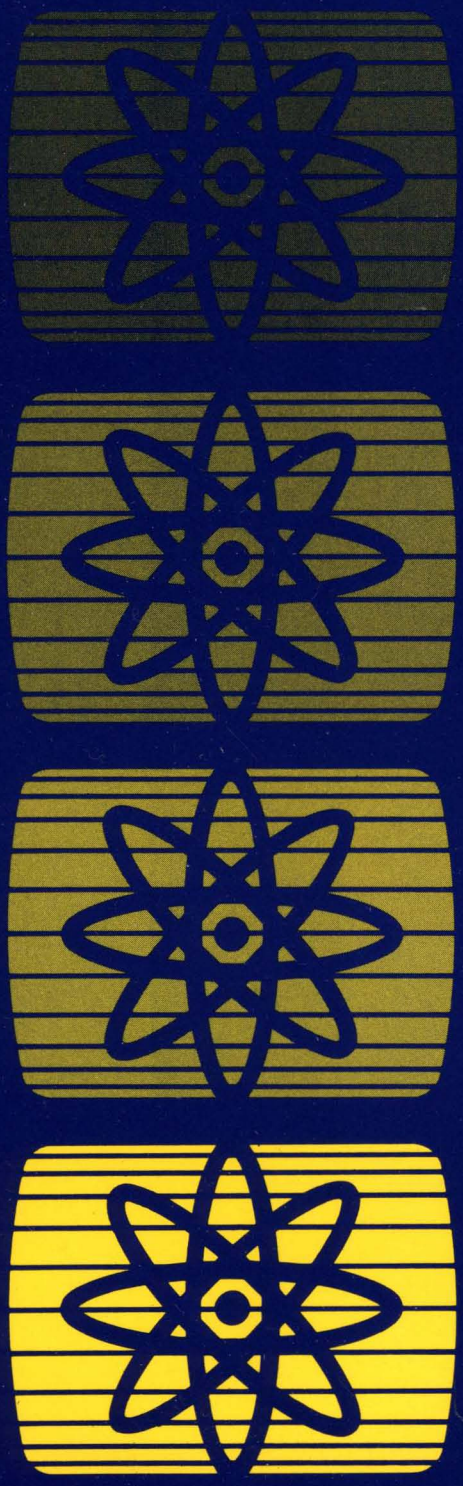


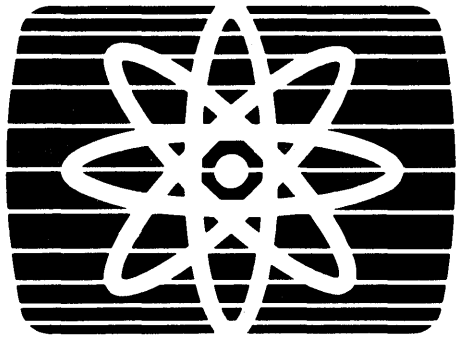
7170

Device Attachment Control Unit

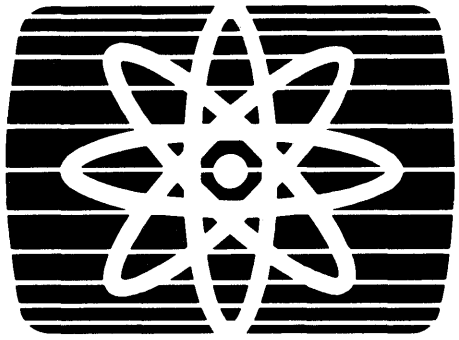
Reference and Operations Manual

SA24-4025-0

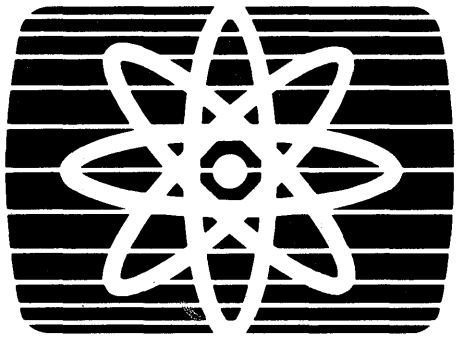
IBM



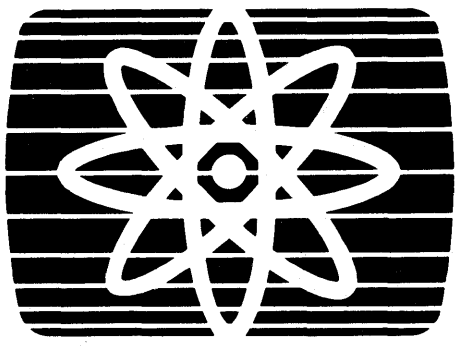
7170



Device Attachment Control Unit



Reference and Operations Manual



SA24-4025-0

IBM

Federal Communications Commission (FCC) Statement

Warning: This equipment generates, uses, and can radiate radio frequency energy and if not installed and used in accordance with the instruction manual, may cause interference to radio communications. It has been tested and found to comply with the limits for a Class A computing device pursuant to Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such interference when operated in a commercial environment. Operation of this equipment in a residential area is likely to cause interference in which case the user at his own expense will be required to take whatever measures may be required to correct the interference.

Third Edition (October 1984)

This edition applies to Release 1.1 of the Device Attachment Control Unit (DACU), and to all subsequent releases of this product until otherwise indicated in new editions or Technical Newsletters.

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM program product in this publication is not intended to state or imply that only IBM's program product may be used. Any functionally equivalent program may be used instead.

Ordering Publications

Requests for IBM publications should be made to your IBM representative or to the IBM branch office serving your locality. Publications are *not* stocked at the address given below. The *7170 Device Attachment and Control Unit Reference and Operations* manual is supplied to the customer when a DACU is purchased.

A form for readers' comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Engineering/Scientific Systems, Department X00, P.O. Box 6, Endicott, NY, U.S.A. 13760. IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Preface

The purpose of this manual is to provide comprehensive documentation for the Device Attachment Control Unit. It covers:

1. Introductory information, including a general description of the hardware and the function of its major logical elements.

2. Programming considerations and examples for both host and local programming.

3. Procedures for installing and operating a DACU as well as procedures for diagnosis and problem resolution.

4. General and physical characteristics and performance considerations.

5. Appendices include sample host programming and reference material.

How this Book is Organized

Among the major divisions of this publication are:

- **Installation Planning** "Chapter 7. Physical Characteristics and Installation Planning Information" on page 137 describes the physical characteristics of the DACU and installation preparation considerations.

- **Customer Set-up and Operating Instructions** see "Chapter 5. Installing and Operating the DACU" on page 91.

- **Maintenance** Recommended techniques for problem analysis and problem resolution are given in "Chapter 6. Customer Problem Analysis and Resolution" on page 105.

- **Host Programming** Techniques and examples are given in "Chapter 3. Programming Considerations" on page 13 as well as in the host programming appendix.

- **Local DACU Programming** Techniques and examples are given in "Chapter 4. Local DACU Programming" on page 59.

About This Manual

This manual is a major re-write of the *7170 Device Attachment Control Unit Reference and Operation* Manual DACU-ROM-0. It is released in conjunction with a hardware change which brings the DACU to level 1.1. The major changes are:

- Inclusion of customer problem determination procedures
- New control space op codes
- Complete listings of host application language libraries HALLS and HALLGE
- Explanation of DACU information and warning messages
- Support of IBM Personal Computer XT
- Support of both Serial I/O ports.

Contents

Figures

Chapter 1. General Product Description

The Device Attachment Control Unit (DACU) is a control unit, adhering to the architecture of the IBM Original Equipment Manufacturers' Interface (OEMI) on an IBM 43XX or 308X Block Multiplexer Channel. The DACU is available in a single configuration (no optional features) which provides interfaces compatible with the UNIBUS[1] parallel, direct memory access interface and the EIA RS-232-C serial interface for the attachment of non-IBM, I/O devices. Control programs provided with the DACU are executed locally by the DACU System Unit (IBM Personal Computer) to support the requirements for I/O associated with attached devices, managing buffering and interrupts in a manner designed to provide flexibility and high overall throughput. Customer provided application- or device-dependent program components may be written and executed locally in the DACU System Unit using supplied macro or subroutine facilities for customer written code.

Throughout this manual the interface which is compatible with the UNIBUS parallel, direct memory access interface will be referred to as the "Parallel I/O" interface. This interface is intended for the attachment of high performance I/O devices. It provides for *emulated* direct memory access I/O transfers to and from host system main storage. Such transfers are buffered in the DACU Interface Unit storage. Facilities for "programmed I/O" are provided as well. The Parallel I/O interface and associated DACU processing handle device interrupts in a manner that simulates I/O handling typical of the environment for which these devices were designed.

EIA RS-232-C is a commonly used interface for attaching serially communicating devices. The DACU includes two such interfaces, providing half and full duplex asynchronous communication.

Throughout this manual, the RS-232-C-compatible, serial interface will be referred to as the "Serial I/O" interface.

---

1 UNIBUS is a registered trademark of the Digital Equipment Corporation. DACU compatibility, as described in this document, is specified by UNIBUS interface definitions as described, for example, in *PDP-11 Bus Handbook,* Digital Equipment Corporation, 1979.

Conceptual Data Flow

The figure below illustrates the *conceptual* flow of data in the DACU Interface Unit.

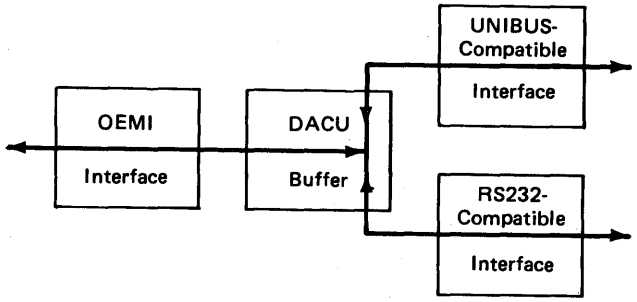


Figure 1. Conceptual DACU Data Flow

Referring to the illustration, there are a number of useful conceptual data flow paths associated with attached UNIBUS-compatible devices:

- With a device Master, a DMA read/write path from/to DACU buffer address space
- Under DACU control, a programmed read/write path from/to the address space of a device
- A Channel read/write path from/to DACU buffer address space
- A path for communicating interrupts from a device to the DACU.

Similar paths exist for managing RS-232-C devices.

Parallel I/O Interface Operations

From an application's point of view, the Parallel I/O interface permits access to a "memory" address space with blocks of 64K bytes. Low memory is reserved for buffer memory, and the top 8K bytes of memory is reserved by convention for registers in attached I/O devices. Given commands for reading or writing blocks of data from the host into the lower 56K address space and a mechanism for transferring control words to the device registers in the 8K address space, DMA transfers in both directions can be performed by the devices.

In addition, a mechanism is needed to signal completion of a DMA transfer by the device. Many devices use an interrupt for this purpose. If the host were to initiate DMA transfers of blocks of data without such a mechanism, it would have to use the alternate method of polling the address space associated with the Parallel I/O device to determine when the DMA transfer was finished.

Both the DMA and interrupt handling functions are provided by the Parallel I/O adapter in the DACU Interface Unit, which is controlled locally by the DACU System Unit processor. Since one of the functions of the processor is to mediate differences between the Parallel I/O interface and IBM channel architectures, it

might be called a "mediator." Another processor function is to provide for interrupt related processing.

Commands are described in a following section which allow the reading and writing of address space related to the Parallel I/O interface and, in addition, allow interrupt handling by host programs.

Serial I/O Operations

The DACU Serial I/O function provides full or half duplex asynchronous communication. Data is sent from the host to a buffer in the DACU Interface Unit. From there it is transmitted a character at a time through one of the serial ports. When the last character is transmitted, the host is interrupted, signaling that the next block of data may be sent. Input through a serial port is managed in a similar fashion.

Configurations

The DACU is available in a single configuration. (That is, there are no options or configurable features.) The configuration permits the attachment of the following classes of non-IBM devices:

1. UNIBUS-compatible I/O devices via conventional techniques of attaching such devices

2. Two Serial I/O devices, one on each of the two DACU Serial I/O ports.

Architectural Considerations

The DACU complies with the following architectural and interface definitions:

1. Channel interface as defined in *IBM System/360 and System/370 I/O Interface Channel to Control Unit Original Equipment Manufacturers' Information,* GA22-6974-5, February 1981

2. UNIBUS interface definitions as described, for example, in *PDP-11 Bus Handbook,* Digital Equipment Corporation, 1979

3. EIA Standard RS-232-C and CCITT Standard V.24 for Serial Binary Data Exchange.

External User Interface

The external user interface is formed by definitions of the following elements:

1. DACU control unit commands

2. DACU Parallel I/O interface commands

3. DACU Serial I/O commands

4. DACU local programming interface.

Compatibility with Programming Support

The DACU was designed to operate compatibly with two IBM operating system environments — OS/VS2 and VM/CMS. Specific areas of compatibility include:

1. Use of a subset of IBM 2250/3250 control unit commands

2. Standard sense information in first two sense bytes

3. Treatment of attentions.

Also, it was designed to be useable with the Graphics Access Method (GAM) available with those operating systems. I/O support can also be provided via EXCP (Execute Channel Program) services or the Start I/O (SIO) instruction.

Customer and Service Information Highlights

The DACU Interface Unit is intended to be installed and serviced by customer personnel. Maintenance and repair is provided by element exchange through the designated repair depot under the condition of the warranty or under the terms of additional maintenance agreements as appropriate. Diagnostic programs provided with the DACU include means for assisting in isolating problems to the control unit versus problems in attached I/O devices and in isolating control unit problems to a replaceable element.

Chapter 2. Hardware Description

Functionally, the Device Attachment Control Unit is defined by its interface boundaries. As a control unit, it attaches to an IBM 43XX or 308X block-multiplexer channel via the OEMI. (That is, as seen by the IBM 43XX or 308X channel, the DACU appears to be a standard control unit.) Functioning as an "interface transformation mechanism," the DACU maps data at that interface to one of the other two interfaces (Parallel I/O or Serial I/O) as specified by control unit commands.

Particularly significant is the fact that the IBM channel I/O architecture and UNIBUS I/O architecture are inherently incompatible. The key difference between them is that, with respect to the OEMI, the channel is always "master" of the interface, and control units and devices are always "slaves"; an I/O device can never control the flow of information from processor main storage to itself. With the UNIBUS architecture, any device on the bus (processor or I/O device) can be "master" once granted bus "mastership" by the bus arbitration mechanism.

Because of this basic incompatibility, the DACU capability must include a processing mechanism that serves a mediation role — mediating the architectural differences. That is, with respect to the OEMI, the DACU is always "slave" to the channel; with respect to the Parallel I/O interface, the DACU may be master or slave to a UNIBUS-compatible I/O device. This mediation or control role is provided by the DACU System Unit — an IBM Personal Computer which is coupled to and controls the DACU Interface Unit.

Basic Data Flow

The basic DACU data flow is illustrated below.

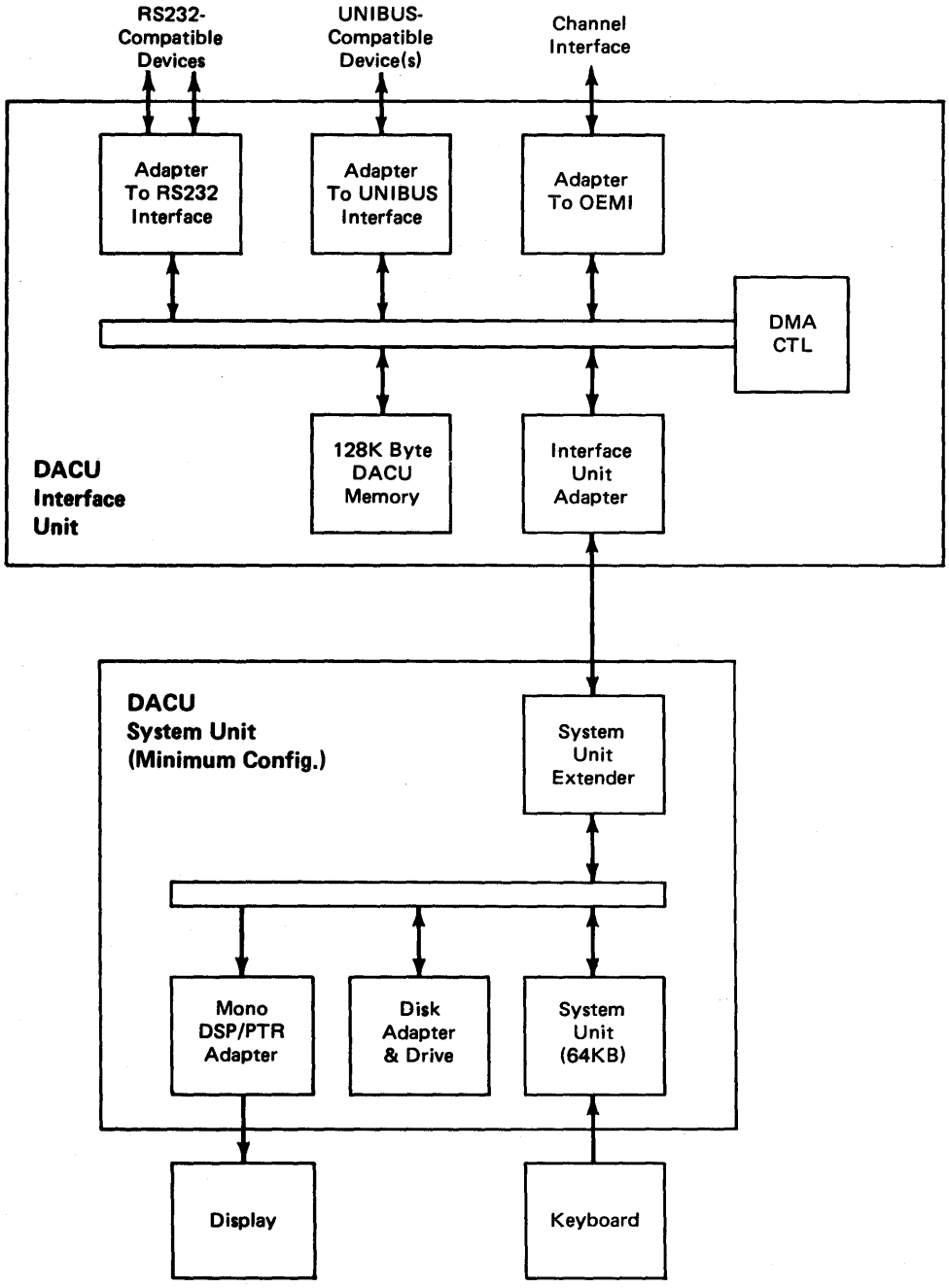


Figure 2. Basic DACU Data Flow

Communication among the functional units in the data flow is carried out on a 16-bit DACU bus.

The functional units include:

- 128K Byte DACU Interface Unit Memory
- OEMI Adapter
- Parallel I/O Interface Adapter
- Serial I/O Adapter
- Interface Unit Adapter
- System Unit Extender.

The Interface Unit Adapter handles communication between the DACU Interface Unit bus and the DACU System Unit via a cable. In doing so, it manages the data buffering and coordination required to adapt the 16-bit DACU Interface Unit bus to the 8-bit data required by the DACU System Unit.

The System Unit Extender is provided for installation in DACU System Unit Expansion Slot. This card accepts the cable from the DACU Interface Unit and manages 8-bit data communication with the DACU System Unit I/O Channel.

Data Flow Unit Descriptions

DACU System Unit

The DACU System Unit is an IBM Personal Computer model 5150, or an IBM Portable Personal Computer model 5155, or an IBM Personal Computer XT model 5160 that is cabled to the DACU Interface Unit. The DACU System Unit provides all of the processing capability required to coordinate data transfers among the DACU functional elements, including the interface adapter hardware. The System Unit controls an interface adapter by communicating commands to registers in each of the adapters. In addition, it provides the capability for developing and executing application- and device-dependent programs that participate in the coordination of data flow.

The following DACU System Unit configuration is required *as a minimum* to support applications in production use:

1. IBM Personal Computer System Unit with 64KB Memory and Keyboard
2. 5-1/4 inch Diskette Drive Adapter
3. 5-1/4 inch Diskette Drive
4. IBM Monochrome Display/Printer Adapter
5. IBM Monochrome Display
6. IBM PC DOS Diskette.

(The DACU may be powered on, initialized, and brought on-line — "enabled" — without the keyboard and display. Nor are the keyboard and display essential for production operation. They are required, however, for running diagnostic test programs and must be restored for that purpose.)

Customers may wish to configure expanded DACU System Units (e.g., additional memory) for those DACU configurations that are to be used for application code development. (For example, a customer may wish to install a configuration suitable for running the IBM PC Macro Assembler or PASCAL Compiler for application development efforts.)

Provided with the DACU Interface Unit is a diskette containing executable PC code which provides support functions for the DACU. These functions provide control (drivers) for the OEMI, Parallel I/O, and Serial I/O interface adapters; a Dispatcher function; and a linkage for customer provided code. (Default code is provided to handle common application functions for those cases where unique application- or device-dependent code is not required.)

DACU Interface Unit

DACU Interface Unit storage comprises 128K bytes of random access memory organized as 64K words (16 bits per word). Interface to the system data flow is via the 16-bit data bus. During alternate byte accesses of storage, the DACU System Unit will access either the left or the right byte-wide bank of RAM. During a DMA flow of data to the Parallel I/O interface under the control of an attached I/O device, both banks will communicate with the Parallel I/O interface adapter via the 16-bit system bus.

A DACU memory map is shown below, and it is followed by a brief description of the normal use of areas in the DACU Interface Unit 128K memory.

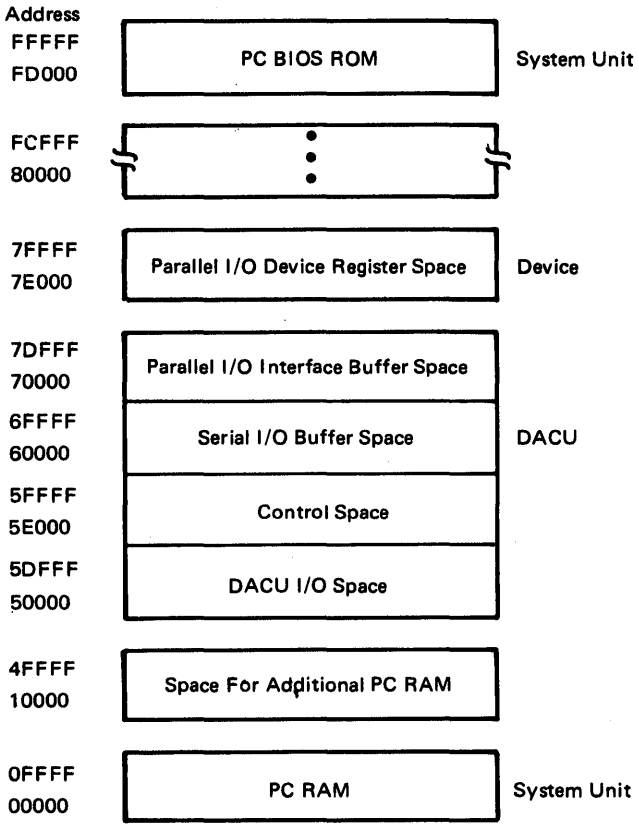


Figure 3. DACU Memory Map.

Three areas of DACU Interface Unit storage (totaling 128K bytes) plus two additional *address spaces* normally are used for the following purposes:

1. DACU I/O SPACE is a 56K address space associated with DACU hardware facilities (e.g., Parallel I/O interface adapter control register). (This is not a region in the DACU 128K memory; rather, it is an *address space* associated with DACU hardware facilities.)

2. CONTROL SPACE is an 8K area in the DACU Interface Unit memory intended to hold control blocks used by the DACU control program and for commands to be interpreted by Parallel I/O interface driver code. Each logical device has a reserved section of this memory. Control space functions are described fully in "DACU Control Space Interpreter" on page 41. In addition, a storage layout of the control space area is illustrated in "Control Space Areas" on page 62.

3. Serial I/O BUFFER SPACE is a 64K area in the DACU memory intended to be used for buffering communications to devices attached to the Serial I/O ports.

4. Parallel I/O INTERFACE BUFFER SPACE is a 56K area in the DACU memory intended to be used for buffering data transfers to and from devices attached to the Parallel I/O interface.

5. Parallel I/O DEVICE ADDRESS SPACE is an 8K address space associated with Parallel I/O devices. The meaning and use of specific addresses varies from device to device, and device manufacturer's documentation should be consulted. (This is not a region in the DACU 128K memory; rather, it is an *address space* associated with facilities in UNIBUS-compatible devices.)

DMA Control

Control of data flow during DMA transfers among system components is established by the DMA control element. It computes word (two bytes) addresses. Four DMA channels are utilized.

- One channel is used to refresh memory.
- One channel is utilized for communication between the OEMI adapter and RAM on a byte basis.
- Two channels are required to establish full duplex communication between the Port 1 Serial I/O adapter and RAM.

OEMI Adapter

The OEMI adapter, controlled by the DACU System Unit, performs all control unit functions associated with the host I/O channel interface. It is compatible with the block multiplexer (non-data streaming) channel interface on the following IBM processor models:

- IBM 4321
- IBM 4331 Model Groups 2 and 11
- IBM 4341 Model Groups 9, 10, 1, 11, 2, and 12
- IBM 4361 Model Groups 4 and 5
- IBM 4381 Model Groups 1 and 2
- IBM 3081 Model Groups D, G, and K
- IBM 3083 Model Groups E, B, and J
- IBM 3084 Model Group Q.

(In the interest of brevity, when the word "host" is used in this manual, it is intended to mean one or more of the above IBM processor models.)

Flow of data involving the OEMI adapter consists of DMA transfers to DACU Interface Unit storage. OEMI transfers are byte transfers to a selected byte-wide bank of DACU storage.

Parallel I/O Interface Adapter

The Parallel I/O interface adapter maps the 16-bit DACU Interface Unit bus to the Parallel I/O interface. This adapter allows a DMA control facility in a UNIBUS-compatible I/O device to access DACU storage on a word (two byte) basis.

Serial I/O Adapter

The Serial I/O Adapter has facilities for serial data transfer under the control of the DACU System Unit. It utilizes byte transfers with DACU storage on a bank select basis.

Byte Coordination

In the IBM host architecture, a two byte entity (half word) is obtained by accessing the high order byte from a given storage address and the low order byte from the next address. Contrast that with the architectures for minicomputers commonly used with UNIBUS-compatible I/O devices and for the 8088 microprocessor (used in the System Unit), where the bytes are reversed. The DACU accounts for this architectural difference by coordinating the flow of bytes based on the logical device addressed on the channel (control unit, Parallel I/O interface facility, or Serial I/O port).

For a data transfer from the channel to an Serial I/O port, data is transferred into DACU storage, byte by byte. The result is that bytes in DACU storage will appear exactly in the same order as bytes in the host.

For a data transfer from the channel to either the control unit or the Parallel I/O interface facility, data is transferred into DACU storage on a byte pair basis. In this case, data transfers from the channel include a hardware reversal of a byte pair on its way into DACU storage. Subsequent word operations involving either the 8088 in the DACU System Unit or a Parallel I/O device will have the bytes in the proper position as a result. Data transfers in the reverse direction operate similarly. This byte swapping occurs with default operation of the Parallel I/O interface facility. However, it is possible to switch, under program control, between the word mode (where the bytes are swapped) and the byte mode. See "Chapter 3. Programming Considerations" on page 13 for details.

Chapter 3. Programming Considerations

The commands communicated by the host channel to the logical devices associated with the DACU must be understood if a DACU with attached physical I/O devices is to be used properly. These commands are summarized generically in the following section.

Subsequent sections will describe the interpretations of the commands by the individual logical devices — Control Unit, Parallel I/O, Serial I/O.

In addition, the options available for host I/O programming are described. Moreover, in cases where application- or device-specific code is to be added to be executed in the DACU itself, the customer interface to DACU control programs must also be understood. This interface is described as well.

Generic Logical Device Commands

Immediately below is a generic description of the interpretation of commands by the several logical devices associated with DACU I/O device addresses. Interpretations for the specific logical devices associated with specific addresses are described in subsequent sections.

Functionally, the DACU can be described from a number of points of view:

- Its behavior as a control unit and a set of logical I/O devices on the channel interface
- Its behavior as perceived by an attached I/O device on the Parallel I/O interface or Serial I/O interface
- Its behavior as perceived at the DACU user interface for customer-provided DACU System Unit code.

In this section, the DACU will be described in terms of its behavior as a control unit and its relationship with devices attached via the Parallel I/O and Serial I/O interfaces.

The DACU occupies four separate logical I/O device addresses on the IBM channel. Although these four logical devices physically share common hardware — components of the DACU — commands directed to a specific device address cause results generally unique to that logical device. The four logical devices are:

Device Address	0:	Control unit itself and associated DACU memory address space.
Device Address	1:	Serial Port One and associated DACU memory address space.
Device Address	2:	Serial Port Two and associated DACU memory address space.
Device Address	4:	Parallel I/O interface adapter and associated DACU memory address space.

Allowable commands for these four logical devices will be defined in a separate section below. However, there are degrees of common treatment of the commands by the different logical devices, and these are described immediately below.

Each of the logical devices responds to the following set of commands:

Command	Cmd Code	Description
No-Operation	(03)	Performs no operation. Typically, the command would be used to obtain status information for the addressed logical device. In response to a No-Operation command, the DACU supplies both Channel-End and Device-End.
Test I/O	(00)	When the host system executes a Test I/O instruction and sends it to a DACU logical device, the DACU responds with a status byte. If no status information is outstanding for the addressed device, an all-zeros status byte is returned. Any status for the addressed device is transmitted to the channel and is then reset at the end of the sequence, except for busy, which is not reset.
Set Stop Register	(07)	This command transfers two bytes from the channel and loads them into a Stop Register in a Device Control Block (DCB) in the DACU. (The content of this register is not currently used by DACU control programming.) Both Channel-End and Device-End status are returned to the channel when the data transfer is complete.

Command	Cmd Code	Description
Set Start Register	(27)	This command initiates the transfer of two bytes from the channel to the Start Register in the DACU DCB. The two bytes (which must represent an even number between X'0000' and X'7FFE') are used as a host directive by all logical device control programs except the OEMI. These directives are, in essence, orders for the *logical* devices. Their interpretation by the logical device control programs vary, depending on the device. (Their domain and interpretation are defined in the sections following.) Channel-end status is returned to the channel on completion of the data transfer; device-end status is returned when the logical device control program completes its processing of control instructions or customer written DACU code completes execution.
Set Address Reg.	(1B)	Execution of this command causes the channel to send four bytes from the host system to the Address Register field in the DCB for the logical device addressed. The content of the Address Register is used to determine the offset of the starting address for a data transfer (Read or Write). Although any four-byte address can be sent, only certain address ranges may be used for data transfers, depending on the logical device. An address register value of X'FFFFxxxx' indicates that the next Read or Write will be from or to the *control space* for the device, which is a space reserved by the logical device control programs for control purposes. (Writes to the control space of the control program for Logical Device 0 are not allowed.) Each use of Set Address Register modifies the previous state of the Address Register.
Read	(02)	This command causes the DACU to send data from the address space of the addressed logical device to the channel. The starting location within the address space is determined by the content of the Address Register in the DCB for the logical device. The Read command terminates when the channel stops accepting data, when the control unit stops sending data, or when a 64K byte DACU page boundary is reached.

Command	Cmd Code	Description
Write	(01)	Execution of a Write command will send data from the channel to the control unit address space, beginning at the address defined by the Address Register in the DCB of the logical device addressed. The Write command terminates when the channel stops sending data or when a DACU 64K byte page boundary is reached. In certain situations, the channel adapter may transfer into memory one more byte of data than sent by the host. (The data content is unpredictable.) Users should leave space between buffers for this extra byte of data when multiple buffers are defined for a logical device.
Sense	(04)	On receipt of this command, the DACU returns four bytes of sense information to the channel. This command is accepted even if the device is in a not-ready state.
Sense ID	(E4)	The DACU returns unit identification bytes. Sense ID returns seven bytes with value X'FF3258003251FF'.
Set Audible Alarm	(0B)	Sounds an audible alarm on the DACU System Unit. (A different tone is associated with each logical device.)
Read Manual Input	(0E)	Sends three "dummy" bytes to the host. (Provided only for compatibility with GAM transmission of 3250 Read Manual Input command.)
Read XY Position	(12)	Sends four "dummy" bytes to the host. (Provided for compatibility with GAM transmission of 3250 Read X,Y Position command and for attention pacing.) See section "Read XY-Position Required after Attention" on page 49.

Status and Sense bytes are defined for each of the logical devices in sections following.

Device 0: Control Unit

The *control unit address space* is that part of the DACU containing the 64K Serial I/O memory space and the 56K Parallel I/O memory space. The control unit address space is addressed by the host system via a Set Address Register command. The Address Register is implemented as four bytes of DACU memory in the control unit's Device Control Block (DCB).

Valid DACU commands follow:

Command	Cmd Code	Description
No-Operation	(03)	Performs no operation. Typically, this command would be used to obtain status information for the Control Unit. In response to a No-Operation command, the DACU supplies both Channel-End and Device-End.
Test I/O	(00)	When the host system executes a Test I/O instruction and sends it to Logical Device 0, the DACU responds with a status byte. If no status information is outstanding for the addressed device, an all-zeros status byte is returned. Any status for the addressed device is transmitted to the channel and is then reset at the end of the sequence, except for busy, which is not reset. No initial or ending status presentations are made in response to a Test I/O.
Set Stop Register	(07)	This command transfers two bytes from the channel and loads them into a Stop Register in the Device Control Block (DCB) in the DACU. (The content of this register is not currently used by DACU control programming.) Both Channel-End and Device-End status are returned to the channel when the data transfer is complete.
Set Start Register	(27)	This command initiates the transfer of two bytes from the channel to the Start Register in the DACU DCB. The two bytes (which must represent an even number between X'0000' and X'7FFF') are not used by the Control Unit Logical Device (Logical Device 0). Channel-End status is returned to the channel on completion of the data transfer; Device-End status is returned immediately. This command performs no useful function for the OEMI logical device.

Command	Cmd Code	Description
Set Address Reg.	(1B)	Execution of this command causes the channel to send four bytes from the host system to the Address Register field in the DCB for the Control Unit. The content of the Address Register is used to determine the offset of the starting address for a data transfer (Read or Write). Although any four-byte address can be sent, only the following address ranges are meaningful for data transfers: X'00000' - X'0FFFF' 64K Section of DACU Memory for Serial I/O Data X'10000' - X'1DFFF' 56K Parallel I/O Interface Facility Address Space. Communication of address register values of X'FFFFxxxx' are not allowed for the Control Unit Logical Device. Each use of Set Address Register modifies the previous state of the Address Register.
Read	(02)	This command causes the DACU to send data from the address space of the Control Unit to the channel. The starting location within the address space is determined by the content of the Address Register in DCB for the Control Unit Logical Device. The Read command terminates when the channel stops accepting data, when the control unit stops sending data, or when a 64K byte DACU page boundary is reached.
Write	(01)	Execution of a Write command will send data from the channel to the control unit address space, beginning at the address defined by the Address Register for the Control Unit Logical Device. The Write command terminates when the channel stops sending data, when the control unit stops receiving data, or when a 64K byte DACU page boundary is reached. In certain situations, the channel adapter may transfer into memory one more byte of data than sent by the host. (The data content is unpredictable.) Users should leave space between buffers for this extra byte of data when multiple buffers are defined for a logical device.
Sense	(04)	On receipt of this command, the DACU returns four bytes of sense information to the channel. This command is accepted even if the device is in a not-ready state.

Command	Cmd Code	Description
Sense ID	(E4)	The DACU returns unit identification bytes. Sense ID returns seven bytes with value X'FF3258003251FF'.
Set Audible Alarm	(0B)	Sounds an audible alarm on the DACU System Unit.
Read Manual Input	(0E)	Sends three "dummy" bytes to the host. (Provided only for compatibility with GAM transmission of 3250 Read Manual Input command.)
Read XY Position	(12)	Sends four "dummy" bytes to the host. (Provided only for compatibility with GAM transmission of 3250 Read X,Y Position command.)

Bit	Meaning
Bit 0, Attention	Used to signal the host that the logical device requires action because of: 1) a real device interrupt, or 2) an application interrupt. Always presented with Bit 6 = 1.
Bit 1, Status Modifier	Not used.
Bit 2, Control Unit End	Not used.
Bit 3, Busy	Used when Set Start Register Channel End but not Device End has been sent. Used during Selective Reset activities in the control unit.
Bit 4, Channel End	Set to show that the DACU no longer requires the use of the channel.
Bit 5, Device End	Set to show that the command has been executed and that the control unit logical device is ready to accept another command.
Bit 6, Unit Check	Set in response to an invalid command code, with Command Reject being set in the Sense Byte. Set to indicate a parity error on Command or Data Out, with Bus Out being set in the Sense Byte. Set to indicate an internal parity error in conjunction with the Data Check bit in the Sense Byte. If the command was accepted but not completely processed, Channel End accompanies Unit Check. If the command was completely processed, Device End is also sent.
Bit 7, Unit Exception	Not used.

Figure 4. Control Unit Logical Device Status Byte

Bit	Meaning
BYTE 0:	
Bit 0, Command Reject	Set when an invalid channel command is received or an illegal address is encountered.
Bit 1, Intervention Required	Not used.
Bit 2, Bus Out Check	Set by a parity error in data or a command received by the DACU.
Bit 3, Equipment Check	Not used.
Bit 4, Data Check	Set by the occurrence of an internal parity error in the DACU.
Bit 5, Overrun	Not used.
Bit 6	Not used.
Bit 7	Not used.
BYTES 1-3:	
	User defined.

Figure 5. Control Unit Logical Device Sense Bytes

Device 4: Parallel I/O Interface Adapter

The DACU address space associated with the UNIBUS-compatible Parallel I/O Interface Adapter (Logical Device 4) is a subset of the total DACU address space. The Parallel I/O interface *address space* is 64K bytes[2]. The first 56K (X'70000' to X'7DFFF') of this space addresses real DACU memory, and the remaining 8K (X'7E000' to X'7FFFF') addresses potential Parallel I/O device registers. Using I/O commands to this logical device, only the address space related to the Parallel I/O interface facility that is associated with real DACU memory (56K) may be directly addressed via channel commands. However, by executing I/O commands for the control unit (Logical Device 0), all of the available (120K — 64K for Serial and 56K for Parallel I/O) address space may be reached.

An additional area of memory is associated with the Parallel I/O Interface Adapter Logical Device called the Parallel I/O Interface Adapter Control Space. This space is addressed when the Address Register is set (via a Set Address Register command) to X'FFFFxxxx', which is not in the normal Parallel I/O interface address space. If the Set Address Register command is then followed by a Write command, the control space may be written into.

---

2 Under control of a bit in a Parallel I/O interface adapter control register, this 64K byte space can be extended to 120K bytes, but at the loss of space by the Serial I/O adapter. (See "Chapter 3. Programming Considerations" on page 13 for more information.)

An interrupt routine exists at all times which will field a device interrupt and accept an interrupt vector from the device. If no customer written code is provided for special interrupt handling, an attention will be sent to the host system, and the interrupt vector will be passed there via sense bytes.

A description of the interpretation of I/O commands for the Parallel I/O Interface Adapter Logical Device follows:

Command	Cmd Code	Description
No-Operation	(03)	Performs no operation. Typically, the command would be used to obtain status information for the Parallel I/O Interface Adapter. In response to a No-Operation command, the DACU supplies both Channel-End and Device-End.
Test I/O	(00)	When the host system executes a Test I/O instruction and sends it to Logical Device 4, the DACU responds with a status byte. If no status information is outstanding for the addressed device, an all-zeros status byte is returned. Any status for the addressed device is transmitted to the channel and is then reset at the end of the sequence, except for busy, which is not reset. No initial or ending status presentations are made in response to a Test I/O.
Set Stop Register	(07)	This command transfers two bytes from the channel and loads them into a Stop Register in the Device Control Block (DCB) in the DACU. (The content of this register is not currently used by DACU control programming.) Both Channel-End and Device-End status are returned to the channel when the data transfer is complete.

Command	Cmd Code	Description
Set Start Register	(27)	This command initiates the transfer of two bytes from the channel to the Start Register field in the DCB for the Parallel I/O Interface Adapter Logical Device. (These two bytes must be an even number between X'0000' and X'7FFE'.) Channel-End status is returned to the channel on completion of the data transfer; Device-End status is returned when the Parallel I/O Interface Adapter control program completes its processing of control space instructions or when the customer provided DACU code completes execution and issues a FAR return to DACU base code. A start register value of X'0006' causes the control data previously written in the control space for the logical device to be executed by a control data interpreter. (See "DACU Control Space Interpreter" on page 41.) A Start Register value of X'0008' causes control to be passed to customer provided DACU code. A Start Register value other than X'0006' or X'0008' will cause a warning message to be displayed on the DACU System Unit and a Device End/Unit Check to be sent to the host.
Set Address Reg.	(1B)	Execution of this command causes the channel to send four bytes from the host system to the Address Register field in the DCB for the Parallel I/O Interface Adapter. The content of the Address Register is used to determine the offset of the starting address for a data transfer to page seven (Read or Write). Although any four-byte address can be sent, only the address ranges X'0000' to X'DFFF' may be used for data transfers to the address space of the Parallel I/O Interface Adapter Logical Device. The reason the address range X'E000' to X'FFFF' is excluded is that this is the range of memory for the device's memory mapped registers. A read or write to these registers must be accomplished via commands to executed by the control space interpreter from control space. An address register value of X'FFFFxxxx' indicates that the next Read or Write will be from/to the control space of the Parallel I/O Interface Adapter. Control space can only be written to from the beginning of the area, i.e., X'FFFF0000', but the control space interpreter may be started at a location other than the beginning. The desired byte offset to start the interpreter is added to X'FFFF0000' and entered as the Address Register value. The interpreter would then be started with the Set Start Register command with a value of X'0006'. Each use of Set Address Register modifies the previous state of the Address Register.

Command	Cmd Code	Description
Read	(02)	This command causes the DACU to send data from the address space of the Parallel I/O Interface Adapter Logical Device to the channel. The starting location within the address space is determined by the content of the Address Register in the adapter's DCB. The Read command terminates when the channel stops accepting data or when the end of the adapter's address space is reached.
Write	(01)	Execution of a Write command will send data from the channel to the adapter address space, beginning at the address defined by the Address Register for the Parallel I/O Interface Adapter Logical Device. The Write command terminates when the channel stops sending data or when the end of the adapter's address space is reached. In certain situations, the channel adapter may transfer into memory one more byte of data than sent by the host. (The data content is unpredictable.) Users should leave space between buffers for this extra byte of data when multiple buffers are defined for a logical device.
Sense	(04)	On receipt of this command, the DACU returns four bytes of sense information to the channel. This command is accepted even if the device is in a not-ready state.
Sense ID	(E4)	The DACU returns unit identification bytes. Sense ID returns seven bytes with value X'FF3258003251FF'.
Set Audible Alarm	(0B)	Sounds an audible alarm on the DACU System Unit.
Read Manual Input	(0E)	Sends three "dummy" bytes to the host. (Provided only for compatibility with GAM transmission of 3250 Read Manual Input command.)
Read XY Position	(12)	Sends four "dummy" bytes to the host. (Provided for compatibility with GAM transmission of 3250 Read X,Y Position command and for attention pacing.) See section "Read XY-Position Required after Attention" on page 48.

Bit	Meaning
Bit 0, Attention	Used to signal the host that the logical device requires action because of: 1) a real device interrupt, or 2) an application interrupt. Always presented with Bit 6 = 1.
Bit 1, Status Modifier	Not used.

Figure 6 (Part 1 of 2). Parallel I/O Interface Logical Device Status Byte

Bit	Meaning
Bit 2, Control Unit End	Not used.
Bit 3, Busy	Set after Channel End sent in response to Set Start Register, and cleared when Device End is sent. Set after a Selective Reset to indicate that re-initialization is being performed.
Bit 4, Channel End	Set to show that the logical device no longer requires the use of the channel.
Bit 5, Device End	Set to show that the command has been executed and that the logical device is ready to accept another command.
Bit 6, Unit Check	Used in conjunction with Bit 0 (Attention) to show a change in the status of the DACU. When this bit and Bit 0 are on, this indicates that a logical device interrupt has occurred. In addition, Unit Check with Bit 0 = 0 has the following uses: (1) Set in response to an invalid command code, with Command Reject being set in the Sense Byte. (2) Set to indicate a parity error on Command or Data Out, with Bus Out being set in the Sense Byte. (3) Set to indicate an internal parity error in conjunction with the Data Check bit in the Sense Byte.
Bit 7, Unit Exception	Not used.

Figure 6 (Part 2 of 2). Parallel I/O Interface Logical Device Status Byte

The conditions which can cause a Device End/Unit Check are:

- A control space interpreter loop counter exceeded. (See Section "DACU Control Space Interpreter" on page 41.)
- An invalid value in the Start Register. (Only values X'0006' and X'0008' are allowed.)
- An invalid control space operation code received.
- Customer written local code signaled a Device End/Unit Check.

Bit	Meaning
BYTE 0:	
Bit 0, Command Reject	Set when an invalid channel command is received, an illegal address is specified in the Address Register, or a Control Space error occurred during Control Space execution.

Figure 7 (Part 1 of 2). Parallel I/O Interface Logical Device Sense Bytes

Bit	Meaning
Bit 1, Intervention Required	Not used.
Bit 2, Bus Out Check	Set by a parity error in data received by the DACU.
Bit 3, Equipment Check	Not used.
Bit 4, Data Check	Set by the occurrence of an internal parity error in the DACU during an operation by the logical device.
Bit 5, Overrun	Not used.
Bit 6	Not used.
Bit 7	Not used.
BYTE 1:	
Bit 0	Attention occurred.
Bits 1-2	Not used.
Bit 3	Parallel I/O interface interrupt. Bytes 2-3 contain interrupt vector.
Bit 4	Application interrupt. Bytes 2-3 contain user defined data.
Bits 5-7	Not used.
BYTES 2-3:	
	With no user programming active in the DACU, these bytes contain the interrupt vector of the interrupting I/O device When user programming is active, these bytes contain data set up by the user program for return.

Figure 7 (Part 2 of 2). Parallel I/O Interface Logical Device Sense Bytes

Device 1 and 2: Serial I/O Ports One and Two

A description of the interpretation of I/O commands for the two Serial I/O ports (Logical Devices 1 and 2) follows:

Command	Cmd Code	Description
No-Operation	(03)	Performs no operation. Typically, the command would be used to obtain status information for the Serial I/O Port Logical Devices. In response to a No-Operation command, the DACU supplies both Channel-End and Device-End.
Test I/O	(00)	When the host system executes a Test I/O instruction and sends it to Logical Device 1 or 2, the DACU responds with a status byte. If no status information is outstanding for the addressed device, an all-zeros status byte is returned. Any status for the addressed device is transmitted to the channel and is then reset at the end of the sequence, except for busy, which is not reset. No initial or ending status presentations are made in response to a Test I/O.
Set Stop Register	(07)	This command transfers two bytes from the channel and loads them into a Stop Register in the Device Control Block (DCB) for the addressed Serial I/O Logical Device. (The content of this register is not currently used by DACU control programming.) Both Channel-End and Device-End status are returned to the channel when the data transfer is complete.

Command	Cmd Code	Description
Set Start Register	(27)	This command initiates the transfer of two bytes from the channel to the Start Register field in the DCB for the addressed Serial I/O Port. (These two bytes must represent an even number between X'0000' and X'7FFE'.) Channel-End status is returned to the channel on completion of the data transfer; Device-End status is returned when the control program for the logical device completes its processing of the Start Register contents. When a Set Start Register command is received, the Serial I/O Port control program causes the following actions to occur depending on the hexadecimal value of the two bytes communicated from the channel: '0002' - Causes the serial port to begin receiving data and transferring it into a DACU buffer area. '0004' - Causes the serial port to begin transmitting data from a DACU buffer area. '0006' - Causes initialization of the serial port to be performed based on control data previously written in the control space for the serial port. See "Programming for Serial I/O Devices" on page 49 for the definitions of these parameters. '0008' - Passes control to customer-provided DACU code. '000A' - Send "Break" over line. '000C' - Re-initialize to power-on state. Channel End is presented at the completion of the two byte data transfer from the channel. Device End is presented when execution of DACU resident code completes.

Command	Cmd Code	Description
Set Address Reg.	(1B)	Execution of this command causes the channel to send four bytes from the host system to the Address Register field in the DCB for the addressed Serial I/O Port Logical Device. The content of this address register is used to determine the offset of the starting address for a data transfer to page six (Read or Write). This offset is measured from the beginning of the write buffer or read buffer (frequently specified as zero). Although any four-byte address can be sent, only the address ranges X'0000' to X'FFFF' may be used for data transfers to the address space of a Serial I/O Port Logical Device. An address register value of X'FFFFxxxx' indicates that the next Read or Write will be from/to the control space of the Serial I/O Port Logical Device addressed. Control space can only be written to from the beginning of the area, Each use of Set Address Register modifies the previous state of the Address Register.
Read	(02)	This command causes the DACU to send data from the address space of a Serial I/O Port Logical Device to the channel. The starting location of this buffer is determined by the content of the Address Register in the Port's DCB. The read command terminates when the channel stops accepting data or when the end of the address space of the logical device is reached.
Write	(01)	Execution of a Write command will send data from the channel to the address space of this logical device beginning at the address defined by the Address Register for the Logical Device. The Write command terminates when the channel stops sending data or when the end of the Logical Device's address space is reached. In certain situations, the channel adapter may transfer into memory one more byte of data than sent by the host. (The data content is unpredictable.) Users should leave space between buffers for this extra byte of data when multiple buffers are defined for a logical device.
Sense	(04)	On receipt of this command, the DACU returns four bytes of sense information to the channel. This command is accepted even if the device is in a not-ready state.
Sense ID	(E4)	The DACU returns unit identification bytes. Sense ID returns seven bytes with value X'FF3258003251FF'.
Set Audible Alarm	(0B)	Sounds an audible alarm on the DACU System Unit. (A different tone is associated with each logical device.)
Read Manual Input	(0E)	Sends three "dummy" bytes to the host. (Provided only for compatibility with GAM transmission of 3250 Read Manual Input command.)

Command	Cmd Code	Description
Read XY Position	(12)	Sends four "dummy" bytes to the host. (Provided for compatibility with GAM transmission of 3250 Read X,Y Position command and for attention pacing.) See section "Read XY-Position Required after Attention" on page 48.

Bit	Meaning
Bit 0, Attention	Used to signal the host that a Serial I/O device requires action. Always presented with Bit 6=1 to show a change in the status of the DACU Serial I/O adapter.
Bit 1, Status Modifier	Not used.
Bit 2, Control Unit End	Not used.
Bit 3, Busy	Set between Channel End and Device End after a Set Start Register is received and during Selective Reset re-initialization activities.
Bit 4, Channel End	Set to show that the Serial I/O logical device no longer needs use of the channel.
Bit 5, Device End	Set to show that the command has been executed and that the Serial I/O logical device is ready to accept another command.
Bit 6, Unit Check	Used in conjunction with Bit 0 (Attention) to show a change in the status of the Serial I/O device. When this bit and Bit 0 are on, this indicates a program in the DACU has signaled attention. The sense data contains further information. Bit 0 = 0 has the following uses: (1) Set in response to an invalid command code, with Command Reject being set in the Sense Byte. (2) Set to indicate a parity error on Command or Data Out, with Bus Out being set in the Sense Byte. (3) Set to indicate an internal parity error in conjunction with the Data Check bit in the Sense Byte. If the command was completely processed, Device End is also sent.
Bit 7, Unit Exception	Not used.

Figure 8. Serial I/O Logical Device Status Byte

Bit	Meaning
BYTE 0:	
Bit 0, Command Reject	Set when an invalid channel command is received, an illegal address is specified in the Address Register, or a Control Space error occurred during Control Space execution.
Bit 1, Intervention Required	Not used.
Bit 2, Bus Out Check	Set by a parity error in data or a command received by the DACU.
Bit 3, Equipment Check	Not used.
Bit 4, Data Check	Set by the occurrence of an internal parity error in the DACU during an operation by the Serial I/O logical device.
Bit 5, Overrun	Not used.
Bit 6	Not used.
Bit 7	Not used.
BYTE 1:	
Bit 0	Attention occurred.
Bits 1-2	Not used.
Bit 3	If zero, indicates end of transmission. If one, indicates end of input. Bytes 2-3 contain the number of characters sent or received.
Bit 4	Application interrupt. Bytes 2-3 contain user defined data.
Bit 5	Error.
Bit 6	Break received.
Bit 7	Not used.
BYTES 2-3:	
	Number of bytes transmitted or received. If the Serial I/O logical device has a user program active, these bytes contain user-defined information necessary to the host application program.

Figure 9. Serial I/O Logical Device Sense Bytes

Host Programming Support

I/O Programming Support

Host system programming must consider mechanisms for:

1. Making the DACU known to the host system control program. (For purposes of system generation, the DACU will appear to be a 2250 Model 3 control unit or "unsupported device.")

2. Creating and initiating channel programs that will communicate the I/O commands (and associated data) that will control, write to, and read from the DACU logical devices. See Appendix A, "Sample Host Programming" on page 147 for examples of channel programming code.

3. Handle attentions returned from the DACU logical devices by the channel. See Appendix A, "Sample Host Programming."

4. Error diagnosis and recovery. (See *OS/VS2 I/O Supervisor Logic,* SY26-3823, or *Virtual Machine/System Product System Programmer's Guide,* SC19-6203 and Appendix A, "Sample Host Programming.")

There are several mechanisms available which can be used by the system programmer to create channel programs — EXCP or Start I/O programming and an access method.

EXCP Programming

The lowest level of DACU host system programming involves building a channel program and issuing an EXCP (Execute Channel Program - SVC 0) or a System/370 Start I/O (SIO) instruction to execute it. (For a reference on EXCP services, see *OS/VS2 MVS Supervisor Services and Macro Instructions,* GC28-0683, or *IBM Virtual Machine/System Product: CMS User's Guide,* SC19-6210. For a reference on Start I/O, see *IBM System/370 Principles of Operation,* GA22-7000.)

GAM Programming

The Graphics Access Method (GAM) is available in two forms:

- As a constituent of Graphic Programming Services (GPS) which is supported under VS1 and MVS

- As Graphics Access Method/System Product (GAM/SP), a Program Product for VM/SP CMS.

(GAM was designed to provide facilities for users of the IBM 3250 Graphics Display System and the IBM 2250 Model 3 Display Unit.)

The I/O commands interpreted by the Device Attachment Control Unit are compatible with the I/O commands of the IBM 3250 and 2250 in the following sense:

1. The set of DACU commands is a subset of 3250 and 2250 commands.

2. Commands which are members of the 3250 and 2250 set but not members of the DACU set are either given command reject or are designed to simulate the 3250 command but transfer meaningless data.

It should be understood that the compatibility exists at the *control unit I/O command* level. In addition, the interpretation of the commands by the DACU may be quite different from that of the 3250. (For example, the 3250 Set Programmed Function Indicators command is used to load four bytes into the DACU Address Register.) *Compatibility does not exist at the level of 3250 and 2250 display orders.* The degree of compatibility that exists is useful for two reasons:

- GAM functions may be used to build and execute the channel programs required to manage the DACU even though the interpretation by the DACU and its response to the host system are quite different from those of the 3250 and 2250.

- During the I/O configuration phase of operating system generation, the DACU should be treated as though it was a 2250 Model 3.

In the context of programming for the DACU, GAM and GAM/SP provide application programs with facilities for:

- Creating buffer loads in the host processor and transmitting them to a DACU logical device

- Attention handling

- Retrieving data associated with interrupts

- Controlling a logical device (e.g., controlling buffer locations).

No explicit statement of DACU support *by* GAM should be inferred. Since compatibility exists in the sense described above, it is possible to use GAM to utilize functions of the DACU. However, all GAM functions are not applicable to the DACU. For example, GAM facilities that generate 3250 display orders are not relevant to the DACU.

The correspondence between the set of DACU I/O commands and the matching 3250/2250 I/O commands is indicated below:

DACU Command	**Cmd Code**	**3250 Command**
No-Operation	(03)	Control No-Operation
Test I/O	(00)	Test I/O
Set Stop Register	(07)	Set Buffer Address Reg. and Stop
Set Start Register	(27)	Set Buffer Address Reg. and Start
Set Address Reg.	(1B)	Set Program Function Indicators

DACU Command	Cmd Code	3250 Command
Read	(02)	Read Buffer
Write	(01)	Write Buffer
Sense	(04)	Sense
Sense ID	(E4)	Sense ID
Set Audible Alarm	(0B)	Set Audible Alarm
Read Manual Input	(0E)	Read Manual Input
Read XY Position	(12)	Read X,Y Position

Attention Handling

I/O device actions may result in an attention being transmitted to the host processor. Attentions from Parallel I/O devices contain an interrupt vector indicating that a device generated an interrupt. Attentions from Serial I/O devices contain the number of bytes transmitted or received. Local code written in the DACU System Unit can also generate attentions, sending data relevant to customer applications. HALLS and HALLGE provide subroutines to handle attentions and present attention data to customer application programs. See Appendix A, "Sample Host Programming" on page 147. Other methods of handling attentions are to use GAM or write a host interrupt handler and install it using the CMS HNDINT macro. (For reference on HNDINT, see *IBM VM/SP CMS Command and Macro Reference*, SC19-6209.)

Reference Documentation

The following documents may be useful when considering host I/O programming:

OS/VS2 MVS Data Management Macro Instructions, GC26-3873

OS/VS2 MVS Data Management Services Guide, GC26-3875

OS/VS2 Supervisor Logic, SY26-3823

Graphics Access Method System Product Users Guide, LC33-0126

OS/VS Graphic Programming Services (GPS) for IBM 2250 Display Unit and IBM 3250 Graphics Display System, GC27-6971

OS/VS2 Graphics Access Method Logic, GY27-7260.

Sample Host Programming Techniques

The host interface characteristics of the DACU were intentionally defined for operational compatibility with an existing IBM control unit. The control unit selected for its channel control unit architectural similarities to the DACU was the IBM 3250 Display System. The DACU is not *fully* compatible to the 3250; but it is sufficiently compatible that some of the programming support for the 3250 will provide support for the DACU. This section will treat functions available in the host control programs and access methods that can be used with the DACU. Development or conversions of application programs to run with the DACU will not be discussed here. Such considerations are generally device and/or application

dependent, and a discussion of those considerations would have to be made in the context of the devices and/or the applications.

There are three control programs with access method support for the 3250:

- VM/SP (Release 1.4 or later)
- MVS
- VS1.

A DACU may be attached to systems running these operating systems. During a system generation, however, it must be identified in the same way as a 3250. (Details for system generation may be found in the System Generation manuals for these systems. The access method of interest is the Graphics Access Method (GAM) that is used for support of the 3250. (It should be noted that there is no IBM programming support *specifically intended* to support the DACU; future changes to GAM could impact DACU operations, although these are not expected.)

It is also possible for a customer to use a customized I/O access path for the DACU, coding channel command words and using the EXCP interface provided in any of the control programs listed above (also supported in the DOS control program), or coding channel command words and executing them using the IBM System/370 Start I/O instruction.

The information that follows is useful in the development of programs that use either the GAM interface, the EXCP interface, or Start I/O.

An appendix contains two sample subroutine libraries, one using GAM and EXCP (HALLGE for MVS and VM) and one using Start I/O (HALLS for VM only). These subroutine libraries may be used as is or tailored to specific customer needs.

System Generation and I/O Configuration Examples

It is necessary to identify the DACU as an I/O control unit and devices to the control program. It is necessary to configure the operating system so that it recognizes a *2250*[3] at the device addresses assigned to DACU.

A Unit Control Word (UCW) must be made active for each I/O device address associated with the DACU configuration. This is done at the host computer operator console by trained personnel. (Follow the normal procedures for your system installation.) Each of the UCWs associated with the DACU should be set for block multiplexer mode (BMPX) and UNSHARED. (Selector Mode and Data Streaming are other options THAT ARE NOT USED.) Because UCW defaults vary from system to system, make sure that the correct options are chosen. It will be necessary to IML the system to make the new UCWs active.

For VM, the control program component (CP) can be configured to recognize the DACU in the manner described below. First, define the device to CP. For definition of the device addresses the RDEVICE macro is specified as follows:

---

[3] Although the more modern device is a 3250, the current practice is to identify it to the operating system as a 2250.

```
DACUDEV RDEVICE ADDRESS=(400,8),DEVTYPE=2250
```

In this example, DACUDEV is the label on the RDEVICE macro, 400-407 is the range of device addresses, and DEVTYPE specifies that the device is like a 2250. There should be one RDEVICE macro for each range of device addresses.

CP will also require the definition of a control unit. For this, an RCTLUNIT macro is required similar to the following:

```
    RCTLU4 RCTLUNIT ADDRESS=400,CUTYPE=2840
or:
    RCTLU4 RCTLUNIT ADDRESS=400,CUTYPE=2250
```

In this example, RCTLU4 is the label on the RCTLUNIT macro defining the control unit to support real device addresses 400-407. The control unit is defined to be like a 2840 or 2250. There should be one RCTLUNIT macro for, in this example, every 8 device addresses.

For MVS, the following macro would define a configuration similar to the one for VM above:

```
IODEVICE ADDRESS=(400,1),UNIT=2250,MODEL=3,PCU=01,
      NUMSECT=128,FEATURE=(PRGMKYBD,ALKYB2250)
IODEVICE ADDRESS=(401,1),UNIT=2250,MODEL=3,PCU=02,
      NUMSECT=128,FEATURE=(PRGMKYBD,ALKYB2250)
IODEVICE ADDRESS=(402,1),UNIT=2250,MODEL=3,PCU=03,
      NUMSECT=128,FEATURE=(PRGMKYBD,ALKYB2250)
IODEVICE ADDRESS=(403,1),UNIT=2250,MODEL=3,PCU=04,
      NUMSECT=128,FEATURE=(PRGMKYBD,ALKYB2250)
IODEVICE ADDRESS=(404,1),UNIT=2250,MODEL=3,PCU=05,
      NUMSECT=128,FEATURE=(PRGMKYBD,ALKYB2250)
```

Following the configuration of the specific control program to support the DACU, it is necessary that GAM be configured to enable it to do buffer management for what appears to be an IBM 3258 Control Unit. This is done by the process referred to as buffer table generation in, for example, the *GAM/SP Users Guide* LC33-0126. One example method for buffer table generation is:

```
BUFTABS  CSECT
         GABBUF
         GABDEV  400,128,FEATURE=(LIGHTPEN,ANKB,PFKB)
         GABBUF
         GABDEV  401,128,FEATURE=(LIGHTPEN,ANKB,PFKB)
         GABBUF
         GABDEV  402,128,FEATURE=(LIGHTPEN,ANKB,PFKB)
         GABBUF
         GABDEV  403,128,FEATURE=(LIGHTPEN,ANKB,PFKB)
         GABBUF
         GABDEV  404,128,FEATURE=(LIGHTPEN,ANKB,PFKB)
         GABBUF  END
         END
```

(The units above are identified as having light pen, alphanumeric keyboard, and programmed function keyboard features to permit compatibility with GAM initialization procedures and with DACU commands and attention processing.)

Assembly and linkedit of this macro (i.e., how to include it with the rest of the GAM generation) is control program dependent, and the appropriate GAM manual will have to be consulted.

Once the control program and GAM are properly configured, the question of how to use GAM by the application program can be considered. Again, there are control program dependencies associated with the particular GAM product being used. The coding examples below have been validated and are sufficiently general that none of the dependencies have been included. Familiarity with the use of existing access methods is assumed.

To define the DCB:

```
SAMPDCB DCB   DSORG=GS,MACRF=(RC,WC),DDNAME=DACU,
              GTYPE=BASIC
```

To OPEN the DCB:

```
OPEN (SAMPDCB)
```

To issue a GREAD or GWRITE:

```
GWRITE   DECB,BUF,SAMPDCB,OCBP

GREAD    DECB,BUF,SAMPDCB,OCBP
```

To simply issue a GREAD or GWRITE is not sufficient to perform I/O for the DACU. (Nor would it be sufficient for the IBM 3250.) To perform a write or read with the 3250, a two byte buffer address must be set with the Set Buffer Address Register and Start (or Stop) command. GAM chains this command automatically to precede the write or read when a GWRITE or GREAD is used. However, the DACU requires a longer address than the 3250. To set the DACU Address Register, which is four bytes long, a command code different from the Set Buffer Address Register and Start (or Stop) command code is used. (The command code — Set Address Register — is identical to the 3250 command code Set Programmed Function Indicators.) To write an address to the DACU Address Register using GAM, use the following, for example:

```
GCNTRL DECB,IND,SAMPDCB,BUFADDR
```

Here, BUFADDR is a four byte field containing the DACU address. It is only necessary to issue this GCNTRL if it is desired to change the DACU address from which a data transfer will start.

Consideration must be given to the problem of synchronization of the I/O operation initiated above — a subsequent operation must not be allowed to interfere with current operations. The Device End/Channel End indications (defined by the S/370 principles of operations and supported by the control programs) are sufficient to support the immediate operations that are completed by the control unit (DACU) itself. However, where an I/O operation causes activity in a device attached to the DACU, it is necessary to synchronize operations with those devices. It is possible to synchronize such operations by using the attention. GAM's Basic Attention Handling capabilities are suitable for this purpose. (GAM's Express Attention Handling facilities are highly 3250 dependent, and use

of those facilities is not recommended.) There are essentially two options when using Basic Attention Handling.

One is to write new code embodying an attention handling routine that is recognized by GAM through the SPAR macro. The GACB pointed to in the SPAR macro must contain the entry point address for this customer written attention handling routine.

The other option for handling attention is to use the ATTNINQ macro of GAM. This allows a program to poll for device completion information or to wait for completion. The following examples apply to this mechanism:

First, define the GACB using the SAEC:

```
GACB SAEC EP=0,DCB=SAMPDCB,COMAREA=COMSAVE,
          ATTNTYP=(LP),PFKMSK=NULL
```

Then issue the SPAR macro:

```
SPAR (GACB),PRTY=0
```

The SAEC and SPAR need only be issued once. After that, attentions may be handled using the ATTNINQ macro.

To wait for an attention:

```
ATTNINQ GACB, MODE=W
```

This will cause the program to wait until an attention occurs. When the attention is received, GAM will perform a Sense and Read XY Position, store information in COMSAVE, and resume program execution.

To poll for an attention:

```
ATTNINQ GACB,MODE=(C,ATTNINT),TYP=LP
```

If an attention occurs, information will be put in the GACB field labelled COMSAVE, and control will be transferred to the instruction with the label ATTNINT. If no attention occurs, control will be given to the instructions immediately following the ATTNINQ macro.

The examples given above relate to operations which are not obvious extensions of the use of GAM with an IBM 3250. They must be supplemented by considerations of the code to be written for specific devices and for specific applications.

EXCP Programming Examples

Programming at the EXCP level yields additional flexibility. The section of the GAM Users Guide on EXCP programming should be referenced. CCW strings coded for the DACU should be similar to those defined in that manual — in particular, the CCW strings defined by the GREAD type STR, GWRITE type STR, and GCNTRL type IND. The CCW for GCNTRL type IND is used to set the DACU address register. The DACU also supports the GREADR type SEN to retrieve sense information from the device. Use of GAM is recommended for all attention handling unless the user is a capable systems programmer. (Sample host

programs using GAM, EXCP, and Start I/O are illustrated in Appendix A, "Sample Host Programming" on page 147.)

Host Programming Considerations for Devices

Programming for Parallel I/O Devices

To control Parallel I/O devices from the host via the DACU, it is necessary to send control and data transfer commands to the DACU. Typically, data sent to the DACU that is destined for a Parallel I/O device contains device *orders* — commands (including data) which are meaningful to the particular UNIBUS-compatible device. Contrast these with DACU commands which are instructions to the DACU that cause, for example, reading or writing of the registers of the UNIBUS-compatible device. In addition to the device's registers, the programmer also must be aware of the DACU's Parallel I/O Control (UBCTL), Parallel I/O Status (UBSTAT) Registers, and Parallel I/O Interrupt Vector Register (UBINTVEC).

DACU Parallel I/O Control and Status Registers

This section contains information on the DACU Parallel I/O control facilities. It is important to understand the operation of the adapter if UNIBUS-compatible devices are to be atached. If DACU default code is used, the actions described in this section are taken care of automatically. The one exception is that a X'3FC0' must be written initially into the control register.

In the discussion that follows, information is given on how to control some of the lines in the Parallel I/O Interface. More information on the function of these lines can be found in, for example, *XPDP-11 Bus Handbook*, Digital Equipment Corporation, 1979.

The lines referred to below and their names are:

NPR Non-Processor Request

NPG Non-Processor Grant

BR4 Bus Request 4

BR5 Bus Request 5

BR6 Bus Request 6

BR7 Bus Request 7

INTR Interrupt Request

INIT Initialize

The DACU Parallel I/O Control Register—UBCTL is 16 bits wide. By writing into this register, the programmer can affect how the Parallel I/O device interacts

with the DACU. (For example, there are bits in the register which can mask off the Bus Request (BR) lines on the Parallel I/O interface so that interrupts cannot occur.) This a write-only register. The DACU Parallel I/O Status Register—UBSTAT gives information on the state of the DACU's Parallel I/O adapter. This is a read-only register. Descriptions of these registers follow. (In the discussion bit numbering scheme associated with many commercially available Parallel I/O devices is used: the most significant bit is 15, and the least significant bit is 0.)

When an interrupt occurs on the interface, it must be cleared before any other Parallel I/O operations can occur. By reading the UBINTVEC register (5000:9002,3) one obtains the interrupt vector of the interrupting device and also sends out the appropriate signals to clear the interrupt process. If one does not want any more interrupts on any particular level for some time, the appropriate bus requests (BR7-BR4) should be masked in the UBCTL register *prior* to reading the UBINTVEC register and unmasked when one can handle those interrupts.

The Parallel I/O adapter is capable of doing reads and writes to any valid Parallel I/O device register. Before doing any register I/O, the UBCTL register must be correctly set up. If the DACU default code is run prior to user provided code, it is left in an initialized state with interrupts disabled and the bus requests masked. To enable the interface, a X'3FC0' should be written to enable interrupts to the code, allow non-processor requests (NPR) to the interface, and allow bus requests (BR). The action of writing the X'3FC0' can be done using the control space interpreter or through local code. If running standalone without any DACU code, a X'3FC0' should be written into the UBCTL to remove the Parallel I/O reset and take the adapter out of self test mode. (See the UBCTL description for an explanation of the bit sequence.)

The 8K Parallel I/O register space is memory mapped into the top 8K of the DACU memory of Page 7. All register operations are buffered by the Parallel I/O adapter and rely on the Parallel I/O Status Register (UBSTAT) to indicate completion. (See the UBSTAT description for an explanation of the bit sequence.) In the case of the Read operation (See RUB macro in "Chapter 4. Local DACU Programming" on page 59.), the device register is read, then bit 10 of the UBSTAT register is tested until set and the data from that device register is read from the UBRDDAT register (5000:9006,7). A read of the UBRDDAT register resets bit 10 of the UBSTAT register. In the case of the Write operation, (See WUB macro in "Chapter 4. Local DACU Programming" on page 59), the device register is written to, then bit 10 of the UBSTAT register is tested until set. This indicates the data has been transferred to the device register. It is then necessary to read the UBRDDAT register to reset bit 10 of the UBSTAT register. In both cases, bit 10 of the UBSTAT register should be set within tens of microseconds from the start of the read or write. The RUB and WUB macros do not check for a time-out condition and will loop forever if presented with an invalid I/O register address.

Another restriction on register I/O is that the NPR mask must be set prior to performing register I/O if the I/O is going to occur during NPR/NPG data transfers. Finally, if an interrupt is pending on the interface, no register I/O can occur until that interrupt is cleared by reading the UBINTVEC register.

The Parallel I/O Adapter may be initialized by writing a zero to bit 7 of the UBCTL register. After writing a zero, a one should be written to allow Parallel I/O to be active. When DACU code is first run, an initialization is performed.

On initialization, a mask is written into the Parallel I/O control register (UBCTL) (low order byte) with bits set in the following manner:

Bit	7	(NOTINIT)	Set to zero to initialize the Parallel I/O Interface.
Bit	6	(ENIRQ)	Set to zero to disable interrupts from the Parallel I/O interface.
Bit	5	(NPRMASK)	Set to one to mask NPR requests.
Bits	4-1	(BRnMASK)	Set to one to mask BRn (n=4,5,6,7) requests.
Bit	0	(BYTEXFER)	Set to 0 so that transfers will be 16 bits.

After the write to UBCTL, UBCTL is written again with the same values except for Bit 7 (INIT), which is set to one to put the interface in a quiescent state.

Bit	Name	
15	+ENABLE 128K	When set, Parallel I/O addressing begins at page X'60000' instead of X'70000'.
14	+WRAP TEST	Bits 8 through 14 are used during testing. Normally, the high byte should be set to X'3F'.
13	-AUX SACK	
12	-MSYN AUX	
11	-TEST MODE	
10	+MODE 2	
9	+MODE 1	
8	+MODE 0	
7	-INITIALIZE	When a zero is written into bit 7, the INIT Parallel I/O line is made active. After writing a zero, a one should be written to allow the Parallel I/O to be active.
6	+ALLOW IRQ'S	When set, interrupts are allowed through to the DACU System Unit's interrupt handler. This should be set if doing interrupt processing.
5	+NPRMASK	When set, NPR's are masked on the interface. This should normally be set to zero.
4	+BR7MASK	Bits 1 through 4 allow masking of Parallel I/O
3	+BR6MASK	Bus Request lines. (BR 4,5,6,7). BR's must be
2	+BR5MASK	unmasked for the particular level of the device if
1	+BR4MASK	interrupt processing is desired.
0	+BYTEXFER	If set, DMA transfers to and from the DACU memory and a device are on a byte rather than 16-bit (word) basis.

Figure 10. Parallel I/O Control Register (UBCTL)

Bit	Name	
15,14,13,12,11		Not used.
10	+DATA XACTION RDY	After a read or write to a Parallel I/O register has been initiated, this bit will be set when that I/O operation is complete. This bit is reset by a read of the UBRDDAT register (5000:9006,7).
9	+TEST IRQ4	Used for self test.
8	-MUX REQ	Used for self test.
7	+INTR	When set the INTR line on the interface is active. No further Parallel I/O processing can occur until the Parallel I/O interrupt vector has been read.
6	+ALLOW IRQ'S	Bit 0 through Bit 6 reflect the status of the corresponding bits in the control register (UBCTL).
5	+NPRMASK	
4	+BR7MASK	
3	+BR6MASK	
2	+BR5MASK	
1	+BR4MASK	
0	+BYTEXFER	

Figure 11. Parallel I/O Status Register (UBSTAT)

DACU Control Space Interpreter

While data can be written anywhere in the DACU normally associated with the Parallel I/O facility (between X'0000' and X'DFFF'), data cannot be written *directly* from the host to a Parallel I/O device register even though that register occupies an address in the Parallel I/O device address space. Reads and writes between the host and the device register space are done *indirectly* by the *Control Space Interpreter* or customer provided code using the RUB and WUB macros or subroutines.

The interpreter also allows access to the DACU Parallel I/O Status Register (UBSTAT) and write access to the DACU Parallel I/O Control Register (UBCTL). However, UBCTL is controlled during control space interpretation to inhibit Parallel I/O activity. That is, Non-Processor Requests (NPRs) and Bus Requests (BRs) are not allowed when there is activity on the DACU bus. To accomplish this, the status register (UBSTAT) is read before the first control space instruction is interpreted. The mask bits for NPR and the BRs are set on and interrupts are not allowed. If no writes to the Parallel I/O control register (UBCTL) are done, the control register is restored at the completion of the control space interpretation. If a Parallel I/O control register write is encountered, it is written into a *copy* of UBCTL. This copy is held until the last instruction is interpreted, and it is then restored in UBCTL.

Before processing any control space commands, the interpreter interrogates the status register to see if an interrupt is pending. If it is, the interrupt vector is read. This allows the Parallel I/O facility to function. At completion of the interpretation of all commands, a Device End is sent to the host. If an interrupt is pending, the appropriate attention is also sent to the host.

Associated with the Control Space Interpreter are the following registers (in DACU storage):

ACCUM Accumulator—This 16-bit register holds the data from the most recent read of a Parallel I/O register, the DACU Parallel I/O Status Register (UBSTAT), or the Parallel I/O interrupt vector.

IP Instruction Pointer—The Instruction Pointer points to the control space instruction currently being interpreted. It increments by one after execution of each instruction unless the instruction is a jump, in which case it is modified to point to the target instruction. It is always initialized to zero on entry to a set of control space instructions.

CTR Instruction Counter—The CTR register keeps a count of how many instructions have been interpreted. This is used in conjunction with the CTRLIM register (see below) to limit the number of interpretations done. In this way the programmer can protect against putting the DACU in an infinite loop. When the CTR is greater than CTRLIM, interpretation ends, an error message is provided to the DACU System Unit, and a Device End/Unit Check is dispatched to the host.

CTRLIM Counter Limit—The default value of this register is 5000. That is, no more than 5000 interpretations can be done. If CTRLIM is set to zero, there is no limit.

SOURCE Source address—An address in Parallel I/O space from which data is obtained. This address can be in the DACU RAM or in the device register space.

TARGET Destination address—An address in Parallel I/O space to which data is sent. This address can be in the DACU RAM or in the device register space.

Allowable Control Space Interpreter commands follow:

Command	OP-Code	OP1	OP2	Comment
END	0	XXXX	XXXX	End interpretation and return. Every control space must contain this command.
WUBCTL	1	XXXX	DATA	Write data to the DACU Parallel I/O Control Register (UBCTL). The data is actually written at the conclusion of all interpretations.
RUBSTAT	2	XXXX	TARGET	Read DACU Parallel I/O Status Register and store result in TARGET and the accumulator.
MVIW	3	TARGET	DATA	Write data into Parallel I/O register or into DACU memory at address TARGET.
MOVE	4	SOURCE	TARGET	Read data from Parallel I/O register or DACU memory at SOURCE and store the 16 bits in Parallel I/O register or memory space at address TARGET and the accumulator.
TJZ	5	INSTR	MASK	Test and Jump on Zero. Test the value of the accumulator under MASK and jump to the instruction INSTR if logical AND between these two values produces a zero.

Command	OP-Code	OP1	OP2	Comment
TJNZ	6	INSTR	MASK	Test and Jump on Not-Zero. Test the value of the accumulator under MASK and jump to the instruction INSTR when ALL of the bits in the accumulator which correspond to ones in the MASK are non-zero. (This instruction is useful in testing for the presence of several READY bits simultaneously.)
JUMP	7	INSTR	XXXX	Jump to INSTR.
RDINTR	8	XXXX	TARGET	Read Parallel I/O interrupt vector and store in address TARGET and the accumulator.
CMPJE	9	INSTR	COMPARAND	Compare and Jump if Equal. Compare the accumulator with COMPARAND and jump to INSTR if they are the same.
CTRLIM	10	XXXX	DATA	Set the value of CTRLIM.
MVIB	11	TARGET	DATA	Write least significant byte of DATA into Parallel I/O register or into DACU memory at address TARGET.
LOAD	12	SOURCE	XXXX	Same as MOVE, but does not store data in memory.

Command	OP-Code	OP1	OP2	Comment
WORDWRITE	13	WWRT	XXXX	If WWRT=1, set word write mode (default). If WWRT=0, use byte write mode. That is, the host byte order is reversed.
HATTN	14	DATA	XXXX	Send attention to host. DATA is attention data which is placed in sense bytes 2 and 3.

The control space interpreter can be started at any location within the control space by specifying the appropriate value of the address register with X'FFFF' in the first two bytes and the offset to start the interpreter in the last two bytes. In this way, it is possible to load two different pieces of control space code and execute the appropriate one when desired without reloading it.

Often, the control space code remains the same between uses except for values such as the word count. It is still possible to utilize the same control space many times by using the capability of moving the word count from DACU memory to a Parallel I/O register via the MOVE instruction rather than using the Move Immediate Word instruction (MVIW). In this case, one could add the word count to the beginning of the data which is sent to the buffer. The control space code would read the word count and send it to the device register before initiating the DMA.

Control Space Interpreter Example

Consider a Parallel I/O device which has the following registers:

DRWC Word count register. A 16 bit word is written into this register which contains the two's complement of the number of (16 bit) words to be transferred during a DMA operation under the control of the Parallel I/O device.

DRBA Address in DACU memory where the DMA should start.
DRST Contains status and control information for the Parallel I/O device.

The following set of control space instructions is an example of what the control space should contain to initiate a DMA transfer under control of the Parallel I/O device:

```
LN  OP   OP1  OP2    CMD              COMMENT

00 0001 0000 3FC0   WUBCTL 3FC0      ALLOW IRQ'S, NPR'S,
                                     AND BR4,5,6,7
01 0003 F53A 1000   MVIW   DRBA,1000 DATA STARTS AT 1000
02 0003 F538 F800   MVIW   DRWC,F800 DMA 2048 WORDS
03 0003 F53C 0003   MVIW   DRST,GO   GO
04 0000 0000 0000   END               END
```

This particular example is quite simple since it requires only writing to the registers to initiate a DMA transfer. However, in addition to the above, consideration must be given to interrupts returning to the host as attentions or polling the device to see if it is ready before initiating another operation involving the device.

A more complicated example follows.

In this example, bits in a device register must be checked before moving on to the next step. This particular example moves a block of 256 words from the device to the DACU memory.

```
LN  OP   OP1  OP2    CMD              COMMENT

00 0001 0000 3FC0   WUBCTL 3FC0      ALLOW IRQ'S, NPR'S,
                                        AND BR4,5,6,7
01 0004 F536 DFF4   MOVE   GPST,DFF4  READ GPST REG

02 0006 0009 1000   TJNZ   0009,1000  IF PRE BIT=1 GOTO 9

03 0004 F53C DFF2   MOVE   DRST,DFF2  PRE BIT=0.
                                        CHK STATUS BITS
04 0005 0006 8000   TJZ    0006,8000  IF ERROR BIT=0,
                                        GO TO 6
05 0000 0000 0000   END               ERROR; ABORT

06 0005 0003 0080   TJZ    0003,0080  IF READY BIT NOT=1,
                                        THEN GOTO 3
07 0003 F53C 0002   MVIW   DRST,0001  SET DIR,
                                        DISABLE INT
08 0003 F536 0400   MVIW   GPST,0400  SET GPST PRE BIT

09 0004 F53C DFF2   MOVE   DRST,DFF2  READ DRST REG,
                                        STORE IN HI MEM
0A 0006 0005 8000   TJNZ   0005,8000  IF ERROR BIT=1,
                                        GO TO 5, ABORT
0B 0005 0009 0080   TJZ    0009,0080  IF NOT READY,
                                        GOTO 9, TRY AGAIN
0C 0003 F538 FF00   MVIW   DRWC,FF00  WRITE CNT (256 DEC)
                                        IN DRWC
0D 0003 F53A 0000   MVIW   DRBA,0000  WRITE DACU RECEIVE
                                        ADDRESS
0E 0003 F53C 0043   MVIW   DRST,0043  SET DMA GO BIT
                                        IN DRST
0F 0000 0000 0000   END               END
```

Parallel I/O Device Programming for Extended Storage

Under default conditions, 56K of DACU storage is available for Parallel I/O operations (plus the 8K Parallel I/O device register space). If more of the DACU storage is desired for Parallel I/O operations, it is possible to gain an additional 64K bytes. However, the 64K bytes gained is the portion of memory normally reserved for Serial I/O use. Therefore, when using 120K (64K plus 56K) for Parallel I/O operations, it is not possible to also use Serial I/O.

To use the entire 120K of available storage for Parallel I/O operations, it is necessary to use *two* of the logical devices associated with the DACU — the Parallel I/O logical device and the control unit logical device (which has access to all of the DACU facilities). These are, respectively, Device 4 and Device 0.

When programming for this extended use of storage, the DACU address range X'00000' to X'0FFFF' corresponds to Personal Computer Page 6 (normally the Serial I/O page), and the DACU address range X'10000' to X'1FFFF' corresponds to Page 7 (the normal, default page for the Parallel I/O facility).

Either of these two pages is accessible to the control unit logical device (Device 0); however, only page 7 is accessible to the Parallel I/O logical device (Device 4). Hence, to operate with 120K of DACU storage available for Parallel I/O operations, two logical devices must be employed. Data transfers to or from the 120K area must involve Device 0. Device 4 can be used to write control space commands to start up a physical device on the Parallel I/O interface.

It is important to recall that a bit (Bit 15) in the Parallel I/O Control Register (UBCTL) must be set to permit DMA data transfers to or from Page 6. If this bit is clear, DMA is permitted between the device and Page 7 only.

DACU Interrupt Handling for Parallel I/O Facility

It is important to understand the operations that take place when an interrupt is received from a device attached to the Parallel I/O facility.

First, an interrupt cannot occur when the DACU is processing commands received from the host, such as control space interpreter commands. This is because the interrupts and the functional code are both at Level 4 of the DACU System Unit's interrupt structure. Therefore, interrupts can be enabled by host commands or by customer supplied code, but the interrupt will not occur until DACU processing returns to a lower level, such as the Dispatcher.

When an interrupt occurs, the following events happen:

1. The interrupt structure is changed so that only interrupts of Level 0, 1, 2, or 3 can occur.

2. If a customer interrupt entry point is loaded:

   a. A Far CALL is issued to start customer supplied code.

   b. Upon return from the user code, the interrupt processing is terminated and the interrupted routine regains control.

3. If no customer interrupt entry point is loaded:

   a. The UBSTAT register is read and saved so that the UBCTL register can be restored after processing the interrupt.

   b. Further bus requests (BRs) are masked so that other interrupts cannot occur.

   c. The interrupt vector is read.

   d. An Attention is sent to the host.

   e. The interrupt processing is terminated and the interrupted routine regains control.

   f. When a Read XY position is received, the UBCTL register is returned to its previous state by default Attention Sent code.

Read XY-Position Required after Attention

The Attention Sent referred to above occurs when a "Read XY-Position" command is received at the DACU after an Attention has been received by the host. The use of this command guarantees that the host has accepted the Attention. It is important to note that one must provide this command whether default or local code is being run.

If GAM is used on the host, the Read XY-Position is sent to the DACU upon receipt of a light pen attention. However, if GAM is not used, it is necessary for the application program to include a Read XY-Position to signal to the DACU that the attention has been accepted by the host.

If a Read XY-Position is not received by the DACU after an attention is sent to the host, no further attentions will be sent and an error condition will result. However, this state of "attention pending" that occurs between the sending of an attention and the receipt of the Read XY-Position can be cleared by sending a Read XY-Position command. The sample host programming library HALLS performs a Read XY Position in its attention handling routines DATTNW and DATTNI. See Appendix A, "Sample Host Programming" on page 147.

Programming for Serial I/O Devices

The DACU has two Serial I/O ports which can drive serial devices at up to 19,200 bits per second asynchronously via half- or full-duplex communication. The first step in preparing to use a Serial I/O port involves specifying a number of control parameters which determine data transmission rate, number of stop bits, etc. Characteristics like these are specified by writing appropriate data to the control space for the Serial I/O port and issuing a Set Start Register command with a value of X'0006' to initialize the appropriate serial port. Data to be communicated to the Serial I/O device is then written to the buffer space for the Serial I/O port, and a Set Start Register command with a value of X'0004' is issued to move the data a character at a time out to the device. Input is accomplished by issuing a Set Start Register command with a value of X'0002'. This starts the port in input mode, and after the device has transferred its data to DACU memory, the data can be read by the host channel.

Serial I/O Port One (Channel Address xx1) can run under control of a DMA (Direct Memory Access) module. This allows characters to be sent and received without intervention by the processor. Also, Port One can run under programmed Input/Output control. Serial I/O Port Two (Channel address xx2) runs *only* under programmed Input/Output control. That is, as each character is sent (or received), an interrupt is generated and the processor prepares the next character to be sent (or receives the character). This permits DACU control over the data communication process on a character by character basis. A programmer must be aware of these differences between the ports to permit full utilization of the Serial I/O capabilities of the DACU.

For example, when using DMA control Port One can transmit or receive only a specified number of characters, whereas Port Two can make decisions as the block is being transferred. Therefore, if the DACU is being attached to a device which sends a specified character as an "end of transmission" indicator, programmed I/O must be used because this requires each character to be examined as it arrives.

However, if a specified, fixed number of characters is to be sent or received, Port One, with DMA control, can be used. The advantage of using DMA control is that there is very little processor overhead associated with the DMA transfers that would cause interference with other DACU activity (i.e., OEMI or DMA Serial I/O activities). DMA control operates at speeds up to 19.2 kilobits per second. Programmed I/O operates at speeds up to 9.6 kilobits per second. However, if both ports are used concurrently under programmed I/O, the maximum speed is 4800 bits per second.

XON/XOFF Behavior

The Serial I/O ports can be used with the XON/XOFF protocol. If this protocol is specified output from the DACU to the device can be controlled by the device sending the XOFF character to the DACU to stop transmission and an XON character to re-start transmission. When the DACU is receiving data from the device, the XON/XOFF protocol is only used to stop the device from sending if the DACU is in a state that might cause it to lose a character. This happens if the DACU is busy handling a channel command. The XON/XOFF protocol is not used to indicate the "end of buffer" during input. If it is desired to have a device stop sending to the DACU when the buffer is full this can be accomplished by monitoring the RTS/CTS lines under either terminal mode or modem mode.

Terminal/Modem Behavior

The Serial I/O ports can behave as either Data Terminal Equipment (DTE) or Data Communication Equipment (DCE). However, the default arrangement is as a terminal. The connectors on the Interface Unit are male 25 pin EIA. If the DACU is to behave as a modem, a "flip" cable must be used. (A flip cable is used to interchange the Transmit Data and Receive Data lines, the CTS/RTS lines, and the DSR/DTR lines, and it holds Data Carrier Detect active.)

The terminal and modem modes are usually used as half-duplex modes. That is, data transmission is in only one direction at a time. The IGNORE mode is usually used for full duplex transmission. However, it can also be used for half-duplex. In half-duplex, there has to be a convention so that the data terminal equipment and the data communication equipment know when to reverse the direction of data flow. This can be done by an agreed upon "end of transmission" character or by altering the states of the control lines.

In the Serial I/O protocol, a terminal requests to send by turning the RTS (Request to Send) ON. The modem responds, when it is ready to receive data, by turning ON CTS (Clear to Send). The data can then flow from terminal to modem. To reverse the direction of flow, the terminal turns RTS OFF and the modem responds by turning CTS OFF. Data can then flow from the modem to the terminal.

Another way to establish half-duplex communication is by sending a character at the end of a data stream which signals "end of transmission." This might be a Carriage Return or the EOT character. When this character is received, the two devices reverse the direction of flow.

Full duplex means that both the Transmit Data and Receive Data lines are open at all times. Even though blocks of data can be sent in both directions at the same time this mode is usually used to control the rate of data transmission or to provide an "echo" function. To control the flow of transmission, one device sends an agreed upon XOFF character. This signals the other device to stop transmitting until an XON character arrives. In echo mode, each character sent on the transmit data line is echoed back on the receive data line. This gives the user confirmation that each character reached its destination.

A description of the DACU modes follows:

- **TERMINAL MODE (default)**

  Transmit—In this mode Request to Send (RTS) is raised and characters are sent if Clear to Send (CTS) is ON. If CTS turns OFF, (generally when the Serial I/O device is no longer able to accept characters), character transmission stops.

  Receive—When the DACU is ready to receive data it turns the RTS line OFF. If it can no longer receive data it turns RTS back on.

- **MODEM MODE**

  In modem mode, it is assumed that a flip cable is attached to the port on the DACU.

  Transmit—The DACU can transmit whenever the device's DTR line is ON. Due to the presence of the flip cable, this is reflected to the DACU as the Data Set Ready (DSR) line.

  Receive—The DACU can receive only after the device turns its Request-to-Send (RTS) line ON which is seen at the DACU as CTS. The DACU turns its TRS ON which is seen at the device as CTS to indicate that the device can begin sending. If a condition such as "buffer full" occurs, the DACU's RTS is turned off to indicate that the device should stop sending.

Parity Interpretation

When a character is transmitted using asynchronous communication, it is constructed of a start bit, seven or eight information bits (the encoded character), optionally a parity bit, and one or more stop bits. The character data is sent with the least significant bit first.

As outlined in the section on Control Parameters that follows, it is possible to set even parity, odd parity, or no parity. Parity control is established by setting or not setting the parity bit to assure that the total number of data bits (not counting start, stop, or parity bits) is an even or odd number.

The designation "no parity" means that no parity bit is generated on transmission or checked for on reception. That is, if seven bits are desired for character data, a total of nine bits must be transmitted for each character: one start bit, seven character data bits, and one stop bit. Some devices use the term "no parity" to signify that the parity bit is present but is not used. That is, it is set on transmission either always at a zero ("space parity") or always at a one ("mark parity"), and it is assumed to be present in that state on reception (but not checked). For example, if a device requires seven data bits plus space parity, eight bits should be specified with "no parity." For a device requiring seven data bits plus mark parity, specify eight bits with "no parity." But, in this second case where mark parity requires the parity bit position to be a one, change the translation table to make the high order bit for each character always one. (Alternately, code translation in the DACU can be bypassed and the parity bit can be forced to one in the host computer.)

Control Parameters

There is a control space associated with each Serial I/O port. Specific areas in the control space are reserved for the purpose of defining control parameters for Serial I/O (RS-232-C) ports. A description of these control bytes is outlined below. When modifying the control parameters, it is suggested that the content of the control space be read into the host, changes be made there, and then the new contents be written back to the DACU. To modify parameters, leaving the translation tables unchanged, only the first 25 bytes need be read since the translation tables begin at byte 64. However, less than 20 bytes should never be read or written because of the extra byte sometimes transferred by the channel adapter. Once the control space has been written, a Set Start Register command with a value X'0006' is to be issued to initialize the port.

The control parameters are defined in terms of words (two bytes) since data transmitted between control space and the host is always done on a word (16 bits) basis. Within each word there will be byte reversal. That is, the high significance byte will be in the first memory location when the word is in the host and in the second memory location in the DACU Interface Unit or DACU System Unit. The bytes will be referred to (unambiguously) as the *high byte* and the *low byte.*

The default value for each parameter is the one assigned by the DACU initialization code.

Bytes	
(0, 1)	Output buffer. A positive integer in the range 0 to 65535. (Default: 8192)
(2, 3)	Input buffer. (Default: 8192)
(4, 5)	Two byte integer specifying data rate in bits per second. Any value between 110 and 19200 is allowed.
(6, 7)	Number of bits per word and parity. Least Significant Byte: Number of bits per word. Allowed values: 7 or 8. (Default: 7) The number of bits includes only the number of information bits. It does not include the parity or the start and stop bits (e.g., for standard ASCII, the number of bits is 7). Most Significant Byte: Parity. 0 = no parity 1 = odd parity 2 = even parity (default for 7 bits)

Bytes	
(8, 9)	Number of stop bits/echo mode. Least Significant Byte: Number of stop bits. 0 = 1 stop bit 1 = 1.5 stop bits 2 = 2 stop bits Most Significant Byte: Echo. If 0, then no echo (default); if 1, then echo. The echo mode is effective only during input. If echo is on, then each character received in the input buffer is echoed through the output port. In this mode, full duplex cannot be used since the output port is busy echoing characters.
(10, 11)	Input termination. Least Significant Byte: Input termination character. Single byte character. (Default: X'04' = ASCII EOT) When this character is encountered in the input stream, the input operation is terminated, and an attention is sent to the host indicating completion of input. No translation is done on this character. That is, the character must be specified as it appears at the port. The input termination character is not placed in the DACU buffer. Most Significant Byte: Enable (default)/disable input termination character. If 1 (default), the termination character mode is enabled. In this mode, an input operation is terminated when the input termination character is encountered in the input stream. When disabled (set to 0), no test is made for a termination character, and input is continued until the buffer fills or until a timeout (if enabled) occurs.

Bytes	
(12, 13)	Mode Least Significant Byte: Control mode. This Mode is used to affect how the Serial I/O control lines are used. By using the proper mode, devices can be supported that do not follow all of the standard Serial I/O conventions. **0** - Terminal mode. (Default) In this mode, the Serial I/O port behaves as a terminal. That is, in output, it will turn RTS (Request To Send) ON and wait until CTS (Clear To Send) is on before transmitting. If the CTS signal goes off, transmission is suspended. In input, the port will receive data whenever the DSR (Data Set Ready) and DCD (Data Carrier Detect) lines are on. **1** - Modem mode. In this mode the DACU acts as a modem. In the modem mode, a "flip" cable is usually necessary. This cable interchanges Transmit Data/Receive Data (2 and 3), Request to Send/Clear to Send (4 and 5), and Data Terminal Ready/Data Set Ready (6 and 20) lines. In addition, RTS (4) on each end is connected to DCD (8) at the other end. In this modem mode, the DACU raises DTR (which is seen at the end of the cable as DSR) when it is ready to receive data. To send data to the DACU the device must first raise RTS (Request To Send) (which is seen at the DACU as CTS). The DACU then raises RTS (which is seen at the device end of the cable as CTS and DCD) when it is ready to receive data. If the device sends data when CTS is not active, there may be an overrun of data. **2** - Ignore Mode. This mode is useful for attaching to devices that do not follow either of the above protocols. In Ignore mode, the DTR signal is ignored on transmitting and the RTS/CTS sequence is ignored on input. The DTR and RTS signals are set ON at all times. If a flip cable is used, the device will see DSR and CTS active at all times. **3** - Set Mode. Set Mode is like Ignore Mode in that DTR and RTS are ON all of the time. However, the DACU will not send data until CTS is ON and will not receive data until DCD is ON. Most Significant Byte: XON/XOFF Protocol. When set to 1, the XON/XOFF protocol is used. When set to 0, XON/XOFF is disabled (default).

Bytes	
(14, 15)	XON/XOFF Characters Least Significant Byte: XON Character. Default = X'11' (ASCII DC1) Most Significant Byte: XOFF Character. Default = X'13' (ASCII DC3)
(16, 17)	Receive Time-Out (Default: 0) This parameter allows the termination of input if a period of time goes by without any activity on the input port. If set to zero (default), there is no timeout, and the port will only terminate on the reception of an input termination character or an overflow of the input buffer. A time-out specification other than zero indicates the number of timer counts to wait until terminating input. (There are approximately 18.2 timer counts per second.)

Bytes	
(18, 19)	Translation/DMA Least Significant Byte: **1** - Translate (Default). In this mode, translation is made between the EBCDIC codes sent from the channel to characters as seen at the port codes according to the translation tables (see below). The default translation is from EBCDIC to ASCII on output and from ASCII to EBCDIC on input. (The translation table used is the same one used by the IBM 3101 ASCII terminal.) Note, however, that this translation is from the 8 bit EBCDIC code to the 7 bit ASCII code. Therefore, not all EBCDIC characters will be translated. If an invalid code is received, it is transmitted as the ASCII null (0). **0** - No translation. In this mode, no translation is done. Most Significant Byte: **1** - Port does transfer via DMA. **0** - Port does transfer via Programmed I/O. Port 1 supports both DMA transfers and programmed I/O. However, Port 2 supports only programmed I/O.

Bytes	
(64-319)	Input Translate Table. See Appendix B, "Serial I/O Translation Tables" on page 191 for the definition of the translation for a character received from the device to the character sent to the host when ASCII translation is selected. For example, if an ASCII "A" is received, it will have the value of X'41'. If this is used as an index into the input translate table, the result will be X'C1' which is an EBCDIC "A." Since ASCII is a 7 bit code the table is only 128 bytes long.
(320-575)	Output Translate Table. See Appendix B, "Serial I/O Translation Tables" on page 191 for the definition of the translation for a character sent from the host to the device when ASCII translation is selected. For example, if an EBCDIC "A" is sent, it will have the value of X'C1'. If this is used as an index into the output translate table, the result will be X'41' which is an ASCII "A."

Chapter 4. Local DACU Programming

Application- and device-specific functions may be programmed by the customer using the normal development tools available for the IBM PC. These include the Disk Operating System (DOS) as well as DOS-compatible tools like the Macro Assembler, PASCAL Compiler, and FORTRAN Compiler.

The base control program provided with the DACU will be loaded at start-up time, and initialization tasks will be performed. At the end of normal initialization, control will be retained by the Dispatcher. If customer local code is to be executed, the user indicates this (through a parameter[4]), and control is passed to DOS with the DACU control program remaining resident. The user then invokes the customer local code.

DACU Control Program Characteristics

The DACU control program includes a driver component for each device adapter (OEMI, Parallel I/O Interface, and Serial I/O Interface) and a Dispatcher. The Dispatcher is the control program component responsible for coordinating the flow of tasks from one driver to another, or to customer local code.

Control programming provided to manage the control unit functions and logical device behavior is interrupt driven. Once initialized, execution will begin only when an interrupt is received from one of the DACU adapters (i.e., OEMI, Parallel I/O Interface, or Serial I/O Interface). Each control program driver component operates on a unique interrupt level as summarized below:

Adapter Driver (or Function)	Interrupt Level
Serial I/O	2
Customer Code from Serial I/O Entry Point	2
OEMI	3
Parallel I/O	4
Customer Code from Parallel I/O Entry Point	4
Customer Code from DOS	7

---

4 This parameter is communicated via the Dispatcher menu at DACU initialization time. (See the section on DACU Initialization.)

Adapter Driver (or Function)	Interrupt Level
Customer Code from Background	7
Serial I/O Complete	7

The adapter drivers receive control directly without going through any supervisory code. Interrupt vectors are initialized to entry points within the specific drivers. Once an adapter driver receives control, it has control of the DACU until a higher level interrupt occurs, until it relinquishes control to the Dispatcher, or until the routine makes a normal exit.

Customer Local Programming Structure

It is expected that customers will desire to write device- or application-dependent, DACU resident code to satisfy unique requirements of their installation. A customer programming interface is provided that satisfies these requirements.

Any customer local initialization code must include linkage to the base control program, which will call the specified entry points to customer local application code. The linkage between customer local code and the base DACU control program is accomplished with a software interrupt, which is handled by the customer interface portion of the Dispatcher. This linkage requires no address resolution either at link time or at load time.

Customer local code for the Parallel or Serial I/O devices can have up to three entry points that can be set up dynamically by the customer program:

1. **INTERRUPT Entry Point**—Control is passed to this address by the control program for a logical device when hardware attached to its interface issues an interrupt. If no customer entry point is specified, default code disables interrupts, reads the interrupt vector, sends an attention to the host system, and sets an Attention Pending bit so subsequent attentions will not be sent until the current one is processed. If a customer entry point is specified, control is passed immediately to the local code. It is the responsibility of the local code to read the interrupt vector in order to allow transfers on the Parallel I/O Interface.

2. **SET START REGISTER Entry Point**—Control is transferred to this address when the host system signals a Set Start Register with the value X'0008' to a specific logical device. When control is returned to the DACU control program, a Device End is sent to the host system. If no customer entry point is specified, default code presents a Device End to the host system.

3. **ATTENTION SENT Entry Point**—Control is transferred to this address when an attention has been sent to the host system and a Read XY Position has been received from the host. Before calling the entry point, the Attention Pending bit is turned off so a subsequent attention can be sent. If no customer entry point is specified, default code re-enables interrupts.

A CALL (far) instruction is used to pass control to these entry points. And, a RET (far) instruction must be used in all cases to return control to the calling routine.

The three entry points are available for the Parallel and Serial I/O logical devices, but do not exist for the OEMI logical device.

A library of macros and subroutines is provided on the diskette that is shipped with the DACU Interface Unit. These allow the following functions to be performed by the application code:

- Send an attention to the host system associated with a specific logical device
- Set up entry points for a particular logical device
- Enqueue a message to be displayed on the IBM Monochrome Display if attached
- Return control to the base control program after customer provided routine execution
- Enqueue a DACU-related task for execution on a background level
- Immediately display a number on the LED's inside the DACU.

The procedure by which customer provided DACU code interacts with the base DACU control code (shipped with each unit) is as follows:

1. The base DACU control program (an IBM PC DOS .COM file) is initiated. An 'R' parameter is passed to the code which indicates that, after initiation, the program should return to DOS but remain resident.
2. The customer provided code (usually an IBM PC .EXE file) is invoked.
3. The customer provided code passes information about its entry points (via provided software interrupts) to the base DACU control program via the USETUP and SSETUP macros or subroutines.
4. Control is passed from the customer provided code to the control program via the SLEEP macro or subroutine.
5. Operation of the customer provided entry point code begins when the control program is interrupted by the DACU adapter corresponding to the type of entry point provided.

All of the facilities associated with the DACU System Unit are available to the application code. These include all of memory physically located in the System Unit, 128K bytes of memory in the DACU, and the 8K Parallel I/O device address space[5]. (There is no storage protection provided.)

---

5 N.B., The 8K I/O device address space is not physically located in the DACU; it is located in the facilities of the I/O devices attached to the Parallel I/O interface.

Control Space Areas

The control spaces for each of the devices are stored in Page 5 at the following offsets:

	OFFSET	SIZE
DEVICE 0	X'E000'	X'0010'
DEVICE 1	X'E010'	X'0300'
DEVICE 2	X'E310'	X'0300'
DEVICE 3	X'E610'	X'0020'
DEVICE 4	X'E630'	X'0200'

Utility Macro Libraries

Utilities in three classes — General, Parallel I/O, and Serial I/O — are provided on the diskette supplied with the DACU for use by the customer in application code. The utilities are made up of macros and equates and are contained in three separate libraries that may be directly included into customer source code.

General Utility Macro Library

This library contains macros that are not specific to a logical device. To access the library, use the IBM PC DOS Macro Assembler Pseudo-Op INCLUDE to the file GENMAC.LIB, which will include it directly into the source code.

SLEEP: Return to base DACU control program.

Format: SLEEP

Purpose: This macro is used to return control to the base DACU control program when customer provided code has completed initialization processing. This macro should only be called once, from an initialization routine. To exit any other routine, called from the control program, a RET (far) should be used.

Remarks: Once the SLEEP is performed, no customer code will be executed until one of the following events occurs:

- An interrupt occurs for which the customer code has previously identified a service routine entry point.
- The host system performs Set Start Register with a value of X'0008' to begin execution of the Set Start Register service routine whose entry point has previously been set up by customer code.
- An Attention-Sent routine is invoked due to a Read XY Position command in response to an attention being accepted by the host system.

- A background job is dispatched that has been enqueued by the customer.

The linkage to the supplied control code is via interrupt X'92' and is handled by the customer interface portion of the Dispatcher.

HATTN: Enqueue an attention to be sent to the host system.

Format: HATTN SB1DA,SB2SB3

Where: SB1DA High Byte = Sense Byte 1.
SB1DA Low Byte = Device Address to send attention.
SB2SB3 High Byte = Sense Byte 2.
SB2SB3 Low Byte = Sense Byte 3.

(See the earlier sections on channel command definitions for the DACU logical devices for details on Sense Byte definitions.)

Purpose: This macro is used to enqueue an attention to be sent to the host system when the Dispatcher regains control. A return code from the macro is placed in the AL register. If AL is 0, the attention was queued successfully, otherwise it could not be queued.

Remarks: The linkage to the supplied control code is via interrupt X'91' and is handled by the customer interface portion of the Dispatcher. Registers AX and DX are used by the macro.

DISPLAY: Enqueue a message to write on the display.

Format: DISPLAY MSG

Where: MSG = DOS format message to be written.

Purpose: This macro is used to queue a message to be written on the IBM Monochrome Display, if attached to the DACU System Unit. The message must end with the character '$'. It will written after control is returned to the Dispatcher by a RET (far) instruction from one of the entry point routines or a background routine, or a SLEEP macro call from the user initialization routine. The DS register must point to the segment of the message before calling the macro.

Remarks: The linkage to the supplied control code is via interrupt X'94' and is handled by the customer interface portion of the Dispatcher. The DX register is used by the macro.

BATCH: Enqueue a background job.

Format: BATCH BATADDR

Where: BATADDR = Pointer to a four byte area containing the offset/segment of the background routine.

Purpose: This macro is used to queue a customer written routine to be executed on the background level of the System Unit. When the Dispatcher regains control, via a RET (far) instruction or SLEEP macro call, it

will begin execution of the background routine if no interrupts are pending. The background routine will execute at interrupt level 7 and may be interrupted by any higher level interrupt. The DS register must point to the segment of the pointer before calling the macro.

Remarks: The linkage to the supplied control code is via interrupt X'95' and is handled by the customer interface portion of the Dispatcher. The DX register is used by the macro.

CONVW: Convert a word from hexadecimal to ASCII.

Format: CONVW INHEX,OUTASC

Where: INHEX = Hexadecimal format word to be converted.
OUTASC = 4 byte ASCII format output from the macro.

Purpose: This macro is used to convert one word from hexadecimal to ASCII format. Conversion is performed by accessing a translation table. The table is in a separate file, called HEXTOASC.TAB, and must be included directly into the customer data segment, using the Macro Assembler Pseudo-Op INCLUDE, if this macro is to be used.

LED: Write to DACU LED's.

Format: LED VALUE

Where: VALUE = Hexadecimal word to be written.

Purpose: This macro is used to immediately write a hexadecimal word to the LED's inside the DACU. It may be used at any time and is primarily useful in code debugging.

The files GENMAC.LIB and HEXTOASC.TAB contain the following source code:

```
;
;       GENMAC : General Usage Macro Library
;
HATTN   MACRO   SB1DA,SB2SB3    ;       HOST ATTENTION MACRO
;     TEST FOR PARM EXISTENCE (GENERATE ASSEMBLY ERRORS)
        IF1
        IFB     <SB1DA>         ;IF NO SB1DA, DISPLAY ERROR
           %OUT HATTN MACRO USED MISSING FIRST PARAMETER
        ENDIF
        IFB     <SB2SB3>        ;IF NO SB2SB3 DISPLAY ERROR
           %OUT HATTN MACRO USED MISSING SECOND PARAMETER
        ENDIF
        ENDIF
;     SEND ATTENTION
        MOV     AX,SB1DA        ;LOAD SENSE 1,DEVICE ADDR
        MOV     DX,SB2SB3       ;LOAD SENSE 2 AND 3
        INT     91H             ;POST ATTN INTERRUPT TO HOST
        ENDM
;
SLEEP   MACRO                   ;       SLEEP MACRO
        INT     92H             ;RETURN TO DACU CODE. (SLEEP)
        ENDM
;
DISPLAY MACRO   MSG             ;       DISPLAY MACRO
;     TEST FOR PARM EXISTENCE (GENERATE ASSEMBLY ERROR)
        IF1
        IFB     <MSG>           ;IF NO MSG, DISPLAY ERROR
           %OUT DISPLAY MACRO USED MISSING PARAMETER
        ENDIF
        ENDIF
;     DISPLAY MESSAGE
        LEA     DX,MSG          ;POINT TO DOS FORMAL MSG
        INT     94H             ;WRITE MESSAGE ON DISPLAY
        ENDM
;
BATCH   MACRO   JOB             ;       BATCH MACRO
;     TEST FOR PARM EXISTENCE (GENERATE ASSEMBLY ERROR)
        IF1
        IFB     <JOB>           ;IF NO JOB, DISPLAY WNG MSG
           %OUT BATCH MACRO USED MISSING PARAMETER
        ENDIF
        ENDIF
;     ENQUE JOB
        LEA     DX,JOB          ;POINT DS:DX TO BACKGROUND
;                                JOB ENTRY POINT
        INT     95H             ;ENQUE THE BACKGROUND JOB
        ENDM
CONVW   MACRO   INHEX,OUTASC    ;HEX/ASCII CONVERSION MACRO
;     TEST FOR PARM EXISTENCE (GENERATE ASSEMBLY ERRORS)
        IF1
        IFB     <INHEX>         ;IF NO INHEX, DISPLAY ERROR
           %OUT CONVW MACRO USED MISSING FIRST PARAMETER
        ENDIF
        IFB     <OUTASC>        ;IF NO OUTASC DISPLAY ERROR
           %OUT CONVW MACRO USED MISSING SECOND PARAMETER
        ENDIF
        ENDIF
        PUSH    AX              ;SAVE ALL REGISTERS USED
        PUSH    BX
        PUSH    CX
        PUSH    DX
        MOV     DX,INHEX        ;GET ALL HEX DATA IN REG
```

```
        MOV     BX,OFFSET HEXTAB ;ADDR OF HEX TABLE FOR XLAT
;
        MOV     AL,DH            ;GET HIGH BYTE
        MOV     CL,4             ;SET SHIFT COUNT
        SHR     AL,CL            ;SHIFT OUT LOW NIBBLE
        XLAT                     ;TRANSLATE HIGH NIBBLE
        MOV     OUTASC,AL        ;SAVE HIGH NIBBLE
;
        MOV     AL,DH            ;GET HIGH BYTE
        AND     AL,0FH           ;GET LOW NIBBLE
        XLAT                     ;TRANSLATE LOW NIBBLE
        MOV     OUTASC+1,AL      ;SAVE LOW NIBBLE
;
        MOV     AL,DL            ;GET LOW BYTE
        SHR     AL,CL            ;SHIFT OUT LOW NIBBLE
        XLAT                     ;TRANSLATE HIGH NIBBLE
        MOV     OUTASC+2,AL      ;SAVE HIGH NIBBLE
;
        MOV     AL,DL            ;GET LOW BYTE
        AND     AL,0FH           ;GET LOW NIBBLE
        XLAT                     ;TRANSLATE LOW NIBBLE
        MOV     OUTASC+3,AL      ;SAVE LOW NIBBLE
;
        POP     DX               ;RESTORE REGISTERS
        POP     CX
        POP     BX
        POP     AX
        ENDM
;
LED     MACRO   VALUE            ;  DACU LED WRITING MACRO
;     TEST FOR PARM EXISTENCE (GENERATE ASSEMBLY ERROR)
        IF1
        IFB     <VALUE>          ;IF NO VALUE, DISPLAY ERROR
           %OUT LED MACRO USED MISSING PARAMETER
        ENDIF
        ENDIF
;     WRITE TO DACU LED'S
        PUSH    ES               ;SAVE ALL REGISTERS USED
        PUSH    AX
        PUSH    CX
        MOV     AX,5000H
        MOV     ES,AX            ;SET ES TO DACU I/O SEGMENT
        MOV     AX,VALUE
        MOV     CL,4             ;SET ROTATE COUNT
        ROL     AH,CL            ;SWAP HIGH/LOW NIBBLE IN AH
        MOV     CL,4             ;SET ROTATE COUNT
        ROL     AL,CL            ;SWAP HIGH/LOW NIBBLE IN AL
        MOV     BYTE PTR[ES:0C200H],AL  ;WRITE TO LED REGS
        MOV     BYTE PTR[ES:0C201H],AH
        POP     CX               ;RESTORE REGISTERS
        POP     AX
        POP     ES
        ENDM
;
;       HEXTOASC: HEX TO ASCII CONVERSION TABLE
;
HEXTAB DB       30H,31H,32H,33H,34H,35H,36H,37H
       DB       38H,39H,41H,42H,43H,44H,45H,46H
```

Parallel I/O Utility Macro Library

This library contains macros and symbolic pointer declarations. To access the library, use the Macro Assembler Pseudo-Op INCLUDE to the file UBUTIL.LIB, which will include it directly in the source code. The library will permit customer written code to service attached UNIBUS-compatible devices in a customized manner.

USETUP: Set up Parallel I/O entry points.

Format: USETUP TABLE,UCERR

Where: TABLE = Twelve byte table of entry points in the following format:

Bytes 0- 3: offset/segment of Interrupt service routine.

Bytes 4- 7: offset/segment of Set Start Register service routine.

Bytes 8-11: offset/segment of Attention Sent service routine.

UCERR = OPTIONAL Unit Check error flag.

Purpose: This macro is used to set up the customer entry points for device interrupts, Set Start Register function calls, and attention-sent events. Once the linkage is set up, execution will return to the caller. The DS register must point to the segment of the table before calling the macro. This macro should only be called once, from an initialization routine. Any table entry that is zero implies that no user entry of that type exists.

An optional Unit Check error flag, containing any value, should be specified if it is desired to be able to send a Device End/Unit Check upon exiting a Set Start Register service routine instead of the default Device End. The Device End/Unit Check would be sent if there was an error in Set Start Register code processing, which would be indicated by placing a X'4321' in the AX register before exiting the Set Start Register service routine with a RET (far).

Remarks: The linkage to the supplied control code is via interrupt X'90' and is handled by the customer interface portion of the Dispatcher. The DX register is used by the macro.

WUB: Write to a UNIBUS-compatible device register.

Format: WUB ADDR,DATA

Where: ADDR = Address of device register to write.
DATA = 1 word of data to write.

Purpose: This macro is used to write one word of data to a UNIBUS compatible device register. Before calling the macro, the ES register must point to segment X'7000', there must be no device interrupts pending, and NPR's should be turned off by writing to Parallel I/O control register (UBCTL) in segment X'5000'.

Remarks: The macro polls the Parallel I/O status register to determine if a successful write has been performed. If a device interrupt is pending or the device register cannot be written to, the macro will loop forever. Once a successful write has been performed, the UBRDDAT register is read to reset the register. The AX register is used by the macro.

RUB: Read from a UNIBUS-compatible device register.

Format: RUB ADDR

Where: ADDR = Address of device register to read.

Purpose: This macro is used to read one word of data from a UNIBUS compatible device register into the AX register. Before calling the macro, the ES register must point to segment X'7000', there must be no device interrupts pending, and NPR's should be turned off by writing to Parallel I/O control register (UBCTL) in segment X'5000'.

Remarks: The macro polls the Parallel I/O status register to determine if a successful read has been performed. If a device interrupt is pending or the device register cannot be read, the macro will loop forever. Once a successful read has been performed, the UBRDDAT register is read to reset the register.

A set of symbolic pointer declarations provides symbolic pointers to Parallel I/O interface address space, including real memory and I/O device space as well as status and control registers.

The file UBUTIL.LIB contains the following source code:

```
;
;       UBUTIL : Parallel I/O Utility Library
;
USETUP MACRO    TABLE,UCERR      ;     PARALLEL I/O SETUP MACRO
;     TEST FOR PARM EXISTENCE (GENERATE ASSEMBLY ERROR)
        IF1
        IFB     <TABLE>          ;IF NO TABLE, DISPLAY ERROR
           %OUT USETUP MACRO USED MISSING PARAMETER
        ENDIF
        ENDIF
        PUSH    CX
        MOV     CX,0             ;SET UNIT CHECK FLAG OFF
;     TEST FOR OPTIONAL UNIT CHECK PARAMETER
        IFNB    <UCERR>
           MOV  CX,4321H         ;SET UNIT CHECK FLAG ON
        ENDIF
;     LOAD ENTRY POINT TABLE
        LEA     DX,TABLE         ;POINT DS:DX TO ENT PT TABLE
        INT     90H              ;TELL DACU CODE USER ENT PTS
        POP     CX
        ENDM
;
WUB     MACRO   ADDR,DATA        ;WT DATA TO DEVICE REG MACRO
        LOCAL   LBW
;     TEST FOR PARM EXISTENCE (GENERATE ASSEMBLY ERRORS)
        IF1
        IFB     <ADDR>           ;IF NO ADDR DISPLAY ERROR
           %OUT WUB MACRO USED MISSING FIRST PARAMETER
        ENDIF
        IFB     <DATA>           ;IF NO DATA DISPLAY ERROR
```

```
          %OUT WUB MACRO USED MISSING SECOND PARAMETER
        ENDIF
        ENDIF
;     ES MUST BE SET TO X'7000' BEFORE CALLING MACRO
;     WRITE DATA
        MOV     AX,DATA
        MOV     ADDR [1],AH     ;WRITE TO DEVICE/
        MOV     ADDR,AL         ;WAIT UNTIL READY
        PUSH    ES
        MOV     AX,DACUIO
        MOV     ES,AX           ;POINT TO DACU I/O SPACE
LBW:    TEST    ES:UBSTAT,0400H ;READY?
        JZ      LBW             ;WILL LOOP FOREVER IF NO RDY
        MOV     AX,ES:UBRDDAT   ;READ REG TO RESET LATCH
        POP     ES              ;RESTORE ES TO POINT TO ...
        ENDM                    ;PARALLEL I/O SPACE
;
RUB     MACRO   ADDR            ;READ FROM DEVICE REG MACRO
        LOCAL   LBR
;     TEST FOR PARM EXISTENCE (GENERATE ASSEMBLY ERROR)
        IF1
        IFB     <ADDR>          ;IF NO ADDR, DISPLAY ERROR
          %OUT RUB MACRO USED MISSING PARAMETER
        ENDIF
        ENDIF
;     ES MUST BE SET TO X'7000' BEFORE CALLING MACRO
;     READ DATA INTO AX
        MOV     AH,ADDR [1]     ;ATTEMPT READ/
        MOV     AL,ADDR         ;WAIT TIL READY
        PUSH    ES
        MOV     AX,DACUIO
        MOV     ES,AX           ;POINT TO DACU I/O SPACE
LBR:    TEST    ES:UBSTAT,0400H ;READY?
        JZ      LBR             ;WILL LOOP FOREVER IF NO RDY
        MOV     AX,ES:UBRDDAT   ;READ REG TO RESET LATCH
        POP     ES              ;RESTORE ES TO POINT TO ...
        ENDM                    ;PARALLEL I/O SPACE
;
;                               PARALLEL I/O MEM PTR DEFNS
DACUIO  EQU     05000H          ;DACU I/O SEGMENT
UBPAGE  EQU     07000H          ;PARALLEL I/O BUFFER SEGMENT
UBCTL   EQU     01000H          ;PARALLEL I/O CONTROL REG
UBSTAT  EQU     09000H          ;PARALLEL I/O STATUS REGI
UBINTVEC EQU    09002H          ;PARALLEL I/O INTR VECTOR
UBRDDAT EQU     09006H          ;PARALLEL I/O READ REGISTER
```

Serial I/O Utility Library

When one of the serial ports is initialized from the host system, application control is done completely from the host system. If the customer wishes to write specialized DACU-resident code, that code must provide *all* Serial I/O initialization and interrupt servicing for both devices. This library contains macro and symbolic pointer definitions which will assist in accomplishing those functions. To access the library, use the Macro Assembler Pseudo-Op INCLUDE to the file SERUTIL.LIB, which will include it directly in source code. Refer to documentation on the Intel 8274 Multi-Protocol Serial Controller chip (MPSC) for information on specialized Serial I/O programming.

SSETUP: Set up Serial I/O entry points.

Format: SSETUP TABLE

Where: TABLE = Twelve byte table of entry points in the following format:

Bytes 0- 3: offset/segment of Interrupt service routine.

Bytes 4- 7: offset/segment of Set Start Register service routine.

Bytes 8-11: offset/segment of Attention Sent service routine.

Purpose: This macro is used to set up the customer entry points for Serial I/O device interrupts, Set Start Register function calls, and attention-sent events. Once the linkage is set up, execution will return to the caller. The DS register must point to the segment of the table before calling the macro. This macro should only be called once, from an initialization routine. Any table entry that is zero implies that no user exit routine of that type exists.

Remarks: The linkage to the supplied control code is via interrupt X'96' and is handled by the customer interface portion of the Dispatcher. The DX register is used by the macro.

A set of symbolic pointer declarations provides symbolic pointers to Serial I/O address space, as well as status and control registers.

The file SERUTIL.LIB contains the following source code:

```
;
;       SERUTIL : Serial I/O Utility Library
;
SSETUP  MACRO   TABLE           ;     SERIAL SETUP MACRO
;    TEST FOR PARM EXISTENCE (GENERATE ASSEMBLY ERROR)
        IF1
        IFB     <TABLE>         ;IF NO TABLE, DISPLAY ERROR
           %OUT SSETUP MACRO USED MISSING PARAMETER
        ENDIF
        ENDIF
;    LOAD ENTRY POINT TABLE
        LEA     DX,TABLE        ;POINT DS:DX TO ENT PT TABLE
        INT     96H             ;TELL DACU CODE USER ENT PTS
        ENDM
;
;                                SERIAL PORT MEM PTR DEFNS
DACUIO  EQU     05000H          ;DACU I/O SEGMENT
SERPAGE EQU     06000H          ;SERIAL BUFFER SEGMENT
;
CHADATR EQU     0C040H          ;CHANNEL A DATA, READ
CHADATW EQU     04040H          ;CHANNEL A DATA,WRITE
CHASTA  EQU     0C042H          ;CHANNEL A STATUS, READ
CHACMD  EQU     04042H          ;CHANNEL A COMMAND, WRITE
CHBDATR EQU     0C041H          ;CHANNEL B DATA, READ
CHBDATW EQU     04041H          ;CHANNEL B DATA, WRITE
CHBSTA  EQU     0C043H          ;CHANNEL B STATUS, READ
CHBCMD  EQU     04043H          ;CHANNEL B COMMAND
;
SIOCCTL EQU     04100H          ;SERIAL I/O CLOCK CONTROL
SIOSTAT EQU     0C100H          ;SERIAL I/O STATUS REGISTER
;                                GIVES STATUS OF DSR,PARITY
;
; 8253 TIMER
```

```
;
LDCTR0  EQU     04010H  ;TIMER, LOAD COUNTER 0
LDCTR1  EQU     04011H  ;TIMER, LOAD COUNTER 1
LDCTR2  EQU     04012H  ;TIMER, LOAD COUNTER 2
;
LDMODEWD EQU    04013H  ;TIME, WRITE MODE WORD
;
RDCTR0  EQU     0C010H  ;TIMER, READ COUNTER 0
RDCTR1  EQU     0C011H  ;TIMER, READ COUNTER 1
RDCTR2  EQU     0C012H  ;TIMER, READ COUNTER 2
```

Sample Assembler Programs

The following code is an example of a customer initialization routine for a UNIBUS-compatible device, using the Parallel I/O Utility and General Usage Macro Libraries:

```
        .SALL
        PAGE    ,132
        TITLE USER User Initialization Program
;
        EXTRN   START:FAR       ;SET START REG SVC ROUTINE
;INCLUDE PARALLEL I/O AND GENERAL USAGE MACRO LIBRARIES
IF1
        INCLUDE B:GENMAC.LIB
        INCLUDE B:UBUTIL.LIB
ENDIF
;
;******** DATA SEGMENT ********
;
USERDS  SEGMENT PUBLIC 'DATA'
; DEC 13 = CARRIAGE RETURN, DEC 10 = LINE FEED, DEC 36 = $
;INITIAL MESSAGE
ID      DB      'USER CODE, 1/10/84',13,10,36
WNG9330 DB      'WUSR9330 UNABLE TO ENQUEUE '
        DB      'ATTENTION          ',13,10,36
MSG     DB      'MESSAGE NUMBER '
NUMBER  DB      '    .',13,10,36
        EVEN
;MAKE BRANCH TABLE SO DACU CODE CAN FIND ENTRY POINTS
ENTADDR DW      INTENT,USERCS   ;INTERRUPT ENTRY POINT
        DW      SSRENT,USERCS   ;SET START REG ENTRY POINT
        DW      ATNENT,USERCS   ;ATTENTION SENT ENTRY POINT
BATPTR  DW      BATENT,USERCS   ;BATCH ROUTINE ENTRY POINT
UCERR   DW      0               ;UNIT CHECK ERROR FLAG
        INCLUDE B:HEXTOASC.TAB
USERDS  ENDS
;
;******** CODE SEGMENT ********
;
USERCS  SEGMENT PUBLIC 'CODE'
        PUBLIC  USER
USER    PROC    FAR
        ASSUME  CS:USERCS,DS:USERDS
        MOV     AX,USERDS
        MOV     DS,AX           ;SET LOCN OF DATA SEGMENT
        DISPLAY ID              ;WRITE ID MSG
        MOV     UCERR,1         ;SET UNIT CHECK ERROR FLAG
        USETUP  ENTADDR,UCERR   ;TELL DACU CODE ENTRY POINTS
        BATCH   BATPTR          ;ENQUE A BACKGROUND JOB
        SLEEP                   ;EXIT INITIALIZATION (SLEEP)
;DONE FOR NOW.  WAIT FOR CALL FROM DACU CODE
;
;******** SET START REGISTER ENTRY POINT ********
```

```
;
SSRENT: CALL    START           ;CALL EXTERNAL START ROUTINE
        RET
USER    ENDP
;
;******* INTERRUPT ENTRY POINT *********
;
INTENT  PROC    FAR
        PUSH    ES
        MOV     AX,DACUIO
        MOV     ES,AX           ;POINT TO DACU I/O PAGE
        MOV     ES:UBCTL,3FBEH ;BLOCK PARALLEL I/O ACTIVITY
        MOV     DX,ES:UBINTVEC ;READ INTERRUPT VECTOR
;WAIT UNTIL INTERRUPT HAS BEEN RESET
WAIT:   MOV     AL,ES:UBSTAT    ;CHECK STATUS REG
        TEST    AL,80H
        JNE     WAIT            ;LOOP UNTIL INTERRUPT RESET
;
; SEND ATTN INTERRUPT TO HOST
        MOV     AX,9004H        ;INT VEC IN SB2SB3, DEVICE 4
        HATTN   AX,DX           ;SEND ATTN
        OR      AL,AL           ;CHECK RETURN CODE
        JZ      INTRET
        PUSH    DS              ;SAVE CALLER'S DS
        MOV     AX,USERDS
        MOV     DS,AX           ;SET DS TO USER DATA SEG
        DISPLAY WNG9330         ;SEND WNG 9330
        POP     DS              ;RESTORE CALLER'S DS
INTRET: POP     ES              ;RESTORE CALLER'S ES
        RET                     ;RETURN TO DACU CODE
INTENT  ENDP
;
;******* ATTENTION SENT ENTRY POINT ********
;
ATNENT  PROC    FAR
        PUSH    ES
        MOV     AX,DACUIO
        MOV     ES,AX           ;POINT TO DACU I/O PAGE
        MOV     ES:UBCTL,3FC0H ;OPEN PARALLEL I/O ACTIVITY
        POP     ES
        RET                     ;RETURN TO DACU CODE
ATNENT  ENDP
;
;******* BACKGROUND ROUTINE ENTRY POINT *******
;
BATENT  PROC    FAR
        PUSH    DS
        MOV     AX,USERDS
        MOV     DS,AX           ;SET DS TO USER DATA SEG
        CONVW   0001H,NUMBER    ;CONVERT MSG NUMBER TO ASCII
        DISPLAY MSG             ;ENQUEUE A MESSAGE
        LED     0001H           ;WRITE TO DACU LED'S
        POP     DS
        RET                     ;RETURN TO DACU CODE
BATENT  ENDP
;
USERCS  ENDS
;
;******************
;  STACK SEGMENT  *
;******************
USERSS  SEGMENT STACK 'STACK'  ;MAKE STACK 512 BYTES
        DB      64 DUP ('STACK   ')
USERSS  ENDS
        END     USER
```

Note that a stack segment is created in the code above. If the customer writes local code a stack must be supplied because the DACU control program will use this stack. A recommended stack size is 512 bytes.

The following sample assembler code is a simple example of a customer provided function that can be initiated by a Set Start Register X'0008' command from the host computer. When the command is executed by the DACU, this code will move the constant X'3FBE' into the Parallel I/O Control Register (UBCTL), write to a sample UNIBUS-compatible device register at location X'F538', and move the constant X'3FC0' into UBCTL. The sample program illustrates proper entry and exit procedures.

```
        PAGE    ,132
        .SALL   ;DON'T EXPAND MACROS
        TITLE START: USER SET START REGISTER ROUTINE
IF1
        INCLUDE B:UBUTIL.LIB
ENDIF
;**********************
;**** DATA SEGMENT ****
;**********************
STRTDS  SEGMENT PUBLIC 'DATA'
COUNT   DW      0FFFEH          ;COUNT TO WRITE TO DEV REG
STRTDS  ENDS
;**********************
;**** CODE SEGMENT ****
;**********************
STRTCS  SEGMENT PUBLIC 'CODE'
        PUBLIC  START
START   PROC    FAR
        ASSUME  CS:STRTCS,DS:STRTDS
        PUSH    DS              ;SAVE CALLER'S DS
        PUSH    ES              ;SAVE CALLER'S ES
        MOV     AX,STRTDS
        MOV     DS,AX           ;SET DS TO USER DATA SEG.
        MOV     AX,DACUIO
        MOV     ES,AX           ;SET ES TO DACU I/O PAGE
        MOV     ES:UBCTL,3FBEH  ;BLOCK PARALLEL I/O ACTIVITY
        MOV     AX,UBPAGE
        MOV     ES,AX           ;SET ES TO PARALLEL I/O PAGE
        WUB     ES:0F538H,COUNT ;WRITE TO DEV REG
        MOV     AX,DACUIO
        MOV     ES,AX           ;SET ES TO DACU I/O PAGE
        MOV     ES:UBCTL,3FC0H  ;OPEN PARALLEL I/O ACTIVITY
        POP     ES              ;RESTORE CALLER'S ES
        POP     DS              ;RESTORE CALLER'S DS
        RET
START   ENDP
STRTCS  ENDS
        END
```

Subroutine Libraries

The subroutine libraries described below contain object code for PC Macro Assembler subroutines that can be called from IBM PC PASCAL or IBM PC FORTRAN routines. Both source code and object code is provided to the customer on the diskette supplied with the DACU. To access a library, link its object module to customer object code and the appropriate language library using the linker program (LINK) to create an EXE file. Each subroutine accessed by

PASCAL must be declared as an EXTERN procedure. This is not necessary when using FORTRAN routines.

To create a single set of library routines callable from either FORTRAN or PASCAL, it is necessary that all parameters passed to library procedures be pointers to the required data. FORTRAN parameters supply 32 bit offset/segment pointers transparently, but PASCAL parameters supply only a 16 bit offset to a default data segment using reference parameters, or actual data using value parameters. Therefore, for consistency with FORTRAN parameter passing, each PASCAL parameter must be a segmented machine address pointing to the actual data. This can be accomplished by declaring variables of type ADS of the required data type and assigning the pointers as ADS data. See the sample PASCAL program below for an example.

Many PC high level languages use a DOS interrupt to write to the DACU System Unit Display. If a write is performed and an interrupt from a logical device occurs, undesirable results may occur. It is therefore suggested not to write to the display while the DACU is involved in any I/O. For debugging purposes an LED subroutine has been supplied to write to the DACU LED's at any time.

General Utility Subroutine Library

This library contains procedures that are not specific to a logical device. The object module for the library is named GENSUB.OBJ.

SLEEP: Return to base DACU control program.

Format: SLEEP

Purpose: This subroutine is used to return control to the base DACU control program when customer provided code has completed initialization processing. This subroutine should only be called once, from an initialization routine. To exit any other routine, called from the control program, a standard procedure exit should be used.

Remarks: Once the SLEEP is performed, no customer code will be executed until one of the following events occurs:

- An interrupt occurs for which the customer code has previously identified a service routine entry point.
- The host system performs Set Start Register with a value of X'0008' to begin execution of the Set Start Register service routine whose entry point has previously been set up by customer code.
- An Attention-Sent routine is invoked due to a Read XY Position command in response to an attention being accepted by the host system.
- A background job is dispatched that has been enqueued by the customer.

The linkage to the supplied control code is via interrupt X'92' and is handled by the customer interface portion of the Dispatcher.

HATTN: Enqueue an attention to be sent to the host system.

Format: HATTN(SB1DA,SB2SB3)

Where: SB1DA High Byte = Sense Byte 1.
SB1DA Low Byte = Device Address to send attention.
SB2SB3 High Byte = Sense Byte 2.
SB2SB3 Low Byte = Sense Byte 3.

(See the earlier sections on channel command definitions for the DACU logical devices for details on Sense Byte definitions.)

Purpose: This subroutine is used to enqueue an attention to be sent to the host system when the Dispatcher regains control. If the attention cannot be enqueued successfully, the following warning message is enqueued: WSUB9330 UNABLE TO ENQUEUE ATTENTION.

Remarks: The linkage to the supplied control code is via interrupt X'91' and is handled by the customer interface portion of the Dispatcher.

Using FORTRAN, the two parameters are passed as two byte decimal integers which the customer has converted from the desired hexadecimal values. Using PASCAL, pointers to two hexadecimal valued variables of type WORD are passed.

BATCH: Enqueue a background job.

Format: BATCH(BATADDR)

Where: BATADDR = procedure name of the background routine.

Purpose: This subroutine is used to enqueue a customer written routine to be executed on the background level of the System Unit. When the Dispatcher regains control from a service subroutine exit or SLEEP subroutine call, it will begin execution of the background routine if no interrupts are pending. The background routine will execute at interrupt level 7 and may be interrupted by any higher level interrupt.

Remarks: The linkage to the supplied control code is via interrupt X'95' and is handled by the customer interface portion of the Dispatcher. This subroutine takes a single procedure name as a procedural parameter. Using FORTRAN, the procedure to be run as a background job must be declared as external so that its name may be passed. Using PASCAL, the procedure to be run as a background job must be declared as an EXTERN, and the dummy procedural parameter must be declared as [PUBLIC]. (See the FORTRAN and PASCAL sample programs for examples of these declarations.)

LED: Write to DACU LED's.

Format: LED(NUMBER)

Where: NUMBER = 2 byte hexadecimal word to be written.

Purpose: This subroutine is used to immediately write a hexadecimal word to the LED's inside the DACU. It may be used at any time and is primarily useful in code debugging.

Remarks: Using FORTRAN, the parameter is passed a two byte decimal integer which the customer has converted from the desired hexadecimal value. Using PASCAL, a pointer to a hexadecimal valued variable of type WORD is passed.

DWRITE: Write to DACU memory.

Format: DWRITE(LENGTH,DATA,OFFSET,SEGMENT)

Where: LENGTH = 2 byte length of data to write in bytes.
DATA = Array of data to write.
OFFSET = 2 byte DACU offset address to write data.
SEGMENT = 2 byte DACU segment address to write data.

Purpose: This subroutine is used to write to DACU memory, in page X'5000', X'6000', or X'7000'. A block move is performed from the data array specified in the DATA parameter to the offset/segment address specified by the OFFSET and SEGMENT parameters. The number of bytes moved is specified in the LENGTH parameter. The subroutine may be used to write to the Parallel I/O control register (UBCTL), Serial I/O memory space (X'6000'), and Parallel I/O memory space (X'7000'), but may not used to write UNIBUS-compatible device registers, which are written to with the WUB subroutine.

DREAD: Read from DACU memory.

Format: DREAD(LENGTH,BUFFER,OFFSET,SEGMENT)

Where: LENGTH = 2 byte length of data to read in bytes.
BUFFER = Array to read data into.
OFFSET = 2 byte DACU offset address to read data from.
SEGMENT = 2 byte DACU segment address to read data from.

Purpose: This subroutine is used to read from DACU memory, in page X'5000', X'6000', or X'7000'. A block move is performed from the offset/segment address specified by the OFFSET and SEGMENT parameters to the buffer array specified in the BUFFER parameter. The number of bytes moved is specified in the LENGTH parameter. The subroutine may be used to read the Parallel I/O status register (UBSTAT), Parallel I/O interrupt vector (UBINTVEC), Serial I/O memory space (X'6000'), and Parallel I/O memory space (X'7000'), but may not used to read UNIBUS-compatible device registers, which are read with the RUB subroutine.

The file GENSUB.OBJ is created by assembling the file GENSUB.ASM, which contains the following source code:

```
        PAGE    ,132
        .SALL                   ;DON'T EXPAND MACROS
;
        TITLE   GENSUB : General Usage Subroutine Library
;
;MACROS
MOVE    MACRO   TO,FROM
        MOV     AX,FROM
        MOV     TO,AX
        ENDM
;
GDSEG   SEGMENT PUBLIC  'CODE'
BATADDR DD      0               ;BACKGROUND JOB ADDRESS
GOBATCH DD      0               ;BACKGROUND DISPATCHER ADDR
SAVEDS DW      ?                ;SAVE AREA FOR CALLER'S DS
WNG9330 DB      'WSUB9330 UNABLE TO ENQUEUE ATTENTION'
        DB     13,10,36
GDSEG   ENDS
;
GCSEG   SEGMENT PUBLIC  'CODE'
        ASSUME CS:GCSEG,DS:GDSEG,ES:GDSEG
        PUBLIC  HATTN,SLEEP,BATCH,DWRITE,DREAD,LED
;
HATTN   PROC    FAR             ;HOST ATTENTION SUBROUTINE
        PUSH    BP              ;SAVE CALLER'S BP
        MOV     BP,SP           ;SET FRAME BASE
        MOV     SI,SS:[BP+10]    ;LD ADDR SENSE 1,DEV ADDR
        MOV     ES,SS:[BP+12]
        MOV     AX,ES:[SI]       ;LD SENSE 1, DEV ADDR
        MOV     SI,SS:[BP+6]     ;LD ADDR SENSE 2,3
        MOV     ES,SS:[BP+8]
        MOV     DX,ES:[SI]       ;LD SENSE 2,3
        INT     91H             ;POST ATTN TO HOST
        OR      AL,AL           ;CHECK RETURN CODE
        JE      ATNRET
        PUSH    DS              ;SAVE CALLER'S DS
        MOVE    DS,GDSEG         ;SET DS TO SEG OF WNG MSG
        LEA     DX,WNG9330       ;POINT TO WARNING MSG
        INT     94H             ;SEND WARNING WSUB9330
        POP     DS              ;RETURN CALLER'S DS
ATNRET: POP     BP              ;RETURN CALLER'S BP
        RET     8               ;RETURN TO CALLER
HATTN   ENDP
;
SLEEP   PROC    FAR             ;      SLEEP SUBROUTINE
        PUSH    BP              ;SAVE CALLER'S BP
        INT     92H             ;RETURN TO BASE CODE (SLEEP)
        POP     BP              ;RETURN CALLER'S BP
        RET
SLEEP   ENDP
;
BATCH   PROC    FAR             ;      BATCH SUBROUTINE
        PUSH    BP              ;SAVE CALLER'S BP
        MOV     BP,SP           ;SET FRAME BASE
        PUSH    DS              ;SAVE CALLER'S DS
        MOVE    DS,GDSEG         ;USE DATA SEG AS NEW DS
        POP     AX              ;SAVE DS FROM ENTRY ROUTINE
;                                IN SAVEDS
        MOV     SAVEDS,AX
        MOV     GOBATCH,OFFSET GOBAT ;LOAD OFFSET/SEGMENT
;                                    OF BATCH DISPATCHER
        MOV     GOBATCH+2,CS
        MOV     SI,SS:[BP+8]     ;GET LOCN OF BATCH ADDR
        MOVE    BATADDR,ES:[SI] ;GET BATCH ADDR
        MOVE    BATADDR+2,ES:[SI+2]
```

```
        LEA     DX,GOBATCH      ;POINT DS:DX TO BATCH
;                               DISPATCHER ADDR
        INT     95H             ;ENQUE BATCH JOB .
        MOV     DS,SAVEDS       ;RETURN CALLER'S DS
        POP     BP              ;RETURN CALLER'S BP
        RET     4               ;RETURN TO CALLER
BATCH   ENDP
;
GOBAT   PROC    FAR             ;INTERNAL DISPATCHER FOR
;                               USER BATCH ROUTINES
        PUSH    DS              ;SAVE CALLER'S DS
        MOVE    DS,SAVEDS       ;RETURN USER'S DS
        PUSH    ES              ;SAVE CALLER'S ES
        MOVE    ES,GDSEG
        LEA     BX,ES:BATADDR     ;POINT TO BATCH ADDR
        CALL    DWORD PTR ES:[BX]  ;CALL BATCH ROUTINE
        POP     ES              ;RESTORE CALLER'S ES
        POP     DS              ;RESTORE CALLER'S DS
        RET
GOBAT   ENDP
;
LED     PROC    FAR             ;     LED WRITE SUBROUTINE
        PUSH    BP              ;SAVE CALLER'S BP
        MOV     BP,SP           ;SET FRAME BASE
        PUSH    ES
        PUSH    DS
        PUSH    AX
        PUSH    CX
        MOV     AX,5000H
        MOV     DS,AX           ;SET ES TO DACU I/O SEGMENT
        MOV     SI,SS:[BP+6]    ;LD ADDR LED VALUE
        MOV     ES,SS:[BP+8]
        MOV     AX,ES:[SI]      ;LD LED VALUE
        MOV     CL,4            ;SET ROTATE COUNT
        ROL     AH,CL           ;SWAP HIGH/LOW NIBBLE IN AH
        MOV     CL,4            ;SET ROTATE COUNT
        ROL     AL,CL           ;SWAP HIGH/LOW NIBBLE IN AL
        MOV     BYTE PTR[DS:0C200H],AL  ;WRITE TO DACU LED'S
        MOV     BYTE PTR[DS:0C201H],AH
        POP     CX              ;RESTORE REGISTERS
        POP     AX
        POP     DS
        POP     ES
        POP     BP
        RET     4
LED     ENDP
;
DWRITE  PROC    FAR       ;WRITE TO DACU BUFFER SUBROUTINE
        PUSH    BP              ;SAVE CALLER'S BP
        MOV     BP,SP           ;SET FRAME BASE
        PUSH    DS              ;SAVE CALLER'S DS
        PUSH    ES              ;SAVE CALLER'S ES
        MOV     SI,SS:[BP+18]   ;LOAD BYTE COUNT.
        MOV     DS,SS:[BP+20]
        MOV     CX,DS:[SI]
        MOV     SI,SS:[BP+10]   ;LOAD OFFSET TO WRITE DATA
        MOV     DS,SS:[BP+12]
        MOV     DI,DS:[SI]
        MOV     SI,SS:[BP+6]    ;LOAD SEGMENT TO WRITE DATA
        MOV     DS,SS:[BP+8]
        MOV     ES,DS:[SI]
        MOV     SI,SS:[BP+14]   ;LOAD ADDR OF DATA TO WRITE
        MOV     DS,SS:[BP+16]
REP     MOVSB                   ;BLOCK MOVE DATA.
        POP     ES              ;RESTORE REGISTERS
```

```
        POP     DS
        POP     BP
        RET     16
DWRITE  ENDP
;
DREAD   PROC    FAR         ;READ FROM DACU BUFFER SUBROUTINE
        PUSH    BP              ;SAVE CALLER'S BP
        MOV     BP,SP           ;SET FRAME BASE
        PUSH    DS              ;SAVE CALLER'S DS
        PUSH    ES              ;SAVE CALLER'S ES
        MOV     DI,SS:[BP+18]    ;LOAD BYTE COUNT.
        MOV     ES,SS:[BP+20]
        MOV     CX,ES:[DI]
        MOV     DI,SS:[BP+10]    ;LOAD OFFSET TO READ DATA
        MOV     ES,SS:[BP+12]
        MOV     SI,ES:[DI]
        MOV     DI,SS:[BP+6]     ;LOAD SEGMENT TO READ DATA
        MOV     ES,SS:[BP+8]
        MOV     DS,ES:[DI]
        MOV     DI,SS:[BP+14]    ;LOAD PARM BUFFER ADDRESS
        MOV     ES,SS:[BP+16]
REP     MOVSB                   ;BLOCK MOVE DATA.
        POP     ES              ;RESTORE REGISTERS
        POP     DS
        POP     BP
        RET     16
DREAD   ENDP
;
GCSEG   ENDS
        END
```

Parallel I/O Subroutine Library

This library contains Parallel I/O specific procedures which may be accessed by linking to the file UBSUB.OBJ.

USETUP: Set up Parallel I/O entry points.

Format: USETUP(ATNFLG,SSRFLG,INTFLG)

Where: ATNFLG = 2 byte flag for Attention Sent service routine.
SSRFLG = 2 byte flag for Set Start Register service routine.
INTFLG = 2 byte flag for Interrupt service routine.

Purpose: This subroutine is used to set up the customer entry points for device interrupts, Set Start Register function calls, and attention-sent events. Once the linkage is set up, execution will return to the caller. This subroutine should only be called once, from an initialization routine. The subroutine takes three flags as parameters, indicating if there is an entry point for Attention Sent, Set Start Register, and Interrupt service routines. Using FORTRAN, three LOGICAL*2 variables, which have been set to .TRUE. or .FALSE. as desired, are passed. Using PASCAL, three pointers to three integer variables, which have been set to 1 or 0, are passed. If an entry point flag is set to TRUE (1), the customer provided entry point is used. Upon linkage, the object module containing the entry point must be specified so that the service routine can be reached. If an entry point flag is set to FALSE (0), default code is used. No code for the unused entry point need be specified when linking. Upon linkage, an Unresolved External error

message will be displayed for each unused entry point. This will not affect program execution since the entry point is not actually used. If entry points are to exist, the service routines must have the following names:

Attention Sent: UATTN

Set Start Register: USSREG

Interrupt: UINTR

Remarks: The linkage to the supplied control code is via interrupt X'90' and is handled by the customer interface portion of the Dispatcher.

A Device End/Unit Check cannot be sent if there is an error in Set Start Register processing using high level languages, which could be done if the Set Start Register routine was coded in IBM Macro Assembler. The reason for this is that the error is indicated by setting a value in the AX register before returning to the DACU base control program. The value of the AX register cannot be set by the high level languages, and processing involved in the return is language dependent and may modify the AX register. Therefore high level language Set Start Register service routines will only send a Device End upon their completion.

WUB: Write to a UNIBUS-compatible device register.

Format: WUB(ADDR,DATA)

Where: ADDR = 2 byte address of device register to write.
DATA = 2 byte word of data to write.

Purpose: This subroutine is used to write one word of data to a UNIBUS compatible device register. It is not used to write to DACU memory. The DWRITE subroutine in the General Utility Subroutine Library (GENSUB) should be used for this purpose.

Remarks: If a device interrupt is pending when the subroutine is called, the Parallel I/O interrupt vector (UBINTVEC) is read to clear the interrupt and the following warning message is enqueued: WSUB4005 INTERRUPT RECEIVED DURING R/W OF DEVICE REG XXXX. Then an attempt will be made to write the device register. The subroutine polls the Parallel I/O status register (UBSTAT) to determine if a successful write has been performed. If the write cannot be performed, the following warning message is enqueued: WSUB4103 SOFTWARE TIME OUT OCCURRED WHILE WRITING DEVICE REGISTER XXXX.

RUB: Read from a UNIBUS-compatible device register.

Format: RUB(ADDR,BUFFER)

Where: ADDR = 2 byte address of device register to read.
BUFFER = 2 byte word buffer to read data into.

Purpose: This subroutine is used to read one word of data from a UNIBUS compatible device register. It is not used to read DACU memory. The DREAD subroutine in the General Utility Subroutine Library (GENSUB) should be used for this purpose.

Remarks: If a device interrupt is pending when the subroutine is called, the Parallel I/O interrupt vector (UBINTVEC) is read to clear the interrupt and the following warning message is enqueued: WSUB4005 INTERRUPT RECEIVED DURING R/W OF DEVICE REG XXXX. Then an attempt will be made to read the device register. The subroutine polls the Parallel I/O status register (UBSTAT) to determine if a successful read has been performed. If the read cannot be performed, the following warning message is enqueued: WSUB4103 SOFTWARE TIME OUT OCCURRED WHILE READING DEVICE REGISTER XXXX.

The file UBSUB.OBJ is created by assembling the file UBSUB.ASM, which contains the following source code:

```
        PAGE    ,132
        .SALL                   ;DON'T EXPAND MACROS
;
        TITLE   UBSUB : Parallel I/O Subroutine Library
;
        EXTRN   UINTR:FAR,USSREG:FAR,UATTN:FAR
;MACROS
MOVE    MACRO   TO,FROM
        MOV     AX,FROM
        MOV     TO,AX
        ENDM
;
IF1
        INCLUDE B:GENMAC.LIB
ENDIF
;
UDSEG   SEGMENT PUBLIC  'CODE'
SAVEDS DW       ?               ;SAVE AREA FOR CALLER'S DS
INTADDR DD      0               ;INTERRUPT ENT PT
SSRADDR DD      0               ;SET START REG ENT PT
ATNADDR DD      0               ;ATTENTION SENT ENT PT
STAIMAGE DB     0
WNG4102 DB 'WSUB4102 SOFTWARE TIME OUT OCCURRED WHILE '
        DB 'READING DEVICE REGISTER '
RUREG   DB '      ',13,10,36
WNG4103 DB 'WSUB4103 SOFTWARE TIME OUT OCCURRED WHILE '
        DB 'WRITING DEVICE REGISTER '
WUREG   DB '      ',13,10,36
WNG4005 DB 'WSUB4005 INTERRUPT RECEIVED DURING R/W OF'
        DB ' DEVICE REG',13,10,36
;                               PARALLEL I/O MEM PTR DEFNS
DACUIO EQU      05000H          ;DACU I/O SEGMENT
UBPAGE EQU      07000H          ;PARALLEL I/O DEVICE SEGMENT
UBCTL  EQU      01000H          ;PARALLEL I/O CONTROL REG
UBSTAT EQU      09000H          ;PARALLEL I/O STATUS REG
UBINTVEC EQU    09002H          ;PARALLEL I/O INTR VECTOR
UBRDDAT EQU     09006H          ;PARALLEL I/O READ REGISTER
;
```

```
        INCLUDE B:HEXTOASC.TAB
UDSEG   ENDS
;
UCSEG   SEGMENT PUBLIC  'CODE'
        ASSUME  CS:UCSEG,DS:UDSEG
        PUBLIC  USETUP,WUB,RUB
;
USETUP  PROC    FAR             ;SETUP PARALLEL I/O ENT PTS
        PUSH    BP              ;SAVE CALLER'S BP
        MOV     BP,SP           ;SET FRAME BASE
        PUSH    DS              ;SAVE CALLER'S DS
        MOVE    DS,UDSEG        ;POINT TO OUR DATA SEG
        POP     AX              ;SAVE DS FROM
;                                ENTRY ROUTINE IN SAVEDS
        MOV     SAVEDS,AX
        MOV     SI,SS:[BP+6]    ;GET INTR FLAG ADDR
        MOVE    ES,SS:[BP+8]
        MOV     AX,ES:[SI]      ;GET INTERRUPT FLAG
        CMP     AL,1            ;INTERRUPT ENTRY POINT?
        JNE     USET1           ;IF NO, DON'T CHANGE TABLE
        MOV     INTADDR,OFFSET INTENT ;IF YES MOVE OFFSET,
;                                       SEGMENT TO TABLE
        MOVE    INTADDR+2,CS
USET1:  MOV     SI,SS:[BP+10]   ;GET SET START FLAG ADDR
        MOVE    ES,SS:[BP+12]
        MOV     AX,ES:[SI]      ;GET SET START FLAG
        CMP     AL,1            ;SET START ENTRY POINT?
        JNE     USET2           ;IF NO, DON'T CHANGE TABLE
        MOV     SSRADDR,OFFSET SSRENT ;IF YES MOVE OFFSET,
;                                       SEGMENT TO TABLE
        MOVE    SSRADDR+2,CS
USET2:  MOV     SI,SS:[BP+14]   ;GET ATTN SENT FLAG ADDR
        MOVE    ES,SS:[BP+16]
        MOV     AX,ES:[SI]      ;GET ATTENTION FLAG
        CMP     AL,1            ;ATTN SENT ENTRY POINT?
        JNE     USET3           ;IF NO, DON'T CHANGE TABLE
        MOV     ATNADDR,OFFSET ATNENT ;IF YES MOVE OFFSET,
;                                       SEGMENT TO TABLE
        MOVE    ATNADDR+2,CS
USET3:  LEA     DX,INTADDR      ;POINT DS:DX TO WNT PT TABLE
        INT     90H             ;TELL BASE CODE USER ENT PTS
        MOV     DS,SAVEDS       ;RETURN CALLER'S DS
        POP     BP              ;RETURN CALLER'S BP
        RET     12              ;FAR RETURN, POP 12 BYTES
USETUP  ENDP
;
INTENT  PROC    FAR             ;INTERNAL INTERRUPT ENT PT
        MOV     DS,SAVEDS       ;RESTORE USER ROUTINE DS
        CALL    UINTR           ;CALL PARALLEL INTR ENT PT
        RET
INTENT  ENDP
;
SSRENT  PROC    FAR             ;INTERNAL SET START ENT PT
        MOV     DS,SAVEDS       ;RESTORE USER ROUTINE DS
        CALL    USSREG          ;CALL PARALLEL SET STRT ENT PT
        RET
SSRENT  ENDP
;
ATNENT  PROC    FAR             ;INTERNAL ATTN SENT ENT PT
        MOV     DS,SAVEDS       ;RESTORE USER ROUTINE DS
        CALL    UATTN           ;CALL PARALLEL ATN SENT ENT PT
        RET
ATNENT  ENDP
;
WUB     PROC    FAR             ;WRITE TO DEVICE REG SUB
```

```
        PUSH    BP              ;SAVE CALLER'S BP
        MOV     BP,SP           ;SET FRAME BASE
        PUSH    DS              ;SAVE CALLER'S DS
        PUSH    ES              ;SAVE CALLER'S ES
        MOV     SI,SS:[BP+6]    ;GET DATA ADDR
        MOVE    ES,SS:[BP+8]
        MOV     DX,ES:[SI]      ;GET DATA TO WRITE
        MOV     SI,SS:[BP+10]   ;GET ADDR OF DEVICE REG ADDR
        MOVE    ES,SS:[BP+12]
        MOV     BX,ES:[SI]      ;GET DEVICE REG ADDR
        MOVE    ES,DACUIO       ;POINT TO DACU I/O SPACE
        MOVE    DS,UDSEG        ;POINT TO OUR DATA SEG
        MOV     AL,ES:UBSTAT    ;GET IMAGE OF UBSTAT
        OR      AL,80H          ;NOT INIT
        MOV     STAIMAGE,AL     ;STORE IMAGE OF UBSTAT
        MOV BYTE PTR ES:UBCTL,0BEH ;DISABLE INTERRUPTS
        TEST BYTE PTR ES:UBSTAT,80H ;CHECK FOR EXTRA INTR
        JZ      WNOINT          ;IF NONE, CONTINUE
        PUSH    DX              ;SAVE DATA TO BE WRITTEN
        DISPLAY WNG4005         ;IF FOUND, DISP WNG 4005
        MOV     DX,ES:UBINTVEC  ;READ THE INTR VECTOR
        POP     DX              ;RESTORE DATA TO BE WRITTEN
WNOINT: MOVE    DS,UBPAGE       ;POINT TO PARALLEL I/O PAGE
        MOV     DS:[BX+1],DH    ;WRITE TO DEV REG HIGH BYTE
        MOV     DS:[BX],DL      ;WRITE TO DEV REG LOW BYTE
;                                WAIT TILL KNOW DATA IS THERE
        MOV     CX,1000         ;SET LOOP COUNTER
        MOVE    DS,UDSEG        ;POINT TO OUR DATA SEG
WRDY:   TEST    ES:UBSTAT,0400H ;READY?
        JNZ     WOK             ;IF OK EXIT LOOP
        LOOP    WRDY
WOK:    MOV     AX,ES:UBRDDAT   ;READ REG TO RESET LATCH
        CMP     CX,0            ;CHECK LOOP CNT
        JNE     WEXIT           ;IF NOT 0 WROTE OK, BAD IF 0
        CONVW   BX,WUREG        ;SAVE REG ADDR IN ASCII
        DISPLAY WNG4103         ;DISPLAY WNG 4103
WEXIT:  MOV     AL,STAIMAGE     ;GET UBSTAT IMAGE
        MOV BYTE PTR ES:UBCTL,AL ;RESTORE UBCTL
        POP     ES              ;RESTORE CALLER'S REGS
        POP     DS
        POP     BP
        RET     8               ;RETURN TO CALLER
WUB     ENDP
;
RUB     PROC    FAR             ;READ DEVICE REG SUB
        PUSH    BP              ;SAVE CALLER'S BP
        MOV     BP,SP           ;SET FRAME BASE
        PUSH    DS              ;SAVE CALLER'S DS
        PUSH    ES              ;SAVE CALLER'S ES
        MOV     SI,SS:[BP+10]   ;GET ADDR OF DEV REG ADDR
        MOVE    ES,SS:[BP+12]
        MOV     BX,ES:[SI]      ;GET DEV REG ADDR
        MOV     SI,SS:[BP+6]    ;GET BUFFER ADDR
        MOVE    ES,SS:[BP+8]
        PUSH    ES              ;SAVE SEG OF BUFFER ADDR
        MOVE    ES,DACUIO       ;POINT TO DACU I/O SPACE
        MOVE    DS,UDSEG        ;POINT TO OUR DATA SEG
        MOV     AL,ES:UBSTAT    ;GET IMAGE OF UBSTAT
        OR      AL,80H          ;NOT INIT
        MOV     STAIMAGE,AL     ;STORE IMAGE OF UBSTAT
        MOV     BYTE PTR ES:UBCTL,0BEH ;DISABLE INTR
        TEST BYTE PTR ES:UBSTAT,80H ;CHECK FOR EXTRA INTR
        JZ      RNOINT          ;IF NONE, CONTINUE
        DISPLAY WNG4005         ;IF FOUND, DISPLAY WNG 4005
        MOV     DX,ES:UBINTVEC  ;READ THE INTR VECTOR
RNOINT: MOVE    DS,UBPAGE       ;POINT TO PARALLEL I/O PAGE
```

```
        MOV     DH,DS:[BX+1]    ;ATTEMPT READ HIGH BYTE
        MOV     DL,DS:[BX]      ;ATTEMPT READ LOW BYTE
        MOVE    ES,DACUIO       ;POINT TO DACU I/O SPACE
        MOV     CX,1000         ;SET LOOP COUNTER
        MOVE    DS,UDSEG        ;POINT TO OUR DATA SEG
RRDY:   TEST    ES:UBSTAT,0400H ;READY?
        JNZ     ROK             ;IF OK EXIT LOOP
        LOOP    RRDY
ROK:    MOV     DX,ES:UBRDDAT   ;READ REG TO RESET LATCH
        POP     ES              ;RESTORE SEG OF BUFFER ADDR
        MOV     ES:[SI],DX      ;STORE REG DATA IN BUFFER
        CMP     CX,0            ;CHECK LOOP CNT
        JNE     REXIT           ;IF NOT 0 READ OK, BAD IF 0
        CONVW   BX,RUREG        ;SAVE REG ADDR IN ASCII
        DISPLAY WNG4102         ;DISPLAY WNG 4102
REXIT:  MOVE    ES,DACUIO       ;POINT TO DACU I/O SPACE
        MOV     AL,STAIMAGE     ;GET UBSTAT IMAGE
        MOV BYTE PTR ES:UBCTL,AL ;RESTORE UBCTL
        POP     ES              ;RESTORE CALLER'S REGS
        POP     DS
        POP     BP
        RET     8               ;RETURN TO CALLER
RUB     ENDP
UCSEG   ENDS
        END
```

Serial I/O Subroutine Library

This library contains a Serial I/O procedure which may be accessed by linking to the file SERSUB.OBJ. Customers writing specialized DACU-resident code must provide *all* Serial I/O initialization and interrupt servicing for both devices.

SSETUP: Set up Serial I/O entry points.

Format: SSETUP(ATNFLG,SSRFLG,INTFLG)

Where: ATNFLG = 2 byte flag for Attention Sent service routine.
SSRFLG = 2 byte flag for Set Start Register service routine.
INTFLG = 2 byte flag for Interrupt service routine.

Purpose: This subroutine is used to set up the customer entry points for device interrupts, Set Start Register function calls, and attention-sent events. Once the linkage is set up, execution will return to the caller. This subroutine should only be called once, from an initialization routine. The subroutine takes three flags as parameters, indicating if there is an entry point for Attention Sent, Set Start Register, and Interrupt service routines. Using FORTRAN, three LOGICAL*2 variables, which have been set to .TRUE. or .FALSE. as desired, are passed. Using PASCAL, three pointers to three integer variables, which have been set to 1 or 0, are passed. If an entry point flag is set to TRUE (1), the customer provided entry point is used. Upon linkage, the object module containing the entry point must be specified so that the service routine can be reached. If an entry point flag is set to FALSE (0), default code is used. No code for the unused entry point need be specified when linking. Upon linkage, an Unresolved External error message will be displayed for each unused entry point. This will not affect program execution since the entry point is not actually used. If

entry points are to exist, the service routines must have the following names:

Attention Sent: SATTN

Set Start Register: SSSREG

Interrupt: SINTR

Remarks: The linkage to the supplied control code is via interrupt X'96' and is handled by the customer interface portion of the Dispatcher.

The file SERSUB.OBJ is created by assembling the file SERSUB.ASM, which contains the following source code:

```
        PAGE    ,132
        .SALL                   ;DON'T EXPAND MACROS
;
        TITLE SERSUB: Serial I/O Subroutine Library
;
        EXTRN   SINTR:FAR,SSSREG:FAR,SATTN:FAR
;MACROS
MOVE    MACRO   TO,FROM
        MOV     AX,FROM
        MOV     TO,AX
        ENDM
;
SDSEG   SEGMENT PUBLIC  'CODE'
SAVEDS DW      ?                ;SAVE AREA FOR CALLER'S DS
INTADDR DD      0               ;INTERRUPT ENT PT
SSRADDR DD      0               ;SET START REG ENT PT
ATNADDR DD      0               ;ATTENTION SENT ENT PT
SDSEG   ENDS
;
SCSEG   SEGMENT PUBLIC  'CODE'
        ASSUME  CS:SCSEG,DS:SDSEG
        PUBLIC  SSETUP
;
SSETUP PROC     FAR             ;SETUP SERIAL PORT ENT PTS
        PUSH    BP              ;SAVE CALLER'S BP
        MOV     BP,SP           ;SET FRAME BASE
        PUSH    DS              ;SAVE CALLER'S DS
        MOVE    DS,SDSEG        ;POINT TO OUR DATA SEG
        POP     AX              ;SAVE DS FROM
;                                ENTRY ROUTINE IN SAVEDS
        MOV     SAVEDS,AX
        MOV     SI,SS:[BP+6]    ;GET INTR FLAG ADDR
        MOVE    ES,SS:[BP+8]
        MOV     AX,ES:[SI]      ;GET INTERRUPT FLAG
        CMP     AL,1            ;INTERRUPT ENTRY POINT?
        JNE     SSET1           ;IF NO, DON'T CHANGE TABLE
        MOV     INTADDR,OFFSET INTENT ;IF YES MOVE OFFSET,
;                                       SEGMENT TO TABLE
        MOVE    INTADDR+2,CS
SSET1: MOV      SI,SS:[BP+10]   ;GET SET START FLAG ADDR
        MOVE    ES,SS:[BP+12]
        MOV     AX,ES:[SI]      ;GET SET START FLAG
        CMP     AL,1            ;SET START ENTRY POINT?
        JNE     SSET2           ;IF NO, DON'T CHANGE TABLE
        MOV     SSRADDR,OFFSET SSRENT ;IF YES MOVE OFFSET,
;                                       SEGMENT TO TABLE
        MOVE    SSRADDR+2,CS
SSET2: MOV      SI,SS:[BP+14]   ;GET ATTN SENT FLAG ADDR
```

```
        MOVE    ES,SS:[BP+16]
        MOV     AX,ES:[SI]      ;GET ATTENTION FLAG
        CMP     AL,1            ;ATTN SENT ENTRY POINT?
        JNE     SSET3           ;IF NO, DON'T CHANGE TABLE
        MOV     ATNADDR,OFFSET ATNENT ;IF YES MOVE OFFSET,
;                                       SEGMENT TO TABLE
        MOVE    ATNADDR+2,CS
SSET3:  LEA     DX,INTADDR      ;POINT DS:DX TO ENT PT TABLE
        INT     96H             ;TELL BASE CODE USER ENT PTS
        MOV     DS,SAVEDS       ;RETURN CALLER'S DS
        POP     BP              ;RETURN CALLER'S BP
        RET     12              ;FAR RETURN, POP 12 BYTES
SSETUP  ENDP
;
INTENT  PROC    FAR             ;INTERNAL INTERRUPT ENT PT
        MOV     DS,SAVEDS       ;RESTORE USER ROUTINE DS
        CALL    SINTR           ;CALL SERIAL INTR ENT PT
        RET
INTENT  ENDP
;
SSRENT  PROC    FAR             ;INTERNAL SET START ENT PT
        MOV     DS,SAVEDS       ;RESTORE USER ROUTINE DS
        CALL    SSSREG          ;CALL SERIAL SET STRT ENT PT
        RET
SSRENT  ENDP
;
ATNENT  PROC    FAR             ;INTERNAL ATTN SENT ENT PT
        MOV     DS,SAVEDS       ;RESTORE USER ROUTINE DS
        CALL    SATTN           ;CALL SERIAL ATN SENT ENT PT
        RET
ATNENT  ENDP
SCSEG   ENDS
        END
```

Sample FORTRAN Programs

The following code is an example of a customer initialization routine for a UNIBUS-compatible device, using the Parallel I/O and General Usage Subroutine Libraries:

```
$STORAGE:2
$LINESIZE:132
        PROGRAM FORUSR
        EXTERNAL BATENT
        INTEGER*2 SB1DA,SB2SB3
        LOGICAL*2 ATNFLG,SSRFLG,INTFLG
C THIS PROGRAM INITS ENTRY PTS & RETNS TO DACU CODE.
C     OK TO WRITE HERE. NO I/O GOING ON.
        WRITE(*,10)
 10     FORMAT(' FORTRAN USER CODE, 1/10/84')
C SET ENTRY POINT FLAGS FOR SET START REG,
C INTERRUPT, AND ATTENTION SENT
        ATNFLG=.TRUE.
        SSRFLG=.TRUE.
        INTFLG=.TRUE.
        CALL USETUP(ATNFLG,SSRFLG,INTFLG)
C ENQUEUE A BACKGROUND JOB.
        CALL BATCH(BATENT)
C RETURN TO DACU BASE CODE.
        CALL SLEEP
        END
```

The following is an example of FORTRAN service routines associated with the FORTRAN main program above:

```
$STORAGE:2
$LINESIZE:132
       SUBROUTINE USSREG
       INTEGER*2 DATA,ADDR
C************************************************
C*      SET START REGISTER SERVICE ROUTINE      *
C************************************************
C   WRITE NEGATIVE WORD COUNT
C   TO DEVICE WORD COUNT REG. X'F538'=-2760
       DATA = -2
       ADDR = -2760
       CALL WUB(ADDR,DATA)
       END
       SUBROUTINE UINTR
C************************************************
C*         INTERRUPT SERVICE ROUTINE            *
C************************************************
       INTEGER*2 SEG,OFFSET,DATA1,DATA2,COUNT
C WRITE UBCTL TO DISABLE PARALLEL I/O.
C I.E. WRITE A X'3FBE'=16318 TO X'5000:1000'=20480:4096
       COUNT = 2
       OFFSET = 4096
       SEG = 20480
       DATA1 = 16318
       CALL DWRITE(COUNT,DATA1,OFFSET,SEG)
C READ INTERRUPT VECTOR (UBINTVEC)
C I.E. READ X'5000:9002'=20480:-28670
       OFFSET = -28670
       CALL DREAD(COUNT,DATA2,OFFSET,SEG)
C READ UBSTAT UNTIL INTR BIT DROPS.
C I.E. READ X'5000:9000'=20480:-28672 UNTIL < X'80'=128
       OFFSET = -28672
 10    CALL DREAD(COUNT,DATA1,OFFSET,SEG)
       IF (DATA1.GE.128) THEN
          GOTO 10
       END IF
C SEND AN ATTENTION TO THE HOST SYSTEM.
C SENSE BYTE 1/DEVICE ADDRESS=X'9004'=-28668
C SENSE BYTE 2/SENSE BYTE 3=INT VECTOR.
 11    DATA1 = -28668
       CALL HATTN(DATA1,DATA2)
       END
       SUBROUTINE UATTN
C************************************************
C*       ATTENTION SENT SERVICE ROUTINE         *
C************************************************
       INTEGER*2 SEG,OFFSET,DATA,COUNT
C WRITE UBCTL TO ENABLE PARALLEL I/O.
C I.E. WRITE A X'3FC0'=16320 TO X'5000:1000'=20480:4096
       COUNT = 2
       OFFSET = 4096
       SEG = 20480
       DATA = 16320
       CALL DWRITE(COUNT,DATA,OFFSET,SEG)
       END
       SUBROUTINE BATENT
C************************************************
C*              BACKGROUND ROUTINE              *
C************************************************
       INTEGER*2 NUMBER
C NO I/O GOING ON. SAFE TO WRITE.
       WRITE(*,10)
```

```
10    FORMAT(' FORTRAN BACKGROUND ROUTINE REACHED.')
      WRITE(*,20)
20    FORMAT (' ')
      NUMBER = 1
      CALL LED(NUMBER)
      END
```

Sample PASCAL Programs

The following code is an example of a customer initialization routine for a UNIBUS-compatible device, using the Parallel I/O and General Usage Subroutine Libraries:

```
(*$LINESIZE:132,$DEBUG-*)
PROGRAM PASUSER (INPUT,OUTPUT);

TYPE
        INTEGPTR = ADS OF INTEGER;

PROCEDURE BATENT; EXTERN;
PROCEDURE SLEEP; EXTERN;
PROCEDURE USETUP(ATNPTR,SSRPTR,INTPTR:INTEGPTR); EXTERN;
PROCEDURE BATCH(PROCEDURE PROCNAME [PUBLIC]); EXTERN;

VAR
        ATNPTR,SSRPTR,INTPTR:INTEGPTR;
        ATNFLG,SSRFLG,INTFLG:INTEGER;

(***********************************************)
(*                MAIN  PROGRAM                *)
(***********************************************)

(* THIS PROGRAM INITIALIZES PARALLEL I/O ENTRY POINTS *)
(* AND RETURNS TO DACU BASE CODE. *)

BEGIN
        WRITELN('PASCAL USER CODE, 1/10/84');
                                        (* OK TO WRITE HERE *)
                                        (* NO I/O GOING ON. *)
(* SET UP ENTRY POINTS FOR SET START REGISTER, *)
(* INTERRUPT, AND ATTENTION SENT. SET FLAGS    *)
(* APPROPRIATELY. INDIRECT POINTERS ARE NEEDED *)
(* SO THAT PARMS LOOK THE SAME AS FORTRAN'S.   *)
        SSRFLG:=1;
        ATNFLG:=1;
        INTFLG:=1;
        ATNPTR:=ADS ATNFLG;
        SSRPTR:=ADS SSRFLG;
        INTPTR:=ADS INTFLG;
        USETUP(ATNPTR,SSRPTR,INTPTR);

(* ENQUEUE A BACKGROUND JOB. *)
        BATCH(BATENT);

(* RETURN TO DACU BASE CODE.   (SLEEP)      *)
        SLEEP;
END.
```

The following is an example of a PASCAL module containing service routines associated with the PASCAL main program above:

```
(*$LINESIZE:132,$DEBUG-*)
MODULE PASSUBS[PUBLIC];
TYPE
        WORDPTR = ADS OF WORD;

PROCEDURE DWRITE(LENPTR,DATAPTR,OFFPTR,
                 SEGPTR:WORDPTR); EXTERN;
PROCEDURE DREAD(LENPTR,BUFPTR,OFFPTR,
                 SEGPTR:WORDPTR); EXTERN;
PROCEDURE HATTN(SB1DAPTR,SB2SB3PTR:WORDPTR); EXTERN;
PROCEDURE LED(NUMPTR:WORDPTR); EXTERN;
PROCEDURE WUB(ADDRPTR,DATAPTR:WORDPTR); EXTERN;

(*********************************************)
(*      SET START REGISTER SERVICE ROUTINE      *)
(*********************************************)
PROCEDURE USSREG;

VAR ADDR,DATA : WORD;
    ADDRPTR,DATAPTR : WORDPTR;

BEGIN                           (* WRITE NEGATIVE WORD COUNT *)
        DATA := -2;             (* TO DEVICE WORD COUNT REG. *)
        ADDR := 16#F538;
        DATAPTR := ADS DATA;
        ADDRPTR := ADS ADDR;
        WUB(ADDRPTR,DATAPTR);
END;

(*********************************************)
(*          INTERRUPT SERVICE ROUTINE           *)
(*********************************************)
PROCEDURE UINTR;
CONST
        DACUIO =   16#5000;
        UBCTL  =   16#1000;
        UBSTAT =   16#9000;
        UBINTVEC = 16#9002;

VAR SEG,OFFSET,DATA1,DATA2,COUNT : WORD;
    OFFSETPTR,COUNTPTR,
    SEGPTR,DATA1PTR,DATA2PTR : WORDPTR;

BEGIN
    (* LOAD INDIRECT DATA POINTERS *)
        SEGPTR := ADS SEG;
        OFFSETPTR := ADS OFFSET;
        DATA1PTR := ADS DATA1;
        DATA2PTR := ADS DATA2;
        COUNTPTR := ADS COUNT;

    (* WRITE UBCTL TO DISABLE PARALLEL I/O. *)
        COUNT := 2;
        OFFSET := UBCTL;
        SEG := DACUIO;
        DATA1 := 16#3FBE;
        DWRITE(COUNTPTR,DATA1PTR,OFFSETPTR,SEGPTR);

    (* READ INTERRUPT VECTOR (UBINTVEC) *)
        OFFSET := UBINTVEC;
        DREAD(COUNTPTR,DATA2PTR,OFFSETPTR,SEGPTR);

    (* READ UBSTAT UNTIL INTR BIT DROPS. *)
        OFFSET := UBSTAT;
        DATA1 := 16#80;
        WHILE (DATA1  = 16#80) DO
```

```
                DREAD(COUNTPTR,DATA1PTR,OFFSETPTR,SEGPTR);

    (* SEND AN ATTENTION TO THE HOST SYSTEM.    *)
    (* SENSE BYTE 1/DEVICE ADDRESS=X'9004'.     *)
    (* SENSE BYTE 2/SENSE BYTE 3=INT VECTOR.    *)
        DATA1 := 16#9004;
        HATTN(DATA1PTR,DATA2PTR)
END;

(************************************************)
(*        ATTENTION SENT SERVICE ROUTINE         *)
(************************************************)
PROCEDURE UATTN;

CONST
        DACUIO =    16#5000;
        UBCTL  =    16#1000;

VAR SEG,OFFSET,DATA,COUNT : WORD;
    OFFSETPTR,COUNTPTR,
    SEGPTR,DATAPTR : WORDPTR;

BEGIN
    (* LOAD INDIRECT DATA POINTERS *)
        SEGPTR := ADS SEG;
        OFFSETPTR := ADS OFFSET;
        DATAPTR := ADS DATA;
        COUNTPTR := ADS COUNT;

    (* WRITE UBCTL TO ENABLE PARALLEL I/O. *)
        COUNT := 2;
        OFFSET := UBCTL;
        SEG := DACUIO;
        DATA := 16#3FC0;
        DWRITE(COUNTPTR,DATAPTR,OFFSETPTR,SEGPTR)
END;

(************************************************)
(*                BACKGROUND ROUTINE             *)
(************************************************)
PROCEDURE BATENT;

VAR NUMBER: WORD;
    NUMPTR : WORDPTR;

BEGIN                   (* NO I/O GOING ON. SAFE TO WRITE. *)
        WRITELN('PASCAL BACKGROUND ROUTINE REACHED');
        NUMBER := 1;
        NUMPTR := ADS NUMBER;
        LED(NUMPTR);
END;
END.
```

Chapter 5. Installing and Operating the DACU

Begin installation by unpacking the DACU System Unit (IBM Personal Computer). Install the System Unit on top of the DACU Interface Unit, and perform the setup and checking procedures outlined in the PC Guide to Operations manual.

Installation—Inside the System Unit

After setup and checkout of the System Unit have been completed successfully, make sure System Unit is powered off, then perform the following steps[6]:

1. Carefully remove the cover from the System Unit.
2. Install the DACU SU EXTENDER CARD in Expansion Slot J1 of the System Unit following the standard procedures for installation of any PC adapter card.
3. Locate the power supply connector, P8, plugged into P1 on the system board and carefully unplug it.
4. Plug P8 from the power supply into the mating connector of the DACU power adapter card, then plug the power adapter card into P1 on the system board.
5. Plug the two-wire cable attached to the DACU POWER ADAPTER CARD into the four-position connector on the DACU SU EXTENDER CARD.
6. Replace the System Unit cover.

This completes the installation of components inside the DACU System Unit.

The DACU Interface Unit expects certain PC resources to be available for proper operation. These resources include hardware interrupts 2, 3, 4, and 7. These interrupts are not available to any other options that may be installed in System Unit expansion slots. See "Packaging" on page 140 for a complete summary of requirements and limitations.

---

6 In general, refer to procedures for PC option installation as outlined in the PC *Guide to Operations* manual.

Installation—Interface Unit

Now, with the System Unit on top of the DACU Interface Unit, install the DACU KEYBOARD TRAY on the front of the Interface Unit (end with the Operator Panel). Use the four screws saved when the tray was removed from its shipping location. The table mounts on four holes located near the top just inside the front door.

System Unit Power

For Non-North America Installations: There is no accessible convenience power outlet for non-U.S. installations. Hence, IGNORE THIS STEP. The DACU System Unit should be plugged into a suitable external outlet.

For U.S. Installations Only: Plug the System Unit power cord into the convenience power outlet located at the lower rear of the DACU Interface Unit on the DACU POWER ASSEMBLY. (Access to this outlet is through the rear door.)

THIS OUTLET SHOULD NEVER BE USED FOR ANY OTHER PURPOSE!

DACU Memory Refresh Control Jumper

The DACU Interface Unit is shipped with the DMA/MEMORY printed circuit card configured for automatic DACU memory refresh control. (See "Chapter 8. Performance Characteristics" on page 143.) If it is desired to operate the DACU for maximum performance (adhering to the guidelines outlined in "Chapter 8. Performance Characteristics" on page 143), then the Memory Refresh Control Jumper must be moved. Use the following procedure to accomplish this:

1. Remove the DMA/MEM card from the card cage inside the DACU Interface Unit. (Refer to "Logic Card Replacement" on page 133 for card removal procedures.)

2. Locate jumper XX01 located near the upper edge of the card.

3. Position the small jumper block over the A2 and A3 pins.

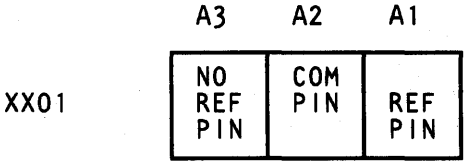


Figure 12. Refresh Control Jumper Block

4. Following the procedures outlined in "Logic Card Replacement" on page 133, replace the DMA/MEM card and continue with DACU installation.

DACU Channel Address Set-Up

The DACU Interface Unit, as shipped, is wired for logical devices addresses X'00' to X'04'. That is, it will respond to addresses X'c00' to X'c04' (where 'c' is the channel address). If it is desired to operate with a different device address range, follow these instructions:

1. Using the instructions outlined in "Logic Card Replacement" on page 133, remove the OEMI ADAPTER card.

2. Locate the jumper blocks on the card in the following format:

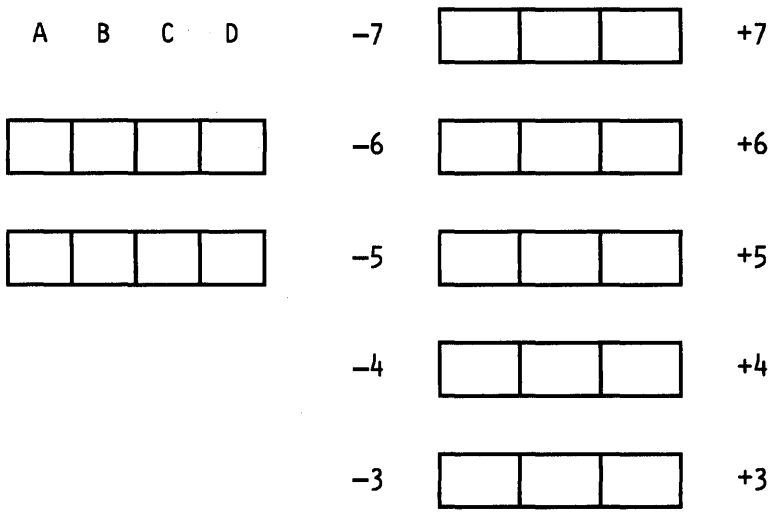


Figure 13. Address Range and Address Selection Jumper Blocks

A block of *eight* addresses is defined by the wiring of the *numbered* jumpers in the following sense: The bits in the addresses for the range, for example, X'10' to X'17' would be:

7	6	5	4	3	2	1	0
0	0	0	1	0	X	X	X

Figure 14. Address Selection Example

Hence, the jumper configuration for this example range would be: -7/-6/-5/+4/-3.

Since the DACU supports only five logical devices, it may be desired to exclude one to three addresses from the eight address range defined by the numbered jumpers. This is done by wiring the *lettered* jumpers appropriately. Installing the A/B/C/D jumpers, excludes addresses in the manner indicated below. Continuing the example above (address range X'10' to X'17'):

VALID RANGE	INSTALL JUMPER(S)		
X'10'-X'17'	none	(excludes NO	addresses)
X'10'-X'16'	A	(excludes LAST	address)
X'10'-X'15'	AB	(excludes LAST TWO	addresses)
X'10'-X'14'	ABC	(excludes LAST THREE	addresses)

3. Install jumpers as indicated above to match the desired address configuration.

4. Reinstall the OEMI ADAPTER card following the procedures outlined in "Logic Card Replacement" on page 133, and continue with DACU installation below.

Cabling and DACU Interface Unit Power

Install the supplied DACU INTERFACE CABLE between the rear-mounted connector on the DACU Interface Unit and the external connector of the DACU SU EXTENDER CARD that was previously installed in Slot J1 of the DACU System Unit.

The power cord for the DACU Interface Unit may now be connected to a suitable power outlet using the power cord provided.

The next step involves attaching the DACU Interface Unit to a block multiplex channel of the IBM host system. This should be performed by an authorized IBM Field Engineer or other qualified person.

Important Note: The host computer should be halted or otherwise disabled when the DACU is physically attached to the channel.

(If the DACU is to be at the very end of a string of control units on that channel, then channel terminator assemblies must be installed in the DACU Interface Unit at this time.) The channel cables[7] should be installed carefully, matching Channel-In and Channel-Out by color (i.e., dark grey to light grey and light grey to dark grey). Connectors for channel cables are located at the lower rear of the DACU Interface Unit (accessed through the rear door).

The connector bracket must be tilted back for ease of channel cable installation. The slotted screw holding the bracket at a 45 degree angle may be removed. (The screw should be saved for subsequent reinstallation.) The connector bracket may be raised (carefully to avoid cable damage) to a more vertical position. Cables and/or terminators may be installed while holding the bracket in the raised position. After installation is complete, the bracket should be lowered to its original (45 degree) position and the screw reinstalled.

After attachment to the channel is completed, the DACU is ready to be powered up and checked out. (The DACU will not function if the ENABLE/DISABLE switch is in the ENABLE position unless the DACU is cabled to an active channel.)

---

7 IBM Part No. 5353920 or 5466456. See *IBM System 360 and 370 I/O Interface Channel to Control Unit Original Equipment Manufacturer's Information,* GA22-6974.

Unit Control Word (UCW) specification for the DACU logical devices on the channel should be specified as UNSHARED and BLOCK MULTIPLEXER.

DACU Operator Switches and Indicators

Operator or user actions are not normally required for the DACU while it is in operation. This section describes the physical controls on the DACU Interface Unit, and it outlines operating procedures associated with the control programs that execute on the attached System Unit.

The following basic controls are provided on the DACU Interface Unit:

1. Power Off/On Switch and Indicator

2. Enable/Disable Switch and Indicator.

The power switch of the DACU System Unit should be left ON at all times. The DACU Interface Unit will provide power to the System Unit when the Interface Unit is turned on.

To disable the DACU (place it off-line), move the ENABLE/DISABLE switch to the DISABLE (down) position. If DACU power is to be turned off, it is essential to wait until the DISABLE indicator turns on, indicating that it is safe to turn power off without disturbing the host system. After the DISABLE indicator comes on, position the POWER ON/OFF switch in the OFF (down) position.

The System Unit may be used as a "stand-alone" Personal Computer while the DACU is attached to the channel, as long as the ENABLE/DISABLE switch is in the DISABLE position. However, the System Unit will see a constant RESET under certain conditions. These are:

1. DACU is detached from the channel AND the ENABLE/DISABLE switch in the ENABLE position.

2. DACU is attached to an "inactive channel" with the ENABLE/DISABLE switch in the ENABLE position.

3. DACU System Unit is detached from the DACU (cable between System Unit and Interface Unit removed) without removing the SU Extender card and Power Adapter card from the System Unit and restoring the System Unit to its original configuration.

The DACU may be powered on with the ENABLE/DISABLE switch in either position. However, there are differences in the behavior of power-on diagnostics depending on the state of the switch. With the switch in either position, the PC power-on self tests *will* be run. However, the DACU diagnostics (RASMONn) *will be bypassed if the switch is in the ENABLE position.* RASMONn will be run if the switch is in the DISABLE position at power-on time.

Installing Parallel I/O Peripheral Adapters

"Back Panel" for Peripheral Adapters

The DACU provides one connector equivalent to a Digital Equipment Corporation DD11-B connector block. An outline of the connector is illustrated below.

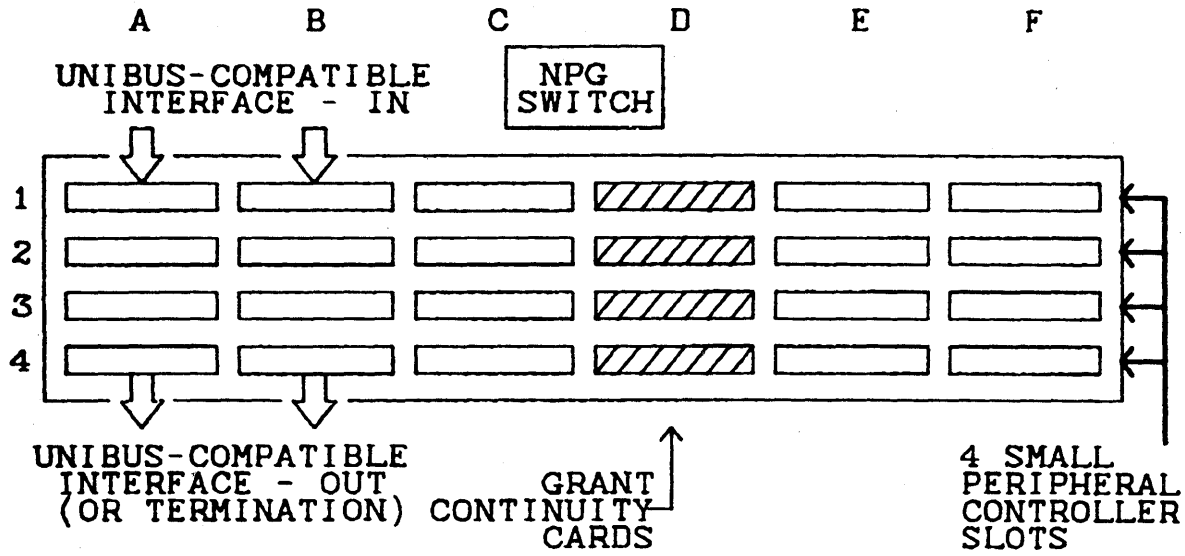


Figure 15. DD-11 Block

The connector is essentially a 4x6 array of connection elements. Two of the positions (indicated above) are required for entrance of the DMA/I/O signals. Two others (also indicated) are required for either termination or continuation of the Parallel I/O interconnection. (In the case of continuation, those two connection elements would be connected via a cable to another DD11 connector — both customer-provided. In the case of termination, a customer provided terminator element[8] must be installed. In addition, the connector must have a terminator installed for proper operation of the DACU diagnostics.)

The remaining connection elements are available for connecting Parallel I/O devices. These remaining elements comprise a maximum of two "hex" slots (six connection elements) and two "quad" slots (four connection elements). Each of these four "Small Peripheral Controller" slots may be used for connection of an I/O device.

Various I/O devices provide the attachment either through the simple Parallel I/O Interface cable connector that plugs into the Parallel I/O Interface slot or through an adapter card containing active circuits (Small Peripheral Controller) that plugs into a "hex" or "quad" slot. The DACU provides the following power for such adapter cards:

---

8 The terminator element must be equivalent a Digital Equipment Corporation Model M930 UNIBUS Terminator Card.

```
+ 5 V  - 12.0 A max to 32.2 deg C (90 deg F) max ambient
         15.0 A max to 22.2 deg C (72 deg F) max ambient
          3.0 A min loading required to guarantee
            1 per cent regulation
+15 V  -  1.0 A max
-15 V  -  1.0 A max
```

Bus Grant Continuity

Any slots not containing a Small Peripheral Controller card must be occupied by a Bus Grant Continuity Card. Four Bus Grant Continuity Cards are provided with the DACU. It will be necessary to remove one Bus Grant Continuity Card from each position to be occupied by a Small Peripheral Controller Card.

Non-Processor Grant

The Parallel I/O Non-Processor Grant (NPG) line has been implemented with a four position DIP switch. The four positions on this switch correspond to the four NPG connections on the DD11-B. The switch is located on the connector mounting bracket and is best accessed with the connector raised to the service position. Looking at the switch in this orientation, the positions are numbered 1 through 4 from the top to bottom. The topmost position (1) corresponds to the leftmost connector slot, and so on. When a switch is in the ON position, continuity of NPG is carried through to the next slot position. To intercept NPG, the switch for the selected socket must be positioned to OFF.

The DD11-B connector mounting bracket has three positions:

1. HOME — Horizontal, locked in position.
2. SERVICE — Vertical, locked in position.
3. MAINTENANCE — Horizontal, outside enclosure.

The HOME position is the normal operating position. To raise the block to the SERVICE position, locate the quarter turn screw at the front of the mounting bracket and release. The entire connector assembly may now be raised and locked against the bracket located near the top rear of the enclosure with the locking mechanism. It may be necessary, depending on the height of the adapter cards installed, to temporarily remove the cards while the bracket is being raised. The procedure for placing this connector into the MAINTENANCE position is identical to placing it in the SERVICE position except for the necessary step of moving the upper support bracket. This is done by loosening the two screws holding the bracket, moving the bracket upward, and tightening the screws to hold it in this position. The connector (with all cards removed) is rotated to the fully horizontal position (outside the enclosure). This position exposes all wiring on the underside of the connector. Precautions should be taken to protect this wiring and the cabling that is still plugged into the top of the connector.

Once any work is completed on the wiring side, the connector should be put in the HOME position and the bracket repositioned to its normal position.

Installation of Parallel I/O Peripheral Adapters

Parallel I/O peripheral adapter cards or Parallel I/O Interface cables must be inserted into an empty slot in the DD11-B connector provided. Interface cables must be installed in the D1 slot directly opposite the internal DACU cable plugged into the A1 slot. Adaptor cards may be either 2, 4, or 6 connector groups wide and will plug into either one of the four 4-wide or one of the two 6-wide peripheral slots.

Caution must be exercised when plugging adapter cards or cables to assure proper orientation since the connector design does not absolutely prevent incorrect insertion. As a guide to correct insertion orientation, note that a card has a notch between each tab section. This notch generally has the same shape as the connector and can be used to determine the proper insertion orientation. Moreover, the component side of adapter cards will normally face to the left while looking into the DACU from the rear.

The following steps are suggested when installing a card or cable:

1. Make sure the DACU Interface Unit and DACU System Unit are both powered down.
2. Locate the peripheral slot where the card is to be plugged.
3. Remove the small Bus Grant Continuity card from that slot if it has not been previously remove.
4. Prepare to insert the card by checking its orientation using the tests outlined above (notch position and component orientation).
5. Plug in the card being careful not to apply so much pressure that the card will be broken. Also, if there are already other cards installed in the DD11-B, be careful not to damage components on those cards during the insertion process.
6. Check that any of the DD11-B slots that are unoccupied by peripheral adapter cards have small Bus Grant Continuity cards installed, as outlined above.
7. Intercept NPG as necessary. See "Non-Processor Grant" on page 97.
8. If a wrap card is installed in the DD11 block, it should be removed and stored at this time.

Wiring Arrangements for Serial Device Cables

When connecting a Serial I/O device to the DACU, it is often necessary to obtain cables or wire cables different from those provided with the serial devices themselves.

In general, if the serial device acts like Data Communication Equipment (e.g., modem), it should have a *female* 25-pin EIA plug. If the device acts like Data Terminal Equipment (e.g., terminal), it should have a *male* 25-pin EIA plug. The

DACU behaves as Data Terminal Equipment. Therefore, the serial connectors for the DACU end are male 25-pin EIA plugs.

If the serial device is Data Terminal Equipment, a female-to-female cable is needed. The wiring of two possible such cables are described below. One is called a "null modem" cable since it is often used to connect two terminals directly together with no modems between. The other is a "flip" cable, which is used to turn the behavior of the DACU from a terminal to a modem.

In either case, the signal names and pin assignments are:

```
PIN       SIGNAL NAME
--- -------------------------
  1  FG - Frame Ground
  2  TD - Transmit Data
  3  RD - Receive Data
  4 RTS - Request To Send
  5 CTS - Clear To Send
  6 DSR - Data Set Ready
  7  SG - Signal Ground
  8 DCD - Data Carrier Detect
 20 DTR - Data Terminal Ready
 22  RI - Ring Indicator
```

Neither of these cables is provided with the DACU. However, it is necessary to have a null modem cable for use with problem determination procedures associated with the Serial I/O DACU function.

Null Modem Cable

A null modem cable connects together each Request-to-Send (RTS) line with its Clear-to-Send (CTS) line and interchanges the Data-Set-Ready (DSR) and Data-Terminal-Ready (DTR) lines as well as the Receive Data (RD) and Transmit Data (TD) lines. It is made from two female, 25 pin EIA connectors wired as follows:

```
END 1                      END 2
-----                      -----
 FG    1--------------1     FG
 TD    2--------------3     RD
 RD    3--------------2     TD
RTS    4--------------8     DCD
       |
CTS    5
DSR    6-------------20     DTR
 SG    7--------------7     SG
DCD    8--------------4     RTS
                      |
                      5     CTS
DTR   20--------------6     DSR
```

Flip Cable

The Flip Cable merely interchanges three pairs of lines.

```
END 1                      END 2
-----                      -----
 FG   1--------------1      FG
 TD   2--------------3      RD
 RD   3--------------2      TD
RTS   4--------------5      CTS
RTS   4--------------8      DCD
CTS   5--------------4      RTS
DSR   6-------------20      DTR
 SG   7--------------7      SG
DCD   8--------------4      RTS
DTR  20--------------6      DSR
```

Installation of Control Programming

Preparing control programming for operation is a simple matter of merging files on the diskette provided with the DACU onto a DOS diskette.

1. Transfer DOS to a formatted diskette as outlined in the IBM PC Disk Operating System manual.

   FORMAT [d:]/S

2. With the diskette just prepared with DOS in the "A" diskette drive and the DACU diskette in the "B" diskette drive, execute:

   COPY B:*.COM

   Even if the system unit has only one drive, enter the same command and a system will prompt when the DACU diskette should be placed in the drive. This command will initiate transfer the DACU control programming files to the DOS diskette.

   DACUnn.COM
   RASMONn.COM

3. Back up the newly configured diskette.

The Supplied DACU Diskette

The diskette supplied with the DACU has the following files:

```
FILE NAME       DESCRIPTION

RASMONn  COM    DACU diagnostics
DACUnn   COM    DACU functional code
DL       COM    Error message log display
UBSUB    ASM    Parallel I/O subroutine library source code
UBSUB    OBJ    Parallel I/O subroutine library object code
SERSUB   ASM    Serial I/O subroutine library source code
SERSUB   OBJ    Serial I/O subroutine library object code
GENSUB   ASM    General usage subroutine library source code
GENSUB   OBJ    General usage subroutine library object code
UBUTIL   LIB    Parallel I/O utility library
SERUTIL  LIB    Serial I/O utility library
GENMAC   LIB    General usage macro library
HEXTOASC TAB    Hex to ASCII conversion table
HALLS    ASM    Host Application Language Library using SIO
HALLGE   ASM    Host Application Language Library using GAM
                and EXCP
```

Note: The HALL code is the code in Appendix A, "Sample Host Programming" on page 147.

DACU Initialization

A PC AUTOEXEC.BAT file containing two or three records may be used to start the DACU at power-on or system reset time. The first record invokes the bring-up diagnostics, the second invokes the DACU functional code, and the third (optional) can invoke user set-up code.

An easy way to create an AUTOEXEC.BAT file is as follows:

Place the diskette which is to contain the AUTOEXEC.BAT file in the diskette drive and enter the following:

```
COPY CON AUTOEXEC.BAT
```

Then type the lines of the AUTOEXEC.BAT ending each line by pressing the ENTER key. When the last line has been entered, press the F6 key, and the new AUTOEXEC.BAT file will be stored.

The first record has the format

```
RASMONn XXXX
```

This record invokes the bring-up diagnostics which perform a checkout of the DACU. If an error is found, the diagnostic program will display a message and wait for operator intervention before continuing.

The second record invokes the DACU functional code. It has the following format:

```
DACUnn uuoor
```

The value for "nn" is set by the manufacturer. The value for "uu" is the starting channel unit address in hexadecimal. This must match the address configuration set up as described in "DACU Channel Address Set-Up" on page 93.

The value of "oo" determines some of the run-time options for the functional code. The value is a hexadecimal representation of a bit mask where each of the bits is described below.

- Bit 7 (most significant) - Not used.
- Bit 6 - Not used.
- Bit 5 - If on, warning messages will be logged to diskette when they occur.
- Bit 4 - If off, an in-memory trace table is kept. This table contains a pointer to the last 200 messages (informational and warning) that have been issued by the functional code. If Bit 4 is on, no trace table is kept. This bit should be turned on, except when debugging, since logging of the messages uses a significant amount of the processing resource.
- Bit 3 - If on, execution will stop after displaying a message. Execution will resume when the operator presses any key.
- Bit 2 - If on, the functional code will terminate execution and return to DOS after a warning message.
- Bit 1 - If on, display of informational messages will be suppressed, but messages will logged on diskette (if logging enabled). This bit should always be on.
- Bit 0 - If on, display of warning messages will be suppressed. If off, warning messages are displayed and an audible alarm is sounded.

The value for "r" may be either "R" or blank. If the user includes a third (optional) record in the AUTOEXEC.BAT file to invoke user code setup, then this field *must* be an "R." If a blank is in this field, the third record should not be included in the AUTOEXEC.BAT file.

Following is an example of an AUTOEXEC.BAT file. In this example, the DACU is attached to the channel at base address x20, it is desired to keep an internal trace of messages, and customer supplied local code (called USER) is to be invoked.

```
RASMONn X
DACUnn 2002R
USER
```

DACU Keyboard Controls

Operator controls available through the keyboard of the DACU System Unit controlling the DACU are described below.

At any time that the System Unit is not servicing interrupts, the DACU functional code polls the keyboard for single keystrokes. The keystrokes are interpreted as follows:

Space Bar =
Return to DOS and leave memory. (If this key is pressed after customer supplied local code is loaded when using DOS 2.0, the PC will halt and have

to be re-booted. If pressed after local code is loaded when using DOS 1.1, only the local code leaves memory.)

0,1,2,4 =
Send attention to host for device 0, 1, 2, or 4, respectively.

5,6,7,9 =
Send Device End for device 0, 1, 2, or 4, respectively.

R = Return to DOS, but remain resident (same as R-option when starting DACU).

I = Suppress informational messages (also controlled by input parameters).

W = Suppress warning messages. (also controlled by input parameters).

D = Display all messages. (also controlled by input parameters).

P = Display trace log (if tracing is active).

Chapter 6. Customer Problem Analysis and Resolution

The customer is responsible for installation and maintenance of the DACU. Support will be provided in the form of initialization and extended diagnostic programs and a field replacement strategy for failed replaceable units.

The power-on initialization tests (bring-up tests) are invoked by the AUTOEXEC.BAT file by including a parameter with the call for the diagnostic (e.g., "RASMONn XXXX"). Any test, including the extended tests, can be manually executed by invoking the diagnostics with no parameters RASMONn") from the keyboard and making the necessary responses to the prompts from RASMONn.

Diagnostic self-test code of two types will be provided with the DACU — power-on initialization tests and extended tests for problem isolation. The intent of the initialization tests is to determine if the DACU is capable of operating correctly. If an initialization test detects an error, the operator will be notified of the error. The operator may be able to either continue the initialization sequence or abort the initialization in order to execute the extended tests. The extended tests will allow the operator to determine if the DACU is failing or if an I/O device attached to the DACU has failed. These tests are organized into logical sections such that the failure can be isolated to a replaceable unit. The extended tests may require the operator to detach cables or install wrap cables to complete the tests. See "Diagnostic Procedures" on page 117 and "Problem Determination Procedures" on page 107 for a complete description of the diagnostics and problem determination procedures.

DACU Messages and Error Logging

There are two types of messages that may be displayed by the DACU control programs. As described in "DACU Initialization" on page 101, display of messages is controlled by an input parameter to the DACU code. But, regardless of the input options, all message codes are displayed (until overwritten) on a four-digit indicator located inside front of the DACU Interface Unit on one of the printed circuit boards. The message types are:

1. WARNINGS—If presented by the functional code, these messages indicate conditions from which the code and hardware can continue, but which may indicate that data is lost or that the machine may not operate properly in the future. Error messages from the diagnostics have this severity.

   If logging is active, then the four digit error code in each warning is written into a file on the diskette. The message numbers in the log can be displayed by using the program DL. This program will display the message numbers in a

LIFO (last in, first out) order. DL will display all messages in the log. The log stores a total of 85 messages, and included with each message number is a time stamp when the message was logged from the time-of-day clock.

2. INFORMATION—Messages issued by control program components that serve as progress indicators, etc. In normal operation, these may be ignored by the operator or user.

All messages issued by the DACU code, with the exception of the normal prompting messages, will have the following format:

MESSAGE ID	MESSAGE TEXT

The message ID is eight characters long and has the following format:

SCCCNNNN

S

is a one character severity indicator with the value "W" (Warning) or "I" (Information).

CCC

is a three digit identifier of the component that originated this message.

NNNN

is a four digit message number. The normal DOS message format follows immediately behind the message ID separated by a space.

The following list shows the message number range and the three-digit code for each component:

Message Header Conventions:

Component	ID	Number
C.U. Driver	CUD	0000-0499
Serial Driver 1	SER	1000-1499
Serial Diagnostic	SER	1500-1999
Serial Driver 2	SER	2000-2499
RAS Monitor	MON	3000-3499 3500-3999
Parallel I/O Driv.	UNI	4000-4499
Parallel I/O Diag.	UNI	4500-4999
OEMI Driver	OEM	6000-6499
OEMI Diagnostic	OEM	6500-6999
Batch Component	BAT	7900-7999

Component	ID	Number
BUS Diagnostic	BUS	9000-9099
Timer/DMA Diag.	TDM	9100-9199
Memory Diagnostic	MEM	9200-9299
Dispatcher	DSP	9900-9999

See Appendix C, "DACU Function Code Messages" on page 195 and Appendix D, "Diagnostic Monitor Messages" on page 225 for a list of messages organized by message ID.

Problem Determination Procedures

Problem determination consists of a set of procedures to help resolve operational or system problems. This guide will direct the operator to make a series of observations and provide a YES or NO answer to a question.

Part of problem determination may include loading the DACU diskette and running the diagnostics included there.

If a diagnostic routine requires some type of manual intervention (e.g., plugging a special cable or card), then this information will be provided as the diagnostic routine executes.

If the problem can be traced to the System Unit, then the problem determination procedures outlined in the PC *Guide to Operations* manual should be used. In this case, it may be necessary to completely disconnect the System Unit from the Interface Unit to complete the problem analysis. (Refer to "Replacement of DACU Unique Components in DACU System Unit" on page 131.)

Attached I/O devices should be disconnected from the DACU to aid in problem determination. If failing symptoms disappear after a device has been disconnected, then that device should be suspected. It should be tested and serviced if necessary.

When removing attached devices be sure that the NPR switch on the Parallel I/O Interface has been correctly set and that the bus continuity cards and bus terminator card have been properly inserted.

Section "Packaging" on page 140 contains a description of each card. This may be of value to the user in determining the failure cause.

Customer Replaceable Units

If the diagnostics point to the Interface Unit as the most likely cause for the problem, then the procedures in this section will assist in determining which of the DACU Customer Replaceable Units (CRU) is at fault. These units are:

Card Cage*
IU adapter card assembly
DMA/MEM card assembly
OEMI adapter card assembly
Adapter card assembly for Parallel I/O Interface.
Power Assembly*

* Units replaceable as entire subassemblies.

See "Replacement of DACU Components in Interface Unit" on page 132 for CRU replacement procedures.

Diagnostics

There are three levels of diagnostics:

1. Standard PC Power On Self Tests (POST). These are run every time the DACU is powered on.

2. DACU Power On. These are run every time the DACU is powered on with the 'Enable/Disable' switch in the 'Disable' position.

3. DACU Extended. These are run only if the 'M' option is selected.

The following notes apply to the above:

1. Items 2 and 3 assume a DACU system diskette is installed in diskette drive A.

2. See Diagnostic Procedures (Section "Diagnostic Procedures" on page 117) for a detailed description of diagnostic operation, and a flowchart to assist in problem determination.

3. The (n) in RASMONn refers to the release level of the diagnostic package shipped with the DACU Interface Unit.

4. The PC Extended Diagnostic package may be helpful for problems that have been isolated to the System Unit. This package is completely separate from any diagnostics supplied with the DACU and should not be confused with the Extended Diagnostics referred to earlier.

Service Information

Once it has been determined that a DACU Interface Unit component is defective, see "Replacement of DACU Components in Interface Unit" on page 132 for replacement procedures. The defective component(s) should be packed for shipment and returned to the manufacturer. All correspondence should refer to the unit serial number, located on the label inside the front door.

```
IT IS THE CUSTOMER'S RESPONSIBILITY TO
ASSURE THAT ALL UNITS RETURNED (UNDER
WARRANTY OR OTHERWISE) ARE SENT TO:

        REPAIR SERVICE CENTER
           IBM DACU Depot
         1 Industry Street
        Poughkeepsie, NY 12603

RETURNS MUST BE PACKAGED ADEQUATELY FOR
SHIPMENT AND SHIPPED F.O.B. DESTINATION.
```

Special Tools

The following tools are required to service the DACU:

1. Medium sized flat blade screwdriver
2. Medium sized screw starter (optional)
3. Multimeter (Similar to Triplett Model 310, manufactured by Triplett Corp. Buffton, OH 45817.)
4. 'Null' Modem Cable
   a. Similar to INMAC 270NB Null Modem cable.
   b. It connects together each Request-to-Send (RTS) line with its Clear-to-Send (CTS) line and interchanges the Data-Set-Ready (DSR) and Data-Terminal-Ready (DTR) lines as well as the Receive Data (RD) and Transmit Data (TD).
5. Wrap Card for the Parallel I/O Interface (Part No. 0174955)

   A Wrap Card was shipped with the DACU Interface Unit, installed in the DD11 connector block. It should have been removed and stored. This card contains information that can assist in the proper installation of devices into the DD11 Peripheral Device Connector.

PDP Entry Point

This is the entry point for all Problem Determination Procedures (PDPs).

Error codes (indicated on the System Unit monitor or the DACU Interface Unit diagnostic indicators, or an audio response during the System Unit's POST) may result from a problem related to the System Unit, the DACU Interface Unit, or an unidentified problem.

If a problem occurs during the personal computer power on self test (post) you should refer to the IBM PC Guide to Operation's manual or Advanced Diagnostics.

If the problem is with the DACU but is unidentified or intermittent then the "Mechanical Section" below or "Power Section" on page 113 of this manual should be referenced.

When there is an error message on the PC display or the DACU diagnostic indicators, the Device Related Problems section, "Device Related Problem" on page 116 should be referenced.

A complete description of the DACU diagnostics will be found following this section along with a flowchart, Figure 20 on page 126 to assist in problem determination.

Mechanical Section

This section should be used when there is an unidentified operational problem where the system diagnostics have been unable to determine the failing element or when there is an intermittent problem.

Check to determine whether the DACU configuration meets the following requirements:

1. The configuration includes an IBM Personal Computer System Unit with at least one diskette drive or an IBM Personal Computer XT. The DACU SU extender card must be used in the System Unit. Do not use in an expansion unit.

2. The DACU SU Extender and Power Adapter cards are installed in the System Unit.

3. None of the following are installed in the System Unit:

   a. More than 320k of memory (total of system board and memory options).

   b. An asynchronous communications card that is not *totally* compatible with the IBM Asynchronous Communications Adapter. (The IBM Asynchronous Communications Adapter has its interrupt disabled during DACU operation by the operating system. It is possible to leave an incompatible card installed during DACU operation so long as no attempt is made to program the interrupt. An IBM Asynchronous Communications Adapter will not function in the interrupt mode as long as the DACU is connected. Polled mode (programmed I/O) will function normally.)

   c. An option that uses hardware interrupts 2,3,4 or 7, unless no attempt is made to run these devices concurrently with DACU operations.

4. The following cards are installed in the DACU Interface Unit:

   a. IU Adapter card

   b. DMA/MEM card

   c. OEMI card

   d. Parallel I/O Interface card.

Does the configuration agree with the above?

NO
Take appropriate action to make configuration agree.

YES
Continue to next question.

Is the SU/IU cable between the System Unit and DACU Interface Unit installed and secure?

NO
Install the SU/IU cable between the System Unit (SU Extender Card) and the connector located on the rear of the DACU.

YES
Continue to next question.

Is the DACU attached to an *active Block-Multiplexer* channel on the host computer?

NO
Attach the DACU to an active 'Block-Multiplexer' channel.

YES
Continue to next question.

If the DACU is the last device on the channel cables, are the proper channel terminators installed in J4 and J5 of the tailgate in the DACU?

NO
Install the correct terminators. Refer to *IBM System 360 and System/370 I/O Interface to Control Unit Original Equipment Manufactures Information,* GA22-6974.

YES
Continue to next question.

Assure that the DACU 'Enable/Disable' switch is in the 'Disable' position (down) for the following steps:

Is the 'Disable' indicator on the DACU operator's panel on?

YES
Continue to next question.

NO
Log this condition and continue below.

1. Position the DACU power switch to the 'off' (down) position.

2. Remove the DACU power cord from the power outlet.

3. Open the rear door of the DACU Interface Unit. (Unplug the SU/IU cable to make this easier, if desired, or rotate the System Unit.)

4. Locate the card cage and check that all cables are securely plugged as follows:

Socket	Conn.	Qty
IU adapter	J4 to J6	(3)
DMA/MEM	J4 to J5	(2)
OEMI	J4 to J11	(8)
OEMI	J18	(1)
Parallel I/O Interface Adapter	J4 to J6	(3)
Motherboard	J7	(1)
All cables to the tailgate #	J2 to J7	(*)
Power Assembly	J1 to J5	(5)

The tailgate contains the channel cables and may contain the Serial I/O and/or other cables.

* The number of cables to the tailgate depends on the configuration.

Are all cables correctly plugged and secure?

NO
Re-plug cables correctly. (See "Logic Card Cage Replacement" on page 132 for assistance.)

YES
Continue to next question.

1. Close the rear door of the DACU Interface Unit.

2. Reinstall the SU/IU cable (if necessary).

3. Check to see that all cabling to the back of the DACU System Unit is properly installed and secure.

4. Open the front door of the DACU and locate the card cage. Assure that all cards are securely seated. If there is any possibility that cards are loose, remove the front card retention bracket, pull the cards out a small amount, then reseat them. *Be careful not to pull the cards out far enough to disturb any cabling. Be sure that the front card retaining bracket is securely reinstalled after this operation.*

5. Check the operator panel for loose or broken wiring.

6. Assure that all adapters and cables plugged in the DD11 connector are securely seated and that cards do not touch each other. Also, assure that any cables plugged into the adapter cards are secure.

Are all cards and/or cables correctly installed and securely seated?

NO
Reseat any cards or re-plug any cables that are incorrectly installed.

YES
Continue to next section.

Power Section

This section should be used when there is an unidentified operational problem where the system diagnostics have been unable to determine the failing element or when there is an intermittent problem.

It is assumed that there is no problem with the DACU Interface Unit power cord or the power outlet in which it is plugged.

Assure that the 'Enable/Disable' switch is in the 'Disable' position. Place the DACU Interface Unit power switch in the 'off' (down) position.

Steps 1 through 5 are common to all DACU power checks.

1. Unplug the SU/IU cable from the connector on the rear of the DACU.

2. Open the rear door and locate the DD11 connector located to the right of the center partition.

3. Remove any non-IBM devices from the DD11 connector. Do not remove the parallel I/O interface terminator card plugged in locations A4-B4. If a parallel I/O interface terminator card is not installed in those locations, install one at this time.

4. Locate the Power Assembly located all the way to the right of the enclosure.

5. FOR UNITS SHIPPED IN NORTH AMERICA ONLY: Locate J4 from the figure below and unplug the System Unit.

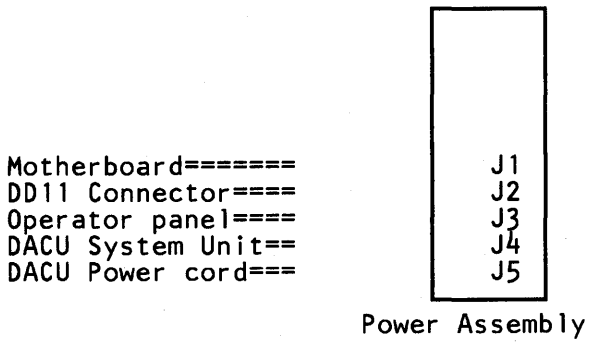


Figure 16. Power Assembly Connectors

* For DACU's shipped in North America only.

DACU Logic Power:

1. Unplug the cable from J1.
2. Position the DACU power switch to the 'ON' (up) position.

DO NOT TRY TO ENABLE THE DACU AT THIS TIME.

3. Measure the voltage on J1 using a voltmeter set to 15V DC min. Place the meter leads between the following sets of pins, observing the polarity shown. The voltage should be between the limits given.

+ Lead(Red)	- Lead(Black)	Voltage
1	2	4.85 - 5.25
3	4	4.85 - 5.25
5	6	4.85 - 5.25
7	8	4.85 - 5.25
9	8	10.8 - 13.2
8	10	10.8 - 13.2

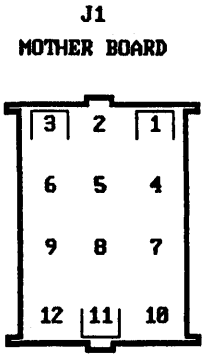


Figure 17. D.C. Power Connector J1

Notice that connector J1 has flats on pins 1, 3 and 11. This information can be used to provide orientation when probing.

Are all readings between the limits listed?

NO
Replace the Power Assembly. Follow instructions given in "Power Assembly Replacement" on page 135 of this document.

YES
Position the DACU power switch to the OFF (down) position, replug the cable into J1, and continue below.

Auxiliary Power (DD11 Connector):

4. Unplug the cable from J2.
5. Position the DACU power switch to the ON (up) position.

DO NOT TRY TO ENABLE THE DACU AT THIS TIME.

6. Measure the voltage on J2 using a voltmeter set to 20V DC min. Place the meter leads between the following sets of pins, observing the polarity shown. The voltage should be between the limits given.

+ Lead(Red)	- Lead(Black)	Voltage
1	2	4.85 - 5.25
3	4	4.85 - 5.25
5	4	13.5 - 16.5
4	6	13.5 - 16.5

Notice that connector J2 has flats on pins 1, 3 and 8. This information can be used to provide orientation when probing.

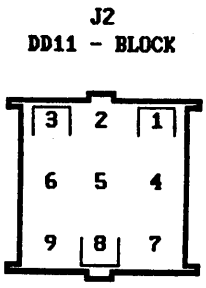


Figure 18. D.C. Power Connector J2

Are all readings between the limits listed?

NO

Replace the Power Assembly. Follow instructions given in "Power Assembly Replacement" on page 135 of this document.

YES

Position the DACU Interface Unit power switch to the OFF (down) position, re-plug the cable into J2, and continue below.

Device Related Problem

Run the diagnostics as outlined in section "Diagnostic Procedures" on page 117. Any deviation to the message sequence given there should be recorded.

The action necessary at this point depends on the failing diagnostic(s) and other indications—that is, the symptoms of the operational problems.

Use the failing diagnostics as an entry point into the chart below. Replace the card with the highest probability for causing the problems encountered.

Refer to Unit Replacement (See "Replacement of DACU Components in Interface Unit" on page 132) for assistance in card replacement.

Adapter Card	Socket	Failing Diagnostic(s)					
		Bus Test	TM/DMA	MEM	SER I/O	PAR'L ADAPT	CHANL ATTCH
SU EXT	PC	<2>	<3>	<3>	<3>	<3>	<4>
IU ADPT	J1	<1>	<2>	<2>	<2>	<2>	<3>
DMA/MEM	J2	<3>	<1>	<1>	<1>	<4>	<2>
OEMI	J4						<1>
PARALLEL	J6					<1>	

Figure 19. Card Replacement Priority Matrix

Where <n> indicates the recommended order of card replacement — the lower the number <n>, the higher the probability of that card causing the problem.

Has the problem disappeared?

NO

Replace the card with the next highest probability of failure shown on the chart. Continue this until the symptoms disappear or until all cards indicated have been replaced. (An alternative to this is to replace the entire card cage.)

YES

Rerun all diagnostics before attempting to place the DACU 'ON LINE'.

If these procedures have been followed and there is still an unresolved problem, technical assistance should be sought by calling the 7170 support telephone number.

Diagnostic Procedures

RASMONn diagnostics are invoked from the DACU System Unit keyboard (anytime the DACU is 'disabled') by entering one of the following after the DOS prompt:

```
A>RASMONn x
```

or

```
A>RASMONn
```

The 'n' indicates the level of the DACU diagnostics.

Pressing CTRL-BREAK at any time will cause a return to DOS.

The 'x' parameter forces *unattended* diagnostics. Leaving the 'x' off causes the diagnostics to run in *'manual entry mode'*.

To run the tests unattended, place the DACU Interface Unit Enable/Disable switch to the disable position and enter the following after the DOS prompt:

```
A>RASMONn x
```

The following display will appear and will scroll off the screen unless CTRL-NUMLOCK is pressed to stop the scrolling (the diagnostics will stop as well):

```
IBUS9051 - STARTING BUS TEST
IBUS9052 - END OF BUS TEST
ITDM9100 - STARTING TIMER AND DMA TESTS
ITDM9107 - INITIALIZATION OF TIMER AND DMA CHIPS COMPLETE
ITDM9111 - TIMER TEST PASSED
ITDM9112 - DMA TEST PASSED
ITDM9112 - TIMER AND DMA TESTS COMPLETE
IMEM9200 STARTING RAM TEST; ADDRESSES AND DATA RIPPLE...

IMEM9205 DATA BUS RIPPLE TEST PASSED
IMEM9203 DATA COMPARES COMPLETED; RAM PASSED
IUNI4500 - PARALLEL I/O INTERFACE HAS BEEN INITIALIZED
IUNI4506 - NOW STARTING IRQ4 TEST
IUNI4502 - IRQ4 TEST PASSED
IUNI4510 - END OF IRQ4 TEST
IUNI4538 - START OF BUS REQ/BUS GRANT TEST
IUNI4500 - PARALLEL I/O INTERFACE HAS BEEN INITIALIZED
IUNI4500 - PARALLEL I/O INTERFACE HAS BEEN INITIALIZED
IUNI4527 - BRBG TEST PASSED
IUNI4539 - END OF BUS REQUEST BUS GRANT TEST
IUNI4507 - NOW STARTING DATA BUFFER TEST
IUNI4500 - PARALLEL I/O INTERFACE HAS BEEN INITIALIZED
IUNI4509 - THE DATA CHECKED OUT
IUNI4511 - END OF DATA BUFFER TEST
IUNI4500 - PARALLEL I/O INTERFACE HAS BEEN INITIALIZED
IUNI4512 - START OF ADDRESS BUFFER TEST
IUNI4515 - ADDRESS BUFFERS ARE OK
IUNI4513 - END OF ADDRESS BUFFER TEST
IOEM6500 OEMI BRING-UP DIAGNOSTICS ENTERED

IOEM6501 OEMI TEST MODE 0 PASSED

IOEM6502 OEMI TEST MODE 1 PASSED
```

```
IOEM6515 INTERRUPT RECEIVED
IOEM6503 OEMI TEST MODE 2 PASSED

IOEM6511 OEMI DATA AND STATUS REGISTER WRAP TEST PASSED
```

This is the end of the 'Unattended diagnostics'.

To run the tests requiring manual intervention place the DACU Enable/Disable switch to 'disable' then enter the following after the DOS prompt:

```
A>RASMONn
```

There are several options which control either the way the diagnostics are run or how they are presented. These may be responded to with just an ENTER to accept a default or with one of the letters indicated (in upper or lower case) to accept the option.

The display starts with this message:

```
IMON3011 - DACU TESTS, LEVEL 1.1
```

PROMPT 1:

```
IMON3000 - I=SUPPRESS INFO. MSGS,W=SUPPRESS WARNINGS,B=BOTH
```

Enter <I> - To inhibit the informational messages associated with the diagnostics.

Enter <W> - To inhibit the warning messages associated with the diagnostics.

Enter <B> - To inhibit all messages.

Default - Pressing <enter> causes all messages, warning and informational, to be displayed.

PROMPT 2:

```
IMON3005 - ENTER S TO STOP ON WARNING MESSAGES
```

Pressing <S> here causes the diagnostics to pause after any warning message. This option is active even if messages have been suppressed.

Default - Pressing <enter>, causes diagnostics to continue to run even if there is an error.

PROMPT 3:

```
IMON3006 - ENTER M TO RUN MANUAL INTERVENTION/EXTENDED TESTS
```

The EXTENDED tests should be run if there is a problem which is not detectable using the normal diagnostics. After invoking the EXTENDED tests, directions may be given at various points to perform some type of manual intervention. Looping on tests where some operator intervention is required (e.g., installing a wrap cable) will require the operator to respond to the 'PRESS ANY KEY WHEN READY' prompt each time thru the loop.

Default - Pressing <enter> will cause only the unattended diagnostics to be run.

PROMPT 4:

```
IMON3006 - ENTER LOOP COUNT, 1-99 (DEFAULT=1, INFINITE=I)
```

This prompt asks for the number of times that the diagnostics are to be repeated. One or all of the diagnostic(s) may be repeated. Running the diagnostic(s) in a loop may assist in pinpointing problems which appear to be intermittent. To exit the loop it is necessary to press <control break>.

Default - Pressing <enter> will cause the diagnostic to be run only once.

Following the prompts, this MENU will be displayed:

```
IMON3008
 -DACU TESTS
1 BUS TEST
2 TIMER/DMA
3 MEMORY
4 SERIAL ADAPTER
5 PARALLEL INTERFACE ADAPTER
6 CHANNEL ATTACHMENT
Z ALL TESTS ABOVE
ENTER ONE OR MORE SELECTIONS
```

The examples given below show exactly what should be expected when the diagnostics are run without problems. Examples of selecting the default to the 3rd prompt (MANUAL INTERVENTION), and entering <m> are both given with the default results coming first.

1 BUS TEST: The following display examples assume <1> was selected from the menu after answering the following to the four prompts above:

```
<enter>,<enter>,<enter>,<enter>

IBUS9051 - STARTING BUS TEST
IBUS9052 - END OF BUS TEST
```

The following display examples assume <1> was selected from the menu after answering the following to the four prompts above:

```
<enter>,<enter>,<M>,<enter>

IBUS9051 - STARTING BUS TEST
IBUS9052 - END OF BUS TEST
IBUS9010 - INDICATOR TEST, OBSERVE AND NOTE PATTERN
IBUS9012 - PRESS ANY KEY WHEN READY TO START
```

Before a key is pressed to start this test, the front door of the Interface Unit should be opened, and the DIAGNOSTIC INDICATORS should be located. They are on the second card from the right in the card cage. They consist of four seven-segment LED (light emitting diode) arrays arranged in a vertical row on the card edge. Normally, these indicators are read from top to bottom, and they display the numerical portion of the Information or Warning messages. For this test they should be read from the bottom up, since the program sequences them through the numbers 0 to 9, one indicator at a time. This test sequences all of the DATA bits and any discrepancies should be noted by indicating what the display was showing compared to what it should have been, as well as which indicator. The "Lamp Test" button should be pressed before proceeding. It is located just below the indicators. All indicators should show an "8" when this button is pressed.

The relationship of the indicators and data bits is:

Top Indicator - data bits 8, 9,10,11
2nd Indicator - data bits 12,13,14,15
3rd Indicator - data bits 0, 1, 2, 3
4th Indicator - data bits 4, 5, 6, 7

The translation from binary to seven-segment indicator is:

Data Bit*	3210	3210	3210	3210	3210
	0000 - 0	0010 - 2	0100 - 4	0110 - 6	1000 - 8
	0001 - 1	0011 - 3	0101 - 5	0111 - 7	1001 - 9

* This translation applies equally to all data bits.

Pressing any key will cause the tests to continue. After the indicators finish sequencing, the following messages will appear:

```
IBUS9011 - INDICATOR TEST COMPLETE
IBUS9053 - END OF EXTENDED BUS TEST
```

Important: Any problem with the Indicator Test (other than a defective indicator) will make any following diagnostic results questionable. The problem cause should be determined before continuing.

2 TIMER/DMA: The following display examples assume a <2> was selected from the menu after answering the following to the four prompts above:

```
<enter>,<enter>,<enter>,<enter>

ITDM9100 - STARTING TIMER AND DMA TESTS
ITDM9107 - INITIALIZATION OF TIMER AND DMA CHIPS COMPLETE
ITDM9111 - TIMER TEST PASSED
ITDM9112 - DMA TEST PASSED
ITDM9101 - TIMER AND DMA TESTS COMPLETE
```

The following display examples assume a <2> was selected from the menu after answering the following to the four prompts above:

```
<enter>,<enter>,<m>,<enter>

ITDM9100 - STARTING TIMER AND DMA TESTS
ITDM9107 - INITIALIZATION OF TIMER AND DMA CHIPS COMPLETE
ITDM9111 - TIMER TEST PASSED
ITDM9112 - DMA TEST PASSED
ITDM9101 - TIMER AND DMA TESTS COMPLETE
ITDM9108 - START OF REFRESH CHECK TEST
ITDM9109 - THIS CAN TAKE UP TO 2 MINUTES DUE TO ... D RAMS
ITDM9116 - RAM FAILED WITHOUT REFRESH IN (nn) HALF-MINUTE(S)
```

Note: The (nn) quantity is a variable which is dependent on the data retention of the dynamic memory used for DACU storage.

```
ITDM9122 - WAITING AND CHECKING RAM
ITDM9123 - REFRESH TEST PASSED
ITDM9110 - END OF REFRESH CHECK TEST
```

3 MEMORY: The following display examples assume a <3> was selected from the menu after answering the following to the four prompts above:

```
<enter>,<enter>,<enter>,<enter>

IMEM9200 STARTING RAM TEST; ADDRESSES AND DATA RIPPLE...

IMEM9205 DATA BUS RIPPLE TEST PASSED
IMEM9203 DATA COMPARES COMPLETED; RAM PASSED
```

The following display examples assume a <3> was selected from the menu after answering the following to the four prompts above:

```
<enter>,<enter>,<m>,<enter>

IMEM9200 STARTING RAM TEST; ADDRESSES AND DATA RIPPLE...

IMEM9205 DATA BUS RIPPLE TEST PASSED
IMEM9203 DATA COMPARES COMPLETED; RAM PASSED
IMEM9202 STARTING EXTENDED DIAGNOSTICS FOR RAM
IMEM9206 DATA COMPARES COMPLETED; RAM PASSED
```

4 SERIAL ADAPTER: The following display examples assume a <4> was selected from the menu after answering the following to the four prompts above:

```
<enter>,<enter>,<M>,<enter>

ISER1501 SERIAL PORT INITIALIZED
ISER1502 SERIAL PORT TEST
ISER1504 COMPLETED POLLING TESTS
ISER1503 SERIAL INTERRUPT OKAY
ISER2501 SERIAL PORT INITIALIZED
ISER2502 SERIAL PORT TEST
ISER2504 COMPLETED POLLING TEST
ISER2503 SERIAL INTERRUPT OKAY
ISER1506 DONE DMA SEND
ISER1507 SERIAL INTERRUPT OKAY (SEND)

PLEASE INSTALL SERIAL WRAP CABLE

 2
PRESS ANY KEY WHEN READY
```

Before proceeding, the rear door of the Interface Unit should be opened. (Rotate the System Unit to make this step easier.) Locate the tailgate (connector bracket) at the left bottom of the enclosure. Install a NULL MODEM cable between J6 (Serial Port 1) and J7 (Serial Port 2).

Now, press any key to continue.

```
ISER1510 CHAR SENT FROM PORT 1 TO PORT 2
ISER1511 CHAR SENT FROM PORT 2 TO PORT 1
ISER1505 DONE DMA RECEIVE
ISER1508 SERIAL DMA INTERRUPT OKAY (RECEIVE)
ISER1511 CHAR SENT FROM PORT 2 TO PORT 1
ISER1506 DONE DMA SEND
1SER1507 SERIAL DMA INTERRUPT OKAY (SEND)
ISER1510 CHAR SENT FROM PORT 1 TO 2
ISER1510 CHAR SENT FROM PORT 1 TO 2
ISER1510 CHAR SENT FROM PORT 1 TO 2
```

5 Parallel I/O ADAPTER: The following display examples assume a <5> was selected from the menu after answering the following to the four prompts above:

```
<enter>,<enter>,<enter>,<enter>

IUNI4500 - PARALLEL I/O INTERFACE HAS BEEN INITIALIZED
IUNI4506 - NOW STARTING IRQ4 TEST
IUNI4502 - IRQ4 TEST PASSED
IUNI4510 - END OF IRQ4 TEST
IUNI4538 - START OF BUS REQ/BUS GRANT TEST
IUNI4500 - PARALLEL I/O INTERFACE HAS BEEN INITIALIZED
IUNI4500 - PARALLEL I/O INTERFACE HAS BEEN INITIALIZED
IUNI4527 - BRBG TEST PASSED
IUNI4539 - END OF BUS REQUEST BUS GRANT TEST
IUNI4507 - NOW STARTING DATA BUFFER TEST
IUNI4500 - PARALLEL I/O INTERFACE HAS BEEN INITIALIZED
IUNI4509 - THE DATA CHECKED OUT
IUNI4511 - END OF DATA BUFFER TEST
IUNI4500 - PARALLEL I/O INTERFACE HAS BEEN INITIALIZED
IUNI4512 - START OF ADDRESS BUFFER TEST
IUNI4515 - ADDRESS BUFFERS ARE OK
IUNI4513 - END OF ADDRESS BUFFER TEST
```

The following display examples assume a <5> was selected from the menu after answering the following to the four prompts above:

```
<enter>,<enter>,<M>,<enter>

IUNI4500 - PARALLEL I/O INTERFACE HAS BEEN INITIALIZED
IUNI4506 - NOW STARTING IRQ4 TEST
IUNI4502 - IRQ4 TEST PASSED
IUNI4510 - END OF IRQ4 TEST
IUNI4538 - START OF BUS REQ/BUS GRANT TEST
IUNI4500 - PARALLEL I/O INTERFACE HAS BEEN INITIALIZED
IUNI4500 - PARALLEL I/O INTERFACE HAS BEEN INITIALIZED
IUNI4527 - BRBG TEST PASSED
IUNI4539 - END OF BUS REQUEST BUS GRANT TEST
IUNI4507 - NOW STARTING DATA BUFFER TEST
IUNI4500 - PARALLEL I/O INTERFACE HAS BEEN INITIALIZED
IUNI4509 - THE DATA CHECKED OUT
IUNI4511 - END OF DATA BUFFER TEST
IUNI4500 - PARALLEL I/O INTERFACE HAS BEEN INITIALIZED
IUNI4512 - START OF ADDRESS BUFFER TEST
IUNI4515 - ADDRESS BUFFERS ARE OK
IUNI4513 - END OF ADDRESS BUFFER TEST
IUNI4519 - THIS IS THE START OF THE WRAP TEST
IUNI4523 - PLEASE INSERT WRAP FIXTURE..... PRESS ANY KEY ..
```

Before continuing, the Wrap Card (provided with the DACU) must be plugged into the DD11 connector, located just to the right of the card cage when viewed from the rear. If all of the peripheral slots are filled, it will be necessary to remove any one of the peripheral controller cards temporarily. The Wrap Card is designed to fit into the connector even if a 'Grant Continuity' card is installed.

```
IUNI4500 - PARALLEL I/O INTERFACE HAS BEEN INITIALIZED
IUNI4524 - ACTUAL DATA ... IS..EAAB  .. EXPECTED ... IS EAAB
IUNI4524 - ACTUAL DATA ... IS..F556  .. EXPECTED ... IS F556
IUNI4521 - THE WRAP TEST PASSED
IUNI4520 - THIS IS THE END OF THE WRAP TEST
IUNI4551 - REMOVE WRAP FIXTURE, PRESS ANY KEY WHEN READY
```

Now, remove the Wrap Card that you installed earlier.

```
IUNI4500 - PARALLEL I/O INTERFACE HAS BEEN INITIALIZED
IUNI4540 - START OF DEVICE REGISTER WRITE/READ TEST
IUNI4549 - ENTER DEVICE REGISTER WRITE/READ ADDRESS
IUNI4552 - FIRST TIME NULL LINE WILL END TEST
           ELSE NULL LINE EXECUTES ADDRESS PREVIOUSLY ENTERED.
```

The address referred to here is any VALID write/read register address in Parallel I/O register space. The address used should be capable of having all bits written into and then read back.

Just pressing <enter> to the request for the first address will abort the test. After an address has been entered, pressing <enter> will retest the previous address.

If an address is entered that is not in Parallel I/O address space, the following will be displayed:

```
<address>
IUNI4500 - PARALLEL I/O INTERFACE HAS BEEN INITIALIZED
WUNI4550 - REGISTER ENTERED NOT IN PARALLEL I/O ADDRESS SPACE...
ABORTING TEST!
```

Entering a valid address brings the following response:

```
<address>
IUNI4500 - PARALLEL I/O INTERFACE HAS BEEN INITIALIZED
IUNI4542 - NOW ATTEMPTING A WRITE OF DATA WORD AAAA
           TO A DEVICE REGISTER LOCATION <address>
IUNI4543 - THE WRITE TRANSACTION IS DONE
IUNI4544 - NOW ATTEMPTING A READ OF DEVICE REGISTER<address>
INUI4545 - READ DATA FROM THE DEVICE REGISTER IS  AAAA
IUNI4542 - NOW ATTEMPTING A WRITE OF DATA WORD 5555
           TO A DEVICE REGISTER LOCATION <address>
IUNI4543 - THE WRITE TRANSACTION IS DONE
IUNI4544 - NOW ATTEMPTING A READ OF DEVICE REGISTER<address>
INUI4545 - READ DATA FROM THE DEVICE REGISTER IS  5555
IUNI4541 - END OF DEVICE REGISTER WRITE/READ TEST

IUNI4553 - NOW BEGINNING A TEST TO FIND ALL ACTIVE DEVICE REGISTERS
           TEST TAKES ABOUT 20 SECONDS - BE PATIENT!
IUNI4500 - PARALLEL I/O INTERFACE HAS BEEN INITIALIZED
IUNI4554 - ACTIVE REGISTERS WILL BE INDICATED BY R OR W.
            0 1   2 3   4 5   6 7   8 9   A B   C D  E F
7000:F500   **    **    **    **    RW    RW    RW   RW
IUNI4554 - ACTIVE REGISTERS WILL BE INDICATED BY R OR W.
            0 1   2 3   4 5   6 7   8 9   A B   C D  E F
7000:F530   **    **    **    RW    RW    RW    RW   RW
IUNI4554 - ACTIVE REGISTERS WILL BE INDICATED BY R OR W.
            0 1   2 3   4 5   6 7   8 9   A B   C D  E F
7000:F540   RW    RW    RW    RW    RW    RW    RW   RW
IUNI4555 - FINISHED TESTING DEVICE REGISTERS
```

The above registers reflect only a typical test situation, and the actual display may vary, depending on the attached Parallel I/O device.

6 CHANNEL ATTACHMENT: The following display examples assume a <6> was selected from the menu after answering the following to the four prompts above:

```
<enter>,<enter>,<enter>,<enter>

IOEM6500 OEMI BRING-UP DIAGNOSTICS ENTERED

IOEM6501 OEMI TEST MODE 0 PASSED

IOEM6502 OEMI TEST MODE 1 PASSED

IOEM6515 INTERRUPT RECEIVED
IOEM6503 OEMI TEST MODE 2 PASSED

IOEM6511 OEMI DATA AND STATUS REGISTER WRAP TEST PASSED
```

The following display examples assume a <6> was selected from the menu after answering the following to the four prompts above:

```
<enter>,<enter>,<m>,<enter>

IOEM6500 OEMI BRING-UP DIAGNOSTICS ENTERED

IOEM6501 OEMI TEST MODE 0 PASSED

IOEM6502 OEMI TEST MODE 1 PASSED

IOEM6515 INTERRUPT RECEIVED
IOEM6503 OEMI TEST MODE 2 PASSED

IOEM6511 OEMI DATA AND STATUS REGISTER WRAP TEST PASSED

IOEM6521 OEMI MANUAL INTERVENTION TEST
         ENABLE CONTROL UNIT ENABLE SWITCH
         HIT ANY KEY WHEN READY
```

Note: The Enable/Disable switch is located on the DACU Operator Panel.

```
IOEM6522 ENABLE SWITCH IS ENABLED
         NOW THROW SWITCH TO DISABLE POSITION
         HIT ANY KEY WHEN DISABLED

IOEM6523 ENABLE SWITCH IS DISABLED

IOEM6526 OEMI DISABLE IRQ BIT SET IN ADAPTER
         TOGGLE ENABLE SWITCH FROM DISABLE
         TO ENABLE, AND BACK TO DISABLE
         HIT ANY KEY WHEN READY

IOEM6528 OEMI DISABLE INTERRUPT TEST SUCCESSFUL

IOEM6534 DMA TEST FROM CHANNEL

         THIS TEST LEAVES THE ADAPTER IN A BAD STATE

         ENTER Y IF YOU WANT TO RUN THIS TEST
IOEM6540 DMA TEST PASSES WHILE READING FROM PAGE 5

IOEM6541 DMA TEST PASSES WHILE READING FROM PAGE 6

IOEM6542 DMA TEST PASSES WHILE READING FROM PAGE 7

IOEM6535 DMA TEST PASSED

         OEMI ADAPTER MAY BE IN A BAD STATE
         PLEASE RESET THE CONTROL UNIT BEFORE GOING ONLINE
```

To reset the control unit, power it off and back on.

This is the end of the 'Extended Diagnostics'. Remove all cards/cables that were installed as part of the diagnostics.

The display examples above assume an error-free DACU, and any problems will cause the diagnostic readouts to vary from what was presented above. Extended diagnostic information and warning messages are listed in Appendix D, "Diagnostic Monitor Messages" on page 225.

Error Response: Below is an example of the displays to be expected if the following is entered in response to the four prompts and if an error occurs during any diagnostic:

```
<enter>,<s>,<enter>,<enter>

WAAANNNN - THE ________ TEST FAILED
IMON3010 - PRESS G TO CONTINUE TESTS, ANY OTHER KEY TO ABORT
```

The first message received starts with a 'W'. The content of the (AAANNNN) field depends on which diagnostic was running when the problem occurred. The actual message is also dependent on the diagnostic. Pressing <G> (either upper or lower case) will cause the tests to continue where they left off. Pressing any other key will return you to the DOS prompt.

The next four pages are flow charts to assist in problem determination. They should be used in conjunction with the Problem Determination Procedures (See "Problem Determination Procedures" on page 107) to assist in correcting any problems experienced with the DACU.

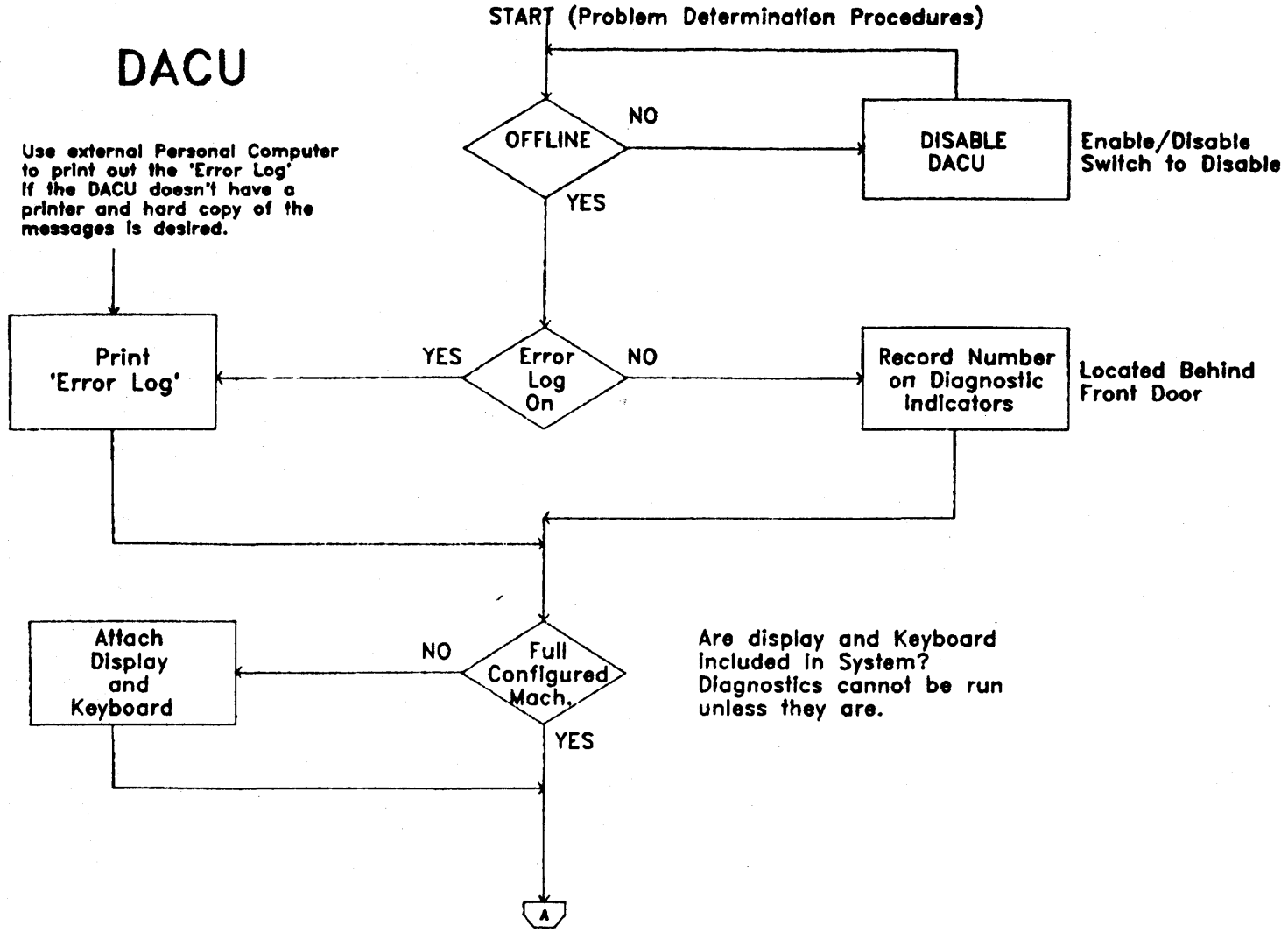
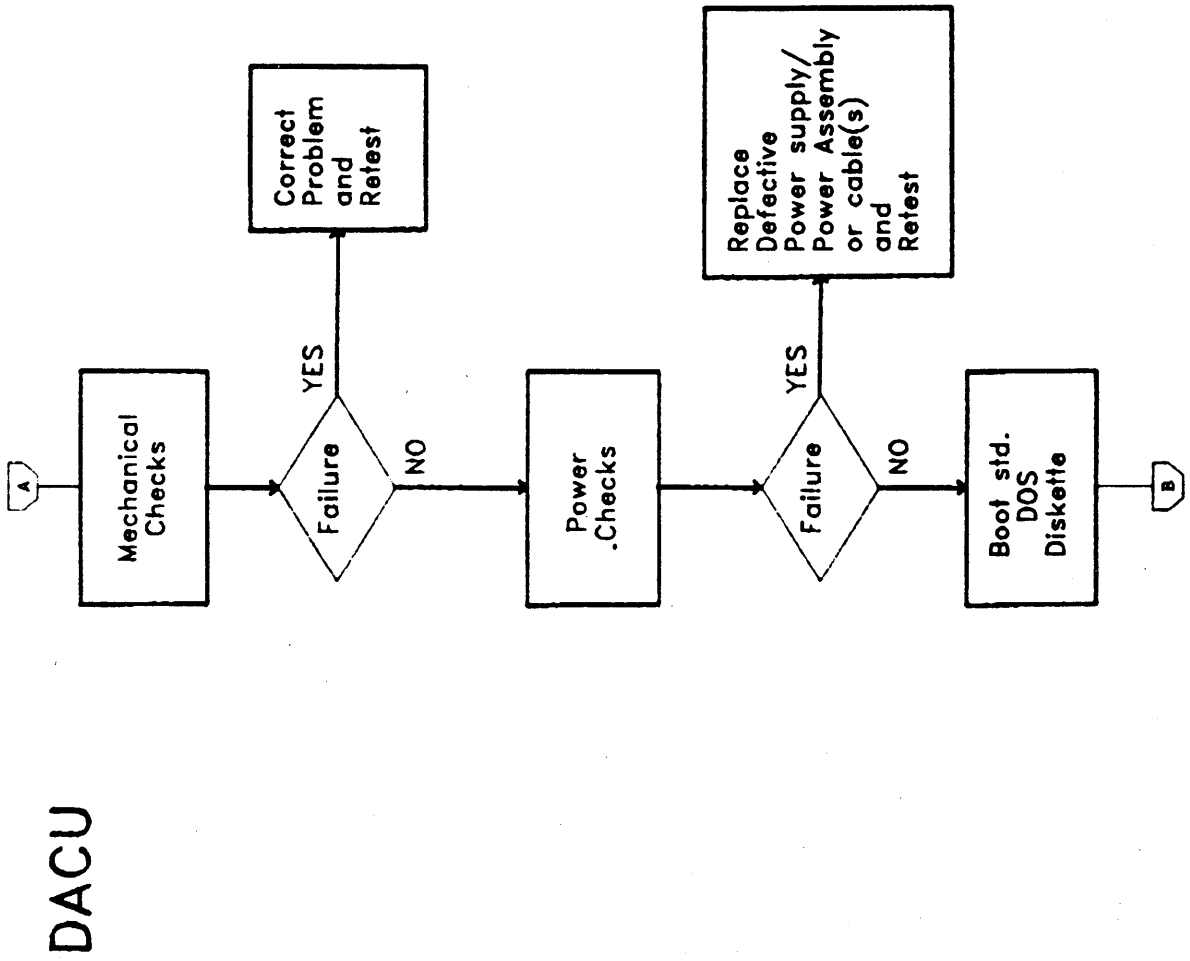


Figure 20 (Part 1 of 4). Problem Determination Flow Charts



DACU

Figure 20 (Part 2 of 4). Problem Determination Flow Charts

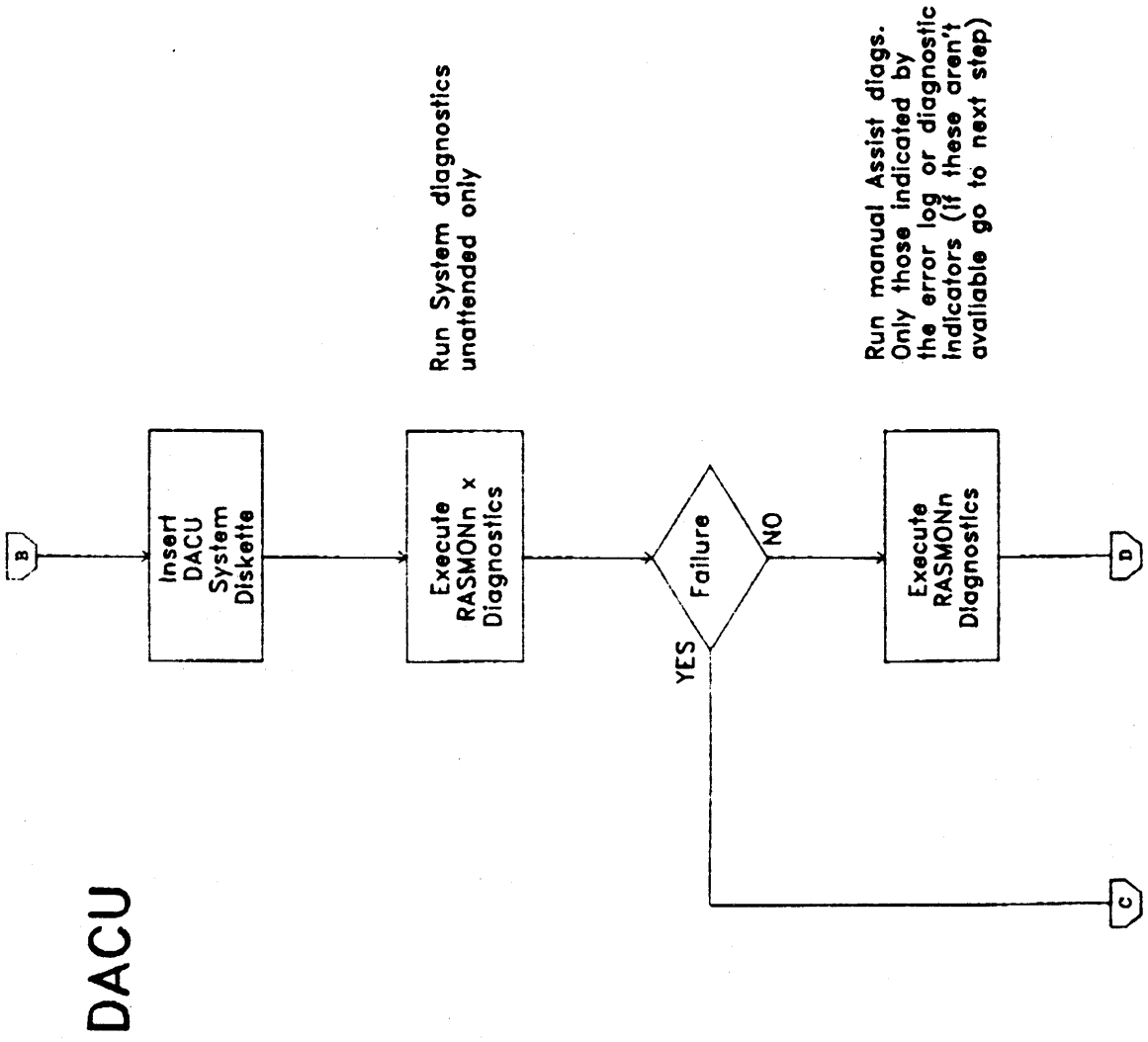


Figure 20 (Part 3 of 4). Problem Determination Flow Charts

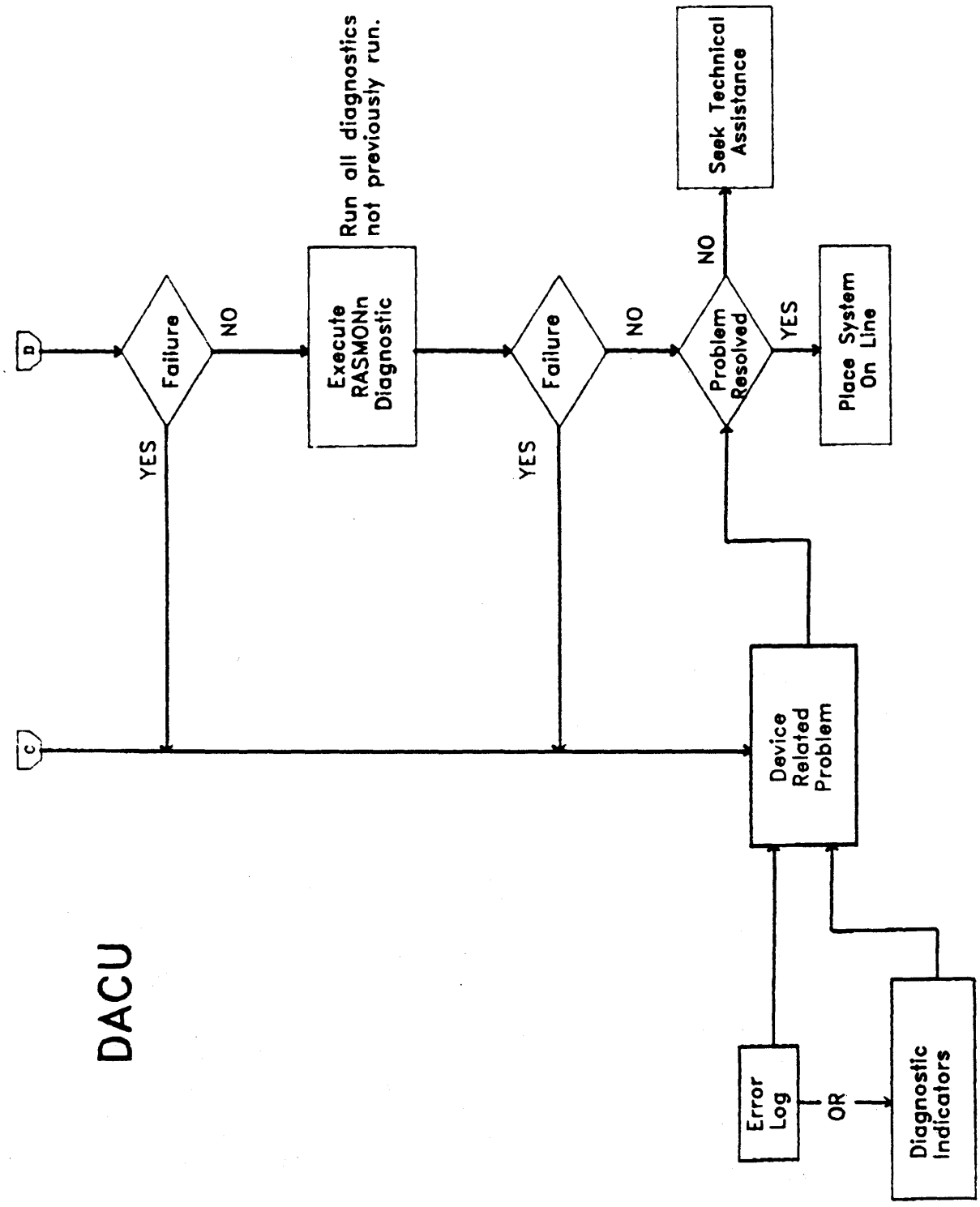


Figure 20 (Part 4 of 4). Problem Determination Flow Charts

Unit Replacement

The customer is responsible for maintenance of the DACU Interface Unit and diagnosis of problems. Repair is provided by the manufacturer at the manufacturer's site. Unit return and replacement or repair can be made at three levels:

- Entire DACU Interface Unit assembly
- Major replaceable assemblies
- Individual field replaceable units.

Down time can be minimized through a spares stocking program.

Field Replaceable Unit

The following units are replaceable by the customer and repairable by the manufacturer:

1. Entire DACU Interface Unit
2. DACU card cage with installed cards
3. Individual cards
4. Designated parts for which the manufacturer provides a price schedule (See below)
5. Power assembly.

Spare Parts

The following parts will be carried in the manufacturer's inventory and may be ordered by customers to replace a failed unit or for spares stocking:

P/N		P/N	
1802691	Power Adapter Card	9130337	Power Assembly (Domestic)
0174971	Interface Unit Adapter Card	0174951	Power Assembly (World Trade)
1802705	Card Cage	4130345	Diskette
4130327	System Unit Adapter Card	0174955	Parallel I/O Wrap Card
0174967	DMA/Memory Card	4130346	Reference and Operations Manual
0174957	OEMI Adapter Card	4130336	Parallel I/O Cable Terminator
0174953	Parallel I/O Adapter Card	1802696	SU/IU Cable
1802692	OEMI Interface Cable	0174959	DD11 Connector Assembly
1802695	OEMI Interface Cable (Channel Out BUS or TAG)	1802693	Serial I/O Interface Cable CH1
1802688	Parallel I/O Interface Cable	4130340	Serial I/O Interface Cable CH2

Replacement of DACU Unique Components in DACU System Unit

Should DACU diagnostics point to a failing component in the DACU System Unit, it may be necessary to replace one of the two assemblies installed there (SU Extender card or Power Adapter card). The IBM PC *Guide to Operations* manual should generally be used to guide adapter card replacement. For the DACU cards, use the following instructions:

1. Position the DACU Enable/Disable switch to "DISABLE."

2. After the disable indicator comes ON, position the DACU power switch to OFF. Further, turn DACU System Unit power off by positioning the System Unit power switch to the OFF position (down).

3. Remove the cable between the DACU System Unit and DACU Interface Unit.

4. Remove the cover of the PC System Unit.

   The instructions that continue describe replacement of the DACU SU Extender card housed in the DACU System Unit.

5. Remove the two wire cable plugged into the four position connector on the SU Extender card.

6. Following the PC *Guide to Operations* manual, remove the SU Extender card. Package it for shipment if it is to be returned to the manufacturer. Temporarily remove the DACU Power Adapter card installed between the System Unit power supply and the System Unit system board. Seat the System Unit power connector on the System Unit power supply connector.

7. Reassemble the System Unit and rerun the PC Power On Self Test procedure. (For North America installations only, it will be necessary to plug the System Unit into an external power outlet other than that in the DACU Interface Unit for these tests. After the tests are complete, re-plug the System Unit power cord in the DACU Interface Unit power outlet.)

8. Turn off the power on the System Unit and remove the System Unit cover again.

9. Unpack the replacement SU Extender card and install it in the System Unit using the same procedure as for original installation. Also, reinstall the Power Adapter card between the System Unit power supply and connector J1 on the System Unit system board. Replug the previously removed two wire cable onto the new SU Extender card. Finally, replace the System Unit cover.

10. Replace the cable between the System Unit and the DACU Interface Unit. Position the System Unit power switch to ON (up).

11. Power on the DACU and rerun all bring-up diagnostics.

Replacement of DACU Components in Interface Unit

Instructions for replacing Interface Unit components are outlined in the sections below. The connector designations used in these sections and their locations are illustrated below.

```
J1    CHANNEL IN          CONTROL UNIT OUT      J6   J7   J8

      J2        J3        J4        J5          J9   J10  J11

      BUS       TAG       BUS       TAG              J12
```

Conn	Function
J1	UNUSED
J2-J5	CHANNEL
J6	SERIAL PORT 1
J7	SERIAL PORT 2
J8-J12	UNUSED

Logic Card Cage Replacement

Use the following instructions to remove the entire DACU Logic Card Cage for replacement:

1. Halt the host computer before proceeding. (Before continuing, it will be necessary to isolate the DACU from the host computer. The actual way this is accomplished depends on the system configuration. The host computer must be stopped if no other way of isolating the DACU is available. If this is not done, "channel errors" will be detected by the host computer.)

2. Position the Enable/Disable switch to DISABLE.

3. After the Disable Indicator comes ON, position the DACU power switch to OFF. REMOVE THE DACU POWER CORD FROM THE POWER SOURCE.

4. Open the rear door of the DACU Interface Unit.

5. Locate the Motherboard — the board into which all of the DACU logic cards are plugged (at the rear of the card cage).

6. Disconnect the power cable from the Motherboard.

7. Disconnect the following signal cables, removing any interfering cable retention devices (possibly cutting cable ties):

Logic Card	Signal Cable
IU ADAPTER	01-J1-J4 through J6
DMA/MEM	01-J2-J4 and J5
OEMI ADAPTER	01-J4-J4 through J11
OEMI ADAPTER	01-J4-J18 (small cable to upper middle part of card)
Parallel Adapter	01-J6-J4 through J6

N.B., the following component location scheme applies here:

```
01-Jx-Jx
 |  |  |
 |  |  Cable Connector
 |  Card Socket
 Card Cage
```

8. Close the DACU rear door.

9. Open the DACU front door and locate the card cage.

10. Disconnect the frame ground strap attached to the bottom of the cage, making sure that the screw and lock washer are retained.

11. Loosen the two quarter turn screws that hold the card cage in the DACU enclosure.

12. Carefully pull the card cage out towards the front of the DACU until it clears the enclosure.

13. Unpack the replacement card cage and install it by reversing Steps 3 to 11 above. Install cable retention using cable ties provided with the replacement card cage.

14. ASSURE THAT THE SAFETY GROUND WIRE IS REATTACHED.

15. Package the removed card cage for shipment if it is to be returned to the manufacturer. Use all packing materials that were provided (e.g., corner braces, foam cushions) with the replacement cage.

After installation of the replacement card cage, turn the DACU power on and rerun the system bring-up diagnostics.

Logic Card Replacement

The same replacement procedures apply to all cards in the card cage. Only the number and location of cables that must be removed will vary from card to card.

Replacement instructions follow:

1. Position the Enable/Disable switch to DISABLE. (If the card to be replaced is the OEMI Adapter it will be necessary to isolate the DACU from the host system before continuing. The actual way this is accomplished depends on the system configuration. The host computer must be stopped if no other way of

isolating the DACU is available. If this is not done, "channel errors" will be detected by the host computer.)

2. When the Disable indicator comes ON, position the DACU power switch to OFF.

3. Open the rear door of the DACU.

4. Locate the card cage.

5. Unplug the cables identified by the chart below. Remove any cable retention at this time. (It may be necessary to cut cable ties.)

Socket	Card	J4	J5	J6	J7	J8	J9	J10	J11
J1	IUA	X	X	X					
J2	DMA/MEM	X	X						
J4	OEMI	X	X	X	X	X	X	X	X
J4	OEMI	Disc J18 from mid upper location							
J6	PAR I/O	X	X	X					

Figure 21. Card Cable Plug Matrix

Because of cable routings, it may be necessary to temporarily remove cables from other cards. (except the IUA card) to the left of the card being removed. (The IUA cables are routed away from all other cards, and will not interfere with removal of any other card.)

If the OEMI card is to be replaced, the host computer must be halted or otherwise disabled.

6. Close the rear door.

7. Open the front door and locate the card cage.

8. Locate the desired card from the following chart:

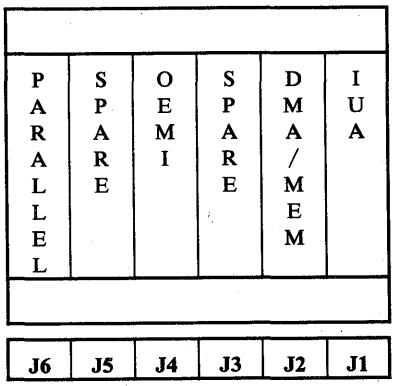


Figure 22. Card Plug Locations

9. Locate the card retention bar on the cage. Remove the screw on the right, *loosen* the screw on the left, and pivot the bar down out of the way. Save the screw removed.

10. Operate the removal levers on the desired card to allow removal.

11. Remove the card and package for shipment if it is to be returned to the manufacturer. (Use packaging materials that will provide proper shipping protection. The packaging materials provided with a replacement card should be used if possible.)

12. Unpack the replacement card and install by reversing Steps 3 through 11 above. Be sure to reposition the card retention bar and tighten in place. Install cable retention using cable ties provided with the replacement card. Verify visually that each connector is inserted properly.

System bring-up diagnostics should be completely run following the replacement of any card.

Power Assembly Replacement

```
THERE ARE NO CUSTOMER SERVICEABLE COMPONENTS
        IN THE POWER ASSEMBLY.
```

1. Position the Enable/Disable switch to DISABLE.

2. When the Disable indicator comes ON, position the DACU power switch to OFF.

3. DISCONNECT THE DACU FROM THE POWER OUTLET.

4. Open the rear door and locate the Power Assembly Unit located to the right of the Parallel I/O DD11-B connector assembly. Raise the hinged DD11-B assembly to the service position.

5. Disconnect the following cables from the Power Assembly:

   a. J1 to J5
   b. Ground Wire (Save screw and lockwashers).

6. Loosen, the four (two on top, two on bottom) screws holding the power assembly in place. The two screws near the back of the machine must be removed and retained for later replacement.

7. Pull the assembly toward the rear of the DACU enclosure until the upper and lower brackets are clear of the screws.

8. Move the front of the assembly to the left until it clears the frame. Then, carefully remove the assembly from the enclosure.

All that remains to be done is to unpack the replacement assembly and install it using Steps 1 through 8 above in reverse.

> ASSURE THAT THE SAFETY GROUND WIRE IS REATTACHED USING ALL OF THE ORIGINAL HARDWARE.

The removed Power Assembly should be repacked using the packaging materials from the replacement unit if the assembly is to be returned to the manufacturer.

Complete bring-up diagnostics should now be run.

Chapter 7. Physical Characteristics and Installation Planning Information

Physical Specifications

Physical specifications given here relate only to the DACU Interface Unit and do not include specifications associated with the required DACU System Unit.

Dimensions

	FRONT	SIDE	HEIGHT	WEIGHT
Millimeters:	500	600	660	45 Kg
(Inches):	19.7	23.6	26.0	100 LB

Power Cable

Length: 1.83 Meters (6 feet)

Size: No. 16 AWG

Environment

Air Temperature:

System On: 60 to 90 degrees F (16 to 32 degrees C)

System Off: 50 to 110 degrees F (10 to 43 degrees C)

Heat Output: 910 BTU/Hr (maximum)

Noise Levels

Not Applicable. (No air-moving device or other mechanical moving devices.)

Electrical

All line voltage figures are in volts AC.

		Feature Code (Outside U. S.)				
	U.S.	9911	2801	2806	2732	2813
Nominal Voltage	120	120	240	200	200	220
Minimum Voltage	104	104	228	190	190	209
Maximum Voltage	127	127	252	210	210	231
KVA (Max)	.227	.227	.227	.227	.227	.227
Line Frequency (HZ)	60	60	60	50	60	50

Figure 23. Line Voltage Specification

Plan View and Service Clearance

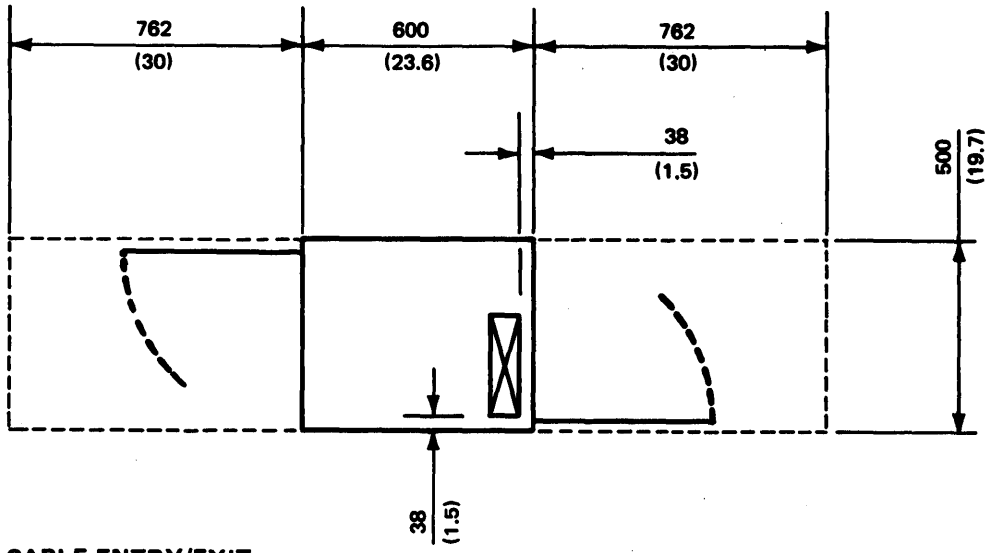


Figure 24. Plan View/Clearance Drawing

Notes:

1. *Cable entry/exit holes are measured from edge of frame, not covers.*
2. *Enclosure design allows for installation on either raised floor or non-raised floor.*
3. *The DACU Interface Unit enclosure does not have skirts. As a result, cable exit points for non-raised floor installations are restricted only by the location of frame legs.*

Customer Provided Cable and Parallel I/O Element Requirements

One cable will be provided with the DACU to interconnect the DACU Interface Unit and the customer-provided DACU System Unit. This cable provides communication with the PC Expansion Slot and power-on reset facility.

It is the responsibility of the customer to provide cables to UNIBUS-compatible devices and RS-232-C cables. In North America only, it is the responsibility of the customer to provide IBM channel cables.

Packaging

The DACU as a functional unit consists of the DACU Interface Unit and an IBM Personal Computer (the DACU System Unit). The DACU Interface Unit consists of a free standing cabinet with the dimensions described above. The DACU System Unit can be located on top of the DACU Interface Unit with or without an accompanying IBM Monochrome Display. (The Monochrome Display and Keyboard are not required for production operation of the DACU; however, they are required for application development and execution of DACU diagnostic programs.) The PC keyboard is accommodated by a detachable keyboard shelf on the front of the Interface Unit cabinet.

Internal to the Interface Unit and accessible via the front cabinet door is a printed circuit card cage housing all of the electronic logic and memory provided with the Interface Unit. Power supplies are also located in this area.

Accessible via the rear cabinet door is the area that houses the DD11 connector block into which are plugged the UNIBUS-compatible device adapter cards provided by the customer.

External I/O cables (IBM channel, Parallel I/O devices, and Serial I/O devices) are routed into the DACU Interface Unit from underneath whether or not the installation includes raised floor.

There are six printed circuit multi-layered cards located in a DACU configuration. Four of these cards are located in the Interface Unit, and the remaining two are located in the System Unit.

The cards in the Interface Unit are plugged in a six slot motherboard. The motherboard provides voltage and signal distribution to all the cards. The cards and motherboard are housed in a card cage which provides support for the cards. The card cage assembly is packaged as a replaceable unit. The function of each card is described below.

System Unit Extender Card

The System Unit Extender (SUE) card, which is installed in the DACU System Unit provides buffering for all signals that are transmitted over the cable between the System Unit and Interface Unit. In addition the DACU System Unit reset ("POWER GOOD") signal is intercepted by this card and multiplexed with the channel reset signal allowing the channel to reset the DACU System Unit as well as the DACU Interface Unit. This card occupies one System Unit expansion slot. The cable that connects the System Unit to the Interface Unit plugs into the connector located on this card. The other end of this cable terminates at the Interface Unit Adapter (IUA) card via a connector located at the rear of the Interface Unit.

Power Adapter Card

The Power Adapter Card, installed in the System Unit, is inserted between one of the System Unit power cables (P8) and the system board connector (J1) to intercept the "POWER GOOD" signal for use as described for the SUE card above.

Interface Unit Adapter Card

The Interface Unit Adapter (IUA) card, installed in the Interface Unit, is the companion of the SUE card in providing transmitters and receivers that buffer all signals transmitted over the cable between the System Unit and Interface Unit. This card provides communication with the System Unit and generates the DACU Interface Unit signal bus.

DMA/Memory Card

The DMA/Memory card, installed in the Interface Unit, contains the following functions:

1. 8253 Timer Counter[9] which performs the following functions:
   a. Memory refresh timing pulse generator
   b. Transmit and Receive clocks for the serial I/O chip (8274).
2. 8237 DMA device[9] which services the following areas:
   a. Memory refresh for the DACU Interface Unit dynamic RAM (128k)
   b. DMA memory/IO transfer from/to the serial IO ports
   c. DMA memory/IO transfer from/to the OEMI channel.
3. 8274 Multi-Protocol Serial Controller[9] Serial I/O (RS-232C serial communications).

The DMA/Memory card also contains the following miscellaneous registers and controls:

1. SIOCCTL REGISTER—A four bit write only register which selects the source for the baud rate clock used for serial transfers.
2. DIAGNOSTIC INDICATORS—Four 7-segment LED indicators provided for presentation of 'Information' or 'Warning' messages. There is a test switch which forces all indicators to display an '8', when pressed.

---

9 For additional information on these functional devices refer to the Intel Component Data catalog.

3. SIOSTAT REGISTER—A four bit read only register, used to indicate miscellaneous status information.

4. PAGE REGISTER—An eight bit write only register. This register performs memory page control by providing address bits A16 and A17 to the DACU Interface Unit address bus for DMA transfers.

5. Parity Error—Reading the DACU memory generates parity. The parity bit is latched on this card and then tested by reading the SIOSTAT REGISTER. The Parity Error latch is reset by this read operation.

OEMI Adapter Card

The OEMI Adapter card, installed in the Interface Unit, contains the circuitry necessary to attach to a 43XX or 308X channel.

Adapter for Parallel I/O Interface

This card, installed in the Interface Unit, provides the circuitry necessary to provide an interface compatible with devices designed for attachment to the Parallel I/O Interface.

Chapter 8. Performance Characteristics

Data Transfer Rates

The DACU has the performance characteristics listed below.

Channel Cable Length (Feet)	Peak Data Rate (Megabytes/sec)	
	Without Refresh Interference Constraint	With
0 - 15	1.90	1.28
15 - 85	1.59	1.07
85 - 155	1.36	0.92
155 - 225	1.19	0.80

Figure 25. Data Transfer Rates

(Maximum performance for one interface facility assumes use of that facility and none of the others simultaneously.)

Max. data rate to UNIBUS-compatible device: (May be limited by the maximum data rate of the device and its bus request scheme.) 2.0 MBytes/sec

Max. RS-232-C data rate: PORT 1: 19.2K bits/sec (asynchronous - send and receive using DMA)

PORT 1 AND PORT 2: 9.6K bits/sec (asynchronous - send and receive using programmed I/O)

Maximum throughput from the host system to a Parallel I/O or Serial device is subject to a number of considerations, including the following:

1. There is an initialization time cost associated with the transfer of each block of data.

2. All data transfers from one interface adapter to another (e.g., host channel to device) are single buffered. Hence, the total data transfer time (not including initialization overhead) is the sum of the time to transfer a block from one adapter to the buffer and the time to transfer it from the buffer to the other adapter.

3. To help in the computation of data throughput, it may be helpful to know some of the times for processing commands in the DACU. For example, an important time to know is the time from receipt of a Set-Start-Register command to the Device End sent back to the host including the time for the control space interpreter. This time is about 12 milliseconds not counting the time spent in the interpreter. The interpreter time is dependent on the application but amounts to about 200 microseconds per command. Also, if the application polls for a condition in the control space, this time becomes device-dependent.

   It takes about 6 milliseconds from receipt of a device interrupt to the acceptance of the attention by the channel.

   The time from the attention to the sense that follows will depend on the host system as will the time from the sense until the Read XY-Position. The time for handling a Read XY-Position command is about eight milliseconds.

4. An improvement of DACU processing time of about 12% can be accomplished by not enabling the in-memory trace of messages.

Programming for Maximum Performance

The DACU design includes a jumper on the Memory/DMA Controller card which may be positioned to either allow or disable refresh interference of OEMI data transfers. As highlighted by Figure 8 on page 29, when positioned to allow refresh interference (making refresh of DACU memory automatic), the peak data rate capability of the OEMI adapter is 1.28M bytes/second (for short cable lengths). On the other hand, if the jumper is positioned to disable automatic refresh cycles (avoiding refresh interference), the peak data rate capability is 1.90M bytes/second. However, in this case, since automatic refresh is disabled, refresh must be assured by other means.

This operational constraint is a function of the size of data blocks and the rate at which they are transferred (which is a function of channel transfer rate and cable length). If automatic refresh is disabled, then the customer must insure that the channel transfer peak rate is at least 128K bytes/second and that the block lengths of the data transfer to/from the DACU is always equal to or greater than 256 bytes.

Summarizing, with the refresh control jumper on the Memory/DMA Controller card positioned to disable automatic refresh, memory integrity can be assured by guaranteeing data block lengths *shorter* than those outlined above or *longer* than 256 bytes.

Concurrency Considerations

Maximum data rates above assume one data path active only (e.g., OEMI-to-Parallel I/O Interface). Simultaneous use of paths may have the effects shown below[10].

Operation Condition	Effect On			
	OEMI Interface	Parallel Interface	Serial Interface	DACU Processor
OEMI in data burst	1.3 Megabytes/sec refresh mode; 1.9 Megabytes/sec non-refresh.	Some Interference	Serial I/O in DMA mode will interfere; Serial I/O overruns may occur under severe conditions.	Some Interference
UNIBUS compatible device in DMA data transfer	Some Interference	2.0 Megabytes/sec achieveable	Some Interference	Some Interference
Serial data transfer	Serial I/O in DMA mode will interfere; Serial I/O overruns may occur under severe conditions.	Some Interference	Under severe conditions, one Serial I/O device may cause overruns on the other.	Some interference
DACU Processor	Some Interference	Some Interference	Some Interference	

Figure 26. Concurrent Data Transfer Interference Matrix

Notes:

1. *Serial I/O in DMA mode will interfere; Serial I/O overruns may occur under severe conditions.*
2. *Under severe conditions, one Serial I/O device may cause overruns on the other.*

---

10 It is assumed that a UNIBUS-compatible device will relinquish the Parallel I/O interface after each DMA data transfer cycle.

Appendix A. Sample Host Programming

Host Application Language Library using Start I/O

Introductory Information

The Host Application Language Library using Start I/O (HALLS) is a package that provides a host high level language interface to the Device Attachment Control Unit (DACU) via calls to IBM System/370 subroutines. The package is written to execute on VM systems. The library contains 2 internal subroutines and 8 user callable subroutines. The internal routines are a Start I/O driver subroutine (STARTIO) and a device interrupt handler subroutine (HANDLER). The user callable routines establish and remove linkage to the interrupt handler (DOPEN and DCLOSE), issue I/O to the DACU (DWRITE, DREAD, DSTART, and DALARM), and wait or check for an attention from the DACU (DATTNW and DATTNI). All routines are described in detail below. All user callable subroutines must pass parameters as addresses to the required data, not the actual data.

HALLS is entered into the system as source code. To create a TEXT file, which may be linked to application TEXT files via the CMS LOAD command, the following two CMS statements must be executed:

```
GLOBAL MACLIB CMSLIB OSMACRO DMSSP
ASSEMBLE HALLS
```

User Callable Subroutine Explanation

DOPEN - DACU Open:

Format: DOPEN(DEVAD)

Where: DEVAD = 4 byte device address to issue I/O to

This routine establishes linkage to the device interrupt handler (HANDLER) via an SVC 202 (CMS HNDINT), and saves the device address for later use by the internal STARTIO subroutine. Only interrupts from the device address specified by the DEVAD parameter will be fielded by the interrupt handler.

DCLOSE - DACU Close:

Format: DCLOSE

This routine removes linkage to the device interrupt handler (HANDLER) via the CMS HNDINT macro. Interrupts generated from the device address specified in the DOPEN subroutine will no longer be trapped by the interrupt handler after execution of the DCLOSE subroutine.

DWRITE - DACU Write:

Format: DWRITE(LENGTH,DATA,ADDRESS,RC,STARTREG)

Where:

LENGTH = 4 byte length of data in bytes
DATA = Array of data to write
ADDRESS = 4 byte DACU address to write data
RC = 4 byte return code
STARTREG = Optional 2 byte start register value

This routine builds the necessary Channel Command Words (CCW's) to perform a Set Address Register and Write to the DACU and then calls the STARTIO subroutine to perform the I/O operation. The return code from the STARTIO subroutine is passed in the RC parameter. The routine performs the equivalent of a GAM GCTRL type IND (Set Program Function Indicators) and a GAM GWRITE (Write to Graphic-Device Buffer).

If the optional STARTREG parameter is specified, then a Set Start Register, equivalent to a GAM GCTRL type STR (Set Buffer Address Register and Start), will also be performed by chaining the Start's CCW to the end of the CCW string.

DREAD - DACU Read:

Format: DREAD(LENGTH,BUFFER,ADDRESS,RC,STARTREG)

Where:

LENGTH = 4 byte length of data in bytes
BUFFER = Array to read data into
ADDRESS = 4 byte DACU address to read data from
RC = 4 byte return code
STARTREG = Optional 2 byte start register value

This routine builds the necessary CCW's to perform a Set Address Register and Read from the DACU and then calls the STARTIO subroutine to perform the I/O operation. The return code from the STARTIO subroutine is passed in the RC parameter. The routine performs the equivalent of a GAM GCTRL type IND (Set Program Function Indicators) and a GAM GREAD (Read from Graphic-Device Buffer).

If the optional STARTREG parameter is specified, then a Set Start Register, equivalent to a GAM GCTRL type STR (Set Buffer Address Register and Start), will also be performed by chaining the Start's CCW to the end of the CCW string.

DSTART - DACU Start:

Format: DSTART(STARTREG,RC,ADDRESS)

Where:

STARTREG = 2 byte start register value
RC = 4 byte return code
ADDRESS = Optional 4 byte DACU address of the start location for the Control Space Interpreter

This routine builds the necessary CCW to perform a Set Start Register and then calls the STARTIO subroutine to perform the I/O operation. The return code from the STARTIO subroutine is passed in the RC parameter. The routine performs the equivalent of a GAM GCTRL type STR (Set Buffer Address Register and Start).

If the optional ADDRESS parameter is specified, then a Set Address Register, equivalent to a GAM GCTRL type IND (Set Program Function Indicators), will also be performed by chaining the SAR's CCW to the beginning of the CCW string. Using this parameter allows starting a Control Space program at a location other that at X'FFFF0000'. The word offset to start the Control Space Interpreter is added to X'FFFF0000' and entered as the ADDRESS parameter. For example, if it is desired to start the Interpreter 12 bytes into the Control Space, then a X'FFFF000C' would be entered as the ADDRESS parameter. It should be noted here that wherever the Interpreter starts execution will be considered instruction 0 for branching instructions. See "Parallel I/O Application Programming Example" on page 152 for a sample of this subroutine's use.

DALARM - DACU Alarm:

Format: DALARM(RC)

Where:

RC = 4 byte return code

This routine calls the STARTIO subroutine to perform the I/O necessary to sound the audible alarm on the DACU. The return code from the STARTIO subroutine is passed in the RC parameter. This particular I/O operation causes a channel end (CE) to be sent separately from the device end (DE). This action causes the host to present a Channel Status Word (CSW) stored condition code (CC = 1) instead of a successful I/O initiation (CC = 0). If only the CE bit (bit 36) is set in the CSW, then the STARTIO subroutine will pass a zero return code. The DALARM routine performs the equivalent of a GAM GCTRL type ALM (Set Audible Alarm).

DATTNW - DACU Attention Wait:

Format: DATTNW(BUFFER,RC)

Where:

BUFFER = 12 byte array to be filled with the following attention data:

X'00'	X'00'	X'00'	X'03'
SENSE DATA			
X-POSITION REGISTER		Y-POSITION REGISTER	

The sense data is 4 sense bytes sent from the DACU.
RC = 4 byte return code

This routine will issue a WAIT macro for an Attention Event Control Block (ECB). When an attention is received from the device address specified in the DOPEN subroutine, the device interrupt handler (HANDLER) gains control and issues a POST macro on the attention ECB. When the ECB is posted, the DATTNW subroutine calls the STARTIO subroutine twice to perform a Sense and Read XY Position Registers (equivalent to a GAM GREADR type SEN - Read Sense Information, and a GAM GCTRL type XYP - Read XY Position Registers). The return code from the STARTIO subroutine for the Read XY Position is passed in the RC parameter. Upon exit, the BUFFER parameter is filled with Sense and XY Position data equivalent to data presented by a GAM ATTNINQ - Attention Inquiry.

DATTNI - DACU Attention Inquiry:

Format: DATTNI(BUFFER,RC)

Where:

BUFFER = 12 byte array to be filled with the following attention data:

X'00'	X'00'	X'00'	X'03'
SENSE DATA			
X-POSITION REGISTER		Y-POSITION REGISTER	

The sense data is 4 sense bytes sent from the DACU.
RC = 4 byte return code

This routine will look at the Attention Event Control Block (ECB) to determine if an attention has been received. When an attention is received from the device address specified in the DOPEN subroutine, the device interrupt handler (HANDLER) gains control, issues a POST macro on the Attention ECB, and returns control to previously executing code. The DATTNI subroutine is able to look at the attention ECB to determine if it has been posted. If the Event Complete bit (bit 1) is set, the DATTNI subroutine calls the STARTIO subroutine twice to perform a Sense and Read XY Position Registers (equivalent to a GAM GREADR

type SEN - Read Sense Information, and a GAM GCTRL type XYP - Read XY Position Registers). The return code from the STARTIO subroutine for the Read XY Position is passed in the RC parameter. Upon exit, the BUFFER parameter is filled with Sense and XY Position data equivalent to data presented by a GAM ATTNINQ - Attention Inquiry. If the Event Complete bit is not set, then the BUFFER and RC parameters are returned with all zero data and no Sense or Read XY Position Registers is performed.

Internal Subroutine Explanation

STARTIO - Start I/O subroutine: This routine issues the START I/O machine instruction (SIO) to initiate an I/O operation to the device address specified in the DOPEN subroutine. The user callable subroutine which calls this subroutine sets the following System/370 General Purpose Registers:

1 - Address of CCW string to be executed
6 - Address to save return code
8 - Return address to calling routine.

The STARTIO subroutine will perform the following:

1. Enter Supervisor State 0 and disable I/O interrupts.
2. Store the CCW address in the Channel Address Word (CAW) and issue the SIO to the device.
3. Save device status from the Program Status Word (PSW) and Channel Status Word (CSW).
4. Enable I/O interrupts and leave Supervisor State 0.
5. Check the Condition Code (CC) from the PSW and set the return code (RC). If CC = 0, then I/O initiated successfully.

   5A. Wait for an Event Control Block (ECB) to be posted by the interrupt handler when a Device End (DE) in received.

   5B. Check CSW for Unit Check bit (UC). If set then an error occurred. RC = the last 4 bytes of the CSW. If UC is not set then the I/O completed without error, RC = 0.

   5C. Return.

6. Determine if CC is 1, 2, or 3.
   If CC = 1, then the CSW is stored, RC = Last 4 bytes of CSW. If only the Channel End bit (CE) is set, wait for an ECB to be posted by the interrupt handler when a DE is received, RC = 0.
   If CC = 2, then the channel or subchannel is busy, RC = last 4 bytes of CSW with Busy bit set.
   If CC = 3, the channel is not operational, RC = X'FFFFFFFF'. This condition code occurs if the device address is not attached or the DACU is not enabled.
7. Return.

Summary of return codes passed back to user callable subroutines:

RC = 0.—Successful I/O completion.

RC = Last 4 bytes of CSW.—This occurs if the Start/IO instruction gets a CC = 1 (CSW stored) or CC = 2 (channel or subchannel busy), or a Unit Check is received indicating an execution error.

RC = X'FFFFFFFF'.—The channel is not operational.

See the *IBM System/370 Principles of Operations,* GA22-7000, for further information on the Start I/O instruction and the CSW.

HANDLER - Device Interrupt Handler subroutine: This routine is installed as the interrupt handler for a specific device using a SVC 202 (CMS HNDINT) in the DOPEN subroutine and removed using a HNDINT CLR in the DCLOSE subroutine. Once installed, this code gains control when any interrupt for the specified device address occurs. Upon entry, registers are set as follows:

0-1 I/O Old PSW
2-3 Channel Status Word (CSW)
4 Address of interrupting device
14 Return Address
15 Entry Point Address.

The CMS HNDINT routine, which calls this routine, performs all saving and restoring of registers. Once called, the interrupt handler will perform the following:

1. Check if the interrupt was a Device End (DE). If it was, then save the CSW, post the SIO ECB, and return to previously executing code.
2. If no DE found, check if the interrupt was an attention. If it was, then the Attention ECB is posted and control is returned to previously executing code.
3. If no attention found, return to previously executing code.

See the *IBM Virtual Machine/System Product CMS Command and Macro Reference,* SC19-6209, for further information on the HNDINT macro.

Parallel I/O Application Programming Example

This programming example contains a host sample program, an exec to run the host program, a local PC program, and results of execution. The order of execution is as follows:

1. Run 'RASMONn X'.
2. Start the DACU base code.
3. Return to DOS by pressing the R key.
4. Execute the local PC sample program.
5. Enable the DACU.
6. Attach to the IBM channel.
7. Execute the host exec to run the host sample program.

Host Sample Program

The following is an example of a host application program, written in VS FORTRAN Release 1.2, that uses HALLS to communicate with the DACU. The program gives an example of establishing and removing linkage to the device interrupt handler, writing to the Control Space and the Parallel I/O Buffer Space, reading from Buffer Space, starting the Control Space Interpreter and local PC code, waiting for an attention, reading sense bytes, and sounding the DACU audible alarm.

```
      PROGRAM SAMPLE
C
C  ********************************************************
C  * THIS PROGRAM PERFORMS THE FOLLOWING FUNCTIONS:       *
C  *   1) ESTABLISHES LINKAGE TO THE INTERRUPT HANDLER    *
C  *      ROUTINE TO FIELD DEVICE ENDS AND ATTENTIONS.    *
C  *      THE ATTACHED DEVICE ADDRESS IS 314.             *
C  *   2) SOUNDS THE DACU AUDIBLE ALARM.                  *
C  *   3) WRITES CONTROL SPACE PROGRAMS TO THE CONTROL    *
C  *      SPACE TO WRITE TO THE PARALLEL I/O CONTROL      *
C  *      REGISTER (UBCTL) AND READ THE PARALLEL I/O      *
C  *      STATUS REGISTER (UBSTAT), AND STARTS THE CONTROL *
C  *      SPACE INTERPRETER TO WRITE TO UBCTL.            *
C  *   4) STARTS THE CONTROL SPACE INTERPRETER TO READ    *
C  *      FROM UBSTAT.                                    *
C  *   5) READS THE VALUE OF UBSTAT STORED IN THE PARALLEL *
C  *      I/O BUFFER SPACE BY THE INTERPRETER.            *
C  *   6) INITIALIZES TWO 1K WORD BUFFERS.                *
C  *   7) WRITES A BUFFER TO THE BUFFER SPACE.            *
C  *   8) READS A BUFFER FROM THE BUFFER SPACE AND STARTS  *
C  *      LOCAL PC CODE TO SEND AN ATTENTION.             *
C  *   9) WAITS FOR THE ATTENTION.                        *
C  *  10) COMPARES BUFFER READ AND WRITE VALUES.          *
C  *  11) REMOVES THE INTERRUPT HANDLER.                  *
C  ********************************************************
C
      INTEGER*4 RC,CSADR1/ZFFFF0000/,CSADR2/ZFFFF000C/
      INTEGER*4 DEVAD/Z0314/
      INTEGER*2 ATTN(6),BUFIN(1024),BUFOUT(1024),CS(12)
      INTEGER*2 OKCNT,SSREG,UBSTAT
C
C********  INSTALL INTERRUPT HANDLER, SOUND ALARM.  *******
      CALL DOPEN(DEVAD)
      CALL DALARM(RC)
C
C********  WRITE CONTROL SPACE PROGRAMS TO THE     ********
C********  CONTROL SPACE TO WRITE TO UBCTL AND     ********
C********  READ FROM UBSTAT, AND START THE CONTROL ********
C********  SPACE INTERPRETER TO WRITE TO UBCTL.    ********
C
C          WUBCTL  XXXX   3FC0
C          END     XXXX   XXXX
C
C          RUBSTAT XXXX   0000
C          END     XXXX   XXXX
C
```

```
      DATA CS(1),  CS(2),  CS(3)  /Z0001,Z0000,Z3FC0/
      DATA CS(4),  CS(5),  CS(6)  /Z0000,Z0000,Z0000/
      DATA CS(7),  CS(8),  CS(9)  /Z0002,Z0000,Z0000/
      DATA CS(10), CS(11), CS(12) /Z0000,Z0000,Z0000/
      SSREG = 6
      CALL DWRITE(24,CS,CSADR1,RC,SSREG)
C
C********  START THE CONTROL SPACE INTERPRETER 12   ********
C********  BYTES INTO CONTROL SPACE TO READ UBSTAT. ********
      CALL DSTART(SSREG,RC,CSADR2)
C
C********  READ THE STATUS REGISTER VALUE.  ********
      CALL DREAD(2,UBSTAT,0,RC)
      WRITE(5,100) UBSTAT
 100  FORMAT(' PARALLEL I/O STATUS REGISTER = ',Z4)
C
C********  INITIALIZE INPUT AND OUTPUT BUFFERS.  ********
      DO 10 I = 1,1024
          BUFIN(I) = 0
          BUFOUT(I) = I
 10   CONTINUE
C
C********  WRITE THE OUTPUT BUFFER.  ********
      CALL DWRITE(2048,BUFOUT,0,RC)
C
C********  READ THE INPUT BUFFER AND START  ********
C********  LOCAL PC CODE TO SEND AN ATTENTION.  ********
      SSREG = 8
      CALL DREAD(2048,BUFIN,0,RC,SSREG)
C
C********  WAIT FOR THE ATTENTION.  ********
      CALL DATTNW(ATTN,RC)
C
C********  COMPARE THE THE TWO BUFFERS.  ********
      OKCNT = 0
      DO 20 I = 1,1024
         IF (BUFIN(I) .EQ. BUFOUT(I))THEN
            OKCNT = OKCNT+1
         END IF
 20   CONTINUE
      WRITE(5,110) OKCNT
      WRITE(5,120) ATTN
 110  FORMAT(' 1024 WORDS WRITE, READ, COMPARE.',I6,' OK.')
 120  FORMAT(' ATTENTION DATA = ',6Z5)
C
C********  REMOVE THE INTERRUPT HANDLER.  ********
      CALL DCLOSE
      STOP ' DONE.'
      END
```

Host Sample Exec

The following exec is run to execute the example host application program. The exec identifies the VS FORTRAN library VFORTLIB for use with the program, establishes that output data from the program be sent to the terminal, loads the application program and HALLS, and starts the application.

```
GLOBAL TXTLIB VFORTLIB
FILEDEF 5 TERMINAL
LOAD SAMPLE HALLS (START
```

Local Sample Program

The following is an example of local PC code run on the DACU. It is to be used in conjunction with the previously described host FORTRAN program. The local PC code displays an initial message, sets up a Parallel I/O start register entry point, and returns to PC base code. The start register entry point sends an attention to the host.

```
        PAGE    ,132
        .SALL
        TITLE SAMPLE: Local PC code for SAMPLE host program.
;
;MOVE MACRO
MOVE    MACRO   TO,FROM
        MOV     AX,FROM
        MOV     TO,AX
        ENDM
;INCLUDE PARALLEL I/O UTILITY AND GENERAL USAGE MACRO LIBS
IF1
        INCLUDE B:GENMAC.LIB
        INCLUDE B:UBUTIL.LIB
ENDIF
;
;******** DATA SEGMENT ********
;
SAMDS   SEGMENT PUBLIC 'DATA'
;INITIAL MESSAGE
ID      DB      'SAMPLE LOCAL CODE RUNNING',13,10,36
WNG9330 DB 'WUSR9330 UNABLE TO ENQUE ATTENTION ',13,10,36
        EVEN
;MAKE BRANCH TABLE SO DACU CODE CAN FIND ENTRY POINTS
ENTADDR DW      0,0             ;INTERRUPT ENTRY POINT
        DW      START,SAMCS     ;SET START REG ENTRY POINT
        DW      0,0             ;ATTENTION SENT ENTRY POINT
SAMDS   ENDS
;
;******** CODE SEGMENT ********
;
SAMCS   SEGMENT PUBLIC 'CODE'
        PUBLIC  SAMPLE
SAMPLE  PROC    FAR
        ASSUME  CS:SAMCS,DS:SAMDS
        MOVE    DS,SAMDS        ;SET LOCN OF DATA SEGMENT
        DISPLAY ID              ;WRITE ID MSG
        USETUP  ENTADDR         ;TELL DACU CODE ENTRY POINTS
        SLEEP                   ;RETURN TO DACU BASE CODE
;DONE FOR NOW. WAIT FOR CALL FROM DACU CODE
;
;******** SET START REGISTER ENTRY POINT ********
;
START:  MOV     AX,8804H        ;LOCAL PC CODE PRESENT, DEV 4
        MOV     DX,1234H        ;RECOGNIZABLE SENSE BYTE DATA
        HATTN   AX,DX           ;SEND ATTN TO HOST
        OR      AL,AL           ;CHECK RETURN CODE
        JZ      ATNRET          ;IF BAD DISPLAY WNG MSG
        PUSH    DS              ;SAVE DS
        MOVE    DS,SAMDS        ;SET DS TO SEG OF WNG MSG
        DISPLAY WNG9330         ;SEND WNG 9330
        POP     DS
ATNRET: RET
SAMPLE  ENDP
SAMCS   ENDS
;
```

```
;******************
;  STACK SEGMENT  *
;******************
SAMSS   SEGMENT STACK 'STACK' ;MAKE STACK 512 BYTES
        DB      64 DUP ('STACK   ')
SAMSS   ENDS
        END     SAMPLE
```

Sample Execution

When the previously mentioned example programs are executed, the following results are displayed at the terminal:

```
 SAMPLE
GLOBAL TXTLIB VFORTLIB
FILEDEF 5 TERMINAL
LOAD SAMPLE HALLS (START
EXECUTION BEGINS...
 PARALLEL I/O STATUS REGISTER = 0040
 1024 WORDS WRITE, READ, COMPARE.  1024 OK.
 ATTENTION DATA =  0000 0003 0088 1234 0000 0000
IFY002I STOP   DONE.
R;
```

Serial I/O Application Programming Example

This programming example contains a host sample program, an exec to run the host program, and results of execution. The order of execution is as follows:

1. Run 'RASMONn X'.
2. Start the DACU base code.
3. Enable the DACU.
4. Attach the IBM channel.
5. Execute the host exec to run the host sample program.

Host Sample Program

The following is an example of a host application program, written in VS FORTRAN Release 1.2, that uses the HALLS to communicate with the DACU. The program gives an example of establishing and removing linkage to the device interrupt handler, reading from Control Space, writing to Control Space and the Serial I/O Buffer Space, initializing the serial port, transmitting data from Buffer Space, waiting for an attention, reading sense bytes, and sounding the DACU audible alarm. It is designed to run with a 9600 baud even parity 7 bit ASCII terminal plugged into port 2 on the DACU. The ASCII characters 'This is a test message.', carriage return, line feed, EOT are transmitted to the ASCII terminal. If no device is attached to port 2, the data transmitted 'falls on the floor'.

```
      PROGRAM RS232C
C
C  ********************************************************
C  * THIS PROGRAM PERFORMS THE FOLLOWING FUNCTIONS:       *
C  *   1) ESTABLISHES LINKAGE TO THE INTERRUPT HANDLER    *
C  *      ROUTINE TO FIELD DEVICE ENDS AND ATTENTIONS.    *
C  *      THE ATTACHED DEVICE ADDRESS IS 312.             *
C  *   2) SOUNDS THE DACU AUDIBLE ALARM.                  *
C  *   3) READS FROM CONTROL SPACE TO RETRIEVE DEFAULT    *
C  *      SERIAL I/O CONTROL PARAMETERS.                  *
C  *   4) CHANGES BYTES 12 AND 13 (ORIGIN 0) OF THE       *
```

```
C  *       CONTROL PARAMETERS TO SET TO IGNORE MODE,          *
C  *       XON/XOFF PROTOCOL.                                 *
C  *    5) WRITES TO CONTROL SPACE TO STORE MODIFIED          *
C  *       CONTROL PARAMETERS AND ISSUES A START TO           *
C  *       INITIALIZE THE SERIAL PORT WITH THE PARAMETERS.    *
C  *    6) WRITES EBCDIC CHARACTERS TO THE SERIAL I/O         *
C  *       BUFFER SPACE.                                      *
C  *    7) STARTS DATA TRANSFER TO THE SERIAL PORT.           *
C  *    8) WAITS FOR AN ATTENTION INDICATING THE TRANSFER     *
C  *       IS COMPLETE.                                       *
C  *    9) COMPARES NUMBER OF BYTES TRANSFERRED WITH THE      *
C  *       NUMBER OF BYTES SENT TO THE DACU.                  *
C  *   10) REMOVES THE INTERRUPT HANDLER.                     *
C  ***********************************************************
C
       INTEGER*4 RC,CSADDR /ZFFFF0000/,DEVAD /Z0312/
       INTEGER*2 ATTN(6),CS(11),OKCNT,SSREG,UBSTAT
       LOGICAL*1 BUFOUT(26) /'T','H','I','S',' ',
     * 'I','S',' ','A',' ','T','E','S','T',' ','M','E','S',
     * 'S','A','G','E','.',' ',' ',' '/
       LOGICAL*1 CR /Z0D/, LF /Z25/, EOT /Z37/
C
C********  INSTALL INTERRUPT HANDLER, SOUND ALARM.  ********
       CALL DOPEN(DEVAD)
       CALL DALARM(RC)
C
C********  READ CONTROL SPACE TO RETRIEVE CONTROL PARMS. ***
C
       CALL DREAD(22,CS,CSADDR,RC)
C
C********  SET CS(7) TO 258 = 1*256 (HI BYTE - XON/XOFF) ***
C********  + 2 (LO BYTE - IGNORE MODE), WRITE PARMS BACK ***
C********  TO CONTROL SPACE, AND INITIALIZE THE SERIAL   ***
C********  PORT WITH THE MODIFIED PARMS.                 ***
C
       CS(7) = 258
       SSREG = 6
       CALL DWRITE(22,CS,CSADDR,RC,SSREG)
C
C********  WRITE THE OUTPUT BUFFER.  ********
       BUFOUT(24) = CR
       BUFOUT(25) = LF
       BUFOUT(26) = EOT
       CALL DWRITE(26,BUFOUT,0,RC)
C
C********  START TRANSFER TO THE SERIAL PORT.  ********
       SSREG = 4
       CALL DSTART(SSREG,RC)
C
C********  WAIT FOR THE ATTENTION.  ********
       CALL DATTNW(ATTN,RC)
C
C********  COMPARE # BYTES SENT WITH # BYTES TRANSFERRED.
C********  THE SENSE BYTES CONTAIN # BYTES TRANSFERRED.
       WRITE(5,100) ATTN(4)
 100   FORMAT(' 26 BYTES SENT TO BUFFER SPACE.',I6,
     *        ' TRANSFERRED.')
C
C********  REMOVE THE INTERRUPT HANDLER  ********
       CALL DCLOSE
       STOP ' DONE.'
       END
```

Host Sample Exec

The following exec is run to execute the example host application program. The exec identifies the VS FORTRAN library VFORTLIB for use with the program, establishes that output data from the program be sent to the terminal, loads the application program and the HALLS, and starts the application.

```
GLOBAL TXTLIB VFORTLIB
FILEDEF 5 TERMINAL
LOAD RS232C HALLS (START
```

Sample Execution

When the previously mentioned example programs are executed, the following results are displayed at the terminal:

```
 RS232C
GLOBAL TXTLIB VFORTLIB
FILEDEF 5 TERMINAL
LOAD RS232C HALLS (START
EXECUTION BEGINS...
 26 BYTES SENT TO BUFFER SPACE.     26 BYTES TRANSFERRED.
IFY002I STOP   DONE.
R;
```

Source Code

The following section contains the source code for the eight user callable and two internal subroutines of the Host Application Language Library using Start I/O (HALLS). The code is entered as a single file and assembled into object code. The object code may then be linked to application object code to perform the operations previously described.

```
         MACRO
&LABEL   ENTER
.*-------------------------------------------------------------
.*       "ENTER" GENERATES THE STANDARD ENTRY CODE
.*-------------------------------------------------------------
&LABEL   STM   14,12,12(13)             SAVE REGISTERS
         ENTRY &LABEL
         USING &LABEL,15                USE R15 FOR A MINUTE
         L     12,=A(&SYSECT)           GET ADDR OF CSECT
         DROP  15
         USING &SYSECT,12
         LA    15,REGSAVE
         ST    13,4(,15)                FORWARD CHAIN
         ST    15,8(,13)                ...AND BACK
         LR    13,15                    POINT TO THE NEW ONE
         B     *+12                     SKIP TO THE START
         DC    CL8'&LABEL'
         MEND
         MACRO
&LABEL   LEAVE &RC=0
.*-------------------------------------------------------------
.*   "LEAVE" GENERATES THE STANDARD EXIT CODE
.*-------------------------------------------------------------
         AIF   ('&RC'(1,1) EQ '(').REG
&LABEL   LA    15,&RC                   GET RETURN CODE,
         AGO   .CHECK
.REG     ANOP
```

```
&LABEL   LR     15,&RC(1)              SAVE THE RETURN CODE
.CHECK   L      13,4(,13)              GET THE SAVE AREA
         L      14,12(,13)             PICK UP RETURN ADDR
         LM     0,12,20(13)            RELOAD THE REGS
         BR     14
         MEND
HALLS    CSECT
         TITLE  'DACU OPEN ROUTINE.'
*************************************************************
*                                                           *
*                      DACU  OPEN                           *
*                                                           *
* SUBROUTINE DOPEN(DEVAD)                                   *
*   WHERE:                                                  *
*     DEVAD = 4 BYTE DEVICE ADDRESS TO ISSUE I/O TO.        *
*                                                           *
* THE FOLLOWING CODE WILL:                                  *
*   1. SAVE THE REGISTERS FROM THE CALLING ROUTINE.         *
*   2. ESTABLISH LINKAGE TO THE INTERRUPT HANDLER           *
*      ROUTINE WHICH WILL POST AN ECB WHEN A DEVICE END     *
*      OR ATTENTION IS RECEIVED.                            *
*   3. RESTORE REGISTERS AND RETURN.                        *
*                                                           *
*************************************************************
*
*     SAVE REGISTERS OF CALLING ROUTINE
*
DOPEN    ENTER
*
*     ESTABLISH INTERRUPT HANDLER LINKAGE
*
         USING NUCON,R0
         L     R2,0(R1)           GET ADDR OF DEVICE ADDRESS
         L     R2,0(R2)           GET DEVICE ADDRESS
         STH   R2,DEVAD           SAVE DEV ADDR FOR HNDINT
         BAL   R1,INSTALL         SETUP HNDINT DATA LINKAGE
         DC    CL8'HNDINT'        DATA FOR HNDINT. DON'T USE
         DC    CL4'SET'           HNDINT MACRO, JUST IT'S
         DC    CL4'DACU'          SVC 202
         DC    AL4(HANDLER)
DEVAD    DC    H'0'
         DC    CL1'ASAP'
         DC    C'C'
         DC    XL4'FFFFFFFF'
INSTALL  SVC   202                INSTALL INTERRUPT HANDLER
         DC    AL4(*+4)           USING CMS HNDINT SVC
*
*     RESTORE REGISTERS AND RETURN
*
         LEAVE RC=0
         TITLE 'DACU CLOSE ROUTINE.'
*************************************************************
*                                                           *
*                      DACU  CLOSE                          *
*                                                           *
* SUBROUTINE DCLOSE                                         *
*                                                           *
* THE FOLLOWING CODE WILL:                                  *
*   1. SAVE THE REGISTERS FROM THE CALLING ROUTINE.         *
*   2. RESTORE THE CMS I/O INTERRUPT HANDLER.               *
*   3. RESTORE REGISTERS AND RETURN.                        *
*                                                           *
```

```
*************************************************************
*
*     SAVE REGISTERS OF CALLING ROUTINE
*
DCLOSE   ENTER
*
*    RESTORE THE CMS I/O INTERRUPT HANDLER
*
         HNDINT CLR,DACU          DON'T TRAP DEVICE INTRS
*
*     RESTORE REGISTERS AND RETURN
*
         LEAVE RC=0
         TITLE 'DACU WRITE ROUTINE.'
*************************************************************
*                                                           *
*                      DACU  WRITE                          *
*                                                           *
* SUBROUTINE DWRITE(LENGTH,DATA,ADDRESS,RC,STARTREG)        *
*   WHERE:                                                  *
*      LENGTH = 4 BYTE LENGTH OF DATA IN BYTES.             *
*      DATA = ARRAY OF DATA TO WRITE.                       *
*      ADDRESS = 4 BYTE DACU ADDRESS TO WRITE DATA.         *
*      RC = 4 BYTE RETURN CODE FROM STARTIO SUBROUTINE.     *
*      STARTREG = OPTIONAL 2 BYTE START REGISTER VALUE.     *
*                                                           *
* THE FOLLOWING CODE WILL:                                  *
*   1. SAVE THE REGISTERS FROM THE CALLING ROUTINE.         *
*   2. READ PARAMETERS FROM THE CALLING ROUTINE AND USE     *
*      THEM TO BUILD CCW'S FOR A SET ADDRESS REGISTER       *
*      AND WRITE CHANNEL PROGRAM EQUIVALENT TO A GAM        *
*      GCTRL TYPE IND - SET PROGRAM FUNCTION INDICATORS,    *
*      AND A GAM GWRITE - WRITE TO GRAPHIC-DEVICE BUFFER.   *
*      IF THE OPTIONAL STARTREG PARAMETER IS SPECIFIED,     *
*      THEN A START, EQUIVALENT TO A GAM GCTRL TYPE STR     *
*      - SET BUFFER ADDRESS REGISTER AND START, WILL BE     *
*      CHAINED TO THE END OF THE CCW STRING. PARMS          *
*      PASSED ARE ADDRESSES.                                *
*      REGISTER 1 POINTS TO THE PARAMETER LIST.             *
*   3. CALL START I/O SUBROUTINE WITH:                      *
*          R1 - POINTER TO CCW STRING                       *
*          R6 - ADDRESS TO SAVE RETURN CODE                 *
*          R8 - RETURN ADDRESS                              *
*   4. RESTORE REGISTERS AND RETURN.                        *
*                                                           *
*************************************************************
*
*     SAVE REGISTERS OF CALLING ROUTINE
*
DWRITE   ENTER
*
*     READ PARM DATA AND BUILD CCW'S
*
         SLR   R2,R2              ZERO WORK REGISTER
         L     R2,0(R1)           LOAD ADDR OF DATA LENGTH
         L     R2,0(R2)           LOAD DATA LENGTH
         STH   R2,WCCW+6          PUT LENGTH INTO CCW
         L     R2,4(R1)           LOAD ADDRESS OF DATA
         STCM  R2,B'0111',WCCW+1  PUT DATA ADDRESS INTO CCW
         L     R2,8(R1)           LOAD ADDR OF DACU ADDR REG
```

```
         L     R2,0(R2)           LOAD DACU ADDRESS REG
         ST    R2,WSARAD          PUT ADDR REG INTO SAR FLD
         L     R6,12(R1)          LOAD RETURN CODE ADDRESS
         MVI   WCCW+4,X'20'       SET SUPPRESSION OF
*                                 POSSIBLE INCORRECT LENGTH
*                                 INDICATION ON
*
*     TEST FOR START REGISTER PARAMETER
*
         TM    12(R1),X'80'       IS RETURN CODE LAST PARM
         BO    WSIO               IF LAST, DON'T CHAIN START
         MVI   WCCW+4,X'60'       CHAIN START CCW
         L     R2,16(R1)          LOAD ADDR OF START REG
         ICM   R2,B'0011',0(R2)   LOAD START REG VALUE
         STH   R2,WSTRTREG        PUT VALUE INTO CCW
*
*     CALL START I/O ROUTINE WITH REGISTERS SET
*
WSIO     LA    R1,WDACU           LOAD CCW STRING POINTER
         BAL   R8,STARTIO         CALL START I/O ROUTINE
*
*     RESTORE REGISTERS AND RETURN
*
         LEAVE RC=0
*
         DS    0D
WDACU    EQU   *
WSARCCW  DC    X'1B',AL3(WSARAD),X'40',X'0',XL2'4'
WCCW     DC    X'01',3X'0',X'20',X'0',XL2'0'
WSTRTCCW DC    X'27',AL3(WSTRTREG),X'00',XL3'2'
WSARAD   DC    F'0'
WSTRTREG DC    H'0'
         TITLE 'DACU READ ROUTINE.'
*************************************************************
*                                                           *
*                        DACU  READ                         *
*                                                           *
* SUBROUTINE DREAD(LENGTH,BUFFER,ADDRESS,RC,STARTREG)       *
*   WHERE:                                                  *
*      LENGTH = 4 BYTE LENGTH OF DATA TO READ IN BYTES.     *
*      BUFFER = ARRAY TO READ DATA INTO.                    *
*      ADDRESS = 4 BYTE DACU ADDRESS TO READ DATA FROM.     *
*      RC = 4 BYTE RETURN CODE FROM STARTIO SUBROUTINE.     *
*      STARTREG = OPTIONAL 2 BYTE START REGISTER VALUE.     *
*                                                           *
* THE FOLLOWING CODE WILL:                                  *
*   1. SAVE THE REGISTERS FROM THE CALLING ROUTINE.         *
*   2. READ PARAMETERS FROM THE CALLING ROUTINE AND USE     *
*      THEM TO BUILD CCW'S FOR A SET ADDRESS REGISTER AND   *
*      READ CHANNEL PROGRAM EQUIVALENT TO A GAM GCTRL TYPE  *
*      IND - SET PROGRAM FUNCTION INDICATORS, AND A GAM     *
*      GREAD - READ FROM GRAPHIC-DEVICE BUFFER.             *
*      IF THE OPTIONAL STARTREG PARAMETER IS SPECIFIED,     *
*      THEN A START, EQUIVALENT TO A GAM GCTRL TYPE STR -   *
*      SET BUFFER ADDRESS REGISTER AND START, WILL CHAINED  *
*      TO THE END OF THE CCW STRING. PARMS PASSED ARE       *
*      ADDRESSES.                                           *
*      REGISTER 1 POINTS TO THE PARAMETER LIST.             *
*   3. CALL START I/O SUBROUTINE WITH:                      *
*         R1 - POINTER TO CCW STRING                        *
*         R6 - ADDRESS TO SAVE RETURN CODE                  *
*         R8 - RETURN ADDRESS                               *
*   4. RESTORE REGISTERS AND RETURN.                        *
*                                                           *
*************************************************************
*
```

```
*     SAVE REGISTERS OF CALLING ROUTINE
*
DREAD    ENTER
*
*     READ PARM DATA AND BUILD CCW'S
*
         SLR   R2,R2              ZERO WORK REGISTER
         L     R2,0(R1)           LOAD ADDR OF DATA LENGTH
         L     R2,0(R2)           LOAD DATA LENGTH TO READ
         STH   R2,RCCW+6          PUT LENGTH INTO CCW
         L     R2,4(R1)           LOAD ADDRESS OF BUFFER
         STCM  R2,B'0111',RCCW+1  PUT BUFFER ADDR INTO CCW
         L     R2,8(R1)           LOAD ADDR OF DACU ADDR REG
         L     R2,0(R2)           LOAD DACU ADDRESS REG
         ST    R2,RSARAD          PUT ADDR REG INTO SAR FLD
         L     R6,12(R1)          LOAD RETURN CODE ADDRESS
         MVI   RCCW+4,X'20'       SET SUPPRESSION OF
*                                  POSSIBLE INCORRECT LENGTH
*                                  INDICATION ON
*
*     TEST FOR START REGISTER PARAMETER
*
         TM    12(R1),X'80'       IS RETURN CODE LAST PARM
         BO    RSIO               IF LAST, DON'T CHAIN START
         MVI   RCCW+4,X'60'       CHAIN START CCW
         L     R2,16(R1)          LOAD ADDR OF START REG
         ICM   R2,B'0011',0(R2)   LOAD START REG VALUE
         STH   R2,RSTRTREG        PUT VALUE INTO CCW
*
*     CALL START I/O ROUTINE WITH REGISTERS SET
*
RSIO     LA    R1,RDACU           LOAD CCW STRING POINTER
         BAL   R8,STARTIO         CALL START I/O ROUTINE
*
*     RESTORE REGISTERS AND RETURN
*
         LEAVE RC=0
*
         DS    0D
RDACU    EQU   *
RSARCCW  DC    X'1B',AL3(RSARAD),X'40',XL3'4'
RCCW     DC    X'02',AL3(0),X'20',XL3'0'
RSTRTCCW DC    X'27',AL3(RSTRTREG),X'00',XL3'2'
RSARAD   DC    F'0'
RSTRTREG DC    H'0'
         TITLE 'DACU START ROUTINE.'
*************************************************************
*                                                           *
*                      DACU  START                          *
*                                                           *
* SUBROUTINE DSTART(STARTREG,RC,ADDRESS)                    *
*   WHERE:                                                  *
*      STARTREG = 2 BYTE START REGISTER VALUE.              *
*      RC = 4 BYTE RETURN CODE FROM STARTIO SUBROUTINE.     *
*      ADDRESS = OPTIONAL 4 BYTE DACU ADDRESS OF START      *
*                LOCATION IN CONTROL SPACE.                 *
*                                                           *
* THE FOLLOWING CODE WILL:                                  *
*   1. SAVE THE REGISTERS FROM THE CALLING ROUTINE.         *
*   2. READ A PARAMETER FROM THE CALLING ROUTINE AND USE    *
*      IT TO BUILD A CCW FOR A SET START REGISTER CHANNEL   *
*      PROGRAM EQUIVALENT TO A GAM GCTRL TYPE STR - SET     *
*      BUFFER ADDRESS AND START.                            *
*      IF THE OPTIONAL ADDRESS PARAMETER IS SPECIFIED,      *
*      THEN A SET ADDRESS REGISTER, EQUIVALENT TO A GAM     *
```

```
*       GCTRL TYPE IND - SET PROGRAM FUNCTION INDICATORS,    *
*       WILL BE CHAINED TO THE FRONT OF THE CCW STRING.      *
*       PARMS ARE PASSED AS ADDRESSES.                       *
*       REGISTER 1 POINTS TO THE PARAMETER LIST.             *
*    3. CALL START I/O SUBROUTINE WITH:                      *
*         R1 - POINTER TO CCW STRING                         *
*         R6 - ADDRESS TO SAVE RETURN CODE                   *
*         R8 - RETURN ADDRESS                                *
*    4. RESTORE REGISTERS AND RETURN.                        *
*                                                            *
**************************************************************
*
*      SAVE REGISTERS OF CALLING ROUTINE
*
DSTART   ENTER
*
*      READ PARM DATA AND BUILD CCW
*
         SLR    R2,R2              ZERO WORK REGISTER
         L      R2,0(R1)           LOAD ADDR OF START REG
         ICM    R2,B'0011',0(R2)   LOAD START REG VALUE
         STH    R2,STRTREG         PUT VALUE INTO CCW
         L      R6,4(R1)           LOAD RETURN CODE ADDRESS
*
*      TEST FOR ADDRESS PARAMETER
*
         TM     4(R1),X'80'        IS RETURN CODE LAST PARM
         BO     SSRSIO             IF LAST, DON'T CHAIN SAR
         L      R2,8(R1)           LOAD ADDR OF ADDRESS REG
         L      R2,0(R2)           LOAD DACU ADDRESS REG
         ST     R2,SSARAD          PUT ADDR REG IN SAR FLD
*
*      CALL START I/O ROUTINE WITH REGISTERS SET TO PERFORM
*      SET ADDRESS REGISTER AND SET START REGISTER.
*
         LA     R1,SSARCCW         LOAD CCW STRING POINTER
         BAL    R8,STARTIO         CALL START I/O ROUTINE
*
*      RESTORE REGISTERS AND RETURN
*
         LEAVE RC=0
*
*      CALL START I/O ROUTINE WITH REGISTERS SET
*      TO PERFORM A SET START REGISTER.
*
SSRSIO   LA     R1,STRTCCW         LOAD CCW STRING POINTER
         BAL    R8,STARTIO         CALL START I/O ROUTINE
*
*      RESTORE REGISTERS AND RETURN
*
         LEAVE RC=0
*
         DS     0D
SSARCCW  EQU    *
         DC     X'1B',AL3(SSARAD),X'40',X'0',XL2'4'
STRTCCW  EQU    *
         DC     X'27',AL3(STRTREG),X'00',XL3'2'
SSARAD   DC     F'0'
STRTREG  DC     H'0'
         TITLE 'DACU AUDIBLE ALARM ROUTINE.'
**************************************************************
*                                                            *
*                      DACU  ALARM                           *
*                                                            *
* SUBROUTINE DALARM(RC)                                      *
*   WHERE:                                                   *
```

```
*      RC = 4 BYTE RETURN CODE FROM STARTIO SUBROUTINE.     *
*                                                           *
* THE FOLLOWING CODE WILL:                                  *
*   1. SAVE THE REGISTERS FROM THE CALLING ROUTINE.         *
*   2. CALL START I/O SUBROUTINE WITH:                      *
*         R1 - POINTER TO CCW STRING                        *
*         R6 - ADDRESS TO SAVE RETURN CODE                  *
*         R8 - RETURN ADDRESS                               *
*      A CHANNEL PROGRAM EQUIVALENT TO A GAM GCTLR TYPE     *
*      ALM - SET AUDIBLE ALARM WILL BE PERFORMED USING A    *
*      CCW WHICH NEEDS NO PARM DATA. THE START I/O CODE     *
*      WILL GET A CC = 1 (CSW STORED) WITH CHANNEL END      *
*      SET, WHICH IS OK, AND WAIT FOR THE SPLIT DEVICE END.*
*   3. RESTORE REGISTERS AND RETURN.                        *
*                                                           *
*************************************************************
*
*     SAVE REGISTERS OF CALLING ROUTINE
*
DALARM   ENTER
         SLR   R2,R2              ZERO WORK REGISTER
         L     R6,0(R1)           LOAD RETURN CODE ADDRESS
*
*     CALL START I/O ROUTINE WITH REGISTERS SET
*
         LA    R1,ALRMCCW         LOAD CCW STRING POINTER
         BAL   R8,STARTIO         CALL START I/O ROUTINE
*
*     RESTORE REGISTERS AND RETURN
*
         LEAVE RC=0
*
         DS    0D
ALRMCCW  EQU   *
         DC    X'0B',AL3(0),X'20',XL3'2'
         TITLE 'DACU ATTENTION WAIT ROUTINE.'
*************************************************************
*                                                           *
*                   DACU ATTENTION WAIT                     *
*                                                           *
* SUBROUTINE DATTNW(BUFFER,RC)                              *
*   WHERE:                                                  *
*      BUFFER = 12 BYTE ARRAY TO BE FILLED WITH THE         *
*         FOLLOWING ATTENTION DATA:                         *
*         -----------------------------------------------   *
*         |   X'00'  |   X'00'  |   X'00'  |   X'03'  |    *
*         -----------------------------------------------   *
*         |                SENSE DATA                 |    *
*         -----------------------------------------------   *
*         | X-POSITION REGISTER | Y-POSITION REGISTER |     *
*         -----------------------------------------------   *
*         THE SENSE DATA IS 4 SENSE BYTES FROM THE DACU.    *
*         THE XY POSITION REGISTER DATA IS RETRIEVED FOR    *
*         CONSISTENCY WITH GAM.                             *
*      RC = 4 BYTE RETURN CODE FROM STARTIO SUBROUTINE.     *
*                                                           *
* THE FOLLOWING CODE WILL:                                  *
*   1. SAVE THE REGISTERS FROM THE CALLING ROUTINE.         *
*   2. READ A BUFFER ADDRESS FROM THE CALLING ROUTINE.      *
*      PARMS PASSED ARE ADDRESSES.                          *
*      REGISTER 1 POINTS TO THE PARAMETER LIST.             *
*   3. PERFORM A WAIT ON THE ATTENTION ECB. THIS ECB WILL   *
*      BE POSTED BY THE INTERRUPT HANDLER WHEN AN           *
*      ATTENTION FOR DEVICE SPECIFIED IN THE OPEN OCCURS.   *
*   4. STORE ATTENTION TYPE (ALWAYS 3 - LIGHT PEN ATTN)     *
```

```
*       IN THE CALLING ROUTINE'S BUFFER.                        *
*       STORE BUFFER ADDRESSES FOR SENSE AND XYP DATA IN        *
*       THE SENSE AND XYP CCW STRINGS.                          *
*    5. EXECUTE CCW STRINGS TO PERFORM A SENSE AND READ XY      *
*       POSITION REGISTERS BY CALLING THE START I/O             *
*       SUBROUTINE TWICE WITH:                                  *
*           R1 - POINTER TO CCW STRING                          *
*           R6 - ADDRESS TO SAVE RETURN CODE                    *
*           R8 - RETURN ADDRESS                                 *
*       CHANNEL PROGRAMS EQUIVALENT TO A GAM GREADR TYPE        *
*       SEN - READ SENSE INFORMATION, AND A GREADR TYPE XYP     *
*       - READ XY POSITION REGISTERS WILL BE PERFORMED.         *
*    6. RESTORE REGISTERS AND RETURN.                           *
*                                                               *
*****************************************************************
*
*      SAVE REGISTERS OF CALLING ROUTINE
*
DATTNW   ENTER
*
*      READ PARM DATA
*
         L     R5,0(R1)            LOAD ADDRESS OF BUFFER
         L     R6,4(R1)            LOAD RETURN CODE ADDRESS
*
*      WAIT FOR AN ATTENTION
*
         WAIT  ECB=ATTNECB         WAIT FOR THE ATTENTION
         NI    ATTNECB,X'00'       TURN OFF ECB COMPLETE BIT
*
*      STORE ATTENTION TYPE, SET ADDRS FOR CCW'S TO SAVE DATA
*
         L     R2,=X'00000003'     ATTN TYPE = 3 (LIGHT PEN)
         ST    R2,0(R5)            STORE ATTN TYPE IN BUFFER
         A     R5,=F'4'            SET ADDR FOR SENSE DATA
         STCM  R5,B'0111',SENSECCW+1 PUT ADDR IN CCW STRING
         A     R5,=F'4'            SET ADDR FOR XYP DATA
         STCM  R5,B'0111',XYPCCW+1 PUT ADDR IN CCW STRING
*
*      CALL START I/O ROUTINE TWICE WITH REGISTERS SET
*
         LA    R1,SENSECCW         LOAD SENSE CCW ADDRESS
         BAL   R8,STARTIO          CALL START I/O ROUTINE
         LA    R1,XYPCCW           LOAD READ XYP CCW ADDRESS
         BAL   R8,STARTIO          CALL START I/O ROUTINE
*
*     RESTORE REGISTERS AND RETURN
*
         LEAVE RC=0
*
         DS    0D
SENSECCW EQU *
         DC    X'04',AL3(0),X'00',XL3'4'
XYPCCW   EQU *
         DC    X'12',AL3(0),X'00',XL3'4'
         TITLE 'DACU ATTENTION INQUIRY ROUTINE.'
*****************************************************************
*                                                               *
*                    DACU ATTENTION INQUIRY                     *
*                                                               *
* SUBROUTINE DATTNI(BUFFER,RC)                                  *
*   WHERE:                                                      *
*      BUFFER = 12 BYTE ARRAY TO BE FILLED WITH THE             *
*         FOLLOWING ATTENTION DATA:                             *
```

```
*         ---------------------------------------------    *
*         |   X'00'  |   X'00'  |   X'00'  |   X'03'  |    *
*         ---------------------------------------------    *
*         |                SENSE DATA                 |    *
*         ---------------------------------------------    *
*         | X-POSITION REGISTER | Y-POSITION REGISTER |    *
*         ---------------------------------------------    *
*         THE SENSE DATA IS 4 SENSE BYTES FROM THE DACU.   *
*         THE XY POSITION REGISTER DATA IS RETRIEVED FOR   *
*         CONSISTENCY WITH GAM.                            *
*      RC = 4 BYTE RETURN CODE FROM STARTIO SUBROUTINE.    *
*                                                          *
* THE FOLLOWING CODE WILL:                                 *
*   1. SAVE THE REGISTERS FROM THE CALLING ROUTINE.        *
*   2. READ A BUFFER ADDRESS FROM THE CALLING ROUTINE.     *
*      PARMS PASSED ARE ADDRESSES.                         *
*      REGISTER 1 POINTS TO THE PARAMETER LIST.            *
*   3. CLEAR CALLER'S ATTENTION BUFFER, RETURN CODE.       *
*   4. LOOK AT THE ATTENTION ECB TO CHECK FOR AN ATTN.     *
*                                                          *
*      IF AN ATTENTION OCCURRED:                           *
*   5. STORE ATTENTION TYPE (ALWAYS 3 - LIGHT PEN ATTN)    *
*      IN THE CALLING ROUTINE'S BUFFER.                    *
*      STORE BUFFER ADDRESSES FOR SENSE AND XYP DATA IN    *
*      THE SENSE AND XYP CCW STRINGS.                      *
*   6. EXECUTE CCW STRINGS TO PERFORM A SENSE AND READ XY  *
*      POSITION REGISTERS BY CALLING THE START I/O         *
*      SUBROUTINE TWICE WITH:                              *
*          R1 - POINTER TO CCW STRING                      *
*          R6 - ADDRESS TO SAVE RETURN CODE                *
*          R8 - RETURN ADDRESS                             *
*      CHANNEL PROGRAMS EQUIVALENT TO A GAM GREADR TYPE    *
*      SEN - READ SENSE INFORMATION, AND A GREADR TYPE XYP *
*      - READ XY POSITION REGISTERS WILL BE PERFORMED.     *
*                                                          *
*        IF NO ATTENTION OCCURRED:                         *
*          THE BUFFER IS LEFT CLEAR, NO SENSE OR READ XYP  *
*          IS PERFORMED.                                   *
*   7. RESTORE REGISTERS AND RETURN.                       *
*                                                          *
************************************************************
*
*     SAVE REGISTERS OF CALLING ROUTINE
*
DATTNI   ENTER
*
*     READ PARM DATA
*
         L     R5,0(R1)           LOAD ADDRESS OF BUFFER
         L     R6,4(R1)           LOAD RETURN CODE ADDRESS
*
*     CLEAR RETURN CODE, ATTENTION BUFFER.
*
         SLR   R2,R2              ZERO WORK REGISTER
         ST    R2,0(R6)           SET RETURN CODE TO 0
         MVC   0(12,R5),ZERO12    SET ATTENTION BUFFER TO 0
*
*     LOOK AT THE ECB TO CHECK FOR AN ATTENTION
*
         TM    ATTNECB,X'40'      LOOK AT ECB EVENT COMPLETE
         BZ    EXITATTN           IF NOT SET, NO ATTENTION
         NI    ATTNECB,X'00'      TURN OFF ECB COMPLETE BIT
*
*     STORE ATTN TYPE, SET ADDRESSES FOR CCW'S TO SAVE DATA
*
```

```
         L     R2,=X'00000003'    ATTN TYPE = 3 (LIGHT PEN)
         ST    R2,0(R5)           STORE ATTN TYPE IN BUFFER
         A     R5,=F'4'           SET ADDR FOR SENSE DATA
         STCM  R5,B'0111',SENSECCW+1 PUT ADDR IN CCW STRING
         A     R5,=F'4'           SET ADDR FOR XYP DATA
         STCM  R5,B'0111',XYPCCW+1 PUT ADDR IN CCW STRING
*
*     CALL START I/O ROUTINE TWICE WITH REGISTERS SET
*
         LA    R1,SENSECCW        LOAD SENSE CCW ADDRESS
         BAL   R8,STARTIO         CALL START I/O ROUTINE
         LA    R1,XYPCCW          LOAD READ XYP CCW ADDRESS
         BAL   R8,STARTIO         CALL START I/O ROUTINE
*
*     RESTORE REGISTERS AND RETURN
*
EXITATTN LEAVE RC=0
         TITLE 'START I/O SUBROUTINE.'
*************************************************************
*                                                           *
*                   START I/O SUBROUTINE                    *
*                                                           *
* THE FOLLOWING CODE WILL:                                  *
*   1. ENTER SUPERVISOR STATE 0 (DANGER ZONE) AND DISABLE   *
*      I/O INTERRUPTS.                                      *
*   2. STORE THE CCW ADDRESS IN THE CAW AND ISSUE A SIO TO  *
*      THE DEVICE.                                          *
*   3. SAVE DEVICE STATUS FROM PSW AND CSW.                 *
*   4. ENABLE I/O INTERRUPTS AND LEAVE SUPERVISOR STATE 0.  *
*   5. CHECK THE CONDITION CODE FROM THE START I/O.         *
*      IF CC = 0, SUCCESSFUL I/O INITIATION.                *
*      5A. WAIT FOR AN ECB. THE ECB WILL BE POSTED BY THE   *
*          INTERRUPT HANDLER WHEN A DEVICE END IS RECEIVED.*
*      5B. CHECK CSW FOR UNIT CHECK BIT. IF SET THEN AN     *
*          ERROR OCCURRED, RC = LAST 4 BYTES OF CSW. IF     *
*          IF NOT SET, THEN THE I/O COMPLETED WITHOUT       *
*          ERROR, RC = 0.                                   *
*      5C. RETURN.                                          *
*   6. DETERMINE IF CC IS 1, 2, OR 3.                       *
*      IF CC = 1, CSW STORED, RC = LAST 4 BYTES OF CSW.     *
*         IF CHANNEL END ALONE IS SET, WAIT FOR AN ECB TO   *
*         BE POSTED BY THE INTERRUPT HANDLER WHEN A DEVICE  *
*         END IS RECEIVED. RC = 0.                          *
*      IF CC = 2, CHANNEL OR SUBCHANNEL BUSY, RC = LAST 4   *
*         BYTES OF CSW WITH BUSY BIT SET.                   *
*      IF CC = 3, CHANNEL NOT OPERATIONAL, RC = X'FFFFFFFF'*
*   7. RETURN.                                              *
*   NOTE: THE SUBROUTINE ASSUMES THE FOLLOWING REGISTERS:   *
*     R1-POINTS TO THE CCW STRING TO BE EXECUTED            *
*     R6-ADDRESS TO SAVE RETURN CODE                        *
*     R8-THE RETURN ADDRESS                                 *
*************************************************************
STARTIO  EQU   *
         ENTRY STARTIO
*
*     ENTER SUPERVISOR STATE 0, DISABLE INTERRUPTS
*
         DMSKEY NUCLEUS           ENTER SUPERVISOR STATE 0
         SSM   DISABLE            DISABLE I/O INTERRUPTS
*
*     STORE CCW ADDRESS, ISSUE START I/O
*
         ST    R1,CAW             STORE CCW ADDRESS IN CAW
         NI    SIOECB,X'00'       TURN OFF ECB COMPLETE BIT
         SLR   R2,R2              ZERO WORK REGISTER
         LH    R2,DEVAD           LOAD DEVICE ADDRESS
```

```
         SIO   0(R2)              START I/O FOR THE DEVICE
*
*     SAVE PSW AND CSW
*
         BALR  R15,0              GET THE DEVICE STATUS
         ST    R15,PSWCC          SAVE LOW 4 BYTES OF PSW
         MVC   SAVECSW(8),CSW     SAVE THE CSW
*
*     ENABLE INTERRUPTS, LEAVE SUPERVISOR STATE 0
*
         SSM   ENABLE             RESTORE SYSTEM MASK
*                                 FOR I/O INTR
         DMSKEY RESET             RETURN TO PROBLEM STATE
*
*     CHECK CONDITION CODE FROM START I/O FOR ERRORS.
*
         TM    PSWCC,B'00110000'  DID I/O INITIATE W/O ERROR
         BNE   SIOERR             IF NO, START I/O ERROR
*
*     WAIT FOR ECB TO BE POSTED BY INTERRUPT HANDLER
*
         WAIT  ECB=SIOECB         WAIT FOR SIO DEVICE END
*
*     TEST FOR UNIT CHECK BIT SET.
*
         TM    SAVECSW+4,X'02'    IS UNIT CHECK BIT SET
         BNZ   ERR1               IF YES, AN ERROR OCCURRED
         SLR   R2,R2              IF NO, I/O COMPLETED OK
         ST    R2,0(R6)           SET ZERO RETURN CODE
         BR    R8                 NORMAL RETURN
*
*     CHECK CC TO DETERMINE TYPE OF ERROR
*
SIOERR   TM    PSWCC,B'00010000'  IS CC 1 OR 3
         BZ    CC2                IF NOT, CC = 2
         TM    PSWCC,B'00100000'  IS CC = 3
         BZ    CC1                IF NOT, CC = 1
         L     R2,=X'FFFFFFFF'    CC = 3
         ST    R2,0(R6)           SET RC TO X'FFFFFFFF'
         BR    R8                 ABNORMAL RETURN.
*
*     CC = 1. CHECK FOR CHANNEL END ALONE. IF SET RC = 0.
*
CC1      CLI   SAVECSW+4,X'08'    IS CHANNEL END SET IN CSW
         BNZ   ERR1               IF NO, AN ERROR OCCURRED
         SLR   R2,R2              IF YES, I/O INITIATED OK
         ST    R2,0(R6)           SET ZERO RETURN CODE
*
*     RC = 0, WAIT FOR ECB TO BE POSTED BY INTERRUPT HANDLER
*
         WAIT  ECB=SIOECB         WAIT FOR SIO DEVICE END
         BR    R8                 NORMAL RETURN
*
*     RC = LAST 4 BYTES OF CSW.
*
ERR1     MVC   0(4,R6),SAVECSW+4  RC = LAST 4 BYTES OF CSW
         BR    R8                 ABNORMAL RETURN
*
*     CC = 2. RC = LAST 4 BYTES OF CSW WITH BUSY BIT SET.
*
CC2      MVC   0(4,R6),SAVECSW+4  RC = LAST 4 BYTES OF CSW
         OI    0(R6),X'10'        SET BUSY BIT OF CSW IN RC
         BR    R8                 ABNORMAL RETURN
*
```

```
         TITLE 'INTERRUPT HANDLER SUBROUTINE'
*************************************************************
*                                                           *
*               INTERRUPT HANDLER SUBROUTINE                *
*                                                           *
* THE FOLLOWING CODE IS INSTALLED AS THE INTERRUPT HANDLER  *
* FOR A SPECIFIC DEVICE USING A SVC 202 (CMS HNDINT) IN     *
* THE OPEN SUBROUTINE AND REMOVED USING A HNDINT CLR IN     *
* THE CLOSE SUBROUTINE. ONCE INSTALLED, THIS CODE GAINS     *
* CONTROL WHEN ANY INTERRUPT FOR THE DEVICE OCCURS. UPON    *
* ENTRY, REGISTERS ARE SET AS FOLLOWS:                      *
*    0-1 I/O OLD PSW                                        *
*    2-3 CHANNEL STATUS WORD (CSW)                          *
*      4 ADDRESS OF INTERRUPTING DEVICE                     *
*     14 RETURN ADDRESS                                     *
*     15 ENTRY POINT ADDRESS                                *
* THE CMS HNDINT ROUTINE, WHICH CALLS THIS ROUTINE          *
* PERFORMS ALL SAVING AND RESTORING OF REGISTERS. ONCE      *
* CALLED, THE INTERRUPT HANDLER WILL:                       *
*    1. CHECK IF THE INTERRUPT WAS A DEVICE END. IF IT WAS  *
*       THEN SAVE THE CSW, POST THE SIO ECB, AND RETURN TO  *
*       PREVIOUSLY EXECUTING CODE.                          *
*    2. IF NO DEVICE END, CHECK IF THE INTERRUPT WAS AN     *
*       ATTENTION. IF IT WAS, THEN POST THE ATTENTION ECB   *
*       AND RETURN CONTROL TO PREVIOUSLY EXECUTING CODE.    *
*    3. IF NO ATTN, RETURN TO PREVIOUSLY EXECUTING CODE.    *
*                                                           *
*************************************************************
HANDLER  EQU *
         ENTRY HANDLER
         USING HANDLER,R15          USE R15 FOR A MINUTE
         L     R12,=A(HALLS)        GET ADDR OF THIS CSECT
         DROP  R15
         USING HALLS,R12
*
*     IS THERE A DEVICE END? R3 = CSW+4.
         L     R5,=X'04000000'
         NR    R5,R3                IF DEVICE END NOT SET
         BZ    ATTNCHK              CHECK FOR AN ATTENTION
         ST    R2,SAVECSW           SAVE THE CSW
         ST    R3,SAVECSW+4
         LA    R1,SIOECB            GET ADDRESS OF SIO ECB
         POST  (R1)                 POST THE ECB
         LA    R15,0                SET RETURN CODE TO 0
         BR    R14                  RETURN
*
*     IS THERE AN ATTENTION? R3 = CSW+4.
ATTNCHK  L     R5,=X'80000000'
         NR    R5,R3                IF ATTN NOT SET, EXIT
         BZ    EXIT
*
*     GOT AN ATTENTION. POST THE ECB.
         LA    R1,ATTNECB           GET ADDR OF ATTN ECB
         POST  (R1)                 POST THE ECB
EXIT     LA    R15,0                SET RETURN CODE TO 0
         BR    R14                  RETURN
*
         TITLE 'DECLARATIVES'
R0       EQU   0
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
```

```
R8        EQU   8
R9        EQU   9
R10       EQU   10
R11       EQU   11
R12       EQU   12
R13       EQU   13
R14       EQU   14
R15       EQU   15
REGSAVE   DC    18F'0'
ENABLE    DC    XL1'FF'
DISABLE   DC    XL1'00'
SAVECSW   DC    D'0'
PSWCC     DC    F'0'
ATTNECB   DC    F'0'
SIOECB    DC    F'0'
ZERO12    DC    3F'0'
          NUCON NUCON
          END
```

Host Application Language Library using GAM and EXCP

Introductory Information

The Host Application Language Library using GAM and EXCP (HALLGE) is a package that provides a host high level language interface to the Device Attachment Control Unit (DACU) via calls to IBM System/370 subroutines. The package is written to execute on VM and MVS systems. The library contains one internal subroutine; an EXCP I/O driver (EXCPIO), and 8 user callable subroutines. The user callable subroutines open and close the Graphics Access Method (GAM) Data Control Block (DCB) which is used by EXCP, and specify that light pen attentions, the only type of attention the DACU sends, will be used (DOPEN and DCLOSE), issue I/O to the DACU (DWRITE, DREAD, DSTART, and DALARM), and wait or check for an attention from the DACU (DATTNW and DATTNI). DOPEN, DCLOSE, DATTNW and DATTNI use GAM macros to perform their functions. DWRITE, DREAD, DSTART, and DALARM use the EXCPIO subroutine to transfer data. All routines are described in detail below. All user callable subroutines must pass parameters as addresses to the required data, not the actual data. For all user callable subroutines, a return code of 0 indicates successful execution. The GAM maintenance level should be no older than 11/82.

HALLGE is entered into the system as IBM System/370 source code. To create a VM TEXT file, which may be linked to application TEXT files via the CMS LOAD command, enter the following two CMS statements:

```
GLOBAL MACLIB CMSLIB OSMACRO OSMACRO1
ASSEMBLE HALLGE
```

To create an MVS OBJ dataset, which may be linked to application OBJ datasets via the TSO LINK statement, enter the following statement:

```
ASMH HALLGE PR(HALLGE) OBJ(HALLGE) LIST ESD XREF(SHORT)
```

User Callable Subroutine Explanation

DOPEN - DACU Open:

Format: DOPEN(RC)

Where: RC = 4 byte return code

This routine performs a GAM OPEN to initialize a DCB for processing and a GAM SPAR to specify light pen attentions. The SPAR return code is passed to the caller via the RC parameter. For explanation of macros and return codes, see the *GAM/SP User's Guide.*

DCLOSE - DACU Close:

Format: DCLOSE

This routine performs a GAM CLOSE to disconnect the DCB. No return code is supplied. The DACU audible alarm automatically sounds when this routine is called. See the *GAM/SP User's Guide* for explanation of the macro.

DWRITE - DACU Write:

Format: DWRITE(LENGTH,DATA,ADDRESS,RC,STARTREG)

Where:

LENGTH = 4 byte length of data in bytes
DATA = Array of data to write
ADDRESS = 4 byte DACU address to write data
RC = 4 byte return code
STARTREG = Optional 2 byte start register value

This routine builds the necessary Channel Command Words (CCW's) to perform a Set Address Register and Write to the DACU and then calls the EXCPIO subroutine to perform the I/O operation. The return code from the EXCPIO subroutine is passed in the RC parameter. The routine performs the equivalent of a GAM GCTRL type IND (Set Program Function Indicators) and a GAM GWRITE (Write to Graphic-Device Buffer).

If the optional STARTREG parameter is specified, then a Set Start Register, equivalent to a GAM GCTRL type STR (Set Buffer Address Register and Start), will also be performed by chaining the Start's CCW to the end of the CCW string.

DREAD - DACU Read:

Format: DREAD(LENGTH,BUFFER,ADDRESS,RC,STARTREG)

Where:

LENGTH = 4 byte length of data in bytes
BUFFER = Array to read data into
ADDRESS = 4 byte DACU address to read data from
RC = 4 byte return code
STARTREG = Optional 2 byte start register value

This routine builds the necessary CCW's to perform a Set Address Register and Read from the DACU and then calls the EXCPIO subroutine to perform the I/O operation. The return code from the EXCPIO subroutine is passed in the RC parameter. The routine performs the equivalent of a GAM GCTRL type IND (Set Program Function Indicators) and a GAM GREAD (Read from Graphic-Device Buffer).

If the optional STARTREG parameter is specified, then a Set Start Register, equivalent to a GAM GCTRL type STR (Set Buffer Address Register and Start), will also be performed by chaining the Start's CCW to the end of the CCW string.

DSTART - DACU Start:

Format: DSTART(STARTREG,RC,ADDRESS)

Where:

STARTREG = 2 byte start register value
RC = 4 byte return code
ADDRESS = Optional 4 byte DACU address of the start location for the Control Space Interpreter

This routine builds the necessary CCW to perform a Set Start Register and then calls the EXCPIO subroutine to perform the I/O operation. The return code from the EXCPIO subroutine is passed in the RC parameter. The routine performs the equivalent of a GAM GCTRL type STR (Set Buffer Address Register and Start).

If the optional ADDRESS parameter is specified, then a Set Address Register, equivalent to a GAM GCTRL type IND (Set Program Function Indicators), will also be performed by chaining the SAR's CCW to the beginning of the CCW string. Using this parameter allows starting a Control Space program at a location other that at X'FFFF0000'. The byte offset to start the Control Space Interpreter is added to X'FFFF0000' and entered as the ADDRESS parameter. For example, if it is desired to start the Interpreter 12 bytes into the Control Space, then a X'FFFF000C' would be entered as the ADDRESS parameter. It should be noted here that wherever the Interpreter starts execution will be considered instruction 0 for branching instructions. See "Parallel I/O Application Programming Example" on page 174 for a sample of this subroutine's use.

DALARM - DACU Alarm:

Format: DALARM(RC)

Where:

RC = 4 byte return code

This routine calls the EXCPIO subroutine to perform the I/O necessary to sound the audible alarm on the DACU. The return code from the EXCPIO subroutine is passed in the RC parameter. This routine performs the equivalent of a GAM GCTRL type ALM (Set Audible Alarm).

DATTNW - DACU Attention Wait:

Format: DATTNW(BUFFER,RC)

Where:

BUFFER = 12 byte array to be filled with the following attention data:

X'00'	X'00'	X'00'	X'03'
SENSE DATA			
X-POSITION REGISTER		Y-POSITION REGISTER	

The sense data is 4 sense bytes sent from the DACU.

RC = 4 byte return code

This routine will perform a GAM ATTNINQ to wait for a light pen attention to be sent from the DACU. The DACU will send four sense bytes to the attention data save area, which is passed to the caller via the BUFFER parameter. The return code from the ATTNINQ macro is passed in the RC parameter. See the *GAM/SP User's Guide* for macro and return code explanation.

DATTNI - DACU Attention Inquiry:

Format: DATTNI(BUFFER,RC)

Where:

BUFFER = 12 byte array to be filled with the following attention data:

X'00'	X'00'	X'00'	X'03'
SENSE DATA			
X-POSITION REGISTER		Y-POSITION REGISTER	

RC = 4 byte return code

This routine will perform a GAM ATTNINQ to check for a light pen attention sent from the DACU. If an attention occurs, the DACU will send four sense bytes to the attention data save area, which is passed to the caller via the BUFFER parameter. If no attention occurs, the BUFFER parameter is filled with zeros. The return code from the ATTNINQ macro is passed in the RC parameter. See the *GAM/SP User's Guide* for macro and return code explanation.

Internal Subroutine Explanation

EXCPIO - EXCP I/O subroutine: This routine issues an EXCP macro call (EXCP) to execute a channel program to perform I/O operations to the device address specified at run time in the VM/CMS FILEDEF or MVS/TSO ALLOCATE statement. The user callable subroutine which calls this subroutine sets the following registers:

1 - Address of CCW string to be executed
6 - Address to save return code
7 - Length of CCW string - 1
8 - Return address to calling routine.

The EXCPIO subroutine will perform the following:

1. Save GAM's I/O Block (IOB) and set the address to the IOB for the CCW to be executed.
2. Move the CCW string to the IOB.
3. Perform a WAIT macro call on the EXCP Event Control Block (ECB) which will be posted when a Device End (DE) occurs.
4. Reset the GAM IOB Address.
5. Save the EXCP return code set in the EXCP ECB. See *OS/VS2 MVS Data Management Macro Instructions,* GC26-3873, for an explanation of the return code. If the I/O completes without error, a return code of 0 is saved.

Parallel I/O Application Programming Example

This programming example contains a host sample program, an exec to run the host program, a local PC program, and results of execution. The order of execution is as follows:

1. Run 'RASMONn X'.
2. Start the DACU base code.
3. Return to DOS by pressing the R key.
4. Execute the local PC sample program.
5. Enable the DACU.
6. Attach the IBM channel.
7. Execute the host exec to run the host sample program.

Host Sample Program

The following is an example of a host application program, written in VS FORTRAN Release 1.2, that uses HALLGE to communicate with the DACU. The program gives an example of opening and closing a DCB, writing to the Control Space and the Parallel I/O Buffer Space, reading from Buffer Space, starting the Control Space Interpreter and local PC code, waiting for an attention, and sounding the DACU audible alarm.

```
      PROGRAM SAMPLE
C
C  ********************************************************
C  * THIS PROGRAM PERFORMS THE FOLLOWING FUNCTIONS:       *
C  *   1) INITIALIZES A GAM DCB AND SPECIFIES LIGHT PEN   *
C  *      ATTENTIONS WILL BE USED.                        *
C  *   2) SOUNDS THE DACU AUDIBLE ALARM.                  *
```

```
C  *    3) WRITES CONTROL SPACE PROGRAMS TO THE CONTROL      *
C  *       SPACE TO WRITE TO THE PARALLEL I/O CONTROL        *
C  *       REGISTER (UBCTL) AND READ FROM THE PARALLEL I/O   *
C  *       STATUS REGISTER (UBSTAT), AND STARTS THE CONTROL  *
C  *       SPACE INTERPRETER TO WRITE TO UBCTL.              *
C  *    4) STARTS THE CONTROL SPACE INTERPRETER TO READ      *
C  *       FROM UBSTAT.                                      *
C  *    5) READS THE VALUE OF UBSTAT STORED IN THE PARALLEL  *
C  *       I/O BUFFER SPACE BY THE INTERPRETER.              *
C  *    6) INITIALIZES TWO 1K WORD BUFFERS.                  *
C  *    7) WRITES A BUFFER TO THE BUFFER SPACE.              *
C  *    8) READS A BUFFER FROM THE BUFFER SPACE AND STARTS   *
C  *       LOCAL PC CODE TO SEND AN ATTENTION.               *
C  *    9) WAITS FOR THE ATTENTION.                          *
C  *   10) COMPARES BUFFER READ AND WRITE VALUES.            *
C  *   11) CLOSES THE GAM DCB.                               *
C  ******************************************************
C
       INTEGER*4 RC,CSADR1 /ZFFFF0000/, CSADR2 /ZFFFF000C/
       INTEGER*2 ATTN(6),BUFIN(1024),BUFOUT(1024)
       INTEGER*2 CS(12),OKCNT,SSREG,UBSTAT
C
C********  OPEN GAM DCB. SOUND ALARM ONCE OPEN.  ********
       CALL DOPEN(RC)
       CALL DALARM(RC)
C
C********  WRITE CONTROL SPACE PROGRAMS TO THE     ********
C********  CONTROL SPACE TO WRITE TO UBCTL AND     ********
C********  READ FROM UBSTAT, AND START THE CONTROL ********
C********  SPACE INTERPRETER TO WRITE TO UBCTL.    ********
C
C           WUBCTL  XXXX   3FC0
C           END     XXXX   XXXX
C
C           RUBSTAT XXXX   0000
C           END     XXXX   XXXX
C
       DATA CS(1),  CS(2),  CS(3)  /Z0001,Z0000,Z3FC0/
       DATA CS(4),  CS(5),  CS(6)  /Z0000,Z0000,Z0000/
       DATA CS(7),  CS(8),  CS(9)  /Z0002,Z0000,Z0000/
       DATA CS(10), CS(11), CS(12) /Z0000,Z0000,Z0000/
       SSREG = 6
       CALL DWRITE(24,CS,CSADR1,RC,SSREG)
C
C********  START THE CONTROL SPACE INTERPRETER 12   ********
C********  BYTES INTO CONTROL SPACE TO READ UBSTAT. ********
       CALL DSTART(SSREG,RC,CSADR2)
C
C********  READ THE STATUS REGISTER VALUE.  ********
       CALL DREAD(2,UBSTAT,0,RC)
       WRITE(5,100) UBSTAT
 100   FORMAT(' PARALLEL I/O STATUS REGISTER = ',Z4)
C
C********  INITIALIZE INPUT AND OUTPUT BUFFERS.  ********
       DO 10 I = 1,1024
           BUFIN(I) = 0
           BUFOUT(I) = I
 10    CONTINUE
C
C********  WRITE THE OUTPUT BUFFER.  ********
       CALL DWRITE(2048,BUFOUT,0,RC)
C
C********  READ THE INPUT BUFFER AND START  ********
C********  LOCAL PC CODE TO SEND AN ATTENTION.  ********
       SSREG = 8
```

```
      CALL DREAD(2048,BUFIN,0,RC,SSREG)
C
C********  WAIT FOR THE ATTENTION.  ********
      CALL DATTNW(ATTN,RC)
C
C********  COMPARE THE TWO BUFFERS.  ********
      OKCNT = 0
      DO 20 I = 1,1024
         IF (BUFIN(I) .EQ. BUFOUT(I))THEN
            OKCNT = OKCNT+1
         END IF
 20   CONTINUE
      WRITE(5,110) OKCNT
      WRITE(5,120) ATTN
 110  FORMAT(' 1024 WORDS WRITE, READ, COMPARE.',I6,' OK.')
 120  FORMAT(' ATTENTION DATA = ',6Z5)
C
C********  CLOSE THE GAM DCB.  ********
      CALL DCLOSE
      STOP ' DONE.'
      END
```

Host Sample Exec

The following exec is run to execute the example host application program. The exec identifies the VS FORTRAN library VFORTLIB for use with the program, establishes that output data from the program be sent to the terminal, establishes that channel address 414 be assigned to the GAM DCB, called DACU, loads the application program and HALLGE, and starts the application.

```
GLOBAL TXTLIB VFORTLIB
FILEDEF 5 TERMINAL
FILEDEF DACU GRAF 414
LOAD SAMPLE HALLGE (START
```

In an MVS environment, the following CLIST will perform the equivalent operation to the above exec:

```
ALLOCATE FILE(DACU) DA(DUMMY) UNIT(414) OLD REUSE
ALLOCATE FILE(FT05F001) DA(*) SHR REUSE
ALLOCATE FILE(FT06F001) DA(*) SHR REUSE
LOADGO (SAMPLE HALLGE) LIB('SYS1.VFORTLIB') EP(SAMPLE)
FREE FILE(DACU FT05F001 FT06F001)
```

Local Sample Program

The following is an example of local PC code run on the DACU. It is to be used in conjunction with the previously described host FORTRAN program. The local PC code displays an initial message, sets up a Parallel I/O Start register entry point, and returns to PC base code. The start register entry point sends an attention to the host.

```
      PAGE    ,132
      .SALL
      TITLE SAMPLE: Local PC code for SAMPLE host program.
;
;MOVE MACRO
MOVE  MACRO   TO,FROM
      MOV     AX,FROM
      MOV     TO,AX
```

```
        ENDM
;INCLUDE PARALLEL I/O UTILITY AND GENERAL USAGE MACRO LIBS
IF1
        INCLUDE B:GENMAC.LIB
        INCLUDE B:UBUTIL.LIB
ENDIF
;
;******** DATA SEGMENT ********
;
SAMDS   SEGMENT PUBLIC 'DATA'
;INITIAL MESSAGE
ID      DB      'SAMPLE LOCAL CODE RUNNING',13,10,36
WNG9330 DB 'WUSR9330 UNABLE TO ENQUE ATTENTION ',13,10,36
        EVEN
;MAKE BRANCH TABLE SO DACU CODE CAN FIND ENTRY POINTS
ENTADDR DW      0,0             ;INTERRUPT ENTRY POINT
        DW      START,SAMCS     ;SET START REG ENTRY POINT
        DW      0,0             ;ATTENTION SENT ENTRY POINT
SAMDS   ENDS
;
;******** CODE SEGMENT ********
;
SAMCS   SEGMENT PUBLIC 'CODE'
        PUBLIC  SAMPLE
SAMPLE  PROC    FAR
        ASSUME  CS:SAMCS,DS:SAMDS
        MOVE    DS,SAMDS        ;SET LOCN OF DATA SEGMENT
        DISPLAY ID              ;WRITE ID MSG
        USETUP  ENTADDR         ;TELL DACU CODE ENTRY POINTS
        SLEEP                   ;RETURN TO DACU BASE CODE
;DONE FOR NOW. WAIT FOR CALL FROM DACU CODE
;
;******** SET START REGISTER ENTRY POINT ********
;
START:  MOV     AX,8804H        ;LOCAL PC CODE PRESENT, DEV 4
        MOV     DX,1234H        ;RECOGNIZABLE SENSE BYTE DATA
        HATTN   AX,DX           ;SEND ATTN TO HOST
        OR      AL,AL           ;CHECK RETURN CODE
        JZ      ATNRET          ;IF BAD DISPLAY WNG MSG
        PUSH    DS              ;SAVE DS
        MOVE    DS,SAMDS        ;SET DS TO SEG OF WNG MSG
        DISPLAY WNG9330         ;SEND WNG 9330
        POP     DS
ATNRET: RET
SAMPLE  ENDP
SAMCS   ENDS
;
;*****************
;  STACK SEGMENT  *
;*****************
SAMSS   SEGMENT STACK 'STACK' ;MAKE STACK 512 BYTES
        DB      64 DUP ('STACK   ')
SAMSS   ENDS
        END     SAMPLE
```

Sample Execution

When the previously mentioned example programs are executed, the following results are displayed at the terminal:

```
 SAMPLE
GLOBAL TXTLIB VFORTLIB
FILEDEF 5 TERMINAL
FILEDEF DACU GRAF 414
LOAD SAMPLE HALLGE (START
EXECUTION BEGINS...
 PARALLEL I/O STATUS REGISTER = 0040
 1024 WORDS WRITE, READ, COMPARE.  1024 OK.
 ATTENTION DATA =  0000 0003 0088 1234 0000 0000
IFY002I STOP   DONE.
R;
```

Serial I/O Application Programming Example

This programming example contains a host sample program, an exec to run the host program, and results of execution. The order of execution is as follows:

1. Run 'RASMONn X'.
2. Start the DACU base code.
3. Execute the local PC sample program.
4. Enable the DACU.
5. Attach the IBM channel.
6. Execute the host exec to run the host sample program.

Host Sample Program

The following is an example of a host application program, written in VS FORTRAN Release 1.2, that uses HALLGE to communicate with the DACU. The program gives an example of opening and closing a DCB, reading from Control Space, writing to Control Space and the Serial I/O Buffer Space, initializing the serial port, transmitting data from Buffer Space, waiting for an attention, and sounding the DACU audible alarm. It is designed to run with a 9600 baud even parity 7 bit ASCII terminal plugged into port 2 on the DACU. The ASCII characters 'This is a test message.', carriage return, line feed, EOT are transmitted to the ASCII terminal. If no device is attached to port 2, the data transmitted 'falls on the floor'.

```
      PROGRAM RS232C
C
C  ********************************************************
C  * THIS PROGRAM PERFORMS THE FOLLOWING FUNCTIONS:       *
C  *    1) OPENS A GAM DCB AND SPECIFIES THAT LIGHT PEN   *
C  *       ATTENTIONS WILL BE USED.                       *
C  *    2) SOUNDS THE DACU AUDIBLE ALARM.                 *
C  *    3) READS FROM CONTROL SPACE TO RETRIEVE DEFAULT   *
C  *       SERIAL I/O CONTROL PARAMETERS.                 *
C  *    4) CHANGES BYTES 12 AND 13 (ORIGIN 0) OF THE      *
C  *       CONTROL PARAMETERS TO SET TO IGNORE MODE,      *
C  *       XON/XOFF PROTOCOL.                             *
C  *    5) WRITES TO CONTROL SPACE TO STORE MODIFIED      *
C  *       CONTROL PARAMETERS AND ISSUES A START TO       *
C  *       INITIALIZE THE SERIAL PORT WITH THE PARAMETERS. *
C  *    6) WRITES EBCDIC CHARACTERS TO THE SERIAL I/O     *
C  *       BUFFER SPACE.                                  *
```

```
C  *    7) STARTS DATA TRANSFER TO THE SERIAL PORT.          *
C  *    8) WAITS FOR AN ATTENTION INDICATING THE TRANSFER    *
C  *       IS COMPLETE.                                      *
C  *    9) COMPARES NUMBER OF BYTES TRANSFERRED WITH THE     *
C  *       NUMBER OF BYTES SENT TO THE DACU.                 *
C  *   10) CLOSES THE GAM DCB.                               *
C  ******************************************************
C
       INTEGER*4 RC,CSADDR /ZFFFF0000/
       INTEGER*2 ATTN(6),CS(11),OKCNT,SSREG,UBSTAT
       LOGICAL*1 BUFOUT(26) /'T','H','I','S',' ',
     * 'I','S',' ','A',' ','T','E','S','T',' ','M','E','S',
     * 'S','A','G','E','.',' ',' ',' '/
       LOGICAL*1 CR /Z0D/, LF /Z25/, EOT /Z37/
C
C********  OPEN GAM DCB. SOUND ALARM ONCE OPEN.  ********
       CALL DOPEN(RC)
       CALL DALARM(RC)
C
C********  READ CONTROL SPACE TO RETRIEVE CONTROL PARMS.  **
C
       CALL DREAD(22,CS,CSADDR,RC)
C
C********  SET CS(7) TO 258 = 1*256 (HI BYTE - XON/XOFF) ***
C********  + 2 (LO BYTE - IGNORE MODE), WRITE PARMS BACK ***
C********  TO CONTROL SPACE, AND INITIALIZE THE SERIAL   ***
C********  PORT WITH THE MODIFIED PARMS.                 ***
C
       CS(7) = 258
       SSREG = 6
       CALL DWRITE(22,CS,CSADDR,RC,SSREG)
C
C********  WRITE THE OUTPUT BUFFER.  ********
       BUFOUT(24) = CR
       BUFOUT(25) = LF
       BUFOUT(26) = EOT
       CALL DWRITE(26,BUFOUT,0,RC)
C
C********  START TRANSFER TO THE SERIAL PORT.  ********
       SSREG = 4
       CALL DSTART(SSREG,RC)
C
C********  WAIT FOR THE ATTENTION.  ********
       CALL DATTNW(ATTN,RC)
C
C********  COMPARE # BYTES SENT WITH # BYTES TRANSFERRED.
C********  THE SENSE BYTES CONTAIN # BYTES TRANSFERRED.
       WRITE(5,100) ATTN(4)
 100   FORMAT(' 26 BYTES SENT TO BUFFER SPACE.',I6,
     *        ' TRANSFERRED.')
C
C********  CLOSE THE GAM DCB.  ********
       CALL DCLOSE
       STOP ' DONE.'
       END
```

Host Sample Exec

The following exec is run to execute the example host application program. The exec identifies the VS FORTRAN library VFORTLIB for use with the program,

establishes that output data from the program be sent to the terminal, establishes that channel address 412 be assigned to the GAM DCB, called DACU, loads the application program and HALLGE, and starts the application.

```
GLOBAL TXTLIB VFORTLIB
FILEDEF 5 TERMINAL
FILEDEF DACU GRAF 412
LOAD RS232C HALLGE (START
```

In an MVS environment, the following CLIST will perform the equivalent operation to the above exec:

```
ALLOCATE FILE(DACU) DA(DUMMY) UNIT(412) OLD REUSE
ALLOCATE FILE(FT05F001) DA(*) SHR REUSE
ALLOCATE FILE(FT06F001) DA(*) SHR REUSE
LOADGO (RS232C HALLGE) LIB('SYS1.VFORTLIB') EP(RS232C)
FREE FILE(DACU FT05F001 FT06F001)
```

Sample Execution

When the previously mentioned example programs are executed, the following results are displayed at the terminal:

```
 RS232C
GLOBAL TXTLIB VFORTLIB
FILEDEF 5 TERMINAL
FILEDEF DACU GRAF 412
LOAD RS232C HALLGE (START
EXECUTION BEGINS...
 26 BYTES SENT TO BUFFER SPACE.     26 BYTES TRANSFERRED.
IFY002I STOP   DONE.
R;
```

Source Code

The following section contains the source code for the eight user callable and one internal subroutines of the Host Application Language Library using GAM and EXCP (HALLGE). The code is entered as a single file and assembled into object code. The object code may then be linked to application object code to perform the operations previously described.

```
         MACRO
&LABEL   ENTER
.*-------------------------------------------------------------
.*       "ENTER" GENERATES THE STANDARD ENTRY CODE
.*-------------------------------------------------------------
&LABEL   STM   14,12,12(13)             SAVE REGISTERS
         ENTRY &LABEL
         USING &LABEL,15                USE R15 FOR A MINUTE
         L     12,=A(&SYSECT)           GET ADDR OF CSECT
         DROP  15
         USING &SYSECT,12
         LA    15,REGSAVE
         ST    13,4(,15)                FORWARD CHAIN
         ST    15,8(,13)                ...AND BACK
         LR    13,15                    POINT TO THE NEW ONE
         B     *+12                     SKIP TO THE START
         DC    CL8'&LABEL'
         MEND
         MACRO
&LABEL   LEAVE &RC=0
```

```
.*-------------------------------------------------------------
.*   "LEAVE" GENERATES THE STANDARD EXIT CODE
.*-------------------------------------------------------------
         AIF   ('&RC'(1,1) EQ '(').REG
&LABEL   LA    15,&RC                    GET RETURN CODE,
         AGO   .CHECK
.REG     ANOP
&LABEL   LR    15,&RC(1)                 SAVE THE RETURN CODE
.CHECK   L     13,4(,13)                 GET THE SAVE AREA
         L     14,12(,13)                PICK UP RETURN ADDR
         LM    0,12,20(13)               RELOAD THE REGS
         BR    14
         MEND
HALLGE   CSECT
         TITLE  'DACU OPEN ROUTINE.'
*************************************************************
*                                                           *
*                        DACU  OPEN                         *
*                                                           *
* SUBROUTINE DOPEN(RC)                                      *
*   WHERE:                                                  *
*      RC = 4 BYTE RETURN CODE FROM GAM SPAR MACRO.         *
*      SEE GAM/SP USER'S GUIDE FOR RETURN CODE EXPLANATION.*
*                                                           *
* THE FOLLOWING CODE WILL:                                  *
*   1. SAVE THE REGISTERS FROM THE CALLING ROUTINE.         *
*   2. PERFORM GAM OPEN - INITIALIZE A DCB FOR PROCESSING. *
*   3. PERFORM GAM SPAR - SPECIFY LIGHT PEN ATTENTIONS.     *
*      THE SPAR RETURN CODE IS PASSED TO THE CALLER.        *
*   4. SAVE RETURN CODE, RESTORE REGISTERS AND RETURN.      *
*                                                           *
*************************************************************
*
*     SAVE REGISTERS OF CALLING ROUTINE
*
DOPEN    ENTER
*
*     PERFORM OPEN AND SPAR
*
         LA    R9,EXCPDCB          ADDR DCB FOR SPAR AND OPEN
         USING IHADCB,R9
         L     R6,0(R1)            LOAD RETURN CODE ADDR
         OPEN  ((R9))              ISSUE OPEN.
         ST    R9,GACB2+4          PUT DCB ADDR IN GAM C.B.
         SPAR  (GACB2),PRTY=0      SPECIFY LIGHT PEN ATTNS
         DROP  R9
*
*     SAVE RETURN CODE, RESTORE REGISTERS AND RETURN.
*
         ST    R15,0(R6)           SAVE RETURN CODE FROM SPAR
         LEAVE RC=0
         TITLE 'DACU CLOSE ROUTINE.'
*************************************************************
*                                                           *
*                       DACU  CLOSE                         *
*                                                           *
* SUBROUTINE DCLOSE                                         *
*                                                           *
* THE FOLLOWING CODE WILL:                                  *
*   1. SAVE THE REGISTERS FROM THE CALLING ROUTINE.         *
*   2. PERFORM A GAM CLOSE - DISCONNECT THE DCB.            *
*   3. RESTORE REGISTERS AND RETURN.                        *
*                                                           *
*************************************************************
*
*     SAVE REGISTERS OF CALLING ROUTINE
```

```
*
DCLOSE   ENTER
*
*     PERFORM CLOSE
*
         LA    R9,EXCPDCB          ADDRESS DCB FOR CLOSE
         CLOSE ((R9))              ISSUE CLOSE.
*
*     RESTORE REGISTERS AND RETURN
*
         LEAVE RC=0
         TITLE 'DACU WRITE ROUTINE.'
************************************************************
*                                                          *
*                      DACU  WRITE                         *
*                                                          *
* SUBROUTINE DWRITE(LENGTH,DATA,ADDRESS,RC,STARTREG)       *
*   WHERE:                                                 *
*      LENGTH = 4 BYTE LENGTH OF DATA IN BYTES.            *
*      DATA = ARRAY OF DATA TO WRITE.                      *
*      ADDRESS = 4 BYTE DACU ADDRESS TO WRITE DATA.        *
*      RC = 4 BYTE RETURN CODE SAVED IN EXCPIO SUBROUTINE. *
*      SEE OS/VS2 MVS DATA MANAGEMENT MACRO INSTRUCTIONS   *
*      MANUAL FOR EXPLANATION OF RETURN CODE. NO ERROR IF  *
*      RC = 0.                                             *
*      STARTREG = OPTIONAL 2 BYTE START REGISTER VALUE.    *
*                                                          *
* THE FOLLOWING CODE WILL:                                 *
*   1. SAVE THE REGISTERS FROM THE CALLING ROUTINE.        *
*   2. READ PARAMETERS FROM THE CALLING ROUTINE AND USE    *
*      THEM TO BUILD CCW'S FOR A SET ADDRESS REGISTER AND  *
*      WRITE CHANNEL PROGRAM EQUIVALENT TO A GAM GCTRL     *
*      TYPE IND - SET PROGRAM FUNCTION INDICATORS, AND A   *
*      GAM GWRITE - WRITE TO GRAPHIC-DEVICE BUFFER.        *
*      IF THE OPTIONAL STARTREG PARAMETER IS SPECIFIED,    *
*      THEN A START, EQUIVALENT TO A GAM GCTRL TYPE STR -  *
*      SET BUFFER ADDRESS REGISTER AND START, WILL CHAINED *
*      TO THE END OF THE CCW STRING. PARMS PASSED ARE      *
*      ADDRESSES. REGISTER 1 POINTS TO THE PARAMETER LIST. *
*   3. CALL EXCP I/O SUBROUTINE WITH:                      *
*          R1 - POINTER TO CCW STRING                      *
*          R6 - ADDRESS TO SAVE RETURN CODE                *
*          R7 - LENGTH OF CCW STRING - 1                   *
*          R8 - RETURN ADDRESS                             *
*   4. RESTORE REGISTERS AND RETURN.                       *
*                                                          *
************************************************************
*
*     SAVE REGISTERS OF CALLING ROUTINE
*
DWRITE   ENTER
*
*     READ PARM DATA AND BUILD CCW'S
*
         SLR   R2,R2               ZERO WORK REGISTER
         L     R2,0(R1)            LOAD ADDR OF DATA LENGTH
         L     R2,0(R2)            LOAD DATA LENGTH
         STH   R2,WCCW+6           PUT LENGTH INTO CCW
         L     R2,4(R1)            LOAD ADDRESS OF DATA
         STCM  R2,B'0111',WCCW+1   PUT DATA ADDRESS INTO CCW
         L     R2,8(R1)            LOAD ADDR OF DACU ADDR REG
         L     R2,0(R2)            LOAD DACU ADDRESS REG
         ST    R2,WSARAD           PUT ADDR REG INTO SAR FLD
         L     R6,12(R1)           LOAD RETURN CODE ADDRESS
         LA    R7,15               LOAD CCW LENGTH - 1
```

```
         MVI   WCCW+4,X'20'       SET SUPPRESSION OF
*                                 POSSIBLE INCORRECT LENGTH
*                                 INDICATION ON
*
*     TEST FOR START REGISTER PARAMETER
*
         TM    12(R1),X'80'       IS RET CODE IS LAST PARM
         BO    WEXCP              IF YES, DON'T CHAIN START
         MVI   WCCW+4,X'60'       CHAIN START CCW
         L     R2,16(R1)          LOAD ADDRESS OF START REG
         ICM   R2,B'0011',0(R2)   LOAD START REG VALUE
         STH   R2,WSTRTREG        PUT VALUE INTO CCW
         LA    R7,23              LOAD CCW LENGTH - 1
*                                 (WITH START CCW)
*
*     CALL EXCP I/O ROUTINE WITH REGISTERS SET
*
WEXCP    LA    R1,WDACU           LOAD CCW STRING POINTER
         BAL   R8,EXCPIO          CALL EXCP I/O ROUTINE
*
*     RESTORE REGISTERS AND RETURN
*
         LEAVE RC=0
*
WDACU    EQU   *
WSARCCW  DC    X'1B',AL3(WSARAD),X'40',X'0',XL2'4'
WCCW     DC    X'01',3X'0',X'20',X'0',XL2'0'
WSTRTCCW DC    X'27',AL3(WSTRTREG),X'00',XL3'2'
WSARAD   DC    F'0'
WSTRTREG DC    H'0'
         TITLE 'DACU READ ROUTINE.'
*************************************************************
*                                                           *
*                        DACU  READ                         *
*                                                           *
* SUBROUTINE DREAD(LENGTH,BUFFER,ADDRESS,RC,STARTREG)       *
*   WHERE:                                                  *
*      LENGTH = 4 BYTE LENGTH OF DATA TO READ IN BYTES.     *
*      BUFFER = ARRAY TO READ DATA INTO.                    *
*      ADDRESS = 4 BYTE DACU ADDRESS TO READ DATA FROM.     *
*      RC = 4 BYTE RETURN CODE SAVED IN EXCPIO SUBROUTINE.  *
*      SEE OS/VS2 MVS DATA MANAGEMENT MACRO INSTRUCTIONS    *
*      MANUAL FOR EXPLANATION OF RETURN CODE. NO ERROR IF   *
*      RC = 0.                                              *
*      STARTREG = OPTIONAL 2 BYTE START REGISTER VALUE.     *
*                                                           *
* THE FOLLOWING CODE WILL:                                  *
*   1. SAVE THE REGISTERS FROM THE CALLING ROUTINE.         *
*   2. READ PARAMETERS FROM THE CALLING ROUTINE AND USE     *
*      THEM TO BUILD CCW'S FOR A SET ADDRESS REGISTER AND   *
*      READ CHANNEL PROGRAM EQUIVALENT TO A GAM GCTRL TYPE  *
*      IND -  SET PROGRAM FUNCTION INDICATORS, AND A GAM    *
*      GREAD - READ FROM GRAPHIC-DEVICE BUFFER.             *
*      IF THE OPTIONAL STARTREG PARAMETER IS SPECIFIED,     *
*      THEN A START, EQUIVALENT TO A GAM GCTRL TYPE STR -   *
*      SET BUFFER ADDRESS REGISTER AND START, WILL CHAINED  *
*      TO THE END OF THE CCW STRING. PARMS PASSED ARE       *
*      ADDRESSES. REGISTER 1 POINTS TO THE PARAMETER LIST.  *
*   3. CALL EXCP I/O SUBROUTINE WITH:                       *
*         R1 - POINTER TO CCW STRING                        *
*         R6 - ADDRESS TO SAVE RETURN CODE                  *
*         R7 - LENGTH OF CCW STRING - 1                     *
*         R8 - RETURN ADDRESS                               *
*   4. RESTORE REGISTERS AND RETURN.                        *
*                                                           *
*************************************************************
```

```
*
*     SAVE REGISTERS OF CALLING ROUTINE
*
DREAD    ENTER
*
*     READ PARM DATA AND BUILD CCW'S
*
         SLR   R2,R2               ZERO WORK REGISTER
         L     R2,0(R1)            LOAD ADDR OF DATA LENGTH
         L     R2,0(R2)            LOAD DATA LENGTH TO READ
         STH   R2,RCCW+6           PUT LENGTH INTO CCW
         L     R2,4(R1)            LOAD ADDRESS OF BUFFER
         STCM  R2,B'0111',RCCW+1   PUT BUFFER ADDR INTO CCW
         L     R2,8(R1)            LOAD ADDR OF DACU ADDR REG
         L     R2,0(R2)            LOAD DACU ADDRESS REG
         ST    R2,RSARAD           PUT ADDR REG INTO SAR FLD
         L     R6,12(R1)           LOAD RETURN CODE ADDRESS
         LA    R7,15               LOAD CCW LENGTH - 1
         MVI   RCCW+4,X'20'        SET SUPPRESSION OF
*                                  POSSIBLE INCORRECT LENGTH
*                                  INDICATION ON
*
*     TEST FOR START REGISTER PARAMETER
*
         TM    12(R1),X'80'        IS RET CODE LAST PARM
         BO    REXCP               IF YES, DON'T CHAIN START
         MVI   RCCW+4,X'60'        CHAIN START CCW
         L     R2,16(R1)           LOAD ADDRESS OF START REG
         ICM   R2,B'0011',0(R2)    LOAD START REG VALUE
         STH   R2,RSTRTREG         PUT VALUE INTO CCW
         LA    R7,23               LOAD CCW LENGTH - 1
*                                  (WITH START CCW)
*
*     CALL EXCP I/O ROUTINE WITH REGISTERS SET
*
REXCP    LA    R1,RDACU            LOAD CCW STRING POINTER
         BAL   R8,EXCPIO           CALL EXCP I/O ROUTINE
*
*     RESTORE REGISTERS AND RETURN
*
         LEAVE RC=0
*
RDACU    EQU   *
RSARCCW  DC    X'1B',AL3(RSARAD),X'40',XL3'4'
RCCW     DC    X'02',AL3(0),X'20',XL3'0'
RSTRTCCW DC    X'27',AL3(RSTRTREG),X'00',XL3'2'
RSARAD   DC    F'0'
RSTRTREG DC    H'0'
         TITLE 'DACU START ROUTINE.'
*************************************************************
*                                                           *
*                      DACU  START                          *
*                                                           *
* SUBROUTINE DSTART(STARTREG,RC,ADDRESS)                    *
*   WHERE:                                                  *
*      STARTREG = 2 BYTE START REGISTER VALUE.              *
*      RC = 4 BYTE RETURN CODE SAVED IN EXCPIO SUBROUTINE.  *
*      SEE OS/VS2 MVS DATA MANAGEMENT MACRO INSTRUCTIONS    *
*      MANUAL FOR EXPLANATION OF RETURN CODE. NO ERROR IF   *
*      RC = 0.                                              *
*      ADDRESS = OPTIONAL 4 BYTE DACU ADDRESS OF START      *
*                LOCATION IN CONTROL SPACE.                 *
*                                                           *
* THE FOLLOWING CODE WILL:                                  *
*   1. SAVE THE REGISTERS FROM THE CALLING ROUTINE.         *
```

```
*    2. READ A PARAMETER FROM THE CALLING ROUTINE AND USE    *
*       IT TO BUILD A CCW FOR A SET START REGISTER CHANNEL   *
*       PROGRAM EQUIVALENT TO A GAM GCTRL TYPE STR - SET     *
*       BUFFER ADDRESS AND START.                            *
*    3. IF THE OPTIONAL ADDRESS PARAMETER IS SPECIFIED,      *
*       THEN A SET ADDRESS REGISTER, EQUIVALENT TO A GAM     *
*       GCTRL TYPE IND - SET PROGRAM FUNCTION INDICATORS,    *
*       WILL BE CHAINED TO THE FRONT OF THE CCW STRING.      *
*       PARMS ARE PASSED AS ADDRESSES. REGISTER 1 POINTS TO  *
*       THE PARAMETER LIST.                                  *
*    4. CALL EXCP I/O SUBROUTINE WITH:                       *
*          R1 - POINTER TO CCW STRING                        *
*          R6 - ADDRESS TO SAVE RETURN CODE                  *
*          R7 - LENGTH OF CCW STRING                         *
*          R8 - RETURN ADDRESS                               *
*    5. RESTORE REGISTERS AND RETURN.                        *
*                                                            *
**************************************************************
*
*     SAVE REGISTERS OF CALLING ROUTINE
*
DSTART   ENTER
*
*     READ PARM DATA AND BUILD CCW
*
         SLR    R2,R2              ZERO WORK REGISTER
         L      R2,0(R1)           LOAD ADDR OF START REG
         ICM    R2,B'0011',0(R2)   LOAD START REG VALUE
         STH    R2,STRTREG         PUT VALUE INTO CCW
         L      R6,4(R1)           LOAD RETURN CODE ADDRESS
*
*     TEST FOR ADDRESS PARAMETER
*
         TM     4(R1),X'80'        IS RET CODE LAST PARM
         BO     SSREXCP            IF YES, DON'T CHAIN SAR
         L      R2,8(R1)           LOAD ADDR OF DACU ADDR REG
         L      R2,0(R2)           LOAD DACU ADDRESS REG
         ST     R2,SSARAD          PUT ADDR REG IN SAR FLD
*
*     CALL EXCP I/O ROUTINE WITH REGISTERS SET TO PERFORM A
*     SET ADDRESS REGISTER AND SET START REGISTER.
*
         LA     R1,SSARCCW         LOAD CCW STRING POINTER
         LA     R7,15              LOAD CCW STRING LENGTH - 1
         BAL    R8,EXCPIO          CALL EXCP I/O ROUTINE
*
*     RESTORE REGISTERS AND RETURN
*
         LEAVE RC=0
*
*     CALL EXCP I/O ROUTINE WITH REGISTERS SET
*     TO PERFORM A SET START REGISTER.
*
SSREXCP  LA     R1,STRTCCW         LOAD CCW STRING POINTER
         LA     R7,7               LOAD CCW STRING LENGTH - 1
         BAL    R8,EXCPIO          CALL EXCP I/O ROUTINE
*
*     RESTORE REGISTERS AND RETURN
*
         LEAVE RC=0
*
SSARCCW  EQU    *
         DC     X'1B',AL3(SSARAD),X'40',X'0',XL2'4'
STRTCCW  EQU    *
```

```
         DC    X'27',AL3(STRTREG),X'00',XL3'2'
SSARAD   DC    F'0'
STRTREG  DC    H'0'
         TITLE 'DACU AUDIBLE ALARM ROUTINE.'
*************************************************************
*                                                           *
*                      DACU  ALARM                          *
*                                                           *
* SUBROUTINE DALARM(RC)                                     *
*   WHERE:                                                  *
*      RC = 4 BYTE RETURN CODE SAVED IN EXCPIO SUBROUTINE.  *
*      SEE OS/VS2 MVS DATA MANAGEMENT MACRO INSTRUCTIONS    *
*      MANUAL FOR EXPLANATION OF RETURN CODE. NO ERROR IF   *
*      RC = 0.                                              *
*                                                           *
* THE FOLLOWING CODE WILL:                                  *
*   1. SAVE THE REGISTERS FROM THE CALLING ROUTINE.         *
*   2. CALL EXCP I/O SUBROUTINE WITH:                       *
*         R1 - POINTER TO CCW STRING                        *
*         R6 - ADDRESS TO SAVE RETURN CODE                  *
*         R7 - LENGTH OF CCW STRING                         *
*         R8 - RETURN ADDRESS                               *
*      A CHANNEL PROGRAM EQUIVALENT TO A GAM GCTLR TYPE     *
*      ALM - SET AUDIBLE ALARM WILL BE PERFORMED USING A    *
*      CCW WHICH NEEDS NO PARM DATA.                        *
*   3. RESTORE REGISTERS AND RETURN.                        *
*                                                           *
*************************************************************
*
*     SAVE REGISTERS OF CALLING ROUTINE
*
DALARM   ENTER
         SLR   R2,R2              ZERO WORK REGISTER
         L     R6,0(R1)           LOAD RETURN CODE ADDRESS
*
*     CALL EXCP I/O ROUTINE WITH REGISTERS SET
*
         LA    R1,ALRMCCW         LOAD CCW STRING POINTER
         LA    R7,7               LOAD CCW STRING LENGTH - 1
         BAL   R8,EXCPIO          CALL EXCP I/O ROUTINE
*
*     RESTORE REGISTERS AND RETURN
*
         LEAVE RC=0
*
ALRMCCW  EQU   *
         DC    X'0B',AL3(0),X'20',XL3'2'
         TITLE 'DACU ATTENTION WAIT ROUTINE.'
*************************************************************
*                                                           *
*                   DACU ATTENTION WAIT                     *
*                                                           *
* SUBROUTINE DATTNW(BUFFER,RC)                              *
*   WHERE:                                                  *
*      BUFFER = 12 BYTE ARRAY TO BE FILLED WITH THE         *
*         FOLLOWING ATTENTION DATA:                         *
*         -----------------------------------------------   *
*         |   X'00'  |   X'00'  |   X'00'  |   X'03'  |    *
*         -----------------------------------------------   *
*         |                SENSE DATA                 |    *
*         -----------------------------------------------   *
*         | X-POSITION REGISTER | Y-POSITION REGISTER |    *
*         -----------------------------------------------   *
*        THE SENSE DATA ARE 4 SENSE BYTES FROM THE DACU.    *
*      RC = 4 BYTE RETURN CODE FROM GAM ATTNINQ MACRO.      *
```

```
*      SEE GAM/SP USER'S GUIDE FOR RETURN CODE EXPLANATION.*
*                                                           *
* THE FOLLOWING CODE WILL:                                  *
*   1. SAVE THE REGISTERS FROM THE CALLING ROUTINE.         *
*   2. READ A BUFFER ADDRESS FROM THE CALLING ROUTINE.      *
*      PARMS PASSED ARE ADDRESSES.                          *
*      REGISTER 1 POINTS TO THE PARAMETER LIST.             *
*   3. PERFORM A GAM ATTNINQ - ATTENTION INQUIRY.           *
*      THE INQUIRY IS DONE IN A WAIT MODE, PLACING THE      *
*      ROUTINE IN A WAIT STATE UNTIL A LIGHT PEN ATTN       *
*      OCCURS.                                              *
*   4. MOVE ATTN DATA TO THE CALLING ROUTINE'S BUFFER.      *
*   5. SAVE RETURN CODE, RESTORE REGISTERS AND RETURN.      *
*                                                           *
*************************************************************
*
*     SAVE REGISTERS OF CALLING ROUTINE
*
DATTNW   ENTER
*
*     READ PARM DATA
*
         LA    R7,12              LOAD ATTENTION DATA LENGTH
         L     R5,0(R1)           LOAD ADDRESS OF BUFFER
         L     R6,4(R1)           LOAD RETURN CODE ADDRESS
*
*     WAIT FOR A LIGHT PEN ATTENTION
*
         ATTNINQ GACB2,MODE=W
*
*     MOVE ATTENTION DATA INTO BUFFER
*
         BCTR  R7,0               DECREMENT FOR EX
         EX    R7,ATTNWMVC        EXECUTE MVC
         XC    COMSAVE(12),COMSAVE CLEAR ATTENTION SAVE AREA
*
*     SAVE RETURN CODE, RESTORE REGISTERS AND RETURN
*
         ST    15,0(R6)           SAVE ATTNINQ RETURN CODE
         LEAVE RC=0
*
ATTNWMVC MVC   0(0,R5),COMSAVE
         TITLE 'DACU ATTENTION INQUIRY ROUTINE.'
*************************************************************
*                                                           *
*                  DACU ATTENTION INQUIRY                   *
*                                                           *
* SUBROUTINE DATTNI(BUFFER,RC)                              *
*   WHERE:                                                  *
*      BUFFER = 12 BYTE ARRAY TO BE FILLED WITH THE         *
*         FOLLOWING ATTENTION DATA:                         *
*         ------------------------------------------        *
*         |   X'00'  |   X'00'  |   X'00'  |   X'03'  |     *
*         ------------------------------------------        *
*         |               SENSE DATA                 |      *
*         ------------------------------------------        *
*         | X-POSITION REGISTER | Y-POSITION REGISTER |     *
*         ------------------------------------------        *
*         THE SENSE DATA ARE 4 SENSE BYTES FROM THE DACU.   *
*      RC = 4 BYTE RETURN CODE FROM GAM ATTNINQ MACRO.      *
*      SEE GAM/SP USER'S GUIDE FOR RETURN CODE EXPLANATION.*
*                                                           *
* THE FOLLOWING CODE WILL:                                  *
*   1. SAVE THE REGISTERS FROM THE CALLING ROUTINE.         *
*   2. READ A BUFFER ADDRESS FROM THE CALLING ROUTINE.      *
*      PARMS PASSED ARE ADDRESSES.                          *
```

```
*       REGISTER 1 POINTS TO THE PARAMETER LIST.              *
*    3. PERFORM A GAM ATTNINQ - ATTENTION INQUIRY.            *
*       THE INQUIRY IS DONE IN A CONDITION MODE, SO ATTN      *
*       DATA IS ONLY PLACED IN THE BUFFER IF AN ATTENTION     *
*       OCCURS, OTHERWISE THE BUFFER IS FILLED WITH ZEROS.    *
*    4. MOVE ATTN DATA TO THE CALLING ROUTINE'S BUFFER.       *
*    5. SAVE RETURN CODE, RESTORE REGISTERS AND RETURN.       *
*                                                             *
***************************************************************
*
*      SAVE REGISTERS OF CALLING ROUTINE
*
DATTNI   ENTER
*
*      READ PARM DATA
*
         LA    R7,12               LOAD ATTENTION DATA LENGTH
         L     R5,0(R1)            LOAD ADDRESS OF BUFFER
         L     R6,4(R1)            LOAD RETURN CODE ADDRESS
*
*      CHECK FOR A LIGHT PEN ATTENTION
*
         ATTNINQ GACB2,MODE=(C,LOADBUF),TYP=LP
         XC    COMSAVE(12),COMSAVE ZERO SAVE AREA IF NO ATTN
*
*      MOVE ATTENTION DATA INTO BUFFER
*
LOADBUF  BCTR  R7,0                DECREMENT FOR EX
         EX    R7,ATTNIMVC         EXECUTE MVC
         XC    COMSAVE(12),COMSAVE CLEAR ATTENTION SAVE AREA
*
*      RESTORE CALLING ROUTINE REGISTERS AND RETURN
*
         ST    R15,0(R6)           SAVE ATTNINQ RETURN CODE
         LEAVE RC=0
*
ATTNIMVC MVC   0(0,R5),COMSAVE
         TITLE 'EXCP I/O SUBROUTINE.'
***************************************************************
*                                                             *
*                    EXCP I/O SUBROUTINE                      *
*                                                             *
* THE FOLLOWING CODE WILL:                                    *
*    1. SAVE GAM'S IOB ADDRESS AND SET THE ADDRESS TO THE     *
*       IOB FOR THE CCW STRING TO BE EXECUTED.                *
*    2. MOVE THE CCW STRING TO THE IOB.                       *
*    3. PERFORM AN SVC 0 TO EXECUTE THE CCW'S BY CALLING      *
*       THE EXCP MACRO.                                       *
*    4. PERFORM A WAIT.                                       *
*    5. RESET GAM IOB ADDRESS.                                *
*    6. SAVE RETURN CODE FROM EXCP AND WAIT.                  *
*       SEE OS/VS2 MVS DATA MANAGEMENT MACRO INSTRUCTIONS     *
*       MANUAL FOR EXPLANATION OF RETURN CODE.                *
*    NOTE: THE SUBROUTINE ASSUMES THE FOLLOWING REGISTERS:    *
*      R1-POINTS TO THE CCW STRING TO BE EXECUTED             *
*      R6-ADDRESS TO SAVE RETURN CODE                         *
*      R7-LENGTH OF CCW STRING - 1                            *
*      R8-IS THE RETURN ADDRESS                               *
***************************************************************
EXCPIO   EQU   *
         LA    R4,EXCPDCB          ADDRESS DCB FOR EXCP.
         USING IHADCB,R4
         L     R5,DCBIOBAD         ADDR IOB FOR EXCP.
         ST    R5,CCWSAV           SAVE IOB ADDR. FROM GAM.
         LA    R5,EXCPIOB          GET MY IOB ADDRESS.
```

```
         ST    R5,DCBIOBAD         PUT IT IN GAM IOB.
         EX    R7,CCWMOV           MOVE CCW STRING TO IOB.
         NI    EXCPECB,X'00'       TURN OFF ECB WAIT BIT.
         EXCP  ((R5))              ISSUE EXCP.
         WAIT  ECB=EXCPECB         WAIT
         L     R5,CCWSAV           GET GAM IOB ADDR.
         ST    R5,DCBIOBAD         RESET IT IN GAM IOB.
         DROP  R4
         CLI   EXCPECB,X'7F'       DID I/O COMPLETE W/O ERROR
         BNE   EXCPERR             RETURN ERROR CODE.
         SLR   R2,R2               ZERO WORK REGISTER
         ST    R2,0(R6)            SET ZERO RETURN CODE.
         BR    R8                  NORMAL RETURN.
         SPACE 1
EXCPERR  EQU   *
         MVC   0(3,R6),EXCPECB     SAVE RETURN CODE
         BR    R8                  ABNORMAL RETURN.
CCWSAV   DC    F'0'                IOB CCW ADDR SAVE AREA
CCWMOV   MVC   EXCPCCWS(0),0(R1)   TARGET OF EX INSTR.
         TITLE 'DECLARATIVES'
R0       EQU   0
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R10      EQU   10
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
REGSAVE  DC    18F'0'
EXCPIOB  DS    0D
         DC    B'11000000'
         DC    B'10000000'
         DC    H'0'
         DC    A(EXCPECB)
         DC    D'0'
         DC    XL1'0'
         DC    AL3(EXCPCCWS)
         DC    XL1'0'
         DC    AL3(EXCPDCB)
         DC    3D'0'
ENDIOB   EQU   *
EXCPECB  EQU   *
         DC    F'0'
ENDECB   EQU   *
EXCPCCWS DS    0D
         DC    3D'0'
EXCPDCB  DCB   DSORG=GS,MACRF=(RC,WC),DDNAME=DACU,GTYPE=BASIC
GACB2    SAEC  EP=0,DCB=(R9),COMAREA=COMSAVE,ATTNTYP=(R,LP),  X
               PFKMSK=NULL
COMSAVE  DC    3F'0'
         DCBD  DSORG=GS
         END
```

Appendix B. Serial I/O Translation Tables

These tables specify the translations done on input and output of the Serial I/O ports. Inbound characters are placed in the input buffer as they arrive and translation applied (if translation is specified) before the host is notified that input has terminated. On output, the characters are sent from the host as EBCDIC characters and stored in the DACU as such. The translation is done in the DACU to convert from EBCDIC to ASCII (if translation is called for) before beginning to send out the port using the Output Translation Table. No translation is applied to control characters such as the Input Termination Character.

Note that the ASCII character '5E'X (caret) has two representations since it is the "logical not" in EBCDIC and caret in ASCII. This is due to the fact that there is no character "caret" in EBCDIC.

ASCII		EBCDIC		ASCII		EBCDIC		ASCII		EBCDIC	
Dec	Hex	Hex	Char	Dec	Hex	Hex	Char	Dec	Hex	Hex	Char
0	00	00	NUL	48	30	F0	0	96	60	79	
1	01	01	SOH	49	31	F1	1	97	61	81	a
2	02	02	STX	50	32	F2	2	98	62	82	b
3	03	03	ETX	51	33	F3	3	99	63	83	c
4	04	37	EOT	52	34	F4	4	100	64	84	d
5	05	2D	ENQ	53	35	F5	5	101	65	85	e
6	06	2E	ACK	54	36	F6	6	102	66	86	f
7	07	2F	BEL	55	37	F7	7	103	67	87	g
8	08	16	BS	56	38	F8	8	104	68	88	h
9	09	05	HT	57	39	F9	9	105	69	89	i
10	0A	25	LF	58	3A	7A	:	106	6A	91	j
11	0B	0B	VT	59	3B	5E	;	107	6B	92	k
12	0C	0C	FF	60	3C	4C	<	108	6C	93	l
13	0D	0D	CR	61	3D	7E	=	109	6D	94	m
14	0E	0E	SO	62	3E	6E	>	110	6E	95	n
15	0F	0F	SI	63	3F	6F	?	111	6F	96	o
16	10	10	DLE	64	40	7C	@	112	70	97	p
17	11	11	DC1	65	41	C1	A	113	71	98	q
18	12	12	DC2	66	42	C2	B	114	72	99	r
19	13	13	DC3	67	43	C3	C	115	73	A2	s
20	14	3C	DC4	68	44	C4	D	116	74	A3	t
21	15	3D	NAK	69	45	C5	E	117	75	A4	u
22	16	32	SYN	70	46	C6	F	118	76	A5	v
23	17	26	ETB	71	47	C7	G	119	77	A6	w
24	18	18	CAN	72	48	C8	H	120	78	A7	x
25	19	19	EM	73	49	C9	I	121	79	A8	y
26	1A	3F	SUB	74	4A	D1	J	122	7A	A9	z
27	1B	27	ESC	75	4B	D2	K	123	7B	C0	{
28	1C	1C	FS	76	4C	D3	L	124	7C	6A	¦
29	1D	1D	GS	77	4D	D4	M	125	7D	D0	}
30	1E	1E	RS	78	4E	D5	N	126	7E	A1	~
31	1F	1F	US	79	4F	D6	O	127	7F	07	DEL
32	20	40	SP	80	50	D7	P				
33	21	5A	!	81	51	D8	Q				
34	22	7F	"	82	52	D9	R				
35	23	7B	#	83	53	E2	S				
36	24	5B	$	84	54	E3	T				
37	25	6C	%	85	55	E4	U				
38	26	50	&	86	56	E5	V				
39	27	7D	'	87	57	E6	W				
40	28	4D	(	88	58	E7	X				
41	29	5D	)	89	59	E8	Y				
42	2A	5C	*	90	5A	E9	Z				
43	2B	4E	+	91	5B	AD	{				
44	2C	6B	,	92	5C	E0	\				
45	2D	60	-	93	5D	BD	]				
46	2E	4B	.	94	5E	5F	^/¬				
47	2F	61	/	95	5F	6D	_				

ASCII		EBCDIC		ASCII		EBCDIC		ASCII		EBCDIC	
Dec	Hex	Hex	Char	Dec	Hex	Hex	Char	Dec	Hex	Hex	Char
0	00	00	NUL	48	30	00		96	60	2D	-
1	01	01	SOH	49	31	00		97	61	2F	/
2	02	02	STX	50	32	16	SYN	98	62	00	
3	03	03	ETX	51	33	00		99	63	00	
4	04	00		52	34	00		100	64	00	
5	05	09	HT	53	35	00		101	65	00	
6	06	00		54	36	00		102	66	00	
7	07	7F	DEL	55	37	04	EOT	103	67	00	
8	08	00		56	38	00		104	68	00	
9	09	00		57	39	00		105	69	00	
10	0A	00		58	3A	00		106	6A	7C	¦
11	0B	0B	VT	59	3B	00		107	6B	2C	,
12	0C	0C	FF	60	3C	14	DC4	108	6C	25	%
13	0D	0D	CR	61	3D	15	NAK	109	6D	5F	_
14	0E	0E	SO	62	3E	00		110	6E	3E	>
15	0F	0F	SI	63	3F	1A	SUB	111	6F	3F	?
16	10	10	DLE	64	40	20	SP	112	70	00	
17	11	11	DC1	65	41	00		113	71	00	
18	12	12	DC2	66	42	00		114	72	00	
19	13	13	DC3	67	43	00		115	73	00	
20	14	00		68	44	00		116	74	00	
21	15	00		69	45	00		117	75	00	
22	16	08	BS	70	46	00		118	76	00	
23	17	00		71	47	00		119	77	00	
24	18	18	CAN	72	48	00		120	78	00	
25	19	19	EM	73	49	00		121	79	60	\
26	1A	00		74	4A	00		122	7A	3A	:
27	1B	00		75	4B	2E	.	123	7B	23	#
28	1C	1C	FS	76	4C	3C	<	124	7C	40	@
29	1D	1D	GS	77	4D	28	(	125	7D	27	'
30	1E	1E	RS	78	4E	2B	+	126	7E	3D	=
31	1F	1F	US	79	4F	00		127	7F	22	"
32	20	00		80	50	26	&	128	80	00	
33	21	00		81	51	00		129	81	61	a
34	22	00		82	52	00		130	82	62	b
35	23	00		83	53	00		131	83	63	c
36	24	00		84	54	00		132	84	64	d
37	25	0A	LF	85	55	00		133	85	65	e
38	26	17	ETB	86	56	00		134	86	66	f
39	27	1B	ESC	87	57	00		135	87	67	g
40	28	00		88	58	00		136	88	68	h
41	29	00		89	59	00		137	89	69	i
42	2A	00		90	5A	21	!	138	8A	00	
43	2B	00		91	5B	24	$	139	8B	00	
44	2C	00		92	5C	2A	*	140	8C	00	
45	2D	05	ENQ	93	5D	29	)	141	8D	00	
46	2E	06	ACK	94	5E	3B	;	142	8E	00	
47	2F	07	BEL	95	5F	5E	¬/^	143	8F	00	

ASCII		EBCDIC		ASCII		EBCDIC		ASCII		EBCDIC	
Dec	Hex	Hex	Char	Dec	Hex	Hex	Char	Dec	Hex	Hex	Char
144	90	00		192	C0	7B	{	240	F0	30	0
145	91	6A	j	193	C1	41	A	241	F1	31	1
146	92	6B	k	194	C2	42	B	242	F2	32	2
147	93	6C	l	195	C3	43	C	243	F3	33	3
148	94	6D	m	196	C4	44	D	244	F4	34	4
149	95	6E	n	197	C5	45	E	245	F5	35	5
150	96	6F	o	198	C6	46	F	246	F6	36	6
151	97	70	p	199	C7	47	G	247	F7	37	7
152	98	71	q	200	C8	48	H	248	F8	38	8
153	99	72	r	201	C9	49	I	249	F9	39	9
154	9A	00		202	CA	00		250	FA	00	
155	9B	00		203	CB	00		251	FB	00	
156	9C	00		204	CC	00		252	FC	00	
157	9D	00		205	CD	00		253	FD	00	
158	9E	00		206	CE	00		254	FE	00	
159	9F	00		207	CF	00		255	FF	00	
160	A0	00		208	D0	7D	}				
161	A1	7E	~	209	D1	4A	J				
162	A2	73	s	210	D2	4B	K				
163	A3	74	t	211	D3	4C	L				
164	A4	75	u	212	D4	4D	M				
165	A5	76	v	213	D5	4E	N				
166	A6	77	w	214	D6	4F	O				
167	A7	78	x	215	D7	50	P				
168	A8	79	y	216	D8	51	Q				
169	A9	7A	z	217	D9	52	R				
170	AA	00		218	DA	00					
171	AB	00		219	DB	00					
172	AC	00		220	DC	00					
173	AD	5B	[	221	DD	00					
174	AE	00		222	DE	00					
175	AF	00		223	DF	00					
176	B0	00		224	E0	5C	\				
177	B1	00		225	E1	00					
178	B2	00		226	E2	53	S				
179	B3	00		227	E3	54	T				
180	B4	00		228	E4	55	U				
181	B5	00		229	E5	56	V				
182	B6	00		230	E6	57	W				
183	B7	00		231	E7	58	X				
184	B8	00		232	E8	59	Y				
185	B9	00		233	E9	5A	Z				
186	BA	00		234	EA	00					
187	BB	00		235	EB	00					
188	BC	00		236	EC	00					
189	BD	5D	]	237	ED	00					
190	BE	00		238	EE	00					
191	BF	00		239	EF	00					

Appendix C. DACU Function Code Messages

The messages for Serial I/O port 2 are the same as for Serial I/O port 1 except that they contain a "SER2" instead of a "SER1".

See section "DACU Messages and Error Logging" on page 105 for an explanation of the message prefix conventions.

Note: When not specified for a specific message, no operator action is implied.

ISER1001 I/O INITIALIZED

Explanation: The serial device is initialized as a result of: (1) a power-on reset, or (2) the DACU being enabled.

System Action: The 8274 module is reset, all serial parameters are set to their default values, and the default buffers are obtained.

ISER1002 RETURN FROM ATTN RECEIVED

Explanation: An attention has been sent to the host. This message indicates that the host software has accepted the attention and acknowledged it with a Read XY Position command.

System Action: An attention pending bit is reset.

ISER1003 SET-START-REG RECEIVED

Explanation: A SET-START-REGISTER command for the serial device has been received from the channel.

System Action: The value of the Start Register is used to determine what action to take.

ISER1004 INITIALIZE SERIAL PORT

Explanation: A SET-START-REG value of '6' has been received.

System Action: The port is initialized using parameters in control space.

ISER1005 BEGIN SENDING CHARS

Explanation: A SET-START-REG value of '4' has been received.

System Action: The first character in the output buffer is sent out the serial port.

ISER1006 BEGIN RECEIVING CHARS

Explanation: A SET-START-REG value of '2' has been received.

System Action: The input serial port is enabled to begin receiving characters. Any characters received are placed in the input buffer.

ISER1007 DONE SENDING CHARS

Explanation: All characters in the buffer have been transmitted.

System Action: An attention is sent to the host.

ISER1008 DONE RECEIVING CHARS

Explanation: The serial port has completed an input operation due to : (1) the input buffer is full, (2) an Input Termination Character has been received, or (3) an input timeout has occurred.

System Action: An attention is sent to the host.

ISER1009 RECEIVE BUFFER FULL

Explanation: The input buffer has no more room.

System Action: An attention is sent to the host which indicates that "receive" has terminated.

ISER1010 SERIAL SELECTIVE RESET

Explanation: An channel selective reset for this device has been received.

System Action: Acknowledgement of receipt of the selective reset sent back to the dispatcher but no further action is taken.

ISER1011 SERIAL DEVICE END SENT

Explanation: A SET-START-REG command has been processed.

System Action: A device end command is sent to the dispatcher which will send a device end to the host.

ISER1012 SERIAL FUNCTION EXIT

Explanation: The serial code has completed its task.

System Action: Control is returned to dispatcher.

ISER1013 BEGINNING OF BREAK RECEIVED

Explanation: The 8274 module has detected the beginning of a "break" on the input port.

System Action: The DACU makes note of this and returns control to interrupted program. Another interrupt will come later when the break is complete.

ISER1014 START TIMEOUT

Explanation: The serial port is in the receive mode and a timeout time other than zero has been specified by the user. This messages indicates that timing has begun.

System Action: If the timeout time specified by the user is exceeded, the input mode will be terminated and an attention will be sent to the host.

ISER1016 BREAK SENT

Explanation: A space of 300 ms has been sent out the port.

System Action: The output of the port is set at a high level (space) and a timer started. After 300 ms the port is returned to normal operation.

ISER1017 END OF BREAK RECEIVED

Explanation: An earlier detected break on the input has now completed.

System Action: A "Break Received" attention is sent to the host.

ISER1021 RESTART SERIAL TRANSMISSION

Explanation: Serial transmission can begin again after an OEMI critical period.

System Action: Serial transmission restarts.

ISER1022 RESTART SERIAL RECEIVE

Explanation: Serial reception can begin again after an OEMI critical period.

System Action: Serial receive is re-enabled.

ISER1025 SET-START-REG TO DISPATCH USER CODE

Explanation: A Set-Start-Register command (x'08') has been received.

System Action: User code is started.

ISER1026 RETURN FROM USER CODE

Explanation: User code has returned control to the DACU code.

System Action: A Device End is sent to the host and control is passed to the Dispatcher.

ISER1027 ATTENTION STACKED

Explanation: An attempt was made to send an attention to the host but one was already pending.

System Action: The attention is stacked and control passed to the dispatcher. When the "return from attention" is received, the stacked attention will be sent.

ISER1028 SEND ATTN ACB FROM BATCH

Explanation: Serial input has completed.

System Action: An attention is sent from the batch code (which did the input translation, if called for) to the functional code.

ISER1029 RECEIVE ATTN ACB FROM BATCH

Explanation: The functional code has received the attention from the batch routine.

System Action: The attention is sent on to the host to indicate completion of input.

ISER1030 DONE BATCH OUTPUT TRANSLATE

Explanation: A command has been received from the host to commence serial output. Before it can begin, the data must be translated from EBCDIC to ASCII (if translation is called for). This message indicates that the translation has completed.

System Action: The functional code is dispatched to begin the actual transmission.

ISER1031 USER INTERRUPT ROUTINE ENTERED

Explanation: A serial interrupt has been received and there is a user entry point for interrupts.

System Action: Control is passed to the user interrupt entry point.

ISER1032 RETURN FROM USER INT ROUTINE

Explanation: The user interrupt routine has passed control back to the DACU code.

System Action: Control is passed to the interrupted routine.

ISER1033 SETUP USER ENTRY POINTS

Explanation: On initialization, user code has passed the table of entry points to the DACU code.

System Action: A pointer is stored which points to the user entry points for use when user code is later invoked.

ISER1034 DONE SENDING CHARS VIA DMA

Explanation: The DMA has been used to transmit characters and all characters have been sent.

System Action: An attention is sent to the host.

ISER1035 DONE RECEIVING CHARS VIA DMA

Explanation: The DMA has been used to receive characters and all characters have been sent to : (1) the input buffer is full, (2) an Input Termination Character has been received, or (3) an input timeout has occurred.

System Action: An attention is sent to the host.

WSER1101 EXTERNAL STATUS INTERRUPT

Explanation: An unexpected transition has occurred on one of the control lines (CTS, CD).

System Action: None.

Operator Response: Determine cause of error by checking control lines.

WSER1102 SPECIAL RX CONDITION

Explanation: An error has occurred on input. Possible errors are: Parity Error, Overrun Error, Framing Error.

System Action: An attention is sent to the host.

Operator Response: Determine cause of error. Possible explanations are: an incorrectly set baud, more characters sent than the buffer can hold, a faulty device.

WSER1103 UNKNOWN SET-START-REG

Explanation: A "Set Start Register" command has been received but the value of the register is not an allowed value.

System Action: A Device End is sent back to the host.

Operator Response: Correct host code.

WSER1104 SYNCH MODE NOT SUPPORTED

Explanation: A command to initiate synchronous mode has been received, but this is not supported by the DACU.

System Action: A Device End command is returned to the channel.

Operator Response: Do not specify anything other than a zero in the synchronous mode byte of the control space.

WSER1106 ERROR IN GIVING UP BUFFER

Explanation: An error has occurred in the buffer management section of the DACU code.

System Action: Buffer management is suspended and control is returned to the host.

Operator Response: Detach from the system, power back on and re-try the operation.

WSER1107 ERROR IN GETTING BUFFER

Explanation: The buffer space required is not available.

System Action: Buffer management is suspended.

Operator Response: Request a smaller buffer. If there is still a problem, the buffer pool has probably become fragmented. Disable the DACU and re-enable.

WSER1108 RECEIVE OVERRUN ERROR

Explanation: The 3 character buffer in the 8274 module is full and another character has arrived.

System Action: Input is terminated and an attention is sent to the host.

Operator Response: Make sure input has been enabled before characters are sent to the DACU. Another possible condition that could cause this error is when other devices in the DACU are active and have slowed the DACU's ability to retrieve characters from the port at the rate they have been received. This situation can usually be rectified by switching (if possible) to the DMA port.

WSER1109 RECEIVE PARITY ERROR

Explanation: An incorrect parity has been detected on an incoming character.

System Action: An Attention is sent to the host which indicates an error.

Operator Response: Verify that the parity of the device and the parity of the DACU are set the same.

WSER1110 UNKNOWN ACB FUNCTION

Explanation: The serial code has been requested to perform a function which is not known to it.

System Action: Control is returned to the dispatcher.

Operator Response: Check to make sure the code on the diskette has not been altered. Re-start the DACU and try again.

WSER1111 REQUESTED BAUD OUT OF RANGE

Explanation: A baud outside the range of 110 to 19200 has been requested.

System Action: This is just a warning. The DACU initialization continues. If the baud requested is above 19200 the DACU may not function correctly. There is no problem with baud values below 110. The warning is given to make sure this baud is what was requested.

Operator Response: Make sure baud is greater than or equal to 110 and less than or equal to 19200.

WSER1112 RECEIVE TIMEOUT

Explanation: More time has elapsed between incoming characters than specified in the receive timeout parameter.

System Action: The characters in the buffer are translated and the host is informed via an Attention that input has terminated.

Operator Response: Specify a longer timeout time or set it to zero to make the DACU wait forever.

WSER1113 SPURIOUS DMA INTERRUPT

Explanation: An interrupt has been received from the DMA port while one is pending.

System Action: Normal operation continues.

Operator Response: If problem persists, check for faulty connections at RS-232-C cable or noise from device on control lines.

WSER1114 ASKED TO RECEIVE AS MODEM BUT NO RTS

Explanation: Modem input mode has been selected but there is no active Request to Send (RTS) signal.

System Action: Input is disabled and a device end is sent to the host.

Operator Response: Determine why the device is not putting RTS ON or change to some other mode (such as "ignore").

WSER1115 DEVICE NOT READY. NO DSR OR DTR

Explanation: In terminal mode and no DSR signal detected or in modem mode and no DTR signal detected.

System Action: Control is returned to dispatcher and a device end sent to the host.

Operator Response: Determine why the device is not putting DSR or activating the DSR (pin 6) at the DACU or the DTR (pin 20) at the device end of a null modem cable.

WSER1116 ATTN STACKED AND TRYING TO SEND NEW

Explanation: A new attention request has arrived before the last has been sent.

System Action: The "attention pending" flag is turned off and control returned to the dispatcher.

Operator Response: Retry the operation. If error keeps occurring, try to determine why the attentions are not reaching the host.

WSER1117 RET FROM ATTN BUT NO ATTN SENT

Explanation: The code has received and ACB informing it that an attention has been sent to the host. However, there is no attention pending.

System Action: The "attention pending" flag is turned off and control returned to the dispatcher.

Operator Response: Restart the DACU.

IUNI4001 INITIALIZATION RECEIVED

Explanation: A power on has occurred or the DACU has just been enabled.

System Action: An INIT is performed on the Parallel I/O Interface and the default value placed in the control register (UBCTL).

IUNI4002 RETURN FROM ATTN RECEIVED

Explanation: An attention has been sent to the host. This message indicates that the host software has accepted the attention and acknowledged it with a Read XY Position command.

System Action: A bit is reset indicating that there are no Attentions pending for this device.

IUNI4003 SET-START-REG RECEIVED

Explanation: A SET-START-REGISTER command has been received from the channel.

System Action: The value of the START-REGISTER will be analyzed to determine what function is desired.

IUNI4004 INTERRUPT RECEIVED

Explanation: A device has interrupted the DACU.

System Action: If user code is present, it is dispatched. If not, an attention is sent to the host with the value of the interrupt vector in sense bytes 2 and 3.

IUNI4005 INTERRUPT JUST BEFORE CONTROL SPACE INTERPRETER

Explanation: An interrupt arrived while the control space interpreter was in the process of inhibiting Parallel I/O activity.

System Action: The interrupt is stacked and handled after the control space interpreter has finished its task.

IUNI4006 SELECTIVE RESET RECEIVED

Explanation: The channel has had a problem and has issued a selective reset to the Parallel I/O Interface.

System Action: The INIT line is made active and then made inactive.

IUNI4007 RETURN FROM ATTN SEND TO USER CODE

Explanation: An Attention sent by user code has been accepted by the channel.

System Action: The user code entry point for Attention Sent is entered.

IUNI4008 RETURN FROM USER RETURN FROM ATN INT

Explanation: The user has processed the Attention Sent routine and control has returned to the DACU code.

System Action: Control is passed back to the dispatcher.

IUNI4009 SET-START-REG TO DISPATCH USER CODE

Explanation: A SET-START-REGISTER value of '8' has been received.

System Action: The user code is invoked.

IUNI4010 RETURN FROM USER SET-START-REG

Explanation: The user code has completed its task.

System Action: Control is returned to the dispatcher.

IUNI4011 CONTROL SPACE INTERPRETER

Explanation: A SET-START-REGISTER value of '6' has been received.

System Action: The control space interpreter is invoked.

IUNI4012 USER INTERRUPT ROUTINE ENTERED

Explanation: A Parallel I/O Interface interrupt has occurred and user code is present.

System Action: The user interrupt routine is invoked.

IUNI4013 RETURN FROM USER INTERRUPT ROUTINE

Explanation: The user interrupt routine has completed its action.

System Action: Control is returned to the interrupted routine.

IUNI4014 SET UP USER ENTRY POINTS

Explanation: The user code as passed the Parallel I/O code its entry points.

System Action: A pointer is stored which points to the three user entry points.

WUNI4101 ATTENTION ATTEMPTED TO BE SENT WHEN ONE IS PENDING

Explanation: An attention has earlier been posted to the host and another attention is attempted to be sent before the host acknowledges the first attention with a Read XY Position command.

System Action: Control is passed to the interrupted routine.

Operator Response: Make sure the attached device is functioning properly.

WUNI4102 SOFTWARE TIME OUT OCCURRED WHILE READING XXXX REGISTER

Explanation: An attempt was made to read a device register but was unsuccessful. The address of the register which has been attempted to read to is displayed in the XXXX field.

WUNI4103 SOFTWARE TIME OUT OCCURRED WHILE WRITING XXXX REGISTER

Explanation: An attempt was made to write a device register but was unsuccessful. The address of the register which has been attempted to write to is displayed in the XXXX field.

System Action: A device end is sent to the channel.

Operator Response: The most common cause of this message is that an attempt was made to read or write a non-existent device register. Check the address requested and any address jumpers on the attached device.

WUNI4104 UNKNOWN SET START REG

Explanation: A Set-Start-Register value other than X'0006' or X'0008' has been received.

System Action: A Device End/Unit Check is sent back to the host.

Operator Response: Check host application code. Make sure bytes are stored in the host in high-byte/low-byte order.

WUNI4105 UNKNOWN CONTROL SPACE OP CODE RECEIVED

Explanation: A command outside the range specified in the section describing the control space interpreter has been received.

System Action: A device end/unit check is sent to the channel.

Operator Response: Check the host application code.

WUNI4106 SPURIOUS INTERRUPT RECEIVED

Explanation: A device interrupt has occurred but the latched bit which indicates an interrupt is not set.

System Action: Control is returned to the interrupted routine.

Operator Response: If this message continues to occur, refer to the Problem Determination Procedures.

WUNI4107 CONTROL SPACE LOOP COUNTER EXCEEDED

Explanation: The control space interpreter has looped on one instruction more times than specified.

System Action: A device end is sent to the channel.

Operator Response: Determine if the control space has been put in an infinite loop. Also, the condition may be brought about by a test and branch sequence in which the device is not ready by the time the loop counter reaches its limit. To correct, make the loop counter limit larger or set it to zero which will allow infinite looping.

WUNI4108 CONTROL SPACE BRANCH OUT OF RANGE

Explanation: The branch in the control space interpreter is to a destination outside of the control space.

System Action: A device end is sent to the channel.

Operator Response: Check the control space code.

WUNI4109 NO USER CODE

Explanation: A Set-Start-Register value of X'0008' was sent but there is no customer supplied code.

System Action: A device end is sent to the channel.

Operator Response: Make sure user code is loaded.

IOEM6000 DISABLE INTERRUPT ROUTINE ENTERED

Explanation: Disable key moved to disable position and detected by hardware.

System Action: DACU support code proceeds to take adapter off-line.

IOEM6001 STOP INTERRUPT ROUTINE ENTERED

Explanation: Data transfer complete from channel.

System Action: DACU support code proceeds to handle that event.

WOEM6002 STOPERR INTERRUPT ROUTINE ENTERED, ERROR REG =

Explanation: Data transfer complete from channel, but error occurred.

System Action: DACU support code proceeds to handle that event.

Operator Response: Record error register and subsequent messages. Refer to Problem Determination Procedures.

See message WOEM6025 for a brief description of error register.

IOEM6003 ZEROCNT INTERRUPT ROUTINE ENTERED

Explanation: Data transfer complete from channel.

System Action: DACU support code proceeds to handle that event.

WOEM6004 ZEROERR INTERRUPT ROUTINE ENTERED

Explanation: Data transfer complete from channel, but error occurred.

System Action: DACU support code proceeds to handle that event.

Operator Response: Record error register and subsequent messages. Refer to Problem Determination Procedures.

IOEM6005 IFDISC INTERRUPT ROUTINE ENTERED

Explanation: Data transfer complete from channel, but host ended transfer prematurely.

System Action: DACU support code proceeds to handle that event nearly like a normal data transfer termination except the DACU must reattach to the channel to present ending status for the current I/O.

WOEM6006 IFDISCERR INTERRUPT ROUTINE ENTERED

Explanation: Data transfer complete from channel, but host ended transfer prematurely and an error was detected.

System Action: DACU support code proceeds to handle that event nearly like a normal data transfer termination except the DACU must reattach to the channel to present ending status for the current I/O. Ending status may contain Unit Check.

Operator Response: Subsequent messages may detail problem that occurred.

IOEM6007 IFAVAIL INTERRUPT ROUTINE ENTERED

Explanation: Hardware has signaled the DACU code that the channel is available.

System Action: DACU support code proceeds to use channel resources if necessary.

IOEM6008 IFDISC STATUS PRESENTATION AFTER A STOP OR ZERO TERMINATION

Explanation: Channel has signaled interface disconnect to the DACU hardware after the data transfer for the current command is complete.

System Action: DACU support code attempts to reattach to the channel and send back ending status.

WOME6010 OEMIERR SUBROUTINE ENTERED, ERROR CODE =

Explanation: The DACU support code detected an unusual condition with the channel adapter.

System Action: DACU is put taken off line.

Operator Response: Operator should re-boot the DACU and retry. If problem persists, then see Problem Determination Procedures.

IOEM6011 NO MORE OEMI INTERRUPTS PENDING

Explanation: The only pending interrupt is the disable interrupt.

System Action: DACU support code starts processing disable interrupt.

WOEM6012 SELECTIVE RESET INTERRUPT FOR DEVICE ADDRESS XX

Explanation: Channel has determined that the DACU device has not performed the expected channel sequences within the proper amount of time. The channel then does a selective reset to the device that is the problem.

System Action: DACU hardware and software terminate the channel activity in progress, but continue to respond to channel commands for all device address once the device driver associated with the selective reset responds to notification of the selective reset.

Operator Response: If problem persists, see Problem Determination Procedures.

WOEM6013 RETRY XX PERFORMED BY HARDWARE FOR A CHANNEL ADAPTER LOGIC PARITY ERROR

Explanation: DACU has detected a problem with the channel attach hardware.

System Action: DACU retries to see if problem was intermittent.

Operator Response: See Problem Determination Procedures if problem persists.

WOEM6014 IFDISC PRIOR TO COMMAND OUT IGNORED

Explanation: The channel has sent an interface disconnect to a DACU device address even though DACU is not in the middle of a channel command sequence.

System Action: DACU ignores this interface disconnect and continues.

Operator Response: See Problem Determination Procedures if problem persists.

IOEM6015 LEAVING INTERRUPT ROUTINE VIA IRQEND

Explanation: The DACU channel attachment code is relinquishing control of the processor.

System Action: DACU is active and ready for channel commands or stimuli from the attached devices.

WOEM6016 20TH CHANNEL ADAPTER LOGIC PARITY ERROR

Explanation: DACU has detected a problem with the channel attach hardware and done 20 retries since DACU was last powered up.

System Action: DACU code assumes channel adapter is not functional and takes the DACU off-line.

Operator Response: See Problem Determination Procedures.

IOEM6017 SELECTIVE RESET ACB SENT TO DEVICE XX

Explanation: DACU channel attachment code has notified the correct device attachment routine of a selective reset for that device.

System Action: DACU code attempts to continue normal operation after selective reset is noted.

WOEM6018 SINGLE ERROR INTERRUPT OCCURRED - IGNORED

Explanation: DACU channel attachment code has detected an error with no apparent cause.

System Action: DACU code attempts to continue normal operation.

Operator Response: See Problem Determination Procedures if problem persist.

WOEM6019 A SELECTIVE RESET HAS OCCURRED FOR DEVICE XX DURING PROCESSING OF PREVIOUS SELECTIVE RESET FOR SAME DEVICE

Explanation: Channel has determined that the DACU device has not performed the expected channel sequences within the proper amount of time. The channel then does a selective reset to the device that is the problem.

System Action: DACU hardware and software terminate the channel activity in progress, but continue to respond to channel commands for all device address once the device driver associated with the selective reset responds to notification of the selective reset.

Operator Response: See Problem Determination Procedures if problem persists.

WOEM6020 CONTROL UNIT DISABLED AND OFF-LINE

Explanation: DACU devices are unavailable to channel due to the position of the enable/disable switch.

System Action: DACU hardware and software are active, but all channel communications are disallowed.

Operator Response: Enable switch must be set to enable position to make DACU available for channel I/O.

WOEM6021 SELECTIVE RESET RECEIVED WHEN DEVICE WAS NOT READY

Explanation: Device for which selective reset was issued, was in not-ready state.

System Action: DACU ignores selective reset.

WOEM6022 INTERRUPT RECEIVED DURING DISABLE

Explanation: A channel attachment interrupt occurred during sequence to take DACU off-line.

System Action: DACU processes interrupt before continuing off-line sequence.

WOEM6023 SPURIOUS DISABLE INTERRUPT RECEIVED

Explanation: An intermittent signal to take the device off-line was received.

System Action: DACU ignores the signal.

WOEM6024 CONTROL UNIT OFF-LINE

Explanation: Control unit has been taken off-line due to a previously detected error.

System Action: DACU ignores all channel stimuli.

Operator Response: DACU must be restarted to continue normal operation. Restart should be attempted only after operator uses Problem Determination Procedures to attempt to locate problem.

WOEM6025 ERROR REGISTER = XX

Explanation: A data transfer error occurred. If the error register has low order bit on (B'XXXXXXX1') and the data transfer was from DACU to the host, then a parity error occurred within the DACU. If the error register has bit one on (B'XXXXXX1X') and the data was being transferred from the host to the DACU, then a bus out parity error occurred. In other cases, the error was probably due to a channel adapter failure.

System Action: DACU sends unit check to host and may update sense information if possible. DACU attempts to continue normal operation.

Operator Response: See Problem Determination Procedures if problem persists.

IOEM6026 SETTING WORD TRANSFER FLAG FOR DEVICE ADDRESS XX

Explanation: Host program has requested word transfer mode be set for this device by setting the start register to '0001'.

System Action: DACU sets up word transfer for all data to and from the buffer for this device.

IOEM6027 CLEARING WORD TRANSFER FLAG FOR DEVICE ADDRESS

Explanation: Host program has requested byte transfer mode be set for this device by setting the start register to '0002'.

System Action: DACU sets up byte transfer for all data to and from the buffer for this device.

WOEM6031 NEWCMD+ERROR ENTERED FOR DEVICE

Explanation: The DACU detected a command from the host as well as an error.

System Action: DACU support code proceeds to analyze the error and report it to the operator. Ending status may contain Unit Check, if no status was pending before the error occurred. Command reject is set in the sense bytes.

Operator Response: Subsequent messages may detail problem that occurred.

WOEM6032 BAD COMMAND - COMMAND REJECT

Explanation: The DACU detected a command from the host that was not supported.

System Action: DACU ignores the command and sends back a unit check in the status and sets command reject in the sense data.

Operator Response: Modify host programming to eliminate invalid command.

IOEM6033 TIO RECEIVED

Explanation: The DACU detected a TIO command from the host.

System Action: DACU proceeds to service this command.

IOEM6034 WRITE RECEIVED

Explanation: The DACU detected a WRITE command from the host.

System Action: DACU proceeds to service this command.

IOEM6035 WRITE TO CONTROL SPACE

Explanation: The DACU detected a WRITE from the host with the address register pointing at control space.

System Action: DACU proceeds to service this command.

IOEM6036 WRITE TO CONTROL UNIT

Explanation: The DACU detected a WRITE command from the host to device 0.

System Action: DACU proceeds to service this command.

WOEM6037 ILLEGAL ADDRESS REG. ON READ/WRITE

Explanation: The DACU determined that the address register was not valid for a read or write to the specific device.

System Action: DACU sends back unit check in the initial status and sets command reject in the sense information.

Operator Response: Correct host programming.

IOEM6038 READ RECEIVED

Explanation: The DACU detected a READ command from the host.

System Action: DACU proceeds to service this command.

IOEM6039 READ FROM CONTROL SPACE

Explanation: The DACU detected a READ from the host and the address register implies a read from control space.

System Action: DACU proceeds to service this command.

IOEM6040 READ FROM CONTROL UNIT

Explanation: The DACU detected a READ command from the host for device 0.

System Action: DACU proceeds to service this command.

IOEM6041 NOP RECEIVED

Explanation: The DACU detected a NOP command from the host.

System Action: DACU proceeds to service this command.

IOEM6042 SENSE RECEIVED

Explanation: The DACU detected a SENSE command from the host.

System Action: DACU proceeds to service this command.

IOEM6044 SENSE ID RECEIVED

Explanation: The DACU detected a SENSE ID command from the host.

System Action: DACU proceeds to service this command.

IOEM6045 SET-STOP REG RECEIVED

Explanation: The DACU detected a SET-STOP reg command from the host.

System Action: DACU proceeds to service this command.

IOEM6046 SET-START-REG RECEIVED

Explanation: The DACU detected a SET-START reg command from the host.

System Action: DACU proceeds to service this command.

IOEM6047 SET-ADDRESS-REG RECEIVED

Explanation: The DACU detected a SET-ADDRESS reg command from the host.

System Action: DACU proceeds to service this command.

IOEM6048 PNDCMD ENTERED - DEVICE IS BUSY

Explanation: Device that was addressed from the channel is busy.

System Action: DACU will attempt to send back busy as initial status.

IOEM6049 PNDCMD ENTERED - STATUS IS PENDING

Explanation: Status is pending to send back to host for given device.

System Action: DACU will attempt to send back busy as initial status.

IOEM6050 TIO PROCESSED IN PNDCMD ROUTINE

Explanation: TIO command processing is in progress.

System Action: DACU will attempt to send back initial status.

IOEM6051 WRITE TO BUFFER

Explanation: Write data is for buffer of device.

System Action: DACU will set up to transfer data.

IOEM6052 READ FROM BUFFER

Explanation: Read data is from buffer of device.

System Action: DACU will set up to transfer data.

WOEM6053 COMMAND RECEIVED WHEN DEVICE IN NOT READY STATE

Explanation: Device is not ready for any channel command except sense until it changes to ready state.

System Action: DACU will send back unit check in status and set the command reject bit in the sense information.

Operator Response: Determine cause of not-ready state and take corrective action. May be due to host-DACU synchronization problem.

WOEM6054 MEMORY PARITY ON NEWCMD IRQ IGNORED

Explanation: Memory parity error occurred during period that it should not.

System Action: DACU ignores the error and treats it like a spurious error.

Operator Response: See Problem Determination Procedures if problem persists.

IOEM6055 SET AUDIBLE ALARM RECEIVED

Explanation: The DACU detected a SET AUDIBLE ALARM command from the host.

System Action: DACU proceeds to service this command.

IOEM6056 READ MANUAL INPUT RECEIVED

Explanation: The DACU detected a READ MANUAL INPUT command from the host.

System Action: DACU proceeds to service this command.

WOEM6057 BUS OUT ERROR ON NEWCMD IRQ

Explanation: The DACU detected a bus out error at the same time a command came from the host.

System Action: If no status is pending for the device, the DACU will respond with Unit check in the initial status and turn on command reject in the sense information. If status is pending or stacked for the device, the DACU will send back the pending or stacked status with the busy bit set.

Operator Response: If problem persists, see Problem Determination Procedures.

IOEM6058 READ XYP RECEIVED

Explanation: The DACU detected a READ X-Y POSITION command from the host.

System Action: DACU proceeds to service this command.

IOEM6059 SENDING ATTN/UC IN NORMAL STATUS

Explanation: The DACU is sending pending attention + unit check status in response to a test I/O command from the host as initial status.

System Action: DACU continues operation.

IOEM6071 INITIALIZATION ACB RECEIVED

Explanation: The channel adapter support code detects an initialization request.

System Action: Initialization proceeds.

IOEM6072 RETURN FROM SET-START-REG (DE) RECEIVED

Explanation: The channel adapter support code detects a request to send back normal ending status to a SET-START-REG or Audible Alarm command.

System Action: Reattachment to channel is attempted as soon as possible.

WOEM6073 RETURN FROM SET-START-REG (DE+UC) RECEIVED

Explanation: The channel adapter support code detects a request to send back ending status to a SET-START-REG command that contains unit check.

System Action: Reattachment to channel is attempted as soon as possible.

IOEM6075 ATTN+UC STATUS PRESENTATION ACB RECEIVED

Explanation: The channel adapter support code detects a request to send back attention status for a device.

System Action: Reattachment to channel is attempted as soon as possible.

WOEM6076 ILLEGAL ACB RECEIVED AND IGNORED

Explanation: The channel adapter support code detects an invalid request from another internal component.

System Action: Request is ignored.

WOEM6077 ACB AFTER SELECTIVE RESET IGNORED

Explanation: The channel adapter support code detects a request from another internal component while a selective reset is in progress.

System Action: Request is ignored.

WOEM6078 ILLEGAL ODD ACB FUNCTION CODE RECEIVED

Explanation: The channel adapter support code detects an illegal request from another internal component.

System Action: Request is ignored.

WOEM6079 DEVICE NOT-READY, ACB IGNORED

Explanation: The channel adapter support code detected a request from another internal component while the associated device was in a not-ready state.

System Action: Request is ignored.

IOEM6081 LEAVING OEMI FUNCTION HANDLER VIA FUNENT

Explanation: The channel adapter support is relinquishing control of the processor until an external stimuli restarts it.

System Action: Quiesce until required.

WOEM6082 UNEXPECTED SELECTIVE-RESET-RETURNED ACB

Explanation: The channel adapter support received a request from another internal component at an inappropriate time.

System Action: Quiesce until required.

WOEM6083 CONTROL UNIT DISABLED

Explanation: The DACU support code has detected the enable switch in the disable position.

System Action: Wait for the switch to be moved to the enable position before going on line to the channel.

WOEM6084 SPURIOUS ENABLE - IGNORED

Explanation: The DACU support code has detected the enable switch in the enable position, but it has not remained in that position.

System Action: Remain in disabled state.

Operator Response: If problem persists, see Problem Determination Procedures.

IOEM6085 ADAPTER ENABLED ... GOING ONLINE

Explanation: The DACU support code has detected the enable switch in the enable position.

System Action: The DACU will attempt to initialize and go online to the channel.

WOEM6086 ADAPTER ONLINE

Explanation: The DACU support code is online to the channel.

System Action: The DACU will attempt to handle any channel interactions.

IOEM6087 NOT-READY TO READY TRANSITION FOR DEVICE

Explanation: The DACU support code has been requested to make a device ready.

System Action: The DACU will make the device ready.

WOEM6088 READY TO NOT-READY TRANSITION FOR DEVICE

Explanation: The DACU support code has been requested to make a device not-ready.

System Action: The DACU will make the device not-ready until the inverse request is received.

WOEM6089 CONTROL UNIT DISABLED, ACB IGNORED

Explanation: The channel support code has received a request from an internal component when the device is not on line.

System Action: The request is ignored.

IOEM6090 A SELECTIVE RESET ACKNOWLEDGE RECEIVED FROM DEVICE ADDRESS XX

Explanation: The channel support code has received an acknowledge from an internal component to a selective reset.

System Action: The device address associated with the acknowledge is made active to the channel.

IOEM6091 SERVICING ANOTHER INTERRUPT

Explanation: The channel support code has detected a channel adapter condition that requires servicing before relinquishing control of the processor.

System Action: The DACU code handles that condition as normal.

IOEM6100 BUSY STATUS ACCEPTED

Explanation: The channel has accepted status with the busy bit on.

System Action: Normal operation continues.

IOEM6101 STATUS ACCEPTED

Explanation: The channel has accepted status.

System Action: Normal operation continues.

IOEM6103 STATUS SUPPRESSED ONLY

Explanation: The channel has suppressed status.

System Action: DACU will attempt to reattach and send back status when the channel is free.

IOEM6104 STATUS STACKED ONLY

Explanation: The channel has stacked status.

System Action: DACU will immediately attempt to reattach and send back status.

IOEM6105 NORMAL STATUS PRESENTATION, STATUS = 00

Explanation: The DACU is starting to send back initial or ending status.

System Action: DACU will continue processing.

IOEM6106 CHAINING INDICATED

Explanation: The DACU has detected a command chain during channel command processing.

System Action: DACU will continue processing command chain.

IOEM6107 NO CHAINING INDICATED

Explanation: The DACU has detected that is not processing a command chain.

System Action: DACU will continue processing in normal manner.

IOEM6108 IFDISC/SELRESET/ERROR IRQ DURING

Explanation: The DACU has detected an error condition while trying to present initial or ending status.

System Action: DACU will attempt to isolate the error.

Operator Response: See any subsequent messages for more information.

WOEM6109 ERROR: CHAIN AND NOCHAIN BOTH SET

Explanation: The DACU has detected an error condition during initial or ending status presentation.

System Action: DACU will go off-line.

Operator Response: See Problem Determination Procedures.

WOEM6110 ERROR: CHAIN AND NOCHAIN BOTH CLEARED

Explanation: The DACU has detected an error condition during initial or ending status presentation.

System Action: DACU will go off-line.

Operator Response: See Problem Determination Procedures.

WOEM6111 ERROR: IFBUSY SET DURING STATUS

Explanation: The DACU has detected an error condition during initial or ending status presentation.

System Action: DACU will go off-line.

Operator Response: See Problem Determination Procedures.

WOEM6112 ERROR: NO RESPONSE TO STATUS PRESENTATION

Explanation: The DACU has detected an error condition during initial or ending status presentation.

System Action: DACU will go off-line.

Operator Response: See Problem Determination Procedures.

IOEM6131 SSP ENTERED FOR REATTACHMENT ATTEMPT, STATUS = XX

Explanation: The DACU is attempting to reattach to the channel to send back unsolicited status.

System Action: DACU will continue processing.

IOEM6132 SEARCHING THROUGH DCBS FOR STACKED

Explanation: The DACU is determining if there is any pending status to send to the host.

System Action: DACU will test for each active device.

IOEM6133 BUSY STATUS ACCEPTED ON REATTACHMENT

Explanation: The DACU has sent back busy status during reattachment.

System Action: DACU will continue normal operation.

IOEM6134 STATUS ACCEPTED ON REATTACHMENT

Explanation: The channel has accepted unsolicited status.

System Action: DACU will continue normal operation.

IOEM6136 STATUS SUPPRESSED W/O BEING STACKED

Explanation: The channel has suppressed status.

System Action: DACU will attempt to reattach and send back status when the channel is free.

IOEM6137 STATUS STACKED ON REATTACHMENT

Explanation: The channel has stacked status.

System Action: DACU will immediately attempt to reattach and send back status.

IOEM6138 INTERRUPT IN RESPONSE TO REATTACHMENT

Explanation: The channel adapter has responded to a reattachment attempt with an interrupt condition.

System Action: DACU will analyze condition and take appropriate action. Status was not accepted at this time.

IOEM6139 NEWCMD INTERRUPT DURING REATTACHMENT

Explanation: The channel adapter has responded to a reattachment attempt with a new command for one of the DACU devices.

System Action: DACU will attempt to process that command.

IOEM6140 INTERFACE BUSY ON REATTACHMENT

Explanation: The channel was busy when reattachment was attempted.

System Action: DACU will attempt to reattach to the channel as soon as it is available.

IOEM6141 SUBROUTINE AVAIL ENTERED

Explanation: The channel adapter has notified the support code that the channel is available.

System Action: DACU will attempt to reattach to the channel and send any pending status.

WOEM6142 SPURIOUS INTERRUPT ON OEMI ADAPTER

Explanation: The DACU support code has detected as spurious interrupt from the channel attachment.

System Action: DACU will ignore the interrupt and attempt to continue normal operation.

Operator Response: If problem persists, see Problem Determination Procedures.

WOEM6144 UNRECOVERABLE SEQUENCER ERROR ON REATTACHMENT.

Explanation: An unrecoverable error has occurred in the channel attachment.

System Action: DACU will be taken off-line.

Operator Response: See Problem Determination Procedures.

WOEM6145 SELECTIVE RESET INTERRUPT DURING REATTACHMENT

Explanation: Channel has determined that the DACU device has not performed the expected channel sequences within the proper amount of time. The channel then does a selective reset to the device that is the problem.

System Action: DACU hardware and software terminate the channel activity in progress, but continue to respond to channel commands for all device address once the device driver associated with the selective reset responds to notification of the selective reset.

Operator Response: If problem persists, see Problem Determination Procedures.

IOEM6148 CHAINING INDICATED ON REATTACHMENT

Explanation: The DACU has detected a command chain during channel command processing.

System Action: DACU will continue processing command chain.

IOEM6149 INTERFACE DISCONNECT INTERRUPT

Explanation: The DACU has detected an interface disconnect while the DACU was trying to reattach to present status.

System Action: DACU will retain status for presentation later.

IOEM6150 IFBUSY DETECTED

Explanation: The DACU has detected that the channel was busy while the DACU was trying to reattach to present status.

System Action: DACU will retain status for presentation later when the channel is free.

IOEM6152 SUPRESSED DETECTED

Explanation: The channel has suppressed status.

System Action: DACU will attempt to reattach and send back status when the channel is free.

IOEM6154 STATUS SUPPRESSED

Explanation: The channel has suppressed status.

System Action: DACU will attempt to reattach and send back status when the channel is free.

IOEM6155 IFBUSY/IFAVAIL SIMULTANEOUS

Explanation: The channel attachment support code has detected interface busy and interface available at approximately the same time.

System Action: DACU will immediately attempt to reattach and send back status.

IOEM6156 INTERRUPT AFTER BEING STACKED

Explanation: The channel attachment support code has detected either an interrupt condition after being stacked.

System Action: DACU will save status and attempt to handle interrupt condition.

IOEM6157 INTERRUPT DETECTED AT SSP32

Explanation: Support personnel use only.

IDSP9900 DACU SUPPORT BEGINNING EXECUTION, RELEASE X.X

Explanation: DACU is beginning execution after a restart.

System Action: DACU will initialize to normal state.

WDSP9901 SPURIOUS INTERRUPT OCCURRED

Explanation: A spurious interrupt has been be detected by the DACU support code.

System Action: DACU will ignore the interrupt attempt to continue normally.

Operator Response: See Problem Determination Procedures if problem persists.

WDSP9902 QUEUE OVERFLOW TO COMP = XX, FROM COMP = XX

Explanation: Internal queue space has been exceeded.

System Action: DACU will ignore the request that caused the overflow and attempt to continue normally.

Operator Response: May be a problem with user code.

WDSP9903 ALL BATCH ACBS IN USE, BATCH REQUEST REJECTED

Explanation: DACU support code has detected that the batch facility has been saturated.

System Action: DACU will ignore the batch request and attempt to continue normally.

Operator Response: May be a problem with user code.

WDSP9904 DACU DISABLED AND OFFLINE

Explanation: DACU devices are unavailable to channel due to the position of the enable/disable switch.

System Action: DACU hardware and software are active, but all channel communications are disallowed.

Operator Response: Enable switch must be set to enable position to make DACU available for channel I/O.

IDSP9905 DACU SUPPORT LEAVING MEMORY

Explanation: DACU support code is returning control to DOS and relinquishing the memory it occupies.

System Action: DACU attempts to reset all DOS facilities that it has modified, before returning.

IDSP9906 REMAINING IN MEMORY

Explanation: DACU support code is returning control to DOS, but remaining in the memory it occupies. Either the users has pressed the 'R' key on the DACU console or the 'R' parameter was included when the support code was invoked.

System Action: DACU returns to DOS.

WDSP9908 TIMER INTERRUPT ROUTINE RESIDENT IS NOT ONE EXPECTED SEE USER'S MANUAL FOR WARNING.

Explanation: At initialization time, the DACU code checks to see that the normal PC timer routine is being used. If it is not then this message is displayed, but DACU initialization is continued.

System Action: DACU assumes that the timer routine loaded will provide the same environment for running as the normal timer routine, including the software interrupt x'1C' exit required for DACU operation and continues initialization as normal.

Operator Response: User should insure that an extension to PC DOS, that would take over the timer interrupt, is not loaded before loading the DACU code.

SSSP9999 PRESS KEY FOR MANUAL REQUEST

Explanation: The DACU support code will accept certain commands from the keyboard. Those commands are displayed to the operator via the menu below.

```
'PRESS R TO RETURN TO DOS BUT REMAIN MEMORY RESIDENT'
'PRESS P TO PRINT TRACE TABLE'
'PRESS 0,1,2,4 TO SEND ATTN/UC TO HOST'
'PRESS 5,6,7,9 TO SEND DE TO HOST'
'PRESS I TO SUPPRESS INFORMATION MSG'
'PRESS W TO SUPPRESS WARNING MSG'
'PRESS D TO DISPLAY ALL MSG'
'PRESS S TO STOP AFTER MSG DISPLAY'
'PRESS SPACE KEY TO LEAVE MEMORY'
```

System Action: DACU operates as normal, and will accept command from keyboard.

Appendix D. Diagnostic Monitor Messages

See section "DACU Messages and Error Logging" on page 105 for an explanation of the message prefix conventions.

Notes:

1. *In some of the messages which refer to the Serial I/O device, the fifth character (the '1') may be a '1' or a '2' depending on which port is being tested.*

2. *When not specified for a specific message, no operator action is implied.*

ISER1501 SERIAL PORT INITIALIZED

Explanation: The serial port is initialized for operation. The port is set for 9600 baud, even parity, one stop.

System Action: Proceeds to tests.

ISER1502 SERIAL PORT TEST

Explanation: The DACU is beginning a test to see if characters can be transmitted from the port. The character 'A' is sent. After verifying that transmission has begun, the character 'B' is sent. A check is then made to ensure that the 'B' has been sent.

System Action: Continues to next test.

ISER1503 SERIAL INTERRUPT OKAY

Explanation: The interrupt capability of the serial ports has been verified. The character 'C' has been transmitted and an interrupt has been received.

System Action: Continues to next test.

ISER1504 COMPLETED POLLING TESTS

Explanation: The above tests are complete.

System Action: Continues to DMA tests for port 1.

ISER1505 DONE DMA RECEIVE

Explanation: An interrupt has occurred indicating that a serial DMA operation is completed, and that operation is a 'receive'.

System Action: Continues to next test.

ISER1506 DONE DMA SEND

Explanation: An interrupt has occurred indicating that a serial DMA operation is completed, and that operation is a 'send'.

System Action: Continues to next test.

ISER1507 SERIAL DMA INTERRUPT OKAY (SEND)

Explanation: The characters 'D', 'E', and 'F' have been successfully transmitted and in interrupt fielded which indicates the end of the DMA operation.

System Action: Continues to next test.

ISER1508 SERIAL DMA INTERRUPT OKAY (RECEIVE)

Explanation: Characters 'G' and 'H' have been transmitted from port 2 to port 1 via the Serial Wrap (Null Modem) cable. Upon completion, an interrupt has been generated.

System Action: Continues to next test.

ISER1509 INTERRUPT RECEIVED

Explanation: The Serial port has successfully received an interrupt.

System Action: Continues to next test.

ISER1510 CHAR SENT FROM PORT 1 TO 2

Explanation: A character has been successfully sent from port 1, through the wrap cable, and received in port 2.

System Action: Continues to next test.

ISER1511 CHAR SENT FROM PORT 2 TO 1

Explanation: A character has been successfully sent from port 2, through the wrap cable, and received in port 1.

System Action: Continues to next test.

WSER1551 ERROR IN RECEIVE PORT

Explanation: At the beginning of the test, a check is made to make sure that there are no characters ready to be read.

System Action: Continue on to next test.

Operator Response: Make sure that a device is not transmitting to the DACU during this test. If not, there is probably a problem with the DMA/MEM card and it should be replaced.

WSER1552 TRANSMIT PORT IN INIT BAD STATE

Explanation: A character is in the transmitter port when one was not expected.

System Action: Continues on to next test.

Operator Response: There is probably a problem with the DMA/MEM card and it should be replaced.

WSER1553 CHAR DID NOT MOVE TO SHIFT REG

Explanation: An attempt has been made to transmit a character from the port. However, the character has not moved.

System Action: Continues on to next test.

Operator Response: Replace the DMA/MEM card.

WSER1554 2ND CHAR SHOULD NOT BE GONE YET

Explanation: When the second character is placed in the transmitter, it must wait until the first character is gone before it can be transmitted. It has been detected that the second character is gone before it should.

System Action: Continues on to next test.

Operator Response: Replace the DMA/MEM card.

WSER1555 2ND CHAR SHOULD BE GONE BY NOW

Explanation: A wait for a transmission of a character has taken longer than expected.

System Action: Continues on to next test.

Operator Response: Replace the DMA/MEM card.

WSER1556 NO SERIAL INTERRUPT

Explanation: Interrupts have been enabled but after sending a character out the port, an interrupt has not occurred.

System Action: Continues on to next test.

Operator Response: This problem could result from a faulty DMA/MEM card or a defective cable from the DACU to the PC.

WSER1557 MORE THAN ONE SERIAL INTERRUPT

Explanation: A test has been run that should generate exactly one interrupt. However, more than one has occurred.

System Action: Continues on to next test.

Operator Response: Replace the DMA/MEM card.

WSER1558 PROBLEM WITH RTS/CTS

Explanation: The DACU has turned the RTS (Request-To-Send) ON. The Serial wrap cable wraps this signal back to the port's CTS. If this message appears it means the CTS (Clear-To-Send) signal is not ON.

System Action: Continues on to next test.

Operator Response: The possible causes for this error message are:

1. The serial wrap cable has not been installed.
2. The cable from the DMA/MEM card to the EIA plug is not in place.
3. The above cable is defective.

WSER1559 PROBLEM WITH DTR/DSR

Explanation: The DTR (Data-Terminal-Ready) line has been turned ON at one port. The serial wrap cable connects this DTR signal to the DSR (Data-Set-Ready) signal at the other port.

System Action: Continues on to next test.

Operator Response: The possible causes for this error message are:

1. The serial wrap cable has not been installed.
2. The cable from the DMA/MEM card to the EIA plug is not in place.
3. The above cable is defective.

WSER1560 SERIAL TRANSMISSION PROBLEM

Explanation: A character has not been received when one is expected.

System Action: Continues on to next test.

Operator Response: The possible causes for this error message are:

1. The serial wrap cable has not been installed.
2. The cable from the DMA/MEM card to the EIA plug is not in place.
3. The above cable is defective.

WSER1561 SPURIOUS INTERRUPT

Explanation: An interrupt has arrived when not expected.

System Action: Continues on to next test.

Operator Response: Make sure a device is not causing noise.

WSER1562 NO SERIAL DMA INTERRUPT (SEND)

Explanation: A DMA transfer was initiated but the expected interrupt did not occur.

System Action: Continues on to next test.

Operator Response: The problem could be in the DMA/MEM card or in the cable which connects the DACU to the PC.

WSER1563 EXTRA DMA INTERRUPT(SEND)

Explanation: Only one DMA interrupt was expected but more than one arrived.

System Action: Continues on to next test.

Operator Response: If problem continues, replace DMA/MEM card.

WSER1564 NO SERIAL DMA INTERRUPT(RECEIVE)

Explanation: A DMA transfer was initiated but the expected interrupt did not occur.

System Action: Continues on to next test.

Operator Response: The problem could be in the DMA/MEM card or in the cable which connects the DACU to the PC.

WSER1565 EXTRA DMA INTERRUPT(RECEIVE)

Explanation: Only one DMA interrupt was expected but more than one arrived.

System Action: Continues on to next test.

Operator Response: If problem continues, replace DMA/MEM card.

WSER1566 TROUBLE SENDING CHAR FROM 1 TO 2

Explanation: A character sent from port 1 did not arrive at port 2.

System Action: Continues on to next test.

Operator Response: The possible causes for this error message are:

1. The serial wrap cable has not been installed.
2. The cable from the DMA/MEM card to the EIA plug is not in place.
3. The above cable is defective.

WSER1567 TROUBLE SENDING CHAR FROM 2 TO 1

Explanation: A character sent from port 2 did not arrive at port 1.

System Action: Continues on to next test.

Operator Response: The possible causes for this error message are:

1. The serial wrap cable has not been installed.
2. The cable from the DMA/MEM card to the EIA plug is not in place.
3. The above cable is defective.

WSER1568 PROBLEM WITH DCD LINE

Explanation: The RTS line has been turned ON on one port. Due to the presence of the null modem cable, this is reflected at the DCD (Data-Carrier-Detect) line at the other port. This expected signal is not present.

System Action: Continues on to next test.

Operator Response: The possible causes for this error message are:

1. The serial wrap cable has not been installed.
2. The cable from the DMA/MEM card to the EIA plug is not in place.
3. The above cable is defective.

IMON3000 I=SUPPRESS INFO. MSGS, W=SUPPRESS WARNINGS, B=BOTH

Explanation: This prompts user to select an option for message display mode.

System Action: DACU will read and parse input. If invalid DACU will prompt user for entry again.

Operator Response: Operator should respond with "I," "W," "B," or just "enter." If response is only "enter," then both informational and warning messages will be displayed.

IMON3001 DIAGNOSTICS NOT RUN, ADAPTER ENABLED

Explanation: Diagnostics will not be run if the enable/disable switch is in the enable position.

System Action: DACU will abort diagnostics.

Operator Response: If operator is attempting to run diagnostics, the enable/disable switch should be put in disable position and the diagnostics rerun.

IMON3002 INVALID ENTRY, RETRY

Explanation: Operator entry does not satisfy diagnostic code.

System Action: DACU will prompt the user for input again.

Operator Response: Examine prompting message and make correct entry.

IMON3004 ENTER S TO STOP ON WARNING MESSAGES

Explanation: DACU is prompting for run option to be taken if a warning message is displayed.

System Action: DACU will read and parse input from operator.

Operator Response: If "S" is entered, the DACU will stop after a warning message and prompt the operator to determine what action to take. If response is only "enter," the diagnostics will continue to execute immediately after any warning messages are displayed.

IMON3005 ENTER M TO RUN MANUAL INTERVENTION/EXTENDED TESTS

Explanation: DACU is prompting operator for which set of diagnostics to run.

System Action: DACU will read and parse input from operator.

Operator Response: If "M" is entered, the DACU will run both the bringup and the manual intervention/extended tests. If response is only "enter," only the bringup diagnostics will be run.

IMON3006 ENTER LOOP COUNT, 1-99 (DEFAULT=1, INFINITE=I)

Explanation: DACU is prompting operator for the number of times to run the diagnostic(s) selected.

System Action: DACU will read and parse input from operator.

Operator Response: Operator response may be only enter, in which case the loop count is set to one. Loop count may be entered as a one or two digit decimal number. If "I" is entered, the diagnostics are run until the program is terminated (by using control-break, by rebooting, etc.).

IMON3007 ENTRY MUST BE NUMERIC, 1-2 DIGITS

Explanation: DACU has parsed loop count and did not like entry.

System Action: DACU will prompt the operator again for the loop count.

Operator Response: Operator should make entry according to prompting message.

IMON3008 DACU TESTS

1 BUS TEST
2 TIMER/DMA
3 MEMORY
4 SERIAL ADAPTER
5 UNIBUS ADAPTER
6 CHANNEL ATTACHMENT
Z ALL TESTS ABOVE

ENTER ONE OR MORE SELECTIONS:

Explanation: DACU is prompting user for test selection.

System Action: DACU will read and parse input. If input is valid, DACU will run diagnostics selected.

Operator Response: Operator should make entry according to prompting message. Multiple tests can be run by selecting Z or typing the specific test ID's separated by blanks.

IMON3009 TEST ID NOT VALID

Explanation: DACU parsed input and found incorrect entry.

System Action: DACU will prompt user again and wait for input.

Operator Response: Operator should make entry according to prompting message.

IMON3010 PRESS G TO CONTINUE TESTS,

Explanation: A warning message has been displayed and the stop-after-warning option has been selected.

System Action: DACU will wait for user input.

Operator Response: If user enters "G," then testing will continue. If any other entry made, then the diagnostic program is aborted.

IMON3011 DACU TESTS, LEVEL 1.1

Explanation: Initial message on execution of the diagnostic (not shown in bring-up mode).

System Action: Prompt operator for input parameters.

Operator Response: None. If any other entry made, then the diagnostic program is aborted.

IUNI4500 PARALLEL I/O INTERFACE HAS BEEN INITIALIZED

Explanation: Software initialization of parallel interface.

System Action: Adapter will assert then negate Parallel interface "INIT" line.

WUNI4501 IRQ4 DID NOT OCCUR

Explanation: There was no IRQ4 generated during this test.

System Action: Continues on to next test.

Operator Response: See Problem Determination Procedures.

IUNI4502 IRQ4 TEST PASSED

Explanation: An IRQ4 was received at the PC.

System Action: Continues on to next test.

IUNI4504 THIS IS AN INTR FROM THE PARALLEL INTERFACE

Explanation: IRQ4 origin is from the parallel interface test hardware.

System Action: Continues on to next test.

WUNI4505 THIS IS A SPURIOUS IRQ4 INTERRUPT

Explanation: IRQ4 occurred, but it wasn't by the interface hardware.

System Action: Continues on to next test.

IUNI4506 NOW STARTING IRQ4 TEST

Explanation: Beginning of IRQ4 test.

System Action: Executes test.

IUNI4507 NOW STARTING DATA BUFFER TEST

Explanation: Beginning of data buffer test.

System Action: Executes test.

WUNI4508 SOMETHING IS WRONG WITH THE DATA

Explanation: Data buffer test failed.

System Action: Displays expected vs. actual data on screen, then continues with test.

Operator Response: See Problem Determination Procedures.

IUNI4509 THE DATA CHECKED OUT

Explanation: Data buffer test passed.

System Action: Continues on to next test.

IUNI4510 END OF IRQ4 TEST

Explanation: Code for IRQ4 test ends.

System Action: Continues on to next test.

IUNI4511 END OF DATA BUFFER TEST

Explanation: Code ends for data buffer test.

System Action: Continues on to next test.

IUNI4512 START OF ADDRESS BUFFER TEST

Explanation: Code starts for address buffet test.

System Action: Executes test.

IUNI4513 END OF ADDRESS BUFFER TEST

Explanation: Code ends for address buffer test.

System Action: Continues on to next test.

WUNI4514 THE ADDRESS BUFFER TEST FAILED

Explanation: There was an error during the address buffer test.

System Action: Error condition will be displayed on the screen.

Operator Response: See Problem Determination Procedures.

IUNI4515 ADDRESS BUFFERS ARE OK

Explanation: Address buffer test passed.

System Action: Continues on to next test.

WUNI4516 SOFTWARE TIME OUT OCCURRED

Explanation: During a read or write to a device register, the device register did not recognize DACU's attempt to transfer data.

System Action: Continues on to next test.

Operator Response: See Problem Determination Procedures.

WUNI4518 ERROR-NO SOFTWARE TIMEOUT OCCURRED

Explanation: Some test require a software timeout for successful completion hence there is an error if no timeout occurred.

System Action: Continues on to next test.

Operator Response: See Problem Determination Procedures.

IUNI4519 THIS IS THE START OF THE WRAP TEST

Explanation: Code starts the wrap test.

System Action: Executes test.

IUNI4520 THIS IS THE END OF THE WRAP TEST

Explanation: Code ends for the wrap test.

System Action: Continues on to next test.

IUNI4521 THE WRAP TEST PASSED

Explanation: Successful completion of the wrap test.

System Action: Continues on to next test.

WUNI4522 THE WRAP TEST FAILED

Explanation: Data mismatch during the wrap test.

System Action: Continues on to next test.

Operator Response: See Problem Determination Procedures.

IUNI4523 PLEASE INSERT WRAP CARD.....PRESS ANY KEY TO CONTINUE

Explanation: This is part of the extended diagnostics and at the proper time, this message will prompt operator to install wrap fixture and press any key to continue.

System Action: Waits until key is pressed and then executes test.

Operator Response: Insert wrap fixture.

IUNI4524 ACTUAL DATA FROM THE WRAP IS..xxxx EXPECTED DATA IS zzzz

Explanation: Message to let operator know what the data is from the wrap test and what it should have been.

System Action: Continues on to next test.

WUNI4525 PARALLEL I/O INTERFACE RESPONDED AND IT SHOULD NOT HAVE IN THIS TEST. ADDRESS USED WAS EEEE

Explanation: The test tries to write to a fictitious device register so there should be no response from the parallel interface but there was indicating a failure.

System Action: Continues on to next test.

Operator Response: See Problem Determination Procedures.

IUNI4526 NO ERROR OCCURRED ..

Explanation: During the BRBGTEST if no failure occurred, this message will be printed.

System Action: Continues on to next test.

IUNI4527 BRBG TEST PASSED

Explanation: Code ends for the bus request/ bus grant test.

System Action: Continues on to next test.

WUNI4528 BRBG TEST FAILED. MSK ON,SACK ON, GOT GRANT

Explanation: One of four failure messages during BRBG test. Which BR/BG pair failed will appear in the next message.

System Action: Continues on to next test.

Operator Response: See Problem Determination Procedures.

WUNI4529 BRBG TEST FAILED. MSK ON, SACK OFF, GOT GRANT

Explanation: One of four failure messages during BRBG test. Which BR/BG pair failed will appear in the next message.

System Action: Continues on to next test.

Operator Response: See Problem Determination Procedures.

WUNI4530 BRBG TEST FAILED. MSK OFF, SACK ON, GOT GRANT

Explanation: One of four failure messages during BRBG test. Which BR/BG pair failed will appear in the next message.

System Action: Continues on to next test.

Operator Response: See Problem Determination Procedures.

WUNI4531 BRBG TEST FAILED. MSK OFF, SACK OFF, NO GRANT

Explanation: One of four failure messages during BRBG test. Which BR/BG pair failed will appear in the next message.

System Action: Continues on to next test.

Operator Response: See Problem Determination Procedures.

IUNI4532 WHILE DOING BR4

Explanation: Message follows one of the failure messages to inform operator which step of the BRBG test failed.

System Action: Continues on to next test.

IUNI4532 WHILE DOING BR5

Explanation: Message follows one of the failure messages to inform operator which step of the BRBG test failed.

System Action: Continues on to next test.

IUNI4532 WHILE DOING BR6

Explanation: Message follows one of the failure messages to inform operator which step of the BRBG test failed.

System Action: Continues on to next test.

IUNI4532 WHILE DOING BR7

Explanation: Message follows one of the failure messages to inform operator which step of the BRBG test failed.

System Action: Continues on to next test.

IUNI4532 WHILE DOING NPR

Explanation: Message follows one of the failure messages to inform operator which step of the BRBG test failed.

System Action: Continues on to next test.

IUNI4532 WHILE DOING UBR

Explanation: Message follows one of the failure messages to inform operator which step of the BRBG test failed.

System Action: Continues on to next test.

IUNI4538 START OF BUS REQ/BUS GRANT TEST

Explanation: Code starts here for bus request bus grant test.

System Action: Executes test.

IUNI4539 END OF BUS REQUEST BUS GRANT TEST

Explanation: Code ends here for bus request bus grant test.

System Action: Continues on to next test.

IUNI4540 START OF DEVICE REGISTER WRITE/READ TEST

Explanation: Code starts for extended write/read of a device register.

System Action: Executes test.

IUNI4541 END OF DEVICE REGISTER WRITE/READ TEST

Explanation: Code ends for extended write/read test.

System Action: Continues on to next test.

IUNI4542 NOW ATTEMPTING A WRITE OF DATA WORD xxxx TO A DEVICE REGISTER LOCATION zzzz

Explanation: Parallel interface adapter will attempt to write to device register zzzz a word of data xxxx.

System Action: Executes test.

IUNI4543 THE WRITE TRANSACTION IS DONE

Explanation: Successful completion of a write to an active device register.

System Action: Continues on to next test.

IUNI4544 NOW ATTEMPTING A READ OF DEVICE REGISTER zzzz

Explanation: parallel interface adapter will attempt to read a device register zzzz.

System Action: Executes test.

IUNI4545 READ DATA FROM THE DEVICE REGISTER IS

Explanation: Displays actual data read back from the device register.

System Action: Continues on to next test.

WUNI4546 NO TRANSFER OF DATA TOOK PLACE

Explanation: Error condition that indicates no communication between the parallel I/O interface and the device register.

System Action: Continues on to next test.

Operator Response: See Problem Determination Procedures.

WUNI4547 DATA COMPARE FAILED-SEE DATA ABOVE

Explanation: Data mismatch occurred. Data will have been displayed by previous message.

System Action: Continues on to next test.

Operator Response: See Problem Determination Procedures.

WUNI4548 TRIED TO WRITE WORD xxxx BUT READ WORD zzzz

Explanation: Write data and read data from register is displayed.

System Action: Continues on to next test.

Operator Response: See Problem Determination Procedures.

IUNI4549 ENTER DEVICE REGISTER WRITE/READ ADDRESS

Explanation: Prompt from the program to the operator requesting valid device register address.

System Action: Waits for address then executes test.

Operator Response: Type in a valid device register address then hit return.

IUNI4550 REGISTER ENTERED NOT IN PARALLEL INTERFACE ADDRESS SPACE. ABORTING TEST!

Explanation: Address entered was not in the range of E000 to FFFF.

System Action: Aborts rest of write read test and exits to mainline.

Operator Response: Try again.

IUNI4551 REMOVE WRAP CARD. PRESS ANY KEY WHEN READY.

Explanation: Wrap test is finished so remove fixture.

System Action: Continues on to next test.

IUNI4552 FIRST TIME NULL LINE WILL END TEST ELSE NULL LINE EXECUTES ADDRESS PREVIOUSLY ENTERED.

Explanation: If you are running the manual intervention tests, and you do not wish to do the write read test, just enter a null line. If you want to loop on the test, just enter the address once, and the next time the prompt for the address comes up, a null line will execute that address.

System Action: Continues on to next test.

IUNI4553 NOW BEGINNING A TEST TO FIND ALL ACTIVE DEVICE REGISTERS

Explanation: Test will interrogate the entire parallel interface register address space and record which device registers responded.

System Action: Continues with test.

Operator Response: None.

IUNI4554 ACTIVE REGISTERS WILL BE INDICATED BY R OR W

Explanation: The active device registers will be displayed and their location in DACU space will be easily noted.

System Action: Continues with test.

Operator Response: None.

IUNI4555 FINISHED TESTING DEVICE REGISTERS

Explanation: Program has finished test.

System Action: Continues on to next test.

Operator Response: None.

IUNI4556 NO ACTIVE DEVICE REGISTERS FOUND

Explanation: After testing all 8k of the parallel interface address space, no active device registers could be found.

System Action: Continues on to next test.

IOEM6500 OEMI BRING-UP DIAGNOSTICS ENTERED

Explanation: The SU-OEMI diagnostics will be invoked.

System Action: The DACU will run the OEMI diagnostic code.

Operator Response: View subsequent messages, and respond to prompts.

IOEM6501 OEMI TEST MODE 0 PASSED

Explanation: The OEMI adapter has set the operational bit in the status register when commanded to do so by the SU.

System Action: The SU placed the OEMI adapter into TEST MODE 0, and then read the status register.

IOEM6502 OEMI TEST MODE 1 PASSED

Explanation: The OEMI adapter microcode has jumped to a prescribed address when commanded by the SU.

System Action: The SU placed the OEMI adapter into TEST MODE 1, and read the error register to verify that the high-order five bits corresponded to the expected address.

IOEM6503 OEMI TEST MODE 2 PASSED

Explanation: The OEMI adapter microcode has set all the bits it can in the OEMI registers, then checked and found them on.

System Action: The SU placed the OEMI adapter into TEST MODE 2, and read the interrupt, status and error registers, expecting to find the bits mentioned above. Also, check is made for an IRQ3 interrupt.

WOEM6504 OEMI TEST MODE 0 FAILED

Explanation: The SU did not sense a sequencer bit as expected.

System Action: Operator will be prompted to continue tests, or stop.

Operator Response: see Problem Determination Procedures.

WOEM6505 OEMI TEST MODE 1 FAILED

Explanation: The SU read a OEMI register, and did not obtain the expected address.

System Action: Operator will be prompted to continue tests, or stop.

Operator Response: see Problem Determination Procedures.

IOEM6507 OEMI ADAPTER ENABLED - NO TESTS RUN

Explanation: Refer to IMON3001.

System Action: N/A

Operator Response: N/A

WOEM6508 OEMI TEST MODE 2 FAILED:

Explanation: The test results referred to in IOEM6503 were not found.

System Action: N/A

Operator Response: See Problem Determination Procedures.

WOEM6509 OEMI STATUS REGISTER WRAP TEST FAILED

Explanation: The data sent by the SU to the OEMI status register did not compare to that received during the wrap test.

System Action: N/A

Operator Response: See Problem Determination Procedures.

WOEM6510 OEMI DATA REGISTER WRAP TEST FAILED

Explanation: The data send by the SU to the OEMI data register did not compare to that received.

System Action: N/A

Operator Response: See Problem Determination Procedures.

IOEM6511 OEMI DATA AND STATUS REGISTER WRAP TEST PASSED

Explanation: The data sent by the SU to the OEMI registers was read back and compared successfully.

System Action: N/A

WOEM6512 OEMI TEST MODE 2 FAILED: INTERRUPT BIT NOT RESET AFTER READ

Explanation: Test could not reset the interrupt register by reading it.

System Action: Leaves bring up diagnostics and continues to next test.

Operator Response: See Problem Determination Procedures.

WOEM6513 OEMI TEST MODE 2 FAILED: ERROR BIT NOT RESET AFTER READ

Explanation: Test could not reset the error register by reading it.

System Action: Leaves bring up diagnostics and continues to next test.

Operator Response: See Problem Determination Procedures.

WOEM6515 OEMI TEST MODE 0 FAILED STATUS REG NOT RESET AFTER READ

Explanation: Test could not reset the status register by reading it.

System Action: Leaves bring up diagnostics and continues to next test.

Operator Response: See Problem Determination Procedures.

WOEM6516 OEMI TEST MODE 2 FAILED: SEQUENCER BIT NOT SET IN IRQ REG

Explanation: Test reads the interrupt register and finds bit missing that should have been set by the sequencer.

System Action: Leaves bring up diagnostics and continues to next test.

Operator Response: See Problem Determination Procedures.

WOEM6517 OEMI TEST MODE 2 FAILED: SEQUENCER BIT NOT SET IN STATUS REG

Explanation: Test reads the status register and finds bits missing that should have been set by the sequencer.

System Action: Leaves bring up diagnostics and continues to next test.

Operator Response: See Problem Determination Procedures.

WOEM6518 OEMI TEST MODE 2 FAILED: SEQUENCER BIT NOT SET IN ERROR REG

Explanation: Test reads the error register and finds bits missing that should have been set by the sequencer.

System Action: Leaves bring up diagnostics and continues to next test.

Operator Response: See Problem Determination Procedures.

WOEM6519 OEMI TEST MODE 2 FAILED: STATUS BIT NOT RESET AFTER READ

Explanation: Test could not reset the status register by reading it.

System Action: Leaves bring up diagnostics and continues to next test.

Operator Response: See Problem Determination Procedures.

IOEM6521 OEMI MANUAL INTERVENTION TEST ENABLE CONTROL UNIT ENABLE SWITCH HIT ANY KEY WHEN ENABLED

Explanation: This test will verify that the ENABLE/DISABLE switch can be recognized by the software.

System Action: N/A

Operator Response: Operator will be prompted.

IOEM6522 ENABLE SWITCH IS ENABLED NOW THROW SWITCH TO DISABLE POSITION HIT ANY KEY WHEN DISABLED

Explanation: The software has detected that the ENABLE/DISABLE switch has been placed in the ENABLE position.

System Action: N/A

IOEM6523 ENABLE SWITCH IS DISABLED

Explanation: The software has detected that the ENABLE/DISABLE switch has been placed in the DISABLE position.

System Action: N/A

WOEM6524 OEMI MANUAL INTERVENTION TEST FAILED: SWITCH NOT ENABLED

Explanation: The action of the ENABLE/DISABLE switch was not detected.

System Action: N/A

Operator Response: See Problem Determination Procedures.

WOEM6525 OEMI MANUAL INTERVENTION TEST FAILED: SWITCH NOT DISABLED

Explanation: Refer to WOEM6524.

System Action: N/A

Operator Response: See Problem Determination Procedures.

IOEM6526 OEMI DISABLE IRQ BIT SET IN ADAPTER TOGGLE ENABLE SWITCH FROM DISABLE TO ENABLE, AND BACK TO DISABLE. HIT ANY KEY WHEN READY.

Explanation: The transition of the ENABLE/DISABLE switch has correctly caused an interrupt.

System Action: N/A

Operator Response: Respond to prompts.

WOEM6527 OEMI MANUAL INTERVENTION TEST FAILED: DISABLE BIT IN OEMI INTERRUPT REGISTER

Explanation: Refer to WOEM6524.

System Action: N/A

Operator Response: See Problem Determination Procedures.

IOEM6528 OEMI DISABLE INTERRUPT TEST SUCCESSFUL

Explanation: The transition of the ENABLE/DISABLE switch correctly caused an interrupt.

System Action: N/A

IOEM6529 OEMI MANUAL INTERVENTION TEST PASSED

Explanation: Tests involving the ENABLE/DISABLE switch passed.

System Action: N/A

WOEM6530 OEMI MANUAL INTERVENTION TEST FAILED: LEVEL 3 INTERRUPT NOT RECEIVED

Explanation: The action of the ENABLE/DISABLE switch did not cause an IRQ3 interrupt, as expected.

System Action: N/A

Operator Response: See Problem Determination Procedures.

IOEM6531 CONTROL UNIT ENABLED, MANUAL INTERVENTION TEST NOT RUN.

Explanation: The manual intervention test can only be run if the ENABLE/DISABLE switch is placed in the DISABLE position.

System Action: N/A

Operator Response: See Problem Determination Procedures.

WOEM6533 OEMI MANUAL INTERVENTION TEST FAILED: INTERRUPT BIT NOT RESET AFTER READ

Explanation: When the SU reads the OEMIIRQ register in response to an interrupt, the bits in the register are supposed to be reset, which would be verified on a subsequent read.

System Action: N/A

Operator Response: See Problem Determination Procedures.

IOEM6534 DMA TEST FROM THE CHANNEL THIS TEST LEAVES THE ADAPTER IN A BAD STATE ENTER Y IF YOU WANT TO RUN THIS TEST

Explanation: Start of DMA test for the OEMI adapter. User is given the option of running or ignoring this test section.

System Action: Continues on to test if yes, ignores test if any other entry is made.

Operator Response: Enter Y to run test; hit enter to ignore.

IOEM6535 DMA TEST PASSED.

Explanation: Adapter passes test criteria.

System Action: Continues on to next test.

WOEM6536 DMA TEST FAILS WHILE READING FROM PAGE 5

Explanation: Adapter fails to move data from specific area of RAM.

System Action: Continues on to next test phase of DMA test.

Operator Response: See Problem Determination Procedures.

WOEM6537 DMA TEST FAILS WHILE READING FROM PAGE 6

Explanation: Adapter fails to move data from specific area of RAM.

System Action: Continues on to next test phase of DMA test.

Operator Response: See Problem Determination Procedures.

WOEM6538 DMA TEST FAILS WHILE READING FROM PAGE 7

Explanation: Adapter fails to move data from specific area of RAM.

System Action: Continues on to next test phase of DMA test.

Operator Response: See Problem Determination Procedures.

IOEM6539 OEMI DMA TEST IGNORED

Explanation: User chose not to run this test section.

System Action: Continues on to next test.

IOEM6540 DMA TEST PASSES WHILE READING FROM PAGE 5

Explanation: Adapter is able to move data from specific area of RAM.

System Action: Continues on to next test phase of DMA test.

IOEM6541 DMA TEST PASSES WHILE READING FROM PAGE 6

Explanation: Adapter is able to move data from specific area of RAM.

System Action: Continues on to next test phase of DMA test.

IOEM6542 DMA TEST PASSES WHILE READING FROM PAGE 7

Explanation: Adapter is able to move data from specific area of RAM.

System Action: Continues on to next test phase of DMA test.

WBUS9000 TIMER 0 ERROR DETECTED

Explanation: A test of the timer chip on the DMA/Memory has found an error with section 0 of this chip.

System Action: N/A

Operator Response: See Problem Determination Procedures.

WBUS9001 TIMER 1 ERROR DETECTED

Explanation: A test of the timer chip on the DMA/Memory has found an error with section 1 of this chip.

System Action: N/A

Operator Response: See Problem Determination Procedures.

WBUS9002 TIMER 2 ERROR DETECTED

Explanation: A test of the timer chip on the DMA/Memory has found an error with section 2 of this chip.

System Action: N/A

Operator Response: See Problem Determination Procedures.

WBUS9003 DMA ERROR DETECTED

Explanation: A test of the DMA chip on the DMA/Memory has found an error.

System Action: N/A

Operator Response: See Problem Determination Procedures.

WBUS9004 SERIAL ERROR, SIOCCTL ERROR

Explanation: A partial test of the serial has found an error.

System Action: N/A

Operator Response: See Problem Determination Procedures.

IBUS9010 INDICATOR TEST, OBSERVE AND NOTE PATTERN PRESS ANY KEY WHEN READY TO START TEST

Explanation: The DACU is ready to start writing a series of patterns to the indicators located inside the front door.

System Action: The DACU will wait for the operator to press a key and then start displaying patterns in the indicators. The sequence of patterns is : 0001, 0002, 0003, 0004, ..., 0009, 0010, 0020, ... 9000.

Operator Response: Observe pattern, and if error indicated, then see Problem Determination Procedures.

IBUS9011 INDICATOR TEST COMPLETE

Explanation: Indicator testing is complete.

System Action: The DACU will continue diagnostic procedures.

IBUS9012 PRESS ANY KEY WHEN READY TO START TEST

Explanation: Prompt in order to synchronize with user.

System Action: The test will start after user presses enter.

IBUS9051 STARTING BUS TEST

Explanation: BUS test is starting.

System Action: The DACU will start execution of the bus tests.

IBUS9052 END OF BUS TEST

Explanation: BUS test is complete (bring-up).

System Action: The DACU will continue diagnostic procedures.

IBUS9053 END OF EXTENDED BUS TEST

Explanation: BUS test is complete (extended).

System Action: The DACU will continue diagnostic procedures.

ITDM9100 STARTING TIMER AND DMA TESTS

Explanation: Beginning code for timer and DMA tests.

System Action: Continues on to test.

ITDM9101 TIMER AND DMA TESTS COMPLETE

Explanation: End of code for timer and DMA tests.

System Action: Ends code.

ITDM9102 FOUND NO AUTO INIT OF DMA CHIP

Explanation: Values were obtained from DMA chip and should have decremented in value to zero and autoinitialized to stored value but didn't.

System Action: Continues on to next test.

ITDM9103 AFTER AT LEAST ONE AUTO INIT OF DMA CHIP

Explanation: Values obtained from DMA chip indicated an autoinitialization occurred.

System Action: Continues on to next test.

WTDM9104 VALUES FROM DMA CHIP NOT DECREMENTING - SAMPLE VALUES ARE XXXX AND YYYY

Explanation: Values obtained from the DMA chip indicated no decrementing occurred. Either timer chip not requesting refreshes or DMA chip is not doing refreshes.

System Action: Displays two values obtained from DMA chip.

Operator Response: See Problem Determination Procedures.

WTDM9105 VALUES FROM TIMER CHIP INDICATED AN AUTOINITIALIZATION DID NOT OCCUR

Explanation: values obtained from timer chip did not decrement to zero and reinitialize.

Operator Response: See Problem Determination Procedures.

WTDM9106 VALUES FROM DMA CHIP INDICATED AN AUTOINITIALIZATION DID NOT OCCUR

Explanation: Values obtained from DMA chip did not decrement to zero and reinitialize.

System Action: Continues on to next test.

Operator Response: See Problem Determination Procedures.

ITDM9107 INITIALIZATION OF TIMER AND DMA CHIPS COMPLETE

Explanation: Timer and DMA chips set up to refresh ram.

System Action: Continues on to next test.

ITDM9108 START OF REFRESH CHECK TEST

Explanation: Manual intervention test for checking validity of refresh.

System Action: Continues on to test.

ITDM9109 THIS CAN TAKE UP TO 2 MINUTES DUE TO DATA RETENTION OF THE DYNAMIC RAMS

Explanation: Dynamic rams will be used to see if refresh is working.

System Action: Continues on to next test.

ITDM9110 END OF REFRESH CHECK TEST

Explanation: Code ends for dynamic ram refresh test.

System Action: Ends test.

ITDM9111 TIMER TEST PASSED

Explanation: Values from timer chip indicated timer was functioning ok.

System Action: Continues on to next test.

ITDM9112 DMA TEST PASSED

Explanation: Values from DMA chip indicated DMA was functioning OK for channel 0 which is refresh.

System Action: Continues on to next test.

WTDM9113 VALUES FROM TIMER NOT DECREMENTING - SAMPLE VALUES ARE XXXX AND YYYY

Explanation: Values obtained from timer chip indicate timer not decrementing.

System Action: Continues on to next test.

Operator Response: See Problem Determination Procedures.

WTDM9114 TIMER CHANNEL 0 TEST HAS FAILED

Explanation: Either the values obtained from timer 0 weren't decrementing or it didn't do an auto-initialize.

System Action: Continues on to next test.

Operator Response: See Problem Determination Procedures.

WTDM9115 DMA CHANNEL 0 TEST HAS FAILED

Explanation: Either the values obtained from DMA channel 0 weren't decrementing or it didn't do an autoinitialize.

System Action: Ends test.

Operator Response: See Problem Determination Procedures.

ITDM9116 RAM FAILED WITHOUT REFRESH IN xx HALF-MINUTE(S)

Explanation: Refresh was turned off and the dynamic ram failed in xx half minute intervals.

System Action: Continues on to next test.

WTDM9117 RAM FAILED WITH REFRESH ON AT START OF TEST

Explanation: Ram is filled with a value and then checked to make sure it is working with refresh on. If it fails this basic test, this error message is printed.

System Action: Ends test.

Operator Response: See Problem Determination Procedures.

WTDM9118 RAM HASN'T FAILED WITH REFRESH TURNED OFF

Explanation: Turned refresh off and ram didn't fail.

System Action: Ends test.

Operator Response: See Problem Determination Procedures.

WTDM9119 TURNED REFRESH OFF THEN ON AND RAM FAILED

Explanation: After a time was determined in which the ram would fail without refresh, refresh was again turned on, ram initialized and checked and found to be in error.

System Action: Ends test.

Operator Response: See Problem Determination Procedures.

WTDM9120 TURNED REFRESH OFF THEN ON AND WAITED TWICE THE REFRESH WAIT TIME AND RAM FAILED

Explanation: After a time was determined in which the ram would fail without refresh, refresh was again turned on, ram initialized and then waited twice the refresh wait/fail time, checked ram and it was found to be in error.

System Action: Ends test.

Operator Response: See Problem Determination Procedures.

ITDM9121 EXPECTED VALUE WAS xxxx AND ACTUAL VALUE IS yyyy

Explanation: If there is an error in the test, the values will be displayed as a visual aid to debug.

System Action: Continues on to next test.

ITDM9122 WAITING AND CHECKING RAM

Explanation: After a time is found in which the ram does not retain it's pattern without refresh, refresh is turned on and the program waits twice as long as the no refresh fail time. Then the ram is checked. This message is intended to let the operator know the machine is working, and not hung up(yet).

System Action: Continues on to next test.

ITDM9123 REFRESH TEST PASSED

Explanation: Refresh test has passed all checks.

System Action: Ends test.

IMEM9200 STARTING RAM TEST; ADDRESSES AND DATA RIPPLE...

Explanation: Indicates start of memory diagnostics.

System Action: DACU starts test.

IMEM9202 STARTING EXTENDED DIAGNOSTICS FOR RAM ....

Explanation: Extended test was selected from the menu.

System Action: DACU starts extended testing of ram.

IMEM9203 DATA COMPARES COMPLETED; RAM PASSED

Explanation: Bring-up diagnostics completed.

System Action: DACU continues to next diagnostic test.

WMEM9204 DATA COMPARE FAILED AT LOCATION

Explanation: Bad data is read from the ram at the address given.

System Action: Flags error and continues to next test.

Operator Response: See Problem Determination Procedures.

IMEM9205 DATA BUS RIPPLE TEST PASSED

Explanation: First section of bring-up diagnostics completed.

System Action: DACU continues to next test.

IMEM9206 DATA COMPARES COMPLETED; RAM PASSED

Explanation: Extended diagnostics completed.

System Action: DACU continues to next test.

WMEM9207 DATA COMPARE FAILED AT LOCATION

Explanation: Bad data is read from ram at the address given.

System Action: Flags error and continues to next test.

Operator Response: See Problem Determination Procedures.

WMEM9208 DATA BUS ERROR, MEMORY ADDRESS 5E000

Explanation: Data ripple test failed. Test was trying to read and write all possible bit patterns to one memory location with an even address.

System Action: Flags error and continues to next test.

Operator Response: See Problem Determination Procedures.

WMEM9209 DATA BUS ERROR, MEMORY ADDRESS 5E001

Explanation: Data ripple test failed. Test was reading and writing all possible bit patterns to one memory location with an odd address.

System Action: Flags error and continues to next test.

Operator Response: See Problem Determination Procedures.

Index

F

G

H

I

J

K

L

M

V

W

Numerics

7170 Device Attachment Control Unit
Reference and Operations Manual
SA24-4025-0

READER'S COMMENT FORM

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Your comments will be sent to the author's department for whatever review and action, if any, are deemed appropriate. Comments may be written in your own language; English is not required.

Note: *Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.*

Note: Staples can cause problems with automated mail sorting equipment. Please use pressure sensitive or other gummed tape to seal this form.

	Yes	*No*
• Does the publication meet your needs?	☐	☐
• Did you find the material:		
Easy to read and understand?	☐	☐
Organized for convenient use?	☐	☐
Complete?	☐	☐
Well illustrated?	☐	☐
Written for your technical level?	☐	☐

• What is your occupation? _____

• How do you use this publication:

As an introduction to the subject?	☐	As an instructor in class?	☐
For advanced knowledge of the subject?	☐	As a student in class?	☐
To learn about operating procedures?	☐	As a reference manual?	☐

Your comments:

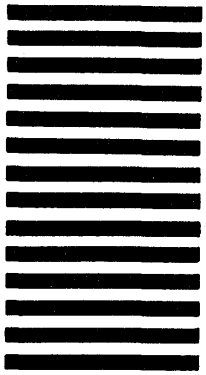
If you would like a reply, please supply your name and address on the reverse side of this form.

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page.)

Reader's Comment Form

Fold and Tape | Please Do Not Staple | Fold and Tape

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES



BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE:

International Business Machines Corporation
Department X00
P. O. Box 6
Endicott, New York 13760

Fold | Fold

If you would like a reply, *please print:*

Your Name _____

Company Name _____ *Department* _____

Street Address _____

City _____

State _____ *Zip Code* _____

IBM Branch Office serving you _____

IBM®

Cut or Fold Along Line

7170 Device Attachment Control Unit Reference and Operations Manual (File No. S370/4300-09) Printed in U.S.A. SA24-4025-0

7170 Device Attachment Control Unit Reference and Operations Manual (File No. S370/4300-09) Printed in U.S.A. SA24-4025-0

SA24-4025-0

SA24-4025-0

7170 Device Attachment Control Unit Reference and Operations Manual (File No. S370/4300-09) Printed in U.S.A. SA24-4025-0

IBM®