AAF01/VMS Subroutine Library User's and Installation Guide

Order Number: AA-GI29D-TE

Revision/Update Information:

Operating system and version:	VMS V5.0 or later
Software version:	AAF01/VMS Subroutine Library V2.0

October 1988

Copyright ©1988

All Rights Reserved.
Printed in U.S.A.

This document was prepared using VAX DOCUMENT, Version 1.1

Contents

PREFACE ix

CHAPTER 1 INTRODUCTION 1–1

1.1 HARDWARE REQUIREMENTS 1–2

1.2 SOFTWARE REQUIREMENTS 1–2

1.3 AAF01 HARDWARE SUMMARY 1–2
1.3.1 The AAF01 Registers 1–2
1.3.1.1 The AAF01 Command and Status Register (ACS) • 1–3
1.3.1.2 The Control Table Address Register (CTA) • 1–3
1.3.1.3 The Control Word Register (CWR) • 1–3
1.3.1.4 The Programmable Clock Register (PCR) • 1–3
1.3.1.5 The Data Buffer Register (DBF) • 1–3
1.3.2 Converter Codes 1–3
1.3.3 D/A Conversion Sequence 1–4

CHAPTER 2 AVAILABLE SUBROUTINES 2–1

2.1 ASSIGN CHANNEL AND DATAPATH 2–2

2.2 AAF$SETCHA SET CHANNEL 2–3

2.3 AAF$FREAST: READ STATUS OF AAF01 2–4

2.4 AAF$FREACO: READ COMMAND OUTPUT BIT 2–5

2.5 AAF$FSETCO: SET COMMAND OUTPUT BIT 2–6

2.6 AAF$FRESET: RESET AAF01 AND DRX11-C 2–7

2.7 AAF$FSTSBE: ENABLE SEQUENCE BREAK 2–8

Contents

2.8	AAF$FCLSBE: DISABLE SEQUENCE BREAK	2–9
2.9	AAF$FENECE: ENABLE EXTERNAL CLOCK	2–10
2.10	AAF$FDSECE: DISABLE EXTERNAL CLOCK	2–11
2.11	AAF$FENMET: ENABLE MEMORY TRANSFER	2–12
2.12	AAF$FDSMET: DISABLE MEMORY TRANSFER	2–13
2.13	AAF$FRECTA: READ FROM CTA	2–14
2.14	AAF$FLOCTA: WRITE TO CTA	2–15
2.15	AAF$FSTPCR: WRITE TO PCR	2–16
2.16	AAF$FLOCWT: WRITE CONTROL TABLE	2–17
2.17	AAF$FRECWT: READ CONTROL TABLE	2–18
2.18	AAF$FANOUT: ANALOGUE OUTPUT	2–19
2.19	AAF$FCVERT: START CONVERSION	2–20
2.20	AAF$CONLAR START LARGE BUFFER ADC CONVERSION	2–24
2.21	IO$_NOTIFY CLEAR LARGE BUFFER OVERFLOW CONDITION	2–27
2.22	AAF$FUNSOL: ENABLE/DISABLE ATTENTION AST	2–28
2.23	AAF$FAEXIT: RELEASE AAF01	2–29
2.24	AAF$FSIGNL: SIGNAL ERROR	2–30

2.25 AAF$FCVCOD: CALCULATE CONVERTER CODE 2–31

2.26 AAF$FAAFID: IDENTIFY AAF01 2–32

2.27 AAF$STATUS GET THE STATUS OF DRX11-C INTERFACE 2–33

2.28 SYS$WAITFR WAIT FOR EVENT FLAG 2–34

2.29 SYS$CLREF CLEAR EVENT FLAG 2–35

CHAPTER 3 PROGRAMMING THE AAF01 3–1

3.1 AAF01$CWT.COM 3–1

3.2 PROGRAMMING THE AAF01 3–3

3.3 AAF01$LINK.COM 3–4

CHAPTER 4 PREPARING TO INSTALL THE AAF01/VMS SUBROUTINE LIBRARY 4–1

4.1 REQUIRED OPERATING SYSTEM COMPONENTS 4–1

4.2 PREREQUISITE AND OPTIONAL SOFTWARE 4–1

4.2.1 Prerequisite Software 4–1

4.2.2 Optional Software 4–2

4.3 PREREQUISITE HARDWARE 4–2

4.4 VMS LICENSE MANAGEMENT FACILITY 4–2

4.5 VAXCLUSTER CONSIDERATIONS 4–3

Contents

4.6 INSTALLATION PROCEDURE REQUIREMENTS 4–3
4.6.1 Time 4–3
4.6.2 Privileges 4–3
4.6.3 Disk Space 4–3
4.6.4 System Parameters 4–4
4.6.5 Process Account Quotas 4–4
4.6.6 VMSINSTAL Requirements 4–4
4.6.7 Backing Up Your System Disk 4–4
4.6.8 License Management Utility Requirements 4–4

CHAPTER 5 INSTALLING THE AAF01/VMS SUBROUTINE LIBRARY 5–1

5.1 GENERAL INFORMATION 5–1
5.1.1 Accessing the Online Release Notes 5–1
5.1.2 Determining the Directories/Files Added to the System 5–1
5.1.3 Running the Installation Verification Procedure (IVP) 5–1
5.1.4 Aborting the Installation 5–2

5.2 THE INSTALLATION PROCEDURE 5–2
5.2.1 Invoking VMSINSTAL 5–2
5.2.2 Installation Questions 5–3
5.2.3 Informational Messages 5–5
5.2.4 Running the Installation Verification Procedure 5–6
5.2.5 Completing the Installation Procedure 5–6

5.3 ERROR RECOVERY 5–6

CHAPTER 6 AFTER INSTALLING THE AAF01/VMS SUBROUTINE LIBRARY 6–1

6.1 CLUSTER CONSIDERATIONS 6–1

6.2 TAILORING YOUR SYSTEM 6–1

6.3 DETERMINING AND REPORTING PROBLEMS 6–1

APPENDIX A SAMPLE INSTALLATION A–1

APPENDIX B DIRECTORIES/FILES INSTALLED ON YOUR SYSTEM B–1

APPENDIX C EXAMPLE PROGRAMS C–1

C.1 AAF01$SNGL (SINGLE BUFFER MODE) C–1

C.2 AAF01$DOBL (ALTERNATE BUFFER MODE) C–3

APPENDIX D AAF01 EXERCISER (ONLINE DIAGNOSTIC) D–1

D.1 AAF01 LOGIC TEST D–2

D.2 AAF01 LINEARITY TEST D–10

D.3 AAF01 CALIBRATION TEST D–11

D.4 AAF01 SELECTABLE OUTPUT TEST D–12

APPENDIX E TABLE OF ABBREVIATIONS E–1

APPENDIX F READER'S COMMENTS F–1

FIGURES

1–1 AAF01 Command and Status Register (ACS) Format 1–5
1–2 AAF01 Control Table Address Register (CTA) Format 1–5
1–3 AAF01 Control Word Register (CWR) Format 1–5
1–4 AAF01 Programmable Clock Register (PCR) Format 1–6
1–5 AAF01 Data Buffer FIFO (DBF) Format 1–6
1–6 AAF01 Control Table Layout 1–8

Contents

TABLES

1–1	Programmable Clock Frequencies and Conversion Rates	1–7
1–2	Conversion Table for Unipolar Operation	1–7
1–3	Conversion Table for Bipolar Operation	1–7
4–1	Disk Space Requirements	4–3

Preface

DOCUMENT OBJECTIVES

This manual describes the installation procedures for the AAF01/VMS Subroutine Library, and the routines available in that library. The available example and test programs are described as well.

INTENDED AUDIENCE

This manual is intended as guide for users performing the installation of the AAF01/VMS Subroutine Library, and as reference manual for programming the AAF01.

ASSOCIATED DOCUMENTS

The following documents are associated with this manual:

- AAF01 Digital/Analogue Conversion Subsystem For DRX11-C Family Of DMA Interfaces, Order number YG-C03ZC-00
- DRX11-C/VMS Driver User's and Installation Guide, Order number AA-HV08A-TE

PREREQUISITES

Readers of this manual are assumed to be familiar with the VMS operating system. Programmers using this manual are assumed to have FORTRAN knowledges.

1 INTRODUCTION

AAF01/VMS Subroutine Library

The AAF01/VMS Subroutine Library provides to the high level language user a simple interface to the AAF01 Digital/Analogue Conversion Subsystem with its functionality.

The AAF01 device has 16 single ended analogue output channels. The interface is controlled by a programmable conversion rate and by a 1K words Control Table RAM. Control Table loading is done either by program or by DMA. The conversion rate can be defined either by the internal clock which is programmable in the range of 400 kHz (2.5 microseconds) through 2.5 kHz (400 microseconds) in steps of 100 nsec, or through an external signal.

General Description:

1. The sequence of the data output is determined by the Control Table (1024 words RAM). This table can be generated by the indirect command procedure AAF01$CWT.COM, or directly in user program.
2. One Control Word consists of the channel address and the operation mode for the respective conversion.
3. The start of the conversion is initiated by calling the subroutine AAF$FCVERT or AAF$CONLAR.
4. The conversion rate is determined by the internal programmable clock unless the external clock input is used.
5. The set up of the AAF01 (loading the registers and the Control Table) is achieved by using the routines described in Chapter 2.
6. The conversion sequence can also be started through the SEQ CONT L input signal, provided mode 3 in the Control Word is selected (see Figure 1-3).

Performance:

A transfer rate of 400 Kwords/sec can be achieved. Saving the data may slow down the transfer rate, depending on the hardware and software used.

1.1 Hardware Requirements

The AAF01 is a high-speed, multichannel analogue output subsystem. It is connected to a VAX UNIBUS or Q-BUS via the DRX11-C interface. For proper installation read the manual AAF01 Digital/Analogue Conversion Subsystem For DRX11-C Family Of DMA Interfaces (order no. YG-C03ZC-00).

1.2 Software Requirements

The AAF01/VMS Subroutine Library uses the functionality of the DRX11-C/VMS driver.

The driver has to be generated and loaded successfully. For details on how to generate and load the driver see the DRX11-C/VMS Driver User's and Installation Guide (order no. AA-HV08A-TE).

1.3 AAF01 Hardware Summary

The following summarizes the layout of the AAF01 Registers, Converter Codes and the sequence for execution of D/A conversion.

1.3.1 The AAF01 Registers

The registers available on the AAF01 are summarized in the following table.

Register name	Mnemonic
AAF01 Data Buffer Register	DBF
AAF01 Command and Status Register	ACS
AAF01 Control Table Address Register	CTA
AAF01 Control Word Register	CWR
AAF01 Programmable Clock Register	PCR

1.3.1.1 The AAF01 Command and Status Register (ACS)

The ACS is a 12 bit read/write register (see Figure 1-1) used to control and indicate the status for the various functions of the data output system.

1.3.1.2 The Control Table Address Register (CTA)

The CTA is a 10 bit read/write counter register (see Figure 1-2). It contains the address of the current control word in the 1K by 12 bit Control Table memory.
The location in the Control Table memory (RAM) which is addressed by the CTA can be written or read via the Control Word Register (see Figure 1-6).

1.3.1.3 The Control Word Register (CWR)

The CWR is a 12 bit read/write register (see Figure 1-3). It is one of 1024 registers in the Control Table. When a conversion is started, the Control Word Register addressed by the CTA register at this time is used.

1.3.1.4 The Programmable Clock Register (PCR)

The PCR is a 12 bit write-only register (see Figure 1-4). The content of the PCR is moved in a 12 bit counter which is clocked by a quartz generator of 10 MHz (cf. Table 1-1).

The counter and the clock are only enabled when a conversion is started and if the current Control Word Mode CWM does not contain mode 3 (see Figure 1-3).

The clock is also available as output signal "RATE CLK OUT" (see Chapter 2.9 and Figure 1-1).

1.3.1.5 The Data Buffer Register (DBF)

The DBF is a 12 bit wide and 64 bit deep write-only silo (FIFO) (see Figure 1-5), in which the converter data are temporarily stored until being transferred to the D/A channel buffer.

1.3.2 Converter Codes

The D/A converter is loaded with the following 12 bit digital codes (4096 states):

- In unipolar operation (range 0V to +10V) complementary binary code (see Figure Table 1-2);
- In bipolar operation (range -10V to +10V) complementary offset binary code (see Table 1-3).

These operation modes are selectable using switch S1 on the AAF01 module (A6009). A new calibration must be done if the operation mode is changed.

For more information on switch settings and I/O signal description (RATE CLK OUT, SEQ CONT L, COUT L) see the manual AAF01 Digital/Analogue Conversion Subsystem For DRX11-C Family Of DMA Interfaces (order no. YG-C03ZC-00).

1.3.3 D/A Conversion Sequence

A digital to analogue conversion sequence is performed, using the AAF01/VMS Subroutine Library, in the following steps:

- Prepare the Control Table (using the indirect command procedure AAF01$CWT.COM). The control table can also be build in user program.
- Prepare the buffer with the analogue output data (using the subroutine AAF$FCVCOD to calculate the converter codes);
- Load the Control Table to the AAF01 (using the subroutine AAF$FLOCWT);
- Set the Control Table address CTA to the beginning of the Control Table (using the subroutine AAF$FLOCTA);
- Load the Programmable Clock Register PCR (using the subroutine AAF$FSTPCR);
- Start the conversion (using the subroutine AAF$FCVERT or AAF$CONLAR);
- Wait for the end of the conversion (using the VMS directive WAITFR and the subroutine AAF$FREAST).

  The End of Conversion Sequence bit EOC gives more information on the conversion end. If the EOC bit is cleared, it was a regular end of the conversion sequence. If the EOC bit is set, then one of the two bits, either the DBE (Data Buffer Empty) bit or the SBR (Sequence Break) bit, is set, too, thus indicating the reason for the (abnormal) end of conversion.

Please note that setting up the AAF01, i.e. loading the registers and the Control Table, can be done by program or by DMA.

Figure 1–1 AAF01 Command and Status Register (ACS) Format

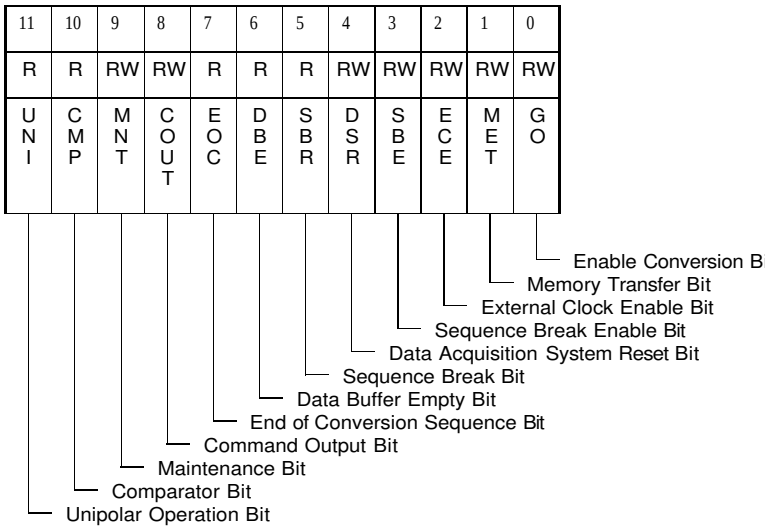


Figure 1–2 AAF01 Control Table Address Register (CTA) Format

11	10	9	8	7	6	5	4	3	2	1	0
	RW										
	CWA										

Figure 1–3 AAF01 Control Word Register (CWR) Format

11	10	9	8	7	6	5	4	3	2	1	0
			RW			RW					
			CWM			CHA					

CWM Control Word Mode; the meaning of the different possible values is given in the following list:

0 Conversion and increment CTA for next Control Word

1 Conversion and go to Control Word 0

2 Dummy cycle for current channel

3 as mode 2, but additionally wait for sequence start pulse and go to next Control Word (SEQ CONT L input signal must be used)

4 as mode 2, but additionally the SYS IN PROG L output signal is deasserted during this conversion

5 as modes 2 and 4, but additionally the Control Table starts at Control Word 0 again after this conversion

6 as modes 2 and 4, but the complement of the 12 bit PCR is used as cycle time for this conversion

7 as modes 2 and 4, but additionally the COUT L output signal is asserted during this conversion

CHA Channel Address for selection of one of the 16 output channels of the AAF01; bit 4 and 5 are provided for expansion to 48 channels via a MUX connector

Figure 1–4 AAF01 Programmable Clock Register (PCR) Format

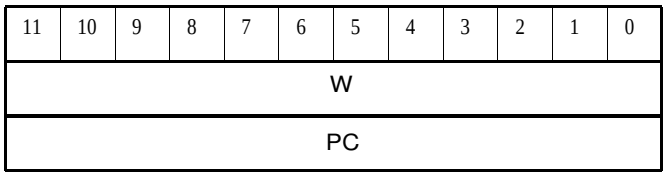


Figure 1–5 AAF01 Data Buffer FIFO (DBF) Format

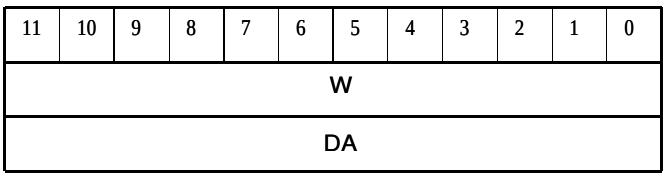


Table 1–1 Programmable Clock Frequencies and Conversion Rates

Number of Steps (decimal)	Frequency (kHz)	Conversion Time (microsec)
25	400	2.5
30	333.33	3.0
40	250	4
50	200	5
60	100	6
100	10	10
1000	2.5	100
4000		400

(Generator 10 MHz; 1 step is 100 nsec)

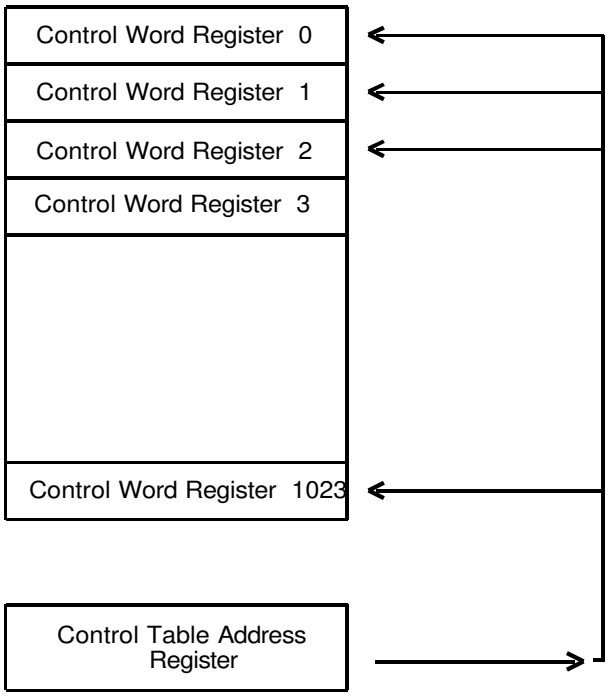
Table 1–2 Conversion Table for Unipolar Operation

Range	Complementary Binary Coding			
0 to +10V	MSB			LSB
+9.9976	000	000	000	000
+7.5000	001	111	111	111
+5.0000	011	111	111	111
+2.5000	101	111	111	111
+0.0024	111	111	111	110
+0.0000	111	111	111	111

Table 1–3 Conversion Table for Bipolar Operation

Range	Complementary Offset Binary Coding			
-10V to +10V	MSB			LSB
+9.9951	000	000	000	000
+7.5000	000	111	111	111
+5.0000	001	111	111	111
+0.0049	011	111	111	110
0.0000	011	111	111	111
- 0.0049	100	000	000	000
- 5.0000	101	111	111	111
- 9.9951	111	111	111	110
-10.0000	111	111	111	111

Figure 1–6 AAF01 Control Table Layout



The Control Table Address Register (CTA) contains the address of one of the 12-bit Control Word Registers in the Control Table.

2 AVAILABLE SUBROUTINES

Software Restrictions

The AAF01/VMS Subroutine Library supports the entire functionality of the AAF01 Digital/Analogue Conversion Subsystem.

Programming the AAF01

The subroutines available in the AAF01 Subroutine Library will be described detailed in this chapter. The steps of a sequence for a digital to analogue conversion are given in Chapter 1.3.3. The programming will be described in Chapter 4, and in Appendix C and Appendix D are sample programs.

2.1 ASSIGN CHANNEL AND DATAPATH

The subroutine AAF$FAINIT assigns a channel number for the specified logical or physical device and chooses the data path.

If you wish to access multiple devices, you must also specify a channel number to be associated with the device.

Calling Sequence:

CALL AAF$FAINIT (IND, %DESCR(DEVNAM), PATH, ICHA)

Input Parameters:

DEVNAM Type = longword (unsigned)
Call mechanism: by description
Address of a character string descriptor pointing to the device name. The string may be either a physical device name or a logical name. If the device name contains a colon, the colon and the characters that follow it are ignored. If the first character in the string is an underscore character (_), the name is considered a physical device name. Otherwise, the name is considered a logical name. A logical name translation is performed until either a physical device name is found or the system default number of translation has been performed.

PATH Type = longword (unsigned)
Call mechanism = by reference
Data Path
1 = Buffered Data Path
else = Direct Data Path

ICHA Type = longword (unsigned)
Call mechanism = by reference
(Optional) Specifies a channel number, 1 through 8, to be associated with the device.

Output Parameters:

IND Type = longword (unsigned)
Call mechanism = by reference

Returns the success or failure status
1 = successful completion
nnn = VAX/VMS status code

SS$_IVCHAN = ICHA<1 or ICHA>8.

2.2 AAF$SETCHA SET CHANNEL

This routine sets the channel to be used by subsequent calls to AAF01 subroutines. The channel must have been initialized with the AAF$FAINIT call.

Calling Sequence:

CALL AAF$SETCHA (IND,ICHA,IPRE)

Input Parameters:

ICHA Type = longword (unsigned)
Call mechanism = by reference
Number of the channel which is to be the current channel.

Output Parameters:

IPRE Type = longword (unsigned)
Call mechanism = by reference
(Optional) Number of the channel before the call. If the current channel is changed in an AST routine, then it should be "reset" to IPRE before AST routine exit.

IND Type = longword (unsigned)
Call mechanism = by reference
Return status.
SS$_NORMAL = Success.
SS$_IVCHAN = Invalid channel number,either ICHA<1 or ICHA>8 or ICHA not initialized.

2.3 AAF$FREAST: Read Status of AAF01

Calling this routine of the AAF01/VMS Subroutine Library will allow to read the read-only bits of the AAF01 Command and Status Register, ACS:

UNI - Unipolar Operation bit

CMP - Comparator Input bit

EOC - End of Conversion Sequence bit

DBE - Data Buffer Empty bit

SBR - Sequence Break bit

Calling sequence:

CALL AAF$FREAST (JUNI, JCMP, JEOC, JDBE, JSBR)

Input parameters:

None

Output parameters:

JUNI Type = word (unsigned)
Call mechanism = by reference
Unipolar Operation bit; indicates the position of switch S1:
JUNI = 1 means unipolar operation (0V to +10V)
JUNI = 0 means bipolar operation (-10V to +10V)

JCMP Type = word (unsigned)
Call mechanism = by reference
Comparator Input bit
JCMP = 1 means that the COMP IN line is more negative than the output of channel 0 of the AAF01

JEOC Type = word (unsigned)
Call mechanism = by reference
End of Conversion Sequence bit
JEOC = 0 means a regular end of the conversion

JEOC = 1 means an abnormal end of the conversion sequence; in this case the DBE bit or the SBR bit gives more information.

JDBE Type = word (unsigned)
Call mechanism = by reference
Data Buffer Empty bit
JDBE = 1 means that the buffer with the data for output was empty during the conversion sequence

JSBR Type = word (unsigned)
Call mechanism = by reference
Sequence Break bit
JSBR = 1 means that the SEQ CONT L input signal was set

More information on these bits may be found in the manual AAF01 Digital/Analogue Conversion Subsystem For DRX11-C Family of DMA Interfaces (order number YG-C03ZC-00).

2.4 AAF$FREACO: Read Command Output Bit

Calling this routine will allow to read the Command Output bit COUT of the AAF01 Command and Status Register, ACS.

Calling sequence:

CALL AAF$FREACO (JNDEX)

Input parameters:

None

Output parameters:

JNDEX Type = word (unsigned)
Call mechanism = by reference
Command Output bit
JNDEX = 1 means that COUT is set
JNDEX = 0 means that COUT is not set

2.5 AAF$FSETCO: Set Command Output Bit

Calling this routine will allow to set the Command Output bit COUT of the AAF01 Command and Status Register, ACS.

Calling sequence:

CALL AAF$FSETCO

Input parameters:

None

Output parameters:

None

2.6 AAF$FRESET: Reset AAF01 and DRX11-C

Calling this routine will allow to reset the AAF01: the registers ACS (AAF01 Command and Status Register), PCR (Programmable Clock Register) and DBF (Data Buffer Register) will be reset as well as the analogue output channels.

Calling sequence:

CALL AAF$FRESET (IND,IFNCT0)

Input parameters:

IFNCT0 Type = longword (unsigned)
Call mechanism = by reference

DRX11-C Reset modifier
0 = without FNCT0
1 = with FNCT0

Output parameters:

IND Type = longword (unsigned)
Call mechanism = by reference

VMS return status

2.7 AAF$FSTSBE: Enable Sequence Break

Calling this routine will allow to set the Sequence Break Enable bit SBE of the AAF01 Command and Status Register, ACS.

Calling sequence:

CALL AAF$FSTSBE

Input parameters:

None

Output parameters:

None

For more information on this bit please refer to the manual AAF01 Digital/Analogue Conversion Subsystem For DRX11-C Family Of DMA Interfaces (order number YG-C03ZC-00).

2.8 AAF$FCLSBE: Disable Sequence Break

Calling this routine will allow to clear the Sequence Break Enable bit SBE of the AAF01 Command and Status Register, ACS.

Calling sequence:

CALL AAF$FCLSBE

Input parameters:

None

Output parameters:

None

2.9 AAF$FENECE: Enable External Clock

Calling this routine will allow to set the External Clock Enable bit ECE of the AAF01 Command and Status Register, ACS.

Calling sequence:

CALL AAF$FENECE

Input parameters:

None

Output parameters:

None

2.10 AAF$FDSECE: Disable External Clock

Calling this routine will allow to clear the External Clock Enable bit ECE of the AAF01 Command and Status Register, ACS.

Calling sequence:

CALL AAF$FDSECE

Input parameters:

None

Output parameters:

None

2.11 AAF$FENMET: Enable Memory Transfer

Calling this routine will allow to set the Memory Transfer bit MET of the AAF01 Command and Status Register, ACS. The DMA-cycle contains the 12-bit contents of the control word which belongs to the following conversion. This affects the conversion data rate by requiring another DMA-cycle.

data buffer:

buffer header

1. control word
1. data word
2. control word
2. data word

etc ...

Calling sequence:

CALL AAF$FENMET

Input parameters:

None

Output parameters:

None

2.12 AAF$FDSMET: Disable Memory Transfer

Calling this routine will allow to clear the Memory Transfer bit MET of the AAF01 Command and Status Register, ACS.

Calling sequence:

CALL AAF$FDSMET

Input parameters:

None

Output parameters:

None

2.13 AAF$FRECTA: Read From CTA

Calling this routine will allow to read the AAF01 Control Table Address Register CTA.

Calling sequence:

CALL AAF$FRECTA (JNDEX)

Input parameters:

None

Output parameters:

None

JNDEX Type = word (unsigned)
Call mechanism = by reference

Actual position within the 1K Control Table, i.e.
$0 \leq$ JNDEX ≤ 1023

2.14 AAF$FLOCTA: Write To CTA

Calling this routine will allow to write to the AAF01 Control Table Address Register CTA.

Calling sequence:

CALL AAF$FLOCTA (JSTAT, JNDEX)

Input parameters:

JNDEX Type = word (unsigned)
Call mechanism = by reference

Position within the 1K Control Table,
i.e.0 $\leq$ JNDEX $\leq$ 1023

Output parameters:

JSTAT Type = word (unsigned)
Call mechanism = by reference

Return status

2.15 AAF$FSTPCR: Write To PCR

Calling this routine will allow to write to the AAF01 Programmable Clock Register PCR.

Calling sequence:

CALL AAF$FSTPCR (JSTAT, JCOUNT)

Input parameters:

JCOUNT Type = word (unsigned)
Call mechanism = by reference

Number of steps (in multiple of 100 nanoseconds);
25 $\leq$ JCOUNT $\leq$ 4095 (cf. Table 1-1)

Output parameters:

JSTAT Type = word (unsigned)
Call mechanism = by reference

Return status

2.16 AAF$FLOCWT: Write Control Table

Calling this routine will allow to write the AAF01 Control Word Register CWR, one or several of them, to the Control Table.

Calling sequence:

CALL AAF$FLOCWT (JSTAT, JTAB, JLOW, JHIGH, JCTA)

Input parameters:

JTAB
Type = word (unsigned)
Call mechanism = by reference
Integer array containing the Control Table

JLOW
Type = word (unsigned)
Call mechanism = by reference
Low position within JTAB where the transfer starts;
$0 \leq$ JLOW ≤ 1023

JHIGH
Type = word (unsigned)
Call mechanism = by reference
High position within JTAB where the transfer ends;
$0 \leq$ JHIGH ≤ 1023

JCTA
Type = word (unsigned)
Call mechanism = by reference
First position within the Control Table where loading starts;
$0 \leq$ JCTA ≤ 1023

Output parameters:

JSTAT
Type = word (unsigned)
Call mechanism = by reference
Return status

JCTA
Type = word (unsigned)
Call mechanism = by reference
Position within the Control Table after the transfer of the Control Words; 0 $\leq$ JCTA ≤ 1023

2.17 AAF$FRECWT: Read Control Table

Calling this routine will allow to read the AAF01 Control Word Register CWR, one or several of them, from the Control Table.

Calling sequence:

CALL AAF$FRECWT (JSTAT, JTAB, JLEN, JLOW, JHIGH)

Input parameters:

JLEN
Type = word (unsigned)
Call mechanism = by reference
Length in bytes of JTAB

JLOW
Type = word (unsigned)
Call mechanism = by reference
Low position within the Control Table where to start the transfer;
$0 \leq$ JLOW ≤ 1023

JHIGH
Type = word (unsigned)
Call mechanism = by reference
High position within the Control Table where to end the transfer;
$0 \leq$ JHIGH ≤ 1023

Output parameters:

JSTAT
Type = word (unsigned)
Call mechanism = by reference
Return status

JTAB
Type = word (unsigned)
Call mechanism = by reference
Integer array into which the Control Word Registers are read

JLOW
Type = word (unsigned)
Call mechanism = by reference
Position + 1 (INTEGER*2) within the Control Table where the transfer ended; $1 \leq$ JLOW ≤ 1024

2.18 AAF$FANOUT: Analogue Output

Calling this routine will allow to output a volt value to one of the analogue output channels of the AAF01.

Calling sequence:

CALL AAF$FANOUT (JSTAT, VOLT, JCHN)

Input parameters:

VOLT Type = D_Floating
Call mechanism = by reference
Value of the voltage to be output

JCHN Type = word (unsigned)
Call mechanism = by reference
Channel of the AAF01 to which the output is to be made

Output parameters:

JSTAT Type = word (unsigned)
Call mechanism = by reference
Return status

2.19 AAF$FCVERT: Start Conversion

Calling this routine will allow to start the conversion of the AAF01, that is to set up the necessary QIO parameters and enable the data transfer.

Calling sequence:

CALL AAF$FCVERT (IEFN, JIOST, JTAB1, JLEN1,
JTAB2, JLEN2, JBLK, JTIME, JBURST

Input parameters:

IEFN Type = longword (unsigned)
Call mechanism = by reference
Number of the event flag to be set each time when one buffer is transferred.

JTAB1 Type = word (unsigned)
Call mechanism = by reference
Address of the first data buffer (word aligned). The first longword of the buffer is used as the bufferheader. Each time the buffer is wtitten, the DRX11-C/VMS driver writes the current buffercount in this header.
The driver decrements the buffercount by one each time a buffer is written. After the buffer is filled/processed by the user, he has to indicate this by clearing the bufferheader. Each time a new buffer is started, the driver checks if the buffer header is zero. If at this time the header is not zero, the driver assumes that the user was not able to process the data. The current transfer is terminated and a SS$_DATAOVERRUN error is returned.

JLEN1 Type = longword (unsigned)
Call mechanism = by reference
Size of the first buffer in bytes excluding the bufferheader. The size must be greater than 0 and even. The maximum size is 129020 bytes.

Note: The buffer should have a reasonable size for proper synchronization, which will heavily depend on the data transfer rate.

JTAB2 Type = word (unsigned)
Call mechanism = by reference
Address of the second data buffer (word aligned). See description of JTAB1.

JLEN2 Type = longword (unsigned)
Call mechanism = by reference
Size of the second buffer in bytes excluding the bufferheader. The size must be greater than 0 and even. The maximum size is 129020 bytes.

JBLK Type = longword (unsigned)
Call mechanism = by reference

Buffer count
Setting the buffer count to 1 means a request to transfer only buffer JTAB1 to the AAF01 (single buffer mode).
Setting the buffer count to >= 2 means a request to transfer buffer JTAB1 and buffer JTAB2 in alternate mode (alternate buffer mode), that is transfer first buffer JTAB1, then buffer JTAB2, then again buffer JTAB1 and so on, until the buffer count is exhausted.
Setting the buffer count to 0 means a request to transfer the two buffers JTAB1 and JTAB2 in alternate mode indefinitely; only an error condition or a $CANCEL I/O request may stop the transfer.

JTIME Type = longword (unsigned)
Call mechanism = by reference
Timeout count specifying the timeout in seconds; this will be the maximum time period within which an interrupt must occur, i.e. one buffer will be output. This is the maximum time allowed to write one buffer. An unspecified timeout or a timeout value of zero will set the timeout to 65536 seconds.

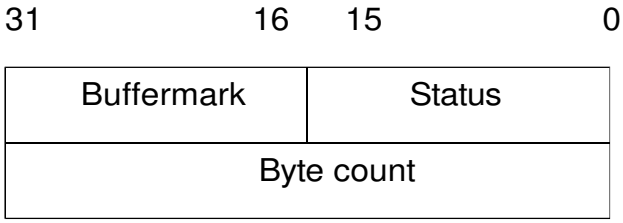
JBURST Type = longword (unsigned)
Call mechanism = by reference
Burst mode modifier
0 = no burst mode
1 = burst mode

Output parameters:

JIOST Type = longword (unsigned)
Call mechanism = by reference
A 2 longword integer array which is to receive the I/O status of the QIO call done by using this subroutine; this I/O status block contains the following information:

first word of first longword:	return status
second word of first longword:	buffermark
second longword:	byte count

The DRX11-C/VMS I/O status block (two longwords) has the following format:



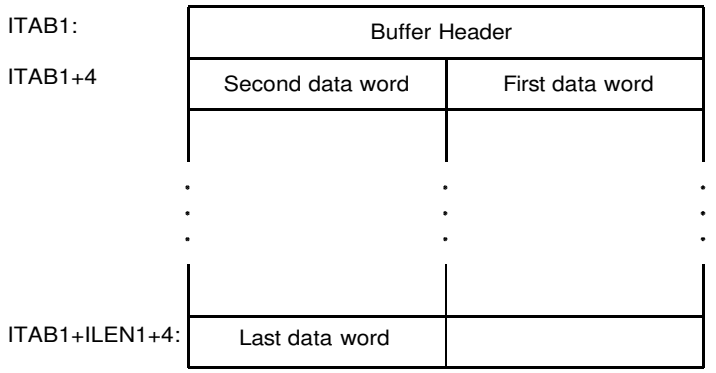
Buffermark:

This is the Block number of the last block being transferred. The buffermark is updated at the completion of each block.

Byte Count:

The byte count of the last block being transferred up to the point of termination of the I/O function.

DATA BUFFER FORMAT



The maximum total size of the two buffers is 251900 bytes.

When transferring data in alternate buffer mode, the user program must acknowledge the completion of each buffer. The user program must also define an I/O status block.

Description of alternate buffer data transfer and how to synchronize the user program with it,

1 The data transfer is started using buffer 1.

2 The hardware completes buffer 1 and switches automatically to buffer 2.

3 The value 1 will be written to the buffermark of the I/O status block. The event flag will be set, and the status SYSTEM$_FACILITY will be returned.

4 The user program recognizes the event flag (and/or the change in the buffermark) and prepares buffer 1 for the next data transfer. Meanwhile the data transfer is continuing using buffer 2.

5 The user program signals that buffer 1 is ready for being transferred by clearing the buffer header in buffer 1 and the event flag.

6 The hardware completes buffer 2 and switches automatically to buffer 1.

7 The value 2 will be written to the buffermark of the I/O status block. The event flag will be set and the status SYSTEM$_FACILITY will be returned.

8 The user program recognizes the event flag (and/or the change in the buffermark) and prepares buffer 2 for the next data transfer. Meanwhile the data transfer is continuing using buffer 1.

9 The user program signals that buffer 2 is ready for being transferred by clearing the buffer header in buffer 2 and the event flag.

10 The steps 2 through 9 will be repeated until the specified buffer count is exhausted. Prior to the last buffer transfer the device will switch back to single buffer mode and the status SS$_NORMAL will be returned.

If during step 3 and step 7 the driver detects that the bufferheader was not cleared by the user program, the transfer is terminated with the return status code SS$_DATAOVERUN (data overrun). In this case the buffermark points to the last buffer used and the byte count is set to the number of bytes transferred from this buffer.

For more details on this please refer to the DRX11-C/VMS Driver User's and Installation Guide (order number AA-HV08A-TE).

Note: The data buffers contain four additional bytes allocated to the "buffer header". This header must be zero at the beginning of the transfer. The driver will write the current block count into the buffer header at the end of each block. The driver, in turn, expects the user program to clear this count in order to mark the buffer available for another transfer.

2.20 AAF$CONLAR Start large buffer ADC Conversion

The subroutine AAF$CONLAR loads the QIO parameters and sets the GO bit in the AAF01 Command and Status Register to start the large buffer conversion.

The DIAGNOSE privilege is needed to use this subroutine.

Calling Sequence:

CALL AAF$CONLAR (IND,%VAL(IEFN),IOST,LARBUF,%VAL(LARLEN), %VAL(SUBLEN),%VAL(SUBBUFS),%VAL(TIMEOUT), %VAL(IBURST))

Input Parameters:

IEFN — Type = longword (unsigned)
Call mechanism = by value
Number of the event flag that is to be set when one sub-buffer is full.

LARBUF — Type = word (unsigned)
Call mechanism = by reference
Address of the large buffer (word aligned) for the data to be written to the AAF01.

LARLEN — Type = longword (unsigned)
Call mechanism = by value
Length of the large buffer in bytes (must not be 0 or odd). Maximum size is restricted only by the free physical memory available. The length must be an even multiple of the sub-buffer length.

SUBLEN — Type = longword (unsigned)
Call mechanism = by value
Length of one sub-buffer in bytes (must not be 0 or odd). Maximum size is 125442 bytes.

Note: The sub-buffer should have a reasonable size for proper synchronization, which will heavily depend on the data transfer rate.

SUBBUFS Type = longword (unsigned)
Call mechanism = by value
Number of sub-buffers (greater than 0).
The large buffer can also be transferred partially, if the subbuffer to largebuffer ratio is N, then the number of subbuffers can be from 1 to N (where N subbuffers is the one large buffer).
If the large buffer should be transferred more than once give a subbuffer count greater than N (ie. 3*N means three times). After N subbuffers a IO$_NOTIFY must be issued to clear the large buffer overflow condition.

TIMEOUT Type = longword (unsigned)
Call mechanism = by value
Time-out count is seconds

IBURST Type = longword (unsigned)
Call mechanism = by value
Burst mode modifier
0 = no burst mode
1 = burst mode

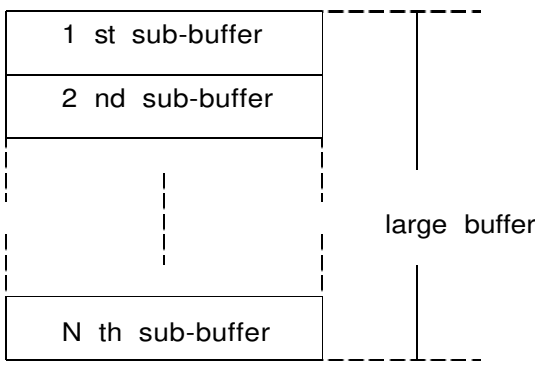
Output Parameters:

IND Type = longword (unsigned)
Call mechanism = by reference
Returns the success or failure status
1 = successful completion
nnn = VAX/VMS status code

IOST Type = longword (unsigned)
Call mechanism = by reference
Address of a quadword I/O status block that is to receive final completion status.

The size of the large buffer must be an even multiple of the sub-buffer:

large buffer size = n * sub-buffer size (where n=2,4,6...)



Format of I/O status block:

First word of first longword:	Return Status
Second word of first longword:	Buffermark
Second longword:	Byte Count

The byte count is the size (or a part of it) of the last large buffer being transferred up to the point of termination of the I/O function.

For more details of large buffer transfers refer to the DRX11-C/VMS driver manual.

2.21 IO$_NOTIFY Clear large buffer overflow condition

This function allows a large buffer transfer to wrap around to the beginning of the large buffer.
If the large buffer should be transferred more than once, after N (where N sub-buffers is one large buffer) subbuffers a IO$_NOTIFY must be issued to clear the large buffer overflow condition.

IND = SYS$QIOW (%VAL(IEFN),%VAL(ICHAN),
%VAL(IO$_NOTIFY),IOSB,,,,,,,,)

Input Parameters:

IEFN Type= longword (unsigned)
Call mechanism = by value
Number of the event flag to be set when the request is completed.

Note: Must be different from the IEFN used by AAF$CONLAR.

ICHAN Type = longword (unsigned)
Call mechanism = by value
Number of the I/O channel assigned to the device to which the request is directed.

Output Parameters:

IOSB Type = word (unsigned)
Call mechanism = by reference
Address of a qudword I/O status block that is to receive the first completion status.

Note: Must be different from the IOST used by AAF$CONLAR.

IND Returns the success or failure status
1 = successful completion
nnn = VAX/VMS status code

The current I/O-channel is stored in ICHAN. To obtain it define the following in your FORTRAN program:

```
INTEGER*2   IOST(4)
INTEGER*4   ICHAN

COMMON  /AAF_DATA/ IOST,ICHAN
```

2.22 AAF$FUNSOL: Enable/Disable Attention AST

Calling this routine will allow to enable/disable attention AST recognition.

Calling sequence:

CALL AAF$FUNSOL (JASTAD,ASTPRM)

Input parameters:

JASTAD Type = longword (unsigned)
Call mechanism = by reference

Depending on the value of this parameter, two different functions of the call are possible:

1 JASTAD = Address of the attention AST routine, then the function of the call is to connect to unsolicited interrupts. The attention AST routine will be executed upon receipt of one of the following interrupting conditions: momentary STAT0 input received or already set, AAF01 is offline, non existent memory trap (faulty DRX11-C hardware).

Upon entry of the attention AST routine, the ASTPRM contains the SCR register content of the DRX11-C interface at the time of the interrupt.

While a conversion is running the AST notification is disabled. If one of the above mentioned interrupt conditions occurs the conversion is stopped and the status SS$_ ENDOFFILE is returned to the calling program.

The function is disabled by issuing a disconnect from unsolicited interrupt, by issuing a $CANCEL system service or by exit of the user program.

2 JASTAD = 0, then the function of the call is to disconnect from unsolicited interrupts. (Disable the unsolicited interrupts.)

Output parameters:

ASTPRM Type = word (unsigned)
Call mechanism = by reference

Upon entry of the attention AST routine, the ASTPRM contains the SCR register content of the DRX11-C interface at the time of the interrupt.

2.23 AAF$FAEXIT: Release AAF01

Calling this routine will allow to reset the DRX11-C.

Calling sequence:

CALL AAF$FAEXIT(MASK)

Input parameters:

MASK Type = word (unsigned)
Call mechanism = by reference

Mask to be placed into the DBR register of DRX11-C

Output parameters:

None

2.24 AAF$FSIGNL: Signal Error

Calling this routine will allow to analyze an error code returned by one of the other routines of the AAF01.

Calling sequence:

CALL AAF$FSIGNL (JERR)

Input parameters:

JERR — Type = word (unsigned)
Call mechanism = by reference

Integer value of an error code encountered by one of the AAF01 subroutines, see below

Output parameters:

None

Depending on the value of the error code JERR, error messages may be output to an implicit logical unit (as is done by the FORTRAN statement: TYPE *,list):

List of error messages:

Error code	Error message
- 2	AAF01—AAF$FLOCWT/AAF$FRECWT: Low limit is out of range
- 3	AAF01—AAF$FLOCWT/AAF$FRECWT: High limit is out of range
- 4	AAF01—AAF$FLOCWT/AAF$FRECWT: Low limit larger than high limit
- 5	AAF01—AAF$FRECWT: Length parameter is too small
- 6	Reserved
- 7	AAF01—AAF$FCVCOD: Voltage value is out of range
- 8	AAF01—AAF$FSTPCR: Clock counter is out of range
- 9	Reserved
- 10	AAF01—AAF$FLOCTA: CTA address is out of range
- 11	AAF01—AAF$FANOUT: Channel number is out of range
<-12	AAF01—AAF$FSIGNL: Error code is not yet defined

2.25 AAF$FCVCOD: Calculate Converter Code

Calling this routine will allow to calculate the converter code for a desired voltage value.

Calling sequence:

CALL AAF$FCVCOD (JSTAT, VOLT, JCOD)

Input parameters:

VOLT Type = D_Floating
Call mechanism = by reference

Voltage value

Output parameters:

JSTAT Type = word (unsigned)
Call mechanism = by reference
Return status

JCOD Type = word (unsigned)
Call mechanism = by reference
Converter code corresponding to the voltage value VOLT for the actual AAF01

This routine will read the Unipolar Operation bit of the AAF01 in order to choose the proper conversion algorithm.

2.26 AAF$FAAFID: Identify AAF01

Calling this routine will allow to get the version number and the patch level of the current AAF01 Subroutine Library.

Calling sequence:

CALL AAF$FAAFID (IVERSN, IPATCH)

Input parameters:

None

Output parameters:

IVERSN Type = word (unsigned)
Call mechanism = by reference
Version number of the current AAF01 Subroutine Library

IPATCH Type = word (unsigned)
Call mechanism = by reference
Patch level of the current AAF01 Subroutine Library

As an example, for the AAF01 Subroutine Library V2.0, the values returned by the routine AAF$FAAFID are: IVERSN = 2, IPATCH = 0
.

2.27 AAF$STATUS GET THE STATUS OF DRX11-C INTERFACE

The subroutine AAF$STATUS will get the contents of the DRX11-C hardware registers.

Calling Sequence:

CALL AAF$STATUS (IND,BUFFER)

Input Parameters:

None

Output Parameters:

IND — Type = longword (unsigned)
Call mechanism = by reference
Returns the success or failure status
1 = successful completion
nnn = VAX/VMS status code

BUFFER — Type = word (unsigned)
Call mechanism = by reference
18-byte long array to get the DRX11-C register contents

WORD 0: - SCR contents
WORD 1: - COR contents
WORD 2: - BAR1 contents
WORD 3: - WCR1 contents
WORD 4: - BAR2 contents
WORD 5: - WCR2 contents
WORD 6: - Current WCO
WORD 7: - Current ACO
WORD 8: - XA22 contents (DRQ11-C only)

For detailed information see the "DRX11-C/VMS Driver User's and Installation Guide" (document no. AA-HV08A-TE).

2.28 SYS$WAITFR Wait for Event Flag

The system service SYS$WAITFR waits for a specific event flag and returns immediately if the flag is set. Otherwise, the process is placed in a wait state until the event flag is set.

Calling Sequence:

CALL SYS$WAITFR (%VAL(IEFN))

Input Parameter:

IEFN Type = longword (unsigned)
Call mechanism = by value

Number of the event flag for which to wait

More information is available from the SYSTEM SERVICE REFERENCE MANUAL.

2.29 SYS$CLREF Clear Event Flag

The system service SYS$CLREF sets a specific event flag in a local or common event flag cluster to 0.

Calling Sequence:

CALL SYS$CLREF (%VAL(IEFN))

Input Parameter:

IEFN Type = longword (unsigned)
Call mechanism = by value

Number of the event flag to be cleared

More information is available in the SYSTEM SERVICES REFERENCE MANUAL.

3 PROGRAMMING THE AAF01

3.1 AAF01$CWT.COM

This indirect command procedure may be used to create the Control Table. For each desired channel the channel number and the Control Word mode will be inserted in one Control Word. The created Control Table will be assembled to have a object file.

A sample run of the indirect command procedure:

```
$ @AAF01$CWT

 Generation of the AAF01 control word table

 Input parameter are:
  Channel: channel address
     for selection of one of the
     16 input channels.

  Control word mode:

 The Control Word mode specifies one of these modes:

 0  Conversion and increment CTA for next Control Word
 1  Conversion and go to Control Word 0
 2  Dummy cycle for current channel
 3  as mode 2, but additionally wait for sequence start
    pulse and go to next Control Word (SEQ CONT L input
    signal must be used)
 4  as mode 2, but additionally the SYS IN PROG L output
    signal is deasserted during this conversion
 5  as modes 2 and 4, but additionally the Control Table
    starts at Control Word 0 again after this conversion
 6  as modes 2 and 4, but the complement of the 12 bit
    PCR is used as cycle time for this conversion
 7  as modes 2 and 4, but additionally the COUT L output
    signal is asserted during this conversion

 ------------------------------------------------------

      NAME FOR THE AAF01 CONTROL WORD TABLE FILE
      E.G. CONTROL
name: CONTROL1
channel: 0
control word mode: 0
channel: 1
control word mode: 0
channel: 2
control word mode: 0
channel: 3
control word mode: 0
channel: 4
control word mode: 1
channel:
$
$ MACRO CONTROL1/LIS
$ SET NOVERIFY
$
```

Example of a control table

```
        .TITLE CONTROLTABLE
;
;        AAF01 control word table
;
        .PSECT AAF_CONTROLTABLE PIC,GBL,SHR,OVR,NOEXE,LONG
;
AAF_CONTROLTABLE:
;
        .WORD 0          ; CHAN = 0 MODE = 0
        .WORD 1          ; CHAN = 1 MODE = 0
        .WORD 2          ; CHAN = 2 MODE = 0
        .WORD 3          ; CHAN = 3 MODE = 0
        .WORD 68         ; CHAN = 4 MODE = 1
;
        .END
```

3.2 Programming the AAF01

A digital to analogue conversion sequence is performed, using the AAF01/VMS Subroutine Library, in the following steps:

- Prepare the Control Table (using the indirect command procedure AAF01$CWT.COM), or build it directly in program.
- Prepare the buffer with the analogue output data (using the subroutine AAF$FCVCOD to calculate the converter codes);
- Load the Control Table to the AAF01 (using the subroutine AAF$FLOCWT);
- Set the Control Table address CTA to the beginning of the Control Table (using the subroutine AAF$FLOCTA);
- Load the Programmable Clock Register PCR (using the subroutine AAF$FSTPCR);
- Start the conversion (using the subroutine AAF$FCVERT or AAF$CONLAR);
- Wait for the end of the conversion (using the VMS directive WAITFR and the subroutine AAF$FREAST).

The End of Conversion Sequence bit EOC gives more information on the conversion end. If the EOC bit is cleared, it was a regular end of the conversion sequence. If the EOC bit is set, then one of the two bits, either the DBE (Data Buffer Empty) bit or the SBR (Sequence Break) bit, is set, too, thus indicating the reason for the (abnormal) end of conversion.

Further information on programming the AAF01 may be found in the example programs which are listed in the distribution kit.

Every FORTRAN user program which is to access the Control Table created by the indirect command procedure AAF01$CWT.COM must have the following specification statements:

```
INTEGER*2       TAB(M)
COMMON /AAF_CONTROLTABLE/TAB
```

AAF_CONTROLTABLE	PSECT of the Control Word Table (Defined by using AAF01$CWT.COM).
M	Is the number of elements in the Control Word Table 1 less or equal to M less or equal to 1024

If unsolicited interrupt recognition is desired, the subroutine AAF$FUNSOL for enabling the attention AST has to be called in the user program right after the initialization of the AAF01.

3.3 AAF01$LINK.COM

Program listing of command file AAF01$LINK.COM:

```
$ GOTO START
!
! AAF01$LINK.COM  COMMAND FILE TO LINK THE CONTROL TABLE FILE
!           AND THE USER PROGRAM WITH THE AAF01/VMS SUBROUTINE LIBRARY
!
! E.G.: @AAF01$LINK AAF01$CONTAB USERPROGRAM
!
$ START:
$ SET VERIFY
$ LINK 'P1','P2'/EXE,AAF01$LIB/LIB
$ SET NOVERIFY
```

Example to link the program USERPROGRAM with the control table CONTROL1.

```
$ @AAF01$LINK CONTROL1 USERPROGRAM
$ LINK CONTROL1, USERPROGRAM/EXE, AAF01$LIB/LIB
```

4 PREPARING TO INSTALL THE AAF01/VMS SUBROUTINE LIBRARY

This chapter discusses the preparations and requirements necessary for installing the AAF01/VMS Subroutine Library.

This manual applies to Version 2.0 of the AAF01/VMS Subroutine Library and all subsequent maintenance releases up to the next major product release.

Your bill of materials (BOM) and indented bills report (BIL) specify the number and contents of your media. Be sure to verify the contents of your kit with this information. If your kit is damaged or if you find that parts of the kit are missing, contact your DIGITAL representative.

The AAF01/VMS distribution kit provides online Release Notes. DIGITAL strongly recommends that you read the Release Notes before proceeding with the installation.

4.1 Required Operating System Components

For VMS V5.x systems, the following VMS classes are required for full functionality of this layered product:

- BASE Base System
- PROG Programming Support
- SYSP System Programming Support

4.2 Prerequisite and Optional Software

This section discusses the software you must have installed on your system before installing the AAF01/VMS Subroutine Library. The section also includes information about software that you can use together with the AAF01/VMS Subroutine Library. Please refer to the System Software Addendum (SSA) for the required versions of these prerequisites. The SSA is part of the Software Product Description (SPD).

4.2.1 Prerequisite Software

A valid version of the VMS V5.x Full Kit, a loaded DRX11-C/VMS Device Driver (with valid license and with the file SYS$SYSROOT:[SYSHLP.EXAMPLES.DRX11C]DRX11C$DEFINC.FOR) and the registered and loaded AAF01-SUBROUTINE-LIBRARY authorization key are needed in order to install the AAF01/VMS Subroutine Library.

Please refer to the System Software Addendum (SSA) for the required versions of these prerequisites. The SSA is part of the Software Product Description (SPD).

4.2.2 Optional Software

- VAX FORTRAN (FORSYSDEF.TLB has also to be build)
- or any other VAX/VMS high level languages

Please refer to the System Software Addendum (SSA) for the required versions of these optional products. The SSA is part of the Software Product Description (SPD).

4.3 Prerequisite Hardware

- one DRE11-C or DRQ11-C interface
- one AAF01 interface
- test connector for running the IVP

4.4 VMS License Management Facility

The VMS License Management Facility is available with Version 5.0 of the VMS operating system. If you are installing the AAF01/VMS Subroutine Library on a VMS V5.0 system, you have to register your software license first.

The Product Authorization Key (PAK), which is shipped with the AAF01/VMS distribution, contains the necessary license registration information. The PAK is a paper certificate that contains information about the license.

It is required that the the AAF01-SUBROUTINE-LIBRARY license has been registered before you perform the installation. During the installation, you are asked if you have registered and loaded the the AAF01-SUBROUTINE-LIBRARY license. If you have not already done so, please answer "NO". Once you have performed the license registration and loaded, you may restart the installation.

To register a license under VMS V5.0, first log in to the system manager's account, SYSTEM, offering you two different possibilities to perform the registration:

- Invoke the SYS$UPDATE:VMSLICENSE.COM procedure. When it prompts you for information, respond with data from your Product Authorization Key (PAK).
- Issue the DCL command "LICENSE REGISTER" with the appropriate qualifiers that correspond to information on the PAK.

For complete information on using LMF, see the manual on the License Management Utility in the VMS V5.0 documentation set.

4.5 VAXcluster Considerations

This layered product is fully supported when installed on any valid and licensed VAXcluster configuration. The software components of the product will be installed common to all accessing systems. Due to the fact, that this product is hardware dependent, the application programs accessing the DRX11-C/AAF01 have to run on the specific node, where the DRX11-C and the AAF01 interfaces are installed.

4.6 Installation Procedure Requirements

The following sections discuss various requirements for installing the AAF01/VMS Subroutine Library.

4.6.1 Time

The installation should take approximately between 5 to 40 minutes, depending on your type of media and your system configuration.

Installing the AAF01/VMS Subroutine Library and running the IVP on a standalone VAX 11/780 system takes approximately 10 minutes.

4.6.2 Privileges

To install the AAF01/VMS Subroutine Library, you have to be logged in to the SYSTEM account.

Note that VMSINSTAL turns off BYPASS privilege at the start of the installation.

4.6.3 Disk Space

Installing the AAF01/VMS Subroutine Library requires a certain amount of free storage disk space during the installation. Once the AAF01/VMS Subroutine Library is installed, less storage space is required. Table 4–1 summarizes the storage requirements for the AAF01/VMS Subroutine Library.

Table 4–1 Disk Space Requirements

Kit	Blocks During Installation	Blocks After Installation
AAF01/VMS Subroutine Library Version 2.0	850	550

To determine the number of free disk blocks on the current system disk, enter the following command at the DCL prompt:

```
$ SHOW DEVICE SYS$SYSDEVICE
```

4.6.4 System Parameters

The required system parameter values are the standard parameter values for the installation. Depending on the kinds of programs and applications running at your site, you might need some higher value settings.

4.6.5 Process Account Quotas

The account you use to install the AAF01/VMS Subroutine Library needs to have sufficient quotas enabling you to perform the installation. The required quotas for the installation are equal to those with the standard SYSTEM account.

4.6.6 VMSINSTAL Requirements

When you invoke VMSINSTAL, the following will be checked:

- Whether you have set your default device and directory to SYS$UPDATE
- Whether you are logged in to the SYSTEM account
- Whether you have adequate quotas for installation
- Whether DECnet is up and running
- Whether any users are logged in to the system

If VMSINSTAL detects any problems during the installation, it notifies you of the problem and asks if you want to continue the installation. In some instances, you can enter "YES" to continue. To stop the installation process and correct the situation, enter "NO" or press RETURN. Then solve the problem and restart the installation.

4.6.7 Backing Up Your System Disk

At the beginning of the installation, VMSINSTAL asks if you have backed up your system disk. DIGITAL recommends that you do a system disk backup before installing any software on top of the operating system.

Use the backup procedures that have been established at your site. For details on performing a system disk backup, see the section on the Backup Utility in the VMS documentation set.

4.6.8 License Management Utility Requirements

You will need to register and load your license for the AAF01/VMS Subroutine Library before you start the installation. Refer to the VMS documentation on the License Management Utility for information about registering your license.

For more information on the VMS License Management Facility, see the VMS documentation set.

5 INSTALLING THE AAF01/VMS SUBROUTINE LIBRARY

This chapter describes how to install the AAF01/VMS Subroutine Library. Section 5.2 contains a step-by-step description of the installation procedure.

5.1 General Information

This section includes information about accessing Release Notes, determining files added to your system, running the Installation Verification Procedure (IVP), and aborting the installation.

5.1.1 Accessing the Online Release Notes

The AAF01/VMS Subroutine Library provides online Release Notes. If you specify OPTIONS N when you invoke VMSINSTAL, the installation procedure asks you if you want to display or print the Release Notes. The Release Notes question comes near the beginning of the installation. You should review the Release Notes in every case.

After the AAF01/VMS Subroutine Library has been installed, the Release Notes are located in the following file:

```
SYS$HELP:AAF01$020.RELEASE_NOTES.
```

5.1.2 Determining the Directories/Files Added to the System

See Appendix B for the directories/files added to the system.

5.1.3 Running the Installation Verification Procedure (IVP)

The Installation Verification Procedure (IVP) for the AAF01/VMS Subroutine Library verifies the installation. During the installation, you are asked if you want to run the IVP as part of the installation. If you respond YES, VMSINSTAL runs the IVP. This is recommended to make sure that the AAF01/VMS Subroutine Library has been installed correctly.

After the AAF01/VMS Subroutine Library has been installed, you can run the IVP independently. You might also want to run the IVP after a system failure to make sure that the AAF01/VMS Subroutine Library remains executable.

```
$ @SYS$SYSROOT:[SYSTEST.AAF01]AAF01$IVP.COM
```

Note: Due to the fact, that the product is hardware dependent, the IVP have to run seperately only on that specific node, where the DRX11-C and the AAF01 interface are installed.

5.1.4 Aborting the Installation

To abort the installation procedure at any time, press CTRL/Y. When you press CTRL/Y, the installation procedure deletes all files it has created up to that point and exits. You can then start the installation again.

5.2 The Installation Procedure

The the AAF01/VMS Subroutine Library installation process consists of a series of questions and informational messages.

5.2.1 Invoking VMSINSTAL

To start the installation, invoke the VMSINSTAL command procedure from a privileged account, such as the SYSTEM account. Next, invoke VMSINSTAL. You use the following syntax to invoke VMSINSTAL:

```
@SYS$UPDATE:VMSINSTAL product-name device-name OPTIONS N
```

product-name

The installation name for the component. For the AAF01/VMS Subroutine Library, use the following installation name:

```
AAF01
```

device-name

The name of the device on which you plan to mount the media. For example, DUA2: is the device name for a disk drive. It is not necessary to use the console drive for this installation. However, if you do use the console drive, you should replace any media you removed once the installation is completed.

OPTIONS N

An optional parameter that indicates you want to be prompted to display or print the Release Notes. If you do not include the OPTIONS N parameter, VMSINSTAL does not prompt you to display or print the Release Notes. You are strongly recommended to read the Release Notes before proceeding with this installation.

VMSINSTAL has several other options; for information, see the VMS documentation on Software Installation.

The following example displays the command to invoke VMSINSTAL to install the AAF01/VMS Subroutine Library from disk drive DUA2: and the system response. This example uses the OPTIONS N parameter for printing or displaying Release Notes.

```
$ @SYS$UPDATE:VMSINSTAL AAF01 DUA2: OPTIONS N

      VAX/VMS Software Product Installation Procedure V5.0


It is 24-JUN-1988 at 10:19.
Enter a question mark (?) at any time for help.
```

If you do not supply either the product name or the device name, VMSINSTAL prompts you for this information later on in the installation procedure. VMSINSTAL does not prompt you for any options, so be sure to include OPTIONS N on the VMSINSTAL command line to access the Release Notes during the installation.

5.2.2 Installation Questions

This section discusses the questions that appear during the installation. See Appendix A for a sample installation procedure.

Each question in the installation is marked with an asterisk (*) at the beginning of the line. Some questions show the default response in brackets, for example [YES]. If you want to give the default response, press the RETURN key.

1 DECnet status

VMSINSTAL notifies you if DECnet is running and displays a list of all active processes. It then asks if you want to continue the installation. You should install the AAF01/VMS Subroutine Library on a standalone system or cluster with DECnet shut down. "NO" is the default response to the question.

```
%VMSINSTAL-W-DECNET, Your DECnet network is up and running.
   .
   .
   .
* Do you want to continue anyway [NO]?
```

2 System backup

VMSINSTAL asks if you are satisfied with your system backup. You should always back up your system disk before performing an installation. If you are satisfied with the backup of your system disk, press RETURN. Otherwise, enter "NO" to discontinue the installation. After you back up your system disk, you can restart the installation.

```
* Are you satisfied with the backup of your system disk [YES]?
```

3 Mounting the media

You should now mount the first distribution volume on the device you specified when you invoked VMSINSTAL. The device name appears in the line preceding the question. VMSINSTAL then asks you if you are ready to continue with the installation.

If you respond "YES" to indicate that you are ready, VMSINSTAL displays a message that the media containing the AAF01/VMS Subroutine Library has been mounted on the specified device and that the installation has begun. For example:

```
Please mount the first volume of the set on DUA2:.
* Are you ready? YES
%MOUNT-I-MOUNTED, AAF01 MOUNTED ON _$$DUA2:  (NODE 1)
The following products will be processed:
   AAF01 V2.0

    Beginning installation of the AAF01 V2.0 at 20:01

%VMSINSTAL-I-RESTORE, Restoring product saveset A...
```

If you entered the wrong device name when you invoked VMSINSTAL and need to restart the installation, enter NO in response to the ''Are you ready?'' question. To abort the installation for other reasons, press CTRL/Y.

4 Additional media volumes

If your installation kit contains more than one volume, VMSINSTAL prompts you to insert the additional volumes and then asks you to indicate that you are ready for the installation to proceed.

```
%BACKUP-I-READYREAD, mount volume 2 on _DUA2: for reading

Enter "YES" when ready:
```

5 Release Notes

If you specified OPTIONS N when you invoked VMSINSTAL, you are now given five options for reviewing the Release Notes.

```
Release Notes Options:

    1. Display Release Notes
    2. Print Release Notes
    3. Both 1 and 2
    4. Copy Release Notes to SYS$HELP
    5. Do not display, print or copy Release Notes

* Select option [2]:
```

If you select option 1, VMSINSTAL displays the Release Notes immediately on the console terminal. You can terminate the display at any time by pressing CTRL/C.

If you select option 2, VMSINSTAL prompts you for the name of the print queue that you want to use:

```
* Queue name [SYS$PRINT]:
```

You can enter a queue name or press RETURN to send the file to the default output print device.

If you select option 3, VMSINSTAL displays the Release Notes immediately on the console terminal and then prompts you for a queue name for the printed version.

6 Continuing the installation

The installation procedure now asks if you want to continue the installation. To continue, enter YES. Otherwise, press RETURN. In either case, the Release Notes are copied to a file in the SYS$HELP directory. For example:

```
* Do you want to continue the installation [N]?: YES
%VMSINSTAL-I-RELMOVED, The products Release Notes have been
successfully moved to SYS$HELP.
```

The Release Notes are located in the following file:.

```
SYS$HELP:AAF01$020.RELEASE_NOTES.
```

Note: The name of the Release Notes file installed by VMSINSTAL consists of the current product name and version number. Do not delete Release Notes for previous versions of the AAF01/VMS Subroutine Library.

7 License Registration

If you have already registered and loaded the PAK, the specific information for this particular license will be displayed:

```
Product:  AAF01-SUBROUTINE-LIBRARY
Producer: DEC
Version:  2.0
Release Date: 1-MAY-1988
```

The installation procedure asks if you have registered and loaded your authorization key for the AAF01/VMS Subroutine Library.

```
* Does this product have an authorization key registered and loaded?
```

If you have not registered and loaded your authorization key, you have to answer "NO" to this question. A successful installation of the AAF01/VMS Subroutine Library depends upon an authorization key being registered and loaded. Please be certain that this has been completed before proceeding with the installation.

8 Choosing to purge files

You have the option to purge files from previous versions of the AAF01/VMS Subroutine Library that are superseded by this installation. Purging is recommended; however, if you need to keep files from the previous version, enter "NO" in response to the question.

```
* Do you want to purge files replaced by this installation [YES]?
```

9 Choosing to run the Installation Verification Procedure (IVP)

The installation procedure now asks if you want to run the Installation Verification Procedure. The IVP for the AAF01/VMS Subroutine Library checks to be sure that the installation was succesful. It is recommended that you run the IVP. The loopback cable must be installed on the AAF01 interface to successfully run the IVP. If the cable is not installed then do not request execution of the IVP.

```
* Do you want to run the IVP after the installation [YES]?
```

5.2.3 Informational Messages

At this point, the installation procedure displays a number of informational messages that report on the progress of the installation. There are no further questions to answer. VMSINSTAL moves the files to their target directories, updates help files, and updates DCL tables, if necessary. The following message is displayed:

```
%VMSINSTAL-I-MOVEFILES, Files will now be moved to their target
directories...
```

5.2.4 Running the Installation Verification Procedure

If you chose to run the IVP, VMSINSTAL runs it now. When the IVP runs successfully, you see the following display:

```
%AAF01-I-SUCCESS, The IVP of the AAF01 V2.0 was successful...

 The AAF01/VMS Subroutine Library Installation Verification Procedure
 AAF01$IVP can be run again using the following command:

 $ @SYS$SYSROOT:[SYSTEST.AAF01]AAF01$IVP
```

5.2.5 Completing the Installation Procedure

The following messages indicate that the entire installation procedure has completed:

```
   Installation of AAF01 V2.0 completed at 10:49

   VMSINSTAL procedure done at 11:04
```

You can now log out of the privileged account:

```
$ LOGOUT
SYSTEM     logged out at 24-JUN-1988 11:05:00.00
```

Note that VMSINSTAL deletes or changes entries in the process symbol tables during the installation. Therefore, if you are going to continue using the system manager's account and you want to restore these symbols, you should log out and log in again.

5.3 Error Recovery

If errors occur during the installation itself or when the IVP is running, VMSINSTAL displays failure messages. If the installation fails, the following message is displayed:

```
%VMSINSTAL-E-INSFAIL, The installation of the AAF01 V2.0
has failed.
```

If the IVP fails, the following messages are displayed:

```
%VMSINSTAL-E-IVPFAIL, The IVP for the AAF01 V2.0 has failed.
```

Errors can occur during the installation if any of the following conditions exist:

- The operating system version is incorrect.
- Quotas necessary for successful installation are insufficient.
- System parameter values for successful installation are insufficient.
- The product is not properly licensed.
- The product is not supported on this processor.

For descriptions of the error messages generated by these conditions, see the VMS documentation on system messages, recovery procedures, and VMS software installation. If you are notified that any of these conditions exist, you should take the appropriate action as described in the message. (You might need to change a system parameter or increase an authorized quota value.) For information on installation requirements, see Chapter 1. The product specific installation errors, warnings and information messages are listed below:

1.	Error	-	%AAF01-E-BADVMSV1, This kit requires VMS Version V5.0 (or later) for correct installation.
		-	%AAF01-E-BADVMSV2, Please install VMS V5.0 (or later) before installing AAF01 V2.0.
	Solution	-	Operating System must have at least version V5.0 prior to the AAF01/VMS Subroutine Library installation.
2.	Error	-	%AAF01-E-NOSPACE, The system disk does not contain enough free blocks to install AAF01 V2.0
	Solution	-	See section 3.4 in this manual for disk space requirements.
3.	Error	-	%AAF01-E-NOLICENSE1, The AAF01 V2.0 installation requires a valid license.
		-	%AAF01-E-NOLICENSE2, Install the Product Authorization Key (PAK) and retry.
	Solution	-	Restart the installation after the license is registered and loaded.
4.	Error	-	%AAF01-E-NOLICENSE3, The AAF01 V2.0 installation requires a valid DRX11C-DRIVER license.
		-	%AAF01-E-NOLICENSE4, Install the Product Authorization Key (PAK) for DRX11C-DRIVER and retry.
	Solution	-	Restart the installation after the DRX11C-DRIVER license is registered and loaded.
5.	Error	-	%AAF01-E-NO_UUDRIVER, The DRX11-C/VMS Driver V6.0 (or later) must be present before AAF01 V2.0 can be installed.
	Solution	-	The current operating system does not have the UUDRIVER.EXE V6.0 (or later) in the SYS$LOADABLE_ IMAGES directory. Restart the installation after the prerequisite software DRX11-C/VMS Driver is installed successfully.
6.	Warning	-	%AAF01-W-NO_IVP The IVP will not be executed because the DRX11-C/VMS Driver is not loaded.

	Solution	-	Complete installation is performed only if the prerequisite software, the DRX11-C/VMS Driver is loaded. The installation continues, but the Installation Verification Procedure (IVP) will not be started. In this case, after loading (installing) the driver, it is absolutely necessary to run IVP and some of the distributed demonstration programs.
7.	Warning	-	%AAF01-W-NO_FOR, AAF01/VMS FORTRAN demo programs not built because of missing FORTRAN V4.8 (or later) Compiler.
		-	%AAF01-W-NO_TLB, AAF01/VMS FORTRAN demo programs not build because of missing FORSYSDEF.TLB.
	Solution	-	The current operating system does not have the FORTRAN.EXE V4.8 (or later) in the SYS$SYSTEM directory or the FORSYSDEF.TLB in the SYS$LIBRARY directory. The installation continues, the prebuild version of FORTRAN demo programs will be copied from the distribution kit to the appropriate directory.
8.	Warning	-	%AAF01-W-NO_FILE AAF01 FORTRAN demo programs not build because of missing DRX11C$DEFINC.FOR
	Solution	-	The current operating system has the prerequisite software DRX11-C/VMS Driver installed, but the include file DRX11C$DEFINC.FOR is not present in the SYS$SYSROOT:[SYSHLP.EXAMPLES.DRX11C] directory. The installation continues, the prebuild version of FORTRAN demo programs will be copied from the distribution kit to the appropriate directory.
9.	Information	-	There are some information messages printed by VMSINSTAL about the status of the installation (what is currently being done), about new directories created, how to start the successfully built example programs, which directories were used and some information for the system manager.
	Solution	-	No immediate action is needed on information messages.

6 AFTER INSTALLING THE AAF01/VMS SUBROUTINE LIBRARY

After installing the AAF01/VMS Subroutine Library, you need not to perform any tasks.

6.1 Cluster Considerations

You can use the product on any other nodes, where a DRE11-C/DRQ11-C and the AAF01 interface is installed, in the VAXcluster. The application programs accessing the DRX11-C/AAF01 interfaces have to run on the specific node where the required hardware is installed.

6.2 Tailoring Your System

This section provides information about special system arrangements and cleanup procedures that you can perform after installing the AAF01/VMS Subroutine Library. All source files and example programs located in SYS$SYSROOT:[SYSHLP.EXAMPLES.AAF01] are independent of the AAF01/VMS Subroutine Library, except the AAF01$CWT.COM, AAF01$LINK.COM, AAF01$DEF.MAR and AAF01$PARINC.FOR. If it is needed to recover disk space, these files can be deleted without any effect upon the use of the product.

6.3 Determining and Reporting Problems

If an error occurs while the AAF01/VMS Subroutine Library is being used and you believe that the error is caused by a problem with the AAF01/VMS Subroutine Library, take one of the following actions:

- If you have a BASIC or DECsupport Software Agreement, you should call your Customer Support Center. (With these services, you receive telephone support that provides high-level advisory and remedial assistance.)
- If you have a Self-Maintenance Software Agreement, you can submit a Software Performance Report (SPR).
- If you purchased the AAF01/VMS Subroutine Library within the past 90 days and you think the problem is caused by a software error, you can submit a Software Performance Report (SPR).

When you prepare to submit an SPR, please do the following:

1. Provide a description of the problem as complete as possible, trying not to overlook any details. Include in the description the version number of the AAF01/VMS Subroutine Library, the current VMS system and the CPU type.
2. Reduce the problem to a size as small as possible, preferably to a program of 200 source lines or less, including user INCLUDE files.

3 Submit program sources on machine-readable media (floppy diskette or streamer tape), if possible. All media will be returned when the SPR is answered.

4 Provide a listing of the program using AAF01/VMS Subroutine Library, compiled with the command switches: /LIST/CROSS_ REFERENCE/SHOW=ALL

5 Remember to include listings of any command files, linker maps, relevant data files, and so on.

If you find an error in the the AAF01/VMS Subroutine Library documentation, you should fill out and submit the Reader's Comments form at the back of the document in which the error was found. Include the section and page number where the error was found.

A SAMPLE INSTALLATION

This appendix contains a sample installation of the AAF01/VMS Subroutine Library. This sample was run on a system that had no previous version of the AAF01/VMS Subroutine Library installed. Depending on which layered products you have on your system, you might see additional messages and questions when you perform your installation.

The sample installation assumes that DECnet has been shut down, that no users are logged on to your system, and that OPTIONS N is specified to print the Release Notes. The Installation Verification Procedure (IVP) runs at the end of the installation.

```
        Welcome to VAX/VMS V5.0

Username: SYSTEM
Password:
        Welcome to VAX/VMS version V5.0 on node CSSSE2
    Last interactive login on Tuesday, 28-JUN-1988 11:19

$ @sys$update:vmsinstal aaf01 dua1: options n

        VAX/VMS Software Product Installation Procedure V5.0


It is 28-JUN-1988 at 11:20.
Enter a question mark (?) at any time for help.
* Are you satisfied with the backup of your system disk [YES]? y <CR>

Please mount the first volume of the set on DUA1:.
* Are you ready? yes
%MOUNT-I-MOUNTED, AAF01      mounted on _DUA1:
The following products will be processed:

  AAF01 V2.0

        Beginning installation of AAF01 V2.0 at 11:20

%VMSINSTAL-I-RESTORE, Restoring product saveset A ...

    Release Notes Options:

1.   Display release notes
2.   Print release notes
3.   Both 1 and 2
4.   Copy release notes to SYS$HELP
5.   Do not display, print or copy release notes

* Select option [2]: 4

%VMSINSTAL-I-RELMOVED , The product's release notes have been successfully
 moved to SYS$HELP.
* Do you want to continue the installation? yes
        Product:      AAF01-SUBROUTINE_LIBRARY
        Product:      DEC
        Version:      2.0
        Release Date: 1-MAY-1988
* Does this product have an authorization key registered and loaded? yes
* Do you want to purge files replaced by this installation [YES]? ? <CR>
```

```
  During this installation, new files will be provided to replace exist-
  ing versions. You may purge these older versions to save disk space,
  or keep them if you feel they may be of use. Purging is recommended.

* Do you want to purge files replaced by this installation [YES]? Y <CR>

        The test connector must be installed on the AAF01 interface
        to successfully run the IVP. If the test connector is not
        installed then do not request execution of the IVP.

* Do you want to run the IVP after the installation [YES]? ? <CR>

        The  Installation Verification Procedure (IVP)  verifies the
        installation of the product.  It is recommended to  run  the
        IVP to ensure that the  AAF01  Subroutine Library is working
        correctly.

* Do you want to run the IVP after the installation [YES]? Y <CR>

        The installation will now proceed with no further questions.
        A few more messages will be printed that contain information
        about the progress of the installation; the text of this
        information is duplicated in the installation section of the
        AAF01/VMS Subroutine Library User's and Installation Guide.

        If no errors occur, the installation will be completed in
        5 to 40 minutes depending on your configuration.

%VMSINSTAL-I-RESTORE, Restoring product saveset B ...
%AAF01-I-BUILDING, Building the AAF01/VMS FORTRAN demo programs ...

        The AAF01/VMS FORTRAN demo programs can be run using a
        DCL command like:

        $ RUN SYS$SYSROOT:[SYSHLP.EXAMPLES.AAF01]AAF01$SNGL

     This installation creates the following directories/files:

     SYS$HELP:   AAF01$020.RELEASE_NOTES ; Release notes file
     SYS$LIBRARY:AAF01$LIB.OLB           ; AAF01/VMS Subroutine Library
     SYS$SYSROOT:[SYSHLP.EXAMPLES.AAF01] ;
                 AAF01$CWT.COM           ; Control Word Table generation
                 AAF01$CONTROL.MAR       ;    indirect command and MACRO
                                         ;    file
                 AAF01$T3.EXE            ; AAF01/VMS FORTRAN test program
                 AAF01$T3.FOR            ;    source and image files
                 AAF01$T24.EXE           ; AAF01/VMS FORTRAN test program
                 AAF01$T24.FOR           ;    source and image files
                 AAF01$T25.EXE           ; AAF01/VMS FORTRAN test program
                 AAF01$T25.FOR           ;    source and image files
                 AAF01$DOBL.EXE          ; AAF01/VMS FORTRAN demo program
                 AAF01$DOBL.FOR          ;    source and image files
                 AAF01$SNGL.FOR          ; Installation Verification
                 AAF01$DOU.FOR           ;    Program (IVP) source
                 AAF01$BIG.FOR           ;       files
                 AAF01$LINK.COM          ; File to link a user program
                 AAF01$DEF.MAR           ; AAF01/VMS MACRO definitions
                 AAF01$PARINC.FOR        ; AAF01/VMS FORTRAN definitions
                 DRX11C$DEFINC.FOR       ; DRX11-C/VMS FORTRAN function
                                         ;    codes
     SYS$SYSROOT:[SYSTEST.AAF01]         ;
                 AAF01$IVP.COM           ; Command procedure to run IVP
                 AAF01$SNGL.EXE          ; Installation Verification
                 AAF01$DOU.EXE           ;    Procedure (IVP) image
                 AAF01$BIG.EXE           ;       files
                 AAF01$DIAG.EXE          ; AAF01/VMS ONLINE DIAGNOSTIC


%VMSINSTAL-I-SYSDIR, This product creates system disk directory
 VMI$ROOT:[SYSHLP.EXAMPLES.AAF01].
%VMSINSTAL-I-SYSDIR, This product creates system disk directory
 VMI$ROOT:[SYSTEST.AAF01].
%VMSINSTAL-I-MOVEFILES, Files will now be moved to their target
 directories...
%AAF01-I-RUNNING, Running the Installation Verification Procedure...
```

```
$ !
$ !     AAF01$IVP.COM  27-jun-1988
$ !
$ !     Installation Verification Procedure for AAF01/VMS
$ !     Subroutine Library using the programs AAF01$SNGL, AAF01$DOU
$ !     and AAF01$BIG.
$ !
$ RUN SYS$SYSROOT:[SYSTEST.AAF01]AAF01$SNGL

***  Program example for single buffer mode for the AAF01  ***

AAF01 Subroutine Library, Version  2.0

Copyright (C) 1984

Digital Equipment GmbH, CSS-SW, Munich, Germany


 Device name e.g.: _UUA0:
UUA0:
DRE11-C status =   200   2020  26030  166170  54450  166170     0  100170
Conversion time as number of 100 nsec steps
     (range: 25 ... 4095, default: 100)               ?:
100

AAF01$SNGL: Operation mode on AAF01 at UUA0:    is bipolar

First value of volts to be output
     (range: -10.0000 ... 9.9951, default: -10.0000)  ?:
-10.0
Second value of volts to be output
     (range: -10.0000 ... 9.9951, default:   9.9951)  ?:
9.9

AAF01$SNGL: Return status  = (decimal)      1
            Buffer pointer = (decimal)      1
            Byte count = (decimal)      64000


AAF01$SNGL
$ !
$ RUN SYS$SYSROOT:[SYSTEST.AAF01]AAF01$DOU
Device name [Default : _UUA0:  ]
UUA0:

 Buffered data path, yes=1, no=0
0
DRE11-C status =   300     0   26004  101400   54450  11610     0     0
 ENTER NUMBER OF CONVERSIONS, THE NUMBER MUST BE
 A MULTIPLE OF 2,  DEF=50000
25000
 Burst mode yes=1, no=0
0
 Control table transfer, yes=1, no=0
0
 ENTER AAF01 THROUGHPUT TIME IN MULTIPLES OF 100 NANOSECONDS
 25 TO 4095   (E.G. 100 = 10 MICROSECONDS) D:50
100

DMA speed =    100.000 kHz

 NUMBER OF BUFFERS TO TRANSFER DEF=10
10

AAF01$DOBL: Return status  = (decimal)   2104
            Buffer pointer = (decimal)      1
            Byte count = (decimal)       1896
%SYSTEM-W-DATAOVERUN, data overrun

DMA speed =    100.000 kHz
```

SAMPLE INSTALLATION

```
$ !
$ RUN SYS$SYSROOT:[SYSTEST.AAF01]AAF01$BIG
 Device name e.g.: _UUA0:
UUA0:
 Enter size of large buffer in bytes ', def=1000000
 Note that large buffer must be an even multiple of sub-buffer
60000
 Enter size of sub-buffer in bytes, D:100000 (max 125442)
10000
 Burst mode yes=1, no=0
0
 Enter AAF01 throughput time in multiples of 100 nanoseconds
 25 TO 4095   (e.g. 100 = 10 microseconds) D:100
100

DMA speed =    100.000 kHz

 Number of sub-buffers to transfer def=           6
6
     1      2 Bytes:       60000
%SYSTEM-S-NORMAL, normal successful completion

DMA speed =    100.000 kHz

$ VERIFY = F$VERIFY(VERIFY)

%AAF01-I-SUCCESS, The IVP of AAF01 V2.0 was successful...

        The AAF01/VMS Installation Verification Procedure AAF01$IVP
        can be run again using the following command:

        $ @SYS$SYSROOT:[SYSTEST.AAF01]AAF01$IVP

        Installation of AAF01 V2.0 completed at 12:00

        VMSINSTAL procedure done at 12:01
```

B DIRECTORIES/FILES INSTALLED ON YOUR SYSTEM

The AAF01/VMS Subroutine Library installation procedure installs a number of files on your system. This installation creates the following directories/files:

SYS$SYSROOT:[SYSTEST.AAF01]:

AAF01$IVP.COM	Command file to start IVP
AAF01$SNGL.EXE	Installation Verification
AAF01$DOU.EXE	Procedure (IVP) image
AAF01$BIG.EXE	files
AAF01$DIAG.EXE	AAF01/VMS online diagnostic

SYS$HELP:

AAF01$020.RELEASE_NOTES

SYS$LIBRARY:

AAF01$LIB.OLB	AAF01/VMS Subroutine Library

SYS$SYSROOT:[SYSHLP.EXAMPLES.AAF01]:

AAF01$CWT.COM	Control Word Table generation
AAF01$CONTROL.MAR	indirect command and MACRO file
AAF01$T3.EXE	AAF01/VMS Fortran test program
AAF01$T3.FOR	source and image files
AAF01$T24.EXE	
AAF01$T24.FOR	
AAF01$T25.EXE	
AAF01$T25.FOR	
AAF01$DOBL.EXE	AAF01/VMS Fortran demo program
AAF01$DOBL.FOR	source and image files
AAF01$SNGL.FOR	Installation Verification
AAF01$DOU.FOR	Program (IVP) source files
AAF01$BIG.FOR	
AAF01$LINK.COM	File to link a user program
AAF01$DEF.MAR	AAF01/VMS MACRO definitions
AAF01$PARINC.FOR	AAF01/VMS FORTRAN definitions

C EXAMPLE PROGRAMS

The AAF01 Subroutine Library provides also some example programs which are intended to be used for demonstration of the capabilities of the AAF01. One example program demonstrates how to program the AAF01 for use in single buffer mode, the other one how to do it for alternate buffer mode. Both programs refer to an attention AST routine in order to recognize unsolicited interrupts.

C.1 AAF01$SNGL (Single Buffer Mode)

This FORTRAN program demonstrates how to program the AAF01 for use in single buffer mode. The program source code is provided thereafter, together with a functional description of the program.

Please note that this program AAF01$SNGL.FOR uses a Control Table which is set up by the program itself, thus without using the indirect command procedure AAF01$CWT.COM for its generation. The Control Table contains the channel addresses 0 through 15, and Control Word mode 0 for all but the last channel (i.e. channel 15), the mode for that one being 1.

The first voltage value will be output repeatedly to the even numbered channels, i.e. channels 0, 2, ..., 14. The second voltage value will be output repeatedly to the odd numbered channels, i.e. channels 1, 3, ..., 15.

Compiling and linking for this example program AAF01$SNGL is done in the following way .

```
$FORT AAF01$SNGL

$LINK AAF01$SNGL,AAF01$LIB/LIB
```

Please note that this example program AAF01$SNGL will be compiled and linked when installing the AAF01 Subroutine Library.

A sample run of this example program AAF01$SNGL follows.

EXAMPLE PROGRAMS

```
$RUN AAF01$SNGL

***  Program example for single buffer mode for the AAF01  ***

AAF01 Subroutine Library, Version  2.0

Copyright (C) 1985

Digital Equipment GmbH, CSS-SW, Munich, Germany

 Device name e.g.: _UUA0:
 UUA0:

 AAF01 status = 200    0   50010  101400   50004  1014000   0    0

Conversion time as number of 100 nsec steps
     (range: 25 ... 4095, default: 100)              ?:

AAF01$SNGL: Operation mode on AAF01 at UUA0: is bipolar

First value of volts to be output
     (range: -10.0000 ... 9.9951, default: -10.0000)  ?:
Second value of volts to be output
     (range: -10.0000 ... 9.9951, default:   9.9951)  ?:

AAF01$SNGL: Return status = (octal)         1
            Buffermark    = (decimal)       1
            Byte count    = (decimal)   12800


AAF01$SNGL

$
```

C.2 AAF01$DOBL (Alternate Buffer Mode)

This FORTRAN program demonstrates how to program the AAF01 for use in alternate buffer mode. The program source code is provided thereafter, together with a functional description of the program.

Please note that this program AAF01$DOBL uses a Control Table which is set up by the program itself, thus without using the indirect command procedure AAF01$CWT.COM for its generation. The Control Table contains the channel addresses 0 through 15, and Control Word mode 0 for all but the last channel (i.e. channel 15), the mode for that one being 1.

The first voltage value will be output repeatedly to the even numbered channels, i.e. channels 0, 2, ..., 14. The second voltage value will be output repeatedly to the odd numbered channels, i.e. channels 1, 3, ..., 15.

Compiling and linking for this example program AAF01$DOBL is done in the following way

```
$FORT AAF01$DOBL

$LINK AAF01$DOBL,AAF01$LIB/LIB
```

Please note that this example program AAF01$DOBL will be compiled and linked when installing the AAF01 Subroutine Library.

A sample run of this example program AAF01$DOBL follows.

```
$RUN AAF01$DOBL


***  Program example for alternate buffer mode for the AAF01  ***

AAF01 Subroutine Library, Version  2.0

Copyright (C) 1985

Digital Equipment GmbH, CSS-SW, Munich, Germany

 Device name e.g.: _UUA0:
 UUA0:

 AAF01 status = 200    0   50010  101400   50004  1014000   0    0

Conversion time as number of 100 nsec steps
     (range: 25 ... 4095, default: 100)               ?:

AAF01$DOBL: Operation mode on AAF01 at UUA0: is bipolar

First value of volts to be output
     (range: -10.0000 ... 9.9951, default: -10.0000)  ?:
Second value of volts to be output
     (range: -10.0000 ... 9.9951, default:   9.9951)  ?:


Number of conversions
     (range: 2 ... 32767, default: 20)              ?:

AAF01$DOBL: Number of transfers done is   1000

AAF01$DOBL: Return status = (octal)         1
            Buffermark    = (decimal)       1
            Byte count = (decimal)       6400


AAF01$DOBL

$
```

D AAF01 EXERCISER (Online Diagnostic)

The AAF01 Subroutine Library provides a test program which is intended to be used for checkout of the AAF01. This exerciser, written in VAX FORTRAN, runs under the VAX/VMS operating system. Before it can be started, the AAF01/VMS Subroutine Library must be installed. A further prerequisite for running the program is that the user must have the LOG_ IO privilege.

The AAF01 exerciser is started by typing the command RUN, followed by a space and the name of the test program, which is AAF01$DIAG.

After starting the program, the user is required to enter the desired AAF01 controller identification letter, e.g. A for the device UUA0:, B for UUB0: and so on. If the device connected to the AAF01 is not an AAF01, or if it is not recognizable as such (e.g. because of wrongly connected cables), the following error messages will appear:

```
Error #9999: This is either not an AAF01, or it is defective.

End of AAF01 Exerciser
```

and the program will exit.

Bit 11 of the ACS register is read and a message is displayed to show whether it indicates unipolar or bipolar output mode e.g.:

```
AAF01 is switched to unipolar operation.
```

The program AAF01$DIAG consists of four separate tests: a logic test, a linearity test, a calibration test and a selectable output test. Each of these tests can be selected separately, or they can be run consecutively in the listed order. The following prompt will be displayed:

```
Test numbers are 0, 1, 2, 3 and 4.
0 selects all tests in sequential order
1 selects the AAF01 logic test
2 selects the AAF01 linearity test
3 selects the AAF01 calibration test
4 selects the AAF01 selectable output test
Enter test number, default is 0 (ie. all tests):
```

The tests are described in detail below.

D.1 AAF01 LOGIC TEST

The AAF01 logic test consists of 21 separate subtests to make sure that the module functions normally. The test connector 2G-M00FA must be installed. The following text will appear:

```
This is the AAF01 logic test.

To run this test, the test connector must be installed.
Use 2G-M00FA on module A6009.
Exerciser will continue only, if the next answer is "Y".
Is the test connector installed (Y/N)? [Y]:

Conversion times are expressed in multiples of 100 nanoseconds and
range from 25 to 4095.
Enter conversion time, default is 200 (ie. 20 microseconds):

The AAF01 logic test will start now. It wil take up to 5 minutes.
```

All subtests begin with a AAF01 reset to put the module into a known state prior to testing. Some subtests are executed more than once.

The 21 subtests that make up the logic test are listed below, with their associated error messages. In addition to these messages, there may also be some VAX/VMS system messages. For an explanation of these, consult the AAF01/VMS Subroutine Library User's and Installation Guide and/or the appropriate VMS documentation.

Note: All register error messages (e.g. ACS) are supplemented by the following information (expressed in octal values):
Expected: xxxx Was: yyyy

Test 1: AAF01 reset test

This test checks that the AAF01 ACS register is correctly set or reset after a AAF01 reset and after a data system reset.

The following errors may be output:

```
Error #10: ACS register incorrect after AAF01 reset

Error #11: ACS register incorrect after AXF01 reset

AAF01 reset test failed
```

Test 2: AAF01 ACS read/write test

This test checks that the read/write bits of the ACS register can all be set, all cleared and individually set.

The following errors may be printed:

```
Error #20: ACS read/write bits could not be set

Error #21: ACS read/write bits could not be cleared

Error #22: ACS read/write bits could not be individually set

AAF01 ACS read/write test failed
```

Test 3: AAF01 CTA read/write test

This test checks that the read/write bits of the CTA register can all be set, all cleared and individually set. The following errors may be printed:

```
Error #30: CTA read/write bits could not be set
Error #31: CTA read/write bits could not be cleared
Error #32: CTA read/write bits could not be individually set
AAF01 CTA read/write test failed
```

Test 4: AAF01 control table data test

This test checks that bits 0 to 8 of each control word can all be set, all cleared, and individually set. The following errors may be printed:

```
Error #40: Control word bits could not be set
Error #41: Control word bits could not be cleared
Error #42: Control word bits could not be individually set
AAF01 control table data test failed
```

Test 5: AAF01 control table address test

This checks for interaction between different words of the control table.

To check the address logic bits 0 to 8, all 1024 control words are loaded with a modulo 511 count. The entire control table is then read and the contents of each word compared with the data which was written to it.

To check bit 9 of the address logic, the first half of the control table is loaded with all zeroes and the second half with all ones. The table is again read to check that the data was correctly written.

The following errors may be printed:

```
Error #50: Control table addressing error, bit 0-8
Error #51: Control table addressing error, bit 9
AAF01 control table address test failed
```

Test 6: AAF01 COUT and SEQ CONT L interrupt test

This checks that the command output (COUT) signal can be set and that the SEQ CONT L input signal can be used to generate an interrupt.

The interrupt enable bit (6) of the AAF01 SCR register is set and in the AAF01 ACS register SBE is set. COUT is then set and cleared. COUT is looped back to the SEQ CONT L input signal via the test connector, and the trailing edge sets SBR in the ACS register. This in turn sets EOC and STAT0 in the AAF01 SCR causing an interrupt.

The following errors may be printed:

```
Error #60: Interrupt timeout after COUT loopback

Error #61: ACS register incorrect after COUT loopback

Error #62: AAF01 SCR register incorrect after COUT loopback

AAF01 COUT and SEQ CONT L interrupt test failed
```

Test 7: AAF01 DMA to registers test

This test performs a DMA write to the ACS and CTA registers. The register contents are then read back under program control and checked.

The following errors may be printed:

```
Error #70: No interrupt after DMA to registers

Error #71: ACS register incorrect after DMA write

Error #72: CTA register incorrect after DMA write

AAF01 DMA to registers test failed
```

Test 8: AAF01 DMA to control table test

This test performs a single block DMA to all 1024 words of the control table. The control table contents are then read back under program control and checked.

The following errors may be printed:

```
Error #80: No interrupt after DMA to control table

Error #81: Control table word incorrect after DMA
           Address: ....   Expected: xxxx   Was: yyyy

AAF01 DMA to control table test failed
```

Test 9: AAF01 CWR mode 0 test

This tests that after each conversion in mode 0, the CTA register is incremented. Conversions are initiated; after they are completed, the CTA register is checked to ensure that it was correctly incremented.

The following errors may be output:

```
Error #90: DRX11-C SCR register incorrect after mode 0 conversions
Error #91: ACS register incorrect after mode 0 conversions
Error #92: CTA register incorrect after mode 0 conversions
AAF01 CWR mode 0 test failed
```

Test 10: AAF01 CWR mode 1 test

This tests that after a conversion sequence in mode 1, the CTA register is reset to 0. Conversions are initiated; after they are completed, the CTA register is checked to ensure that it was correctly reset.

The following errors may be output:

```
Error #100: DRX11-C SCR register incorrect after mode 1 conversions
Error #101: ACS register incorrect after mode 1 conversions
Error #102: CTA register incorrect after mode 1 conversions
AAF01 CWR mode 1 test failed
```

Test 11: AAF01 CWR mode 2 test

This tests that dummy control table loads can be made by using mode 2. The SCR interrupt enable bit is set and the DMA is started. Conversions are then initiated and the program waits for them to complete.

The following errors may be printed:

```
Error #110: DRX11-C SCR register incorrect after mode 2 conversions
Error #111: ACS register incorrect after mode 2 conversions
Error #112: CTA register incorrect after mode 2 conversions
AAF01 CWR mode 2 test failed
```

Test 12: AAF01 CWR mode 3, RATE CLOCK OUT and SEQ CONT test

This tests that the SEQ CONT signal can initiate and reinitiate conversions in mode 3. The SCR interrupt enable bit is set and the DMA is started. The RATE CLK OUT signal is connected to the SEQ CONT input via the test connector and so triggers the conversions. The CTA register is checked to ensure that this occurred.

The following errors may be printed:

```
Error #120: DRX11-C SCR register incorrect after SEQ CONT conversions
Error #121: ACS register incorrect after SEQ CONT conversions
Error #122: CTA register incorrect after SEQ CONT conversions
AAF01 CWR mode 3, RATE CLOCK OUT and SEQ CONT test failed
```

Test 13: AAF01 programmable clock test

This tests the speed and count functionality of the programmable clock. The PCR is loaded with the value 4000 to give a conversion time of 400 microseconds. The control table is set up to perform 2048 conversions at this rate and a check is made that the conversions require between 0.8 and 1.3 seconds to complete.

The conversions are made in mode 0 (except last CTA word, using mode 1).

The following errors may be output:

```
Error #130: Conversion is too fast in clock test
Error #131: Conversion is too slow in clock test
Error #132: ACS register incorrect in clock test
Error #133: CTA register incorrect in clock test
AAF01 Programmable clock test failed
```

Test 14: AAF01 CWR mode 4 test

This tests that dummy control table loads can be made by using mode 4. The SCR interrupt enable bit is set and the DMA is started. Conversions are then initiated and the program waits for them to complete.

The following errors may be printed:

```
Error #140: DRX11-C SCR register incorrect after mode 4 conversions
Error #141: ACS register incorrect after mode 4 conversions
Error #142: CTA register incorrect after mode 4 conversions
AAF01 CWR mode 4 test failed
```

Test 15: AAF01 CWR mode 5 test

This tests that dummy control table loads can be made by using mode 5. The SCR interrupt enable bit is set and the DMA is started. Conversions are then initiated and the program waits for them to complete.

The following errors may be printed:

```
Error #150: DRX11-C SCR register incorrect after mode 5 conversions
Error #151: ACS register incorrect after mode 5 conversions
Error #152: CTA register incorrect after mode 5 conversions
AAF01 CWR mode 5 test failed
```

Test 16: AAF01 CWR mode 6 test

This tests that dummy control table loads can be made by using mode 6. The PCR is loaded with a conversion time of 400 microseconds. The control table is then set up to perform 1024 conversions. All except the first and last conversions are made in mode 6. The first and last are made in mode 0 and mode 1.

The SCR interrupt enable bit is set and a DMA is started. Conversions are then initiated and the program checks that the conversions require between 0.3 and 0.5 seconds to complete.

The following errors may be printed:

```
Error #160: Conversions too fast in CWR mode 6
Error #161: Conversions too slow in CWR mode 6
Error #162: ACS register incorrect after mode 6 DMA output
Error #163: CTA register incorrect after mode 6 DMA output
AAF01 CWR mode 6 test failed
```

Test 17: AAF01 CWR mode 7 test

This tests that dummy control table loads can be made by using mode 7. The SCR interrupt enable and ACS SBE bits are set and the DMA is started. Conversions are then initiated. As the second conversion completes, COUT goes low, causing the input SEQ CONT L to go low. The program checks that an SBR interrupt then occurs and that conversions stop with the CTA register pointing to the desired control word.

The following errors may be printed:

```
Error #170: DRX11-C SCR register incorrect after mode 7 conversions
Error #171: ACS register incorrect after mode 7 conversions
Error #172: CTA register incorrect after mode 7 conversions
AAF01 CWR mode 7 test failed
```

Test 18: AAF01 data buffer empty test

This tests that an interrupt can be generated when the AAF01 data buffer becomes empty.

First the ACS external clock enable bit is set to prevent conversion. The control table is then set up for conversions in mode 0.

The DMA is then initiated. DRX11-C COR maintenance and run bits are set and the SCR wait bit is 64 times cleared to get the FIFO fully loaded. The last data word is also written into the ACS to allow conversions. A check is made that the DBE bit causes an interrupt and that the CTA register indicates the correct number of conversions took place.

The following errors may be printed:

```
Error #180: No data buffer empty interrupt
Error #181: DRX11-C SCR register incorrect after data buffer empty test
Error #182: ACS register incorrect after data buffer empty test
Error #183: CTA register incorrect after data buffer empty test
AAF01 data buffer empty test failed
```

Test 19: AAF01 external clock test

This checks that the RATE CLOCK IN/OUT line can be used to trigger conversions. The external clock enable and maintenance bits are set and a DMA is enabled. The program then waits for the data to be loaded into the FIFO, after which COUT is set and cleared the required number of times. This signal is looped back to the RATE CLOCK IN/OUT line via the test connector. After the last pulse a check is made that an end of conversion interrupt occurred and that the SCR and ACS registers are correct.

The following errors may be printed:

```
Error #190: No interrupt after external clock test
Error #191: ACS register incorrect after external clock test
AAF01 external clock test failed
```

Test 20: AAF01 DMA output comparison test

This checks that conversions can be made under DMA control. The control table is set up to perform 2 outputs, on channels 0 and 15. An internal buffer is set up to contain the data 1777 and 3777. These are selected so that the ACS CMP bit will be set. The AAF01 is then set up for a 2 word DMA and conversions are initiated by setting the GO bit of the ACS register. After the DMA has completed, the ACS register CMP bit is checked to be set. The test is then repeated but with the data words interchanged so that the ACS CMP bit is not set.

The following errors may be printed:

```
Error #200: DRX11-C SCR register incorrect after DMA output comparison

Error #201: ACS register incorrect after DMA output comparison

AAF01 DMA output comparison test failed
```

Test 21: AAF01 memory transfer conversion test

This tests DMA conversions in memory transfer mode. At the end of the conversions, the ACS CMP bit will be set.

The AAF01 is set up to start a DMA. The GO and MET bits of the ACS register are then set. After the DMA has completed, the ACS register MET and CMP bits are checked to be set.

The check is then repeated but differently, so that after the conversions the ACS CMP bit is not set.

The following errors may be printed:

```
Error#210: DRX11-C SCR register incorrect after memory mode DMA conversion

Error#211: ACS register incorrect after memory mode DMA conversion

AAF01 memory transfer conversion test failed
```

If one or more of the subtests fail, the following error messages are also displayed at the end of the logic test:

```
AAF01 logic test failed

AAF01 Exerciser failed
```

In some circumstances, the next error message can appear:

```
Error starting conversion.
Expected number of conversions: xxxx  Was: yyyy
```

This message is always displayed in addition to a numbered message, to provide supplemental information.

If a small conversion time is selected and any errors appear, the following remark is displayed:

```
NOTE: Since the selected conversion time is smaller than
      60 steps (6 microseconds), the AAF01 logic test error(s)
      probably indicate(s) only that the AAF01 cannot supply
      the data quickly enough.
      You can increase the conversion time until this test
      runs without error; that conversion time will give the
      maximum throughput of the system for this test.
```

D.2 AAF01 LINEARITY TEST

This test checks the linearity of the AAF01.

First the operator will be asked if the test connector is installed. After that, the next question will appear:

```
The +/- tolerance of the conversion accuracy ranges from 0 to 4095.
An error occurs if the deviation from the desired value is greater.
Enter +/- tolerance, default is 7:
```

Then each possible value is output to channel 15. For each of these outputs, two outputs are made on channel 0, the first using a value "tolerance" greater than that on channel 15, and the second, "tolerance" lower. A check is made that the first output on channel 0 results in a logical 0 from the comparator logic, and that the second results in a 1.

All outputs are made by program using the DBF register.

At the range boundaries, only one comparison with channel 0 is made.

The following errors may be printed:

```
Error#220: ACS register incorrect after DBF output comparison at octal
           output value xxxx on channel 15 and yyyy on channel 0.

           AAF01 linearity test failed
```

If an error message occurs, the following text is also displayed:

```
AAF01 Exerciser failed
```

D.3 AAF01 CALIBRATION TEST

This test allows the AAF01 analogue output to be calibrated. The user will be asked if a digital voltmeter is connected, then for the channels to be tested on the following way:

```
Select the channel numbers which should be tested (0 - 15).
Enter first channel, default is 0:
Enter last channel, default is 15:
```

These allow the user to concentrate testing on a single channel or on a group of channels. By default, 16 channels will be tested. If the last channel number is smaller than that of the first channel, it will be used as the first channel instead.

Calibrating for BIPOLAR mode:

In bipolar mode each of the selected channel is loaded with the value 7777 and the user is asked to check that the DVM reads -10.000 volts +/- 4.88 millivolts on the selected channel(s). If the output is not within this tolerance, the user is asked to adjust the offset potentiometer RV1 until the outputs are correct.

Each channel is then loaded with the value 0000 and the user asked to check that the DVM now reads 9.9951 volts +/- 4.88 millivolts. If the output is not within this tolerance, he is asked to adjust the gain potentiometer RV2 until the output is correct.

Calibrating for UNIPOLAR mode:

In unipolar mode, each of the channels selected is loaded with the value 7777 and the user is asked to check that the DVM reads 0.0000 volts +/- 4.88 millivolts on the selected channel(s). If the output is not within this tolerance, the user is asked to adjust the offset potentiometer RV1 until outputs are correct.

Each channel is then loaded with the value 0000 and the user asked to check that the DVM now reads 9.9976 volts +/- 4.88 millivolts. If the output is not within this tolerance, he is asked to adjust the gain potentiometer RV2 until the output is correct.

All outputs are made directly through the DBF register.

There are no error messages for this test.

D.4 AAF01 SELECTABLE OUTPUT TEST

This test allows the user to select output patterns for oscilloscope monitoring.

The test begins with the questions about the conversion time and the channels to be used, as described under the logic test and the calibration test.

Then two output voltages will be requested (default: 0 V). After the user has typed in each value, the program displays the actual value which will be output and its equivalent bit pattern.

Two outputs are made on each of the channels selected. Either both values are output for each channel or the values switch with each new channel according to the response to the next question:

```
Either both values alternate on each channel or
they appear as levels on successive channels.
Use both values for each channel (Y/N)? [Y]:
```

The control table is set up to make all outputs except the last in mode 0. The last is in mode 1 to force a return back to the begining of the table. Continuous outputs are made with the AAF01 in double buffer mode until the user types control C (CTRL/C).

Note: Because this test must be stopped with CTRL/C, it can only be run interactively.

Note: All register error messages (e.g. ACS) are supplemented by the following information (expressed in octal values):
Expected: xxxx Was: yyyy

The following errors may be printed:

```
Error#240: DRX11-C SCR attention bit set during double buffer DMA output

Error#241: Interrupt timeout during double buffer DMA output

AAF01 selectable output test failed
```

If an error message occurs, the following text is also displayed:

```
AAF01 Exerciser failed
```

If a small conversion time is selected and any errors appear, the following remark is displayed:

```
NOTE: Since the selected conversion time is small, the last
      error is probably only an indication that the AAF01
      cannot supply the data quickly enough.
      You can increase the conversion time until this test
      runs without error; that conversion time will give the
      maximum throughput of the system for this test.
```

A sample run of the test program follows.

```
$ RUN AAF01$DIAG

This is the AAF01 Exerciser.
AAF01 Subroutine Library, Version 2.0
Copyright (C) 1988 Digital Equipment GmbH, CSS, Munich, Germany.

To all of the following questions you can respond with Ctrl-Z for exit
(ie. return to test selection if already in a test,
exit the AAF01 Exerciser otherwise).

The controller identification is one character between A and Z and
selects the AAF01 (_UUA0: to _UUZ0:) to which the AAF01 is connected.
Enter controller identification letter, default is A (ie. _UUA0:): B

AAF01 is switched to bipolar operation.

Test numbers are 0, 1, 2, 3 and 4.
0 selects all tests in sequential order
1 selects the AAF01 logic test
2 selects the AAF01 linearity test
3 selects the AAF01 calibration test
4 selects the AAF01 selectable output test
Enter test number, default is 0 (ie. all tests): 0

This is the AAF01 logic test.

To run this test, the test connector must be installed.
Use 2G-M00FA on module A6009.
Exerciser will continue only, if the next answer is "Y".
Is the test connector installed (Y/N)? [Y]: Y

Conversion times are expressed in multiples of 100 nanoseconds and
range from 25 to 4095.
Enter conversion time, default is 200 (ie. 20 microseconds): 400

The AAF01 logic test will start now. It will take up to 5 minutes.

AAF01 reset test running
AAF01 ACS read/write test running
AAF01 CTA read/write test running
AAF01 control table data test running
AAF01 control table address test running
AAF01 COUT and SEQ CONT L interrupt test running
AAF01 DMA to registers test running
AAF01 DMA to control table test running
AAF01 CWR mode 0 test running
AAF01 CWR mode 1 test running
AAF01 CWR mode 2 test running
AAF01 CWR mode 3, RATE CLOCK OUT and SEQ CONT test running
AAF01 programmable clock test running
AAF01 CWR mode 4 test running
AAF01 CWR mode 5 test running
AAF01 CWR mode 6 test running
AAF01 CWR mode 7 test running
AAF01 data buffer empty test running
AAF01 external clock test running
AAF01 DMA output comparison test running
AAF01 memory transfer conversion test running

This is the AAF01 linearity test.

The +/- tolerance of the conversion accuracy ranges from 0 to 4095.
An error occurs if the deviation from the desired value is greater.
Enter +/- tolerance, default is 7:

The AAF01 linearity test will start now. It will take up to 7 minutes.

This is the AAF01 calibration test.

To run this test, a digital voltmeter must be
connected to the ANA OUT and ANA RTN signals.
Exerciser will continue only, if the next answer is "Y".
Is the voltmeter connected (Y/N)? [Y]: Y
```

```
Select the channel numbers which should be tested (0 - 15).
Enter first channel, default is 0: 1
Enter last channel, default is 15: 10

Calibrating for bipolar  operation.

Check each channel output is -10.0000 volts +/- 4.88 millivolts.
Are the outputs within the allowed tolerance? (Y/N) [Y]: Y

Check each channel output is  9.9951 volts +/- 4.88 millivolts.
Are the outputs within the allowed tolerance? (Y/N) [Y]: Y

This is the AAF01 selectable output test.

This test allows you to select output patterns for
monitoring with an oscilloscope.

Conversion times are expressed in multiples of 100 nanoseconds and
range from 25 to 4095.
Enter conversion time, default is 200 (ie. 20 microseconds): 200

Select the channel numbers which should be tested (0 - 15).
Enter first channel, default is 0: 0
Enter last channel, default is 15: 15

Next two output voltages will be requested.
Specify any voltage between -10.0000 and  9.9951 V.

Enter desired voltage, default is 0.0000: 5
Actual output voltage (V):   5.0000
The octal output value is:  1777

Enter desired voltage, default is 0.0000: -3
Actual output voltage (V):  -2.9980
The octal output value is:  5145

Either both values alternate on each channel or
they appear as levels on successive channels.
Use both values for each channel (Y/N)? [Y]: Y

AAF01 selectable output test running. Type CTRL/C to return to
the Exerciser.

     (User typed CTRL/C)

Test numbers are 0, 1, 2, 3 and 4.
0 selects all tests in sequential order
1 selects the AAF01 logic test
2 selects the AAF01 linearity test
3 selects the AAF01 calibration test
4 selects the AAF01 selectable output test
Enter test number, default is 0 (ie. all tests): ^Z

End of AAF01 Exerciser

$
```

E TABLE OF ABBREVIATIONS

ACS	AAF01 Command and Status Register
AST	Asynchronous System Trap
CHA	Channel Address in CTA Register
CMP	Comparator Input bit in ACS Register
COUT	Command Output bit in ACS Register
COUT L	User output signal corresponding to the COUT bit in the ACS Register
CTA	Control Table Address Register
CWA	Control Word Address in CTA Register
CWM	Control Word Mode in CWR Register
CWR	Control Word Register
DBE	Data Buffer Empty bit in ACS Register
DBF	Data Buffer Silo (FIFO)
DMA	Direct Memory Access
DSR	Data Acquisition Reset bit in ACS Register
ECE	External Clock Enable bit in ACS Register
EOC	End of Conversion Sequence bit in ACS Register
GO	Go bit to enable conversion in ACS Register
MET	Memory Transfer bit in ACS Register
MHz	Megahertz
MNT	Maintenance bit in ACS Register
PCR	Programmable Clock Register
RATE CLK OUT	User output signal RATE CLOCK OUTPUT
SBE	Sequence Break Enable bit in ACS Register
SBR	Sequence Break bit in ACS Register
SCR	Status Control Register of the AAF01
SEQ CONT L	Sequence Control Input Signal
SYS IN PROG L	User output signal System In Progress
UNI	Unipolar Operation bit in ACS Register

F READER'S COMMENTS

READER'S COMMENTS

AAF/VMS Subroutine Library V2.0
User's and Installation Guide
Documentation No. AA-GI29D-TE
Last revision: September 1988

READER'S COMMENTS

Note: This form is for documents only. Digital will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well organized? Please make suggestions for improvement.

Did you find errors in this manual? If so, specify the error and the page number.

Please indicate the type of user/reader that you most nearly represent.

___ Assembly language programmer
___ Higher-level language programmer
(language: _____)
___ Occasional programmer (experienced)
___ User with little programming experience
___ Student programmer
___ Other (please specify) _____

Name _____ Date _____

Organization _____

Street _____

City _____ Zip Code _____

State _____ Country _____

READER'S COMMENTS

DIGITAL EQUIPMENT GMBH
COMPUTER SPECIAL SYSTEMS
Software Engineering

Freischuetzstrasse 91
D-8000 MUENCHEN 81
German Federal Republic

READER'S COMMENTS