DAA/LANGLEY

DOMINION UNIVERSITY RESEARCH FOUNDATION

DEPARTMENT OF GEOLOGY
SCHOOL OF SCIENCES AND HEALTH PROFESSIONS
OLD DOMINION UNIVERSITY
NORFOLK, VIRGINIA 23508

TECHNICAL REPORT GSTR-86-3

THEORY AND OPERATION OF THE GOULD 32/27 PROGRAMS ABLE-2A AND EBLE FOR THE TROPOSPHERIC AIR MOTION MEASUREMENT SYSTEM

By

Carolyn Butler

Submitted by Earl C. Kindle, Principal Investigator

Final Report
For the period January 1, 1985 to December 31, 1985

Prepared for the
National Aeronautics and Space Administration
Langley Research Center
Hampton, VA 23665

Under
Research Grant NCC1-28
Dr. Edward Browell, Technical Monitor
Chemistry and Dynamics Branch

June 1986

DEPARTMENT OF GEOLOGY
SCHOOL OF SCIENCES AND HEALTH PROFESSIONS
OLD DOMINION UNIVERSITY
NORFOLK, VIRGINIA 23508

TECHNICAL REPORT GSTR-86-3

THEORY AND OPERATION OF THE GOULD 32/27 PROGRAMS ABLE-2A AND EBLE FOR THE TROPOSPHERIC AIR MOTION MEASUREMENT SYSTEM

By

Carolyn Butler

Submitted by Earl C. Kindle, Principal Investigator

Final Report
For the period January 1, 1985 to December 31, 1985

Prepared for the
National Aeronautics and Space Administration
Langley Research Center
Hampton, VA 23665

Under
Research Grant NCC1-28
Dr. Edward Browell, Technical Monitor
Chemistry and Dynamics Branch

Submitted by the
Old Dominion University Research Foundation
P.O. Box 6369
Norfolk, Virginia 23508-0369

June 1986

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

THEORY AND OPERATION OF THE GOULD 32/27 PROGRAMS ABLE-2A AND EBLE FOR THE TROPOSPHERIC AIR MTION MEASUREMENT SYSTEM

By
Carolyn Butler

1. INTRODUCTION

This report documents work performed under Research Grant NCCI-28 to develop software for the Tropospheric Air Motion Measurement System (TAMMS). The TAMMS is flown on the NASA/Goddard Flight Center Electra air-craft and serves to collect a number of different types of data that are essential to the various experiment objectives. Two projects have been conducted so far with the TAMM system operational. In July/August 1985, the Electra was flown to Brazil for ABLE-2A (Amazon Boundary Layer Experiment). During this experiment 19 missions were flown (including the transit flights). The TAMM system recorded position data, meteorology, flux measurements, aircraft motion and more. The 3-step programs written for this experiment are described in chapter 3. In December 1985, several flights were performed to assess the contamination and boundary layer of the Electra (Electra Boundary Layer Experiment). The Gould recorded position data, flow angles, pressure transducer measurements and more. The programs written for ABLE-2A had to be extensively modified due to timing considerations for this particular experiment. The 3-step programs written for EBLE are described in detail in chapter 4.

The programs have been written in Fortran and designed for hands-off operation. Power up and log-on procedures are discussed in chapter 5. Some additional information on Gould operation and some hints on program hang-ups and aborts are also given in this chapter. Once the computer is powered on and the program started the screen will update data every 5 seconds and the printer will print every ten seconds. The operator need only occasionally monitor the output data -- other than that no operator intervention is required. Fortran was used so that other programmers could easily understand the software. However, the use of interrupts and the need to interrogate peripheral devices (the RTP and CDS) requires the use of subroutines developed by Gould. These subroutines and descriptions of their calling variables or blocks are collected in the appendix (reproduced by permission of Gould).

Chapter 6 describes a few editing techniques for the user who may want to try some modification of the programs.

2. THE TROPOSPHERIC AIR MOTION MEASUREMENT SYSTEM

The TAMMS computer is a Gould 32/27 including an 80 Mbyte Winchester disk drive for program storage and retrieval, a 40 Mbyte cartridge tape drive for data storage, a CRT for communications and real-time data display, and a printer for real-time data presentation. The Gould interfaces to two data acquisition

devices: a Colorado Data Systems (CDS) 53A Smart Hardware System and a Computer Products, Inc. Real Time Peripherals Universal Input/Output Controller (RTP). These will be referred to as the CDS and RTP systems throughout this report. Two RTP's were daisy-chained together during EBLE. The CDS contains ARINC cards which allow the host computer to access inputs from the aircraft's navigation system The CDS is connected to the Gould through an IEEE interface by fiber optics. The RTP's con-tain analog input, gate, analog to digital converters, pulse counters and synchro converter cards. The RTP is connected to the Gould through an analog/digital interface board via a high speed serial link. Figure 1 shows a schematic of the TAMMS.

The programs written for both ABLE-2A and EBLE each consist of three programs which run concurrently. The source files for ABLE-2A are stored under the file names BZL1, BZL2 and BZL3 and the source files for EBLE are stored under the names EBLE1, EBLE2, and EBLE3. The load images of either ABLE-2A or EBLE programs are stored under STEP1, STEP2, and STEP3. STEP1 is the main program ("MAINPROG") which the user will execute. This pro-gram activates the interrupt driver ("SETINTRP") then waits for a buffer of data to be sent from the third program ("ACQUISIT"). On receiving data, MAINPROG will do necessary conversions and print and display the values of interest. File STEP2 is the short program SETINTRP which uses the Gould internal clock to cause an interrupt at specified intervals. SETINTRP will activate the program ACQUISIT then suspend itself until the next interrupt occurs at which time it will run the program ACQUISIT again, and so on. STEP3 is the program ACQUISIT which performs all the data acquisition from the RTP (subroutine GETRTP) and from the CDS (subroutine GETCDS), buffers the data into one long record of 20 data sets which it then stores on cartridge tape, and sends buffers back to MAINPROG for real-time display. A double-buffering technique is used where one buffer is being filled while the other is being written to tape (each buffer consists of 20 data sets).

Each source file contains the necessary job control commands before and after the listing such that any of the 6 files BZLi or EBLEi can be compiled, catalogued and stored under the file names STEPi (i = 1, 2, or 3). More on program modification will be discussed in chapter 6.

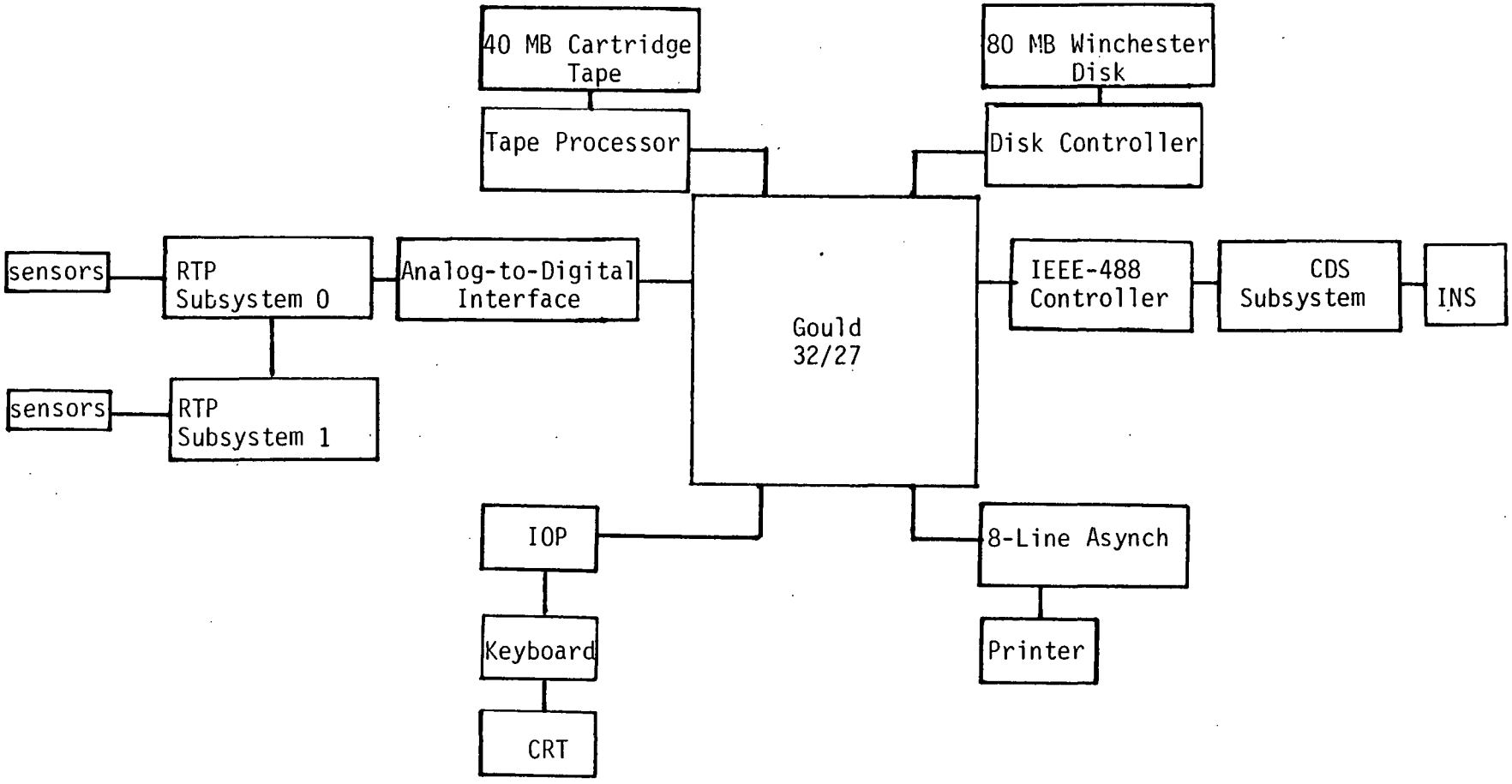


Figure 1. Schematic of the Troposheric Air Motion Measurement System.

3. ABLE-2A PROGRAMS

The ABLE-2A programs consist of three programs which run simultaneously. The source listings reside in the three files BZL1, BZL2 and BZL3 which when compiled and catalogued are called STEP1, STEP2 and STEP3. The operator executes the file STEP1 (or the batch file GTE which will be discussed in chapter 5). STEP1 automatically activates STEP2 and STEP2 activates STEP3. Flow charts for the three programs are shown in figures 2 and 3.

The data sampled during ABLE-2A consisted of five groups of data -- one from the CDS and four from the RTP. One of the data sets from the RTP was sampled at 20 Hz while the other three groups from the RTP as well as the CDS group were sampled at 5 Hz. The interrupt into ACQUISIT was set at 3 tics (1/20 th sec) at which time a fast set and a slow set was sampled:

```
interrupt #1        retrieve fast data set
                    retrieve slow data set group A
interrupt #2        retrieve fast data set
                    retrieve slow data set group B
interrupt #3        retrieve fast data set
                    retrieve slow data set group C
interrupt #4        retrieve fast data set
                    retrieve CDS data set
```

Each data set is preceded by 4 bytes of time information pinpointing the exact time (hours, minutes, seconds and tics) of that data set. The format and description of the complete data set is shown in Table 1.

Due to a timing problem in the handshake of the Gould IEEE-488 interface and the CDS communication card, the subroutine GETCDS was set up to run independently of the data retrieval. On the first pass through GETCDS control blocks are intialized and the CDS is interrogated for data using the "no-wait" option. That is, the commands are sent to the IEEE controller and the subroutine GETCDS does a return to the main program ACQUISIT. When the IEEE controller signals the program that the data from the CDS is available the program interrupts to a routine that loads the latest data into a storage buffer. The routine then sets up for another CDS interrogation and exits. Meanwhile, when ACQUISIT wants the latest CDS data it retrieves that data from the storage buffer. The four time bytes preceding the CDS data in the data array point to that time when the data was last updated. This handshaking problem will later be eliminated by the purchase of a new programmable communications card for the CDS.

Figure 4 shows the RTP system set-up during the ABLE-2A experiment. Program listings of the files BZL1, BZL2, and BZL3 follow.

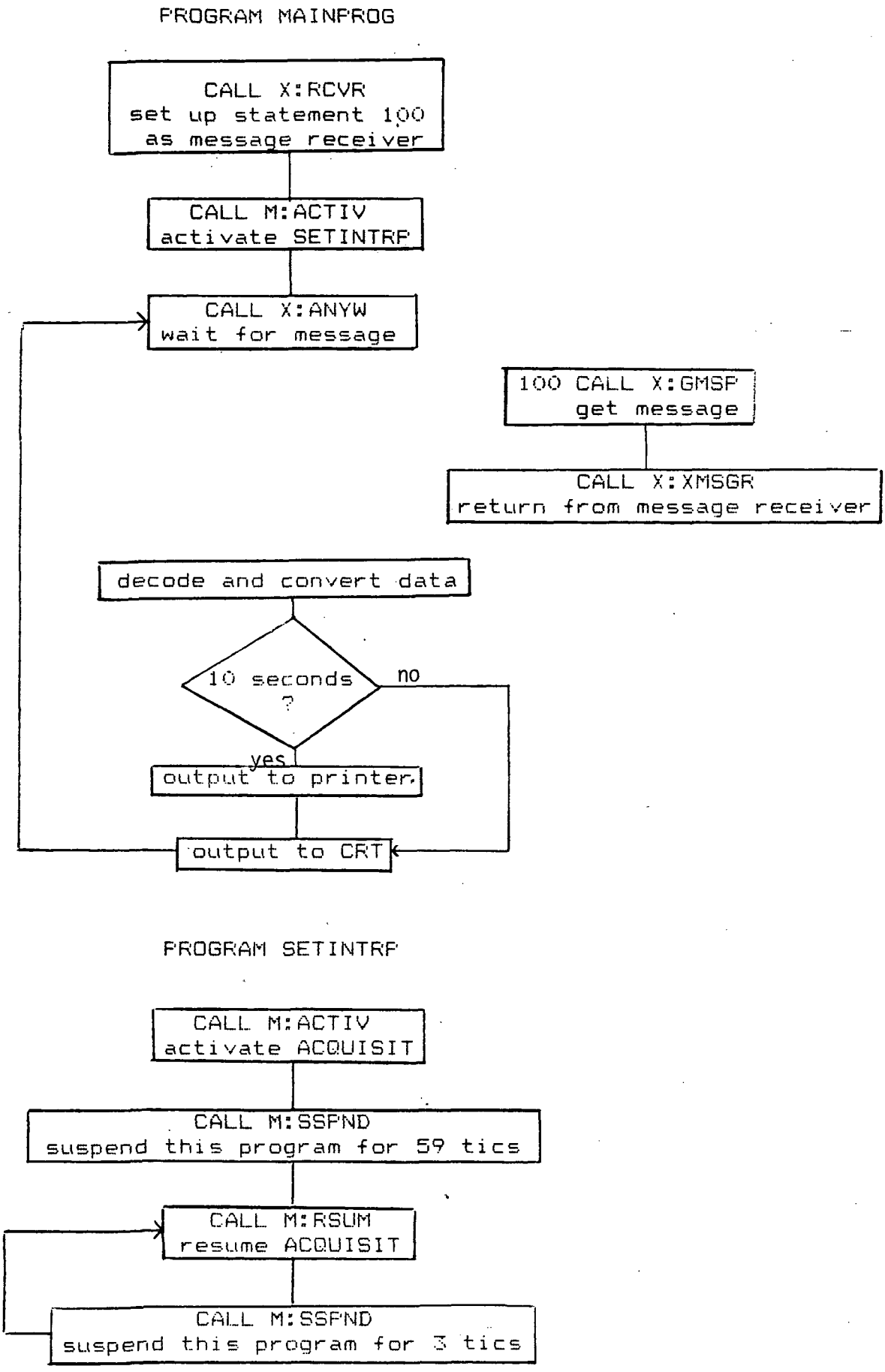


Figure 2. Flowcharts of ABLE-2A versions of MAINPROG and SETINTRP.

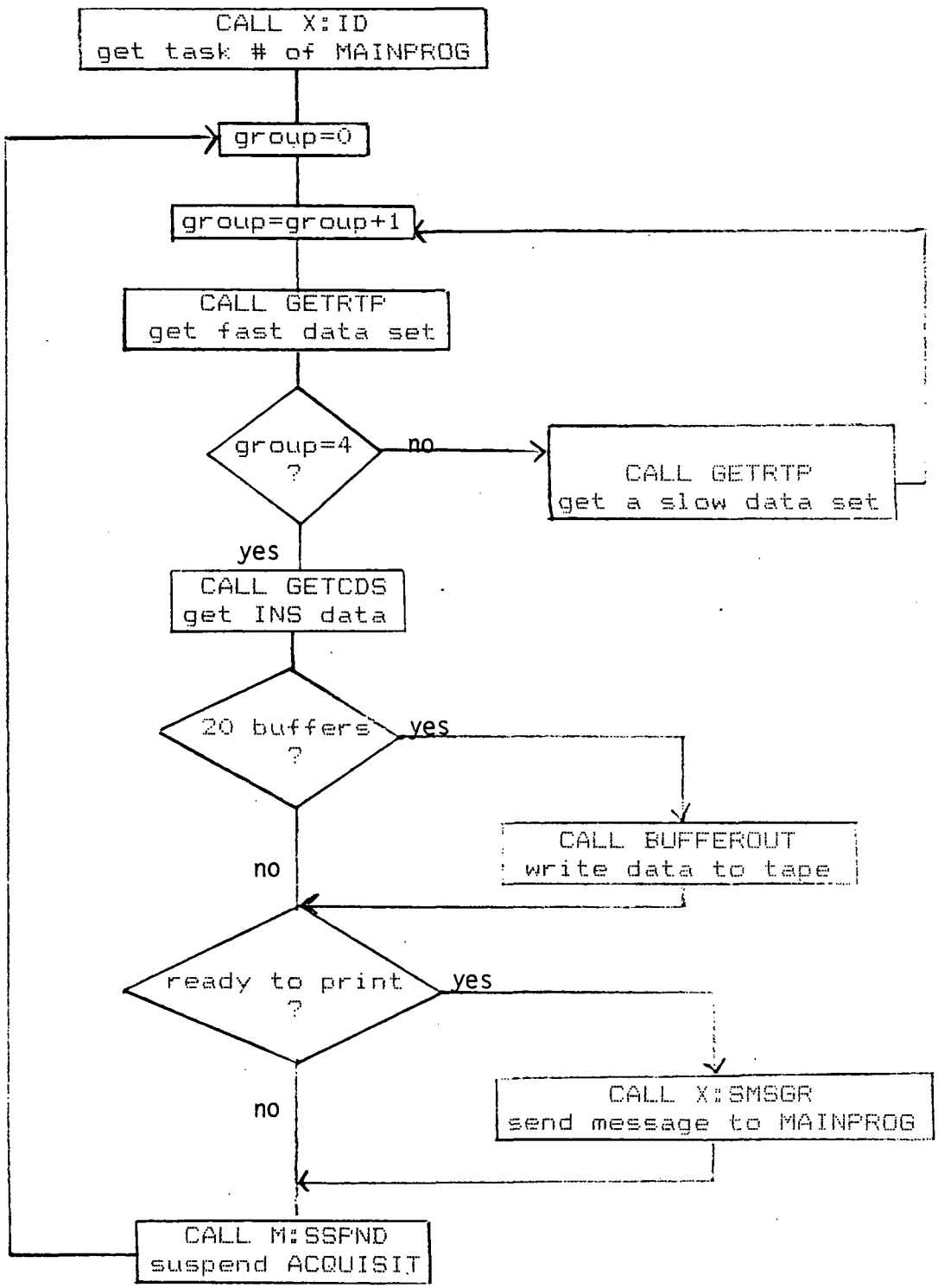


Figure 3. Flowchart of ABLE-2A version of ACQUISIT.

Table 1. ABLE-2A Data Format

RTP Fast Data Set

Location in Array	# Bytes	Description	Source
1,51,103,157	4	time	Gould clock
5,55,107,161	2	B-vane	Synchro #1
7,57,109,163	2	A-vane	Synchro #2
9,59,111,165	2	Pitch	Synchro #3
11,61,113,167	2	Roll	Synchro #4
13,63,115,169	2	Platform Heading	Synchro #5
15,65,117,171	2	Heading	Synchro #6
17,67,119,173	2	Static Pressure	Pulse Counter #1
19,69,121,175	2	Diff. Pressure	Pulse Counter #2
21,71,123,177	2	CO	A/D #1
23,73,125,179	2	O3	A/D #2
25,75,127,181	2	-----	A/D #3
27,77,129,183	2	-----	A/D #4

RTP Slow Data Set - Group A

Location in Array	# Bytes	Description	Source
29	4	time	Gould clock
33	2	baro-alt (fine)	Synchro #7
35	2	baro-alt (coarse)	Synchro #8
37	2	temp (fast)	A/D #5
39	2	temp (normal)	A/D #6
41	2	dew point (high)	A/D #7
43	2	dew point (low)	A/D #8
45	2	abs. pressure	A/D #9
47	2	diff. pressure	A/D #10
49	2	surf. temp (PRT-5)	A/D #11

RTP Slow Data Set - Group B

Location in Array	# Bytes	Description	Source
79	4	time	Gould clock
83	2	UV flux up	A/D #12
85	2	UV flux down	A/D #13
87	2	altitude	A/D #14
89	2	----	A/D #15
91	2	DPT warn signal	A/D #16
93-		not used	A/D #17 - #21

RTP Slow Data Set - Group C (A/D #22 - #32 not used)

CDS Slow Data Set

Location in Array	# Bytes	Description	Label
185	4	time	Gould clock
189	4	latitude	8
193	4	longitude	9
201	4	ground speed	A
205	4	track angle	B
209	4	true heading	C

CDS Data set (con't)

Location in Array	# Bytes	Description	Label
213	4	wind speed	D
217	4	wind dir	E

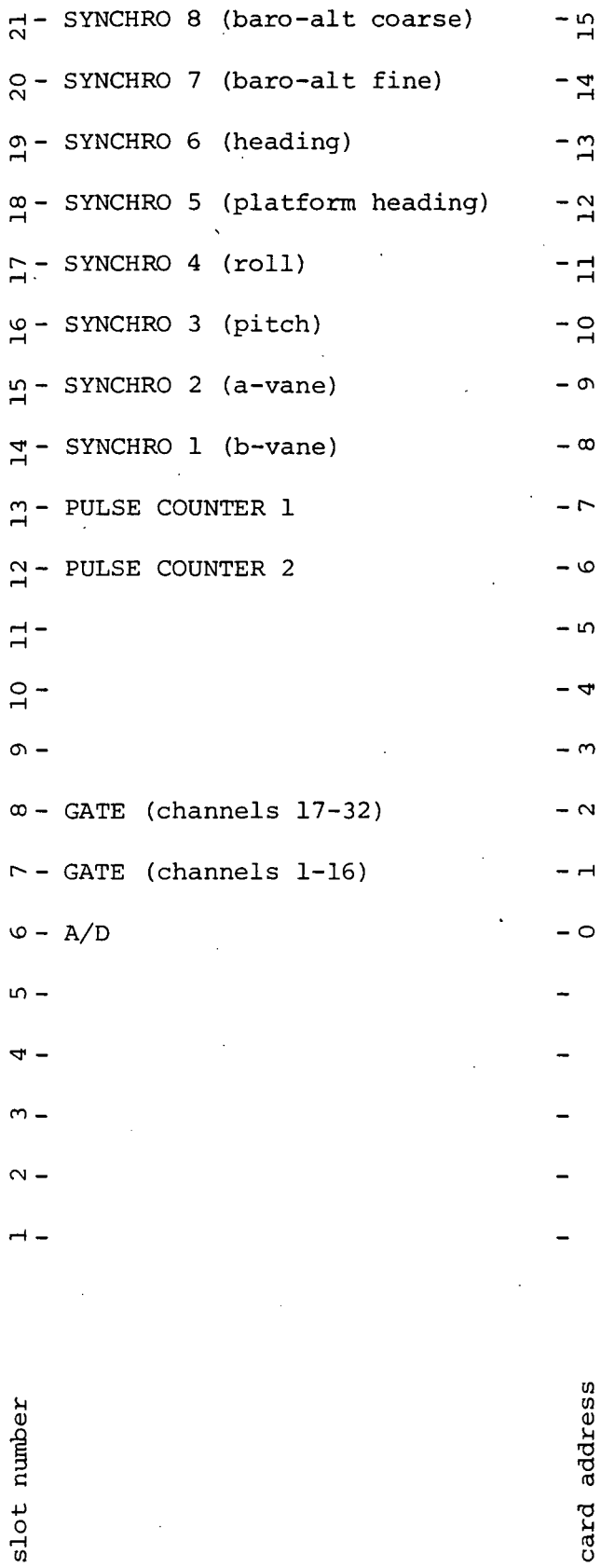


Figure 4. RTP card cage set-up for ABLE-2A.

```
$JOB MAINPROG
$ASSIGN LO TO SLOFOR
$OPTION 5
$EXECUTE FORTRAN
      PROGRAM MAINPROG
      INTEGER*8 ITASK
      INTEGER*1 ITIME(3)
      INTEGER*2 TRHD,WARN
      INTEGER*4 LAT,LON,LST1,LST2
      INTEGER   WS,WD,TRKANG,TRUHD,GRDSPD
      INTEGER*2 CO,ABSALT,BARALT,TEMP,DPT,PRT,UVUP,UVDN,O3
      INTEGER*2 FAST(12),GRPA(9),GRPB(8),GRPC(11),FAST2(12)
      INTEGER*1 GRPD(32)
      INTEGER FCB(16),PRB(5),RXB(2)              !PARAMETER BLOCKS
      INTEGER*1 IA(280)
      DIMENSION SYNC(6)
      EQUIVALENCE (SYNC(1),BETA)
      EQUIVALENCE (SYNC(2),ALPHA)
      EQUIVALENCE (IA(1),ITIME(1))
      EQUIVALENCE (IA(21),CO)
      EQUIVALENCE (IA(23),O3)
      EQUIVALENCE (IA(35),BARALT)
      EQUIVALENCE (IA(37),TEMP)
      EQUIVALENCE (IA(41),DPT)
      EQUIVALENCE (IA(49),PRT)
      EQUIVALENCE (IA(83),UVUP)
      EQUIVALENCE (IA(85),UVDN)
      EQUIVALENCE (IA(189),LAT)
      EQUIVALENCE (IA(193),LON)
      EQUIVALENCE (IA(213),WS)
      EQUIVALENCE (IA(217),WD)
      EQUIVALENCE (IA(15),TRHD)
      EQUIVALENCE (IA(201),GRDSPD)
      EQUIVALENCE (IA(205),TRKANG)
      EQUIVALENCE (IA(209),TRUHD)
      EQUIVALENCE (IA(91),WARN)
      EQUIVALENCE (IA(5),FAST(1))          !FIRST SET OF FAST DATA
      EQUIVALENCE (IA(55),FAST2(1))        !SECOND SET OF FAST DATA
      EQUIVALENCE (IA(33),GRPA(1))
      EQUIVALENCE (IA(83),GRPB(1))
      EQUIVALENCE (IA(135),GRPC(1))
      EQUIVALENCE (IA(189),GRPD(1))
      INTEGER*1 DONEFLAG(4)     !FLAG FOR MORE DATA TO PRINT
      CALL X:RCVR(*100,ISTAT,*995)  !WHEN DATA IS RECEIVED GOTO 100
      PRB(1)=ADDR(IA)           !ADDRESS OF ARRAY TO BE RECEIVED
      PRB(2)='00DC0000'X        !# BYTES THAT ARE RECEIVED FROM ACQUISIT
      ITASK='SETINTRP'          !ROUTINE THAT SETS INTERRUPTS
      CALL M:ACTIV(ITASK,ISTAT,*997)   !ACTIVATE INTERRUPT ROUTINE
90    CALL X:ANYW(0,ISTAT,*996)        !WAIT FOR ACQUISIT TO SEND DATA
C**** LAT AND LON ARE FROM OMEGA -- HEX FORMAT.............
```

```
      LAT=IBITS(LAT,0,21)
      TRHD=IBITS(TRHD,2,14)
      XTRHD=360.-TRHD*.0219727
      LALT=IBITS(BARALT,2,14)
      XALTC=LALT*.0219727
      XALTC=(360.-XALTC)*375.
C **** DATA OUTPUT TO PRINTER.....................................
      IF(LINE.EQ.0.AND.ITEN.EQ.2)WRITE(2,2000)
2000  FORMAT(1HO,4X,'TIME',5X,'LAT',5X,'LON',1X,'TRK ANG',6X,'WS',
     1       6X,'WD',              4X,'BARALT',7X,'SAT',
     2       7X,'DPT',7X,'PRT',8X,'CO'  ,5X,'OZONE',5X,'UV UP',
     3       5X,'UV DN')
      XTEMP=20.*6.25E-4*TEMP-50.
      XDPT=15.*6.25E-4*DPT-75.
      XPRT=10.*6.25E-4*PRT-10.
      XCO=6.25E-4*CO
      XO3=.105*O3
      XUVUP=439.0*6.25E-4*UVUP
      XUVDN=840.0*6.25E-4*UVDN
       DO 555 I=1,6
       FAST(I)=IBITS(FAST(I),2,14)
       SYNC(I)=FAST(I)*.0219727
555    CONTINUE
      SYNC(5)=360.-SYNC(5)
      SYNC(6)=360.-SYNC(6)
      I1=     IBITS(FAST(7),0,16)
      I2=     IBITS(FAST(8),0,16)
      J1=     IBITS(FAST2(7),0,16)
      J2=     IBITS(FAST2(8),0,16)
      FRE1=J1-I1
      FRE2=J2-I2
      IF(FRE1.LT.0)FRE1=FRE1+65535
      IF(FRE2.LT.0)FRE2=FRE2+65535
C
      F1=25.85312/(.05E6/FRE1)
      F2=26.81997/(.05E6/FRE2)
      FRE1=162.923*(1.-F1)-91.9366*(1.-F1)**2
      FRE2=53.3395*(1.-F2)-27.2355*(1.-F2)**2
      XM2=5.*((FRE2/FRE1+1)**.2857-1)
      SAT=(XTEMP+273.16)/(1.+.2*XM2)
      SAT=SAT-273.16
      XM=0.
      IF(XM2.GT.0)XM=SQRT(XM2)
      TAS=38.967*XM*SQRT(SAT  +273.16)
      IF (ITEN.NE.2)GOTO 444
      ITEN=0
C **** DATA OUTPUT TO PRINTER .....................................
      WRITE(2,2005)ITIME, LAT, LON,TRKANG ,WS,WD,XALTC,
     1             SAT,XDPT,XPRT,XCO,XO3,XUVUP,XUVDN
2005  FORMAT(1X,2(I2,':'),I2,2(2X,Z6),       3(2X,Z6) ,8F10.2)
      LINE=LINE+1
```

```
      IF(LINE.EQ.49)LINE=0
444   ITEN=ITEN+1
C **** DATA OUTPUT TO CRT MONITOR ..............................
      WRITE (3,3000) ITIME
3000  FORMAT (' TIME  ',2(I2,':'),I2,5X,'********************')
      WRITE (3,3020)XALTC,SAT  ,XDPT,XPRT
3020  FORMAT (' ALT ',F6.0,'  SAT ',F5.1,'  DPT ',F5.1,
     1        '  PRT ',F5.1)
      WRITE (3,3030)TRKANG,TRUHD,GRDSPD,TAS
3030  FORMAT (' TK ',Z6,'  TH ',Z6,'  GS ',Z6,'  TAS ',F4.0)
      WRITE (3,3040)LAT,LON,WS,WD
3040  FORMAT (' LAT ',Z6,'  LNG ',Z6,'  WS ',Z6,'  WD ',Z6)
      WRITE (3,3050)ALPHA,BETA,WARN
3050  FORMAT (' ALPHA ',F8.3,'  BETA ',F8.3,'   WARN ',I5)
      GOTO 90
C  THE FOLLOWING IS THE RECEIVER BUFFER ROUTINE
100   CALL X:GMSGP(PRB,ISTAT,*994)          !GET MESSAGE (BUFFER)
      LSTAT=IBITS(PRB(1),0,8)
      RXB(1)=LSTAT+ADDR(DONEFLAG)
      RXB(2)=2
      CALL X:XMSGR(RXB,ISTAT,*993)          !EXIT THIS ROUTINE
C ****   ERROR RETURNS AND MESSAGES   ****
993   WRITE (3,2045)ISTAT
2045  FORMAT(' X:XMSGR ERROR:   ',I5)
      GOTO 999
994   WRITE(3,2040)ISTAT
2040  FORMAT(' X:GMSGP ERROR:   ',I5)
      GOTO 999
995   WRITE(3,2035)ISTAT
2035  FORMAT(' X:RCVR ERROR:    ',I5)
      GOTO 999
996   WRITE(3,2030)ISTAT
2030  FORMAT(' X:ANYW ERROR:    ',I5)
      GOTO 999
997   WRITE(3,2020)ISTAT
2020  FORMAT(' M:ACTIV ERROR:   ',I5)
      GOTO 999
998   WRITE(3,2015)FCB(4)
2015  FORMAT(' I/O ERROR, FCB(4)=',Z8)
999   STOP
      END
$ASSIGN SLO TO SLODMP
$ASSIGN DIR TO IEEEDIR
$ASSIGN LIB TO IEEELIB
$EXECUTE CATALOG
ASSIGN SI TO SYC
ASSIGN 2 TO DEV=TY7EC6
ASSIGN 3 TO LFC=UT
CATALOG MAINPROG
$EOJ
```

```
$$
$JOB SETINTRP
$NOTE SET TIMING INTERRUPT FOR 20 TIMES/SEC
$ASSIGN LO TO SLOFOR
$OPTION 5
$EXECUTE FORTRAN
      PROGRAM SETINTRP
      INTEGER*8 ITASK2
      ITASK2='ACQUISIT'
      CALL M:ACTIV(ITASK2,*999)    !ACTIVATE ACQUISIT ROUTINE
      CALL M:SSPND(59,*999)        !WAIT WHOLE MINUTE FIRST TIME
10    CALL M:RSUM(ITASK2,*999)     !RESUME DATA ACQUISITION
15    CALL M:SSPND(3,*999)         !SUSPEND THIS JOB FOR 3 TICS
      GOTO 10
999   CONTINUE
      WRITE(6,6000)
6000  FORMAT(' SETINTRP ERROR')    !GETS HERE ONLY IF ERROR
      STOP
      END
$ASSIGN SLO TO SLODMP
$EXECUTE CATALOG
ASSIGN 6 TO INTERR
CATALOG SETINTRP
$EOJ
```

```
$$
$JOB ACQUISIT
$NOTE DATA ACQUISITION PROGRAM
$ASSIGN LO TO SLOFOR
$OPTION 5
$EXECUTE FORTRAN
      PROGRAM ACQUISIT
      INTEGER*3 TASKNAM,OWNRNAM,PSEUDO
      INTEGER GROUP
      INTEGER*1 IA(220,40)
      INTEGER        PSB(8),RBA(2)
C     ************    DETERMINE TASK # OF MAIN PROGRAM
      TASKNAM='MAINPROG'
      INDEX=0
      ITASK=0
      OWNRNAM=0
      PSEUDO=0
      CALL X:ID(INDEX,ITASK,TASKNAM,OWNRNAM,PSEUDO,ISTAT,*900)
C        ************       SET UP PSB FOR LATER TRANSMISSION
       PSB(1)=ITASK     !TASK # "MAINPROG" FROM X:ID
       PSB(2)=0         !MUST BE 0
       PSB(3)=220   !# BYTES TO TRANSMIT
       PSB(5)=Z'100000'   !# BYTES TO RECEIVE  (2)
       PSB(6)=ADDR(RBA)   !PURE ADDRESS OF RECEIVER BUFFER
       PSB(7)=ADDR($800) !NO WAIT ERROR RETURN
      GROUP=0             !
      IPRINT=1          !PRINT CONTROL FLAG
      MTERRS=0          !# MAG TAPE ERRORS
      II=1              !INDEX THRU IA BUFFER
      JJ=1              !INDEX THRU 40 IA BUFFERS
      IFLAG=0         !TAPE DONE FLAG
100      GROUP=GROUP+1
      CALL GETRTP(IA,II,JJ,0)  !FAST DATA SET
        IF(GROUP.LT.4)CALL GETRTP(IA,II,JJ,GROUP)  !GET A SLOW DATA SET
        IF(GROUP.EQ.4)CALL GETCDS(IA,II,JJ)        !GET CDS DATA SET
        IF(GROUP.EQ.4)GROUP=0          !4 RTP GROUPS AND 1 CDS GROUP
        IF(II.LT.200)GOTO 500          !DON'T HAVE ALL THE DATA YET
        II=1                        !RESET ARRAY INDEX BACK TO BEGINNING
        IF (JJ.EQ.20)GOTO 420  !HAVE WE GOT 20 IA BUFFERS YET?
        IF (JJ.EQ.40)GOTO 420  !HAVE WE GOT 20 IA BUFFERS YET?
        GOTO 430
420     IF (IFLAG.EQ.1)GOTO 905        !TAPE READY?
        IFLAG=1                        !TAPE WILL NOT BE READY NOW
        CALL BUFFEROUT(6,1,IA(1,JJ-19),1100,$305,$910) !WRITE TO TAPE
430     IPRINT=IPRINT+1          !INCREMENT PRINT FLAG
        IF(IPRINT.LT.25)GOTO 450    !SEND BUFFER EVERY 5 SECONDS
        IPRINT=1           !READY TO SEND -- RESET PRINT FLAG
        PSB(4)=ADDR(IA(1,JJ))     !ADDRESS OF LAST DATA BUFFER
        CALL X:SMSGR(PSB,ISTAT,*915)    !SEND BUFFER TO MAINPROG
450     JJ=JJ+1          !NEXT IA BUFFER
```

```
 500    CALL M:SSPND(0,*920)     !SUSPEND THIS PROGRAM UNTIL INTERRUPT
        GOTO 100          !GOTO 100 WHEN INTERRUPT OCCURS FROM SETINTRP
C    *********   END ACTION ROUTINES   ****
 800    CALL X:XMEA      !EXIT SEND MESSAGE
 805    IFLAG=0                  !MAG TAPE DONE
        CALL STATUS(6,ISTAT)
        IF(ISTAT.NE.2) WRITE(5,8005)ISTAT
 8005   FORMAT(' MAGTAPE STATUS ERROR: ',I5)
        CALL X:XNWIO
C     *********   ERROR MESSAGES OUTPUT TO FILE "ACQERR"
 900    WRITE(5,9000)ISTAT
 9000   FORMAT(' X:ID ERROR: ',I5)
        GOTO 999
 905    MTERRS=MTERRS+1
        WRITE(5,9005)MTERRS
 9005   FORMAT(' MAGTAPE BUSYF FLAG SET, ERRORS=',I5)
        GOTO 500
 910    WRITE(5,9010)
 9010   FORMAT(' MAGTAPE HARDWARE ERROR')
        GOTO 999
 915    WRITE(5,9015)ISTAT
 9015   FORMAT(' X:SMSGR ERROR: ',I5)
        GOTO 999
 920    WRITE(5,9020)
 9020   FORMAT(' M:SSPND ERROR')
        GOTO 999
 999    STOP
        END
        SUBROUTINE GETRTP(IA,II,JJ,GROUP)
        INTEGER KPTS(5)
        INTEGER GROUP
        INTEGER*1 IA(220,40),ITIME(4)
C
       INTEGER IOCHF(3),IOCBF(70),SIZEF
       INTEGER IOCHA(3),IOCBA(100),SIZEA
       INTEGER IOCHB(3),IOCBB(130),SIZEB
       INTEGER IOCHC(3),IOCBC(142),SIZEC
       INTEGER BUSY
       INTEGER*2 ICHNF(24),OUTF(40)
       INTEGER*2 ICHNA(24),OUTA(40)
       INTEGER*2 ICHNB(24),OUTB(40)
       INTEGER*2 ICHNC(24),OUTC(40)
       INTEGER*2 INP(20)
       INTEGER*1 IDATA(40)
       INTEGER ADD1,ADD2,ADD3,ADD4,ADD5,ADD6
       INTEGER ADD7,ADD8,ADD9,ADD110,ADD11,ADD12
       EQUIVALENCE (IDATA,INP(1))
       DATA KPTS/12,9,10,11,15/
C
       IG=GROUP+1
       IPTS=KPTS(IG)
```

```
      IF(INITDONE.GE.1) GOTO 10
C ************   INITIALIZE SECTION   ***
C ******   SET UP IOCH/IOCB FOR EACH GROUP
 1    ADD1=LOCF(ICHNF)
      ADD2=LOCF(INP)
      ADD3=LOCF(OUTF)
      IOCHF(1)=Z'00300000'          !IOM ADDRESS IS 30
      IOCHF(2)=ADDR(IOCBF)
      IOCHF(3)=Z'20002'             !NOP INSTRUCTION
      IOCBF(1)=Z'E2000006'  !SYNCHROS: IN,SEQ,DIG; 6 HALFWORDS
      IOCBF(2)=ADD1
      IOCBF(3)=ADD2
      IOCBF(4)=0
      IOCBF(5)=Z'E2000002'  !PULSE COUNTERS: IN,SEQ,DIG; 2 HALFWORDS
      IOCBF(6)=ADD1+2
      IOCBF(7)=ADD2+12
      IOCBF(8)=0
      L1=1
      L2=16
      L3=0
      LL=0
      DO 101 I=1,4          !A/D CHANNELS 1-4: REQUIRES BOTH DO AND DI
      IOCBF(9+LL)=Z'22000001'  !OUT,RAN,DIG
      IOCBF(10+LL)=ADD1+L1
      IOCBF(11+LL)=ADD3+L3
      IOCBF(12+LL)=0
      IOCBF(13+LL)=Z'A2000001' !IN ,RAN,DIG
      IOCBF(14+LL)=ADD1+L1+2
      IOCBF(15+LL)=ADD2+L2
      IOCBF(16+LL)=0
      LL=LL+8
      L1=L1+4
      L2=L2+2
      L3=L3+2
101   CONTINUE
      IOCBF(37)=Z'A0000001'   !NO COMMAND CHAIN ON LAST IOCB
      ICHNF(1)=8              !SYNCHRO SLOT ADDRESS IS 8
      ICHNF(2)=6              !PULSE COUNTER SLOT ADDRESS IS 6
      DO 102 I=3,10,2         !A/D
      ICHNF(I)=Z'1000'  !BYPASS READY TEST; A/D IS IN SLOT 0
      ICHNF(I+1)=0
102   CONTINUE
      OUTF(1)=Z'2080'   !GAIN GARD SLOT 1, CHANNEL 0, GAIN 1
      OUTF(2)=Z'3080'   !GAIN CARD SLOT 1, CHANNEL 1, GAIN 1
      OUTF(3)=Z'2090'   !GAIN CARD SLOT 1, CHANNEL 2, GAIN 1
      OUTF(4)=Z'3090'   !GAIN CARD SLOT 1, CHANNEL 3, GAIN 1
      SIZEF=70
      BUSY=LOCF($800)
      CALL ADIUSR(IOCHF,IOCBF,SIZEF,BUSY) !INITIALIZE FOR GROUP 1
      WRITE (5,5991)
      ADD4=LOCF(ICHNA)
```

```
      ADD5=LOCF(INP)
      ADD6=LOCF(OUTA)
      IOCHA(1)=Z'00300000'     !IOM ADDRESS OF 30
      IOCHA(2)=ADDR(IOCBA)
      IOCHA(3)=Z'20002'       !NOP
      IOCBA(1)=Z'E2000002'  !SYNCHROS: IN,SEQ,DIG; 2 HALFWORDS
      IOCBA(2)=ADD4
      IOCBA(3)=ADD5
      IOCBA(4)=0
      L1=2
      L2=4
      L3=0
      LL=0
      DO 201 I=1,7          !A/D: NEED BOTH D0 AND D1
      IOCBA(5+LL)=Z'22000001' !OUT,RAN,DIG; 1 HALFWORD
      IOCBA(6+LL)=ADD4+L1
      IOCBA(7+LL)=ADD6+L3
      IOCBA(8+LL)=0
      IOCBA(9+LL)=Z'A2000001' !IN ,RAN,DIG; 1 HALFWORD
      IOCBA(10+LL)=ADD4+L1+2
      IOCBA(11+LL)=ADD5+L2
      IOCBA(12+LL)=0
      LL=LL+8
      L1=L1+4
      L2=L2+2
      L3=L3+2
  201  CONTINUE
      IOCBA(57)=Z'A0000001'  !LAST IOCB DOES NOT CHAIN
      ICHNA(1)=Z'E'      !SYNCHRO IN SLOT 14
      DO 202 I=2,14,2
      ICHNA(I)=Z'1000' !BYPASS READY TEST; A/D CARD IN SLOT 0
      ICHNA(I+1)=0
  202  CONTINUE
      OUTA(1)=Z'20A0'   !GAIN CARD SLOT 1, CHANNEL 4, GAIN 1
      OUTA(2)=Z'30A0'   !GAIN CARD SLOT 1, CHANNEL 5, GAIN 1
      OUTA(3)=Z'20B0'   !GAIN CARD SLOT 1, CHANNEL 6, GAIN 1
      OUTA(4)=Z'30B0'   !GAIN CARD SLOT 1, CHANNEL 7, GAIN 1
      OUTA(5)=Z'20C0'   !GAIN CARD SLOT 1, CHANNEL 8, GAIN 1
      OUTA(6)=Z'30C0'   !GAIN CARD SLOT 1, CHANNEL 9, GAIN 1
      OUTA(7)=Z'20D0'   !GAIN CARD SLOT 1, CHANNEL 10, GAIN 1
      SIZEA=100
      CALL ADIUSR(IOCHA,IOCBA,SIZEA,BUSY)  !INITIALIZE GROUP 2
      WRITE (5,5992)
3     ADD7=LOCF(ICHNB)
      ADD8=LOCF(INP)
      ADD9=LOCF(OUTB)
      IOCHB(1)=Z'00300000'    !IOM ADDRESS OF 30
      IOCHB(2)=ADDR(IOCBB)
      IOCHB(3)=Z'20002'            !NOP
      L1=0
      L2=0
```

```
      L3=0
      LL=0
      DO 301 I=1,10     !A/D:  NEED BOTH DO AND DI
      IOCBB(1+LL)=Z'22000001' !OUT,RAN,DIG; 1 HALFWORD
      IOCBB(2+LL)=ADD7+L1
      IOCBB(3+LL)=ADD9+L3
      IOCBB(4+LL)=0
      IOCBB(5+LL)=Z'A2000001' !IN ,RAN,DIG; 1 HALFWORD
      IOCBB(6+LL)=ADD7+L1+2
      IOCBB(7+LL)=ADD8+L2
      IOCBB(8+LL)=0
      LL=LL+8
      L1=L1+4
      L2=L2+2
      L3=L3+2
301    CONTINUE
      IOCBB(77)=Z'A0000001'          !NO CHAIN ON LAST IOCB
      DO 302 I=1,19,2
      ICHNB(I)=Z'1000'      !BYPASS READY TEST; A/D IN SLOT 0
      ICHNB(I+1)=0
302    CONTINUE
      OUTB(1)=Z'30D0'   !GAIN CARD SLOT 1, CHANNEL 11, GAIN 1
      OUTB(2)=Z'20E0'   !GAIN CARD SLOT 1, CHANNEL 12, GAIN 1
      OUTB(3)=Z'30E0'   !GAIN CARD SLOT 1, CHANNEL 13, GAIN 1
      OUTB(4)=Z'20F0'   !GAIN CARD SLOT 1, CHANNEL 14, GAIN 1
      OUTB(5)=Z'30F0'   !GAIN CARD SLOT 1, CHANNEL 15, GAIN 1
      OUTB(6)=Z'2100'   !GAIN CARD SLOT 2, CHANNEL 0, GAIN 1
      OUTB(7)=Z'3100'   !GAIN CARD SLOT 2, CHANNEL 1, GAIN 1
      OUTB(8)=Z'2110'   !GAIN CARD SLOT 2, CHANNEL 2, GAIN 1
      OUTB(9)=Z'3110'   !GAIN CARD SLOT 2, CHANNEL 3, GAIN 1
      OUTB(10)=Z'2120'  !GAIN CARD SLOT 2, CHANNEL 4, GAIN 1
      SIZEB=130
      CALL ADIUSR(IOCHB,IOCBB,SIZEB,BUSY) !INITIALIZE GROUP 3
      WRITE (5,5993)
4     ADD10=LOCF(ICHNC)
      ADD11=LOCF(INP)
      ADD12=LOCF(OUTC)
      IOCHC(1)=Z'00300000'      !IOM ADDRESS OF 30
      IOCHC(2)=ADDR(IOCBC)
      IOCHC(3)=Z'20002'         !NOP
      L1=0
      L2=0
      L3=0
      LL=0
      DO 401 I=1,11         !A/D: NEEDS BOTH DO AND DI
      IOCBC(1+LL)=Z'22000001'    !OUT,RAN,DIG; 1 HALFWORD
      IOCBC(2+LL)=ADD10+L1
      IOCBC(3+LL)=ADD12+L3
      IOCBC(4+LL)=0
      IOCBC(5+LL)=Z'A2000001'    !IN,RAN,DIG; 1 HALFWORD
```

```
      IOCBC(7+LL)=ADD11+L2
      IOCBC(8+LL)=0
      LL=LL+8
      L1=L1+4
      L2=L2+2
      L3=L3+2
  401  CONTINUE
      IOCBC(85)=Z'A0000001'          !NO CHAIN ON LAST IOCB
      DO 402 I=1,21,2      !
      ICHNC(I)=Z'1000'  !BYPASS READY TEST; A/D SLOT 0
      ICHNC(I+1)=0
  402  CONTINUE
      OUTC(1)=Z'3120'  !GAIN CARD SLOT 2, CHANNEL 5, GAIN 1
      OUTC(2)=Z'2130'  !GAIN CARD SLOT 2, CHANNEL 6, GAIN 1
      OUTC(3)=Z'3130'  !GAIN CARD SLOT 2, CHANNEL 7, GAIN 1
      OUTC(4)=Z'2140'  !GAIN CARD SLOT 2, CHANNEL 8, GAIN 1
      OUTC(5)=Z'3140'  !GAIN CARD SLOT 2, CHANNEL 9, GAIN 1
      OUTC(6)=Z'2150'  !GAIN CARD SLOT 2, CHANNEL 10, GAIN 1
      OUTC(7)=Z'3150'  !GAIN CARD SLOT 2, CHANNEL 11, GAIN 1
      OUTC(8)=Z'2160'  !GAIN CARD SLOT 2, CHANNEL 12, GAIN 1
      OUTC(9)=Z'3160'  !GAIN CARD SLOT 2, CHANNEL 13, GAIN 1
      OUTC(10)=Z'2170' !GAIN CARD SLOT 2, CHANNEL 14, GAIN 1
      OUTC(11)=Z'3170' !GAIN CARD SLOT 2, CHANNEL 15, GAIN 1
      SIZEC=142
      CALL ADIUSR(IOCHC,IOCBC,SIZEC,BUSY)   !INITIALIZE GROUP 4
      WRITE (5,5994)
 5991 FORMAT (' FAST INIT')
 5992 FORMAT (' GRPA INIT')
 5993 FORMAT (' GRPB INIT')
 5994 FORMAT (' GRPC INIT')
      INITDONE=1
  10  CONTINUE
      GOTO (15,20,25,30)IG
  15  CALL ADIUSR(IOCHF,IOCBF,SIZEF,BUSY)
      ISTAT=IOCBF(40)
      GOTO 50
  20  CALL ADIUSR(IOCHA,IOCBA,SIZEA,BUSY)
      ISTAT=IOCBA(60)
      GOTO 50
  25  CALL ADIUSR(IOCHB,IOCBB,SIZEB,BUSY)
      ISTAT=IOCBB(80)
      GOTO 50
  30  CALL ADIUSR(IOCHC,IOCBC,SIZEC,BUSY)
      ISTAT=IOCBC(88)
  50   CALL MTDAY(ITIME)          !GET TIME FOR EACH GROUP
       IA(II,JJ)=ITIME(1)
       IA(II+1,JJ)=ITIME(2)
       IA(II+2,JJ)=ITIME(3)
       IA(II+3,JJ)=ITIME(4)
       IBYTE=3
      DO 90 I=1,IPTS*2
```

```
      IA(II+I+IBYTE,JJ)=IDATA(I)
   90 CONTINUE
      GOTO 999
  800 WRITE(5,5000)ISTAT
 5000 FORMAT(' ADI STATUS ERROR: ',Z8)
  999 II=II+4+IPTS*2          !NUMBER OF BYTES OF INPUT
      RETURN
      END
      SUBROUTINE GETCDS(IA,II,JJ)
C     ********** SUBROUTINE TO RETRIEVE CDS DATA
      CHARACTER*8 CONTNAME
      INTEGER CONTUNII
      INTEGER*1 IA(220,40),ITIME(4)
      INTEGER FCB(16),IOCL(130),SENSBUFF(4)
      INTEGER*1 CMDBUF(8,8) !INSTRUCTION + LABEL TABLE FOR CDS
      INTEGER*1 INPBUF(64),OUBUF(64)
      INTEGER URTC,UTDA,RDDA,ESRQ,DSRQ,CHAIN,SII
      CHARACTER*1 NOTALK,NOLISN,ITALK,ILISN,UTALK,ULISN
      CHARACTER*4 TLKBUF,LSNBUF
      EXTERNAL IEEEXCPM,IEEELOCF,IEEESFMT,IEEECST
      DATA URTC/Z'11000000'/,WTDA/Z'01000000'/
      DATA RDDA/Z'02000000'/,ESRQ/Z'63000000'/
      DATA DSRQ/Z'73000000'/,CHAIN/Z'40000000'/
      DATA SII/Z'20000000'/
      DATA NOTALK/' '/,NOLISN/'?'/,ITALK/'@'/,ILISN/' '/
      DATA UTALK/'C'/,ULISN/'#'/
      DATA CMDBUF/Z'40',Z'30',Z'36',Z'41',Z'43',Z'53',Z'08',0,
     .           Z'40',Z'30',Z'36',Z'41',Z'43',Z'53',Z'09',0,
     .           Z'40',Z'30',Z'34',Z'41',Z'43',Z'53',Z'23',0,
     .           Z'40',Z'30',Z'36',Z'41',Z'43',Z'53',Z'0A',0,
     .           Z'40',Z'30',Z'36',Z'41',Z'43',Z'53',Z'0B',0,
     .           Z'40',Z'30',Z'36',Z'41',Z'43',Z'53',Z'0C',0,
     .           Z'40',Z'30',Z'36',Z'41',Z'43',Z'53',Z'0D',0,
     .           Z'40',Z'30',Z'36',Z'41',Z'43',Z'53',Z'0E',0/
C     ****** NEXT STATEMENTS WILL INITIALIZE THE CDS
C     ****** THEY ARE PERFORMED ONLY ON FIRST TIME INPU
      IPTS=8
      IF(INITDONE.EQ.1)GOTO 20
      INITDONE=1
      FCB=0
      FCB(3)=5              !TIMEOUT VALUE OF 5 SECONDS
      FCB(10)=0
      CONTUNIT='IEE'
      CONTNAME='U07E80  '
C CONNECT CONTROLLER TO THIS TASK
        CALL IEEECSI(CONTUNIT,CONTNAME,FCB,-1,ISTAT)
      IF(ISTAT.NE.0)GOTO 900
      FCB(3)=Z'80000005'          !NO-WAIT BIT SET
      FCB(9)=IEEELOCF(IOCL(1))
      FCB(7)=IEEELOCF(*915)       !WAIT I/O ERROR RETURN
      FCB(14)=IEEELOCF(*10)  !NO WAIT RETURN ADDRESS
```

```
      FCB(15)=IEEELOCF(*15)  !NO WAIT ERROR RETURN ADDRESS
      TLKBUF=NOTALK//NOLISN//ULISN//ITALK
      LSNBUF=NOTALK//NOLISN//ILISN//UTALK
      IOCL(1)=WRTC+IEEELOCF(TLKBUF)
      IOCL(2)=CHAIN+2
      IBYTE=0
      DO 9 M=1,8
      MM=2+8*(M-1)
      IOCL(MM+1)=WRTC+IEEELOCF(TLKBUF)+2
      IOCL(MM+2)=CHAIN+2          !SEND ULISN,ITALK
      IOCL(MM+3)=WTDA+IEEELOCF(CMDBUF(1,M))
      IOCL(MM+4)=CHAIN+7          !SEND COMMAND + LABEL
      IOCL(MM+5)=WRTC+IEEELOCF(LSNBUF)+2
      IOCL(MM+6)=CHAIN+2          !SEND ILISN,UTALK
      IOCL(MM+7)=RDDA+IEEELOCF(INPBUF)+IBYTE
      IOCL(MM+8)=CHAIN+5          !RECEIVE 5 BYTES
      IBYTE=IBYTE+3               !ONLY 3 BYTES OF DATA + CR,LF
9     CONTINUE
      IOCL(64)=5                    !NO CHAIN ON LAST IOCL
      CALL IEEEXCPM(FCB,ISTAT)
      GOTO 999                    !DON'T WAIT FOR I/O -- EXIT SUB
C PROGRAM WILL TRAP HERE WHEN I/O IS DONE ......................
10    CALL M:TDAY(OUBUF)          !GET TIME OF THIS DATA SET
      DO 12 I=1,IPTS              !TRANSFER DATA TO OUT ARRAY
      J=4*(I-1)*4
      K=(I-1)*3
      OUBUF(J+1)=0
      OUBUF(J+2)=INPBUF(K+1)
      OUBUF(J+3)=INPBUF(K+2)
      OUBUF(J+4)=INPBUF(K+3)
12    CONTINUE
      CALL IEEEXCPM(FCB,ISTAT)    !SET UP ANOTHER TRANSFER
      CALL X:XNWIO                !EXIT THIS NO-WAIT ROUTINE
C TRAPS HERE ON ERROR ...  NEED TO RE-INITIALIZE EVERYTHING
15    IF(IJFLG.EQ.0)  WRITE(5,5998)  !PRINTS THIS MESSAGE ONLY ONCE
      IJFLG=1
      FCB(3)=5          !CLEAR NO-WAIT, SET TIMEOUT TO 5 SECONDS
      FCB(1)=0
      FCB(10)=0
      CALL IEEEDST(CONTUNIT,FCB,ISTAT)      !DISCONNECT FROM TASK
      IF(ISTAT.NE.0)GOTO 905
5998  FORMAT(' CDS ERROR ON NO-WAIT RETURN')
25    DO 11 I=1,IPTS*3
      INPBUF(I)=0           !CLEAR THE INPUT BUFFER
11    CONTINUE
      INITDONE=0          !NEXT TIME THRU NEED TO RE-INITIALIZE
      CALL X:XNWIO        !EXIT THIS NO-WAIT ERROR ROUTINE
C AFTER INITIALIZATION, A CALL TO CDS WILL COME TO STATEMENT 30.....
C THIS CODE MERELY TRANSFERS THE MOST RECENT VALUES OF OUBUF TO IA.....
30    IBYTE=0
      DO 45 I=1,IPTS+1
```

```
          IA(II+IBYTE  ,JJ)=OUBUF(IBYTE+1)
          IA(II+IBYTE+1,JJ)=OUBUF(IBYTE+2)
          IA(II+IBYTE+2,JJ)=OUBUF(IBYTE+3)
          IA(II+IBYTE+3,JJ)=OUBUF(IBYTE+4)
          IBYTE=IBYTE+4
   45     CONTINUE
   50     CONTINUE
          GOTO 999
C     **********    ERROR RETURNS     *******
  900     WRITE(5,5000)ISTAT
 5000     FORMAT(' IEEECST ERROR: ',I8)
          GOTO 999
  905     WRITE(5,5005)ISTAT
 5005     FORMAT(' IEEEDST ERROR: ',I8)
          GOTO 25
  915     WRITE(5,5015)FCB(4)
 5015     FORMAT(' I/O ERROR, FCB(4)=',Z8)
          GOTO 999
  920     WRITE(5,5020)ISTAT
 5020     FORMAT(' IAEEXCPM ERROR, STATUS=',Z8)
  700     CONTINUE
  999     II=II+4+IPTS*4
          RETURN
          END
$ASSIGN LIB TO IEEELIB
$ASSIGN DIR TO IEEEDIR
$ASSIGN SLO TO SLODMP
$EXECUTE CATALOG
ASSIGN 5 TO ACDERR
ASSIGN 6 TO DEV=MS1000,BLOC=N
ENVI TEST
CATALOG ACQUISIT 5
$EOJ
```

4. EBLE PROGRAMS

The EBLE programs consist of three programs which run simultaneously. The source listings reside in the three files EBLE1, EBLE2 and EBLE3 which when compiled and catalogued are called STEP1, STEP2 and STEP3. The operator executes the file STEP1 (or the batch file GTE which will be discussed in chapter 5). STEP1 automatically activates STEP2 and STEP2 activates STEP3. Flow charts for the three programs are shown in figures 5 and 6.

The data sampled during EBLE consisted of four groups of data -- one from the CDS and three from the RTP's. Sampling of these data sets was done at the rate of 1 Hz. The format and description of the complete data set is shown in Table 2.

The first RTP data set consists of flow angles input from the synchro-to-digital converters on RTP subsystem 0. The second set consists of position data from the aircraft navigation system input through the synchro-to-digital converters in RTP subsystem 1. The third set of RTP data are readings of five pressure transducer rakes, each rake having 11 ports. These are input through two analog input/output cards in RTP subsystem 0. A certain amount of time is required between the output and input to and from the pressure transducers. Therefore, the program logic is set up to output to the transducers, then retrieve one CDS data label, then input from the transducers. This is repeated 11 times (for each of the ports). Time is recorded at the beginning of the entire data set and at the end of the data set.

Figure 7 shows the RTP and CDS systems set-up during the EBLE experiment. Program listings of the files EBLE1, EBLE2 and EBLE3 follow.

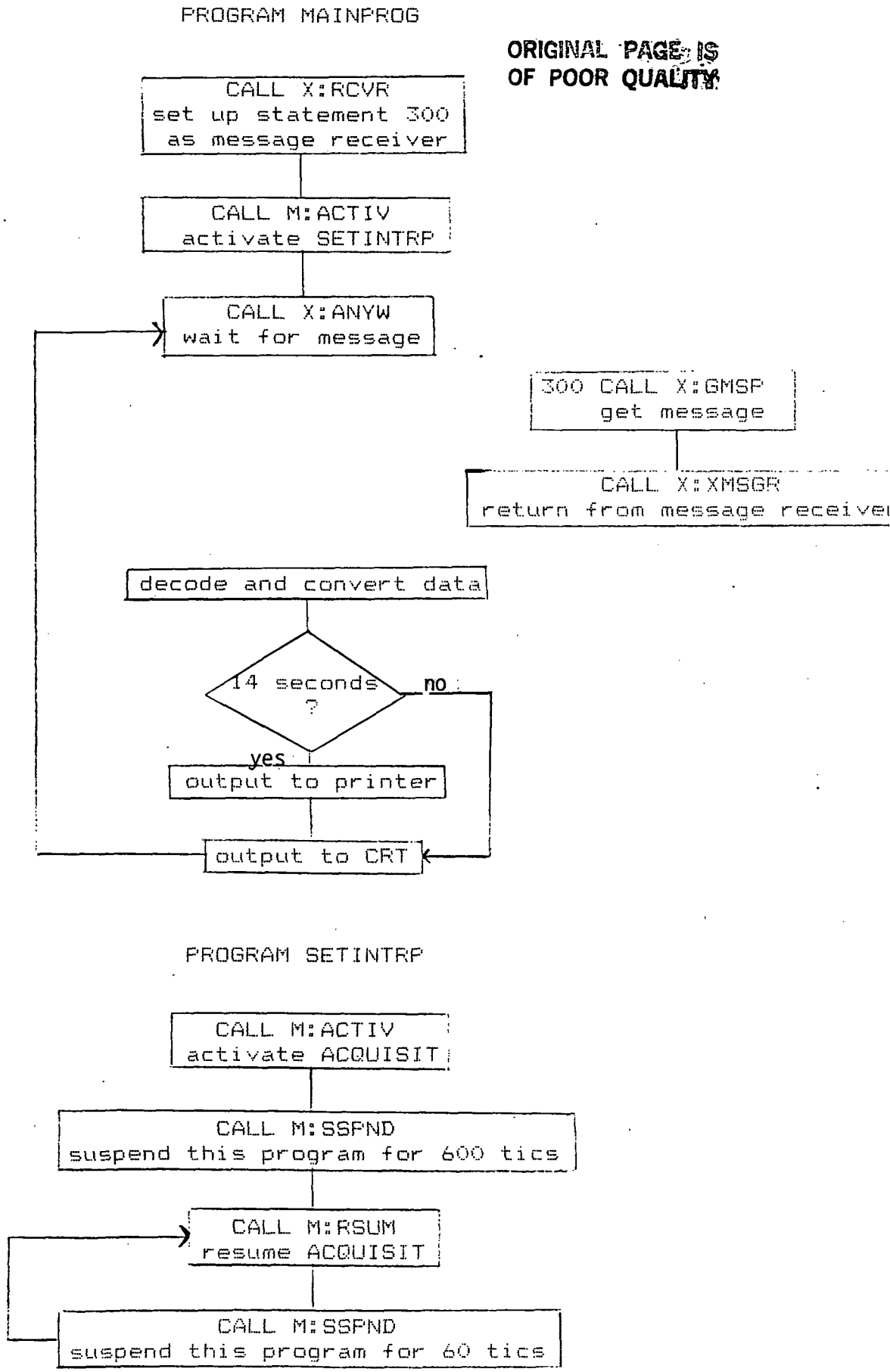


Figure 5. Flowcharts of EBLE versions of MAINPROG and SETINTRP.

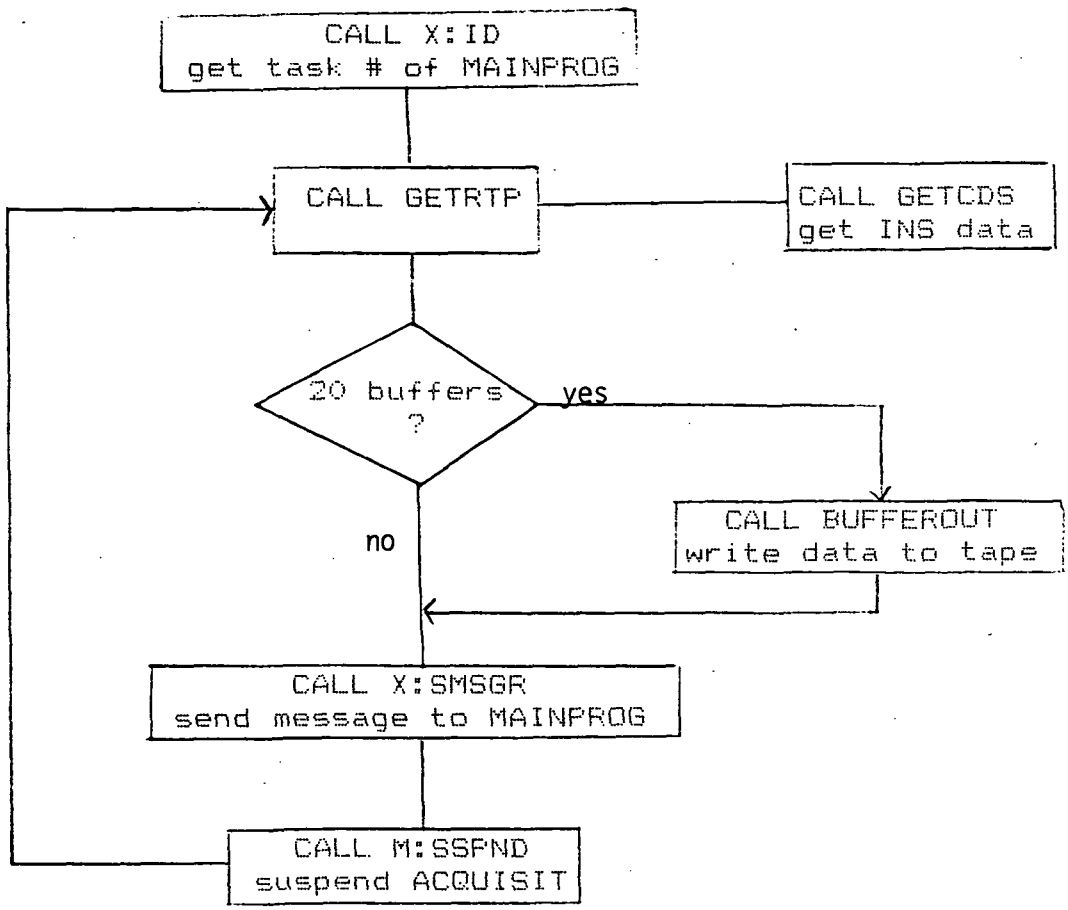


Figure 6. Flowchart of EBLE version of ACQUISIT.

Table 2. EBLE Data Format

Location in Array	# Bytes	Description	Source
1	4	time	Gould clock
5	2	flow angle 1	Synchro #1 (0)
7	2	flow angle 2	Synchro #2 (0)
9	2	flow angle 3	Synchro #3 (0)
11	2	flow angle 4	Synchro #4 (0)
13	2	flow angle 5	Synchro #5 (0)
15	2	pitch	Synchro #1 (1)
17	2	roll	Synchro #2 (1)
21	2	heading	Synchro #3 (1)
21	2	baro-alt (fine)	Synchro #4 (1)
23	2	baro-alt (coarse)	Synchro #5 (1)
25	2	rake 1 - port 1	
27	2	rake 2 - port 1	
29	2	rake 3 - port 1	
31	2	rake 4 - port 1	
33	2	rake 5 - port 1	
35	2	rake 1 - port 2	
etc.			
125	2	rake 1 - port 11	
etc.			
133	2	rake 5 - port 11	
			Label (hex)
135	4	track error	5
139	4	drift	6
143	4	latitude	8
147	4	longitude	9
151	4	ground speed	A
155	4	track angle	B
159	4	true heading	C
163	4	wind speed	D
167	4	wind dir	E
171	4	N/S velocity	B6
175	4	E/W velocity	B7
179	4	time	Gould clock

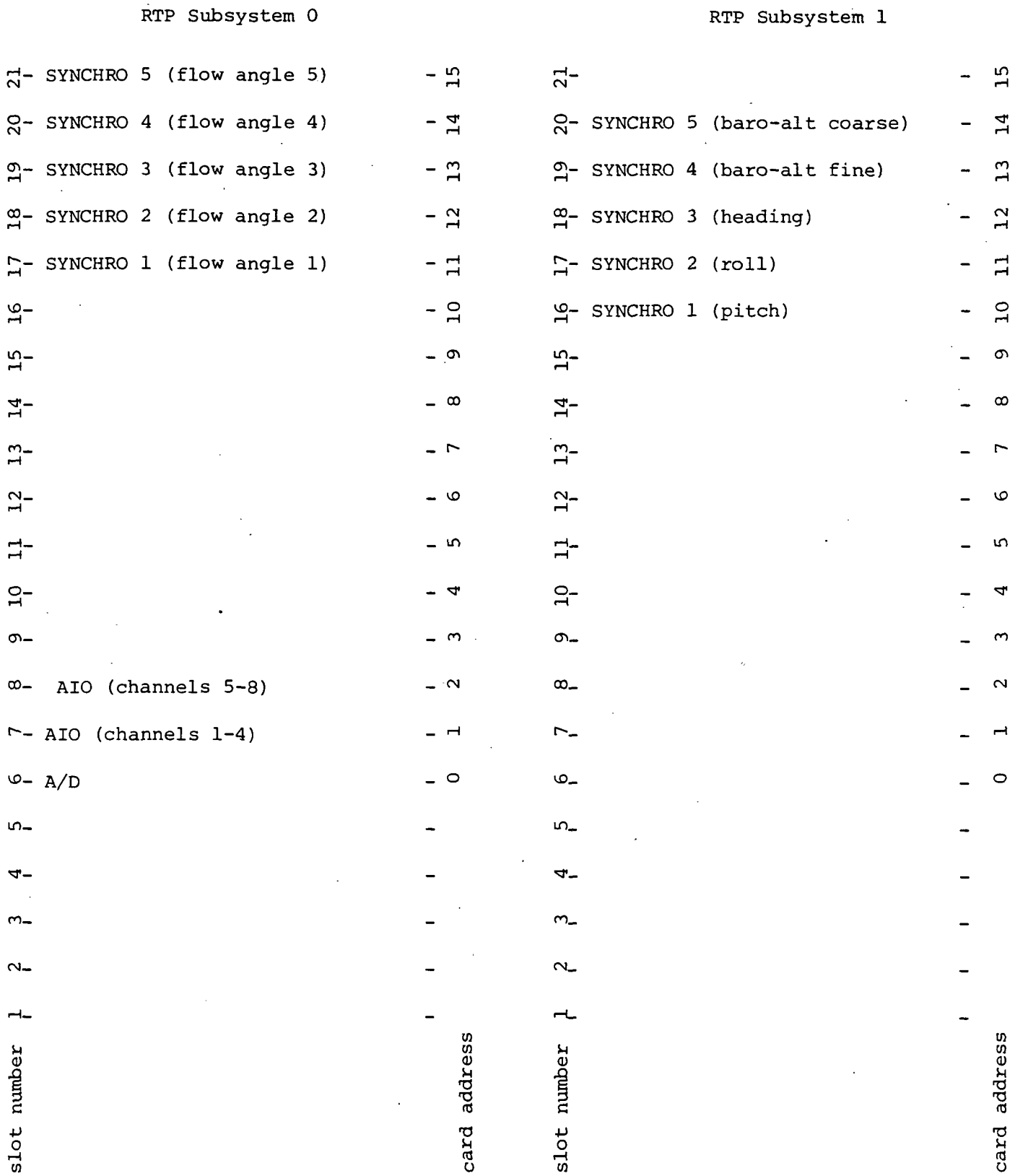


Figure 7. RTP card cage set-up for EBLE.

```
$!
$JOB MAINPROG
$ASSIGN LO TO SLOFOR
$OPTION 5
$EXECUTE FORTRAN
      PROGRAM MAINPROG
      INTEGER*8 ITASK
      INTEGER*1 ITIME1(4),ITIME2(4)
      INTEGER*2 FLOW(5),NAV(5),PT(5,11)
      INTEGER*4 LAT,LON
      INTEGER   WS,WD,TRKANG,TRUHD,GRDSPD,TRKERR,DRIFT
      INTEGER NSVEL,EWVEL
      INTEGER PRB(16),PRB(5),RXB(2)              !PARAMETER BLOCKS
      INTEGER*1 IA(200)
      DIMENSION SYNC1(5),SYNC2(5),IX(11),XPT(5,11)
      EQUIVALENCE (IA(135),TRKERR)
      EQUIVALENCE (IA(139),DRIFT)
      EQUIVALENCE (IA(1),ITIME1(1))
      EQUIVALENCE (IA(179),ITIME2(1))
      EQUIVALENCE (IA(143),LAT)
      EQUIVALENCE (IA(147),LON)
      EQUIVALENCE (IA(151),GRDSPD)
      EQUIVALENCE (IA(155),TRKANG)
      EQUIVALENCE (IA(159),TRUHD)
      EQUIVALENCE (IA(163),WS)
      EQUIVALENCE (IA(167),WD)
      EQUIVALENCE (IA(171),NSVEL)
      EQUIVALENCE (IA(175),EWVEL)
      EQUIVALENCE (IA(5),FLOW(1))
      EQUIVALENCE (IA(15),NAV(1))
      EQUIVALENCE (IA(25),PT(1,1))
      INTEGER*1 DONEFLAG(4)     !FLAG FOR MORE DATA TO PRINT
      DATA IX/1,2,3,4,5,6,7,8,9,10,11/
C **** SET UP BUFFER RECEIVER PARAMETERS
      CALL X:RCVR(*300,ISTAT,*995)  !WHEN DATA IS RECEIVED GOTO 300
      PRB(1)=ADDR(IA)
      PRB(2)=Z'00C80000'          !# BYTES THAT ARE RECEIVED FROM ACQUISIT
C **** ACTIVATE THE INTERRUPT GENERATING ROUTINE
      ITASK='SETINTRP'
      CALL M:ACTIV(ITASK,ISTAT,*997)
C **** WAIT FOR BUFFER FROM PROGRAM ACQUISIT.....
90    CALL X:ANYW(0,ISTAT,*996)
C **** DO THE CONVERSIONS FROM BINARY TO REAL UNITS
       DO 555 I=1,5
       FLOW(I)=IBITS(FLOW(I),2,14)
       NAV(I)=IBITS(NAV(I),2,14)
       SYNC1(I)=FLOW(I)*.0219727
       SYNC2(I)=NAV(I)*.0219727
555    CONTINUE
      SYNC2(1)=360.-SYNC2(1)     !SOME SYNCHROS ARE OFF 360.
```

```
      SYNC2(5)=(360.0-SYNC2(5))*375.  !COARSE ALTIMETER
      DO 666 I=1,5
      DO 666 J=1,11
      XPT(I,J)=(PT(I,J)*3.125E-4)          !PRESSURE TRANSDUCERS
666   CONTINUE
C **** LAT AND LON ARE IN HEX
      LAT=IBITS(LAT,0,21)
      LON=IBITS(LON,0,21)
C **** PRINT TO CRT MONITOR ..................................
      WRITE(3,3000)ITIME1,ITIME2
3000  FORMAT(1H0,3(I2,':'),I2,10X,3(I2,':'),I2)
      WRITE(3,3005)SYNC2(5),LAT,LON,WS,WD,SYNC2(1),
     1  SYNC2(2),SYNC2(3)
3005  FORMAT(5X,'ALT',5X,'LAT',5X,'LON',6X,'WS',6X,'WD',3X,
     1  'PITCH',4X,'ROLL',4X,'HEAD'/1X,F7.0,4(2X,Z6),3F8.0)
      WRITE(3,3010)IX
3010  FORMAT(1X,'SENSOR',3X,11I5,3X,'FLOW')
      DO 100 II=1,5
      WRITE(3,3015)II, (XPT(II,I),I=1,11),SYNC1(II)
100   CONTINUE
3015  FORMAT(1X,I3,6X,11F5.2,F7.0)
      IF (ITEN.NE.2)GOTO 444
      ITEN=0
C **** PRINT TO PRINTER.....................................
      WRITE(2,2000)(ITIME1(I),I=1,3),SYNC2(5),LAT,LON,
     1  WS,WD,SYNC2(1),SYNC2(2),SYNC2(3)
2000  FORMAT(1H0,6X,'TIME',7X,'ALT',7X,'LAT',7X,'LON',3X,
     1  'WIND SP',2X,'WIND DIR',5X,'PITCH',6X,'ROLL',3X,
     2  'HEADING'/3X,2(I2,':'),I2,F10.0,4(4X,Z6),3F10.0)
      WRITE(2,2005)IX
2005  FORMAT(1H0,'STATION / SENSOR',11I7,' / FLOW ANGLE')
      DO 200 II=1,5
      WRITE(2,2010)II,(XPT(II,I),I=1,11),SYNC1(II)
200   CONTINUE
2010  FORMAT(1X,I4,12X,11F7.2,F10.0)
444   ITEN=ITEN+1
      GOTO 90
C **** NEW BUFFER IS AVAILABLE ...........................
300   CALL X:GMSGP(PRB,ISTAT,*994)
      LSTAT=IBITS(PRB(1),0,8)
      RXB(1)=LSTAT+ADDR(DONEFLAG)
      RXB(2)=2
C **** EXIT MESSAGE RECEIVER -- RETURN TO STATEMENT FOLLOWING
C     THE CALL TO "X:ANYW".....................................
      CALL X:XMSGR(RXB,ISTAT,*993)
C ****   ERROR RETURNS AND MESSAGES    ****
993   WRITE (3,2045)ISTAT
2045  FORMAT(' X:XMSGR ERROR:  ',I5)
      GOTO 999
994   WRITE(3,2040)ISTAT
2040  FORMAT(' X:GMSGP ERROR:  ',I5)
```

```
      GOTO 999
995   WRITE(3,2035)ISTAT
2035  FORMAT(' X:RCVR ERROR:  ',I5)
      GOTO 999
996   WRITE(3,2030)ISTAT
2030  FORMAT(' X:ANYW ERROR:  ',I5)
      GOTO 999
997   WRITE(3,2020)ISTAT
2020  FORMAT(' M:ACTIV ERROR:  ',I5)
      GOTO 999
998   WRITE(3,2015)FCB(4)
2015  FORMAT(' I/O ERROR, FCB(4)=',Z8)
999   STOP
      END
$ASSIGN SLO TO SLODMP
$ASSIGN DIR TO IEEEDIR
$ASSIGN LIB TO IEEELIB
$EXECUTE CATALOG
ASSIGN SI TO SYC
ASSIGN 2 TO DEV=TY7EC6
ASSIGN 3 TO LFC=UT
CATALOG MAINPROG
$EOJ
```

```
$$
$JOB SETINTRP
$ASSIGN LO TO SLOFOR
$OPTION 5
$EXECUTE FORTRAN
      PROGRAM SETINTRP
      INTEGER*8 ITASK2
      ITASK2='ACQUISIT'
      CALL M:ACTIV(ITASK2,*999)          !ACTIVATE PROGRAM ACQUISIT
      CALL M:SSPND(600,*999)          !WAIT WHOLE MINUTE FIRST TIME
10    CALL M:RSUM(ITASK2,*999)     !RESUME DATA ACQUISITION
15    CALL M:SSPND(60,*999)        !SUSPEND THIS JOB FOR 60 TICS
      GOTO 10
999   CONTINUE
      WRITE(6,6000)
6000  FORMAT(' SETINTRP ERROR')    !GETS HERE ONLY IF ERROR
      STOP
      END
$ASSIGN SLO TO SLODMP
$EXECUTE CATALOG
ASSIGN 6 TO INTERR
CATALOG SETINTRP
$EOJ
```

```
$$
$JOB ACQUISIT
$NOTE DATA ACQUISITION PROGRAM
$ASSIGN LO TO SLOFOR
$OPTION 5
$EXECUTE FORTRAN
      PROGRAM ACQUISIT
      INTEGER*8 TASKNAM,OWNRNAM,PSEUDO
      INTEGER*1 IA(200,40)
      INTEGER       PSB(8),RBA(2)
C **** DETERMINE TASK # OF CALLING PROGRAM "MAINPROG"
      TASKNAM='MAINPROG'
      INDEX=0
      ITASK=0
      OWNRNAM=0
      PSEUDO=0
      CALL X:ID(INDEX,ITASK,TASKNAM,OWNRNAM,PSEUDO,ISTAT,*900)
C **** SET UP PSB FOR LATER TRANSMISSION OF BUFFER.....
       PSB(1)=ITASK     !TASK # OF "MAINPROG" FROM X:ID
       PSB(2)=0         !MUST BE 0
       PSB(3)=200   !# BYTES TO TRANSMIT
       PSB(5)=Z'100000'  !# BYTES TO RECEIVE BACK (2)...
       PSB(6)=ADDR(RBA)  !PURE ADDRESS OF RECEIVER BUFFER
       PSB(7)=ADDR($800) !NO WAIT ERROR RETURN
      IPRINT=1          !PRINT CONTROL
      MTERRS=0          !# MAG TAPE ERRORS
      JJ=1              !INDEX FOR BUFFERS-- 20 WILL BE STACKED
                        !PRIOR TO WRITING TO TAPE
      IFLAG=0          !TAPE ERROR FLAG
100   CALL GETRIP(IA,JJ)          !GET RIP AND CDS DATA
        IF (JJ.EQ.20)GOTO 420     !2 STACKS OF 20 AVAILABLE -- 1 FOR
        IF (JJ.EQ.40)GOTO 420     !FILLING AND ONE FOR WRITING TO TAPE
        GOTO 430
420     IF (IFLAG.EQ.1)GOTO 905   !IS MAG TAPE STILL BUSY?
        IFLAG=1
C **** WRITE BUFFER TO TAPE.....
        CALL BUFFEROUT(6,1,IA(1,JJ-19),1000,$805,$910)
C **** SEND BUFFER BACK TO CALLING PROGRAM.....
430     PSB(4)=ADDR(IA(1,JJ))
        CALL X:SMSGR(PSB,ISTAT,*915)
 450    JJ=JJ+1
        IF(JJ.EQ.41)JJ=1
C **** SUSPEND THIS ROUTINE UNTIL NEXT INTERRUPT.....
 500    CALL M:SSPND(0,*920)
        GOTO 100
C    *********   END ACTION ROUTINES   ****
 800    CALL X:XMEA      !EXIT SEND MESSAGE
 805    IFLAG=0          !MAG TAPE DONE
        CALL STATUS(6,ISTAT)  !ANY ERRORS?
        IF(ISTAT.NE.2) WRITE(5,8005)ISTAT
 8005   FORMAT(' MAGTAPE STATUS ERROR: ',I5)
```

```
      CALL X:XNWIO
C     *********   ERROR MESSAGES OUTPUT TO FILE "ACOERR"
  900 WRITE(5,9000)ISTAT
 9000 FORMAT(' X:ID ERROR: ',I5)
      GOTO 999
  905 MTERRS=MTERRS+1
      WRITE(5,9005)MTERRS
 9005 FORMAT(' MAGTAPE BUSYF FLAG SET, ERRORS=',I5)
      GOTO 500
  910 WRITE(5,9010)
 9010 FORMAT(' MAGTAPE HARDWARE ERROR')
      GOTO 999
  915 WRITE(5,9015)ISTAT
 9015 FORMAT(' X:SMSGR ERROR: ',I5)
      GOTO 999
  920 WRITE(5,9020)
 9020 FORMAT(' M:SSPND ERROR')
      GOTO 999
  999 STOP
      END
      SUBROUTINE GETRTP(IA,JJ)
      INTEGER*1 IA(200,40),ITIME(4),IDATA(40)
      INTEGER IOCHFA(3),IOCHNV(3),IOCHO(3,11),IOCHI(3)
      INTEGER IOCBFA(16),IOCBNV(16),IOCBO(16,11),IOCBI(70)
      INTEGER*2 ICHNFA,ICHNNV,ICHNO(8),ICHNI,ICHN11
      INTEGER*2 OUTA(8,11),OUTB(5),INP(20)
C
      INTEGER BUSY,SIZE,SIZEN,MSK(4)
      INTEGER ADD1,ADD2,ADD3,ADD7,ADD8,ADD9
      EQUIVALENCE (IDATA(1),INP(1))
C
      IF(INITDONE.GE.1) GOTO 100
C ************   INITIALIZE SECTION   **********************
C ******   SET UP IOCH/IOCB FOR EACH GROUP   ***************
C THE FIRST SET OF BUFFERS ARE TO READ 5 FLOW ANGLES FROM THE SYNCHROS
      ADD1=LOCF(ICHNFA)      !CHANNEL HALFWORD TABLE ADDRESS
      ADD2=LOCF(INP)         !INPUT BUFFER ADDRESS
      IOCHFA(1)=Z'00300000' !IOM ADDRESS OF 30
      IOCHFA(2)=ADDR(IOCBFA)  !IOCB ADDRESS
      IOCHFA(3)=Z'20002'      !NOP
      IOCBFA(1)=Z'E0000005'   !DIGITAL INPUT 5 HALFWORDS
      IOCBFA(2)=LOCF(ICHNFA)  !CHANNEL HALFWORD TABLE
      IOCBFA(3)=LOCF(INP)     !INPUT BUFFER
      IOCBFA(4)=0             !ALWAYS ZERO (CONTAINS STATUS ON RETURN
      ICHNFA=Z'10B'           !FIRST SYNCHRO TO READ IS IN SLOT 11
      SIZE=16
      BUSY=LOCF($800)
      CALL ADIUSR (IOCHFA,IOCBFA,SIZE,BUSY)  !INITIALIZE EA BUFFERS
      WRITE  (5,5001)
 5001 FORMAT(' THIS SECOND SET OF BUFFERS ARE TO READ NAV DATA FROM 5 SYNCHROS
```

```
      IOCHNV(2)=ADDR(IOCBNV)
      IOCHNV(3)=Z'20002'              !NOP
      IOCBNV(1)=Z'E0000005'           !IN,SEQ,DIG; 5 HALFWORDS
      IOCBNV(2)=LOCF(ICHNNV)
      IOCBNV(3)=ADD2
      IOCBNV(4)=0
      ICHNNV=Z'210A'          !SUBSYSTEM=1, SLOT=10
       CALL ADIUSR (IOCHNV,IOCBNV,SIZE,BUSY)  !INITIAL NV BUFFERS
      WRITE (5,5002)
C THE FOLLOWING SET OF 11 BUFFERS WILL OUTPUT VOLTAGES TO THE
C PRESSURE TRANSDUCERS THRU THE AIO CARDS
      DO 10 I=1,4
      ICHNO(I)=Z'C02'  !8-CHANNEL I/O SLOT 2; 10V DAC OUTPUT FS
      ICHNO(I+4)=Z'C01'!8-CHANNEL I/O SLOT 1; 10V DAC OUTPUT FS
10    CONTINUE
      ADD1=LOCF(ICHNO)
      DO 40 I=1,11
      ADD3=LOCF(OUTA(1,I))
      IOCHO(1,I)=Z'00300000'          !IOM ADDRESS OF 30
      IOCHO(2,I)=ADDR(IOCBO(1,I))     !ADDRESS OF ONE OF 11 IOCB'S
      IOCHO(3,I)=Z'20002'             !NOP
      IOCBO(1,I)=Z'20000008'          !OUT,RAN,DIG; 8 HALFWORDS
      IOCBO(2,I)=ADD1
      IOCBO(3,I)=ADD3
      IOCBO(4,I)=0
C IN ANY PASS THRU THIS "DO 40" LOOP THE MSK ARRAY DEFINES WHICH
C LINES TO PASS 10V TO... THEREBY SIGNALING THE RTP WHICH SIGNALS
C TO PUT ON THE LINES.
      MSK(1)=0
      MSK(2)=0
      MSK(3)=0
      MSK(4)=0
      II=I-1
      IF(II.EQ.0)GOTO 25
      I1=II/2
      IF(I1*2.NE.II) MSK(1)=Z'FFF'
      I2=I1/2
      IF(I2*2.NE.I1) MSK(2)=Z'FFF'
      I3=I2/2
      IF(I3*2.NE.I2) MSK(3)=Z'FFF'
      I4=I3/2
      IF(I4*2.NE.I3) MSK(4)=Z'FFF'
25    DO 30 J=1,4
      JJ=J-1
      OUTA(J,I)=JJ*Z'2000'+MSK(J)     !CHANNEL # + VOLTS (0V OR 10V)
      OUTA(J+4,I)=OUTA(J,I)           !SAME AS ABOVE BUT FOR OTHER AIO
30    CONTINUE
      CALL ADIUSR (IOCHO(1,I),IOCBO(1,I),SIZE,BUSY) !INIT OUT BUFFERS
      WRITE (5,5003) I
40    CONTINUE
C THE FOLLOWING BUFFERS WILL READ PRESSURE TRANSDUCERS THRU THE
```

```
C AIO CARDS AS DEFINED BY THE PREVIOUS OUTPUT BUFFERS.
      ADD8 =LOCF(INP)
      ADD9 =LOCF(OUTB)
      IOCHI(1)=Z'00300000'          !IOM ADDRESS OF 30
      IOCHI(2)=ADDR(IOCBI)
      IOCHI(3)=Z'20002'             !NOP
      L2=0
      LL=0
      DO 50 I=1,5
      IOCBI(1+LL)=Z'22000001' !OUT,RAN,DIG; 1 HALFWORD
      IOCBI(2+LL)=LOCF(ICHNI)
      IOCBI(3+LL)=ADD9 +L2
      IOCBI(4+LL)=0
      IOCBI(5+LL)=Z'A2000001' !IN, RAN,DIG; 1 HALFWORD
      IOCBI(6+LL)=LOCF(ICHNII)
      IOCBI(7+LL)=ADD8 +L2
      IOCBI(8+LL)=0
      LL=LL+8
      L2=L2+2
50     CONTINUE
      IOCBI(37)=Z'A0000001'         !NO CHAIN ON LAST IOCB
      ICHNI=Z'1100'          !BYPASS READY TEST,REPEAT ACCESS,SLOT 0
      ICHNII=Z'100'         !REPEAT ACCESS, SLOT 0
      OUTB(1)=Z'3081'    !MODE=3;CARD SLOT=1;CHANNEL=0;GAIN=2
      OUTB(2)=Z'3101'    !MODE=3;CARD SLOT=2;CHANNEL=0;GAIN=2
      OUTB(3)=Z'3111'    !MODE=3;CARD SLOT=2;CHANNEL=1;GAIN=2
      OUTB(4)=Z'3121'    !MODE=3;CARD SLOT=2;CHANNEL=2;GAIN=2
      OUTB(5)=Z'3131'    !MODE=3;CARD SLOT=2;CHANNEL=3;GAIN=2
      SIZEN=70
      CALL ADIUSR(IOCHI,IOCBI,SIZEN,BUSY)  !INIT INPUT BUFFERS
      WRITE (5,5004)
 5001 FORMAT (' SYN RTP #2 INIT DONE')
 5002 FORMAT (' SYN RTP #1 INIT DONE')
 5003 FORMAT (' ANALOG CHANNEL # ',I3,' INIT DONE')
 5004 FORMAT (' ANALOG INPUT INIT DONE')
      INITDONE=1
C AFTER INITIALIZATION, A CALL TO RTP ALWAYS COMES HERE .....
100    CALL M:TDAY(ITIME)           !GET TIME OF DATA SET
       IA(1,JJ)=ITIME(1)
       IA(2,JJ)=ITIME(2)
       IA(3,JJ)=ITIME(3)
       IA(4,JJ)=ITIME(4)
      CALL ADIUSR(IOCHFA,IOCBFA,SIZE,BUSY)    !GET FLOW ANGLES
      ISTAT=IOCBFA(4)
      II=5
      DO 150 I=1,10
      IA(II+I-1,JJ)=IDATA(I)
150   CONTINUE
       CALL ADIUSR(IOCHNV,IOCBNV,SIZE,BUSY)  !GET NAV DATA
      ISTAT=IOCBNV(4)
      II=15
```

```
      DO 160 I=1,10
      IA(II+I-1,JJ)=IDATA(I)
160    CONTINUE
C READING THE PRESSURE TRANSDUCERS IS A TWO STEP PROCESS.  FIRST,
C YOU MUST OUTPUT THE CHANNEL NUMBER AND ALLOW A CERTAIN AMOUNT OF
C TIME FOR THE SIGNALS TO SETTLE ON THE LINES.  THEN YOU CAN READ
C THE CHANNELS.  IN ORDER TO ASSURE ENOUGH TIME FOR SIGNALS TO SETTLE
C THIS PROGRAM SENDS CHANNELS, THEN READS IN A PIECE OF CDS DATA,
C THEN COMES BACK AND READS THE CHANNELS.
      DO 200 I=1,11
      CALL ADIUSR(IOCHO(1,I),IOCBO(1,I),SIZE,BUSY) !SET SIGNAL LINES
      ISTAT=IOCBO(4,I)
      CALL GETCDS(IA,JJ,I)                         !GET NEXT CDS DATA
      CALL ADIUSR(IOCHI,IOCBI,SIZEN,BUSY)          !READ SIGNAL LINES
      ISTAT=IOCBI(40)
      II=25+(I-1)*10
      DO 170 J=1,10
      IA(II+J-1,JJ)=IDATA(J)
170    CONTINUE
200   CONTINUE
      CALL M:TDAY(ITIME)          !GET TIME AT END OF DATA SET
      IA(179,JJ)=ITIME(1)
      IA(180 ,JJ)=ITIME(2)
      IA(181 ,JJ)=ITIME(3)
      IA(182 ,JJ)=ITIME(4)
      GOTO 999
300   WRITE(5,5000)ISTAT
 5000 FORMAT(' ADI STATUS ERROR: ',Z8)
999   RETURN
      END
      SUBROUTINE GETCDS(IA,JJ,MLAB)
C     ************   SUBROUTINE TO RETRIEVE
      CHARACTER*8 CONTNAME
      INTEGER CONTUNIT
      INTEGER*1 IA(200,40),ITIME(4)
      INTEGER ECB(16),IOCL(130),SENSBUFF(4)
      INTEGER*1 CMDBUF(8,11)
      INTEGER*1 INPBUF(64),OUBUF(64)
      INTEGER WRTC,WTDA,RDDA,ESRQ,DSRQ,CHAIN,SIL
      CHARACTER*1 NOTALK,NOLISN,ITALK,ILISN,UTALK,ULISN
      CHARACTER*4 TLKBUF,LSNBUF
      EXTERNAL IEEEXCPM,IEEELOCF,IEEESFMT,IEEECST
      DATA WRTC/Z'11000000'/,WTDA/Z'01000000'/
      DATA RDDA/Z'02000000'/,ESRQ/Z'63000000'/
      DATA DSRQ/Z'73000000'/,CHAIN/Z'40000000'/
      DATA SIL/Z'20000000'/
      DATA NOTALK/'_'/,NOLISN/'?'/,ITALK/'@'/,ILISN/' '/
      DATA UTALK/'C'/,ULISN/'#'/
      DATA CMDBUF/Z'40',Z'30',Z'36',Z'41',Z'43',Z'53',Z'05',0,
     *           Z'40',Z'30',Z'36',Z'41',Z'43',Z'53',Z'06',0,
```

```
     ,              Z'40',Z'30',Z'36',Z'41',Z'43',Z'53',Z'09',0,
     ,              Z'40',Z'30',Z'36',Z'41',Z'43',Z'53',Z'0A',0,
     ,              Z'40',Z'30',Z'36',Z'41',Z'43',Z'53',Z'0B',0,
     ,              Z'40',Z'30',Z'36',Z'41',Z'43',Z'53',Z'0C',0,
     ,              Z'40',Z'30',Z'36',Z'41',Z'43',Z'53',Z'0D',0,
     ,              Z'40',Z'30',Z'36',Z'41',Z'43',Z'53',Z'0E',0,
     ,              Z'40',Z'30',Z'36',Z'41',Z'43',Z'53',Z'B6',0,
     ,              Z'40',Z'30',Z'36',Z'41',Z'43',Z'53',Z'B7',0/
C    ******  NEXT STATEMENTS WILL INITIALI
C    ******  THEY ARE PERFORMED ONLY ON FIRST CALL TO SUBROUTINE  ****
      IF(INITDONE.EQ.1)GOTO 30
      INITDONE=1
      FCB=0
      FCB(3)=5          !TIMEOUT AT 5 SECONDS
      FCB(10)=Z'10000000'+IEEELOCF(SENSBUFF)
      CONTUNIT='IEE'
      CONTNAME='U07E80  '
      CALL IEEECST(CONTUNIT,CONTNAME,FCB,-1,ISTAT)
      IF(ISTAT.NE.0)GOTO 900
      FCB(9)=IEEELOCF(IOCL(1))
      FCB(7)=IEEELOCF(*915)          !WAIT I/O ERROR RETURN
      TLKBUF=NOTALK//NOLISN//ULISN//ITALK
      LSNBUF=NOTALK//NOLISN//ILISN//UTALK
      IOCL(1)=WRTC+IEEELOCF(TLKBUF)+2
      IOCL(2)=CHAIN+2           !ULISN,ITALK
      IOCL(4)=CHAIN+7           !COMMAND + LABEL (7 BYTES)
      IOCL(5)=WRTC+IEEELOCF(LSNBUF)+2
      IOCL(6)=CHAIN+2           !ILISN,UTALK
      IOCL(7)=RDDA+IEEELOCF(INPBUF)
      IOCL(8)=5                 !RECEIVE 5 BYTES
C AFTER INITIALIZATION, CALLS TO CDS COME HERE
30    IOCL(3)=WTDA+IEEELOCF(CMDBUF(1,MLAB))
      CALL IEEEXCPM(FCB,ISTAT)
      IF (ISTAT.NE.0)GOTO 920
      II=(MLAB-1)*4+135
      IA(II,JJ)=0
      IA(II+1,JJ)=INPBUF(1)
      IA(II+2,JJ)=INPBUF(2)
      IA(II+3,JJ)=INPBUF(3)
      GOTO 999
5998  FORMAT(' CDS ERROR ON NO-WAIT RETURN')
C     CALL M:END
C    **********   ERROR RETURNS    *******
 900  WRITE(5,5000)ISTAT
 5000 FORMAT(' IEEECST ERROR: ',I8)
      GOTO 999
 905  WRITE(5,5005)ISTAT
5005  FORMAT(' IEEEDST ERROR: ',I8)
 915  WRITE(5,5015)FCB(4)
 5015 FORMAT(' I/O ERROR, FCB(4)=',Z8)
      GOTO 999
```

```
  920     WRITE(5,5020)ISTAT
 5020     FORMAT(' IAEEXCPM ERROR, STATUS=',Z8)
  700     CONTINUE
  999     RETURN
          END
$ASSIGN LIB TO IEEELIB
$ASSIGN DIR TO IEEEDIR
$ASSIGN SLO TO SLODMP
$EXECUTE CATALOG
ASSIGN 5 TO ACQERR
ASSIGN 6 TO DEV=M91000 BLOC=N
ENVI RESI
CATALOG ACQUISIT 5
$EOJ
```

5. LOG-ON PROCEDURE AND PROGRAM START-UP

1. Turn power on to computer and peripherals.
2. Load cartridge tape.
3. Set CAPS LOCK on keyboard.
4. Hit carriage return (CR).
5. Computer should respond with "operator error" and the prompt //. Then type CLE (CR).
6. Again the prompt //, type RST.
7. Computer will respond with a PSW and INST then prompt //. Type IPL=0800 (CR).
8. Computer will boot from disk and issue the prompt >>. Type TE (CR).
9. Computer will initialize memory and then request date and time. Use the format : MM/DD/YY;HH:MM:SS (CR).
10. Enter CR in response to question about swap volume channel.
11. Computer will then initialize the terminal set-up and the IEEE-488 controller. The computer will then ask you to press attention for TSM. Type @@A.
12. When asked for the ownername, type SYSTEM (CR).
13. When asked for the key, type CR.
14. You are now in the MPX-32 operating system.

At this point you can enter the text editor (see chapter 6) by typing EDIT XX (CR). Or you can run the batch file GTE which will run the latest load modules STEP1, STEP2, and STEP3 (depending on which files have last been compiled and catalogued of BZLi or EBLEi).

If you run GTE the computer will ask you to mount the cartridge tape. Check to see that the green load light on the tape unit is on before typing in your response R (CR). You will know that all is working well if you are receiving print outs to the CRT and printer. If there is a program hang-up the safest thing to do is to type @@P. When you get the // prompt type HALT (CR). Then proceed from step 5. It is usually best to flick the power switches of the CDS and RTP's before bringing the system back up to clear any possible hang-ups in these devices.

A sample copy of the log-on procedure is shown on the following page.

```
PSW CC00EDIA INST 00070001              WAIT          HALT
//CLE
//RST
PSW 00000000 INST 00000000                            HALT
//IPL=0800

    00000000      00000000                             RUN

  >>TE

MEMORY INITIALIZATION STARTED.....
MEMORY INITIALIZATION COMPLETED....
ENTER DATE AND TIME:05/29/86:09:30:25
ENTER SWAP VOLUME CHANNEL AND SUBADDRESS
   (OR <CR> IF SYSTEM VOLUME)
INITIALIZATION COMPLETE
TERMINAL SET-UP COMPLETE
IEEE 488 CONTROLLER INITIALIZATION
FULL INITIALIZATION
CONTROLLER NAME=U27E80
CONTROLLER U27E80   INITIALIZED SUCCESSFULLY
SYSTEM READY    PRESS ATTENTION FOR TSM...



MPX-32 REVISION 2.2  MPXSYS11 TSM
(C) COPYRIGHT 1981 GOULD INC., COMPUTER SYSTEMS DIVISION, ALL RIGHTS RESERVED
ENTER YOUR OWNERNAME:SYSTEM
ENTER KEY
●●●●●●●●●●●●●●●●
  ************************************************************
  *                                                          *
  *        GOULD INC., COMPUTER SYSTEMS DIVISION             *
  *                                                          *
  *             MPX-32 REVISION 2 RELEASE 2.2                *
  *                                                          *
  ************************************************************
TSM>GTE
```

6. EDITING AND MODIFYING THE PROGRAMS

A summary of the EDIT directives is shown on pages 42 and 43 (ref. 6). To enter the text editor you type EDIT XX, where XX is any 2-letter workfile name (if it does not exist the editor creates the file for you). Then you would call on the USE directive to select the file you wish to edit (eg. USE BZL1). At this point you would use the other directives to modify your file. The format is as follows:

DIR I/J

where DIR is the directive, I is the starting line number and J the ending line number. Some examples:

LIST 1/20	lists lines 1 thru 20
DEL 1,2	deletes lines 1 and 2
COL 10	collects code startong at line 10 and continuing until you type a CR in response to the line number prompt
INS 10	inserts one line of code at line 10
CHA 1/10 \AAA\\BBB\	change the character string AAA in lines 1 thru 10 to the string BBB
COP 3/6 to 10.1	copies lines 3,4,5,6 to lines 10.1 thru 10.4

The letters L (last), F (first), A (all), C (current), or N (next) can be substituted for the line numbers:

LIST F/L	list from first to last lines of file
LIST A	list all of file (same as above)
CHA A \AAA\\BBB\	change all occurrences of AAA to BBB

Once you have completed all modification you would store the file under some filename:

STO NEWFILE UNN SCR

The UNN specifies that the file be stored un-numbered and the SCR specifies that if NEWFILE already exists it should be scratched and replaced by the current contents of you workfile.

Since the files BZLi and EBLEi have all control directives necessary for compiling and cataloguing as batch files you need then only type NEWFILE to obtain a load module STEPi of your modified file. If errors have occurred during Fortran compilation you can list the file SLOFOR (LIST SLOFOR) to determine the cause. If an error occcurs during cataloguing, you can list the file SLODMP (LIST SLODMP).

1.2 Directive Summary

EDIT directives are summarized below and described in detail in the Directives section. Most EDIT directives can be abbreviated to three characters. Valid abbreviations are indicated by underlining. EDIT directives are keywords and cannot be used as file names. Back slashes, forward slashes, and commas are special characters used in EDIT directives and cannot be used in file names.

The break key can be used to interrupt operations in progress. Use of the break key is not recommended during SAVE and STORE operations.

Most EDIT directive parameters can be entered in any order. If directive parameter input is order-dependent, the restriction is noted in the directive description.

Directive	Function
APPEND	Appends text to end of a line or group of lines.
BATCH	Copies work file or specified file into batch stream.
CHANGE	Replaces a character string with another character string.
CLEAR	Clears work file.
COLLECT	Adds lines of text.
COMMAND	Displays the last four directives performed.
COPY	Copies existing text to work file.
DELETE	Deletes lines.
EXIT	Ends current EDIT session.
INSERT	Inserts lines of text.
LIST	Lists text on terminal screen.
MODIFY	Allows a one-for-one replacement of characters in an existing line.
MOVE	Moves lines of text within the work file.
NUMBER	Renumbers lines in work file.
PREFACE	Inserts characters at beginning of lines.
PRINT	Copies work file or specified permanent file to SLO file.
PUNCH	Copies work file or specified permanent file to SBO file.
REPLACE	Replaces existing lines with new lines.

RUN	Copies work file or specified file into batch stream (same as the BATCH directive).
SAVE	Saves work file compressed in permanent file.
SCRATCH	Deletes a permanent file.
SET DELTA	Sets an increment for line numbering.
SET TABS	Modifies tab settings.
SET VFA	Sets automatic verification.
SET VFN	Inhibits automatic verification.
SHOW	Shows current line increment, files, or tab settings.
STORE	Stores work file uncompressed in permanent disc file.
USE	Copies a permanent file into a cleared work file.
WORKFILE	Accesses a different work file.

REFERENCES

Gould Inc., S.E.L. Computer Systems Division Manuals:

1. Gould MPX-32 Release 2.2 Reference Manual, Vol. 1, July 1984, PN 323-001011-500.

2. Gould MPX-32 Release 2.2 Reference Manual, Vol. 2, July 1984, PN 323-001012-500.

3. Gould MPX-32 Release 2.2 Reference Manual, Vol. 3, July 1984, PN 323-001013-500.

4. Fortran-77+ Release 4.0 Reference Manual, June 1983, PN 323-001470-100.

5. Scientific Run-Time Library Release 4.0 Reference Manual, June 1983, PN 323-001060-300.

6. Gould MPX-32 Utilities Release 1.1 Reference Manual, February 1984, PN 323-003960-000.

7. Analog/Digital Interface (ADI) Release 2.0 Reference Manual, August 1983, PN 323-000440-100.

8. Analog/Digital Interface Models 7410 and 7410-1 Technical Manual, October 1976, PN 303-000440-000.

9. IEEE-488 Bus Controller Software Manual Release 1.0, April 1983, PN 321-003030-000.

10. Ieee-488 Bus Controller Model 8024 Technical Manual, February 1983, PN 303-003-030-000.

Computer Products, Inc., Measurement and Control Systems Division:

11. RTP7430 Series Digital and RTP7431/RTP7433 Series Universal Input/Output Controllers, October 1984, PN 980-0070-004X.

12. RTP7436 Series Universal Analog Input Card Set Technical Manual, August 1984, PN 980-0021-211H.

13. RTP7435/33 Pulse Counter Card Technical Manual, November 1981, PN 980-0021-060L.

14. RTP7438/80 Synchro/Resolver-to-Digital Input Card Technical Manual, August 1983, PN 980-0021-240A.

15. RTP7438/50 8-Channel Analog Input/Output Card Technical Manual, May 1984, PN 980-0021-244A.

Colorado Data Systems:

16. 53A Smart Hardware System Operating Manual
53A-127 IEEE-4888 Communication Card, September 1983.
53A-171 Control Card, March 1978.
53A-420 Presorting Arinc-429 Receiver Card, March 1984.

APPENDIX A

SRTL SUBROUTINE CALLS

4.2.2 M:ACTIV (ref. 5)

The M:ACTIV subroutine activates a specified task. The task assumes the owner name of the caller.

Calling Sequence

CALL M:ACTIV (taskname,[istatus],[*label])

taskname — An INTEGER*8 variable that specifies the load module name. This argument must be one to eight ASCII characters, left justified and blank filled.

istatus — An INTEGER variable. If specified, the type of error that occurred during allocation is returned. Possible istatus values are:

0	Successful activation.
1	Invalid attempt to multicopy a unique task.
2	Specified file not in System Directory.
3	File is password protected.
4	File does not contain valid data.
5	No dispatch queue entry available.
6	Read error on System Directory.
7	Read error on load module.
8	No free MIDL space.
9	Insufficient memory.
10	No physical memory available.
50	Missing parameter (e.g., taskname).

*label — A statement label to which control is returned if an error exists.

Programming Consideration

If istatus and *label are omitted and an error occurs, the current task aborts with an RSxx, where xx is one of the listed istatus codes.

4.2.20 M:RSUM (ref. 5)

The M:RSUM subroutine resumes a suspended task.

Calling Sequence

CALL M:RSUM (itask, *label)

itask — An INTEGER*8 variable that specifies the task name or task number. A task name must be one to eight ASCII characters, left justified and blank filled. A task number must be right justified in the doubleword and zero filled.

*label — The statement label to which control is returned if the task is not in the CPU dispatch queue.

Programming Considerations

If the task is not suspended or not in CPU dispatch queue, the request is ignored.

Both the task making the M:RSUM call and the task being resumed must have the same ownername.

The suspended task must have been cataloged into the system directory.

4.2.23 M:SSPND

(ref. 5)

The M:SSPND subroutine suspends the calling task for either a specified number of time units or an indefinite time, as requested (the time unit is set at SYSGEN). A one-shot timer entry resumes a task suspended for a specified time interval. M:RSUM resumes either a task suspended for an indefinite time interval or a task suspended for a specified number of time units that have not yet expired. To suspend another task, use the subroutine X:SUSP.

Calling Sequence

CALL M:SSPND (itime, *label)

itime — An INTEGER variable interpreted as follows:

- 0 A suspension for an indefinite period of time.
- +n The positive number of units to elapse before the caller is resumed. The actual time elapsed can vary by one time unit because of system design.

*label — The statement label to which control is returned if timed suspension is requested and no timer entries are available.

4.2.25 M:TDAY

(ref. 5)

The M:TDAY subroutine returns the time of day as computed from the real-time clock interrupt counter. The counter is initialized by a SYSGEN parameter. The clock rolls over at midnight.

Calling Sequence

CALL M:TDAY (itime)

itime — An INTEGER*1 array that contains, upon return, the following four elements:

- itime(1) Hours (0-23)
- itime(2) Minutes (0-59)
- itime(3) Seconds (0-59)
- itime(4) Number of interrupts (within current second)

4.3.1 X:ANYW (ref. 5)

The X:ANYW subroutine places the calling task in a state waiting for the completion of any no-wait request, the receipt of a message, or a break interrupt. The task is removed from the associated ready-to-run list and placed in the any-wait list. A return is not made until one of the wait conditions has been satisfied or until the specified time-out value has expired.

Calling Sequence

CALL X:ANYW (itime,[istatus],[*label])

itime — An INTEGER variable interpreted as follows:

- 0 — If wait for an indefinite period is requested.
- -n — Contains the negative number of time units to elapse before the wait is terminated. The actual time elapsed can vary by one time unit because of system design.

istatus — An INTEGER variable. If specified, istatus is set to a completion code upon return from the subroutine. Possible istatus values are:

0	Normal completion.
47	Invalid time interval request.
50	Missing parameter.
51	Invalid parameter.

*label — The statement label to which control is returned if an error exists.

Programming Consideration

If istatus and *label are omitted and an error occurs, the current task aborts with an RSxx, where xx is one of the listed istatus values.

4.3.22 X:GMSGP (ref. 5)

The X:GMSGP subroutine, when called from the message receiver routine of a task that has received a message interrupt, transfers message parameters into the designated receiver buffer and posts the owner name and task number of the sending task into the Parameter Receive Block (PRB).

Calling Sequence

CALL X:GMSGP (prbname,[istatus],[*label])

prbname — An INTEGER array that specifies the PRB. Refer to the MPX-32 System Reference Manual, Volume 1, for a description of the contents of the PRB. Note that the parameter receiver buffer address within the PRB must be aligned on a word boundary.

istatus — An INTEGER variable. If specified, istatus is set to a completion code upon return from the subroutine. Possible istatus values are:

0	Normal status.
1	Invalid PRB address.
2	Invalid receiver buffer address.
3	No active send request.
4	Receiver buffer length exceeded.

*label — The statement label to which control is returned if an error exists.

Programming Consideration

If istatus and *label are omitted and an error occurs, the current task aborts with an RSxx, where xx is one of the listed istatus values.

2.7.6.2 Parameter Receive Block (PRB) (ref. 1)

The Parameter Receive Block (PRB) is used to control the storage of passed parameters into the receiver buffer of the destination task. The same format PRB is used for both message and run requests. The address of the PRB must be presented when either the M.GMSGP or M.GRUNP services are invoked by the receiving task.

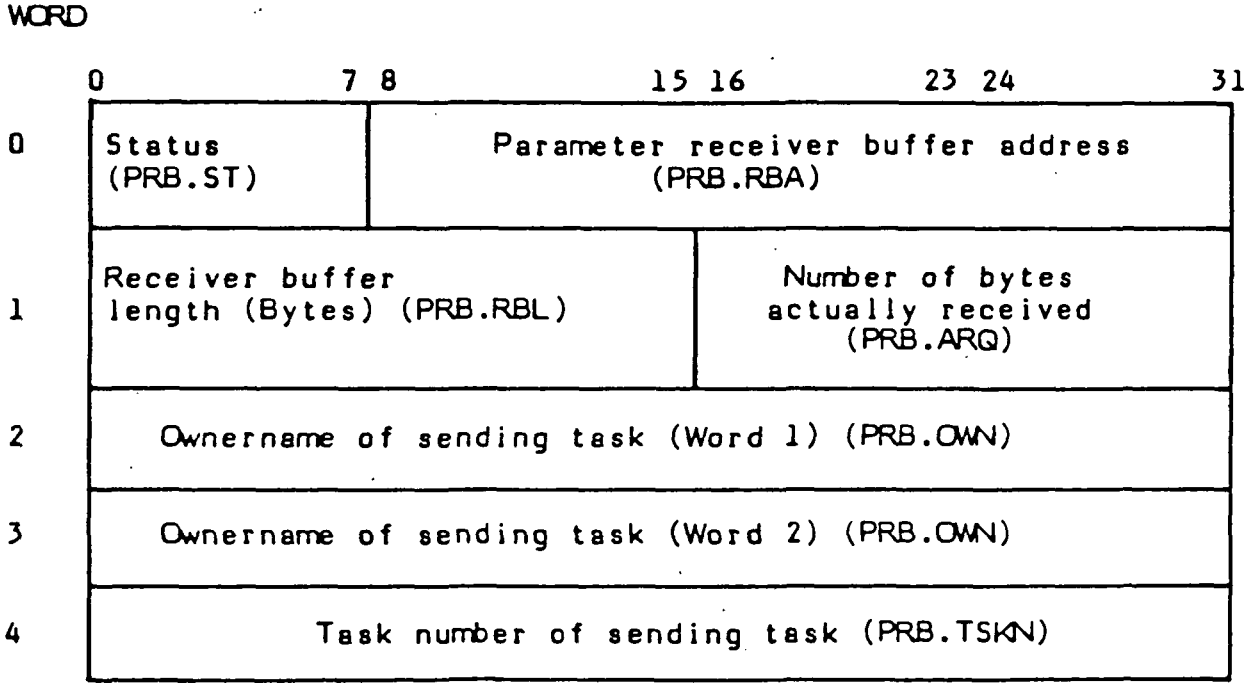


Figure 2-5. Parameter Receive Block (PRB)

WORD 0

Bits 0-7 Status-value encoded status byte

Code	Definition
0	Normal status
1	Invalid PRB address (PRB.ER01)
2	Invalid receiver buffer address or size detected during parameter validation (PRB.RBAE)
3	No active send request (PRB.NSRE)
4	Receiver buffer length exceeded (PRB.RBLE)

Bits 8-31 Parameter Receiver Buffer Address - This field contains the word address of the buffer, into which the sent parameters are stored.

WORD 1

Bits 0-15 Receiver Buffer Length - Contains the length of the receiver buffer (0-768 bytes).

Bits 16-31 Number of Bytes Actually Received - This value is set by the operating system and is clamped to a maximum equal to the receiver buffer length.

WORDS 2,3

Bits 0-63 Ownername of Sending Task - Set by the operating system to contain the ownername of the task which issued the parameter send request.

WORD 4

Bits 0-31 Task Number of Sending Task - Set by the operating system to contain the task activation sequence number of the task which issued the parameter send request.

4.3.24 X:ID (ref. 5)

The X:ID subroutine obtains information about an active task, including the calling task, when one of the following parameters is known:

- Task number
- Task load module name
- Owner name
- Pseudonym name of the task

The caller must supply at least one of these parameters; remaining parameters are returned by the subroutine. Repeated calls may be made to the subroutine to find all copies of a multicopied task.

Calling Sequence

CALL X:ID (index, taskno, taskname, ownername, pseudo,[istatus],[*label])

index — An INTEGER variable that must be zero for the initial search. This argument is used by the system subroutine to control the points at which previous searches are discontinued and a search of the dispatch queue is begun. Index must not be a constant because the X:ID subroutine updates the contents of this parameter for retrieval of further entries when repeated calls are made. When all matching tasks are identified, index is returned with value zero.

taskno — An INTEGER variable that specifies the task number. It must be set to zero if the task number is unknown.

taskname — An INTEGER*8 variable or character variable that specifies the task name. This argument must be one to eight ASCII characters, left justified and blank filled.

ownername — An INTEGER*8 variable or character variable that specifies the task's owner name. This argument must be one to eight ASCII characters, left justified and blank filled. If the task owner name is unknown, ownername must contain either a zero or all blanks.

pseudo — An INTEGER*8 variable or character variable that specifies pseudonym. This argument must be one to eight ASCII characters, left justified and blank filled. If the pseudonym is unknown, pseudo must contain either a zero or all blanks.

istatus — An INTEGER variable. If specified, this argument is set to a completion code upon return from the subroutine. Possible istatus values are:

0	Normal completion.
50	Missing parameter.
51	Task not found.
52	All tasks found.

*label — The statement label to which control is returned if an error exists.

Programming Consideration

If istatus and *label are omitted and an error occurs, the current task aborts with an RSxx, where xx is one of the listed istatus values.

4.3.26 X:RCVR (ref. 5)

The X:RCVR subroutine establishes the address of a routine that is to be entered to receive messages sent by other tasks.

Calling Sequence

CALL X:RCVR (*label$_1$,[istatus],[*label$_2$])

*label$_1$ The statement label of the-beginning of the task's message receiver routine.

istatus An INTEGER variable. If specified, this argument is set to a completion code upon return from the subroutine. Possible istatus values are:

0	Normal completion.
50	Missing parameter.
51	Parameter out of range (invalid receiver address).

*label$_2$ The statement label to which control is returned if an error exists.

Programming Consideration

If istatus and *label$_2$ are omitted and an error occurs, the current task aborts with an RSxx, where xx is one of the listed istatus values.

X:RSML

4.3.29 X:SMSGR

The X:SMSGR subroutine sends up to 768 bytes to a specified destination task. This subroutine may also accept up to 768 bytes as return parameters.

Calling Sequence

CALL X:SMSGR (ipsb,[istatus],[*label])

ipsb — An INTEGER array that specifies the Parameter Send Block (PSB). Note that the send buffer address and the return parameter buffer address within the Parameter Receive Block (PRB) must be aligned on a word boundary.

istatus — An INTEGER variable. If specified, this argument is set to a completion code upon return from the subroutine. Possible istatus values are:

0	Normal initial status.
1	Task not found in Dispatch Queue Entry (DQE).
10	Invalid priority.
11	Invalid send buffer address.
12	Invalid return buffer address.
13	Invalid no-wait mode end-action routine address.
14	Memory pool unavailable.
15	Destination task queue depth exceeded.
16	Invalid PSB address.

*label — The statement label to which control is returned if an error exists.

Programming Consideration

If istatus and *label are omitted and an error occurs, the current task aborts with an RSxx, where xx is one of the listed istatus values.

2.7.6.1 Parameter Send Block (PSB) (ref. 1)

The Parameter Send Block (PSB) is used to describe a send request issued from one task to another. The same PSB format is used for both message and run requests. The address of the PSB (doubleword bounded) must be presented as an argument when either the M.SMSGR or M.SRUNR services are invoked.

When a load module name is supplied in Words 0 and 1 of the PSB, the operating system defaults to a search in the system directory only. For activations in other than the system directory, a pathname or RID vector must be supplied.

Please note a task number, not a load module name, must be used if sending a message request or if sending a run request to a multicopied task which is waiting for a run request.

WORD

	0 7	8 15	16 23	24 31
0	Load module name (or task number if message or run request to multicopied task)			
1	Load module name or pathname vector or RID vector if activation (or zero if message or run request to multicopied task)			
2	Priority (PSB.PRI)	Reserved	Number of bytes to be sent (PSB.SQUA)	
3	Reserved	Send buffer address (PSB.SBA)		
4	Return parameter buffer length (bytes) (PSB.RPBL)		Number of bytes actually returned (PSB.ACRP)	
5	Reserved	Return parameter buffer address (PSB.RBA)		
6	Reserved	No-wait request end action address (PSB.EAA)		
7	Completion status (PSB.CST)	Processing start status (PSB.IST)	User Status (PSB.UST)	Options (PSB.OPT)

Figure 2-4. Parameter Send Block (PSB)

WORD 0

Bits 0-31

For Send Message: Task Number of the task to receive the message.
For Run Request: Zero if using pathname vector or rid vector in word 1, else task number (word 1 must be zero), else characters 1-4 of the name of the load module to receive the run request.

WORD 1

Bits 0-31	For Send Message: Zero. For Run Request: Zero if using task number in word 0, else pathname vector or rid vector (word 0 must be zero), else characters 5-8 of the load module to receive the run request.

WORD 2

Bits 0-7	Priority - This field contains the send request priority (1-64). When sending messages, this field indicates the priority of the message in the queue. When sending run requests, this field indicates the execution priority of the task to be activated. If the value of this field is zero, the priority defaults to the execution priority of the sending task. This field is examined for all sending tasks. The requested message or run request priority must be within the field's range. For unprivileged tasks, the requested priority must be lower or equal to the sending task's priority. If the requested priority is not within the field's range, an error occurs.
Bits 8-15	Reserved.
Bits 16-31	Number of Bytes to be Sent - This field specifies the number of bytes to be passed (0-768) with the message or run request.

WORD 3

Bits 0-7	Reserved.
Bits 8-31	Send Buffer Address - This field contains the word address of the buffer containing the parameters to be sent.

WORD 4

Bits 0-15	Return Parameter Buffer Length - Contains the maximum number of bytes (0-768) that may be accepted as returned parameters.
Bits 16-31	Number of Bytes Actually Returned - This field is set by the send message or run request service upon completion of the request.

WORD 5

Bits 0-7	Reserved.
Bits 8-31	Return Parameter Buffer Address - Contains the word address of the buffer into which any returned parameters will be stored.

WORD 6

Bits 0-7 — Reserved.

Bits 8-31 — No-Wait Request End Action Address - Contains the address of a user routine to be executed at an interrupt level upon completion of the request.

WORD 7

Bits 0-7 — Completion Status - This bit encoded field contains completion status information posted by the operating system as follows:

Bit	Meaning When Set
0	Operation in progress (busy).
1	Destination task was aborted before completion of processing for this request.
2	Destination task was deleted before completion of processing for this request.
3	Return parameters truncated (attempted return exceeds return parameter buffer length).
4	Send parameters truncated (attempted send exceeds destination task receiver buffer length).
5	User end action routine not executed because of task abort outstanding for this task (may be examined in abort receiver to determine incomplete operation).
6-7	Reserved.

Bits 8-15 Processing Start (Initial) Status - This value encoded field contains initial status information posted by the operating system as follows:

Code	Definition
0	Normal initial status.
1	Message request task number invalid.
2	Run request load module name not found in directory.
3	Reserved.
4	File associated with run request load module name does not have a valid load module format.
5	Dispatch Queue Entry (DQE) space is unavailable for activation of the load module specified by a run request.
6	An I/O error was encountered while reading the directory to obtain the file definition of the load module specified in a run request.
7	An I/O error was encountered while reading the file containing the load module specified in a run request.
8	Memory unavailable.
9	Invalid task number for run request to multicopied load module in RUNW state.
10	Invalid priority specification. Note: An unprivileged task may not specify a priority which is higher than its own execution priority.
11	Invalid send buffer address or size.
12	Invalid return buffer address or size.
13	Invalid no-wait mode end action routine address.
14	Memory pool unavailable.
15	Destination task receiver queue is full.

Bits 16-23 User Status - As defined by sending and receiving tasks.

Bits 24-31 Options - This field contains user request control specification. It is bit encoded as follows:

Bit	Meaning When Set
24	Request is to be issued in no-wait mode.
25	Do not post completion status or accept return parameters. This bit is examined only if bit 24 is set. When this bit is set, the request is said to have been issued in the "no call-back mode".

4.3.36 X:XMEA (ref. 5)

The X:XMEA subroutine exits an end action routine associated with a no-wait message send request.

Calling Sequence

```
CALL X:XMEA
```

4.3.37 X:XMSGR (ref. 5)

The X:XMSGR subroutine exits the message receiver routine of the calling task. This subroutine must be called after the task has received a message from another task.

Calling Sequence

CALL X:XMSGR (rxbname,[istatus],[*label])

rxbname An INTEGER array that specifies the Receiver Exit Block (RXB).

istatus An INTEGER variable. If specified, this argument is set to a completion code upon return from the subroutine. Possible istatus values are:

0	Normal completion.
50	Missing parameter.
51	Parameter out of range.

*label The statement label to which control is returned if an error exists.

Programming Consideration

If istatus and *label are omitted and an error occurs, the current task aborts with an RSxx, where xx is one of the listed istatus values.

2.7.6.3 Receiver Exit Block (RXB) (ref. 1)

The Receiver Exit Block (RXB) is used to control the return of parameters and status from the destination (receiving) task to the task that issued the send request. It is also used to specify receiver exit-type options. The same format RXB is used for both messages and run requests. The address of the RXB must be presented as an argument when either the M.XMSGR or M.XRUNR services are called.

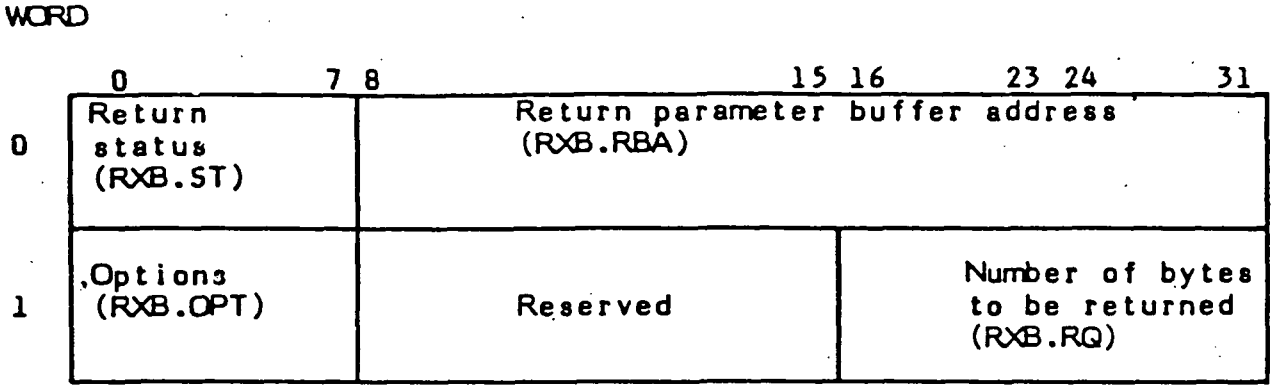


Figure 2-6. Receiver Exit Block (RXB)

WORD 0

Bits 0-7 Return Status - Contains status as defined by the receiver task. Used to set the user status byte in the Parameter Send Block (PSB) of the task which issued the send request.

Bits 8-31 Return Parameter Buffer Address - Contains the word address of the buffer containing the parameters which are to be returned to the task which issued the send request.

WORD 1

Bits 0-7 Options - This field contains receiver exit control options. It is encoded as follows:

Value	Exit Type	Meaning
0	M.XRUNR	Wait for next run request.
	M.XMSGR	Return to point of task interrupt.
1	M.XRUNR	Exit task, process any additional run requests. If none exist, perform a standard exit.
	M.XMSGR	N/A

Bits 8-15 Reserved.

Bits 16-31 Number of Bytes to be Returned - This field contains the number of bytes (0-768) of information to be returned to the sending task.

4.3.38 X:XNWIO (ref. 5)

The X:XNWIO subroutine must be called to exit a no-wait I/O end action routine (both normal and error end action routines). The subroutine returns control to the point of interruption.

Calling Sequence

```
CALL X:XNWIO
```

6.2.4.1 IBITS (ref. 5)

The IBITS function extracts a specified bit subfield from a specified argument. The result field is right justified and the remaining bits in the returned value are set to zero.

Calling Sequence

iresult = IBITS (ivalue, isrcpos, len)

ivalue, isrcpos, len Integer expressions

A bit field is extracted from the value of ivalue, starting from bit position isrcpos and extending left for len bits. The value of (isrcpos+len) must be less than or equal to 32.

APPENDIX B

ANALOG-TO-DIGITAL INTERFACE CALLS

2.2.4.9 ADIUSR

(ref. 7)

ADIUSR is a subroutine used to support the feature of the ADI accepting a user buffer in which to build the physical IOCBs. ADIUSR is only a calling mechanism, as the user will be responsible for constructing the proper IOCH and IOCBs and defining the user buffer. This service will be available only to resident tasks.

Calling Sequence:

```
CALL ADIUSR(ioch,userbuff,sizebuff,busy[,eobeaddr])

(or)

BL      ADIUSR
DATAW   5W or 4W
ACW     IOCH
ACW     USERBUFF
ACW     SIZEBUFF
ACW     BUSY
ACW     EOBEADDR    (Optional)
```

INTEGER*n

ioch	n=4	specifies the address of the IOCH.
userbuff	n=4	specifies address of user buffer in which to build the physical IOCBs.
sizebuff	n=4	specifies the size of the user buffer in which to build the physical IOCBs (must be six times the number of IOCBs plus 10 (words)).
busy	n=4	specifies error return address.
eobeaddr	n=4	specifies the no-wait I/O EOB end action address (Optional).

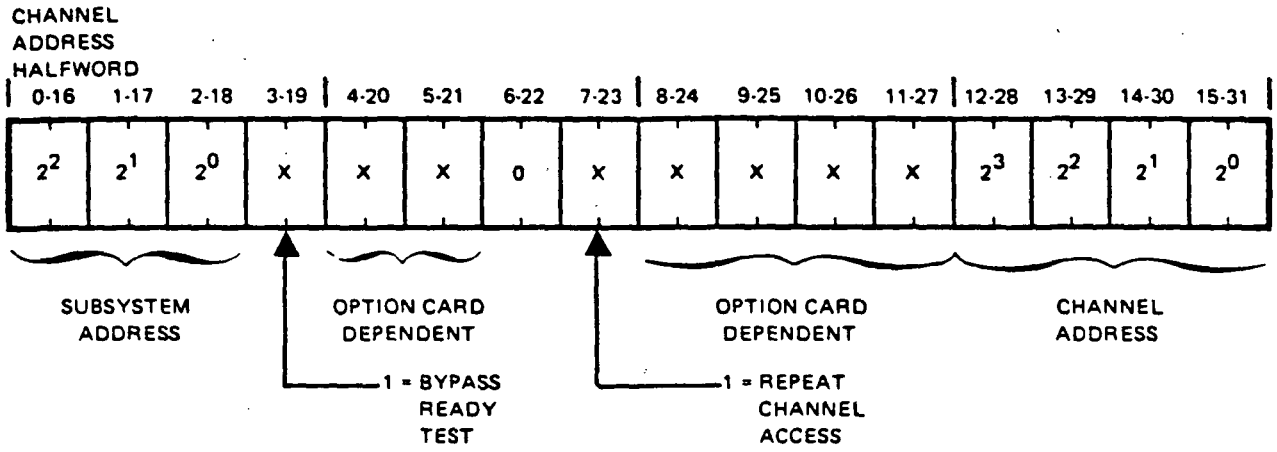
1.9.3 Digital Output (DO) Subsystem

(ref. 7)

Digital Output transfers can be used with the Digital or Universal Input/Output subsystems which contain various types of output option channels.

Output can be performed in the Random, Repeat, or Sequential mode. The Repeat capability is specified by setting bit 7 of the Channel Address Halfword (CAHW) and resetting bit 1 of the IOCB Opcode.

To provide better performance on Digital Outputs using the Sequential mode, the ADI automatically increments the subsystem address after the last channel in a subsystem has been processed. This allows several subsystems to be addressed in sequential mode with a single IOCB, eliminating the need for multiple IOCBs with Command Chaining.



NOTE: FOR MOST OPTION CARDS:

BITS 4-20	BITS 5-21	
0	0	NO CHANGE
0	1	DISABLE EAI
1	0	ENABLE EAI
1	1	INVALID

820684

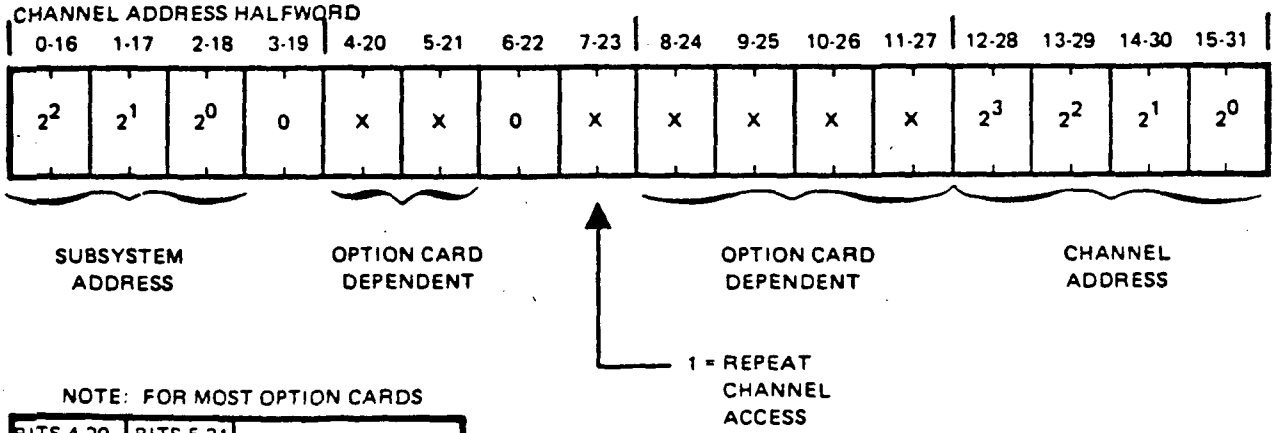
The Output mode provides the capability to bypass the ready test. When bit 3 of the Channel Address Halfword (CAHW) is set, the specified function is performed immediately, whether or not the channel is ready. This feature should be used when processing analog inputs in a Universal subsystem. To process analog inputs, both an output sequence and an input sequence are required. The output sequence is needed to start the digitizing of the data, while the input sequence is needed to input the digitized data. The Bypass Ready Capability should be specified on the output sequence to allow all digitizing to be started in parallel. Without the ability to bypass the ready status, the digitizing of the second channel of multichannel input would not be started until the digitizing of the first channel was complete.

The Command Transfer operation which is specified in the IOCB Opcode is also provided by the ADI controller. When this mode is specified, the ADI transfers only the Channel Address Halfwords (CAHW) to the subsystem and channel selected by the CAHW. The CAHW contains the channel dependent control bits which allow the software to command a block of channels (e.g. enable or disable interrupts) without performing an actual data transfer. These channels may be either input or output.

These channel dependent control bits are also passed to the subsystems on data transfer (Digital Inputs or Digital Outputs). The Command Transfer function should be specified only for Digital Outputs in Random or Sequential transfer modes. The Repeat mode is not available when using command transfers.

1.9.4 Digital Input (DI) Subsystem (ref. 7)

Digital Input transfers are used with the Digital or Universal Input/Output subsystems, which contain various types of input option channels. Digital input uses most of the same addressing modes as Digital Output, however the Bypass Ready capability is not provided. A Digital Input transfer will not complete until the channel provides ready status to the ADI. When an operation specifies an input sequence of two channels, the second channel will not be addressed until the first channel has completed the data transfer. This mode of operation will cause the ADI to stop processing a sequence of channels in the event a nonexistent channel is addressed. If required, a software watchdog timer routine can be provided to issue a terminate to recover from such a situation.



NOTE: FOR MOST OPTION CARDS

BITS 4-20	BITS 5-21	
0	0	NO CHANGE
0	1	DISABLE EAI
1	0	ENABLE EAI
1	1	INVALID

820683

(ref. 7)

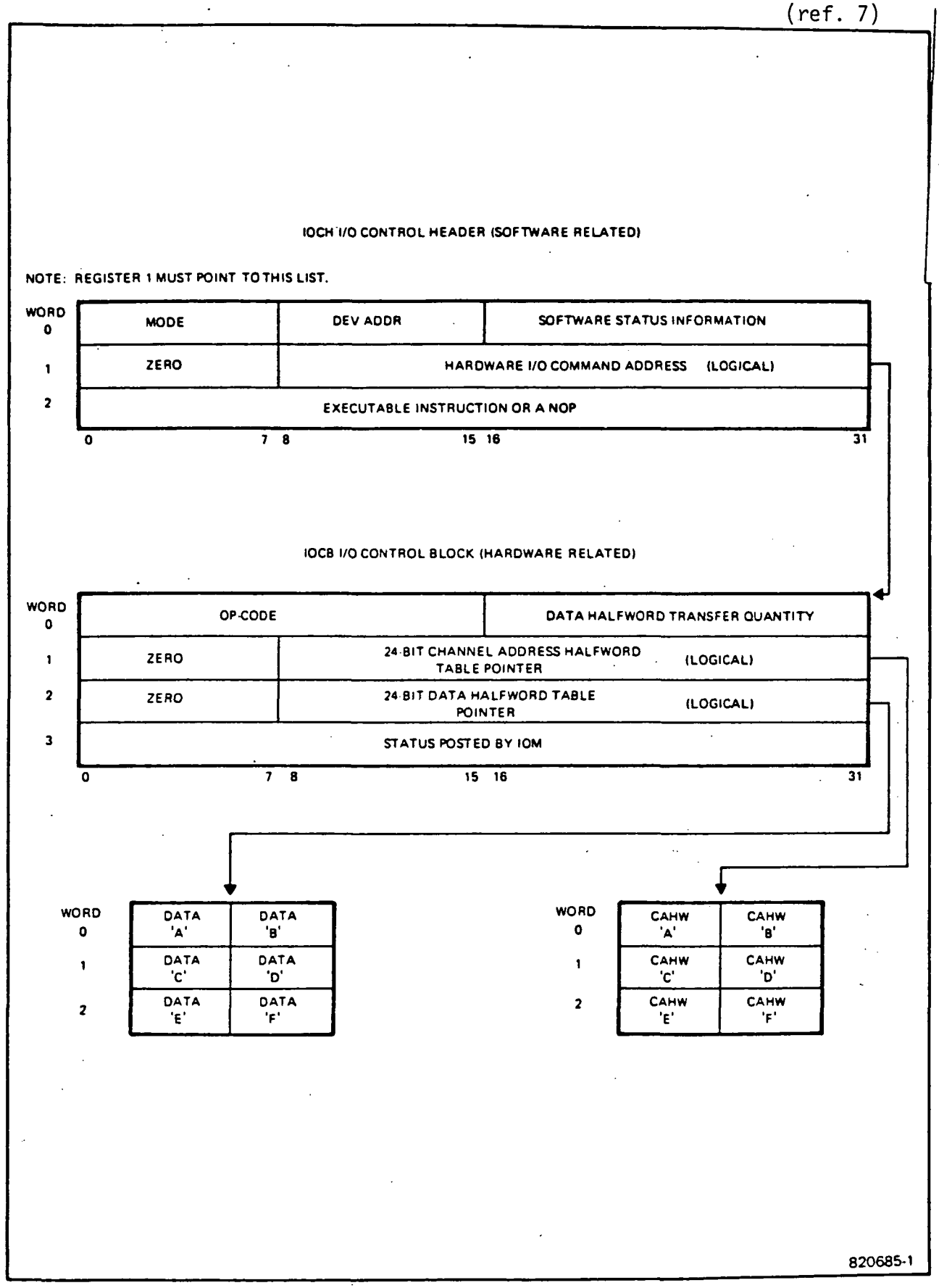


Figure 2-1. I/O Format Description
(Sheet 1 of 3)

(ref 7)

IOCH
SOFTWARE RELATED

WORD 0

BITS 0-7 MODE SPECIFICATION. THESE EIGHT BITS ENABLE THE USER TO SPECIFY THE FOLLOWING CONTROL INFORMATION.

BIT	MEANING
0	SETTING BIT ZERO CAUSES THE I/O TO BE ISSUED IN THE NON-WAIT MODE. IF THIS BIT IS NOT SET, THE USER'S PROGRAM WILL BE SUSPENDED.
1	SETTING THIS BIT WILL CAUSE THE HANDLER TO INHIBIT THE SOFTWARE TIME OUT FEATURE OF MPX-32.
2	SETTING THIS BIT WILL PROHIBIT THE I/O OPERATION OF THE OTHER DEVICE BUS WHILE THIS I/O COMMAND IS BEING SERVICED BY THE HANDLER.
3	SETTING THIS BIT SPECIFIES THE USER IS TO BE ASYNCHRONOUSLY NOTIFIED UPON THE COMPLETION OF PROCESSING OF AN END-OF-BLOCK (EOB) INTERRUPT. (TO USE THIS FEATURE, THE I/O REQUEST MUST BE ISSUED IN THE NO-WAIT MODE AND AN EOB END ACTION RECEIVER ADDRESS (REGISTER 6) MUST BE SUPPLIED.)
4	SETTING THIS BIT INDICATES THE PHYSICAL IOCB(S) ARE TO BE CONSTRUCTED WITHIN THE USER'S DEFINED BUFFER AND NOT IN MEMORY POOL. (TO USE THIS FEATURE, THE USER MUST SUPPLY THE ADDRESS OF THE BUFFER IN REGISTER 4 AND THE SIZE OF THE BUFFER IN REGISTER 5 WHEN INITIATING THE I/O REQUEST.)
5	SETTING THIS BIT WILL CONVERT A TIC I/O OPERATION INTO AN EOL TO TERMINATE THE I/O OPERATION. THIS BIT HAS TO BE SET BY THE USER DURING THE TIC I/O OPERATION.
6,7	MUST BE ZERO.

BITS 8-15 THESE BITS CONTAIN THE DEVICE ADDRESS OF THE IOM THAT THE USER WISHES TO ADDRESS.

BITS 16-31 SOFTWARE STATUS INFORMATION. THESE 16 BITS CONTAIN THE SOFTWARE CONTROL AND STATUS INFORMATION AS DESCRIBED BELOW.

BIT	MEANING
16	I/O IN PROGRESS. THIS BIT IS SET BY THE SVC PROCESSOR WHEN THE I/O IS INITIATED AND IS RESET BY THE HANDLER WHEN END OF LIST IS RECOGNIZED.
17	ERROR CONDITION PRESENT. THIS BIT IS SET BY THE HANDLER WHENEVER A HARDWARE ERROR HAS BEEN DETECTED.
18	LOST INTERRUPT. THIS BIT IS SET WHENEVER THE LOST INTERRUPT TIMER EXPIRES.
19	MUST BE ZERO. (THE ADI HANDLER USES BIT 19 TO INDICATE I/O COMPLETION HAS BEEN REPORTED TO THE EXEC.)
20	MUST BE ZERO. THE ADI SVC PROCESSOR USES BIT 20 TO INDICATE THE IOCB(S) HAVE BEEN BUILT WITHIN THE USER'S BUFFER AND ALL ADDRESSES HAVE BEEN CONVERTED TO PHYSICAL FOR PROCESSING BY THE ADI.
21-31	MUST BE ZERO.

WORD 1

BITS 0-7 MUST BE ZERO.

BITS 8-31 HARDWARE I/O COMMAND ADDRESS POINTER. THIS 24-BIT ADDRESS IS PLACED IN THE TI DEDICATED LOCATION TO ENABLE THE IOM TO PROCESS THE USER'S IOCBS.

NOTE: ANY UNUSED ADDRESS BITS MUST BE ZERO (BITS 8-12 IF 24-BIT ADDRESSING IS NOT PERMITTED).

WORD 2

BITS 0-31 WORD 2 OF THE IOCH ALLOWS THE PRIVILEGED USER TO HAVE AN INSTRUCTION EXECUTED AT THE SI LEVEL OF THE HANDLER, SUCH AS ON RI INSTRUCTION. IF THE USER DOES NOT WANT TO USE THIS FEATURE, HE MUST PLACE A NOP-NOP IN THIS WORD. WORD 2 OF THE IOCH WILL BE EXECUTED WHENEVER AN EOB OR EOL INTERRUPT OCCURS. THIS FEATURE IS ONLY PROVIDED FOR THE PRIVILEGED USER. WORD 2 OF THE IOCH IS NOT EXECUTED IF THE USER IS UNPRIVILEGED.

830541

Figure 2-1. I/O Format Description
(Sheet 2 of 3)

IOCB
HARDWARE RELATED

WORD 0

BITS 0-15 OP-CODE - THESE 16 BITS PROVIDE THE USER WITH A MEANS OF CONTROLLING THE VARIOUS REAL-TIME PERIPHERALS WHICH MIGHT BE CONFIGURED.

BIT	MEANING
0	SET = INPUT; RESET = OUTPUT DATA TRANSFERS
1	SET = SEQUENTIAL; RESET = RANDOM CHANNEL ACCESS
2	SET = DIGITAL; RESET = ANALOG SUBSYSTEM
3	SET = CONTINUE; RESET = TERMINATE ON ERROR
4	SET = INTERRUPT ON END OF BLOCK (IOCB)
5	SET = TRANSFER IN CHANNEL
6	SET = COMMAND CHAIN
7	SET = COMMAND DATA TRANSFER (DO ONLY)
8 AND 9	= 0
10	SET = AUTO GAIN RANGE ENABLE (WRAI ONLY)
11	SET = OTD SCAN (WRAI ONLY)
12	SET = OTD - NEGATIVE (WRAI ONLY)
13-15	CONTAIN SUBSYSTEM ADDRESS (ANALOG ONLY)

BITS 16-31 TRANSFER QUANTITY. NOTE: THE USER IS TO PROVIDE THE HALFWORD TRANSFER QUANTITY.

WORD 1

BITS 0-7 MUST BE ZERO

BITS 8-31 CONTAIN THE 24-BIT CHANNEL ADDRESS HALFWORD TABLE POINTER OR, FOR A TIC, THE ADDRESS OF THE NEXT IOCB (HARDWARE).

WORD 2

BIT 0 THIS BIT IS SET BY THE ADI SVC PROCESSOR WHEN AN UNPRIVILEGED USER PERFORMS I/O, THIS TELLS THE SYSTEM THAT THIS BUFFER ADDRESS HAS BEEN VALIDATED AND, THEREFORE, REDUCES SYSTEM OVERHEAD.

BITS 1-7 MUST BE ZERO

BITS 8-31 24-BIT INPUT/OUTPUT DATA BUFFER POINTER.

WORD 3

BITS 0-31 ARE INITIALLY SET TO ZERO BY THE USER AND WILL CONTAIN ERROR STATUS ONLY WHEN IT OCCURS. SEE FIGURE 2-2.

820685-3

Figure 2-1. I/O Format Description (Sheet 3 of 3)

IOCB status is posted in Word 3 of the last IOCB processed within an IOCL. Status is posted upon normal or abnormal completion.

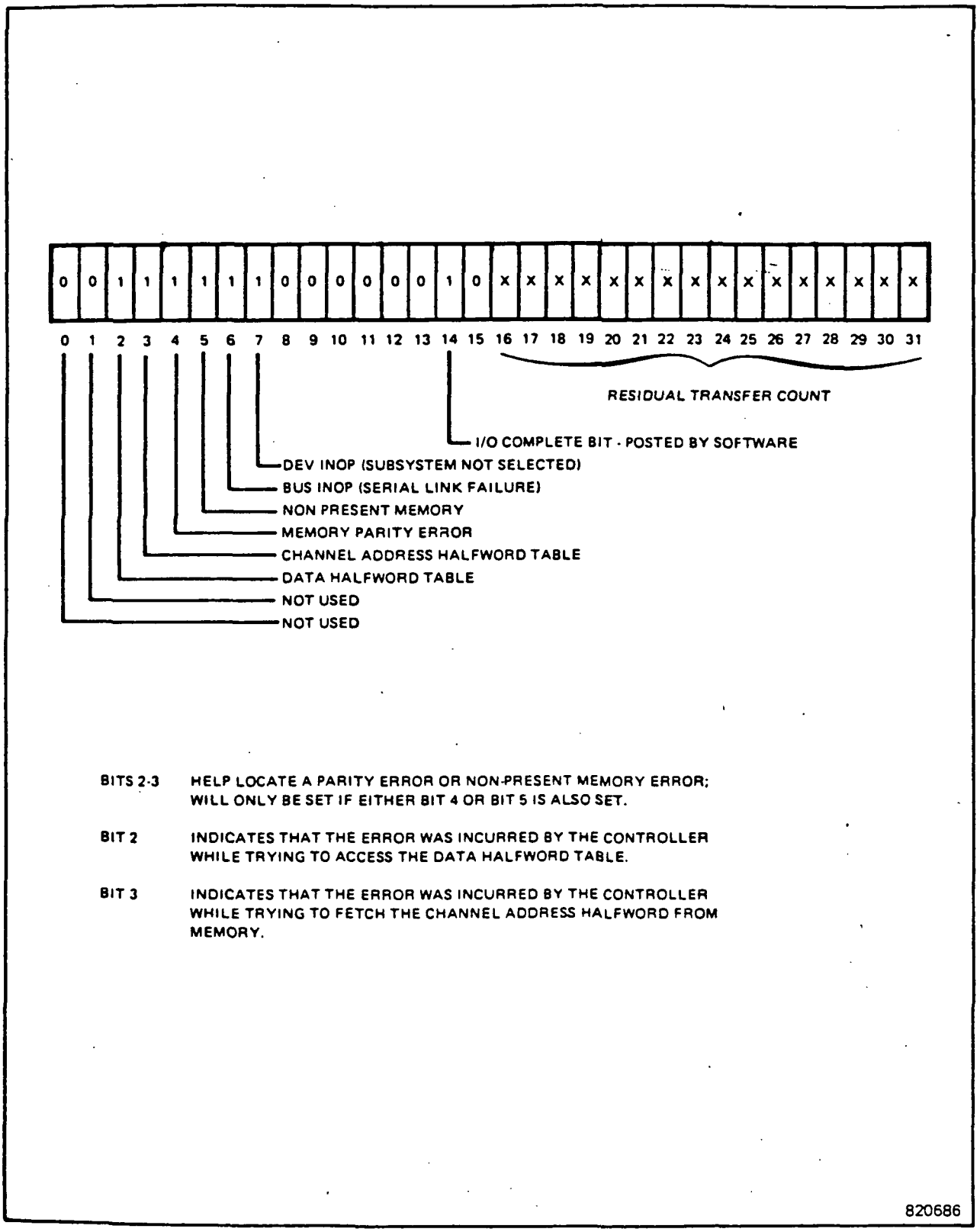
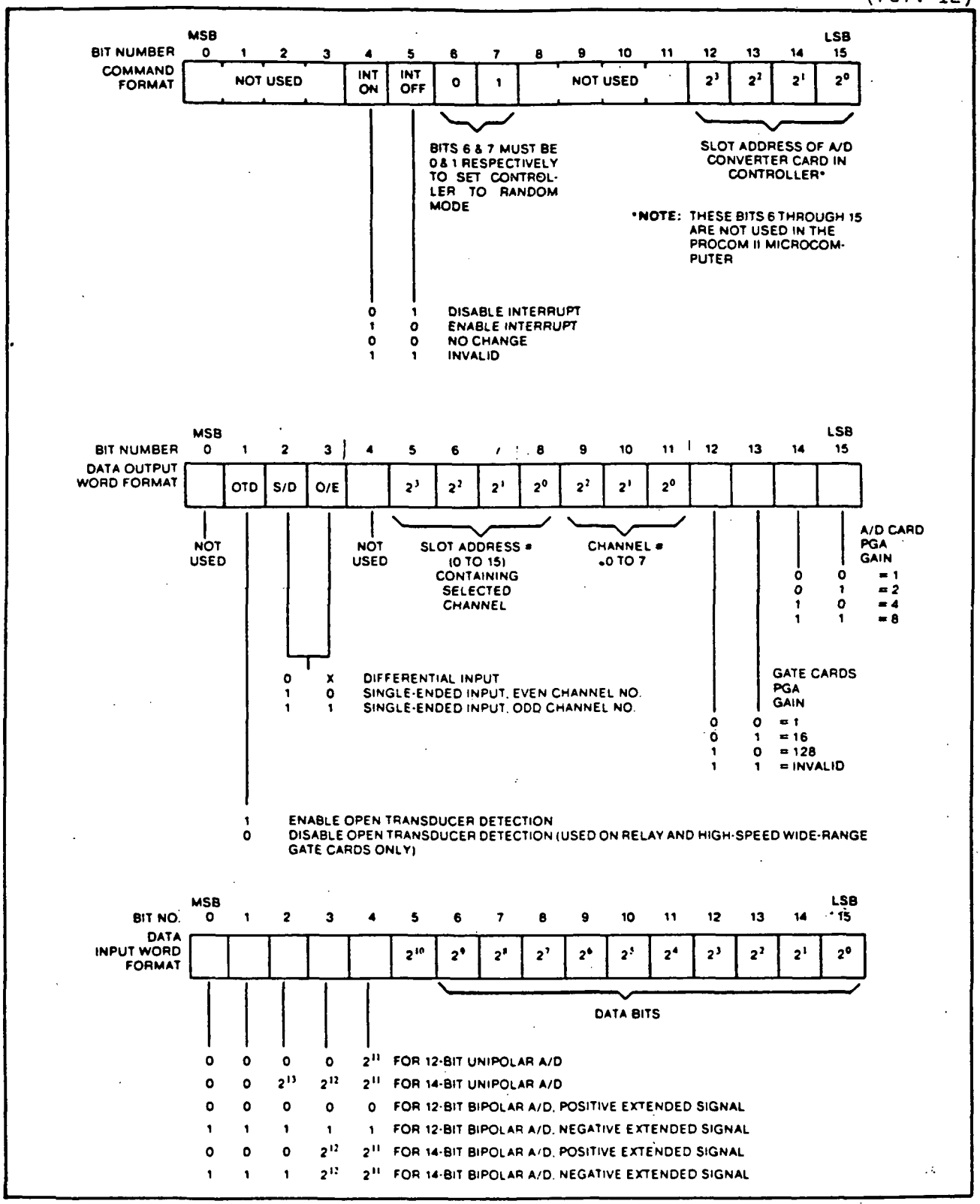


Figure 2-2. IOCB Status (Word 3 of the IOCB)



PA070-5282(A)

Figure 2-1. RTP Command and Data Word Formats

RTP7436 Universal Analog Input Gate card (ref.12)

Table 2-7. PROCOM II Analog Input Addresses: Solid-State, Relay and High-Speed Wide-Range Gates (8-Channel Differential Inputs)

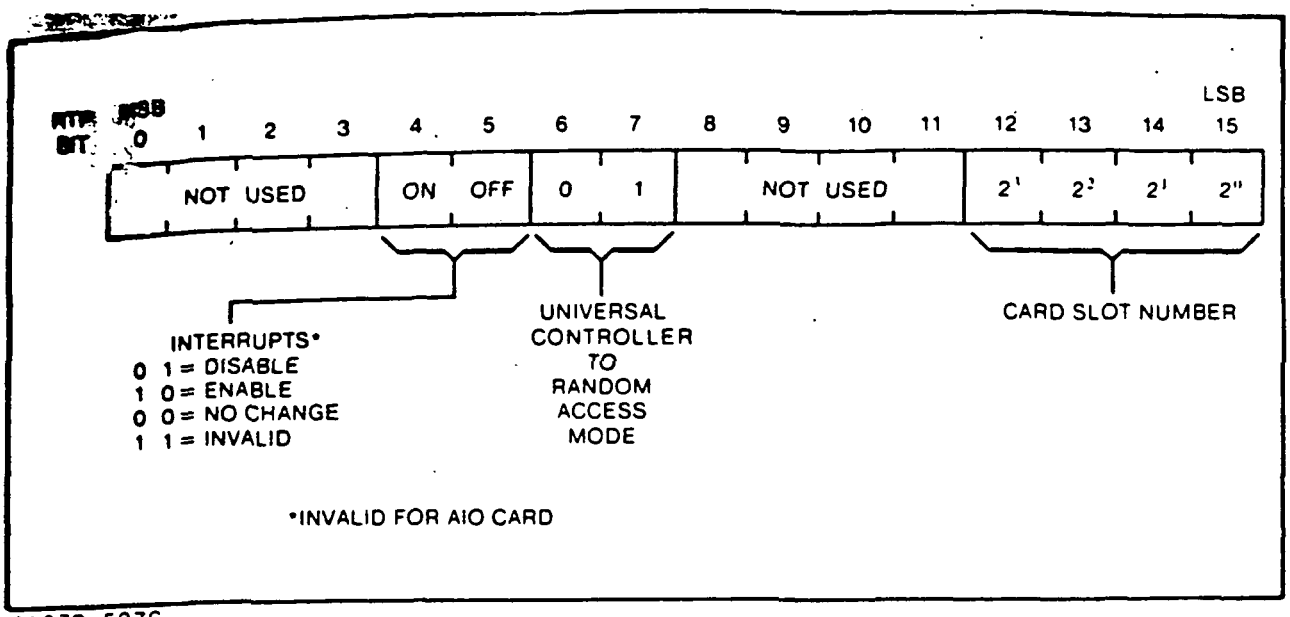
CARD SLOT	CARD DEVICE #	CHANNEL ADDRESSES							
		CH. # 00	CH. # 01	CH. # 02	CH. # 03	CH. # 04	CH. # 05	CH. # 06	CH. # 07
1	64 ('40')	0	1	2	3	4	5	6	7
2	68 ('44')	8	9	10	11	12	13	14	15
3	72 ('48')	16	17	18	19	20	21	22	23
4	76 ('4C')	24	25	26	27	28	29	30	31
5	80 ('50')	32	33	34	35	36	37	38	39
6	84 ('54')	40	41	42	43	44	45	46	47
7	88 ('58')	48	49	50	51	52	53	54	55
8	72 ('5C')	56	57	58	59	60	61	62	63

NOTE: The A/D card occupies one card slot. Those addresses will not be available for input channels.
'YY' = hexadecimal

Table 2-8. Universal Analog Input Card Set Programmable Gain Bit Selection For Various Full Scale Input Ranges of the Relay and High-Speed Wide-Range Gates

DATA OUTPUT WORD				PROGRAMMABLE GAINS				
BIT 12	BIT 13	BIT 14	BIT 15	GATE CARD	A D CARD	TOTAL CARD SET GAIN	FULL SCALE INPUT OF RELAY GATE CARD	FULL SCALE INPUT OF SOLID-STATE GATE CARD
0	0	0	0	1	1	1	10.24 V	10.24 V
0	0	0	1	1	2	2	5.12 V	5.12 V
0	0	1	0	1	4	4	2.56 V	2.56 V
0	0	1	1	1	8	8	1.28 V	1.28 V
0	1	0	0	16	1	16	640 mV	N/A
0	1	0	1	16	2	32	320 mV	N/A
0	1	1	0	16	4	64	160 mV	N/A
0	1	1	1	16	8	128	80 mV (Don't use)	N/A
1	0	0	0	128	1	128	80 mV (Preferred)	N/A
1	0	0	1	128	2	256	40 mV	N/A
1	0	1	0	128	4	512	20 mV	N/A
1	0	1	1	128	8	1024	10 mV	N/A

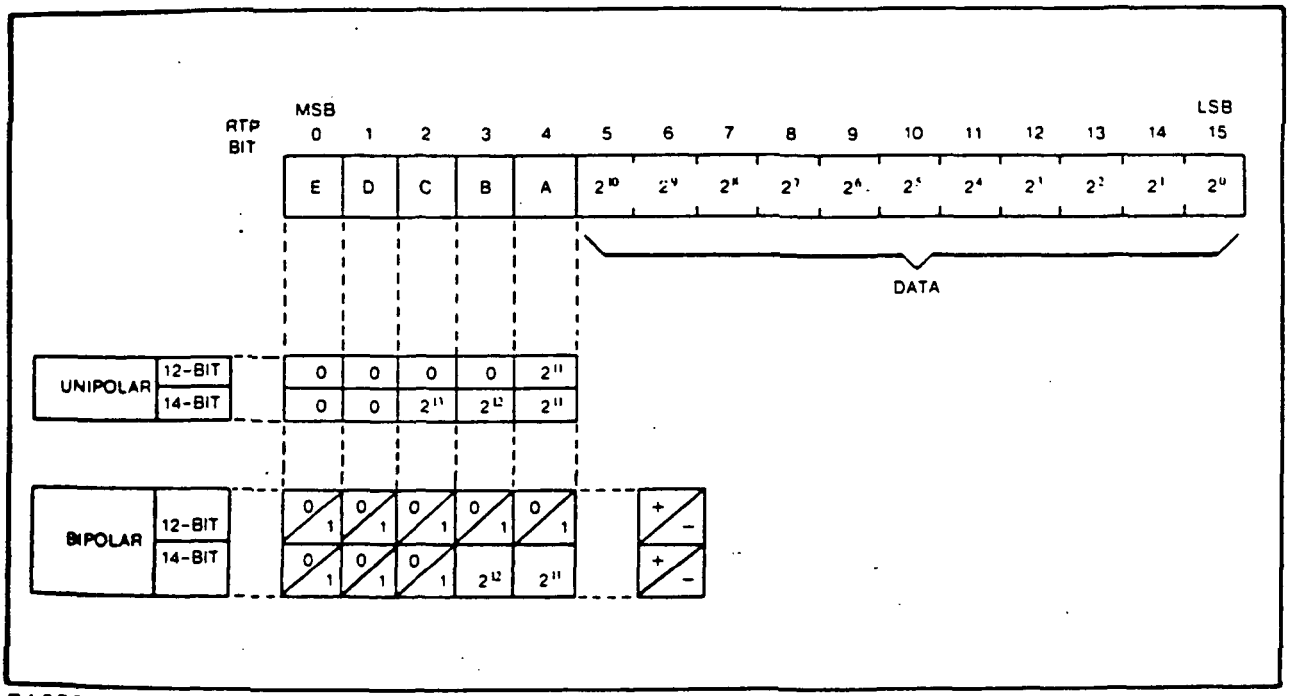
RTP7436 Universal Analog Input -gate card (ref 12).



PA070-5376

Figure 3-4. RTP Command (Input Data Transfers); Word Format

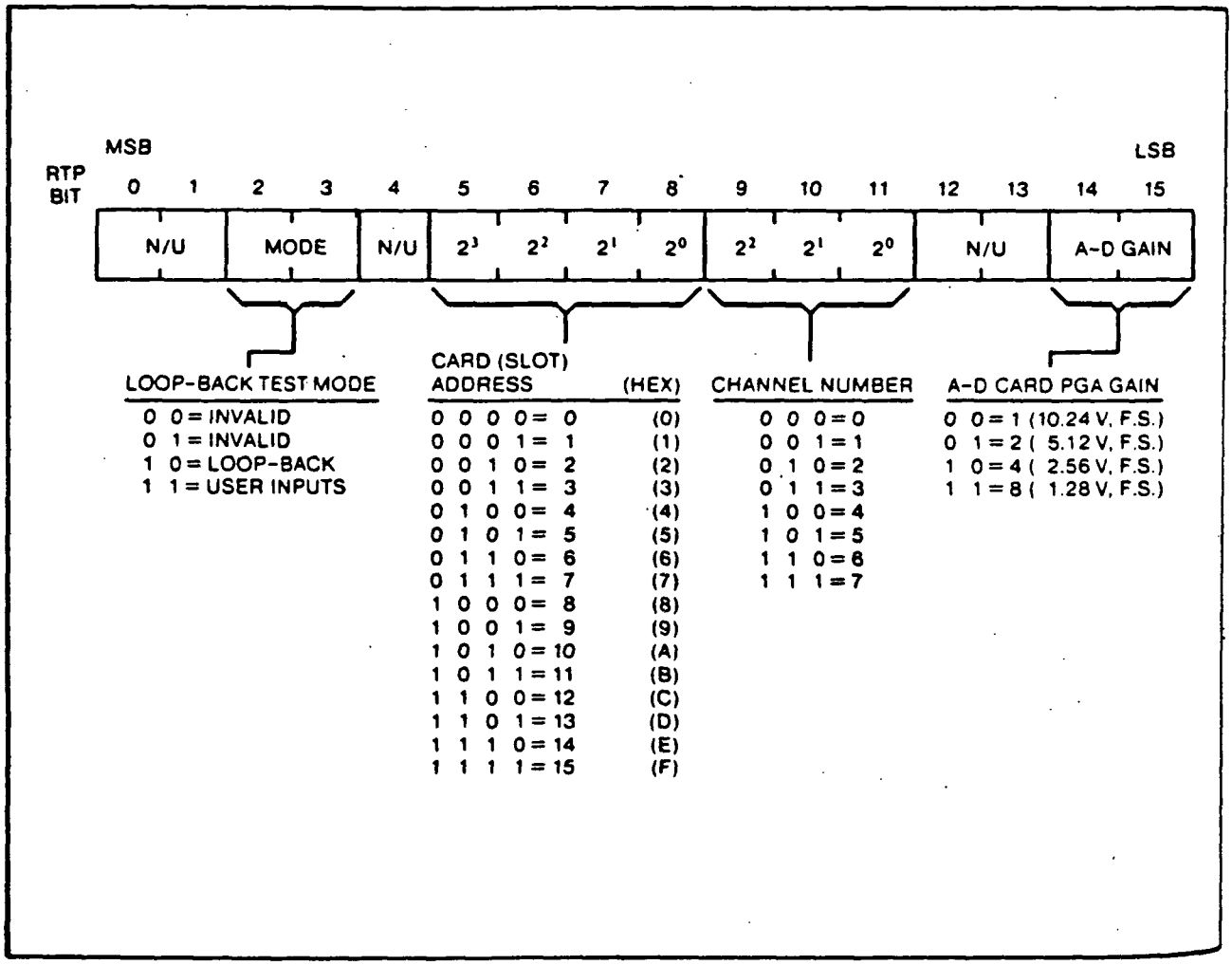
RTP7438/50 8-Cannel Analog Input/Output (ref. 15)



PA070-5377

Figure 3-5. RTP Input (Input Data Transfers); Word Format

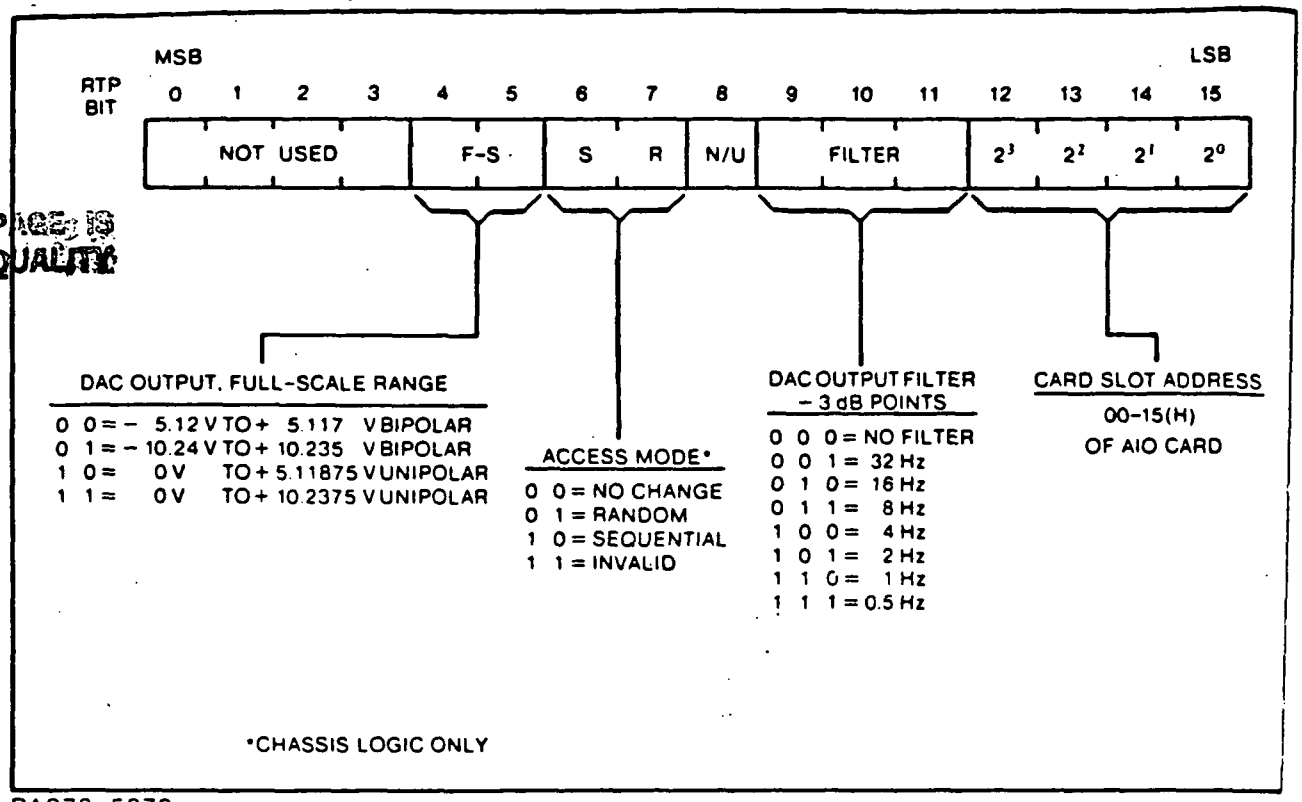
RTP7438/50 8-Channel Analog Input/Output (ref.15)



PA070-5378

Figure 3-6. RTP Output (Input Data Transfers); Word Format
RTP7438/50 8-Channel Analog Input/Output (ref. 15)

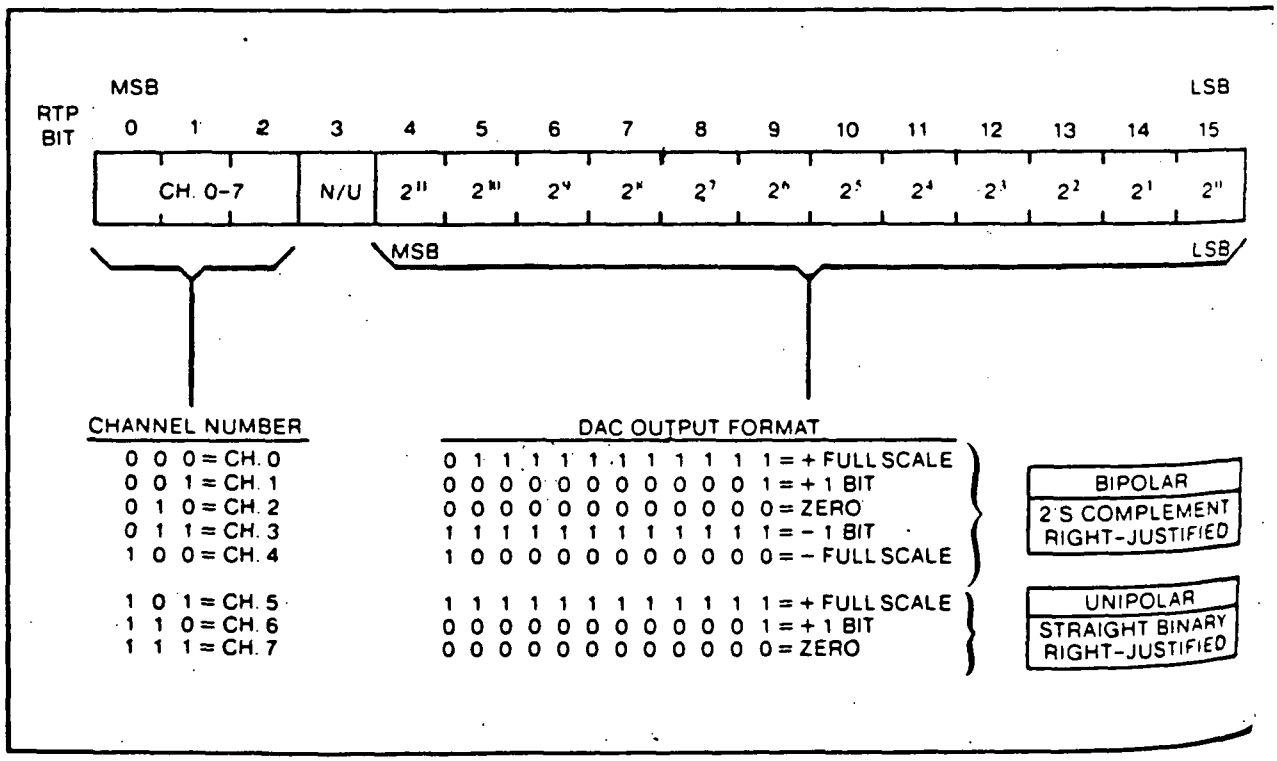
3-6



PA070-5379

Figure 3-8. RTP Command (Output Data Transfers); Word Format

RTP7438/50 8-Channel Analog Input/Output (ref.15)



PA070-5380

Figure 3-9. RTP Output (Output Data Transfers); Word Format

RTP 7438/50 8-Channel Input/Output (ref.15)

RTP I/O Card Slot 7, PSI Sensor #1

→ Input analog data at

Pin 404 CH 1

RTP I/O Card Slot 8, PSI Sensors 2,3,4+5

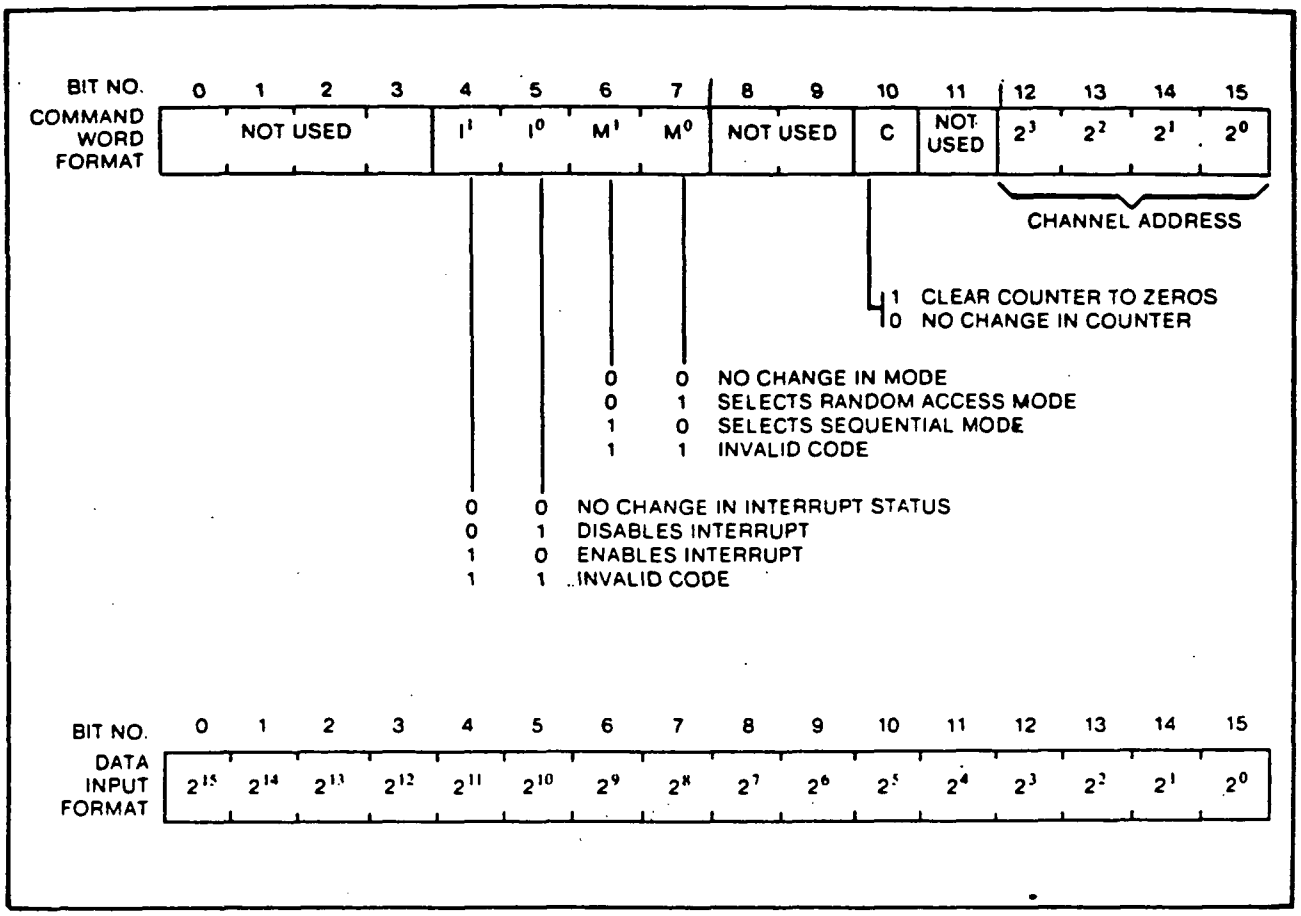
→ Input analog data on

Pin 407 CH 2
410 CH 3
413 CH 4
416 CH 5

Channel Selection Code For ESP-16 TL Pressure Sensors

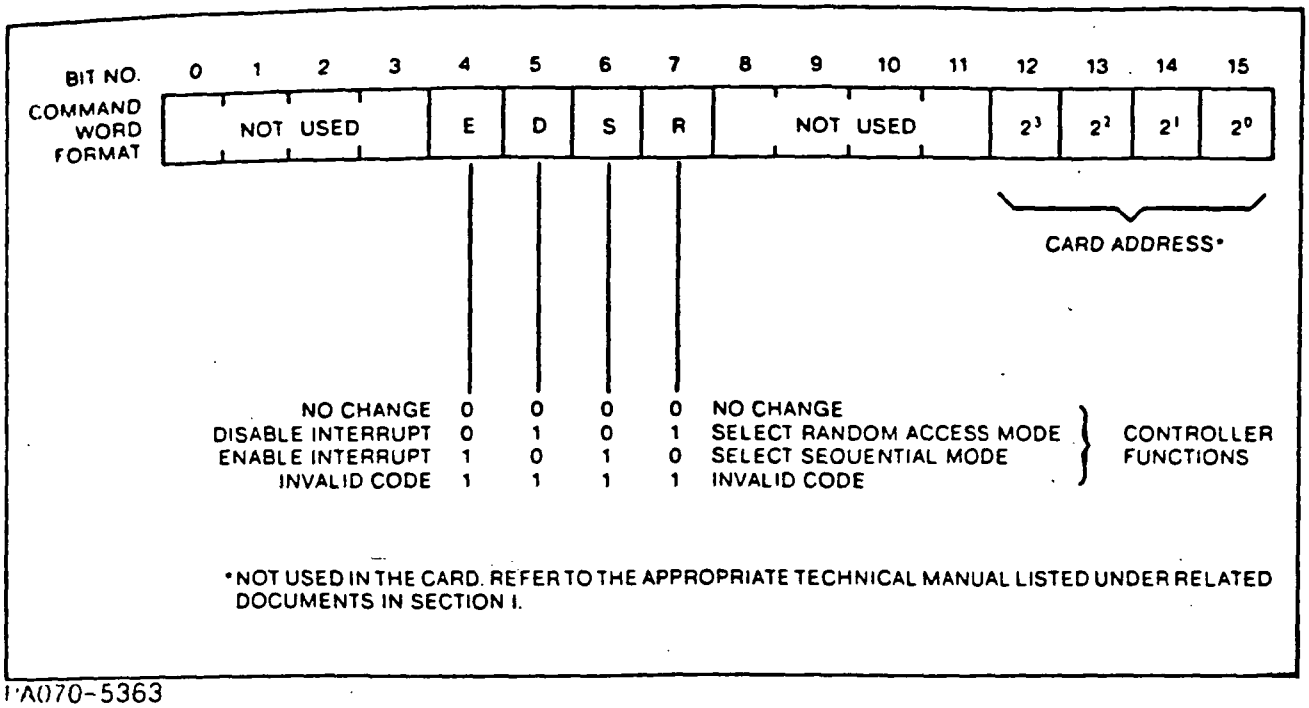
RTP I/O Pin #s	80 IFC Pin #s	Signal
301 CH0	29	A0 LSB
304 CH1	27	A1
307 CH2	25	A2
310 CH3	21	A3 MSB

Channel No.	A3	A2	A1	A0
1	0	0	0	0
2	0	0	0	1
3			1	0
4			1	1
5		1	0	0
6		1	0	1
7		1	1	0
8		1	1	1
9	1	0	0	0
10	1	0	0	1
11	1	0	1	0



PA070-5011

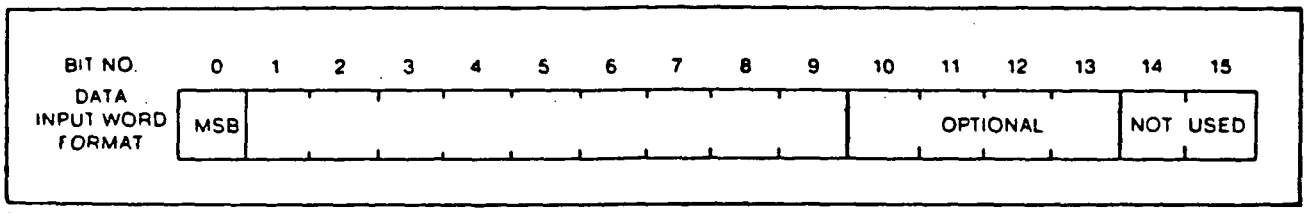
Figure 3-6. Command and Data Word Formats
RTP7435/33 Pulse Counter (ref. 13)



I'A070-5363

Figure 3-1. RTP Command; Word Format

RTP7438/80 Synchro Resolver-to-Digital (ref. 14)



I'A070-5364

Figure 3-2. Data Input Word; Word Format

RTP7438/80 Synchro Resolver-to-Digital (ref. 14)

Table 3-1. RTP Data Input Word; Bit Weights

BIT NO	2^N	LSB AS % OF FULL SCALE	RADIANS/BIT	DEGREES/BIT	MINUTES/BIT	SECONDS/BIT	MILS/BIT
0	2	50.	3 14159265	180	10,800.	648,000.	3200
1	4.	25	1.57079633	90.	5,400.	324,000.	1600
2	8	12 5	.78539816	45.	2,700.	162,000.	800
3	16.	6.25	.39269908	22.5	1,350.	81,000.	400
4	32.	3 125	.19634954	11.25	675.	40,500.	200
5	64	1.5625	09817477	5.625	337.5	20,250.	100
6	128	78125	04908739	2 8125	168 75	10,125.	50
7	256	390625	02454369	1 40625	84 375	5,062 5	25
8	512	1953125	.01227185	.703125	42 1875	2,531.25	12 5
9	1,024	09755625	00613592	.3515625	21.09375	1,265 6250	6.25
10	2,048	04882813	00306796	.1757813	10 54688	632 8125	3.125
11	4,096	02441406	.00153398	0878906	5.27344	316 4063	1.5625
12	8,192	01220703	.00076699	0439453	2 63672	158.2031	0 78125
13	16 384	00610352	.00038350	0219727	1 31836	79.1016	0.390625

LSB = Least Significant Bit

1 Radian = 57.2957795

RTP7438/80 Synchro Resolver-to-Digital (ref. 14)

APPENDIX C

IEEE-488 CONTROLLER CALLS

5.3.5 IEEECST

(ref. 9)

This subroutine allows a task to gain complete control of an IEEE-488 bus controller, thus creating a single task environment. In a single task environment, you must set up file control blocks (FCB) and logical input output control lists (IOCL). IEEECST dynamically assigns the controller.

Calling Sequence

CALL IEEECST(contunit,name,fcb,time,status)

Input:

contunit — an integer word variable or expression specifying the controller unit number.

name — an 8 byte character variable, left-justified and space-filled, holding the controller name as specified to SYSGEN.

fcb — the name of the 16 word FCB integer array to be used in all single task transfers for this controller.

time — an integer word variable used to specify the amount of time the calling routine will wait for connection. A positive value represents the number of timer units to wait before returning an error status. A value of zero signifies an indefinite wait. A negative value signifies that anything but immediate connection will fail.

Output:

status — an integer word variable that contains the status after execution of the called procedure.

5.3.8 IEEEDST

(ref. 9)

This subroutine disconnects a task running in single task mode and frees the controller for use by other tasks. Any parallel polls configured by subroutine IEEEPPC are automatically unconfigured by this call.

Calling Sequence

CALL IEEEDST(contunit,fcb,status)

Input:

contunit — an integer word variable or expression specifying the controller unit number.

fcb — the name of the 16 word FCB integer array used to connect the controller.

Output:

status — an integer word variable that contains the status after execution of the called procedure.

5.3.10 IEEELOCF

(ref. 9)

This subroutine, which is equivalent to LOCF in FORTRAN-77+, returns the logical address of a variable. It is used mainly in the single task environment to establish IOCLs.

Calling Sequence

address=IEEELOCF(variable)

Input:

variable — a variable name, subroutine name, or label (FORTRAN-77+ only). A label is expressed as *n where n is the label number.

Output:

address — the address of the specified variable.

5.3.26 IEEEXCPM

This subroutine provides a FORTRAN interface to the Execute Channel Program (EXCPM) entry for the handler in a single task environment.

Calling Sequence

CALL IEEEXCPM(fcb,status)

Input:

fcb — the name of the 16 word FCB integer array used to connect the controller.

Output:

status — an integer word variable that contains the status after execution of the called procedure. For further status information, FCB word 3 returns channel and device status while the sense buffer returns controller specific status.

Programming Considerations

1. You must set up a file control block (FCB), logical input/output command list (IOCL), and sense buffers before calling this routine.
2. I/O status is only checked for wait I/O. Otherwise, software status is returned in the status parameter. Examine the FCB and sense buffer for I/O status. A status value other than zero or three implies no I/O has been queued.

APPENDIX C

(ref. 9)

I/O COMMAND DOUBLEWORDS

C.1 Introduction

An I/O Command Doubleword (IOCD) contains information (e.g., command codes, buffer address, transfer count, and flags) describing an I/O operation. An I/O Command List (IOCL) comprises one or more IOCDs. All interface subroutines initiating I/O use IOCDs and IOCLs. In multitask mode, IOCLs are built by the interface subroutine. In single task mode, you are responsible for building IOCLs.

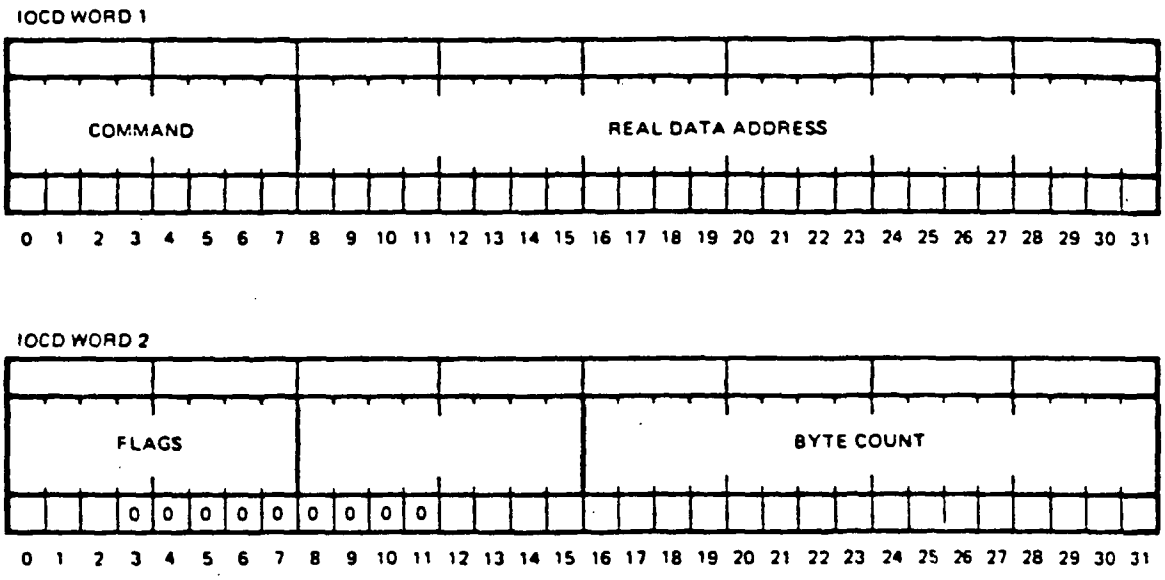
The IEEE-488 bus controller software uses the Execute Channel Program (EXCPM) facility for all I/O you initiate. This gives you direct control of the actions of the hardware. The EXCPM facility allows the use of physical or logical IOCLs, both of which will appear to be the IOCLs used to drive the hardware. However, note the following.

- An IOCL is specified as physical or logical by setting or resetting a bit in the FCB.
- Physical IOCLs use absolute buffer addresses. No address conversion is performed by the operating system. The IOCL is passed directly to the hardware. A task must be privileged and resident to use physical IOCLs.
- Privileged, resident tasks may also specify logical IOCLs.
- The address of the physical or logical IOCL is specified in word nine of the FCB.
- Full access to IOCLs and FCBs is available in the single task environment. Therefore, either physical or logical IOCLs can be used.
- Full access to IOCLs and FCBs is not available in the multitask environment. Therefore, only logical IOCLs can be used. The IOCL order is established by the interface subroutines to execute the required function.

C.2 IOCD Format

(ref. 9)

The format of an IOCD and an explanation of each field follows.



830393

The IOCD command field (Word 1, bits 0-7) specifies the command to be executed. Command codes (see Table C-1) are passed to the IEEE-488 bus controller for action. For example, the write control (WRTC) command instructs the controller to request an IOP write. The controller then transmits the data written to it by the IOP to a device or devices on the IEEE-488 bus (the ATN line is set to indicate the data is interpreted as IEEE-488 commands). The IOCD commands and their definitions are presented in Section C.3. Refer to the IEEE-488 Hardware Technical Manual for detailed information concerning each command.

The real data address field (Word 1, bits 8-31) contains the starting byte address of a data buffer. Data sent across the IEEE-488 bus (either to a device or from a device) as a result of an IOCD being executed are in the form of multiline messages, device dependent commands, or data. These messages, commands, and data are located at the address specified by the real data address field. For example, when a task calls IEEEREAD, untalk (UNT) is the first of several multiline messages sent across the bus to the specified device. The address of these multiline messages is specified in the real data address field of the IOCD. In the execution of an IOCD containing a read data (RDDA) command, the data buffer specified by the real data address field will receive data sent across the bus by the specified device.

(ref. 9)

The flags field (Word 2, bits 0-7) contains channel programming flag bits. When set, the bits indicate specific action to be taken as follows:

Bit	Action	Result
0	Data chain	The present operation continues once the current IOCD byte count is exhausted. The operation is continued using the real data address, control flags, and byte count of the next IOCD in sequence. The command field of the next IOCD is ignored.
1	Command chain	IOCL execution continues once the current IOCD byte count is exhausted. Execution continues using the command field, real data address, control flags, and byte count of the next IOCD in sequence.
2	Suppress incorrect length	When an incorrect length occurs in the status doubleword, command chaining is to continue.
3	Reserved	Must be zero.
4	Reserved	The Post Program Controlled Interrupt (PPCI) normally indicated by this flag bit is not available to users of this software.
5-7	Reserved	Must be zero.

Word 2 bits 8-15 are reserved and must be zero.

The byte count field (Word 2, bits 16-31) specifies the number of bytes to be transferred to or from memory. The byte count field in an IOCD that excutes a read data (RDDA), for example, indicates the number of bytes to be transferred to memory beginning at the specified real data address.

C.3 IOCD Commands

Table C-1 lists the IOCD commands and their bit configuration. The IOP software convention for defining data lines differs from the IEEE-488 convention; the most significant bit for the IOP is bit 0, while the most significant bit for the IEEE-488 is bit 8. This manual uses the IOP convention, i.e., bit 0 is the most significant bit.

A brief description of each command follows Table C-1. For more detailed information about the IOCD commands, refer to the IEEE-488 Bus Controller Technical Manual.

(ref. 9)

C.3.15 Read Data Command (op code = 02_{16})

The Read Data command causes data to be read by the controller by setting ATN fa and requesting a read. If the controller is a slave, the Read Data command causes t controller to wait to be addressed by the master controller before receiving data fr the master controller.

C.3.24 Write Control Command (op code = 11_{16})

The Write Control command instructs the controller to request an IOP write. The data that are written to the controller are then sent across the IEEE-488 bus with the ATN line asserted, i.e., they are interpreted as multiline messages.

C.3.25 Write Data Command (op code = 01_{16})

The Write Data command causes data to be sent across the IEEE-488 bus with the ATN line deasserted. When the controller is the slave, the write command causes the controller to request a single burst of data from the CPU and then wait to be addressed as a talker by the master controller. The controller then makes another CPU data request and begins transmitting data to the IEEE-488 bus.

C.4 IOCD Execution

IOCDs are executed by the IOP. The command specified within an IOCD is not passed to devices on the bus; rather, the command informs the IOP what operation is to be performed. For example, assume you want to read data from a digital nanovoltmeter on the IEEE-488 bus, and you are in the multitask environment (the necessary IOCDs are built by the interface subroutine). The procedure is as follows:

1. Prepare the bus by transmitting the unlisten multiline message.
2. Command the controller to listen.
3. Command the device, i.e., the nanovoltmeter, to talk.
4. Allow the transfer to take place by deasserting ATN with a command such as RDDA.
5. Prepare the bus once again by transmitting the unlisten and untalk multiline messages.

(ref. 1)

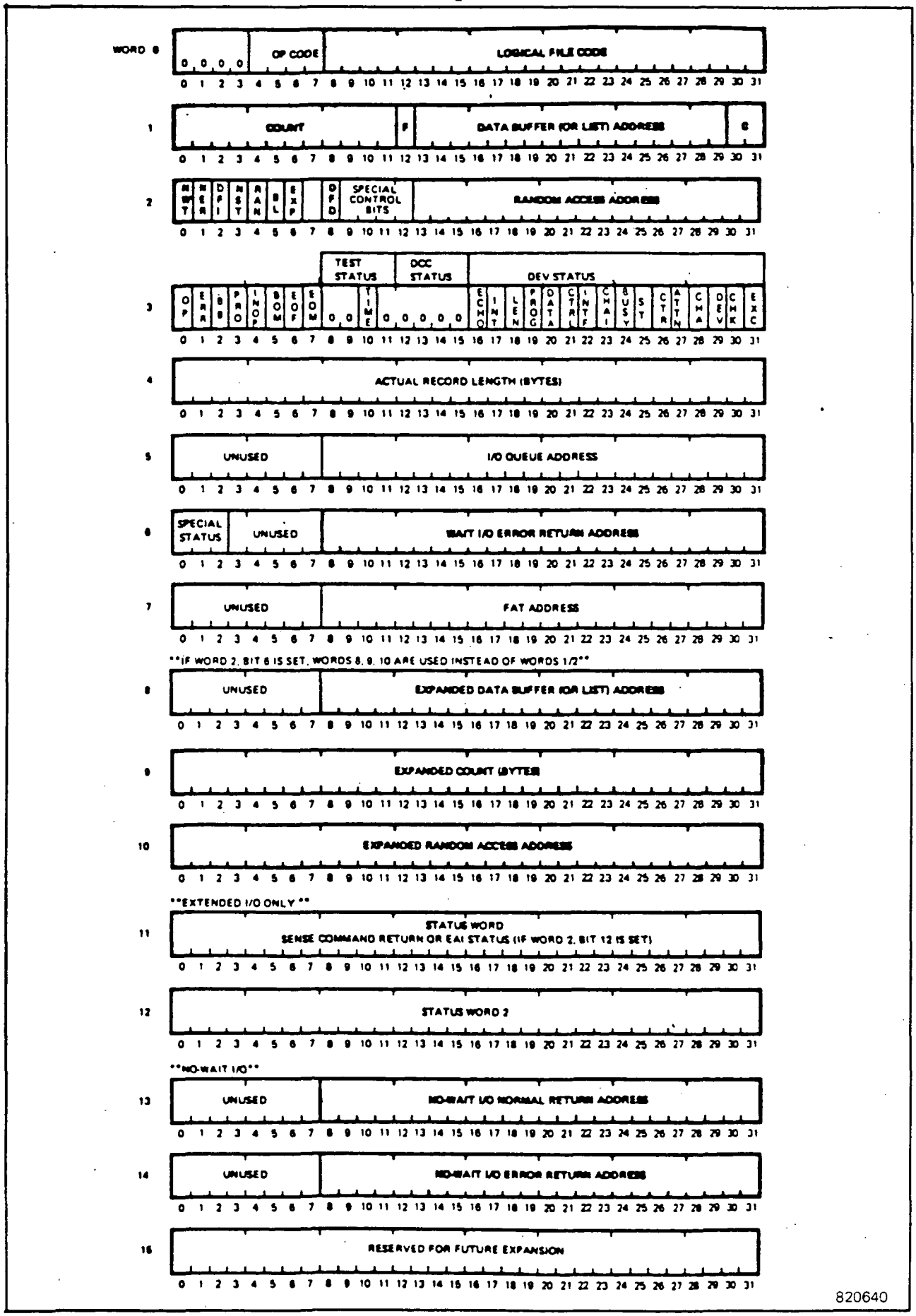


Figure 7-3. File Control Block

(ref. 1)

Word/Bit	Descriptor	Description
		Buffer Address Definitions Set by User
1/12	F	Format: 1=byte, 0=other
1/30,31	C	Code: If Format = 1: byte number. If Format = 0: 00=word, 01=left halfword, 11=right halfword
		Control Flags - Special Format Indicators Set by User
2/0	NWT*	No-Wait I/O (Words 13-14). Else: Wait I/O
2/1	NER	No error return processing. Else: see Section 7.6.1.3.
2/2	DFI*	Data Format Inhibit. Use format in Bits 8-12. Default: IOCS provides formats. See Table 7-7.
2/3	NST	No Handler Status checked. Handler status normally returned in Word 3 will not be returned, however, system status will be returned. Else: see Section 7.6.1.3.
2/4	RAN	Random Access (user supplies address in Bits 13-31). Default: Sequential. IOCS supplies address.
2/5	BL	Blocked I/O, disc/tape only. Else: based on assignment. See Section 7.5.4.
2/6	EXP	Go to Words 8, 9, 10 instead of Words 1 and 2. Default: Words 1 and 2. *For terminal I/O, see also TSM, Section 5.6.4, this volume.
2/7	Reserved.	
2/8	DFD	Device Format Definition. When set, special definitions for 7-track tapes, ALIMs, ADSs, etc. are indicated in bits 9-12. (See Table 7-7.)
		Status Flags Set by Handlers and System
3/0	OP	Operation in Progress. (I/O request has been queued.) Bit is reset after I/O post processing is complete.
3/1	ERR	Error Condition found
3/2	BB	Invalid blocking buffer control pointers encountered in blocking or deblocking
3/3	PRO	Write protect violation
3/4	INOP	Device is inoperable
3/5	BOM	BOM (load point) or illegal volume number (multi-volume) on mag tape
3/6	EOF	End of File (TSM also sets this bit when <CTRL C> typed on terminal)
3/7	EOM	End of Medium (TSM also sets if other than <CR> at bottom of screen.) (End of tape, end of disc file.)

Table 7-3
FCB Bit Interpretations (Page 1 of 2)

		Status for Extended I/O Devices Returned by Handlers
3/10	TIME	Last command exceeded timeout value and was terminated
3/16	ECHO	Echo on Channel
3/17	INT	Post-Task-Controlled interrupt on channel
3/18	LEN	Incorrect record length on channel
3/19	PROG	Channel program check
3/20	DATA	Channel data check
3/21	CTRL	Channel control check
3/22	INTF	Interface check
3/23	CHAI	Channel chaining check
3/24	BUSY	Controller/device busy
3/25	ST	Controller/device status modified
3/26	CTR	Controller end
3/27	ATTN	Attention
3/28	CHA	Channel end
3/29	DEV	Device end
3/30	CHK	Unit check
3/31	EXC	Unit exception
		Status for Non-Extended I/O Devices Returned by Handler
3/8-11	Test Status	Testing Status as received from a 8000 level test device (TD) issued by handler
3/12-15	DCC Status	Controller status (DCC) as received from a 4000 level TD instruction
3/16-31	Device Status	Device status as received from a 2000 level TD instruction. Not applicable for PT, CR, or TY. See Table 7-4.
		Special I/O Status
6/0		No-Wait I/O Normal End Action address not executed.
6/1		No-Wait I/O Error End Action address not executed.
6/2		KILL command. I/O was not issued.

Table 7-3
FCB Bit Interpretations (Page 2 of 2)

7.6.1 FCB Word Descriptions

7.6.1.1 Word 0

IOCS defines the I/O operation indicated by the task (READ, WRITE, etc.) in terms of the operation allowed on an assigned device or file; user defines the logical file code (lfc) used externally to assign a file or device for the operation.

Bits 0-3 This field is always zero.

Bits 4-7 Operation Code - IOCS uses a single hex digit to indicate the type of function for the device handler. Allowable functions and their definitions are unique to each peripheral device. (See Table 7-5.)

Bits 8-31 Logical File Code - Any combination of three ASCII characters is acceptable. Must be supplied by user.

7.6.1.2 Word 1

This word (or Words 8 and 9 if bit 6 of Word 2 is set) supplies a Transfer Control Word (TCW) used to access a data buffer or IOCL for I/O (see below). If no TCW definition is supplied, the transfer buffer defaults to location 0 of the task's logical address space (below the operating system) and is 4096 words (4KW) maximum.

Bits 0-11 Count - Three hex digits specify the number of units (bytes, halfwords, or words) to be transferred to or from a device or file. The count must include a carriage control character, if applicable. The units the count relates to are determined by the data buffer address in bits 12-31. The maximum value of this field is 4096 words. For blocked files, 254 bytes is the maximum transfer. For unblocked files, the maximum value of this field is 4096 words. A zero, negative, or greater than maximum count will default to the maximum transfer count.

or

Number of Data or Command Chain Doublewords - For data/command chaining (e.g., to a GPMC or ADI controller via the Execute Channel service) SVC 1,X'25', bits 0-11 indicate the number of data or command chain doublewords in the I/O Command List (IOCL). The address of the list is provided in bits 12-31.

The F bit (12) and C bits (30 and 31) of the data buffer address are set by IOCS according to the definitions in the following bits.

Bits 12, 30 and 31 — Format Code - These bits specify byte, halfword, or word addressing for data transfers. They are interpreted as follows:

Type of Transfer	F (Bit 12)	CC (Bits 30-31)
Byte	1	xx
Halfword	0	y1
Word –	0	00

where:

xx - Byte number (00, 01, 10 or 11)

y - 0 Left Halfword
1 Right Halfword

00 - Word

If a halfword or word transfer is specified and a device accepts only bytes, IOCS adjusts the count accordingly (x2 or 4). If a byte transfer is specified and a device accepts only halfwords or words, IOCS checks to see that the number of bytes in the buffer is an even multiple of the requested transfer and that the data buffer address is on an acceptable boundary. If these conditions exist, IOCS adjusts count accordingly (times 0, 2, or 4) and initiates the transfer. If the conditions are not met, the request is treated as a specification error. Table 7-6 indicates transfers that are acceptable for various devices. Addresses that will produce specification errors are flagged with an asterisk.

Note that IOCS operations described above enable the user to specify byte transfers beginning on a word boundary or word transfers on any device, whether the device operates on bytes, words, or halfwords.

Doubleword addressing is not allowed; IOCS will abort the task.

Bits 13-29 — Data Buffer Address - Specifies the start address of a data buffer reserved by the user for reads and writes. Bounding requirements are indicated in Table 7-6.

or

Data/Command Chain Address - Specifies the address of an IOCL to use when the Execute Channel service (SVC 1,X'25') is called. The IOCL in turn supplies an IOCD entry describing the transfer count, buffer address and other control information for each command or data transfer to the device.

7.6.1.3 Word 2

Word 2 provides optional control specifications for I/O.

Bits 0-7 Operational Specifications - These eight bits enable the user to specify that special operations such as no-wait I/O be performed by IOCS. The meaning of each bit is as follows:

0 - If this bit is set, IOCS will return to the user immediately after the I/O operation is queued. If the bit is reset, IOCS will exit to the calling program only when the requested operation has been completed.

1 - If this bit is set, error processing will not be performed by either the device handler or IOCS. Normal error processing for disc and magnetic tape is automatic error retry. Error processing for unit record devices except the system console is accomplished by IOCS typing the message "INOP" to the console which allows the operator to retry or abort the I/O operation. If the operator aborts the I/O operation, or if automatic error retry for disc or magnetic tape is unsuccessful, an error status message is typed to the console. An error return address is not applicable if this bit is set, however, the device status will be posted in the FCB (unless bit 3 is set).

2 - When this bit is set, data formatting is inhibited. Otherwise, data formatting is performed by the appropriate device handler. (See Bits 8 through 12 for further explanation.)

3 - If this bit is set, the device handlers perform no status checking and no status information is returned. Hence all I/O will appear to complete without error.

4 - When this bit is set, file accessing will occur in the random mode. Otherwise, sequential accessing will be performed.

5 - If set, a blocked file is specified (disc or tape assignments only).

6 - Expanded FCB present (Words 8-15). This takes advantage of a larger I/O transfer quantity (in bytes), a 24-bit addressing field, and a 32-bit random access address. For Extended I/O operations, up to two interrupt status words are then returned after I/O complete. When this bit is set, IOCS assumes the FCB is 16 words long. The information in Words 8 and 9 is used instead of the data in Word 1. Also, the random access address in Word 10 is used instead of the data in Word 2.

7 - If this bit is set, a task will not be aborted even if an error condition occurs that would cause the task to be aborted.

Bits 8-12 Device Control Specification - These bits contain control specifications unique to certain devices. Device handlers interpret and process the specifications. A bit setting is meaningful only when a particular type of device is assigned as indicated in Table 7-7, columns 2 and 3. (Column 1 indicates default control.)

Bits 13-31 Random Access Address - This field contains a block number (zero origin) relative to the beginning of a disc file, and specifies the base address for read or write operations.

Note: If bit 6 of Word 2 is set, the expanded random access address in Word 10 is used instead of bits 13-31 above.

For High Speed Data (HSD) Interface applications, Word 2 bit meanings are as follows:

Bit 2 For Execute Channel Program, if set, physical IOCL is specified.

Bit 8 Request Device Status After Transfer - This bit indicates an IOCB should be added to the IOCL to retrieve device specific status after the data transfer has completed.

Bit 9 Send Device Command Prior to Data Transfer - This bit indicates an IOCB should prefix the data transfer to transmit a device command word to the device. The value sent is the 32-bit expanded random access address.

Bit 10 Disable Timeout for this Request - This bit indicates the operation will take an indeterminable period of time and the handler should wait an indefinite period of time for the I/O to complete. This generally only has meaning on read operations.

Bit 11 Set UDDCMD from Least Significant Byte of Word 2 - This bit indicates that the UDDCMD byte in the data transfer IOCB should be set to the least significant byte of the random access field of the FCB. This provides the ability to pass additional control information to the device without modifying the device driver.

Bits 24-31 If bit 11 is set, these bits define the UDDCMD field of the generated IOCB, overriding the default value from a handler table.

7.6.1.4 Word 3

Word 3 returns I/O status. IOCS uses 32 indicator bits to return the status, error, and abnormal conditions detected by handlers during the previous or current device operation. The task can examine these bits as needed. Individual bit assignments for bits 0-7 apply to any device. Bits 8-31 mean different things depending on the device. For extended I/O devices, individual bits are shown in the FCB Word 3 area of Figure 7-3 and described in Table 7-3. For non-extended I/O devices, test status, controller (DCC) status, and device status are returned as described in Table 7-4, which accompanies the FCB diagram in Figure 7-3.

For High Speed Data (HSD) Interface applications, Word 3 error status bits have the following meanings:

Bits 17-18	Unused
Bits 19	Record length error
Bit 20	Parity Error
Bit 21	Nonpresent memory (NPM)
Bit 22	Invalid opcode in IOCB Word 0
Bit 23	Device inoperable
Bit 24	Data buffer overflow

7.6.1.5 Words 4 and 5

Word 4 defines record length. This word is used by IOCS to indicate the actual number of bytes transferred during a read or write.

Word 5 defines I/O queue address in bits 8-31. This field is set by IOCS to point to the I/O queue for an I/O request initiated from this FCB.

7.6.1.6 Word 6

In bits 0-7, IOCS returns special status bits as described in Table 7-3.

Word 6 defines a wait I/O error return address in bits 8-31. Specify the address to transfer control to in the case of an unrecoverable error. If this field is not defined, an unrecoverable error is detected, and the user has not set bits 1 and 3 of Word 2 to inhibit error processing, IOCS aborts the task.

7.6.1.7 Word 7

Word 7 must not be written by the user. It defines the File Assignment Table (FAT) entry associated with all I/O performed on behalf of this FCB. The FAT address is supplied by IOCS.

7.6.1.8 Word 8

Word 8 begins expanded TCW definition. This area of the FCB can be used to define transfers larger than 4096 words, e.g., for extended I/O devices.

Bits 8-31	Expanded data buffer address - specifies the start address of a data buffer reserved by the user for reads or writes. This must be a logical byte address with no format bit present. Word bounding is required for some devices if unblocked.
	or
	Expanded Data/Command Chain List Address - Word address that points to the data or command chain list (IOCL) if using Execute Channel service, SVC 1,X'25'.

7.6.1.9 Word 9

Word 9 continues the expanded TCW definition.

Bits 0-31	Expanded transfer count - eight digits specify the number of bytes to be transferred. Note that the transfer count supplied here is always in byte units. Must include the carriage control character, if applicable. A zero or negative count defaults to the maximum allowable count (254 bytes if blocked).

7.6.1.10 Word 10

Word 10 defines expanded random address. This word contains a block number (zero origin relative to the beginning of the disc file). It is the start address for the current read or write operation.

For High Speed Data (HSD) Interface requests in non-Execute Channel Program format, this word defines a device command.

7.6.1.11 Word 11

Word 11 returns status.

Bits 0-31 — Status word 1 - for extended I/O, if an error is detected during an I/O operation, these 32 bits are returned by the SENSE command.

or

Status EAI - for communications adapter (CA) interface, returns external asynchronous interrupt (EIA) status if bit 12 of Word 2 is set.

7.6.1.12 Word 12

Word 12 returns status word 2. This is the second status word returned from extended I/O hardware.

For High Speed Data (HSD) Interface applications, this word contains status sent from the user's device.

7.6.1.13 Word 13

Word 13 defines normal return address for no-wait I/O. In bits 8-31, specifies address to transfer control to when a no-wait I/O operation is complete. The code at this address must be terminated with a call to IOCS for post-processing service (SVC 1,X'2C'). See Section 7.4.2.

For High Speed Data (HSD) Interface applications, this address plus 1 word is the location to which control is transferred on asynchronous notification.

7.6.1.14 Word 14

Word 14 defines an error return address for no-wait I/O. In bits 8-31, specifies (optionally) the address to transfer control to when no-wait I/O completes with an error. The code at this address must be terminated with a call to IOCS for post processing service (SVC 1,X'2C'). See Section 7.4.2.

7.6.1.15 Word 15

Word 15 is reserved for I/O service expansion.

C-2

53A-420 PRESORTING ARINC-429 RECEIVER CARD

DESCRIPTION

The 53A-420 Presorting ARINC-429 Receiver Card is a printed circuit board assembly for use in the 53A System. It is used to receive data from the Mark 33 DITS (Digital Information Transfer System) found on civil aircraft. The 53A System will reformat the data received into 8-bit bytes and retransmit it to the system controller (calculator or computer).

The presorting feature is provided on the Receiver Card to solve the data rate problem imposed on the system controller by the DITS bus. This problem can be stated as follows: the DITS bus can have a data rate of up to 100,000 bits per second, but some labels may appear only once every 1/2 second. Without presorting, to find the most current message of a particular label may require that the system controller sift through a continuous stream of 50,000 bits of data.

With presorting, the DITS data is sorted and stored in RAM located on the Receiver Card, thus allowing immediate access by the system controller to the DITS data associated with a particular label. The presorting can operate in one of two modes:

ALL LABEL Mode - In this mode, the Receiver Card has 256 RAM storage registers, one for each possible DITS word label. The Receiver Card will constantly monitor the incoming data and refresh the appropriate register, therefore, insuring that the most current DITS word, for all labels, is always available to the system controller.

SINGLE LABEL Mode - In this mode, the system controller sets a "trigger label. The Receiver Card then monitors the incoming data for that single trigger label, and when found will store the associated DITS word in RAM. This feature allows the Receiver Card to sequentially store up to 256 DITS words from a single label.

In both of these modes, the system controller can read "on the fly" (read without interrupting the loading of new data) a specified DITS word from RAM. This feature is provided to prevent the RAM from missing a new update while being read by the system controller.

ADDRESS SELECT SWITCH

The Receiver Card has a miniature ten-position switch labeled ADDRESS. The switch is used to select the address (Ø-9) of the Receiver Card. The cover of the switch hinges open to allow access for reselecting the address. A small flat-blade screwdriver should be used to turn the cam-action wiper to the desired position.

OPERATION

The Receiver Card is programmed by 8-bit bytes issued from the system controller (claculator or computer)to the 53A System Communications Card. The Receiver Card is connected to the Communications Card via the 53A-002 Card Cage backplane.

LABELS

The labels, of the messages that will later be input by the system controller, are first output by the system controller to the Receiver Card. Each label is output as a single byte of data.

Label ARINC - 429

1	2	3	4	5	6	7	8
7	6	5	4	3	2	1	Ø
1	Ø	Ø	Ø	Ø	Ø	1	Ø

Octal 222 which is VOR Omnibearing.

Bit 7 is the most significant bit of each byte. The numbers Ø through 7 represent the bits as they are labeled on the CDS Backplane.

DATA

ARINC-429 bits 9 through 32 are input by the system controller from the Receiver Card. The ARINC data is returned as three data bytes followed by Carriage-Return and Line-Feed characters. Three typical data bytes are shown below.

Bits 9 through 32 of the DITS word.

32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9
1	0	1	1	0	0	0	1	0	1	0	0	1	1	1	1	1	1	1	0	0	1	1	1
M			261								117								347				
S																							
B	First Input Byte							Second Input Byte								Third Input Byte							

Note: The interpretation of bit 32 will depend on the mode to which the Receiver Card is programmed (SINGLE or ALL LABEL Mode).
See: Data Errors Page 420-8.

When programming the Receiver Card the ASCII characters used along with the actions initiated are listed below:

ASCII Characters	Action
@XYA	This command is used to program the Receiver Card to the "ALL LABEL Mode" of operation. The @ character is a delimiter used by the 53A system. The two characters following the @ are Mainframe and System Card address re-

spectively. Once a System Card is addressed it will remain addressed until the 53A system detects a new @ character.

The X in the command sequence is the Mainframe address (Ø-9) which has been selected on the Control Card.

The Y in the command sequence is the Receiver Card address (Ø-9) selected by the card's Address Select Switch.

The A in the command sequence puts the Receiver Card in the "ALL LABEL Mode" of operation. In all the following examples it is assumed that the Receiver Card has address 7 and is located in a Mainframe with address Ø.

Example - The command sequence @Ø7A would put the card in the "ALL LABEL Mode."
Status:
Power LED - out.
DDT LED - lit if card is receiving data.
F/S LED - lit if card is set for fast bit rate.
RDE LED - lit if there has been an error.
A/B LED - lit.

ASCII Characters	Action
@XYC	This command will Clear the RDE LED and latch. See: Data Errors in the Overview Section of this manual. The @XY characters are as defined previously. The C in the command sequence will Clear the data error latch.

Example - The command sequence @Ø7C would clear the data error latch.
Status:
Power LED - out.
DDT LED - lit if card is receiving data.
F/S LED - lit if card is set for fast bit rate.
RDE LED - out.
A/B LED - lit if system is in "A" mode.

@XYSb_1

This command is used to set a specific RAM address whose data will be returned the next time the system controller goes to input and requests data.

The @XY characters are as defined previously.

The S in the command sequence is the Set command. The Set command allows the Receiver Card to load the data from the RAM location associated with b_1 into the buffer for input by the system controller.

The RAM locations are numbered from ØØØ to 377 octal, the same as the labels are numbered in ARINC specification 429 Page 10.

b_1 is a single 8-bit binary label. This label is taken from Page 10 of ARINC specification 429. For example, b_1 could be octal 125 which is the label for "Greenwich Mean Time."
Note: octal 125 would actually be output from the system controller as one 8-bit byte.

When in the "ALL LABEL Mode" RAM locations correspond to individual labels.

When in the "SINGLE LABEL Mode" RAM locations correspond to the sequence in which DITS words are loaded into RAM - i.e., ØØØ is the first word loaded, ØØ1 the second, ØØ2 the third, and etc.
See: Data Errors, Page 420-8, for a discussion of the DITS Bit-32 returned to the system controller.

Example - This example assumes the Receiver Card is in the "ALL LABEL Mode." If the byte b_1 represents an octal 23, the command sequence @Ø7Sb_1 would load data associated with the Receiver Card RAM label Ø23 (ADF Frequency) into the card's input buffer.
Status:
Power LED - out.
DDT LED - lit if card is receiving data.
F/S LED - lit if card is set for fast bit rate.
RDE LED - lit if there has been an error.
A/B LED - lit.

After issuing the S command, the system controller will normally request data from the 53A-420 card. The data returned will consist of three 8-bit binary bytes of ARINC-429 data, followed by Carriage-Return and Line-Feed characters. All five bytes must be input by the system controller. The format of the returned ARINC-429 data is described at the beginning of the OPERATIONS Section.