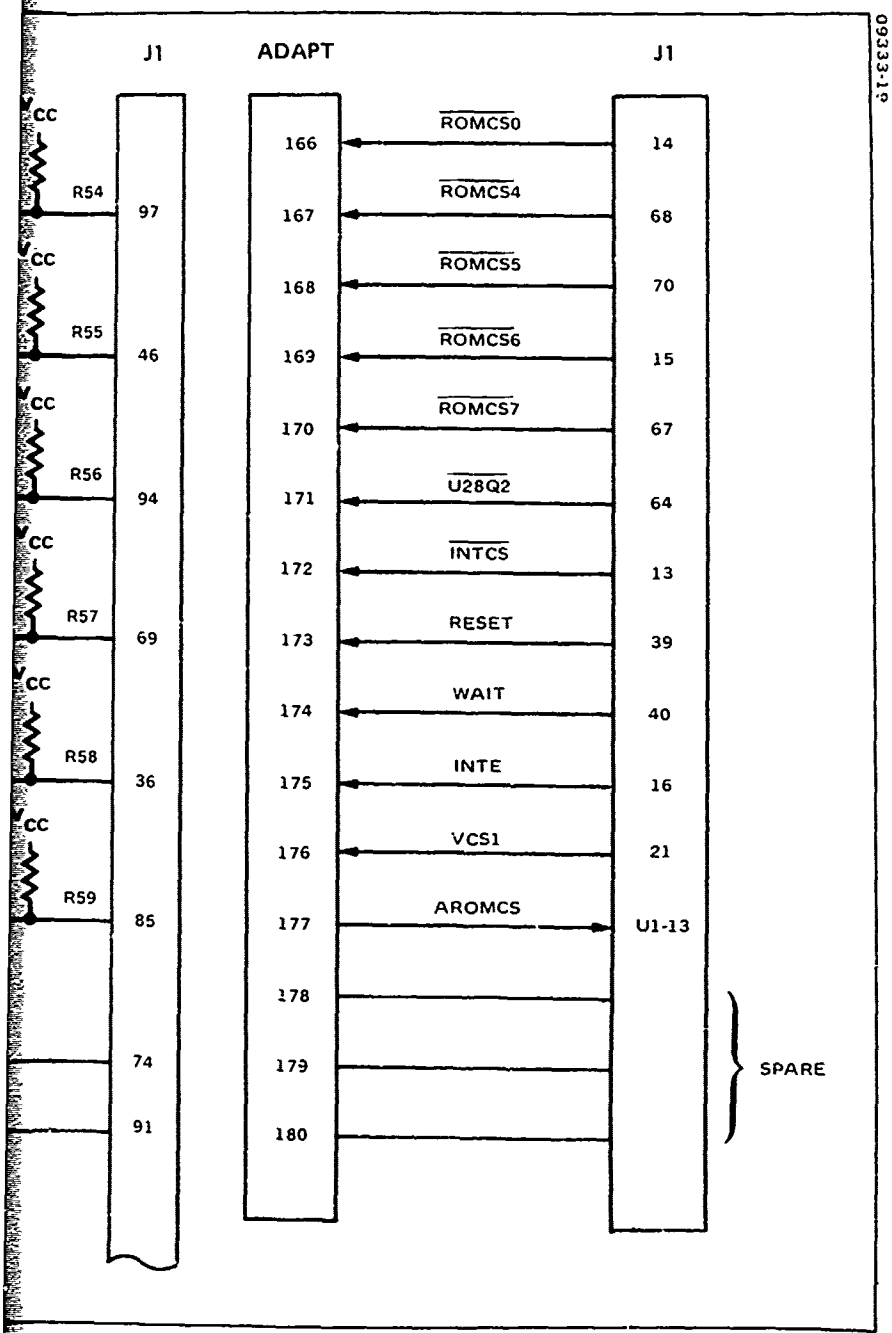

J1
ADAPT
J1
09333-18
59
73
87
88
86
38
37
34
96
49
122
123
124
125
126
127
128
129
130
131
132
133
134
134
XPB1
XBP0
PA7
PA6
PA5
PA4
PA3
U7
U4
11
10
13
12
1
2
3
4
5
6
9
8
11
10
V_{CC}
R39
R40
R41
R42
R43
R44
R45
47
44
92
83
82
30
81

2

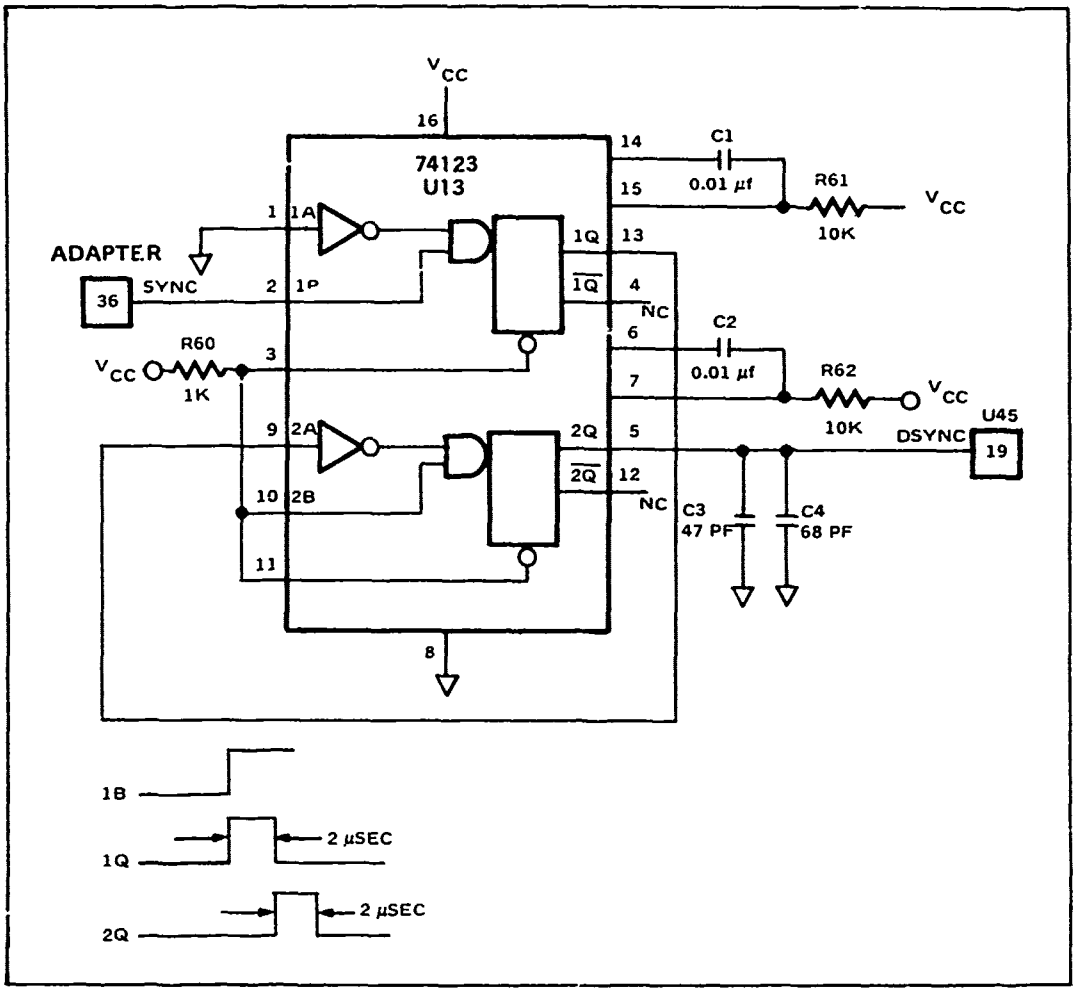
APPENDIX B – Schematics
Section B. 2 – PN 1635972 Circuit Board Test Adapter

B. 2. 5 – 8255 PPI I/O BUFFER AND PULL UP; PART 5 OF 6




J1
ADAPT
J1
09333-19
CC
R54
97
CC
R55
46
CC
R56
94
CC
R57
69
CC
R58
36
CC
R59
85
74
91
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
ROMCS0
ROMCS4
ROMCS5
ROMCS6
ROMCS7
U28Q2
INTCS
RESET
WAIT
INTE
VCS1
AROMCS
14
68
70
15
67
64
13
39
40
16
21
U1-13
SPARE

B.2.6 – 8228 SCBD, STROBE DELAY CIRCUIT; PART 6 OF 6



APPENDIX B – Schematics
Section B.2 – PN 1635972 Circuit Board Test Adapter

B.2.7 – TEST ADAPTER PARTS LIST

Parts List

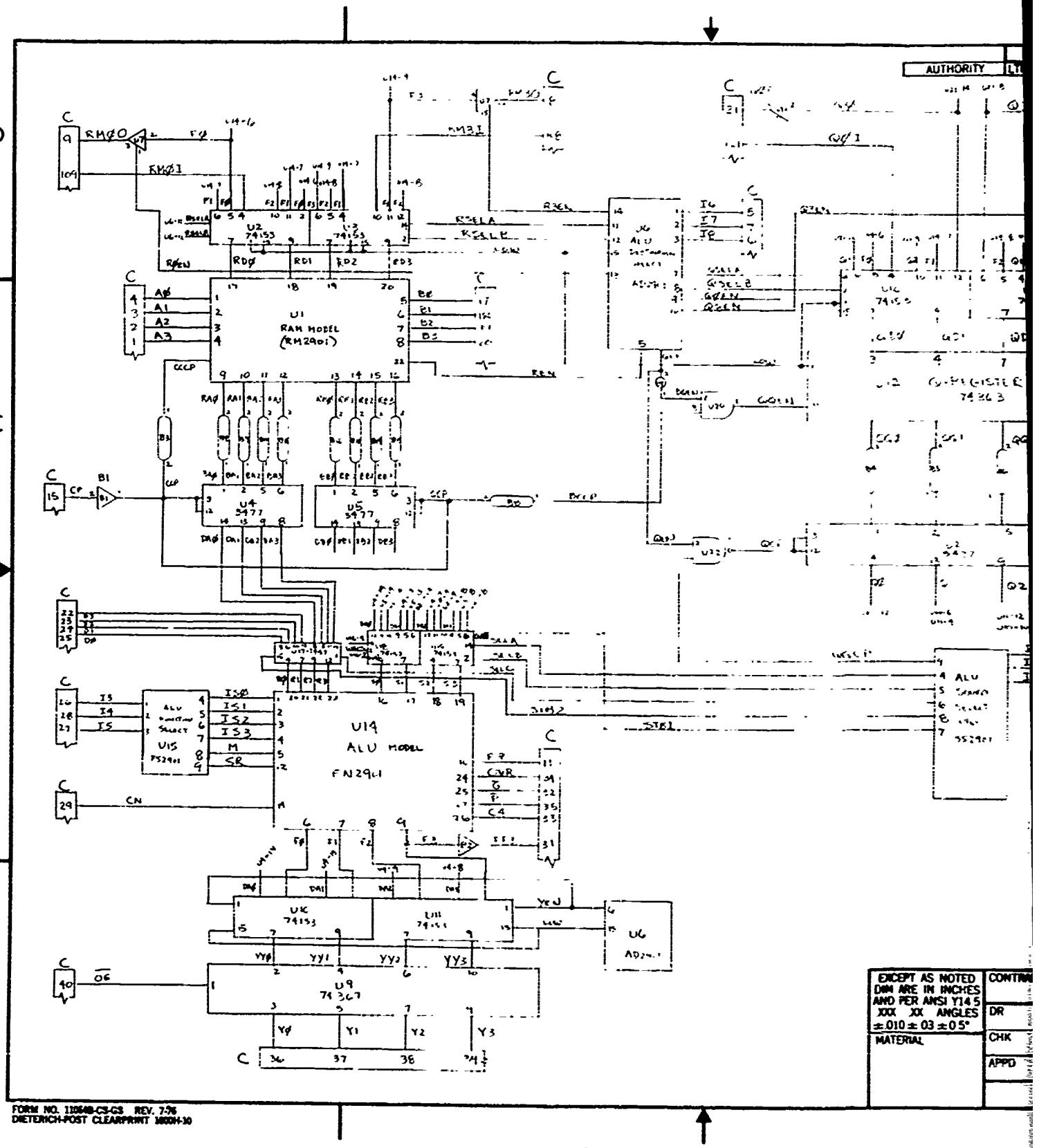
Ref. Designator	Part Number	Quantity
U1 - U10	7417	10
U13	74123	1
U11 - U12	93446 (ROM)	2
C3	47 Pf	1
C4	68 Pf	1
R1 - R60	1K (1/4W, 5%)	60
C1, C2	0.01 μf	2
R61, R62	10K (1/4W, 5%)	2
DS1 - DS17	LED - 547-2007	17
DIP SOCKET	16 pin - AUGAT-D	20

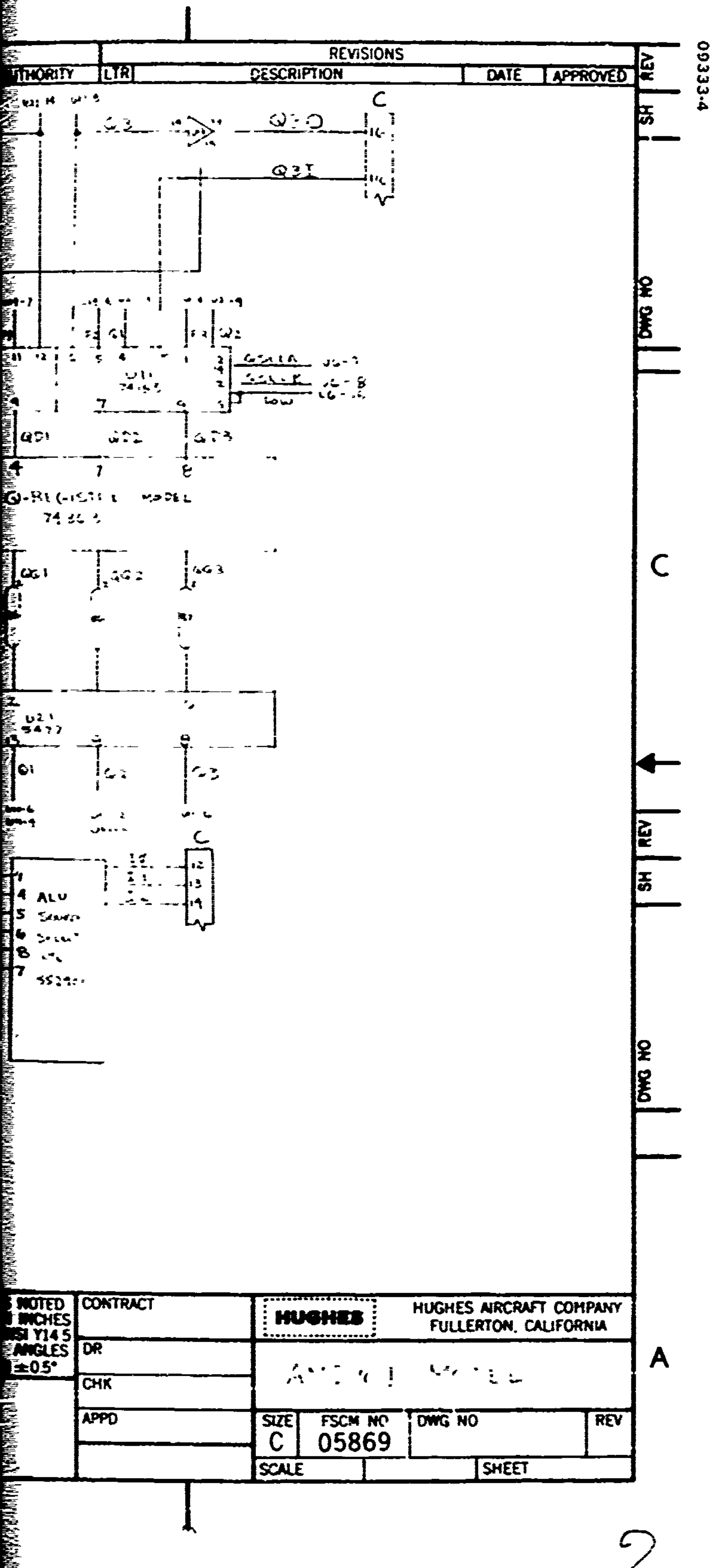
SECTION B.3

PN 1646178 CIRCUIT BOARD MODEL

APPENDIX B – Schematics
Section B. 3 – PN 1646178 Circuit Board Model

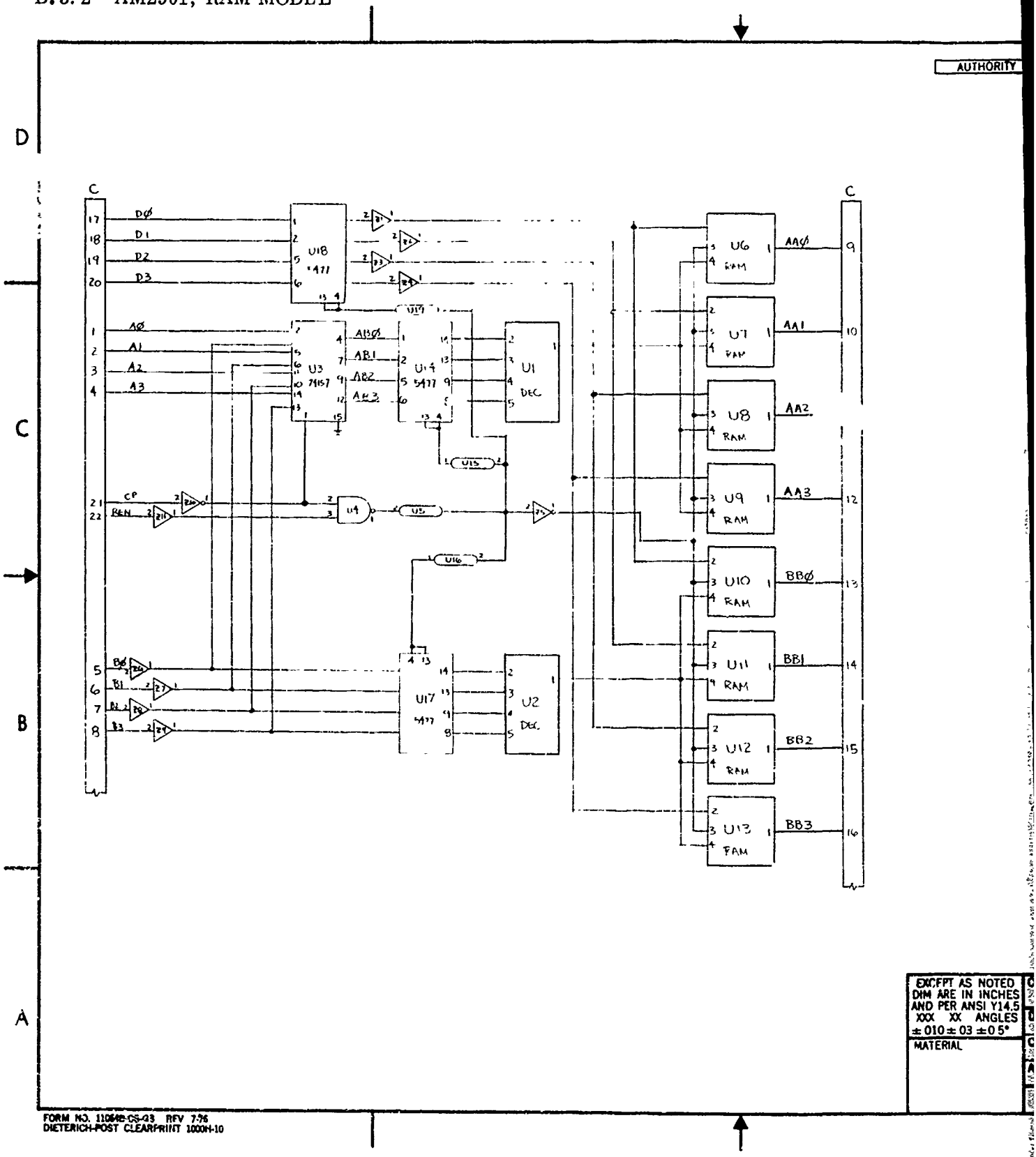
B. 3. 1 – AM2901 MODEL

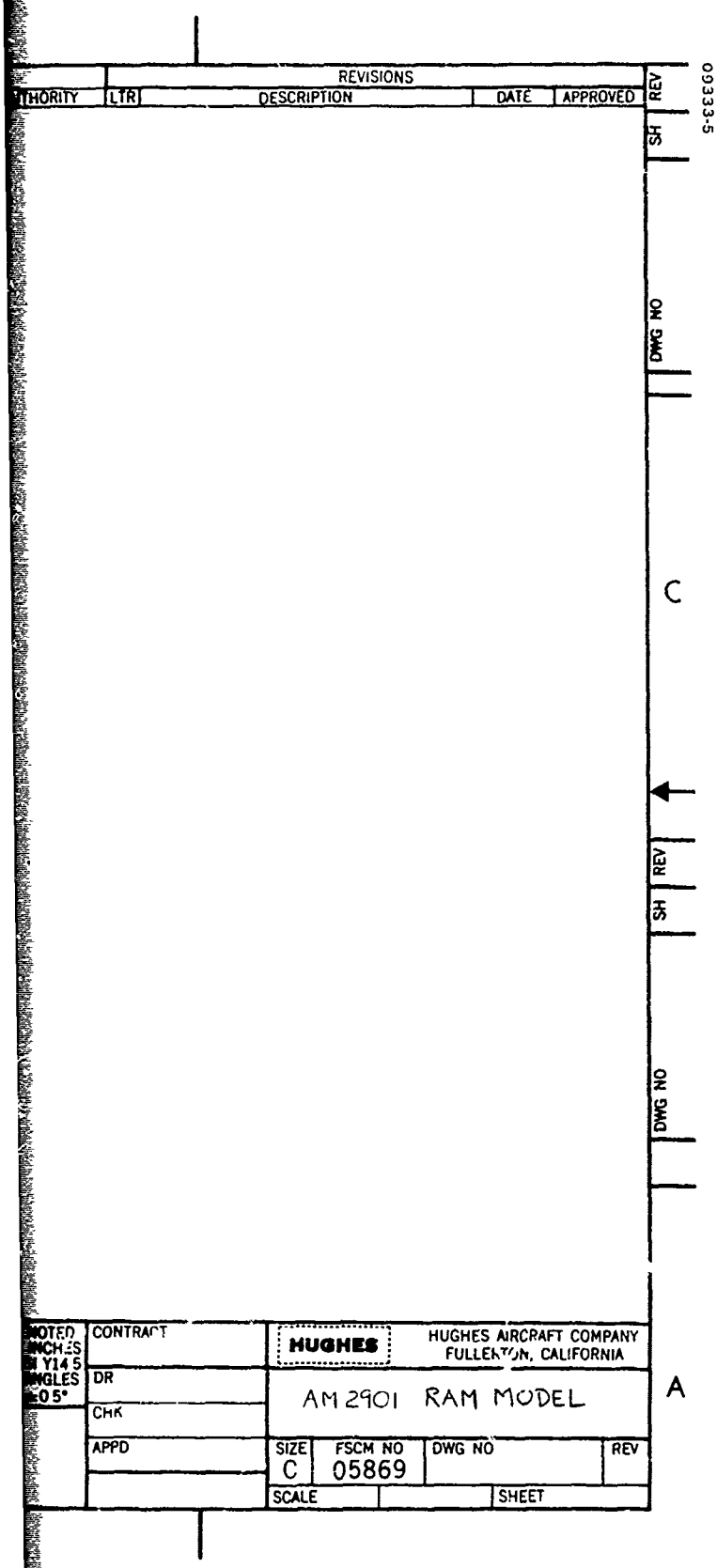



REVISIONS
AUTHORITY
LTR
DESCRIPTION
DATE
APPROVED
REV
SH
DWG NO
C
A
NOTED
INCHES
Y14.5
ANGLES
±0.5°
CONTRACT
DR
CHK
APPD
HUGHES
HUGHES AIRCRAFT COMPANY
FULLERTON, CALIFORNIA
SIZE
C
FSCM NO
05869
DWG NO
REV
SCALE
SHEET
2

APPENDIX B – Schematics
Section B. 3 – PN 1646178 Circuit Board Model

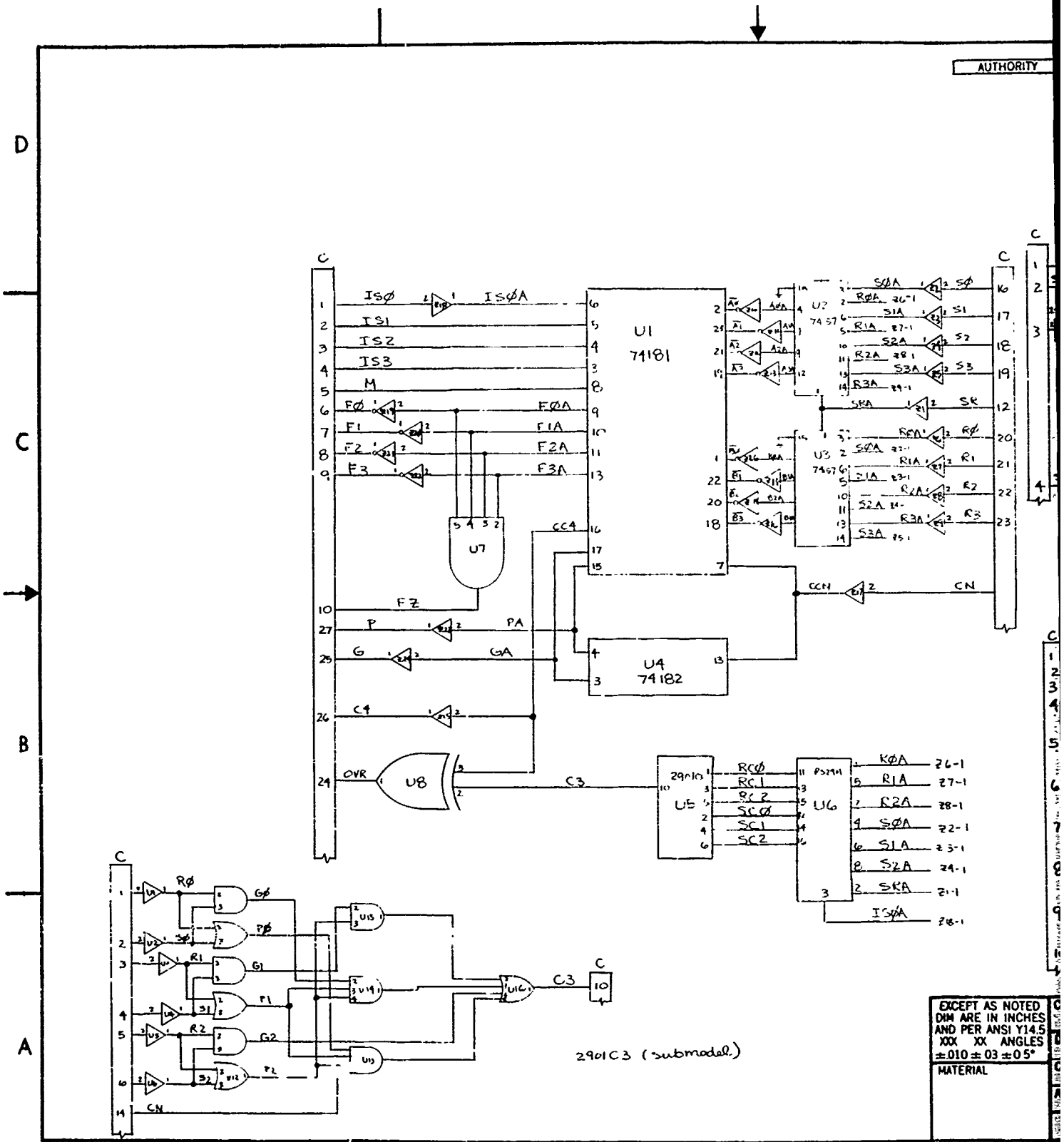
B. 3. 2 – AM2901, RAM MODEL




REVISIONS
THORITY
LTR
DESCRIPTION
DATE
APPROVED
REV
09333-5
SH
DWG NO
C
REV
SH
DWG NO
A
NOTED
INCHES
Y14.5
NGLES
±0.5°
CONTRACT
DR
CHK
APPD
HUGHES
HUGHES AIRCRAFT COMPANY
FULLERTON, CALIFORNIA
AM2901 RAM MODEL
SIZE
C
FSCM NO
05869
DWG NO
REV
SCALE
SHEET
1
2

APPENDIX B – Schematics
Section B. 3 – PN 1646173 Circuit Board Model

B. 3. 2 – AM 2901, ALU MODEL

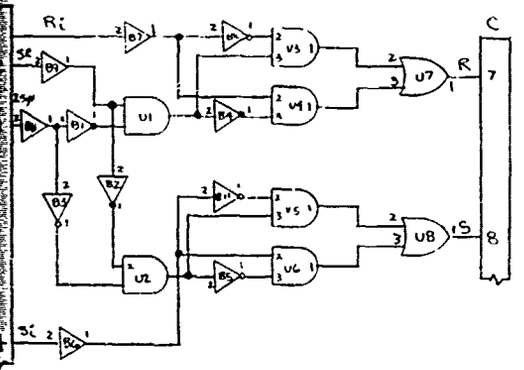


REVISIONS				
LTR	DESCRIPTION	DATE	APPROVED	REV

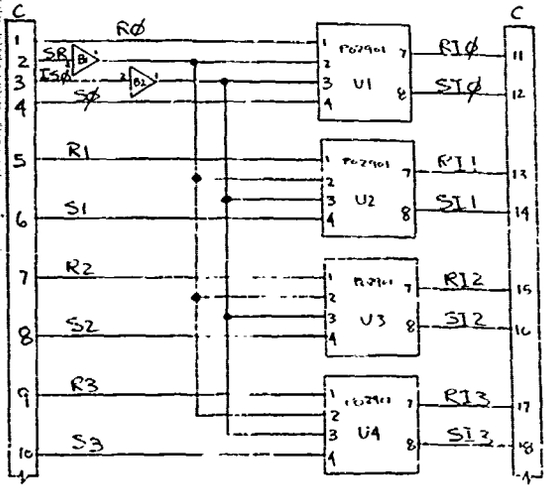
SH

DWG NO

PO2901 (SUBMODEL)



RS2901 (SUBMODEL)



REV

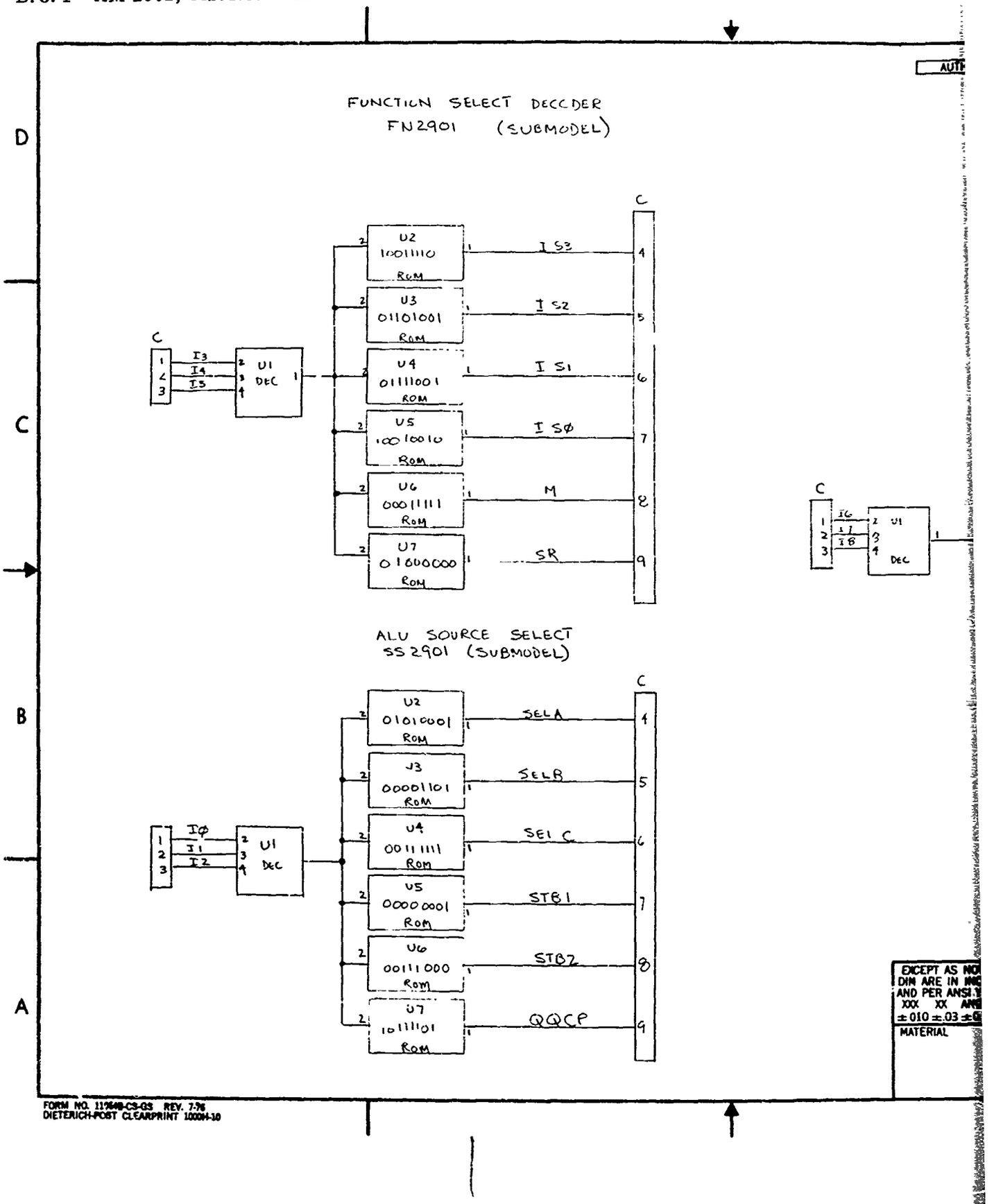
SH

DWG NO

CONTRACT	HUGHES	HUGHES AIRCRAFT COMPANY FULLERTON, CALIFORNIA		
DR	ALU MODEL, AM2901			
CHK				
AFPD	SIZE C	FSCM NO 05869	DWG NO	REV
	SCALE		SHEET	

A

B. 3. 4 – AM 2901, MICROINSTRUCTION DECODER

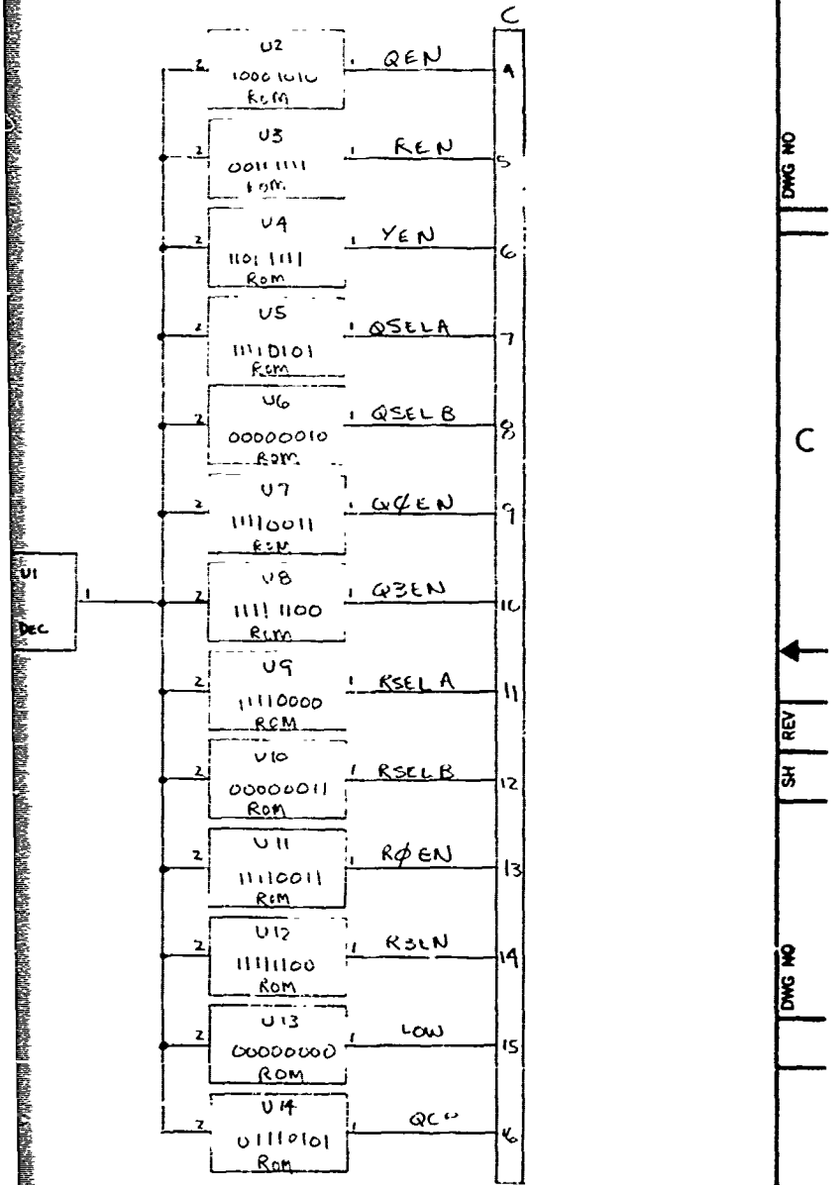


AUTHORITY	REVISIONS LTR	DESCRIPTION	DATE	APPROVED

09333-7

REV SH DWG NO REV SH DWG NO

ALU DESTINATION SELECT
AD 2901 (SUBMODEL)



EXCEPT AS NOTED DIM ARE IN INCHES AND PER ANSI Y14.5 XXX XX ANGLES .010 ± 03 ± 0.5° MATERIAL	CONTRACT	HUGHES	HUGHES AIRCRAFT COMPANY FULLERTON, CALIFORNIA
	DR		AM2901, MICROINSTR DECODE
	CHK		
	APPD	SIZE C / FSCM NO 05869 / DWG NO / REV	
		SCALE / SHEET	

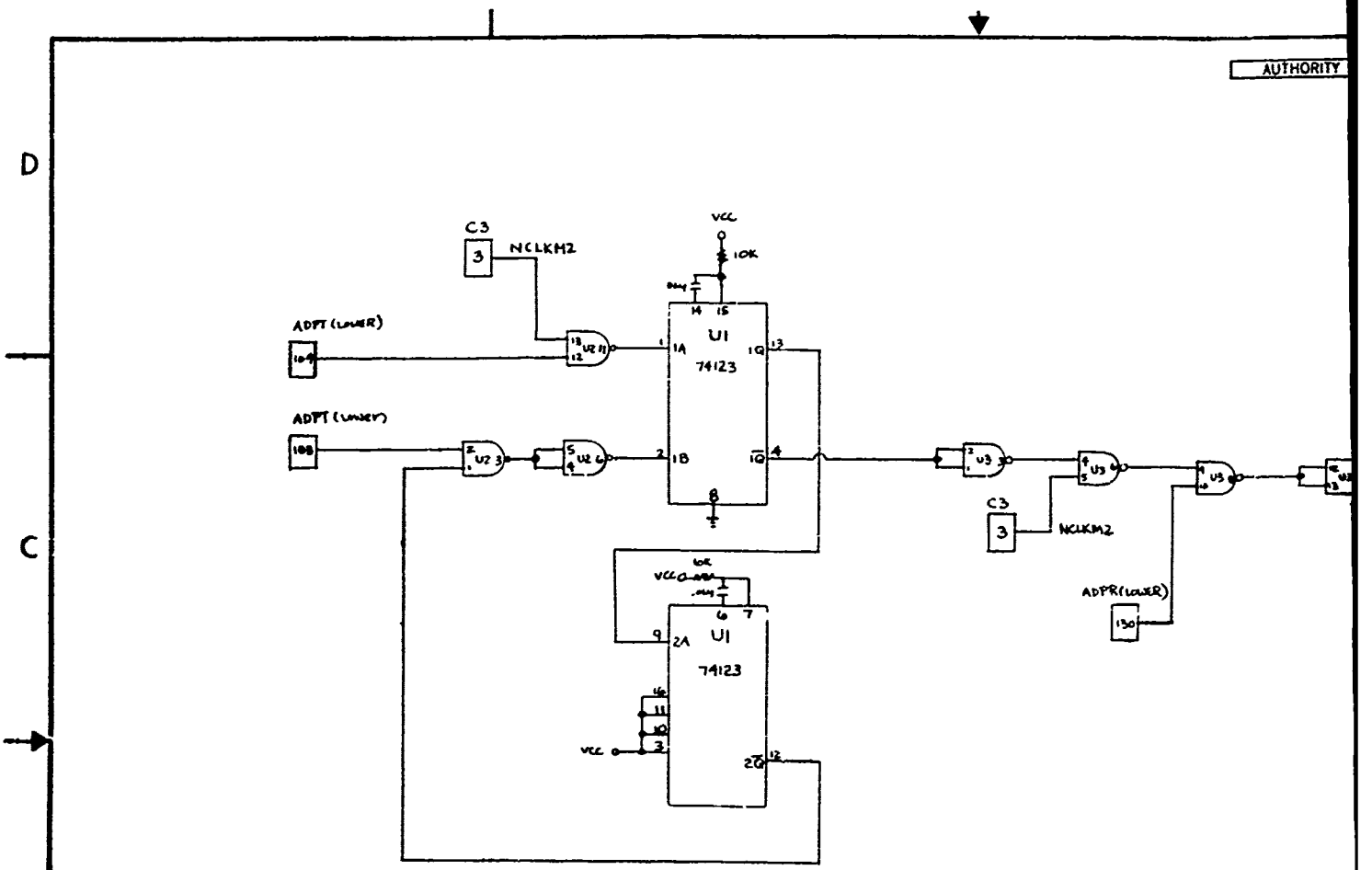
2

SECTION B.4

PN 1646178 CIRCUIT BOARD TEST ADAPTER

APPENDIX B – Schematics
Section B. 4 – PN 1646178 Circuit Board Test Adapter

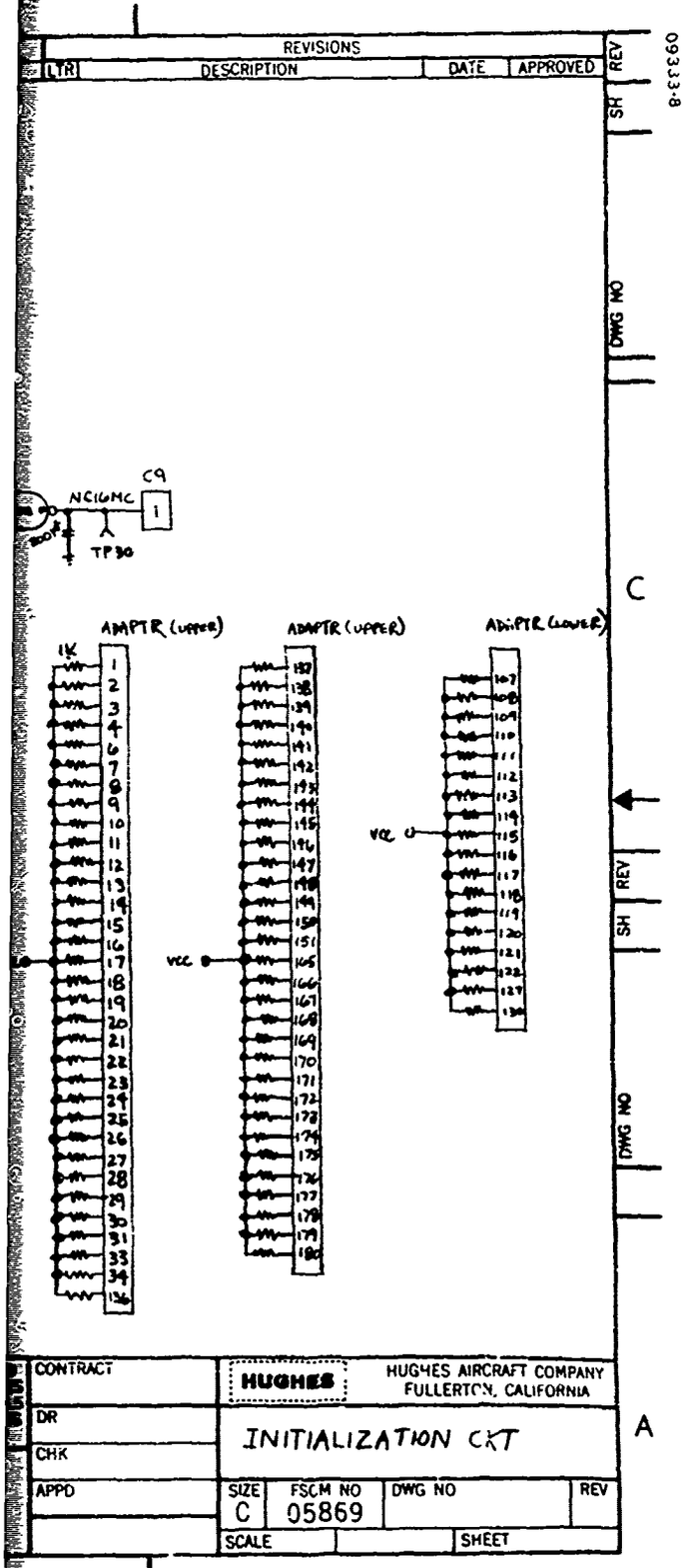
B. 4. 1 – INITIALIZATION CIRCUIT



REF NO	PART NO.
U1	74123
U2	7400
U3	7400

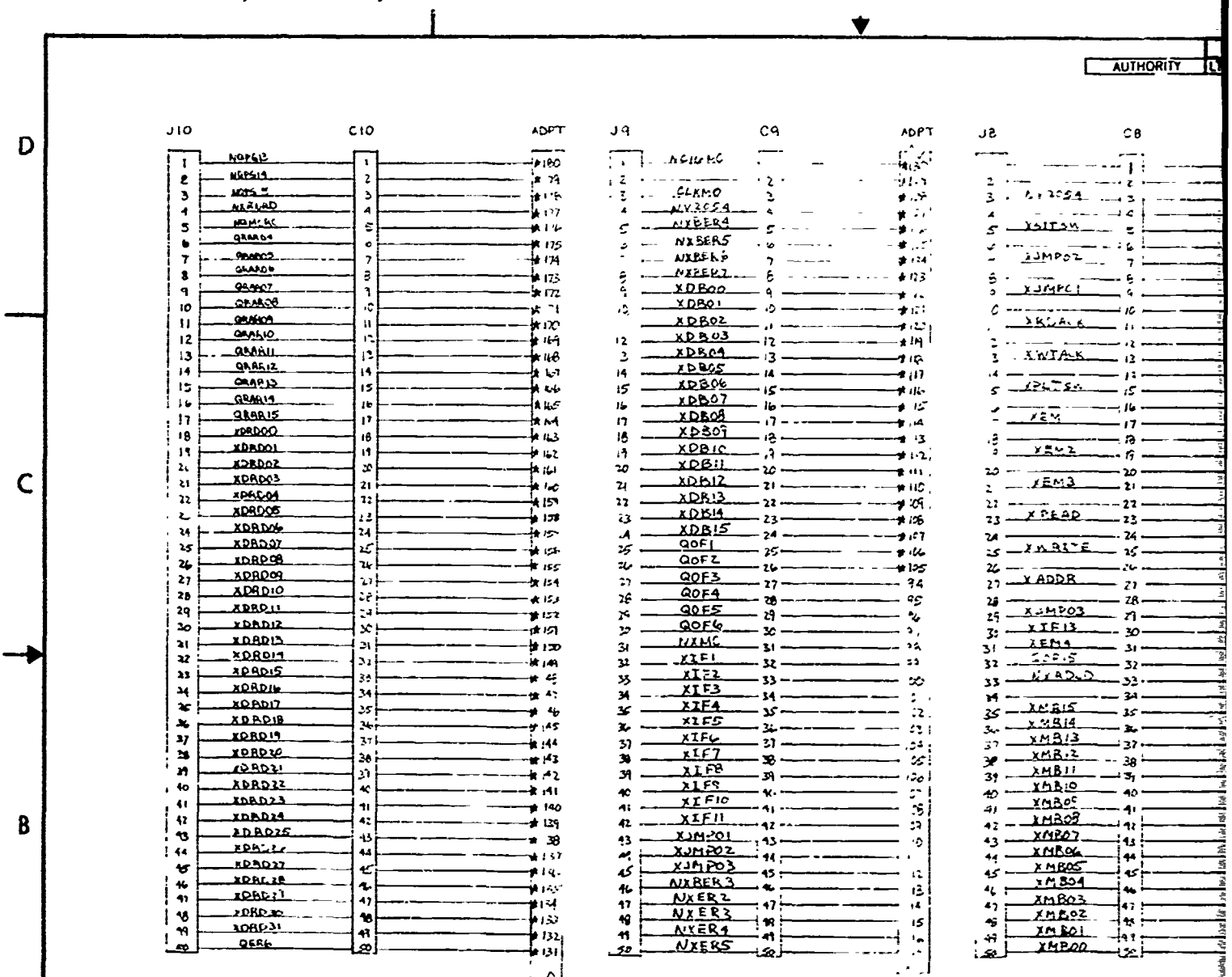
EXCEPT AS NOTED DIM ARE IN INCHES AND PER ANSI Y14.5
XXX XX ANGLES
±.010 ± 03 ± 05°
MATERIAL

FORM NO. 110648-CS-GS REV. 7-76
DIETERICH-POST CLEARPRINT 1600H-10


REVISIONS
LTR
DESCRIPTION
DATE
APPROVED
REV
SH
DWG NO
09333-8
C9
NC16MC
1
TP30
C
ADAPTR (UPPER)
ADAPTR (UPPER)
ADAPTR (LOWER)
1K
VCC
VCC
REV
SH
DWG NO
CONTRACT
HUGHES
HUGHES AIRCRAFT COMPANY
FULLERTON, CALIFORNIA
DR
CHK
APPD
INITIALIZATION CKT
A
SIZE
C
FSCM NO
05869
DWG NO
REV
SCALE
SHEET

APPENDIX B – Schematics
Section B.4 – PN 1646178 Circuit Board Test Adapter

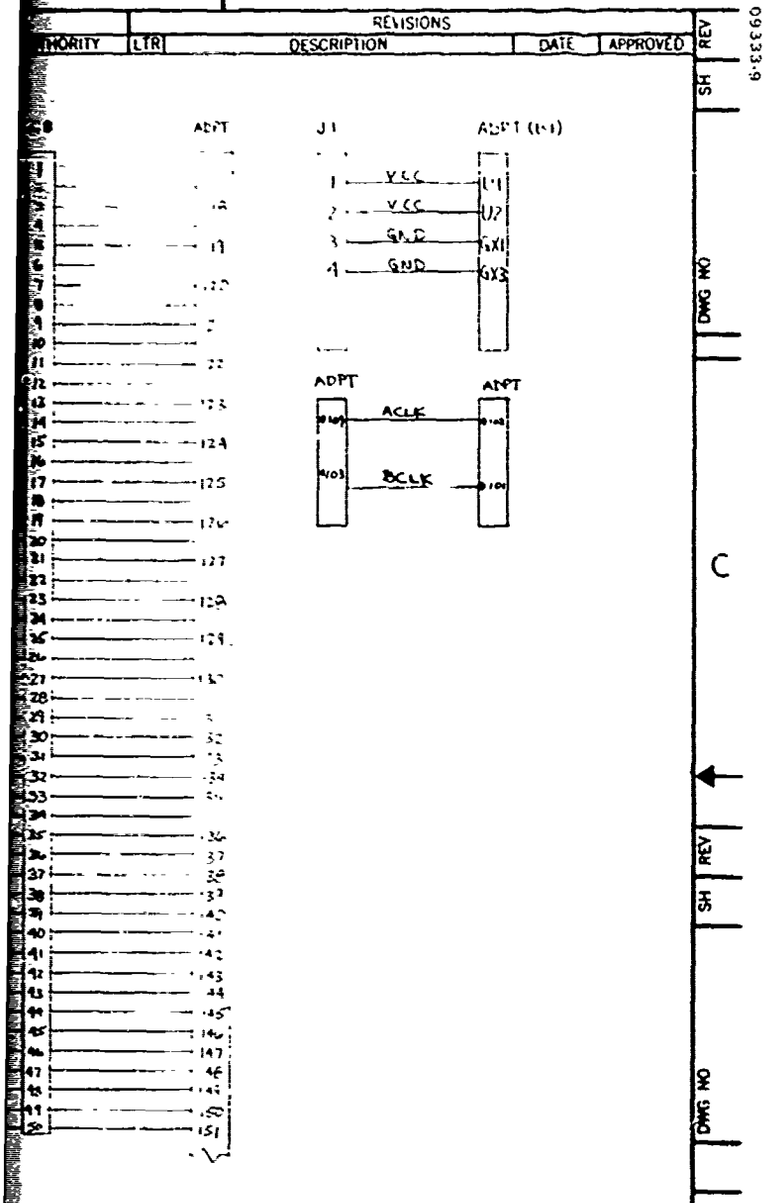
B.4.2 – WIRE LIST, ADAPTER, PART 1



NOTE: * Designates Pin on Lower Adapter Board

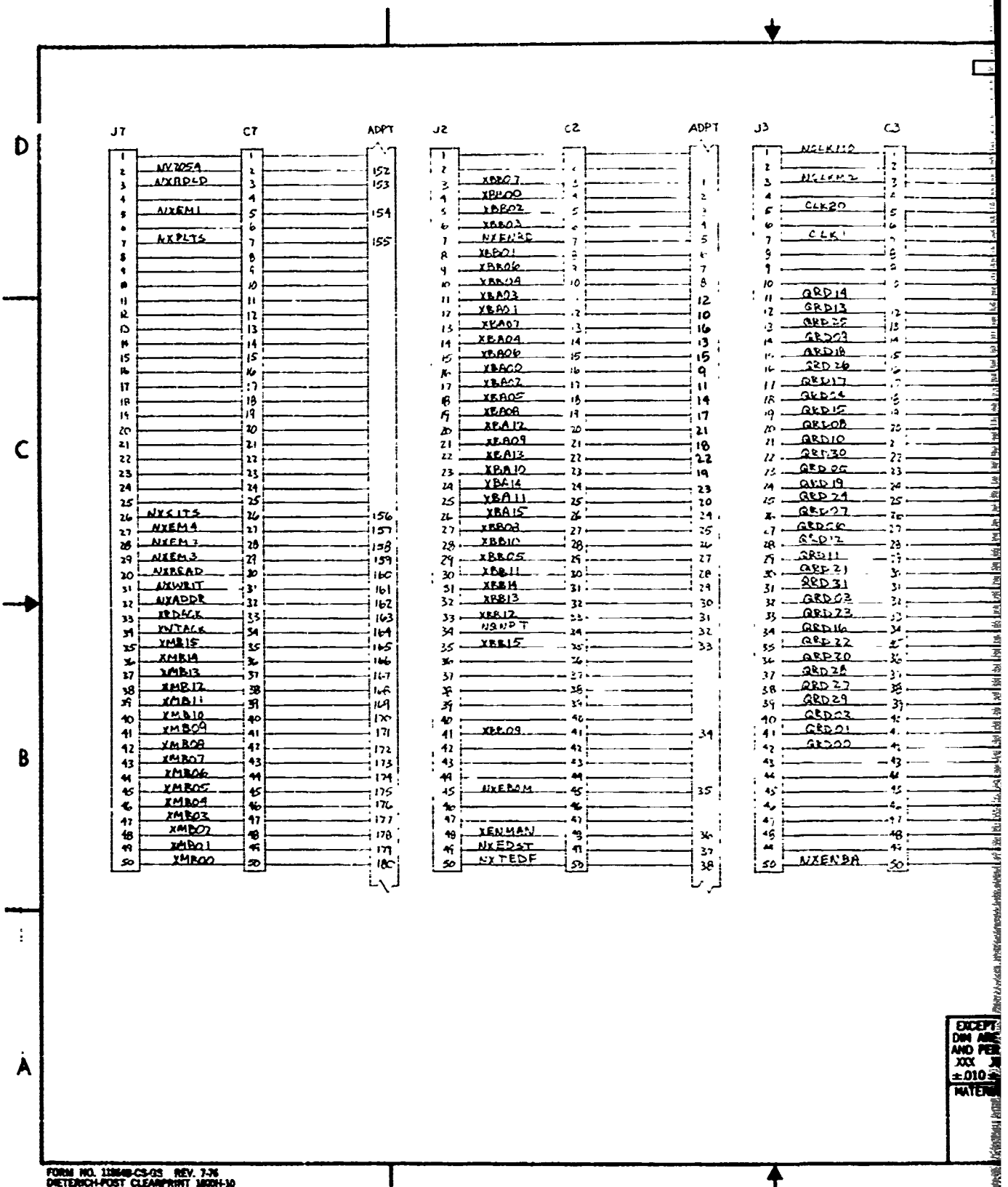
EXCEPT AS NOTED DIM ARE IN INCHES AND PER ANSI Y14.5 XXX XX ANGLES ±.010 ±.03 ±0.5°	CONT
MATERIAL	DR
	CHK
	APPD

FORM NO. 110648-CS-03 REV. 7-76
DIETERICH-POST CLEARPRINT 1800H-10


REVISIONS
AUTHORITY
LTR
DESCRIPTION
DATE
APPROVED
REV
SH
DWG NO
ADPT
J1
ADPT (1:1)
1 VCC
2 VCC
3 GND
4 GND
ADPT
ADPT
ACLK
BCLK
C
A
NOTED
IN INCHES
ANSI Y14.5
ANGLES
±0.5°
CONTRACT
DR E. Levitch
CHK
APPD
4-23-80
HUGHES
HUGHES AIRCRAFT COMPANY
FULLERTON CALIFORNIA
WIRE LIST, ADAPTER
SIZE
C
FSCM NO
05869
DWG NO
REV
SCALE
SHEET 1 OF 2

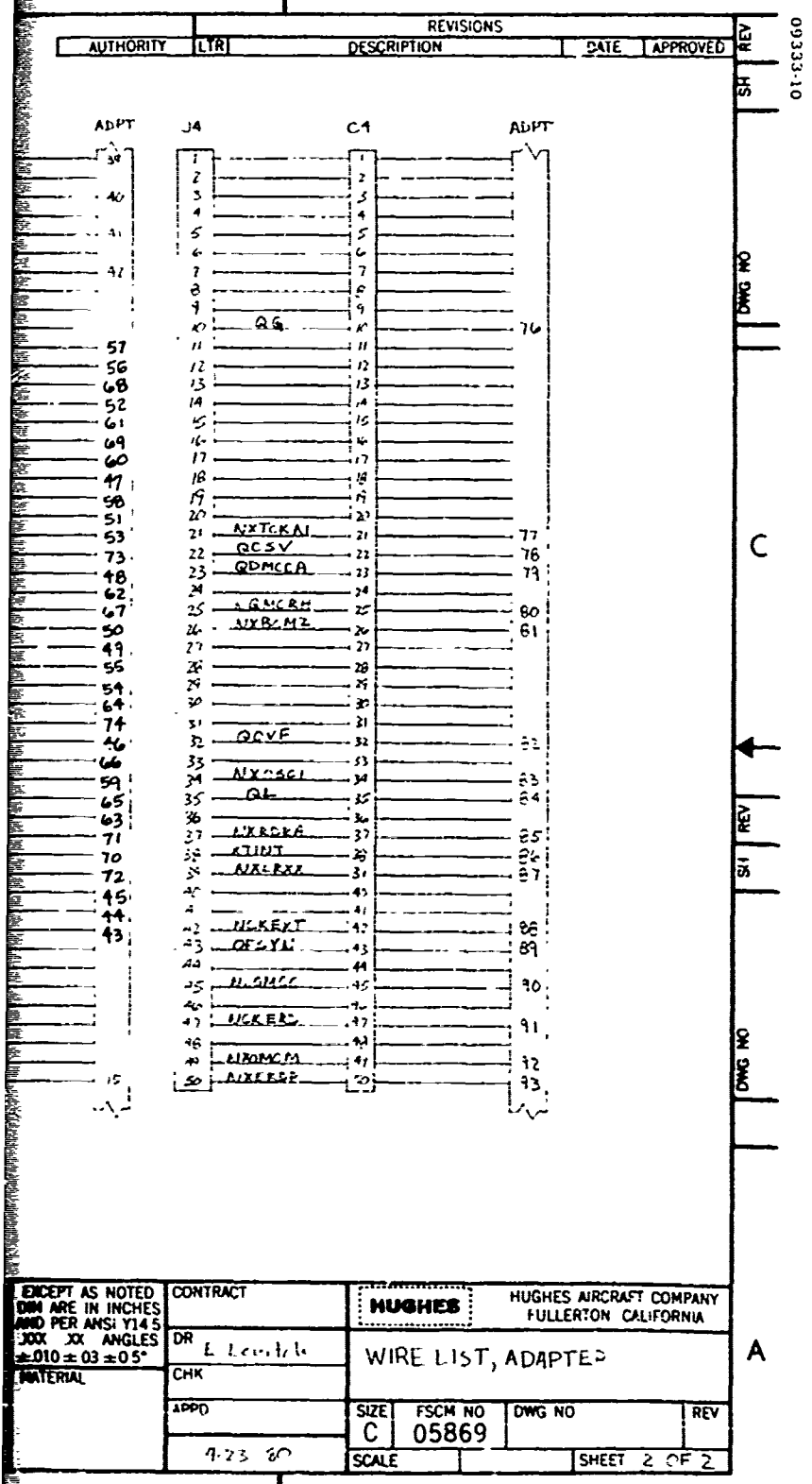
APPENDIX B – Schematics
Section B. 4 – PN 1646178 Circuit Board Test Adapter

B. 4. 3 – WIRE LIST, ADAPTER, PART 2

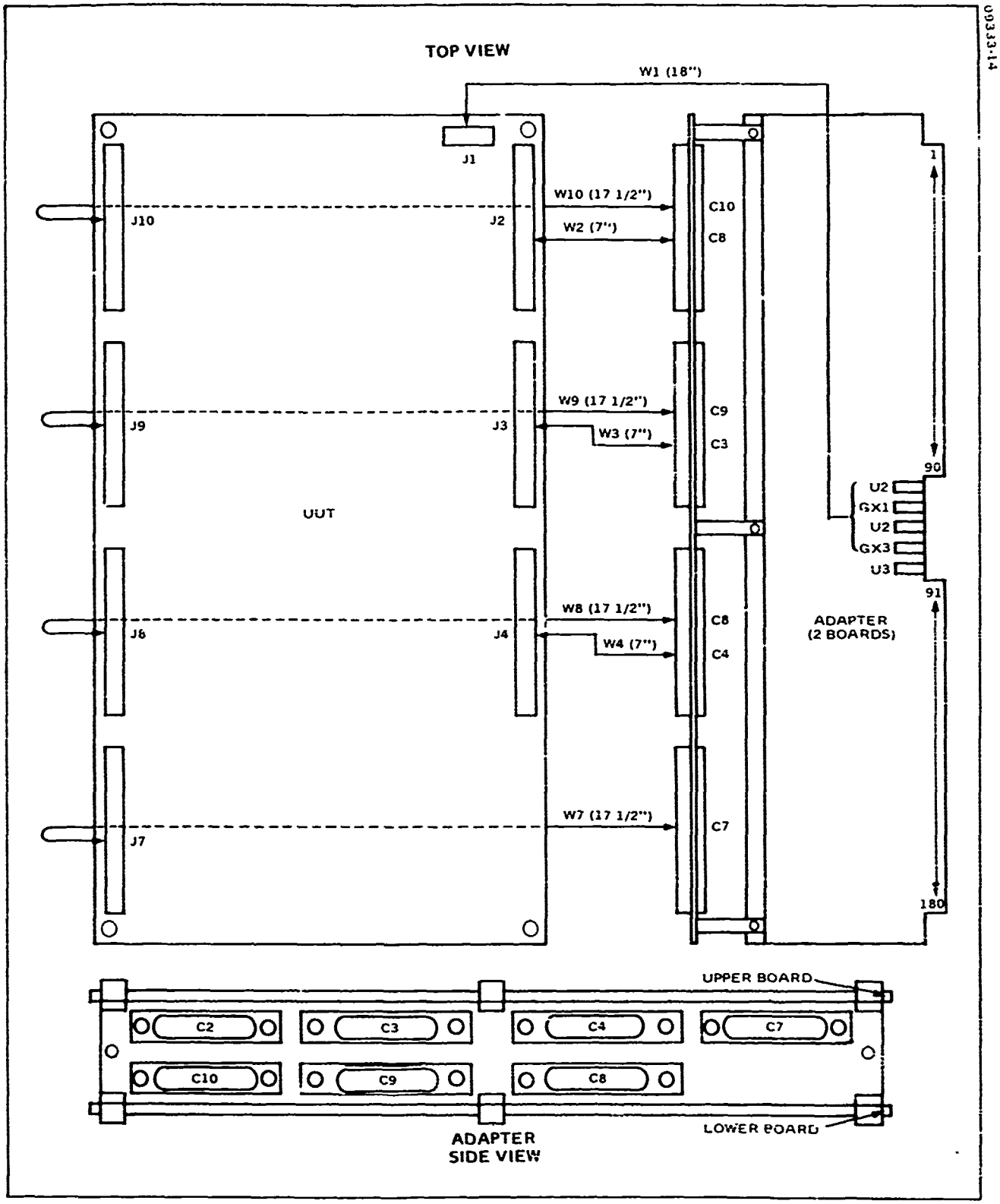


EXCEPT
DIM ARE
AND PER
XXX
±.010
MATERI

FORM NO. 1186468-CS-03 REV. 7-76
DIETERICH-POST CLEARPRINT 1800H-10


REVISIONS
AUTHORITY
LTR
DESCRIPTION
DATE
APPROVED
REV
SH
DWG NO
ADPT
J4
C4
ADPT
QG
NXTCKAI
QCSV
QDMCCA
QCVF
NXCSCI
QL
NXRCKA
XTINT
NXERXX
HCKEXT
QFSYN
HCKERR
NXRMCM
NXERRP
C
REV
SH
DWG NO
A
EXCEPT AS NOTED DIM ARE IN INCHES AND PER ANSI Y14.5 XXX XX ANGLES ±.010 ± .03 ±.05°
MATERIAL
CONTRACT
DR E Lewis
CHK
APPD
9-23-80
HUGHES
HUGHES AIRCRAFT COMPANY
FULLERTON CALIFORNIA
WIRE LIST, ADAPTER
SIZE C
FSCM NO 05869
DWG NO
REV
SCALE
SHEET 2 OF 2
09233-10

B.4.4 – LAYOUT AND INTER CONNECTIONS. ADAPTER



APPENDIX C – REVISIONS

SECTION C.1

REVISIONS TO DTS-70 IMPLEMENTATION PLAN

APPENDIX C – Revisions
Section 1 – Revisions to DTS 70 Implementation Plan

C.1.1 DTS 70 IMPLEMENTATION PLAN CDRL A006, 21 JULY 1980, REVISIONS

REVISIONS

Graph II, page 13:

The data point must be revised for the MC 8080 A/B microprocessor using Signature Analysis with the DTS-70 system at Hughes Fullerton.

Later information revealed that test program time required through hardware verification increased. In addition a correction in the number of ICs for the 1635972 D/PCB is required which includes the MC 8080 A/B microprocessor. Therefore the data point changes as follows:

Item	Was	Is
MC 8080 A/B	38 IC	28 IC
Hughes Fullerton	14 Man Weeks	23 Man Weeks

Page 14 Paragraph B:

General Dynamics DTS-70; 8085

Was: "In the GD DTS-70 data point case, the 8085 was modeled directly with logic primitives."

Is: "In the GD DTS-70 data point case, the 8085 was functionally modeled."

This information was received during the Industry Demonstration from a representative of General Dynamics, Pomona, Ca.

REVISIONS

Page 14 Paragraph C:

HAC (DFI) DTS-70; 8080

Was: "Using the test technique outlined total programming time for this PCB on the DTS-70 required 14 man weeks. For the same PCB (38 ICs including the 8080), the GR-195 programming time is estimated at 19 man weeks or 36 percent longer."

Is: "Using the test technique outlined total programming time for this PCB on the DTS-70 required 23 man weeks. Relative to the GD 8085 data point, it is observed that the Signature Analysis functional test of the 8080 vs. functional modeling of the 8085 achieves a reduction in test program time in the ratio of 23/45 or very nearly 1:2.

Page 15 Paragraph E:

DTS-70; 2901

Subparagraph 3

Was: ". substantial reduction-----functionally tested instead."

Is: "An 8080 type LSI device when functionally tested using Signature Analysis can substantially reduce test programming time by a ratio approaching 1:2 as compared to a functionally modeled 8085 test program."