HYBRID COMMUNICATION ROUTINES AND LINKAGE DIAGNOSTIC PROGRAM

4

APPLIED DYNAMICS FOUR

ADI

APPLIED DYNAMICS INTERNATIONAL

3800 STONE SCHOOL ROAD / ANN ARBOR, MICHIGAN 48104 / PHONE 313-973-1300

AD/FOUR

HYBRID USER'S MANUAL

5-FEB-1976

TABLE OF CONTENTS

1.0 INTRODUCTION

The Hybrid Communication Routines (HCR's) provide a hybrid communication path between the AD/FOUR and the PDP-11. The HCR's can provide communication in a single or multi-console (up to four consoles) System. The HCR's have the following characteristics:

> They are written in assembly language, and are USASI Basic Fortran compatible;
>
> Fixed argument lists are used in all calls to HCR's - therefore, all arguments must be present as shown in this manual.
>
> All hybrid data is two's complement and right justified in the word, with the sign bit extended where necessary.

The HCR's provide all the basic AD/FOUR control panel operations, and also provide communication with a completely expanded interface, which includes sense and control lines, DAC's and ADC's, and DMA operations.

The Linkage Diagnostic Program (LKD) consists of a mainline Fortran program and a number of subroutines which are used to test the operation of the hardware in the AD/FOUR and the hybrid interface.

1.1 HYBRID COMMUNICATION ROUTINES

Section 1.1 consists of an introduction to the operation of the HCR's and a description of the Test HCR's, the High-speed HCR's, and the Possibly Useful Programs (PUP's). The descriptions of the individual HCR's are arranged in the following format:

```
ROUTINE or FUNCTION NAME:  (A list of all .GLOBL entry points
                            in the module)

FILE NAME:  (The file name of the module)

CALL SEQUENCE:
(An example call of the routine with the necessary arguments)
DESCRIPTION OF OPERATION:
(The user's description of the operation of the subroutine)
```

1.2 Accessing the HCR's

Figure 1.2 shows a block diagram of the organization of the HCR's in the SYSTEM 511

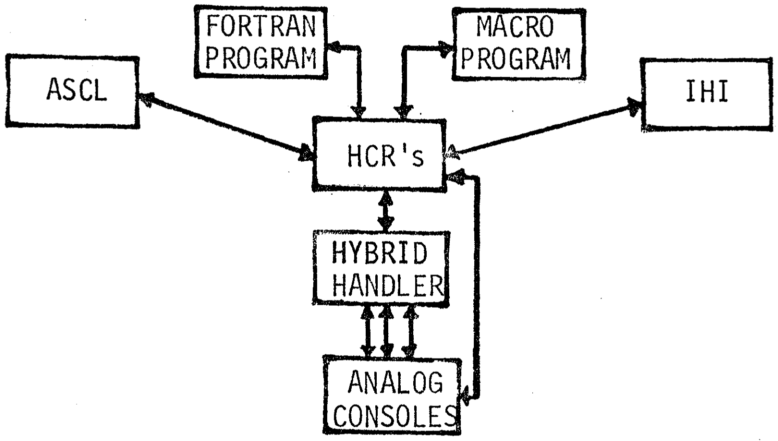


Figure 1.2 Functional Organization of the HCR's in the AD/FOUR HYBRID SYSTEM

The HCR's can be accessed from FORTRAN, MACRO, Interactive Hybrid Interpreter (IHI), and are used internally by the Analog Setup and Checkout Language (ASCL). The HCR's communicate with the analog consoles through the hybrid handler.

In order to be able to use the HCR's from FROTRAN, the user must first call three HCR's in sequence, as shown below:

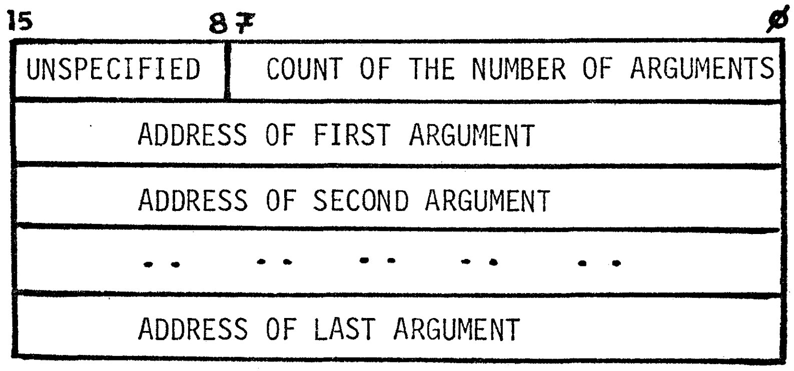
```
CALL STLUN(IERR,ICNS,ILUN)
CALL STEFN(IERR,ICNS,IEFN)
CALL ADATT(IERR,IMSK)
```

In this sequence the user creates a link between the HCR's and the hybrid handler; he first specifies an RSX-11 Logical Unit Number, then specifies a hybrid event flag, and finally attaches his task to the hybrid handler, which allows direct access to the external page addresses of the consoles specified by IMSK. The other HCR's can now be called as desired, and the user should consult the descriptions of the individual HCR's in the further sections of this manual.

In order to call the HCR's from FORTRAN, the user types CALL followed by the ROUTINE NAME and all necessary arguments. HCR functions are called by referencing the FUNCTION NAME and all necessary arguments, as with any standard FORTRAN function subprogram call.

In order to call the HCR's from MACRO, the user must first create a TABLE with the format shown in Figure 2.1. Then, the following operations are performed:

```
MOV   #TABLE,R5
JSR   PC, HCRNAME
```


15
8 7
∅
UNSPECIFIED
COUNT OF THE NUMBER OF ARGUMENTS
ADDRESS OF FIRST ARGUMENT
ADDRESS OF SECOND ARGUMENT
ADDRESS OF LAST ARGUMENT

Fortran Call Site

The majority of the HCR's do not have a console number as one of the arguments. When the user calls INITA or CONSO, he specifies the console with which the HCR's will communicate by default. Any console with which the user expects to communicate must be attached with a CALL ADATT. ADATT can attach one or more consoles; the default is then selected by the user. INITA will select the default console and initialize it; CONSO will select the default console without changing its status. Thus the user can change the default console during the run of his program by CALL CONSO.

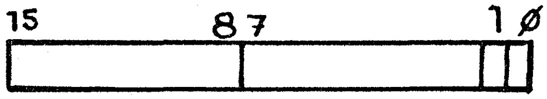
1.3 Error Testing

The Test HCR's can be run in two modes, a TEST mode in which a check is made to see if errors have occured, and RUN mode, in which there is no error detection and the error argument is not accessed, so that all HCR's are as fast and efficient as possible.

In the TEST mode all error detection is made. This mode is used during the problem setup and debugging stages. When an error is detected, the HCR number (the rightmost two digits of the HCR FILE NAME) and a code representing the error type which was encountered are put into the error word. This word can be accessed by the user's program to allow control of program decisions based on the presence or lack of errors. If the user wishes, he can halt his program if an error occurs.

When printing is requested, any error detected in an HCR also causes the error routine to print (on the principal print device) a message to the operator, specifying the HCR name and the type of error.

The HCR HYTST selects the mode of operation for the Test HCR's. See the description of this routine in the section of this manual on Test HCR's. If error testing is not desired (RUN mode) a saving of memory can be achieved by never calling HYTST in the program. The error word has the following format:



BITS ∅-1	ERROR TYPE
BITS 2-7	HCR NUMBER (∅-63)
BITS 7-15	CLASS 3 COEFFICIENT DEVICE ADDRESS (only for HCR numbers 7 and 8, and error type ∅)

The error types are:

ØØ	-	CANNOT SET POT NNNN	- NNNN is the address of the CLASS 3 coefficient device which failed to set to within the current tolerance.
Ø1	-	BUSY	- The analog console is BUSY (i.e., Hybrid access is "off") and cannot be accessed.
1Ø	-	DATA	- The range of an argument is exceeded, or the number of arguments is incorrect.
11	-	ADDR	- The routine has asked for a non-valid address, or the address is incompatible with the routine.

If no errors are detected in the Test mode, the error argument is set to zero. Pups which have an error argument return an HCR number of Ø and cannot cause any printing of error messages (i.e., they always run in TEST mode).

1.4 Values Returned by HCR's

All HCR values are returned in integer format.

A subroutine can return one or many values, which are stored in memory and are accessed by the arguments of the subroutines.

HCR functions return Ø or 1 as the value of the FUNCTION NAME: The functions can be used as variables in IF statements and in arithmetic expressions.

Analog values returned by HCR's are returned in tens of millivolts. 1ØØØ, for example, represents 1Ø volts or Ø.1ØØØ machine unit, and 1ØØØØ is analog reference.

1.5 HCR Arguments

All HCR arguments must be supplied as required for each subroutine or function, and all arguments are in integer format. In cases where the information for an argument is in octal format, the user must convert octal to decimal integer. He can also use the PDP-11 FORTRAN feature which allows him to convert octal to decimal by using a single quote mark before the octal value, i.e., "1Ø will be interpreted as 8 decimal.

2.0 TEST HCR's

TØØØ	HYTST	TØ2Ø	STEP
TØØ1	INITA	TØ21	IC
TØØ2	CONSO	TØ22	HOLD
TØØ3	INMUX	TØ23	OP
TØØ4	HOFF	TØ24	LODE
TØØ5	READ	TØ25	LSTOP/STP
TØØ6	READA	TØ26	LRUN
TØØ7	STIND	TØ27	VER
TØØ8	STINA	TØ28	LEX
TØØ9	SETVS	TØ29	STREF
TØ1Ø	UPDAT	TØ3Ø	ITEST
TØ11	ADHOL	TØ31	ITSTM
TØ12	ADSAM	TØ32	ISTAT
TØ13	SETLI	TØ33	INTR
1Ø14	DACU	TØ34	INTRM
TØ15	DACUR	TØ35	SELVS
TØ16	ADCU	TØ36	STITR
TØ17	ADCUR	TØ37	SELIT
TØ18	TSCAL	TØ38	DFGSU
TØ19	CLRAT	TØ39	BLPKT

Note: in the following descriptions of the test HCR's, the variable IE refers to the error word.

ROUTINE NAME:	HYTST
FILE NAME:	TØØØ
CALL SEQUENCE:	CALL HYTST (MODE)

DESCRIPTION OF OPERATION:

Selects mode of operation for all HCR's.

```
MODE=1: run mode (no testing performed)
MODE=2: test mode, no printing
MODE=3: test mode, with printing
```

A call to HYTST will force a large section of code to be loaded for error printing, and need not be used to select mode 1 only (if modes 2 and 3 will not be used), since this is the initial default.

ROUTINE NAME:	INITA
FILE NAME:	TØØ1
CALL SEQUENCE:	CALL INITA (IE, N)

DESCRIPTION OF OPERATION:

Selects and initializes console N as follows:

1. Power clear command to RIF
2. Clear all DAC update registers
3. Clear all DAC/DCU's
4. Clear all S/H registers
5. Clear all control line registers
6. Read all sense line registers
7. Read all interrupt line registers
8. Clear generalized sense register
9. Clear analog voltage source
10. Clear IRC
11. Set IRA to first MUX address
12. Set scaling to *1

ROUTINE NAME:	CONSO
FILE NAME:	TØØ2
CALL SEQUENCE:	CALL CONSO (IE, N)

DESCRIPTION OF OPERATION:

Allows all other HCR's to communicate with console N.

In a single-console configuration, CONSO is a no-op.

ROUTINE NAME:	INMUX
FILE NAME:	TØØ3
CALL SEQUENCE:	CALL INMUX (IE, N)

DESCRIPTION OF OPERATION:

Sets the IRA to the first MUX address.

N=Ø	No delay
N≠Ø	Explicit delay of 7 msecs.

INMUX is a no-op in an independent ADC control system.

ROUTINE NAME:	HOFF
FILE NAME:	TØØ4
CALL SEQUENCE:	CALL HOFF (IE, N)

DESCRIPTION OF OPERATION:

Enables manual operation of console N.

ROUTINE NAME:	READ
FILE NAME:	TØØ5
CALL SEQUENCE:	CALL READ (IE, IADR, IVLU)

DESCRIPTION OF OPERATION:

Addresses analog device IADR (decimal) and returns its value in IVLU (in tens of millivolts).

Note: in an independent ADC control system, this routine will work only with non-MUX addresses.

ROUTINE NAME: READA

FILE NAME: TØØ6

CALL SEQUENCE: CALL READA (IE, IADR, IVLU, I, J, N)

DESCRIPTION OF OPERATION:

Addresses the N analog devices in the array IADR (I), ..., IADR (I+N-1) and returns their values in the array IVLU (J),..., IVLU (J+N-1).

The arrays IADR and IVLU are assumed to be 1 word integer arrays.

An immediate exit is made from READA at the occurence of the first error, regardless of the value of N.

ROUTINE NAME: STIND

FILE NAME: TØØ7

CALL SEQUENCE: CALL STIND (IE, IADR, ICOF)

DESCRIPTION OF OPERATION:

Sets the coefficient device in IADR (decimal) to the value specified in ICOF.

If IADR specifies a pot address, the value is read back and if it is not within Z$TOLR (default 30 mv), an attempt is made to reset the pot.

ROUTINE NAME: STINA

FILE NAME: TØØ8

CALL SEQUENCE: CALL STINA (IE, IADR, ICOF, I, J, N)

DESCRIPTION OF OPERATION:

Sets the N coefficient devices specified in the array IADR (I),..., IADR (I+N-1) to the values in the array ICOF (J),..., ICOF (J+N-1).

IADR and ICOF are assumed to be one-word integer arrays.

If setting a pot, the value is read back, and if it is not within Z$TOLR (default 30 mv), an attempt is made to reset the pot.

ROUTINE NAME: SETVS

FILE NAME: TØØ9

CALL SEQUENCE: CALL SETVS (IE, IVLU)

DESCRIPTION OF OPERATION:

Sets analog source voltage to the value specified in IVLU.

In systems without an analog voltage source, SETVS may be used to set the AD/4 display.

ROUTINE NAME:	UPDAT
FILE NAME:	TØ1Ø
CALL SEQUENCE:	CALL UPDAT (IE, IB1, IB2, IB3)

DESCRIPTION OF OPERATION:

Sends an update command to all DAC's with a group code corresponding to any of the IB's which are one:

IBn=1	Update DAC's with group code n.

ROUTINE NAME:	ADHOL
FILE NAME:	TØ11
CALL SEQUENCE:	CALL ADHOL (IE, IB1, IB2, IB3)

DESCRIPTION OF OPERATION:

Places the MUX channels with group codes corresponding to any IB's which are one into HOLD mode:

IBn = 1	Place all MUX channels with group code n into HOLD.
IBn = Ø	Leave all MUX channels with group code n in previous mode.

MUX channels with group code Ø cannot be placed in HOLD mode.

If no multiple S/H registers are present, ADHOL is a no-op.

ROUTINE NAME: ADSAM

FILE NAME: TØ12

CALL SEQUENCE: CALL ADSAM (IE, IB1, IB2, IB3)

DESCRIPTION OF OPERATION:

Places the MUX channels with group codes corresponding to any of the IB's which are one into sample mode:

IBn = 1 Place all MUX channels with group code n into sample.

IBn = Ø Leave all MUX channels with group code n in previous mode.

ADSAM is a no-op if no multiple S/H registers are present.

ROUTINE NAME: SETLI

FILE NAME: TØ13

CALL SEQUENCE: CALL SETLI (IE, N, IH, IB)

DESCRIPTION OF OPERATION:

Sets line number IH of control line register N of the current console to IB.

This routine depends on the core image table being properly set up. This can be accomplished by first calling either INITA or SETWD.

ROUTINE NAME: DACU

FILE NAME: TØ14

CALL SEQUENCE: CALL DACU (IE, IADR, IQ)

DESCRIPTION OF OPERATION:

Sets group update resiter of DAC's associated with IADR (decimal) to IQ.

ROUTINE NAME: DACUR

FILE NAME: TØ15

CALL SEQUENCE: CALL DACUR (IE, IADR, IVLU)

DESCRIPTION OF OPERATION:

Returns the value of the DAC group register associated with IADR (decimal) to IVLU.

ROUTINE NAME: ADCU

FILE NAME: TØ16

CALL SEQUENCE: CALL ADCU (IE, IADR, IQ)

DESCRIPTION OF OPERATION:

Sets the S/H register associated with the MUX group corresponding to IADR (decimal) to IQ.

This routine is a no-op if no multiple S/H group registers are present.

ROUTINE NAME: ADCUR

FILE NAME: TØ17

CALL SEQUENCE: CALL ADCUR (IE, IADR, IVLU)

DESCRIPTION OF OPERATION:

Stores in IVLU the value of the S/H group register corresponding to IADR.

This routine is a no-op if no multiple S/H group registers are present.

ROUTINE NAME: TSCAL

FILE NAME: TØ18

CALL SEQUENCE: CALL TSCAL (IE, IQ)

DESCRIPTION OF OPERATION:

Selects the time scale multiplier on the current console according to:

IQ	time scale
Ø	* 1
1	* 1Ø
2	* 1ØØ
3	* 1ØØØ

ROUTINE NAME: CLRAT

FILE NAME: TØ19

CALL SEQUENCE: CALL CLRAT (IE, IT)

DESCRIPTION OF OPERATION:

Selects the clock output of the current console according to:

IT	clock output
Ø	1 MHz
1	Digital selected rate
2	External rate

ROUTINE NAME: STEP

FILE NAME: TØ2Ø

CALL SEQUENCE: CALL STEP (IE)

DESCRIPTION OF OPERATION:

Generates a logic step pulse on the current console.

ROUTINE NAME: IC

FILE NAME: TØ21

CALL SEQUENCE: CALL IC (IE)

DESCRIPTION OF OPERATION:

Selects analog mode IC on the current console.

ROUTINE NAME: HOLD

FILE NAME: TØ22

CALL SEQUENCE: CALL HOLD (IE)

DESCRIPTION OF OPERATION:

Selects analog mode HOLD on the current console.

ROUTINE NAME: OP

FILE NAME: TØ23

CALL SEQUENCE: CALL OP (IE)

DESCRIPTION OF OPERATION:

Selects analog mode OP on the current console.

ROUTINE NAME:	LODE
FILE NAME:	TØ24
CALL SEQUENCE:	CALL LODE (IE)

DESCRIPTION OF OPERATION:

Selects analog mode LOAD on the current console.

ROUTINE NAMES:	LSTOP, STP
FILE NAME:	TØ25
CALL SEQUENCE:	CALL LSTOP (IE)

DESCRIPTION OF OPERATION:

Selects logic mode STOP on current console.

ROUTINE NAME:	LRUN
FILE NAME:	TØ26
CALL SEQUENCE:	CALL LRUN (IE)

DESCRIPTION OF OPERATION:

Selects logic mode RUN on current console.

ROUTINE NAME:	VER
FILE NAME:	TØ27
CALL SEQUENCE:	CALL VER (IE,IB)

DESCRIPTION OF OPERATION:

Selects problem verify according to:

IB	mode
Ø	OFF
1	ON

ROUTINE NAME: LEX

FILE NAME: TØ28

CALL SEQUENCE: CALL LEX (IE, IB)

DESCRIPTION OF OPERATION:

Selects logic execute on/off on the current console according to:

IB	mode
Ø	OFF
1	ON

ROUTINE NAME: STREF

FILE NAME: TØ29

CALL SEQUENCE: CALL STREF (IE, IB)

DESCRIPTION OF OPERATION:

Selects COEF*IN on/off mode on the current console according to:

IB	mode
Ø	OFF
1	ON

ROUTINE NAME: ITEST

FILE NAME: TØ3Ø

CALL SEQUENCE: IVLU = ITEST (IE, N, IH)

DESCRIPTION OF OPERATION:

Sets IVLU to the value of line IH of Sense Register N of the current console.

ROUTINE NAME: ITSTM

FILE NAME: TØ31

CALL SEQUENCE: IVLU = ITSTM (IE, N, IM)

DESCRIPTION OF OPERATION:

Sets IVLU to Ø or 1 according to whether all the mask bits in IM are off in Sense Line register N on the current console or not.

ROUTINE NAME: ISTAT

FILE NAME: TØ32

CALL SEQUENCE: IVLU = ISTAT (IE, IH)

DESCRIPTION OF OPERATION:

Sets IVLU to the value of HIF status bit IH on the current console.

FUNCTION NAME: INTR

FILE NAME: TØ33

CALL SEQUENCES: IVLU = INTR (IE, N, M)

DESCRIPTION OF OPERATION:

Sets IVLU to the value of line M of Interrupt Line register N of the current console.

FUNCTION NAME: INTRM

FILE NAME: TØ34

CALL SEQUENCE: IVLU = INTRM (IE, N,IM)

DESCRIPTION OF OPERATION:

Sets IVLU to Ø or 1 according to whether all the bits in IM are off/on in Interrupt Line register N of the current console.

ROUTINE NAME: SELVS

FILE NAME: TØ35

CALL SEQUENCE: CALL SELVS (IE, IQ)

DESCRIPTION OF OPERATION:

Selects the V-signals to the Hybrid Interval timer of the current console according to;

IQ	V-signals
Ø	1S
1	.1S
2	.Ø1S
3	.ØØ1S

ROUTINE NAME: STITR

FILE NAME: TØ36

CALL SEQUENCE: CALL STITR (IE, IC1, IC2, IC3)

DESCRIPTION OF OPERATION:

Sets the hybrid interval timer A, B, and C periods to IC1, IC2, and IC3, respectively (on the current console).

ROUTINE NAME: SELIT

FILE NAME: TØ37

CALL SEQUENCE: CALL SELIT (IE, IB)

DESCRIPTION OF OPERATION:

Selects the mode of the hybrid interval timer on the current console according to:

<u>IB</u>	<u>mode</u>
Ø	thumbwheels and current V-signal rate
1	hybrid periods and hybrid selected V-signal rate.

ROUTINE NAME: DFGSU

FILE NAME: TØ38

CALL SEQUENCE: CALL DFGSU (IE, IARY, IT, ID)

DESCRIPTION OF OPERATION:

Transmits the 21 voltage values in the array IARY to DFG table IT and performs a closed-loop set into DSDFG channel ID. The values are in the order of independent variable breakpoints at Ø.Ø, Ø.1, ---,1.Ø, -Ø.1,---,-1.Ø.

ROUTINE NAME: BLPKT

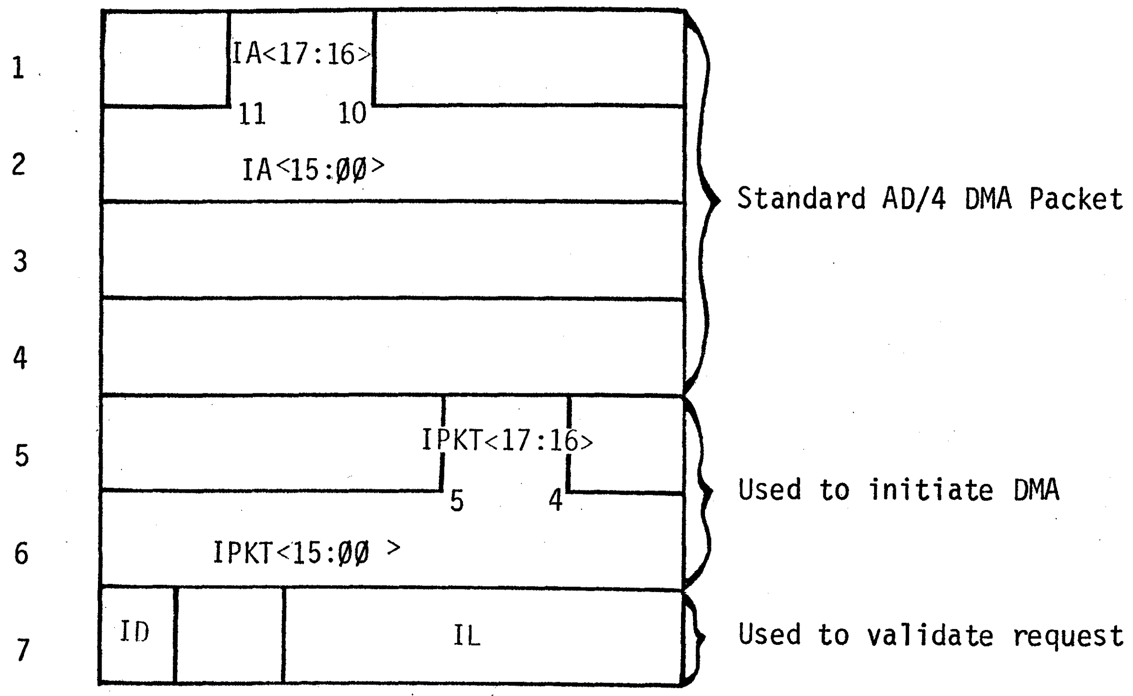
FILE NAME: TØ39

CALL SEQUENCE: CALL BLPKT (IE, IPKT, IA, IL, ID)

DESCRIPTION OF OPERATION:

Sets up the 7-word packet IPKT so that DMA can be performed using memory address IA, word count IC, and direction (Ø=read, 1=write) ID. The packet actually contains:

IPKT (I)



2.1 HIGH SPEED HCR's

FILE	SYMBOL
HØ64	SETIA
HØ65	SENIA
HØ66	SETIC
HØ67	SENIC
HØ68	SETWD
HØ69	SENSW
HØ70	INTRW
HØ71	STATW
HØ72	IERRA
HØ73	IBUSY
HØ74	IOVLD
HØ75	STSIN/STINH
HØ76	RDSIN/READH
HØ77	STARY/STBLK
HØ78	RDARY
HØ79	STSEQ
HØ80	RDSEQ/SCANH
HØ81	BEGST
HØ82	BEGRD
HØ83	IOBZY
HØ84	ADCSH
HØ85	DFGRD
HØ86	DFGWT
HØ87	DFGSI

ROUTINE NAME:	SETIA
FILE NAME:	HØ64
CALL SEQUENCE:	CALL SETIA (IVAL)

DESCRIPTION OF OPERATION:

Stores the contents of IVAL in the IRA of the current console.

ROUTINE NAME:	SENIA
FILE NAME:	HØ65
CALL SEQUENCE:	CALL SENIA (IVLU)

DESCRIPTION OF OPERATION:

Returns value of the IRA of the current console to IVLU.

ROUTINE NAME:	SETIC
FILE NAME:	HØ66
CALL SEQUENCE:	CALL SETIC (IVLU)

DESCRIPTION OF OPERATION:

Sets the IRC of the current console to the value of IVLU.

ROUTINE NAME: SENIC

FILE NAME: HØ67

CALL SEQUENCE: CALL SENIC (IVLU)

DESCRIPTION OF OPERATION:

Returns value of the IRC of the current console to IVLU.

ROUTINE NAME: SETWD

FILE NAME: HØ68

CALL SEQUENCE: CALL SETWD (N, I)

DESCRIPTION OF OPERATION:

Sets control line register N of the current console to the value of I.

ROUTINE NAME: SENSW

FILE NAME: HØ69

CALL SEQUENCE: CALL SENSW (N, IVLU)

DESCRIPTION OF OPERATION:

Returns the value of sense line N on the current console to IVLU.

ROUTINE NAME: INTRW

FILE NAME: HØ7Ø

CALL SEQUENCE: CALL INTRW (N, IVLU)

DESCRIPTION OF OPERATION:

Returns the value of interrupt line register N of the current console to IVLU.

ROUTINE NAME: STATW

FILE NAME: HØ71

CALL SEQUENCE: CALL STATW (IVLU)

DESCRIPTION OF OPERATION:

Returns the value of the HIF status register in the current console to IVLU.

FUNCTION NAME: IERRA

FILE NAME: HØ72

CALL SEQUENCE: IVLU = IERRA (IDUMMY)

DESCRIPTION OF OPERATION:

Stores the value of the HIF status bit Ø of the current console in IVLU.

FUNCTION NAME: IBUSY

FILE NAME: HØ73

CALL SEQUENCE: IVLU = IBUSY (IDUMMY)

DESCRIPTION OF OPERATION:

IVLU receives the value of HIF status bit 1Ø of the current console.

FUNCTION NAME: IOVLD

FILE NAME: HØ74

CALL SEQUENCE: IVLU = IOVLD (IDUMMY)

DESCRIPTION OF OPERATION:

Returns the value of HIF status bit 2 of the current console to IVLU.

ROUTINE NAME:	STSIN, STINH
FILE NAME:	HØ75
CALL SEQUENCE:	CALL STSIN (JADR, ICOF)

DESCRIPTION OF OPERATION:

Sets the DAC/DCU on the current console specified by JADR (octal) to the value in ICOF.

ROUTINE NAME:	RDSIN, READH
FILE NAME:	HØ76
CALL SEQUENCE:	CALL RDSIN (ICH, IVLU)

DESCRIPTION OF OPERATION:

Returns the value of ADC-MUX address ICH (octal)on the current console to IVLU.

ROUTINE NAME: STBLK, STARY

FILE NAME: HØ77

CALL SEQUENCE: CALL STARY (JADR, ICH, N)

DESCRIPTION OF OPERATION:

Sets the DAC/DCU's on the current console specified by the array JADR (1), JADR (2),..., JADR (N) (in octal) to the values in the array ICH (1), ICH (2),..., ICH (N).

ROUTINE NAME: RDARY

FILE NAME: HØ78

CALL SEQUENCE: CALL RDARY (ICH, IVLU, N)

DESCRIPTION OF OPERATION:

Returns the values of the MUX addresses (on the current console) specified by the array ICH (1), ICH (2),..., ICH (N) (in octal) to the array IVLU (1), IVLU (2),..., IVLU (N).

ROUTINE NAME: STSEQ

FILE NAME: HØ79

CALL SEQUENCE: CALL STSEQ (JADR, ICOF, N)

DESCRIPTION OF OPERATION:

Sets the DAC/DCU's (on the current console) specified by the array JADR, JADR+1,..., JADR+N-1 (in octal) to the values ICOF (1), ICOF (2), ..., ICOF (N).

ROUTINE NAME: RDSEQ, SCANH

FILE NAME: HØ8Ø

CALL SEQUENCE: CALL RDSEQ (IADR, IVLU, N)

DESCRIPTION OF OPERATION:

Returns to the array IVLU (1),..., IVLU (N) the values of MUX channels J, J+1,..., J+N-1, where J is the channel number associated with patchboard address IADR (in octal) on the current console.

ROUTINE NAME:	BEGST
FILE NAME:	HØ81
CALL SEQUENCE:	CALL BEGST (JADR, IPKT, N)

DESCRIPTION OF OPERATION:

Initiates under DMA the task equivalent to STSEQ (HØ79). The DMA packet in IPKT should not be modified until IOBZY (Ø) returns Ø. Neither should communications with the HIF be attempted. See the descriptions of HCR's HØ79 and TØ39 for further details.

ROUTINE NAME:	BEGRD
FILE NAME:	HØ82
CALL SEQUENCE:	CALL BEGRD (ICH, IPKT, N)

DESCRIPTION OF OPERATION:

Initiates under DMA, a task equivalent to RDSEQ (HØ8Ø) except a DMA packet (from a previous call to BLPKT) is given instead of the value array.

The user should not attempt to use the data in the value array (in IPKT), nor communicate with the HIF until IOBZY (Ø) returns a Ø.

See the description of HCR's TØ39 and HØ8Ø for further details.

FUNCTION NAME: IOBZY

FILE NAME: HØ83

CALL SEQUENCE: IVLU = IOBZY (IDEV)

DESCRIPTION OF OPERATION:

IVLU is set to Ø (1) if a DMA operation is not (is) in progress. In an independent controller system, the device tested is specified by:

IDEV	Device
Ø	RIF
1	ADC

In an non-independent ADC system, the RIF is always tested and IDEV ignored.

ROUTINE NAME: ADCSH

FILE NAME: HØ84

CALL SEQUENCE: CALL ADCSH (1)

DESCRIPTION OF OPERATION:

The contents of I are sent to the S/H register of the current console.

ROUTINE NAME: DFGRD

FILE NAME: HØ85

CALL SEQUENCE: CALL DFGRD (IVLU, ITABLE)

DESCRIPTION OF OPERATION:

Places the 21 DAC settings in the DFG table ITABLE of the current console into the array IVLU.

ROUTINE NAME: DFGWT

FILE NAME: HØ86

CALL SEQUENCE: CALL DFGWT (IVLU, ITABLE)

DESCRIPTION OF OPERATION:

Sends the 21 DAC settings found in the array IVLU into the DFG table ITABLE of the current console.

ROUTINE NAME: DFGSI

FILE NAME: HØ87

CALL SEQUENCE: CALL DFGSI (ITABLE, IDFG)

DESCRIPTION OF OPERATION:

Causes the DAC values stored in DFG table ITABLE to be "immediate set" into the DFG specified by IDFG on the current console.

2.3 POSSIBLY USEFUL PROGRAMS (PUP's)

FILE	SYMBOLS
PØØ1	WATE
PØØ2	ADDR
PØØ3	DOCAD
PØØ4	WTOP
PØØ5	RDOP
PØØ6	CNNCT
PØØ7	DMAST
PØØ8	MOVE
PØØ9	ISW
PØ1Ø	DSPLY
PØ11	HYLOG
PØ12	HYTOL
PØ13	DSINH
PØ14	ENINH
PØ15	STLUN
PØ16	STEFN
PØ17	ADATT
PØ18	ADDET

ROUTINE NAME: WATE

FILE NAME: PØØ1

CALL SEQUENCE: CALL WATE (I)

DESCRIPTION OF OPERATION:

Causes a delay of approximately 1ØØ*I micro-seconds.

ROUTINE NAME: ADDR

FILE NAME: PØØ2

CALL SEQUENCE: CALL ADDR (IA, IAB)

DESCRIPTION OF OPERATION:

Returns to IAB the octal equivalent of a decimal component address in IA.

```
    IA  =  a*1ØØØ  + b*1ØØ  +c *1Ø  + d is converted to
    IAB =  a*512   + b*64   +c*8    + d
```

ROUTINE NAME: DOCAD

FILE NAME: PØØ3

CALL SEQUENCE: CALL DOCAD (IAB,IA)

DESCRIPTION OF OPERATION:

Converts an address from octal to decimal.

```
    IAB = a*512  + b*64   + c*8  + d is converted to
    IA  = a*1ØØØ + b*1ØØ  + c*1Ø + d
```

DOCAD is the inverse function of ADDR.

ROUTINE NAME: WTOP

FILE NAME: PØØ4

CALL SEQUENCE: CALL WTOP (I, M)

DESCRIPTION OF OPERATION:

Sends op code I and data M to current RIF.

ROUTINE NAME: RDOP

FILE NAME: PØØ5

CALL SEQUENCE: CALL RDOP (I, M)

DESCRIPTION OF OPERATION:

Sends op code I to current RIF and returns the data read back to M.

ROUTINE NAME: CNNCT

FILE NAME: PØØ6

CALL SEQUENCE: CALL CNNCT (RTINE, MASK)

DESCRIPTION OF OPERATION:

Connects AST's from the current console to routine RTINE and transmits MASK to the RIF CSR. Thereafter subroutine RTINE is called whenever an AST occurs. The call site is:

```
    SUBROUTINE RTINE ~~(IARY)~~
    ~~DIMENSION  IARY (14)~~
```

with

```
    IA (1) = RIF interrupt index
    IA (2) = RIF CSR
    IA (3) = Interrupted console
    IA (4) = Interrupting console
    IA (6) - IA(12) = saved RØ-R7
    IA (13) = saved PSW
    IA (14) = saved DSW
```

ROUTINE NAME: DMAST

FILE NAME: PØØ7

CALL SEQUENCE: CALL DMAST (IPKT)

DESCRIPTION OF OPERATION:

Starts the DMA with the packet specified by IPKT. IPKT should have previously been built by a call to BLPKT (test HCR # 39).

ROUTINE NAME: MOVE

FILE NAME: PØØ8

CALL SEQUENCE: CALL MOVE (I, J)

DESCRIPTION OF OPERATION:

Stores the virtual address of variable I in J.

FUNCTION NAME: ISW

FILE NAME: PØØ9

CALL SEQUENCE: IVLU = ISW (IH)

DESCRIPTION OF OPERATION:

Stores the value of the PDP-11 console switch number IH in IVLU.

ROUTINE NAME:	DSPLY
FILE NAME:	PØ1Ø
CALL SEQUENCE:	CALL DSPLY (I)

DESCRIPTION OF OPERATION:

Loads the PDP-11 display register with the contents of I.

*This routine requires modification of the RSX-11D and RSX-11M executives to function properly (it is, therefore, usually a no-op).

ROUTINE NAME:	HYLOG
FILE NAME:	PØ11
CALL SEQUENCE:	CALL HYLOG

DESCRIPTION OF OPERATION:

No-op on RSX system.

ROUTINE NAME: HYTOL

FILE NAME: PØ12

CALL SEQUENCE: CALL HYTOL (ITOL)

DESCRIPTION OF OPERATION:

Sets the tolerance for pot-setting errors to ITOL (expressed in tens of millivolts).

ROUTINE NAME: DSINH

FILE NAME: PØ13

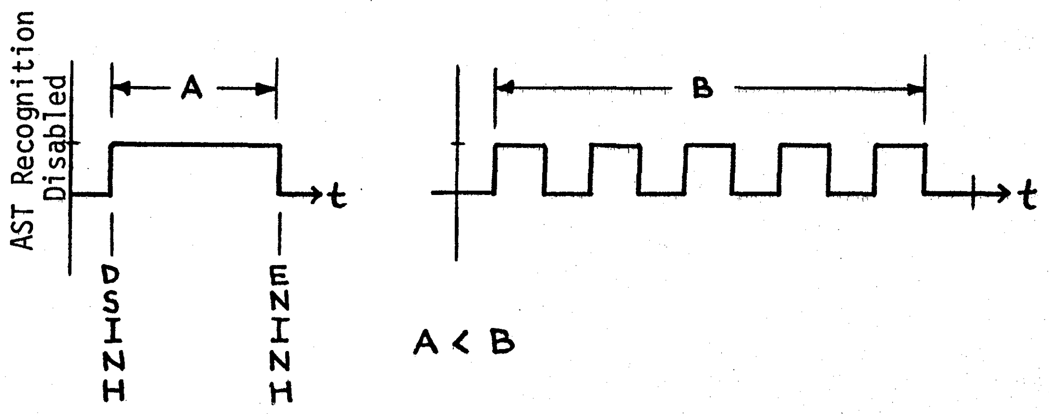
CALL SEQUENCE: CALL DSINH

DESCRIPTION OF OPERATION:

Disables AST recognition, and disables the "per HCR" inhibition and re-enabling of AST recognition. Calling DSINH and ENINH (see PØ14) around a set of HCR calls may save considerable overhead within the HCR's.

Example:

```
CALL DSINH
CALL HCR          CALL HCR
    :                 :
CALL HCR     vs.      :
CALL ENINH        CALL HCR
```



ROUTINE NAME: ENINH

FILE NAME: PØ14

CALL SEQUENCE: CALL ENINH

DESCRIPTION OF OPERATION:

Negates the effect of DSINH (see PØ13)

ROUTINE NAME: STLUN

FILE NAME: PØ15

CALL SEQUENCE: CALL STLUN (IE, ICNS, ILUN)

DESCRIPTION OF OPERATION:

Specifies RSX Logical Unit Number ILUN for use in communication with console ICNS.

ROUTINE NAME: STEFN

FILE NAME: PØ16

CALL SEQUENCE: CALL STEFN (IE, ICNS, IEFN)

DESCRIPTION OF OPERATION:

Specifies event flag number IEFN for use in communication with console ICNS.

ROUTINE NAME: ADATT

FILE NAME: PØ17

CALL SEQUENCE: CALL ADATT (IE, MASK)

DESCRIPTION OF OPERATION:

Attaches the console (s) specified by MASK. Mask bits are associated with analog consoles by the System Manager at SYSGEN time.

ROUTINE NAME: ADDET

FILE NAME: PØ18

CALL SEQUENCE: CALL ADDET (IE)

DESCRIPTION OF OPERATION:

Detaches all consoles currently attached.

3.0 INTRODUCTION

The Linkage Diagnostic Program (LKD) consists of a mainline Fortran program and a number of independent diagnostic and support subroutines which enable the user to check the hardware operation of the AD/FOUR and the system HIF/RIF. LKD engages the user in a console dialog which has been made to be as easy to understand and as self-explanatory as possible. Error messages from LKD are also self-explanatory.

LKD is built with the task name "...LKD" and is run as an ordinary RSX-11 system program by typing:

```
                MCR > LKD <CR>

          or    MCR > RUN ...LKD
```

The space between RUN and ... is mandatory. LKD will then identify itself, and the mainline program will automatically call LKD ØØ in order to allow the user to initialize the program as he chooses. After the user has responded to the choices offered by LKD ØØ, the mainline program requests the user to enter the number of an LKD subroutine. The user may then enter any valid LKD subroutine number and the subroutine will be run. The valid subroutine numbers consist of the LKD subroutines discussed in this chapter, along with any subroutines the user may have written himself. If the user enters an invalid positive LKD subroutine number, the mainline program will simply prompt him for another number. If he enters a negative subroutine number, the program will be re-initialized, as if it had just been loaded. LKD 99 is reserved to make a rational exit back to MCR.

LKD ØØ allows the user to pause after an error has been detected by an LKD subroutine. This pause control is discussed in the explanation of LKD ØØ. If the user does not elect to specify pause control "on" in LKD ØØ, he may still pause the program while it is running by use of the PDP-11 switches Ø - 3. This feature is also discussed in the explanation of LKD ØØ.

The user may enter individual LKD subroutine numbers in order to run the subroutines one at a time, or he may call LKD 98 in order to run the subroutines automatically. LKD 98 will request the user to enter a cycle count which will determine how many cycles will be run of those tests (such as LKD Ø4) which require a cycle count, and then will run LKD Ø1 - LKD 11 in succession.

The called routine types its number and name. When an error is encountered, an error message is typed, giving error type, address, register number, etc. If pause mode was selected, the operator types 0 or 1 to continue, 2 to return to the beginning of the current routine, or 3 to exit from the current routine. (Note: A <CR> is interpreted as a Ø response).

If pause mode is not selected, the console switches can be used to control a routine in much the same manner. Setting switch Ø has the same effect as an error and switches 1, 2, and 3 are interpreted the same as the corresponding operator response. The operator should first select switch 1, 2, or 3 and then set switch Ø to enable recognition of the request.

Some LKD subroutines require the operator to fulfill certain patching instructions before the LKD can complete its tests. These are discussed in the descriptions of the individual LKD's in this chapter.

If an LKD subroutine encounters a hardware malfunction, an error message is printed to inform the user of the malfunction. The tolerance for tests of the analog hardware is 100 millivolts initially; this tolerance may be changed by requesting LKD 97. If an analog device fails to perform its function within the current tolerance, an error message will be printed. The error messages are self-explanatory, and LKD ØØ allows the user to pause in the program after any errors are encountered if he so desires. Some errors may be encountered in LKD ØØ which will return the user to MCR. LKD may then be run again as described above.

Bit patterns used in the tests are formed by moving a "one" through a field of "zeros" and a "zero" through a field of "ones". The field length is equal to the particular register being tested.

The LKD Diagnostic Subroutines and the LKD Support Subroutines are discussed individually in the rest of this manual, along with descriptions of error messages and patching instructions where these are useful.

3.1 LKD Diagnostic Subroutines (LKDØØ - LKD89)

LKDØØ INITIALIZATION

DESCRIPTION OF OPERATION:

LKDØØ allows the user to change the console number with which the subroutines will communicate, allows the user to select the error log device, and allows the user to select pause control. The console dialog explains the various choices to the user.

The pause control, if selected by the user, will cause the program to pause after an error occurs, and user intervention is then necessary to cause the program to continue, restart, or abort the routine.

If the operator does not select pause control via LKD ØØ, he may still use the PDP-11 switches Ø - 3 to effect control of the program in case an error occurs. The switches have the following effects:

SWITCH	Meaning when asserted (Logic 1):
Ø	Enables recognition of the status of switches 1-3.
1	Restarts the current routine.
2	Aborts the current routine.
3	Aborts an automatic run which was initiated via LKD 98.

The error messages for LKD ØØ are self-expanatory.

LKD1 - STATUS BIT TEST

By causing appropriate error and non-error conditions. LKD1 tests all bits of the status bus, except bits 4, 6, 8, and 11.

ERROR MESSAGES

ERROR - BIT nn IS s

where nn is the bit number (0-15)

s is its status (0 or 1)

NOTE: This routine will pause regardless of pause action decided by the operator.

(Refer to Section 3.0)

LKD2 - IRA TEST

This routine requests "ENTER 14 TEST CYCLES." The operator should enter the number of times the test will be performed. The register is tested by writing a series of data words into the address register, reading them back, and comparing the results with the data sent.

ERROR MESSAGES

```
ERROR
        SENT                    REC'D
xxxxxxxxxxxxxxxx      yyyyyyyyyyyyyyyy
```

where xxxxxxxxxxxxxxxx is the bit pattern sent

yyyyyyyyyyyyyyyy is the bit pattern received

LKD3 - IRC TEST

This test operates in the same manner as LKD2, except the control register is tested.

LKD4 - CONTROL, SENSE, INTERRUPT REGISTER TEST

The first time LKD4 is called, the total number of each register in the console and "PATCH ACCORDING TO MANUAL" are printed out. Patching instructions are:

Control register lines 0 - 15 to sense register lines 0 - 15

Control register lines 8 - 15 to interrupt register lines 0 - 7.

Patch control register 0 to sense register 0 to interrupt register 0;

Control register 1 to sense register 1 to interrupt register 1;

The routine then requests "ENTER(14) TEST CYCLES". The operator should enter the number of times the test will be performed. The registers are tested by sending a series of data words to the control register, reading the sense and interrupt registers and comparing with the data sent.

ERROR MESSAGES (PRECEDED BY IDENTIFICATION OF REGISTER BEING TESTED)

```
ERROR
          SENT                     REC'D
     xxxxxxxxxxxxxxxx       yyyyyyyyyyyyyyyy
```

xxxxxxxxxxxxxxxx is the bit pattern sent

yyyyyyyyyyyyyyyy is the bit pattern received

LKD5 - DAC UPDATE REGISTER TEST

LKD5 uses a subroutine to find the addresses of all valid DACs. If there are none, a message is printed out and the routine is skipped. Each DAC is tested by writing each update code (0-3) into the update register, reading it back, and comparing with the code sent.

ERROR MESSAGES

```
ERROR - REGISTER ADDR.2aaa SENT nn RECVD nn
Where 2 is the component class
      aaa is the address
       nn are the values sent and received.
```

LKD6 - ADC SAMPLE/HOLD REGISTER TEST

If there are no S/H registers, a message is printed out and the routine is skipped. Each S/H register is tested by writing each S/H code (0-3) into the register, reading it back, and comparing with the original.

ERROR MESSAGES

```
ERROR  -  REGISTER ADDR. 6aaa SENT nn RECVD nn
where 6 is the component class
      aaa is the address
      nn are the values sent and received.
```

LKD7 - DAC/DCU TEST

LKD7 uses a subroutine to find the addresses of all valid DAC's and DCU's. If there are no DAC's and/or DCU's a message is printed out and the appropriate part of the test is skipped. The first time LKD7 is called, a table of valid DAC addresses and "PATCH ACCORDING TO MANUAL are typed. Patching instructions are:

> Patch plus and minus reference to all multiplying DAC's in the valid address table. Each DAC is tested by writing a series of values into it, reading them back, and comparing the results with the values sent. Each DCU is tested in the same manner.

ERROR MESSAGES

```
ADDR.2aaa - ERROR
    SENT      ERR
nnnnn      eeeee
```

where 2 is the component class

aaa is the DAC or DCU address

±nnnnn is the value sent

±eeeee is the amount of the error

LKD8 - DAC UPDATE TEST

LKD8 uses a subroutine to find all valid DAC addresses. Each DAC is tested with each update code before and after the update command is sent. The DAC passes the test if it updates only when its update code is zero or when its update command is sent.

ERROR MESSAGES

ERROR - DAC 2 aaa CODE c UPDATE $u_1u_2u_3$ SENT nnnnn REC'd nnnnn

where 2 is the component class

aaa is the DAC address

c is the update code sent

$u_1u_2u_3$ are the update commands sent

u_i = 1 Group i should have updated

u_i = 0 Group i should not have updated

u_i = 8 No update command was sent

±nnnnn are the values sent and received

LKD9 - SERVO SET POT TEST

LKD9 uses a subroutine to find valid servo pot addresses. If there are no servo pots, a message is printed out and the routine is skipped. The first time LKD9 is called, a table of valid pot address is printed out; "ENTER (4(I4,1X)) FOUR POT ADDRESSES" is then printed out and the operator should enter four pot addresses from the table, with no embedded blanks or other characters. These four pots are tested by sending values to each, reading them back and comparing with the values sent.

ERROR MESSAGES

ERROR - POT 3aaa SENT nnnnn RECVD nnnnn

where 3 is the component class

aaa is the pot address

nnnnn are the values sent and received.

LKD10 - ADC SYSTEM TEST

LKD10 uses a subroutine to find the first valid DAC or DCU, then finds four valid amplifiers and types "PATCH ACCORDING TO MANUAL" and the DAC or DCU and AMPLIFIER addresses. Patching should be according to the following diagram.

NOTE: DAC wiring will be complete if the test has been performed in sequence. Each ADC channel is tested by sending a series of values to the DAC or DCU reading the DAC or DCU amplifiers and ADC channels and comparing values. A DCU is used only if there are no DAC's in the console being tested.

ERROR MESSAGES

ERROR - DAC or DCU 2aaa SENT nnnnn ERR eeeee

ERROR - AMP 0aaa SENT nnnnn ERR eeeee

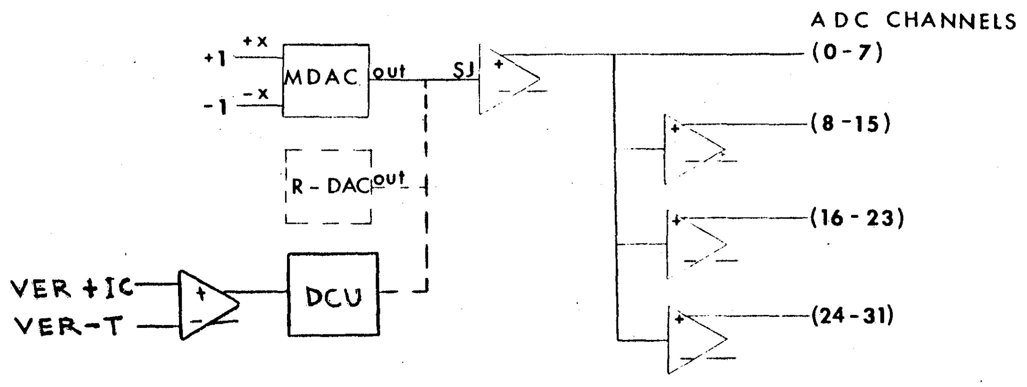
ERROR - ADC 6aaa SENT nnnnn ERR eeeee

where N(2, Ø, 6) is component type

aaa is the address

±nnnnn is the value sent

±eeeee is the amount of the error



LKD11 - SAMPLE/HOLD TEST

If LKD5 has not been run, a message is printed out and the routine is skipped. Otherwise, a subroutine finds the address of a valid DAC or DCU. If there are none, a message is printed out and the routine is skipped. If a valid DAC or DCU is found, four amplifiers are located and their addresses, the DAC or DCU address, and the instructions "PATCH ACCORDING TO MANUAL", are printed out. Patching instructions are the same as for LKD10. Each S/H register is tested with each S/H code (0-3) and all combinations of hold commands. A value is sent to the DAC or DCU; the amplifiers, and S/H registers are read and compared. A register passes the test if it holds only when its hold code is sent.

NOTE: A DCU is only used if there are no DAC's in the console being tested.

ERROR - DAC or DCU 2aaa SENT nnnnn RECVD nnnnn

ERROR - AMP 0aaa SENT nnnnn RECVD nnnnn

ERROR - ADC 6aaa S/H CODE c HOLD $h_1h_2h_3$
SENT nnnnn RECVD nnnnn

where N(2,Ø,6) is component type

aaa is the address

c is the sample/hold code sent

$h_1h_2h_3$ is the hold command sent

h_i = 1 Group i should have held

h_1 = 0 Group i should have sampled

±nnnnn are the values sent and received.

LKD96 - RUN TIME CONFIGURATION DEFINITION

LKD96 allows the operator to define the configuration variables used by LKD at Run Time. This avoids re-building LKD to reflect a temporary change in the system configuration.

LKD97 - SET TOLERANCE

LKD97 types "ENTER (I2) NEW TOLERANCE FOR ANALOG VALUE TESTS" reads the value and types "NEW TOLERANCE IS PLUS OR MINUS nnØ MV." The operator should enter desired tolerance expresses as a fraction of reference. Two leading zeros are always assumed. Thus, for a tolerance of .0055 of reference (550 mv) only the 55 must be typed. Default Value = .0010 of reference (100 mv).

LKD98 - AUTOMATIC RUN MODE

LKD98 allows the operator to automatically run LKDØØ-LKD11. LKD98 requests a cycle count, which will be used by all routines which normally request a cycle count.