MAX-PLANCK-INSTITUT FÜR PLASMAPHYSIK

GARCHING BEI MÜNCHEN

DIOS Device Handlers
Reference Manual

K. Engelhardt
E. Müller
R. Lathe

IPP R/25 October 1977

Die nachstehende Arbeit wurde im Rahmen des Vertrages zwischen dem Max-Planck-Institut für Plasmaphysik und der Europäischen Atomgemeinschaft über die Zusammenarbeit auf dem Gebiete der Plasmaphysik durchgeführt.

R/25

DIOS Device Handlers Reference Manual

K. Engelhardt
E. Mueller
R. Lathe

Abstract

This manual is designed to be used by system programmers and other advanced programmers wishing to use standard DIOS device handlers. It contains a description of the functions of all of the currently supported standard DIOS devices.

DIOS Device Handlers
Reference Manual

Data Acquisition Project
Institute for Plasma Physics
8046 Garching
West Germany

Table of Contents

Preface

0.1 Manual Objectives

This manual is designed to be used by system programmers and other advanced programmers wishing to use standard DIOS device handlers. It contains a description of the functions of all of the currently supported standard DIOS devices.

It is assumed that the reader is familiar with the operation of standard RSX-11M device handlers and their related executive directives. These points are fully covered in the manuals RSX-11M Executive Reference Manual and RSX-11M I/O Drivers Reference Manual. The user should also be familiar with the information contained in the DIOS Operations Manual, in particular, the loading and unloading procedures and standard I/O functions. It is recommended that the reader also refer to the RSX-11M Guide to Writing an I/O Driver.

0.2 Structure of the Manual

This manual is organized into chapters. The first chapter gives a description of the format of the remaining chapters. The remainder of the manual consists of detail descriptions of each of the DIOS standard device handlers.

CHAPTER 1

Introduction

Each of the following chapters gives a detailed description of a specific device handler supported by the DIOS system. The descriptions adhere to a standardized format which is outlined here.

1. Introduction to the Hardware and Software

   This section gives a brief description of the hardware operation and the features which are supported by the software. The reader should note the restrictions mentioned, as in many cases, not all hardware functions are supported. The reader should also note that some hardware functions are changed somewhat (in the software) in order to simplify the user interface; these changes are also described here.

2. Loading the Module

   The next section provides information on how the device handler can be loaded. The MCB format details deserve particular attention as its contents are critical for the proper functioning of the DIOS system. Note that only the device specific MCB parameters are described and that general information about MCB's is contained in the DIOS Operations Manual.

3. Unloading the Module

   The third section gives a brief summary of the procedure used for unloading a module.

4. QIO Functions to the Loaded Module

   This section provides information of high importance for the proper functioning of the module. The I/O functions specific to the device are described in detail, in particular, the device specific parameters associated with a device initialization request (IO.INI). The reader should note that, although only one form of the QIO directive macro is given, all forms ($, $C, $S) of the QIO as well as QIOW macro may be used. Care should be taken that the parameter lists associated with the macros have the proper form. These points are discussed in the DIOS Operations Manual and the RSX-11M I/O Drivers Reference Manual.

5. Status Returns and Error Handling

   The fifth section gives a description of the error codes which may be returned in the I/O Status Block and their possible causes.

6. Programming Hints

   The final section gives useful hints on using the device and warnings concerning common pitfalls.

CHAPTER 2

CMDRV: IPP CAMAC Memory Module

2.1 Introduction

The CAMMEM is a random access memory in CAMAC norm with 2048 (CAMAC single width module) or 4096 (CAMAC double width module) 16-bit words. The CAMMEM may be accessed via the Dataway or from an external data bus. Addressing for access via the dataway is done by setting an address register and then transferring the data or in auto-increment mode. Loading memory from the external bus is always done with automatic address increment. Mutiple CAMMEMs may be arranged up to 32K words capacity, where 2K and 4K modules may be intermixed. Multiple CAMMEMs connected to form one logical CAMMEM must occupy consecutive CAMAC stations.

The DIOS driver for the CAMMEM, CMDRV, operates on the module as follows: at initialization time the user determines the physical configuration of the logical CAMMEM to be used, and whether access is via the dataway or from the external bus. A read or write function always switches the access lines to the dataway, starting I/O at the address specified at initialization time. Any number of data (16-bit words) available may be read or written with a single QIO.

2.2 Loading the Module

The module is loaded by either of the two macro calls

```
QIO$S    #IO.LOD,#$LDR,....,<mcb,lmcb,lun>
```

or

```
LOAD     lun,mcb,sts,flg
```

The use of the macros and meaning of the arguments are given in the DIOS Operations Manual.

2.2.1 MCB Format

In the macro calls above, mcb is the address of the Module Control Block, described in the DIOS Operations Manual. The device specific portions of the MCB should be set as follows:

Offset	Contents
M.TYP	"CM" -- 2-letter type code for CAMMEM
M.UNIT	Unit number of CAMMEM.
M.ACP	2-Letter code of ACP containing the CAMMEM driver.
M.CTL	Control bits, set as follows: MC.CAM=1 Indicates the CAMMEM is a CAMAC module. MC.INT=0 Indicates interrupt service is not required by the CAMMEM.
M.ADR	CAMAC address in BCNA format (A=0) of the module.

2.3 Unloading the Module

The module is unloaded by either of the two macro calls:

```
QIO$S    #IO.UNL,#lun,...
```

or

UNLOAD lun,sts,flg

The use of these macros and the meaning of the arguments are given in the DIOS Operations Manual.

2.4 QIO Functions to the Loaded Module

This section summarizes the standard and device-specific QIO requests processable by the driver.

2.4.1 Standard Functions

	Format	Function
QIO$S	#IO.VAT,...	;Attach Module
QIO$S	#IO.VDT,...	;Detach Module
QIO$S	#IO.KIL,...	;Cancel I/O on Module
QIO$S	#IO.UNL,...	;Unload Module

2.4.2 Module-specific Functions

	Format	Function
QIO$S	#IO.INI,...,<ddp,lpm>	;Initialize CAMMEM
QIO$S	#IO.TER,...,<0,2>	;Terminate CAMMEM
QIO$S	#IO.RVB,...,<buf,lbuf>	;Read data from CAMMEM
QIO$S	#IO.WVB,...,<buf,lbuf>	;Write data to CAMMEM

where

ddp is the address of a block of device-dependent parameters defining the mode in which the CAMMEM is initialized.
lpm is the length of the block in bytes.
buf is the address of the buffer into which data are read or from which data are written.
lbuf is the length of the data buffer in bytes.

2.4.2.1 IO.INI - Initialize the CAMMEM

IO.INI passes a buffer specified in the parameter list of the QIO containing device-dependent parameters specifying the mode of initialization. Depending upon the contents of the buffer, the following actions are performed:

1. One or more physical CAMMEMs are logically connected to create a single, large external storage module.
2. The start address, from which data will be read or into which data will be written, is established.
3. The data bus, either the CAMMEM bus or the CAMAC Dataway, is enabled.

The buffer consists of 8 bytes which are formatted as follows:

Offset	Name	Type	Meaning
0	CORMAP	[D]	CAMMEM core map; this is a bit pattern which describes the physical modules of one logical CAMMEM. The word consists 8 entries of two bits each, where the low one is the logical online bit (bit set means module online) and the high bit shows the storage capacity of the module (bit set means 4K module, bit clear means 2K module).
2	RELADR	[D]	Memory start address relative to the first module.
4	ACCESS	[A]	set to "E" if external access is desired, or set to "I" if I/O is done via the dataway.
5	RESRV		reserved for future use.

The driver sets the status as defined by the device dependent parameters, where all parameters are checked for legality. Errors are reported via setting the status block accordingly.

2.4.2.2 IO.TER - Terminate Operations on the CAMMEM

IO.TER always sets the logical CAMMEM, defined by a prior IO.INI function, to be accessed via the dataway. The start address is reset to zero.

2.4.2.3 IO.RVB - Read Data from the CAMMEM

The CAMMEM I/O lines are switched to the dataway. Reading is done in auto-increment mode and starts at the location set by an IO.INI or IO.TER function or one word behind the address of the last read/write cycle. External input does not alter the location counter for a read function. Data are read and returned to the user buffer until the given buffer length is exhausted or until the physical end of the logically connected CAMMEMs is detected.

2.4.2.4 IO.WVB - Write Data to the CAMMEM

The same actions are taken as with IO.RVB, except that data are written from the user buffer to the CAMMEM. If more data are to be written than the capacity of the CAMMEM allows, an error status is returned and all following attempts to write data are ignored until the next IO.INI.

2.5 Status Returns and Error Handling

When a QIO function is completed, status information about the functions is returned in the I/O status block if specified in the QIO macro call.

2.5.1 First Status Word (low byte)

The error codes listed below may be returned by the CAMMEM Driver.

Code	Meaning
IS.SUC	A QIO for IO.INI, IO.TER, IO.RVB or IO.WVB was successfully completed.
IE.IFC	A function code other than IO.INI, IO.TER IO.RVB or IO.WVB or the standard functions was encountered.
IE.BAD	Returned from IO.INI, IO.TER, IO.RVB or IO.WVB if any bad parameters were encountered.
IE.OFL	Returned from IO.INI or IO.TER if the module is not at the given CAMAC station or the crate is offline.
IE.EOV	Returned from IO.RVB or IO.WVB if the storage capacity becomes exhausted during a read or write operation.

2.5.2 First Status Word (high byte)

DIOS drivers use this byte to group error conditions. Bit n on selects error code group n+1. All error codes used by the CMDRV belong to error code group zero.

2.5.3 Second Status Word

This word shows in all cases the number of bytes actually transferred to or from the user buffer.

2.6 Programming Hints

In DIOS the CAMMEM is provided to be filled from the external bus (e.g. with experimental data) and then to be read out via the data way. For this purpose the user initializes a logical CAMMEM for external access as outlined obove (IO.INI). After information is stored in the CAMMEM the user merely has to perform succeeding QIOs with function code IO.RVB to obtain the data stored in the CAMMEM, where the amount of data to be read each time is given by the user's buffer length. Data are returned in the same order as they are transferred to the CAMMEM, so the user is not concerned with any addressing. However, if the user wants to specify his own start address he might do so with another initialization, which does not alter the contents of the CAMMEM.

CHAPTER 3

CNDRV: Nuclear Enterprises CAMAC Memory Module

3.1 Introduction

The Nuclear Enterprises CAMAC Random Access Store (CN) is a double-width CAMAC module capable of storing either 2048 or 4096 16-bit words, depending on whether it is a 2K or 4K model. The memory is accessible via 2 ports: an external port which allows other hardware modules to store their data directly in the memory via a special I/O interface, and the CAMAC Dataway, over which data may be transferred between the memory and the computer.

3.1.1 External Input/Output Port

The external I/O port has the following features:

1. Data is transferred by way of a 52-pin Cannon plug for which sockets are mounted on the front and back of the module. Among the pins are 12 address lines, 5 control lines and 16 I/O data bus lines.

2. External modules may transfer data in 16-bit words or 8-bit bytes. It is up to the external module to supply the address on the address lines, as well as to set the byte/word control line properly.

3. The external port provides for a memory increment option, with which an external module may cause the contents of a selected word location to be incremented by one: this may serve, for example, as a multi-channel counter/store for PHA.

4. Access to the external port may be enabled and disabled by CAMAC commands from the computer.

3.1.2 CAMAC Dataway Interface

The Dataway interface allows the following operations:

1. Loading an address register prior to reading or writing, thus providing for random addressing to any word or byte location.

2. Reading or writing the contents of the memory currently addressed by the address register, either in byte or work mode (depending on the subaddress in the CAMAC command). The address is then incremented in one of the following modes, also selected by the subaddress:

   Increment by 1 word or 1 byte
   Increment by 64 words or 2 bytes
   Decrement by 1 word or 1 byte
   Decrement by 64 words or 2 bytes

3. Reading or writing in masked mode, in which the logical "and" of the data and the contents of a preloaded mask register are transferred. The same conventions as otherwise apply to a normal read or write hold here.

4. Loading the contents of the mask register.

5. Enabling and disabling external address.

6. Reading the address register.

7. Enabling, disabling, clearing, and testing the module LAM. LAM is set whenever the module is ready to accept a read or write command and LAM was enabled.

3.1.3 Driver Capabilities

The driver for the CN, CNDRV, utilizes a limited number of these hardware options, allowing the user to perform the following basic functions:

1. Read and write successive blocks of data bytewise or wordwise.

2. Initialize the memory to accept input from or deliver data to external modules.

3. Optionally fill the memory with zeros on initialization.

4. Specify a starting address at initialization at which subsequent reads and writes are to begin.

3.2 Loading the Module

The module is loaded by either of the two macro calls:

```
QIO$S    #IO.LOD,#$LDR,....,<mcb,lmcb,lun>
```

or

```
LOAD     lun,mcb,sts,flg
```

The use of the macros and meaning of the arguments are given in the DIOS Operations Manual.

3.2.1 MCB Format

In the macro calls above, mcb is the address of the Module Control Block, described in the DIOS Operations Manual. The device specific portions of the MCB should be set as follows:

Offset	Contents
M.TYP	"CN" -- 2-letter module type code for the Nuclear Enterprise Memory.
M.UNIT	Unit number of Nuclear Enterprise Memory.
M.ACP	2-Letter code of the ACP containing the Nuclear Enterprise Memory driver.
M.CTL	Control bits, set as follows: MC.CAM=1 Indicates the Nuclear Enterprise Memory is a CAMAC module. MC.INT=0 Indicates interrupt service is not required by the Nuclear Enterprise Memory.
M.ADR	CAMAC address in BCNA format (A=0) of the module.

3.3 Unloading the Module

The module is unloaded by either of the two macro calls:

```
QIO$S   #IO.UNL,#lun,...
```

or

```
UNLOAD  lun,sts,flg
```

The use of these macros and the meaning of the arguments are given in the DIOS Operations Manual.

3.4 QIO Functions to the Loaded Module

This section summarizes the standard and device-specific QIO requests processable by the driver.

3.4.1 Standard Functions

```
         Format                        Function

QIO$S    #IO.VAT,...                 ;Attach Module

QIO$S    #IO.VDT,...                 ;Detach Module

QIO$S    #IO.KIL,...                 ;Cancel I/O on Module

QIO$S    #IO.UNL,...                 ;Unload Module
```

3.4.2 Module-specific Functions

```
         Format                        Function

QIO$S    #IO.INI,...,<ddp,lpm>       ;Initialize Module

QIO$S    #IO.TER,...,<0,2>           ;Terminate Module

QIO$S    #IO.RVB,...,<buf,lbuf>      ;Read data from Module
```

3.4.2.1 IO.INI - Initialize the Store

IO.INI passes a buffer specified in the parameter list of the QIO containing device-dependent parameters specifying the mode of initialization. Depending upon the contents of the buffer, the following actions are performed:

1. One or more storage modules are logically connected to create a single logical store.

2. The start address, from which data will be read or into which data will be written, is established.

3. Either the External Port or the CAMAC Dataway is established for use in data transfers.

4. Byte or word width transfers are enabled.

5. The store is cleared.

The buffer consists of 8 bytes which are formatted as follows:

0	CORMAP	[D]	Core map. Bit 0 is always set; bit 1 is set if the module is a 4K (word) memory, clear if it is a 2K unit.
2	RELADR	[D]	Start address. The byte location from which to begin reading or writing in the first transfer after the IO.INI.
4	ACCESS	[A]	External/internal mode selection. Contains an ASCII "E" if the memory is to be initialized with external access enabled; contains an "I" if external access is to be disabled.
5	MODE	[Y]	Mode bits. Bit 0 is set if the memory should be read/write bytewise and cleared if wordwise read/write is desired. Bit 1 is set if the memory should be cleared on initialization.
6	RETRY	[Y]	Repeat count. This value gives the maximum number of times an access to a given location should be retried before a hardware error is declared. Failure is signalled by a Q-response of zero from an attempted CAMAC read/write operation. If a zero is given in this field, the driver repeats the attempt up to 256 times.
7	RESRV		Reserved for future use.

On initiation, subsequent read and write operations are enabled if they were disabled due to a proceding IO.TER. Further actions (clearing the memory, enabling external access, etc.) are carried out as implied in the description of the buffer above.

3.4.2.2 IO.TER - Terminate Operations on the Store

Access to the external port is disabled. Read/write access to the memory is prevented until another IO.INI is issued.

3.4.2.3 IO.RVB - Read Data from the Store

The contents of the memory are read starting at the current address, and transferred to the user buffer specified in the parmeter list of the QIO by buf, lbuf as above. On the first read after an IO.INI, the current address is the start address specified in the initialization. After the read operation is completed, the new current address points to the byte after the last one read, or to one byte after the end of memory if an end of volume occurred.

Thus successive reads read out consecutive blocks of memory. Also note that if the memory is read in byte mode, the high byte of each word is transferred to the user buffer before the low byte, thus the bytes in each word are swapped.

Before reading the memory, IO.RVB causes the external access to be disabled if it was enabled.

3.4.2.4 IO.WVB - Write Data to the Store

The contents of the user buffer specified in the QIO parameters are transferred to the memory starting at the current address. The current address is initialized and updated as in IO.RVB. The effect of choosing the byte or word mode is identical to that in IO.RVB. The current address is the same for read or write - this means that a read following a write starts where the write left off, and vice-versa.

3.5 Status Returns and Error Handling

When a QIO function is completed, status information about the functions is returned in the I/O status block if specified in the QIO macro call.

3.5.1 First Status Word (low byte)

The error codes listed below may be returned by the Nuclear Enterprises Memory driver.

Code	Meaning
IS.SUC	A QIO was successfully completed.
IE.IFC	This code is returned if an I/O function other than IO.RVB, IO.WVB, IO.INI, IO.TER or one of the standard functions was submitted to the module.
IE.OFL	The module is nonexistent, or the crate is off-line. This code may be retured from IO.INI, IO.TER, IO.RVB or IO.WVB.
IE.EOV	An attempt was made to read or write past the physical end of the memory as determined by the 2K/4K bit passed on IO.INI. The current address is left at the end of memory, and all bytes between the previous address and the end are transferred.
IE.FHE	Returned from IO.RVB or IO.WVB if the driver was unable to read or write a given location of the memory after repeated attempts. The maximum number of retries is set by the Repeat Count field of the IO.INI buffer. This error may also occur on IO.INI if Clear was specified and a given location could not be cleared within the allowed number of retries.
IE.DNR	Returned from IO.RVB or IO.WVB if issued after an IO.TER and before the next IO.INI.
IE.BAD	Returned from IO.INI if one of the following is true:

1. Bit 0 of the Core Map is 0

2. The Start Address is past the end of memory

3. Word mode is specified and an odd start address is given.

4. The buffer passed on IO.INI is shorter than 12(8) bytes.

3.5.2 First Status Word (high byte)

DIOS drivers use this byte to group error conditions. Bit n on selects error code group n+1.

3.5.3 Second Status Word

On IO.RVB of IO.WVB the second status word always contains the actual number of bytes transferred successfully; if an error occurred this number is less than the number requested. On IO.INI and IO.TER, this word is set to zero.

3.6 Programming Hints

3.6.1 Using Byte Mode or Word Mode

If the memory is used to return byte data entered by an external module, using byte mode ensures the data are returned in the correct order. If word mode were used, the bytes would be switched within each word from the correct order. Using byte mode also allows reading an odd number of bytes and starting at an odd address.

On the other hand, byte mode is slower than word mode, and for time critical applications it may be advisable to switch the bytes per software after issuing a read in word mode.

3.6.2 Mode of Transfer

Since the transfers are processed synchronously in system state, RSX is held up for the duration of the transfer. This manifests itself only for long transfers; for example, a single read for 8192 bytes in byte mode requires about 1 second.

CHAPTER 4

FKDRV: Function Keyboard

4.1 Introduction

The Function Keyboard (FKB) is a simple Input/Output device with 16 keys, each associated with a lamp. It is designed to be connected to PDP11s via the general interface DR11A or DR11C. The keys with associated lamps are considered to be connected to the I/O registers of the interface, where key n is attached to the respective bit of the input register and lamp n is attached to the respective bit of the output register.

The DIOS driver for the FKB, FKDRV, operates on the module as follows: Initialization and termination functions cause the interrupt logic to be disabled and output and input buffer to be cleared, thus turning off all lamps and setting the status to no key pressed. More than one datum (equal one byte) may be transferred with a single QIO. A read function enables the interrupt logic. On occurance of an interrupt the lamp of the pressed key is turned on and the key number is returned to the user buffer. A write function causes the lamps of the key numbers given in the user buffer to be turned on.

4.2 Loading the Module

The module is loaded by either of the two macro calls

```
QIO$S    #IO.LOD,#$LDR,...<mcb,lmcb,lun>
```

or

```
LOAD     lun,mcb,sts,flg
```

The use of the macros and meaning of the arguments are given in the DIOS Operations Manual.

4.2.1 MCB Format

In the macro calls above, mcb is the address of the Module Control Block, described in the DIOS Operations Manual. The device specific portions of the MCB should be set as follows:

Offset	Contents
M.TYP	"FK" -- 2-letter module type code for FKB.
M.UNIT	Unit number of FKB.
M.ACP	2-Letter code of the ACP containing the FKB driver.
M.CTL	Control bits, set as follows: MC.CAM=0 Indicates the FKB is not a CAMAC module. MC.INT=1 Indicates interrupt service is required by the FKB.
M.ADR	CSR address of the module.

4.3 Unloading the Module

The module is unloaded by either of the two macro calls:

```
QIO$S    #IO.UNL,#lun,...
```

or

```
UNLOAD   lun,sts,flg
```

The use of these macros and the meaning of the arguments are given in the DIOS Operations Manual.

4.4 QIO Functions to the Loaded Module

This section summarizes the standard and device-specific QIO requests processable by the driver.

4.4.1 Standard Functions

```
        Format                       Function

QIO$S   #IO.VAT,...                 ;Attach Module

QIO$S   #IO.VDT,...                 ;Detach Module

QIO$S   #IO.KIL,...                 ;Cancel I/O on Module

QIO$S   #IO.UNL,...                 ;Unload Module
```

4.4.2 Module-specific Finctions

```
        Format                       Function

QIO$S   #IO.INI,...,<ddp,lpm>       ;Initialize FKB

QIO$S   #IO.TER,...,<0,2>           ;Terminate FKB

QIO$S   #IO.RVB,...,<buf,lbuf>      ;Read data from FKB

QIO$S   #IO.WVB,...,<buf,lbuf>      ;Write data to FKB
```

where

ddp	is the address of a block of device-dependent parameters defining the mode in which the Function Keyboard is initialized.
lpm	is the length of the block in bytes.
buf	is the address of the buffer into which data are read or from which data are written.
lbuf	is the length of the data buffer in bytes.

4.4.2.1 IO.INI - Initialize the FKB

IO.INI passes a buffer specified in the parameter list of the QIO containing device-dependent parameters specifying the mode of initialization. The buffer consists of four bytes which are at present not significant, but are reserved for future use. The driver disables the interrupt logic of the module and clears the input and output buffer registers of the interface.

4.4.2.2 IO.TER - Terminate Operations on the FKB

The same action is taken as with I/O operation code IO.INI.

4.4.2.3 IO.RVB - Read Data from the FKB

The interrupt logic of the module is enabled. On occurance of an interrupt, further interrupts are locked out and the input buffer register is scanned for a bit set beginning with the least significant bit. The lamp associated with the key number attached to the bit, which was first found set, is turned on and the key number is returned in the user buffer. If there are more selections requested, the interrupt logic is reenabled and the next interrupt is accepted. This procedure is repeated until all selections are received.

4.4.2.4 IO.WVB - Write Data to the FKB

Key numbers are taken from the user buffer and the lamps, associated with the keys are turned on. Key numbers given by the user must range from zero to 15.

4.5 Status Returns and Error Handling

When a QIO function is completed, status information about the function is returned in the I/O status block if specified in the QIO macro call.

4.5.1 First Status Word (low byte)

The error codes listed below may be returned by the FKB Driver.

Code	Meaning
IS.SUC	A QIO for IO.INI, IO.RVB, IO.WVB or IO.TER was successfully completed.
IE.IFC	A function code other than IO.INI, IO.RVB, IO.WVB IO.TER or the standard functions was encountered.
IE.BAD	Returned from IO.WVB if a key number was found, which was greater than 15.
IE.ABO	Returned from IO.KIL if a read operation was in progress.

4.5.2 First Status Word (high byte)

DIOS drivers use this byte to group error conditions. Bit n on selects error group n+1. All error codes returned by the FKDRV belong to error code group zero.

4.5.3 Second Status Word

This word shows in all cases the number of bytes actually transferred to the user buffer.

CHAPTER 5

MADRV: Canberra Model 8100

5.1 Introduction

The Canberra Model 8100 MCA allows the experimentalist to collect, display, and record pulse-height spectra or pulse-frequency vs. time data. If provided with Option 14 (Serial Computer Interface), the MCA may be controlled from a PDP-11 via a DL11-E asynchronous line interface. Collected data may be read from the MCA, or the MCA may be pre-loaded with data from the computer.

The MCA is incorporated into the DIOS system as module type MA, with corresponding driver MADRV, described in the following.

5.1.1 Data Collection

Data may be collected in one of five modes:

1. PHA-add: accumulate a pulse-height spectrum in the selected group of memory channels.

2. PHA-subtract: subtract the accumulating spectrum from the previous data in memory.

3. MCSS (Multi-scaling, single sweep): sample and record the rate of incoming pulses as a function of time, for a single cycle of the process being analyzed.

4. MCSR (Multi-scaling, repetitive sweep): same as MCSS, but add in data from repeated sweeps to average over several cycles.

5. MMCS (Multi-input multi-scaling): accumulate multi-scaling data from several sources concurrently, with each source directing its data to its assigned memory group.

The driver supports all of these modes except MMCS.

5.1.2 Memory Capacity (see ref. 1, sec. 2.2)

Data are accumulated and stored in a semiconductor (optionally core) memory, consisting of 1024 channels (4096 with core), each capable of holding up to 999,999 counts.

Memory is further subdivided into the following groups:

Group			Ch. 0	Size
A	1/1	Whole Memory	0	n
B	1/2	First Half	0	n/2
C	2/2	Second Half	n/2	n/2
D	1/4	First Quarter	0	n/4
E	2/4	Second Quarter	n/4	n/4
F	3/4	Third Quarter	2n/4	n/4
G	4/4	Fourth Quarter	3n/4	n/4

n = 1024 for semiconductor memory; n = 4096 for core.

It is possible to specify the group in which data is to be collected or from which to read data. In addition, it is possible to have an external routing module specify into which half or quarter of the memory to send a given datum:

H	A/2	Collect into half given by routing module
I	A/4	Collect into quarter given by routing module

5.1.3 Display

The data from one of the memory groups (A - G) may be displayed on a 5 x 5 inch screen on the front of the MCA. The display may include several options, allowing the user to specify and intensify certain bands of consecutive channels. An integral mode displays the integral across the selected bands.

To enter the special display modes from the computer, it is necessary to include the IO.CMD option when the driver is assembled, allowing the special display commands to be given.

5.1.4 Serial Computer Interface

The MCA may be interfaced to the PDP-11 by including options 4 (Basic I/O), 12 (Memory I/O), and 14 (Serial Line Interface) in the MCA. The PDP-11 must be equipped with a DL11-E Asynchronous Line Interface.

5.1.4.1 DL11 Interface -

Commands and Data are transferred via the DL11-E interface as sequences of ASCII characters. The basic techniques are described in ref. 2. In addition, the following things should be noted:

-- All transactions are asynchronous, interrupt-driven.

-- The transmitter interrupt is enabled only when a command string is being output.

-- The receiver interrupt is enabled only when data is being input, or the driver is waiting for a command to be acknowleged by a "!" character from the MCA.

-- The parity of incoming characters is not checked, and the parity bit (bit 7) is cleared.

-- Dataset control in the DL11-E is not used.

5.1.4.2 Command Output -

Commands consist of sequences of 5 ASCII control characters preceded by a "#" character. The exact format and meaning of the commands are described in Ref. 1, Sec. 5.5.

5.1.4.3 Data Input -

Data are input in two stages: first the MCA is set into "Remote Out" mode by issuing the I/O command

```
#mxxIB
```

where m denotes the memory group to be read, xx are arbitrary, I sets the I/O mode, and B has no effect.

Then the contents of the memory group are read, starting with channel zero and ending with the last channel in the group. The transmission of each channel is started by outputting a "%" character to the MCA. The contents are then received as ASCII strings of the following form:

[<RO>],n1,n2,n3,n4,n5,n6,<SP>|<CR><LF>

where

<RO>	is ASCII rubout -- 177 (8)
<SP>	is a blank -- 40(8)
<CR>	is a carriage return -- 15 (8)
<LF>	is a line feed -- 12(8)
n1-n6	are the six ASCII decimal digits of the value.

Rubout is omitted in the first number of each line; space follows each number not at the end of a line; the last number in each line is finished by the <CR><LF> combination.

The transmission of each channel is started by sending a "%" character to the MCA; the characters of the number are then received until the closing space or line-feed is seen; at this point the MCA waits for the next "%" character.

The first "%" after the I/O command causes a leading <CR><LF> combination to be sent to the computer. The driver ignores these two characters.

When the last channel has been read, the MCA sends a "!" in response to the next "%" to indicate no more data are available.

Channel 0 contains special timing data, depending on the collect mode used.

5.2 Loading the Module

The module is loaded by either of the two macro calls

```
QIO$S   #IO.LOD,#$LDR,...,<mcb,lmcb,lun>
```

or

```
LOAD    lun,mcb,sts,flg
```

The use of the macros and meaning of the arguments are given in the DIOS Operations Manual.

5.2.1 MCB Format

In the macro calls above, mcb is the address of the Module Control Block, which has the following contents.

```
Offset  Contents

M.TYP   "MA" -- 2-character type code for MCA.
M.UNIT  Unit number of MCA
M.ACP   2-character id of ACP containing MA driver.
M.CTL   Control Bits

        MC.CAM = 0      not a CAMAC module
        MC.INT = 1      uses interrupts

M.VCT   <receiver interrupt vector address>/4
M.PRI   <interrupt priority> x 40(8) (in upper 3 bits)
M.ADR   Address of Receiver CSR of the DL11-E
M.DLN   4 if only double integer data wanted.
        1 if ASCII data or IO.CMD function desired.
M.DFM   2 -- Integer Data
```

5.3 Unloading the Module

The module is unloaded by either of the two macro calls

```
QIO$S   #IO.UNL,#lun,...
```

or

```
UNLOAD  lun
```

The use of these macros and the meaning or the arguments are given in the DIOS Operations Manual.

5.4 QIO Functions to the Loaded Module

5.4.1 Standard Functions

Format	Function
QIO$S #IO.VAT,...	Attach Module
QIO$S #IO.VDT,...	Detach Module
QIO$S #IO.KIL,...	Cancel outstanding I/O
QIO$S #IO.UNL,...	Unload Module (Ref. 3)

5.4.2 Module-specific Functions

Format	Function
QIO$S #IO.INI,...,<ddp,lpm>	Initialize the MCA
QIO$S #IO.TER,...,<#0,#10>	Terminate MCA operations
QIO$S #IO.RVB,...,<buf,lbuf>	Read data from MCA store

where

ddp is the address of a block of device-dependent parameters defining the mode in which the MCA is initialized.

lpm is the length of the block in bytes.

buf is the address of the buffer into which data are read.

lbuf is the length of the data buffer in bytes.

5.4.2.1 IO.INI - Initialize the MCA

IO.INI passes to the driver a buffer containing device dependent parameters specifying the mode in which to initialize the MCA. Depending on these parameters, the following actions are taken.

1. The transmit interrupt vector is linked to the driver.

2. The module is reset: any ongoing processes are stopped.

3. The memory group specified in GROUP is shown on the screen.

4. If CLEAR is set, the memory group is cleared.

5. If automatic or flipped mode is specified, the driver begins collecting data into the specified memory group.

5.4.2.1.1 Device Dependent Parameters -

The DDP's are passed in a buffer specified in the QIO, with the following contents:

Offset	Name	Type(+)	Meaning
0	MODE	[A]	Desired operating mode: M Manual Collect A Automatic Collect F Flipped Collect
1	GROUP	[A]	Memory group into which data are collected (MODE = A and F) and from which they are read (all modes). A (1/1) Whole memory B (1/2) First half C (2/2) Second half * D (1/4) First quarter * E (2/4) Second quarter * F (3/4) Third quarter * G (4/4) Fourth quarter * These groups are illegal for MODE = F.

2	COLMOD	[A]	Collect mode: A PHA - add B PHA - subtract C MCSS D MCSR Note: Mode E (MMCS) is not allowed.
3	TIM0	[A]	One-digit mantissa of preset time. Range: 0 to 9.
4	TIM1	[A]	One-digit exponent of preset time: Range: 0 to 5.
5:Bit0	ASCII	[B]	If set, return data directly in ASCII form on IO.RVB. If clear, return data as Integer*4.
5:Bit1	CLEAR	[B]	If set, clear memory group if MODE = A or F. Ignored if MODE = M.
6	TIME	[D]	Total time in seconds to collect data. If TIME = 0, collect indefinitely. If MODE = A, collect until TIME expires. If MODE = F, complete current full period (first half / second half) and stop collecting when TIME has expired.
10	FTIME	[Y]	Flip interval in seconds: amount of time to collect data in each half of the memory group before flipping to the other half. Used for MODE = F only.

(+) Type Codes: A = ASCII Byte, B = Bit, Y = Octal Byte, D = Decimal Word.

5.4.2.1.2 Manual Mode (MODE = M) -

Initialization in Manual Mode allows the user to collect data by manual controls on the MCA, then read the collected data to the computer. The IO.INI may be done either before or after collecting the data. The following actions are performed:

-- The selected memory group is displayed.

-- The driver is prepared to read the desired group when the next IO.RVB is issued.

The following should be noted:

-- CLEAR is ignored -- it is assumed the data are or will be in the proper form after the manual phase is over.

-- TIME, FTIME, COLMOD, TIM1, and TIM0 are all ignored.

Data may be collected in any mode (including MMCS) and any memory group allowed by the manual controls. Care must be taken when specifying GROUP to be sure the correct data are being read.

5.4.2.1.3 Automatic Mode (MODE = A) -

The selected memory group is displayed. If CLEAR = 1, it is then cleared. Within two seconds, a command is issued to the MCA to begin collecting data in the desired mode and memory group (sec. 4.1.5)

After issuing the collect command, the driver begins timing the collect period; if TIME = 0, no timing is done.

Collection is stopped by one of the following:

-- TIME expires, causing the driver to display GROUP.

-- The preset time specified by TIM0 and TIM1 expires, causing the MCA to display the collected group.

-- An IO.RVB is issued, causing the driver to display and then read out GROUP.

-- An IO.TER, IO.INI, or IO.CMD function is issued.

5.4.2.1.4 Flipped mode (MODE = F) -

In flipped mode, the specified memory group is divided in half and each half collected separately; thus only groups A, B, or C may be specified.

The selected memory group is displayed. If CLEAR = 1, it is cleared. Within two seconds, the driver begins collecting data in the following sequence:

1. Collect the first half by issuing the collect command for the first half group.

2. Wait n seconds, where n is given in FTIME.

3. Collect the second half group.

4. Wait n seconds.

5. Repeat the above cycle.

Collection is stopped by one of the following:

-- The interval specified by TIME expires.

-- An IO.RVB is issued.

-- An IO.INI, IO.TER, or IO.CMD is issued.

The first two conditions stop collecting only after the second half group of the collect cycle has finished. This ensures that data are collected into each memory half for the same total length of time. The memory group is then displayed. On IO.RVB, the MCA is set into I/O mode and the data are read.

On IO.INI or IO.TER (also IO.CMD, for test version of MADRV only), collection is stopped immediately.

The assignment of memory halves is as follows:

GROUP	first half	second half
A (1/1)	B (1/2)	C (2/2)
B (1/2)	D (1/4)	E (2/4)
C (2/2)	F (3/4)	G (4/4)

5.4.2.1.5 Collecting Modes -

These parameters are ignored for MODE = M.

For MODE = F or MODE = A, the computer initiates data collection with a collect command of the following form:

#mtecB

where

m is the desired memory group (GROUP for mode A; first or second half for mode F) into which to collect data.

t is the contents of TIM0, specifying the preset time mantissa.
e is the contents of TIM1, the preset time exponent.
c is the contents of COLMOD, giving the desired mode of collection.
B is required but has no effect in a collect command.

For the PHA modes (COLMOD = A or COLMOD = B), TIM0 and TIM1 specify the total time during which to analyze pulses, as follows:

$$\text{preset time} = \text{TIM0} \times 10^{\text{TIM1}}$$

Note that in A and F modes, if preset time is shorter than the expiration times TIME or FTIME, respectively, the MCA will stop collecting before the computer issues a stop command. If this is undesirable, TIM0 and TIM1 should specify a longer preset time, most conveniently 9 and 5 (the longest possible). Note that the MCA timing is slightly more accurate than computer timing, thus sometimes TIM0 and TIM1 should be used.

For Multiscaling modes (COLMOD = C or D), TIM0 and TIM1 specify the dwell time as described in ref. 1:

$$\text{dwell time} = \text{TIM0} \times 10^{-\text{TIM1}}$$

Their value depends on the experiment, and does not affect driver function.

NOTE

In modes A and F, data collection occurs as an autonomous driver process. No QIO is active on the module. It is dangerous to unload the MCA while data collection is going on -- it should be terminated with IO.TER first.

5.4.2.2 IO.RVB - Read Data from Memory Group

On the first IO.RVB after an IO.INI:

1. Stop the collect process if mode A or F. (See sections 4.1.3 and 4.1.4 for details)

2. Display the memory group selected.

3. When ready, set the MCA in I/O mode by issuing an I/O command of the form:

   #mxxIB

   where m is the group in GROUP and I the I/O command character. The I/O light on the MCA should go on at this point.

4. Read the data from the memory as described in section 1.4.3. Convert each channel to the proper form (see below) and place in user buffer.

Subsequent IO.RVB functions continue reading from the next channel. The MCA is still in I/O mode from the previous IO.RVB.

When all the data from the given group have been read, the MCA returns to display mode. If any further data are requested, an end of message character is received: this is reported to the user and the IO.RVB is broken off. Further IO.RVB's without intervening IO.INI also report end of message.

If an error is encountered (a "?" is received, or a bad character is received) this is reported and the IO.RVB is broken off. Subsequent IO.RVB's without intervening IO.INI continue to report an error.

5.4.2.2.1 Data Formats -

ASCII = 0:

Each number is converted to a Integer*4. Only entire numbers are transferred to the buffer. The buffer length must be a multiple of 4.

ASCII = 1: (only if ASCII option is assembled)

The ASCII characters received (See section 1.4.3) are copied to the user buffer after stripping off the parity bit (bit 7). As many bytes as are requested are returned (up to end of message). If the buffer will only hold part of a number, the rest is saved and transferred on the next IO.RVB.

Numbers are either 7, 8 or 9 characters long. Each 9 character number is followed by a 7-character number. The user should allow 8 characters per number in the data buffer.

5.4.2.2.2 Readout Sequences -

MODE = A:

Channel 0 is read first, followed by channels 1 to n-1 where n is the size of the memory group.

Channel 0 contains the collect time in seconds (PHA modes) or the number of sweeps recorded (Multiscaling modes).

Channels 1 to n-1 contain the count data.

MODE = F:

Channel 0 is read first, followed by channels 1 to n-1 where n is the size of the memory group.

Channel 0 contains the collect time or sweep count of the first half of the memory group. Channel n/2 contains the collect time or sweep count of the second half.

MODE = M:

Channel 0 always contains a collect time or sweep count. If data were manually collected into subgroups of the selected memory group, the corresponding channels (n/2 or n/4,...) will contain the channel 0 data from the following memory subgroup.

5.4.2.3 IO.TER - Terminate MCA Operations

This function does the following:

1. Stop any ongoing collect process.

2. Display the entire memory (group A) if possible.

3. Reset the transmit interrupt vector.

It is strongly recommended to issue an IO.TER before unloading the module.

5.4.2.4 IO.CMD - Issue Command to MCA

(Allowed only if IO.CMD option is assembled)

The buffer specified in the QIO must contain a 5-character command which is simply issued to the MCA. The format of the command string is as follows:

<CR1>,<CR2>,<CR3>,<CR4>,<CR5>

where each of the CRn entries denotes the contents of the corresponding command register as described in ref. 1 sec 5.5. The individual characters are not checked. The necessary "#" character is added to the front of the command by the driver.

If the fourth character <CR4> is an F (display), K (clear) or L (clear channel 0), the MCA must send an acknowlege; the driver waits for this character. If it is not received, the IO.CMD will time out.

If data is being collected following an IO.INI, the IO.CMD function will break off the process.

5.5 Status Returns and Error Handling

When a QIO function is completed, status information about the function is returned in the I/O status block if specified in the QIO macro call. The first word contains an error or success code; the second word contains the number of bytes transferred in a transfer operation.

5.5.1 First Status Word (Error Code)

Code Meaning

IE.IFC INVALID FUNCTION CODE
This code is returned if a QIO function other than the standard functions, IO.RVB, IO.INI, or IO.TER is submitted. If the option was assembled into the driver, IO.CMD is also allowed.

IE.BAD BAD PARAMETERS
From IO,CMD -- The command passed in the buffer is shorter than 5 characters.
From IO.INI:
-- the DDP buffer may be shorter than 11(8) bytes.
-- MODE is illegal (must be A, F, or M).
-- COLMOD is illegal (must be A - D).
-- GROUP is illegal
If MODE = A or M, may be A - G.
If MODE = F, may be A - C.
-- TIM0 is other than 0 - 9.
-- TIM1 is other than 0 - 5.
-- TIME is less than 0.
-- FTIME is equal to 0 (checked only if MODE = F).

IE.RBG ILLEGAL RECORD SIZE
Returned from IO.RVB if the buffer length specified is not a multiple of 4 bytes, unless ASCII = 1 was specified on IO.INI.

IE.IDS INCONSISTENT WITH DEVICE STATE
Returned from IO.RVB if an IO.INI was not completed successfully since the module was loaded or the last IO.TER operation was done.

IE.OFL DEVICE OFFLINE
Returned from a LOAD if the DL11-E does not exist at the specified CSR address.

IE.DNR DEVICE NOT READY
The MCA is not turned on.

IE.FHE FATAL HARDWARE ERROR
Returned from all functions if:
-- a "?" character was sent by the MCA in response to a command or a request for the next channel contents. This most likely means that the MCA is not set in external I/O mode.
-- If an overrun or framing error takes place on receiving a character (see ref. 2). This could mean trouble with the DL11 or the MCA line interface.

Note: If IE.FHE has been returned from an IO.RVB function, subsequent IO.RVB'S continue to report IE.FHE until an IO.INI is performed.

IE.EOV END OF VOLUME DETECTED
Returned from IO.RVB if all channels from the group specified in GROUP on the last IO.INI have been read. Once IE.EOV has been reported, further IO.RVB's continue to return IE.EOV until another IO.INI is performed.

IE.ABO QIO ABORTED
The previous I/O request was aborted by an IO.KIL function issued by the task.

IE.PWF POWERFAIL ABORT
The previous I/O operation was aborted due to powerfail. No recovery is done; however, an IO.INI in MODE = M may be done to re-read the data stored (core only).

5.5.2 Second Status Word

On IO.RVB, the second word of the I/O Status Block always contains the number of bytes of data actually transferred to the user buffer. In ASCII mode, this is equal to the number of characters read from the MCA. In Integer mode (ASCII=0), this is equal to 4 times the number of channels actually read. (No partial double integers are transferred).

On IO.INI, IO.TER, and IO.CMD, this word is always equal to zero.

5.6 Programming Hints

5.6.1 Driver Assembly Options

The standard version of the driver does not allow the following:

-- executing an IO.CMD function
-- reading data in ASCII.

To include these features, the driver must be reassembled, preceding the driver source with the following definitions:

```
A$$SC = 0        ;ASSEMBLE CODE TO READ DATA IN ASCII

C$$MD = 0        ;ASSEMBLE CODE TO DO IO.CMD FUNCTION
```

5.7 References

1. Multichannel Analyzer Model 8100 Instruction Manual, (Canberra Industries).

2. DL11 Asynchronous Line Interface Manual, (Digital Equipment Corporation)

CHAPTER 6

MXDRV:IPP Multiplex ADC

6.1 Introduction

The MUXADC (MUX) is a double width CAMAC module consisting of a Multiplexer with Sample and Hold and ADC (Analog to Digital Converter). There are 16 single ended inputs (channels) to the Multiplexer on the front panel, where the maximum throuput rate is 100KHz at 12 bit resolution. Three operation modes are supported by the MUX: Random, Sequential Triggered and Sequential Scan mode.

Random:

> The number of the channel to be converted is written to the MUX CSR, which causes the selected channel to be displayed on the front panel. An internal trigger pulse is generated to initiate the conversion of the selected channel. 10usec after writing the CSR, the converted value can be read from the data register via the Dataway.

Sequential Triggered:

> In this mode, all channels starting with channel number zero and ending with a presettable end channel are converted. The conversion is initiated via an external trigger; for each external trigger pulse one channel is converted and the Multiplexer is advanced to the next channel. After conversion of the selected endchannel, the Multiplexer is reset to channel zero.

Sequential Scan:

> Same as with Sequential Triggered mode, with the exception that for one external trigger pulse all channels, starting from channel zero up to the se-

lected endchannel, are converted. This mode may be used for the fastest conversion.

A MUX operated in Random Mode may be attached to extenders to increase the number of single ended inputs. The extender type presently supported is a 32 channel extender manufactured by DORNIER Electronics. Each of the single ended inputs of the MUX may be attached to a DORNIER extender, thus increasing inputs to a maximum of 512 channels. MUX and extenders must occupy sequential station numbers always starting with the MUX. Hardware characteristics, specification and operation may be obtained from the reference documentation.

The DIOS driver for the MUX, MXDRV, operates on the module as follows: At initialization time the user determines the mode of operation and a possible end channel. If the selected mode is one of the Sequential modes, it is assumed that the MUX delivers data not to the MXDRV but to an external device such as the CAMMEM. This is provided by the MXDRV because of the possible high operation speed in these modes, which cannot be satisfied by the driver under all circumstances. The MXDRV supports two distinct Random modes, Random and Programmed Random, where the latter mode is used to read data in sequence starting with channel zero. The other random mode allows the user to specify the sequence in which channels shall be read. In both modes any number of channels available may be read with a single QIO. Data returned are formatted, where the low 12 bits represent the converted value and the high 4 bits give the MUX channel number from which the datum derives.

6.2 Loading the Module

The module is loaded by either of the two macro calls

```
QIO$S   #IO.LOD,#$LDR,....,<mcb,lmcb,lun>
```

or

```
LOAD    lun,mcb,sts,flg
```

The use of the macros and meaning of the arguments are given in the DIOS Operations Manual.

6.2.1 MCB Format

In the macro calls above, mcb is the address of the Module Control Block, described in the DIOS Operations Manual. The device specific portions of the MCB should be set as follows:

Offset	Contents
M.TYP	"MX" -- 2-letter module type code for MUX.
M.UNIT	Unit number of MUX.
M.ACP	2-Letter code of ACP containing the MUX driver.
M.CTL	Control bits, set as follows: MC.CAM=1 Indicates the MUX is a CAMAC module. MC.INT=0 Indicates interrupt service is not required by the MUX.
M.ADR	CAMAC address in BCNA format (A=0) of the module.

6.3 Unloading the Module

The module is unloaded by either of the two macro calls:

```
QIO$S   #IO.UNL,#lun,...
```

or

```
UNLOAD  lun,sts,flg
```

6.4 QIO Functions to the Loaded Module

This section summarizes the standard and device-specific QIO requests processable by the driver.

6.4.1 Standard Functions

	Format	Function
QIO$S	#IO.VAT,...	;Attach Module
QIO$S	#IO.VDT,...	;Detach Module
QIO$S	#IO.KIL,...	;Cancel I/O on Module
QIO$S	#IO.UNL,...	;Unload Module

6.4.2 Module-specific Functions

	Format	Function
QIO$S	#IO.INI,...,<ddp,lpm>	;Initialize MUX
QIO$S	#IO.TER,...,<0,2>	;Terminate MUX
QIO$S	#IO.RVB,...,<buf,lbuf>	;Read in sequence
QIO$S	#IO.RRD,...,<buf,lbuf>	;Read random

where

ddp	is the address of a block of device-dependent parameters defining the mode in which the MUXADC is initialized.
lpm	is the length of the block in bytes.
buf	is the address of the buffer into which data are read or from which data are written.
lbuf	is the length of the data buffer in bytes.

6.4.2.1 IO.INI - Initialize the MUXADC

IO.INI passes a buffer specified in the parameter list of the QIO containing device-dependent parameters specifying the mode of initialization. Depending upon the contents of the buffer, the following actions are performed:

1. The logical multiplexer configuration consisting of one or more physical multiplexers is established.

2. The end channel for scanning modes is set. This is indicated on the MUXADC front panel.

3. The multiplexing mode is set.

The buffer consists of 8 bytes which are formatted as follows:

Offset	Name	Type	Meaning
0	EXTMAP	[D]	MUX extender map. This is a bit pattern which shows, whether extenders are connected to the MUX and if any, to which MUX channels they are attached. Bit n set means a DORNIER extender is connected via MUX channel n. MUX and extenders have to be arranged as outlined in the introduction. Extenders have to be arranged as follows: The extender coupled to the lowest numbered MUX channel is placed in the station immediately following the MUX, followed by the extender connected to the next higher numbered MUX channel and so on.
2	ECHAN	[Y]	End channel of MUX, which is only relevant for Sequential modes.
3	MUXMOD	[A]	ASCII mode key, where "R" indicates Random, "T" Sequential Triggered and "S" Sequetial Scan mode.
4	RESEV		Reserved for future use.

The driver sets the status as defined by the device dependent parameters, where all parameters are checked for legality. Errors are reported via setting the status block accordingly.

6.4.2.2 IO.TER - Terminate Operations on the MUXADC

IO.TER always sets the MUX to Random mode with channel zero selected.

6.4.2.3 IO.RVB - Read Data from the MUXADC

The mode of the MUX is checked and, if it is not set to Random mode the read request is rejected with an appropriate error code set in the user status block. Otherwise MUX channels are read in sequence starting with channel number zero and the contents are returned to the user buffer. Each datum (equal two bytes) consists of 12 bit data and 4 bits MUX channel indicator. For example if a user wants to read 20 channels and MUX channel one is connected to an extender, one datum is taken from MUX channel zero and the remaining 19 data are taken from MUX channel one. Thus the user must differentiate between MUX channel numbers and logical channel numbers.

6.4.2.4 IO.RRD - Random Read Data from the MUXADC

This function enables the user to determine his own sequence in reading logical channels. The logical channel numbers are specified in the user buffer as integers, where each word holds one number. On completion of the request, the words are overwritten with the contents of the previously specified channels. The user is reminded that one physical MUX channel attached to an extender is considered as 32 logical channels, where all 32 data are marked as deriving from that MUX channel.

6.5 Status Returns and Error Handling

When a QIO function is completed, status information about the functions is returned in the I/O status block as specified in the QIO macro call.

6.5.1 First Status Word (low byte)

The error codes listed below may be returned by the MUX Driver.

Code	Meaning
IS.SUC	A QIO for IO.INI, IO.TER, IO.RVB or IO.RRD was successfully completed.
IE.IFC	A function code other than IO.INI, IO.TER IO.RVB or IO.RRD or the standard functions was encountered.
IE.BAD	Returned from IO.INI, IO.TER, IO.RVB or IO.RRD if any bad parameters were encountered.
IE.OFL	Returned from IO.INI, IO.TER, IO.RVB or IO.RRD if module(s) is(are) not at the given CAMAC station or the crate is offline.
IE.IDS	Returned from IO.RVB or IO.RRD if the MUX was not set to Random mode by a prior IO.INI operation.

6.5.2 First Status Word (high byte)

DIOS drivers use this byte to group error conditions. Bit n on selects error code group n+1. All error codes used by the MXDRV belong to error code group zero.

6.5.3 Second Status Word

This word shows in all cases the number of bytes actually transferred to or from the user buffer.

CHAPTER 7

PGDRV: Periodic Pulse Generator

7.1 Introduction

The Periodic Pulse Generator (PPG) is a single-width CAMAC module which delivers a pre-programmed sequence of pulse bursts of the following form:

Each burst consists of z pulses separated by intervals of $t*m$ micro-seconds where

$0 < z < 4096.$
$0 < t < 1024.$

and m is a range multiplier equal to 1, 10, 100, or 1000. There may be up to 16 bursts in the sequence. The sequence may be started by a CAMAC command or by an external trigger signal. The pulses are delivered at a LEMO socket on the front panel. They have a potential of -5 volts, width 330 nano-seconds, and 50 Ohm output impedance.

The DIOS driver for the PPG, PGDRV, allows the user to program the PPG with a pulse sequence defined in a more intuitive way. A sequence of up to 16 bursts is defined by giving for each burst its duration and the number of pulses it contains. The driver decomposes these burst specifications into a sequence in a hardware-compatible form which reproduces the requested sequence up to rounding errors.

The rounding errors may be compensated during the data analysis by reading back the exact sequence with the function IO.RVB processed by the driver. The pulse sequence is read back from the PPG, converted and returned to the user buffer in a format identical to that in which the initial values were specified. If the rounding errors (between .03 and .3 percent) can be neglected, this function need not be used.

7.2 Loading the Module

The module is loaded by either of the two macro calls:

```
QIO$S    #IO.LOD,#$LDR,....,<mcb,lmcb,lun>
```

or

```
LOAD     lun,mcb,sts,flg
```

The use of the macros and meaning of the arguments are given in the DIOS Operations Manual.

7.2.1 MCB Format

In the macro calls above, mcb is the address of the Module Control Block, described in the DIOS Operations Manual. The device specific portions of the MCB should be set as follows:

Offset	Contents
M.TYP	"PG" -- 2-letter module type code for PG.
M.UNIT	Unit number of PG.
M.ACP	2-Letter code of ACP containing the PG driver.
M.CTL	Control bits, set as follows: MC.CAM=1 Indicates the PG is a CAMAC module. MC.INT=0 Indicates interrupt service is not required by the PG.
M.ADR	CAMAC address in BCNA format (A=0) of the module.

7.3 Unloading the Module

The module is unloaded by either or the two macro calls:

```
QIO$S    #IO.UNL,#lun,...
```

or

```
UNLOAD   lun,sts,flg
```

The use of these macros and the meaning of the arguments are given in the DIOS Operations Manual.

7.4 QIO Functions to the Loaded Module

This section summarizes the standard and device-specific QIO requests processable by the driver.

7.4.1 Standard Functions

	Format	Function
QIO$S	#IO.VAT,...	;Attach Module
QIO$S	#IO.VDT,...	;Detach Module
QIO$S	#IO.KIL,...	;Cancel I/O on Module
QIO$S	#IO.UNL,...	;Unload Module

7.4.2 Module-specific Functions

	Format	Function
QIO$S	#IO.INI,...,<ddp,lpm>	;Initialize Module
QIO$S	#IO.TER,...,<0,2>	;Terminate Module
QIO$S	#IO.RVB,...,<buf,lbuf>	;Read data from Module.

where

- ddp is the address of a block of device-dependent parameters defining the mode in which the PPG is initialized.
- lpm is the length of the block in bytes.
- buf is the address of the buffer into which data are read.
- lbuf is the length of the data buffer in bytes.

7.4.2.1 IO.INI - Initialize the PPG

IO.INI passes a buffer specified in the parameter list of the QIO containing device-dependent parameters specifying the mode of initialization. The buffer has the following format:

0	TIME	[E]	Floating point value of the duration (in seconds) of the first pulse burst.
4	PULSE	[E]	Floating point value of the number of pulses to be output in the first burst.
			....
4i	TIME	[E]	Duration of the (i+1)th pulse burst.
4i+4	PULSE	[E]	Number of pulses in the (i+1)th burst.
			.... (Up to a maximum of 16 bursts).

The driver then converts these values to hardware-compatible burst specifiers and loads them into the module with the following algorithm:

1. Find the interval between pulses desired by dividing the duration by the number of pulses.

2. Find the proper range for the multiplier m as follows:

   For 1 micro-second < interval < 1024 micro-seconds, m=1
   For 1.03 milli-seconds < interval < 10.24 milli-seconds, m=10
   For 10.3 milli-seconds < interval < 102.4 milli-seconds, m=100
   For 103 milli-seconds < interval < 1024 milli-seconds, m=1000

3. Convert the interval to an integer value expressing the time in units of m micro-seconds.

4. Obtain the logarithm (10) of m; code this value along with the interval in units of m micro-seconds into a single word.

5. Load the number of pulses into the corresponding subaddress with a CAMAC Write command.

6. Load the time interval word into the proper subaddress with a CAMAC Write command.

After loading the pulse sequence, the driver enables the external trigger of the PPG and enables the pulse outputs. The QIO request is then finished, leaving the module armed to produce the pulses on receipt of the external trigger signal.

7.4.2.2 IO.TER - Terminating Operations on the PPG

The pulse output is arrested if in progress and the first pulse number register is cleared to prevent further triggers from starting a burst sequence.

7.4.2.3 IO.RVB - Read Data from the PPG

The pulse sequence actually loaded is read from the PPG with CAMAC read commands and returned to the user buffer in pairs of floating point numbers identical in format to the parameter buffer passed to the module on initialization. The user program may use these values as the actual time intervals and numbers of pulses produced by the PPG.

7.5 Status Returns and Error Handling

When a QIO function is completed, status information about the functions is returned in the I/O status block if specified in the QIO macro call.

7.5.1 First Status Word (low byte)

The error codes listed below may be returned by the PPG Driver.

Code	Meaning
IS.SUC	A QIO for IO.INI, IO.TER, or IO.RVB has been successfully completed.
IE.IFC	A function code other than IO.RVB, IO.INI, IO.TER or the standard funtions was encountered.
IE.BAD	Returned from IO.INI if any of the following are true:

1. An illegal burst duration was specified. The time must lie in the range

   1 micro-second <= duration <= 4195 seconds.

2. An illegal pulse number was specified. The limits are:

   1 <= number of pulses <= 4096.

3. The time between pulses is less than 1 micro-second or greater than 1.024 seconds.

IE.OFL Returned from IO.INI, IO.RVB or IO.TER if the module is not at the given CAMAC station or the crate is offline.

IE.EOV Returned from IO.INI if the number of pulse bursts exceeds 16. This error code is also returned from IO.RVB if the user requested more data than the total number of pulse bursts actually defined, at 8 bytes per defined burst.

IE.DAO Returned from IO.RVB in case the user requested fewer than the number of bytes needed to define all the actual pulse bursts.

IE.IDS Returned from IO.RVB if the module was producing pulses at the time the request was processed.

7.5.2 First Status Word (high byte)

DIOS drivers use this byte to group error conditions. Bit n on selects error code group n+1.

7.5.3 Second Status Word

On IO.INI, the second status word contains the total number of pulses actually defined. For all errors but IE.EOV, this will be zero. For IE.EOV and IS.SUC, the number will be less than or equal to the total number requested. On IO.RVB, the second status word contains the total number of bytes actually passed to the user buffer, at 8 bytes per defined burst.

CHAPTER 8

QDDRV: Ortec Charge Digitizer

8.1 Introduction

The QD808 Charge Digitizer is a single width CAMAC module with eight charge sensitive 8 bit analog-to-digital converters (ADC's) that operate with a common gate input. The outputs of the 8-bit scalers are paired to form 16-bit data words (QD808 Module Registers). The pairing pattern is scalers 0 and 4, 1 and 5, 2 and 6 and 3 and 7. Within each pair, the output from the higher numbered scaler is fed to the data way lines R1 to R8 and from the other scaler to R9 to R16. Hardware characteristics, specifications and operation may be obtained from the reference documentation.

The DIOS driver for the QD808, QDDRV, operates on the module as follows: Input from the QD808 CAMAC module is done via interrupts (LAM's). More than one datum (equal to one byte) may be read with a single QIO. Because one signal to the common input gate produces a maximum of 8 significant digitized values, it must be kept in mind, that there must be [(n-1)modulo 8]+1 pulses to the common input gate to obtain n data. For example, three pulses are required for the successful completion of a request to read 20 values. Data are returned in a ordered manner, starting with the datum of scaler zero, followed by the datum of scaler one and so on.

8.2 Loading the Module

The module is loaded by either of the two macro calls:

QIO$S #IO.LOD,#$LDR,....,<mcb,lmcb,lun>

or

LOAD lun,mcb,sts,flg

The use of the macros and meaning of the arguments are given in the DIOS Operations Manual.

8.2.1 MCB Format

In the macro calls above, mcb is the address of the Module Control Block, described in the DIOS Operations Manual. The device specific portions of the MCB should be set as follows:

Offset	Contents
M.TYP	"QD" -- 2-letter module type code for QD808.
M.UNIT	Unit number of QD808.
M.ACP	2-Letter code of ACP containing the QD808 driver.
M.CTL	Control bits, set as follows: MC.CAM=1 Indicates the QD808 is a CAMAC module. MC.INT=1 Indicates interrupt service is required by the QD808.
M.ADR	CAMAC address in BCNA format (A=0) of the module.

8.3 Unloading the Module

The module is unloaded by either of the two macro calls:

QIO$S #IO.UNL,#lun,...

or

UNLOAD lun,sts,flg

The use of these macros and the meaning of the arguments are given in the DIOS Operations Manual.

8.4 QIO Functions to the Loaded Module

This section summarizes the standard and device-specific QIO requests processable by the driver.

8.4.1 Standard Functions

	Format	Function
QIO$S	#IO.VAT,...	;Attach Module
QIO$S	#IO.VDT,...	;Detach Module
QIO$S	#IO.KIL,...	;Cancel I/O on Module
QIO$S	#IO.UNL,...	;Unload Module

8.4.2 Module-specific Functions

	Format	Function
QIO$S	#IO.INI,...,<ddp,lpm>	;Initialize QD808
QIO$S	#IO.TER,...,<0,2>	;Terminate QD808
QIO$S	#IO.RVB,...,<buf,lbuf>	;Read data from QD808

where

ddp	is the address of a block of device-dependent parameters defining the mode in which the QD808 is initialized.
lpm	is the length of the block in bytes.
buf	is the address of the buffer into which data are read.
lbuf	is the length of the data buffer in bytes.

8.4.2.1 IO.INI - Initialize the QD808

IO.INI passes a buffer specified in the parameter list of the QIO containing device-dependent parameters specifying the mode of initialization. The buffer consists of four bytes which are at present not significant, but are reserved for future use. The driver disables the interrupt logic of the module and clears the four module registers.

8.4.2.2 IO.TER - Terminate Operations on the QD808

The same action is taken as with I/O operation code IO.INI.

8.4.2.3 IO.RVB - Read Data from the QD808

The interrupt logic of the module is enabled. On occurance of a LAM signal all four module registers are read, cleared and the contents stored locally. If the user count is not satisfied, LAM's are reenabled and the next interrupt is accepted. Once the user count is satisfied, the locally stored data are returned to the user.

8.5 Status Returns and Error Handling

When a QIO function is completed, status information about the functions is returned in the I/O status block if specified in the QIO macro call.

8.5.1 First Status Word (low byte)

The error codes listed below may be returned by the QD808 Driver.

Code	Meaning
IS.SUC	A QIO for IO.INI, IO.RVB or IO.TER was successfully completed.
IE.IFC	A function code other than IO.INI, IO.RVB, IO.TER or the standard functions was encountered.
IE.OFL	Returned from IO.INI, IO.TER or IO.RVB if the module is not at the given CAMAC station or the crate is offline.
IE.UPN	Returned from IO.RVB if insufficient dynamic storage is available for storing data locally. If this error occurs permanently or often, the request count should be reduced or the system pool space should be increased.
IE.ABO	Returned from IO.KIL or from IO.RVB if power failure has occured.

8.5.1.1 First Status Word (high byte) -

DIOS drivers use this byte to group error conditions. Bit n on selects error group n+1. All error codes returned by the QDDRV belong to error code group zero, except for code IE.UPN, which belongs to DIOS error code group one.

8.5.2 Second Status Word

This word shows in all cases the number of bytes actually transferred to the user buffer.

8.6 Programming Hints

CAUTION

The user should notice that there has to be a delay between two signals to the common gate input to obtain proper measurement results with the QD808. This delay depends on the computer on which the QDDRV runs. For a PDP11/45 the minimum delay is 1 msec, for a PDP11/20 the delay time should be about 1.6 msec.

For proper functioning the QD808 module should be arranged within a CAMAC crate, so that it will be cooled optimally.

CHAPTER 9

TGDRV: Experiment Trigger Input - DR-11 Interface

9.1 Introduction

The TRIGGER (TG) module merely consists of a interface to a PDP11, so the TGDRV is designed to operate with standard PDP11 interfaces DR11A and DR11C. Its purpose is not to perform any I/O via the interface but to inform the user about external events, which are signalled via the interrupt logic of the interface. Hardware characteristics, specifications and operation may be obtained from the reference documentation.

The DIOS driver for the TG, TGDRV, operates on the module as follows: Initialization and termination functions cause the interrupt logic to be disabled and output and input buffer to be cleared. A read function enables the interrupt logic. On occurance of an interrupt further interrupts are disabled, buffer registers are cleared and the read request is finished, thus informing the user about the external event by setting his WAIT flag.

9.2 Loading the Module

The module is loaded by either of the two macro calls:

```
QIO$S    #IO.LOD,#$LDR,....,<mcb,lmcb,lun>
```

or

```
LOAD     lun,mcb,sts,flg
```

The use of the macros and meaning of the arguments are given in the DIOS Operations Manual.

9.2.1 MCB Format

In the macro calls above, mcb is the address of the Module Control Block, described in the DIOS Operations Manual. The device specific portions of the MCB should be set as follows:

Offset	Contents
M.TYP	"TG" -- 2-letter module type code for TG.
M.UNIT	Unit number of TG.
M.ACP	2-Letter code of ACP containing the TG driver.
M.CTL	Control bits, set as follows: MC.CAM=0 Indicates the TG is not a CAMAC module. MC.INT=1 Indicates interrupt service is required by the TG.
M.ADR	CSR address of the module.

9.3 Unloading the Module

The module is unloaded by either of the two macro calls:

```
QIO$S    #IO.UNL,#lun,...
```

or

```
UNLOAD  lun,sts,flg
```

The use of the macros and the meaning of the arguments is given in the DIOS Operations Manual.

9.4 QIO Functions to the Loaded Module

This section summarizes the standard and device-specific QIO requests processable by the driver.

9.4.1 Standard Functions

	Format	Function
QIO$S	#IO.VAT,...	;Attach Module
QIO$S	#IO.VDT,...	;Detach Module
QIO$S	#IO.KIL,...	;Cancel I/O on Module
QIO$S	#IO.UNL,...	;Unload Module

9.4.2 Module-specific Functions

	Format	Function
QIO$S	#IO.INI,...,<ddp,lpm>	;Initialize TG
QIO$S	#IO.TER,...,<0,2>	;Terminate TG
QIO$S	#IO.RVB,...,<buf,lbuf>	;Await external event

where

- ddp is the address of a block of device-dependent parameters defining the mode in which the Triggr is initialized.
- lpm is the length of the block in bytes.
- buf is the address of the buffer into which data are read.
- lbuf is the length of the data buffer in bytes.

9.4.2.1 IO.INI - Initialize the Triggr

IO.INI passes a buffer specified in the parameter list of the QIO containing device-dependent parameters specifying the mode of initialization. The buffer consists of four bytes which are at present not significant, but are reserved for future use. The driver disables the interrupt logic of the interface and clears its input and output buffer registers.

9.4.2.2 IO.TER - Terminate Operations on the Trigger

The same action is taken as with I/O operation code IO.INI.

9.4.2.3 IO.RVB - Read Data from the Trigger

The interrupt logic of the module is enabled. On occurance of an interrupt, further interrupts are locked out. Input and output buffer registers of the interface are cleared and the request is finished.

9.5 Status Returns and Error Handling

When a QIO function is completed, status information about the functions is returned in the I/O status block if specified in the QIO macro call.

9.5.1 First Status Word (low byte)

The error codes listed below may be returned by the TG Driver.

Code	Meaning
IS.SUC	A QIO for IO.INI, IO.RVB or IO.TER was successfully completed.
IE.IFC	A function code other than IO.INI, IO.RVB, IO.TER or the standard functions was encountered.
IE.ABO	Returned from IO.KIL if a read operation was in progress.

9.5.2 First Status Word (high byte)

DIOS drivers use this byte to group error conditions. Bit n on selects error group n+1. All error codes returned by the TGDRV belong to error code group zero.

9.5.3 Second Status Word

This word shows in all cases the number of bytes actually transferred to the user buffer.

CHAPTER 10

TSDRV: Culham Time Sequence Generator

10.1 Introduction

The Culham Time Sequence Generator (TSG) is a single-width CAMAC module which delivers a pre-programmed sequence of pulse bursts of the following form:

Each burst consists of 2**m pulses separated by intervals of 2**n*100 nano-seconds where $0<m<8$ and $0<n<16$. There may be up to 32 bursts in the sequence. The sequence may be started by a CAMAC command or by an external trigger signal. The pulses are delivered in parallel at 6 LEMO sockets on the front panel. They have a potential of -5 volts, width 50 nano-seconds and 50 Ohm output impedance.

The DIOS driver for the TSG, TSDRV, allows the user to program the TSG with a pulse sequence defined in a more intuitive way. A sequence of up to 7 bursts is defined by giving for each burst its duration and the number of pulses it contains. The driver decomposes these burst specifications into a sequence of a hardware-compatible form which approximately reproduces the requested sequence.

Since the interval between pulses must be chosen from 15 discrete powers of 2 (x 100 nano-seconds), not every combination of burst duration and pulse number can be reproduced. The driver attempts to fit the next lower number of pulses and next lower pulse frequency to come as close to the time interval as possible. This means that the actual sequence delivered is different from the one requested. A read function is provided to make the correct sequence available to the user's data analysis programs.

10.2 Loading the Module

The module is loaded by eithr of the two macro calls:

```
QIO$S   #IO.LOD,#$LDR,....,<mcb,lmcb,lun>
```

or

```
LOAD    lun,mcb,sts,flg
```

The use of the macros and meaning of the arguments are given in the DIOS Operations Manual.

10.2.1 MCB Format

In the macro calls above, mcb is the address of the Module Control Block, described in the DIOS Operations Manual. The device specific portions of the MCB should be set as follows:

Offset	Contents
M.TYP	"TS" -- 2-letter module type code for TS.
M.UNIT	Unit number of TS.
M.ACP	2-Letter code of ACP containing the TS driver.
M.CTL	Control bits, set as follows: MC.CAM=1 Indicates the TS is a CAMAC module. MC.INT=0 Indicates interrupt service is not required by the TS.
M.ADR	CAMAC address in BCNA format (A=0) of the module.

10.3 Unloading the Module

The module is unloaded by either of the two macro calls:

```
QIO$S   #IO.UNL,lun,...
```

or

```
UNLOAD  lun,sts,flg
```

The use of the macros and the meaning of the arguments are given in the DIOS Operations Manual.

10.4 QIO Functions to the Loaded Module

This section summarizes the standard and device-specific QIO requests processable by the driver.

10.4.1 Standard Functions

	Format	Function
QIO$S	#IO.VAT,...	;Attach Module
QIO$S	#IO.VDT,...	;Detach Module
QIO$S	#IO.KIL,...	;Cancel I/O on Module
QIO$S	#IO.UNL,...	;Unload Module

10.4.2 Module-specific Functions

	Format	Function
QIO$S	#IO.INI,...,<ddp,lpm>	;Initialize Module
QIO$S	#IO.TER,...,<0,2>	;Terminate Module
QIO$S	#IO.RVB,...,<buf,lbuf>	;Read data from Module.

where

ddp is the address of a block of device-dependent parameters defining the mode in which the TSG is initialized.
lpm is the length of the block in bytes.
buf is the address of the buffer into which data are read.
lbuf is the length of the data buffer in bytes.

10.4.2.1 IO.INI - Initialize the TSG

IO.INI passes a buffer specified in the parameter list of the QIO containing device-dependent parameters specifying the mode of initialization. The buffer has the following format:

0	PULSE	[E]	Duration (in seconds) of the first pulse burst.
4	FREQ	[E]	Number of pulses to be output in the first burst
			.....
4i	PULSE	[E]	Duration of the (i+1)th pulse burst.
4i+4	FREQ	[E]	Number of pulses in the (i+1)th burst.

The driver then converts these values to hardware-compatible burst specifiers and loads them into the module with the following algorithm:

1. Find the interval between pulses desired in units of 100 nano-seconds, by dividing the duration by the number of pulses.

2. Find the next higher pure binary multiple of 100 nano-seconds, i.e., the next larger actually realizable pulse spacing interval.

3. Recompute the total number of pulses needed to fill the duration of the burst at this spacing interval, by dividing the duration by the new spacing interval.

4. Decompose this number of pulses into a sum of pure binary numbers between 2 and 128, code the exponent of each along with the pulse spacing interval into a hardware burst specifier and load the word into the TSG.

In short, it is attempted to keep the time interval close to the desired value, adjusting the number of pulses downward when needed to match the nearest possible pulse interval.

After loading the pulse sequence, the driver enables the external trigger of the TSG and enables the pulse outputs. The QIO request is then finished, leaving the module armed to produce the pulses on receipt of the external trigger signal.

10.4.2.2 IO.TER - Terminate Operations on the TSG

The trigger input and pulse output of the TSG are disabled so that no further pulse trains can be produced.

10.4.2.3 IO.RVB - Read Data from the TSG

The actually calculated pulse sequence is returned to the user buffer in pairs of floating point numbers identical in format to the parameter buffer passed to the module on initialization. The user program may use these values as the actual time intervals and numbers of pulses produced by the TSG.

NOTE

> After IO.INI, the information from IO.RVB is available only until the next IO.TER or IO.UNL is requested. Any subsequent IO.RVB's will produce an error.

10.5 Status Returns and Error Handling

When a QIO function is completed, status information about the functions is returned in the I/O status block if specified in the QIO macro call.

10.5.1 First Status Word (low byte)

The error codes listed below may be returned by the TSG Driver.

Code	Meaning
IS.SUC	A QIO was successfully completed.
IE.IFC	A function code other than IO.RVB, IO.INI, IO.TER or the standard functions was encountered.
IE.BAD	Returned from IO.INI if any of the following are true: 1. An illegal burst duration was specified. The time must lie in the range: 500 nS <= duration <= 13.4218 seconds. 2. An illegal pulse number was specified. The limits are: 2 <= number of pulses <= 4096. 3. The time between pulses is less than 200 nano-seconds or greater than 3.2768 milli-seconds.
IO.OFL	Returned from IO.INI or IO.TER if the module is not at the given CAMAC station or the crate is offline.
IE.EOV	Returned from IO.INI if the number of pulse bursts exceeds 7, or if more than 32 hardware bursts are needed to reproduce the sequence. In either case, the maximum number of pulses possible is defined, the sequence being cut off at the end. The module is armed to deliver the partial pulse train. On a subsequent IO.RVB, the actual pulse sequence define will be returned. IE.EOV is returned from IO.RVB if the user requested more data than the total number of pulse bursts actually defined, at 8 bytes per defined burst.
IE.DAO	Returned from IO.RVB in case the user requested fewer than the number of bytes needed to define all the actual pulse bursts.
IE.IDS	Returned from IO.RVB if the module was unloaded or terminated since the last IO.INI.

10.5.2 First Status Word (high byte)

DIOS drivers use this byte to group error conditions. Bit n on selects error code group n+1.

10.5.3 Second Status Word

On IO.INI, the second status word contains the total number of pulses actually defined. For all errors but IE.EOV, this will be zero. For IE.EOV and IS.SUC, the number will be less than or equal to the total number requested.

On IO.RVB, the second status word contains the total number of bytes actually passed to the user buffer, at 8 bytes per defined burst.

10.6 Programming Hints

10.6.1 Precautions in Defining a Pulse Burst

Since the actual values loaded into the TSG usually differ from the assigned values, the user should always read back the pulse sequence and make sure the bursts are acceptable before validating any data generated by modules using the pulses from the TSG.

READER'S COMMENTS

NOTE: THIS FORM IS FOR DOCUMENT COMMENTS ONLY.

Did you find errors in this manual? If so, specify by page.

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

Is there sufficient documentation on associated system programs required for use of the software described in this manual? If not, what material is missing and where should it be placed?

Please turn over

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

[ ] Assembly language programmer

[ ] Higher-level language programmer

[ ] Occasional programmer (experienced)

[ ] User with little programming experience

[ ] Student programmer

[ ] Non-programmer

If you desire to have your name put on the PDE documentation mailing list, please indicate so here...... [ ]

NAME________________________________ DATE____________

ORGANIZATION____

STREET____

CITY____

STATE__________________________ ZIP CODE______________

COUNTRY____

RETURN TO:

PDE PROJEKT DATENERFASSUNG
INSTITUTE FOR PLASMAPHYSICS
D-8046 GARCHING
WEST GERMANY