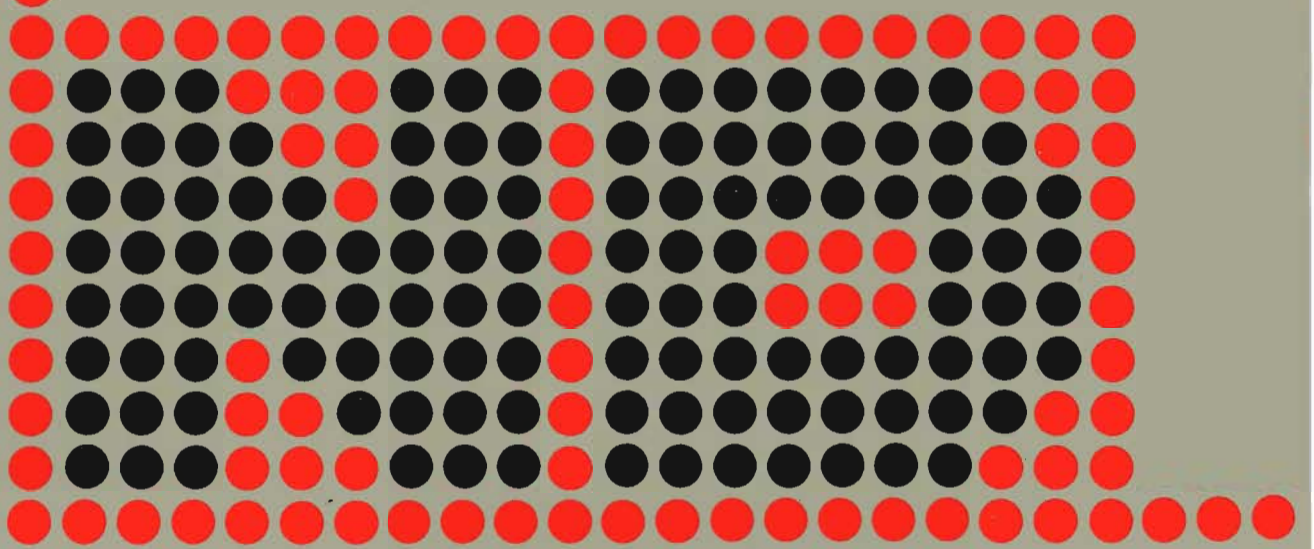
Norsk Data

General Purpose Interface Bus (GPIB) User Guide

ND-12.023.02



General Purpose Interface Bus (GPIB) User Guide

ND-12.023.02

NOTICE

The information in this document is subject to change without notice. Norsk Data A.S assumes no responsibility for any errors that may appear in this document. Norsk Data A.S assumes no responsibility for the use or reliability of its software on equipment that is not furnished or supported by Norsk Data A.S.

PRINTING RECORD

Printing	Notes
11/82	Version 01
08/84	Version 02

General Purpose Interface Bus
(GPIB) User Guide
Publ.No. ND-12.023.02
August 1984

NORSK DATA A.S
P.O. Box 25, Bogerud
0621 Oslo 6, Norway

Manuals can be updated in two ways, new versions and revisions. New versions consist of a complete new manual which replaces the old manual. New versions incorporate all revisions since the previous version. Revisions consist of one or more single pages to be merged into the manual by the user, each revised page being listed on the new printing record sent out with the revision. The old printing record should be replaced by the new one.

New versions and revisions are announced in the ND Bulletin and can be ordered as described below.

The reader's comments form at the back of this manual can be used both to report errors in the manual and to give an evaluation of the manual. Both detailed and general comments are welcome.

These forms, together with all types of inquiry and requests for documentation should be sent to the local ND office or (in Norway) to:

Documentation Department
Norsk Data A.S
P.O. Box 25, Bogerud
0621 Oslo 6, Norway

Preface:

THE PRODUCT

The ND system controller for the General Purpose Interface Bus (GPIB) is used to connect and control programmable instrumentation and non-programmable instrumentation like plotters and printers. These peripheral devices must be compatible with the GPIB standard. The ND system controller consists of advanced hardware and software. This gives the users effective control of the system.

THE READER

This manual is in general intended for both new and experienced users of a GPIB system. It is particularly intended for programmers who use FORTRAN programs to control devices on GPIB. The manual is also intended for operators who want to control devices on GPIB without programs, but through a set of powerful commands.

PREREQUISITE KNOWLEDGE

The reader should have some basic knowledge of ND's operating system - SINTRAN III. Programmers must know how to create and run programs on an ND computer. The reader need not have basic knowledge of the GPIB standard to understand this manual. Basic understanding of digital technology is recommended to understand the short description of the GPIB standard.

THE MANUAL

This manual mainly contains information about how to use the user-oriented software in the product. It also contains a short description of the GPIB standard. It is mainly oriented sequentially. Some of the manual is structured to be used as reference when the reader gets familiar with the product. Examples are used to give the reader training and better understanding of the product the first time reading this manual. Important explanations, like notes and communication between the user and the computer, are put into boxes. The characters typed by the user on the terminal are underlined. Special keys typed by the user are presented symbolically, ie., "↵" means the key frequently named "carriage return". Numbers which not are presented in the decimal system are appended radixs to avoid confusion. I.e, 12_8 means the octal number 12, which is equal to the decimal number $10.^8$

RELATED MANUALS

Users who not are familiar with ND's operating system SINTRAN III find useful information in these manuals:

SINTRAN III	INTRODUCTION	ND 60.125
SINTRAN III	TIMESHARING GUIDE	ND 60.132
SINTRAN III	REFERENCE MANUAL	ND 60.128

Programmers who do not know how to create and run a FORTRAN program on an ND computer find useful information in the manual:

ND FORTRAN REFERENCE MANUAL ND 60.145

All users who need general and detailed information about the GPIB standard should read a official publication describing this. The official GPIB standards are IEEE-488 and IEC-625.

The user should thoroughly read the documentation of all the devices used in the GPIB-system, because many operating facilities are device-dependent.

TABLE OF CONTENTS

Section Page

Section Page

APPENDIX

LIST OF ILLUSTRATIONS

1 INTRODUCTION

1.1 What is the Advantage of the GPIB?

The main function of the General Purpose Interface Bus is to create an effective communication link between the devices on the bus where data can be exchanged.

The devices on the bus can be programmable instrumentation such as logic analyzers, signal generators and digital multimeters. There can also exist devices on the bus which are non-programmable instrumentation, ie., plotters and printers.

There are many advantages with a common interface bus between instrumentation devices, when we have measurements:

- requiring high reproducibility and accuracy.
- requiring simultaneous testing of input and output characteristics.
- requiring immediate further processing of the measured data to allow decision-making.
- which are extremely diversified or constantly recurring.
- which have many parameters.
- which only have a few interesting results.

The system controller is the managing and sequencing device on the interface bus. It decides when other devices shall transmit and receive data. It is also able to select a group of all the devices on GPIB for exchange of data.

ND's system controller consists of both hardware and software. The hardware is a card called the GPIB controller. This card is located in the computer's card rack. Many users can use the system controller simultaneously. Implemented software handles interaction between the users and the GPIB controller card. The software is implemented as a SINTRAN subsystem. This make the GPIB controller easy to use. It is also possible to have more than one GPIB controller card in a computer. This makes it possible to extensively expand your instrumentation system. These facilities are more closely described in the next sections. Fig. 1. shows a instrumentation system with one controller card.

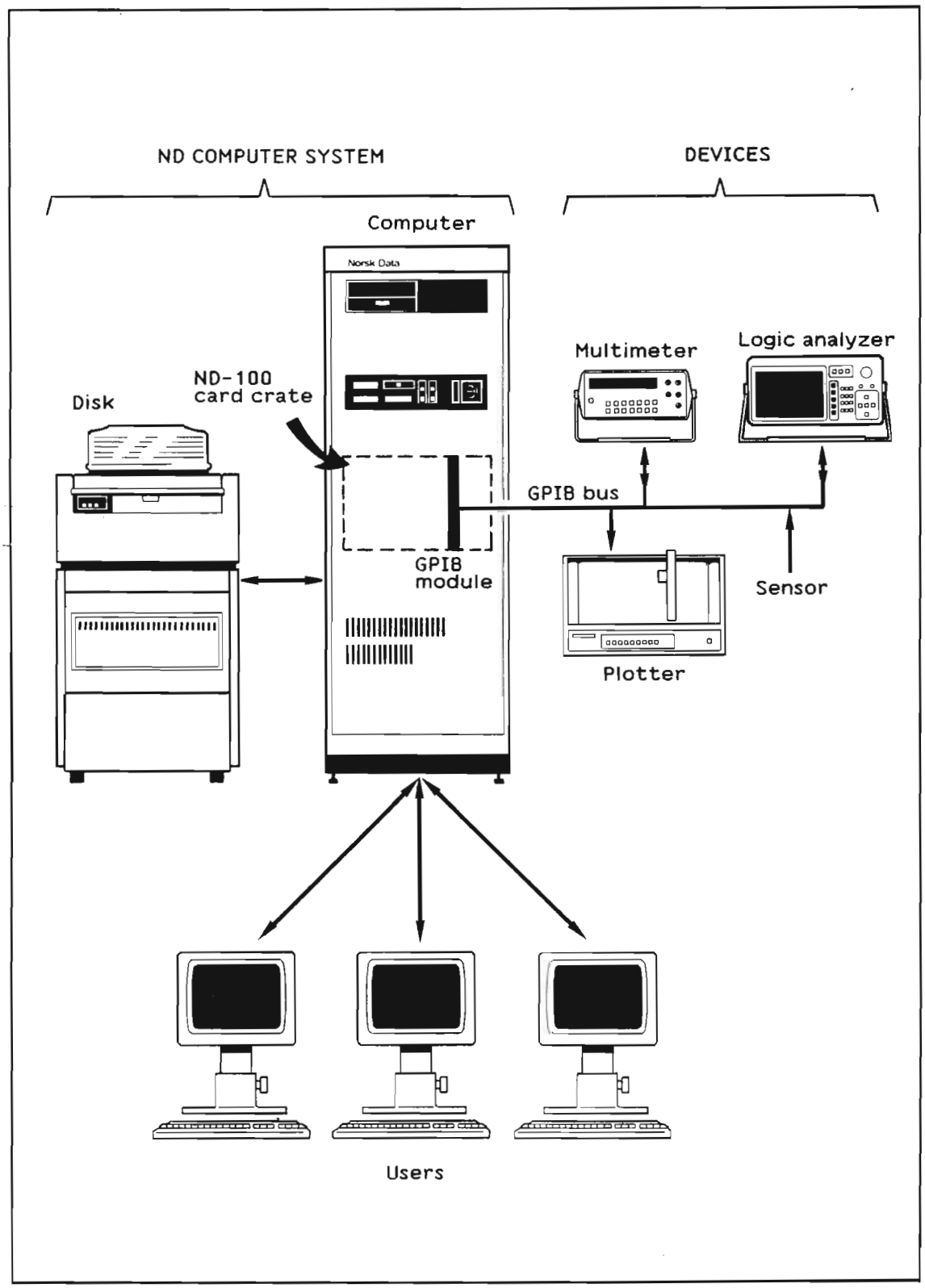


Fig. 1. GPIB Instrumentation Example on an ND Computer.

ND's system controller follows the standards outlined in the IEEE-488 and IEC-625 standards. These standards are also referred to as HP-IB, IEEE-bus and IEC-bus by other companies. If your instrumentation devices are mentioned as having one of these interface buses, you should be able to use them together with ND's system controller.

1.2 Listening and Talking Devices on GPIB

The interface bus is common for all devices. To be able to select only one device for exchange of data, device addresses must exist. Each device has its own addresses for talking and listening. The device addresses are set on the device with separate switches, or via an operators panel.

Four types of devices can exist on the interface bus in a specific configuration, as seen from the interface bus. These four device types are based on the basic functions a device can perform - listen and talk.

A device acting as a listener is able to receive data from the bus when addressed by the controller. The received data can, for example, be used to make an output on external lines (analogous or digital), corresponding to the received data. Such a device could be a plotter or a signal generator. A device which is able to listen can be configurated by the controller.

A device acting as a talker is able to transmit data onto the bus when addressed by the controller. The transmitted data can, for example, be the result of a measurement. Such a device could be a temperature measuring system. A device only able to talk cannot be configurated from the controlling device.This must be done separately on the respective device, for example by switches or via an operators panel.

A device acting as both a talker and a listener is able to perform both of the above functions when addressed, but the device is not able to both talk and listen simultaneously. Received data could be used to select different kinds of measurements in the device. Transmitted data could be a result of a measurement. Such a device could be a digital multimeter.

A device acting as the controller is able to both talk and listen. It is the sequencing part in the system, meaning that it is the only device able to:

- configurate other devices.

- address other devices for listening and talking.

ND's system controller must also know the proper device addresses. This is ensured when the configuration of a device is put into the controller. The controller manages the device addresses, and you will instead remember a device by a symbolic device name and a device number.

Only one active controller may exits at a time (which addresses devices for listening and talking). The GPIB standard defines two main categories of controllers - system controller and controller-in-charge. The system controller is the most powerful of these functions. A device acting as controller-in-charge takes most of the control from the system controller device, except for a few powerful commands. ND's system controller is always both system controller and controller-in-charge. It can not pass any control function to another device.

Fig. 2. shows an example on devices acting as listeners and talkers on the interface bus.

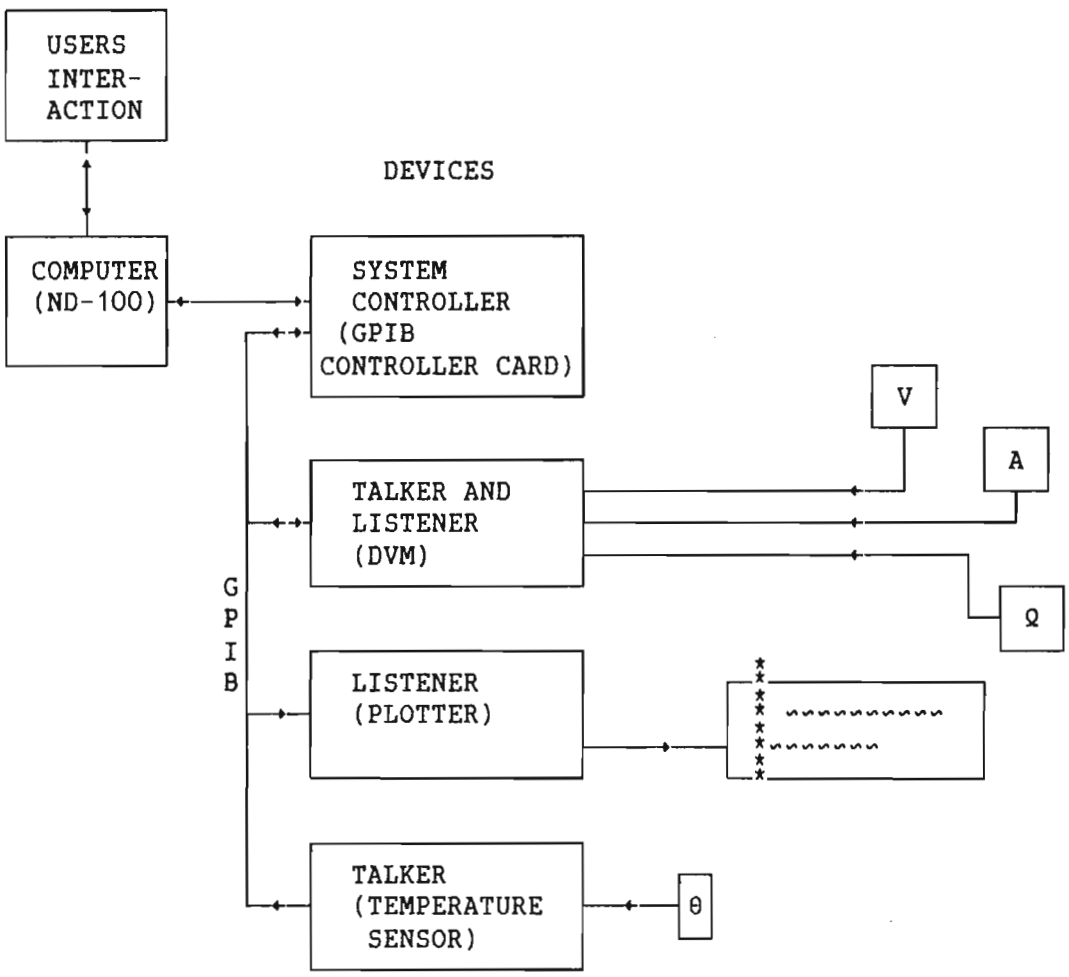


Fig. 2. Example of Listeners and Talkers on GPIB

1.3 The GPIB Controller Card

The GPIB controller card is quite independent of the rest of the computer system. This is obtained its own microprocessor and software being located on the card. This software communicates directly with the SINTRAN subsystem for GPIB, through a communication system called XMSG.

However, this communication is an invisible part of the GPIB- system as seen from the user's point of view. You need only be concerned with how to use the software implemented in the SINTRAN subsystem for GPIB

The microprocessor controls the interface bus on which the devices are connected. This make it possible to avoid too many interrupts to the computer's CPU. The result is a significantly reduced load onto the computer's bus. This means that the computer is free to be used in other timesharing jobs most of the time.

The GPIB-system on an ND-computer is a multi-user system. This makes it possible for many users to perform different instrumentation tasks simultaneously. A device connected to the bus can either be used by many users simultaneously, or be reserved for one user. This depends on the facilities in the device.

A controller card can be used by as much as 16 users simultaneously. Each user on the controller gets a user number when logging on. This is a number between 0 and 15. User number 0 is privileged. The user who logs on and gets this user number, is responsible for configuration of all devices connected to the controller card. This means that this user is privileged. User SYSTEM in SINTRAN is a privileged user in the standard version of GPIB, but it is also possible to select another privileged user (see chapter 3 for an explanation of how to do this.)

A complete GPIB-system on an ND computer can consist of as much as 8 controller cards. Each controller card has a controller number. This is a number between 0 and 7. In this way you are able to log onto different controllers simultaneously.

The microprocessor on the card takes care of creating new users, deleting old users, reserving devices for a user, enabling for a new device that has been connected to the bus, deleting old devices, etc. It also tests the commands from the user according to device list, device configuration, device status, reserved devices, etc.

The microprocessor on the card makes a list of all devices in the system. For each device, this list contains information about listener address, talker address, configuration of the device, end of string character, and who the device's users are. This list is located in RAM on the controller card.

A copy of this list is also placed in the secondary storage of the computer. This file is named the GPIB Configuration File. When a cold start is performed on the computer, the controller card is cleared. The GPIB configuration file can then be read, and new and unnecessary configuration of the system is avoided. The format of this file is shown i fig. 3.

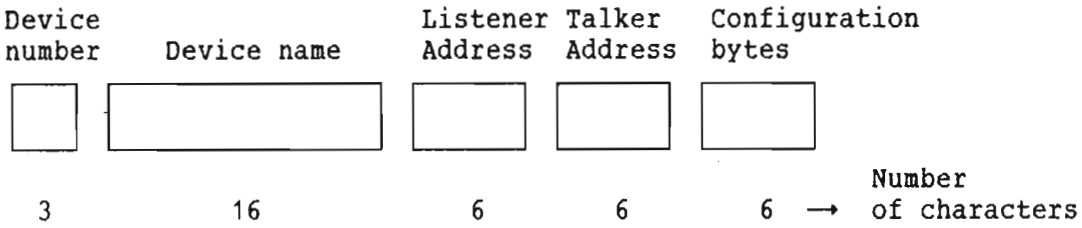


Fig. 3. Format of the Configuration File

1.4 Implemented Software

The interaction between the users and a controller card is performed by software implemented in a SINTRAN subsystem for GPIB.

The software consists of a MONITOR and a LIBRARY. The MONITOR consists of commands. The LIBRARY consists of callable routines from FORTRAN programs.

The software can handle both the GPIB standarized software and device-dependent software within a device. The device dependent software is handled by general commands/routines for sending and receiving of data. It is very important that you are familiar with the devices' facilities, both for operation and configurion of devices.

When a nonprivileged user wants to use the system, he/she has to:

1) Ask to be logged on a controller as user.

2) Reserve the needed devices.

3) Execute the desired commands/calls on the reserved devices.

4) Release devices as soon as they are no longer needed, to enable for new users.

5) Ask to be logged off the controller as user.

Fig. 4. visualizes the implemented software.

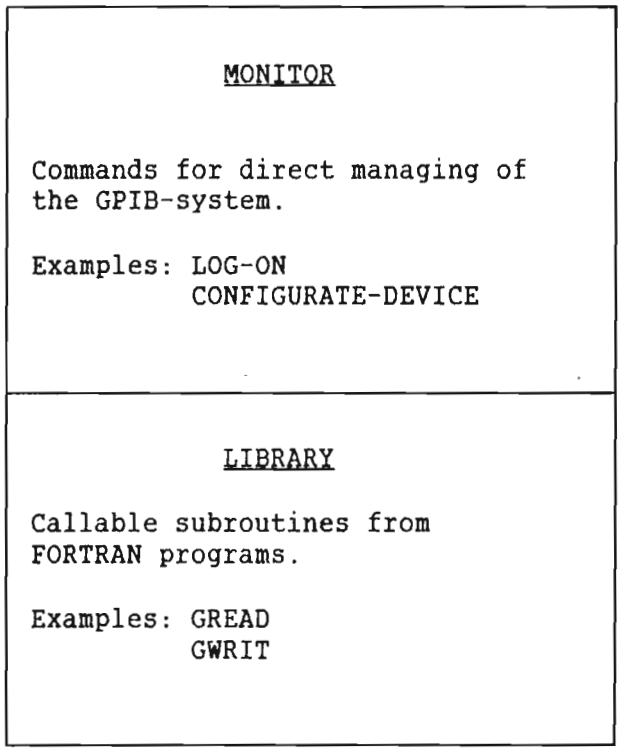


Fig. 4. Implemented software

2 SHORT DESCRIPTION OF THE GPIB-STANDARD

2.1 Introduction

This chapter contains only a short description of the public standard for the GPIB. It is intended to give new users of a GPIB- system an introduction to this standard. Users who need a more detailed description should read a separate publication describing this.

2.2 Bus Structure

The interface bus between the devices consists of a set of physical lines where all data and signals exchanged between devices must pass. The signals on the bus follow the standard for negative logic. That means the signal's low state is the active one. When a signal is sent in the active state, it is said to be sent true. The bus signals are shown in fig. 6.

2.2.1 Data Bus - (DIO1 - DIO8)

The bidirectional bus of eight lines makes it possible to send data in serial or byte representation. DIO1 is the least significant bit, DIO8 is the most significant bit.

2.2.2 Handshake Lines

These three lines make it possible to ensure that data flow between all devices on the bus is performed in an unambiguous way. A timing diagram for the handshake cycles is shown in fig. 5.

Data Valid - DAV

This signal indicates that data is available and valid from a device acting as a talker.

Not Ready For Data - NRFD

This signal indicates that device(s) acting as listener(s) are not ready to receive data.

Not Data Accepted - NDAC

This signal indicates that device(s) acting as listener(s) have not accepted data from the talking device.

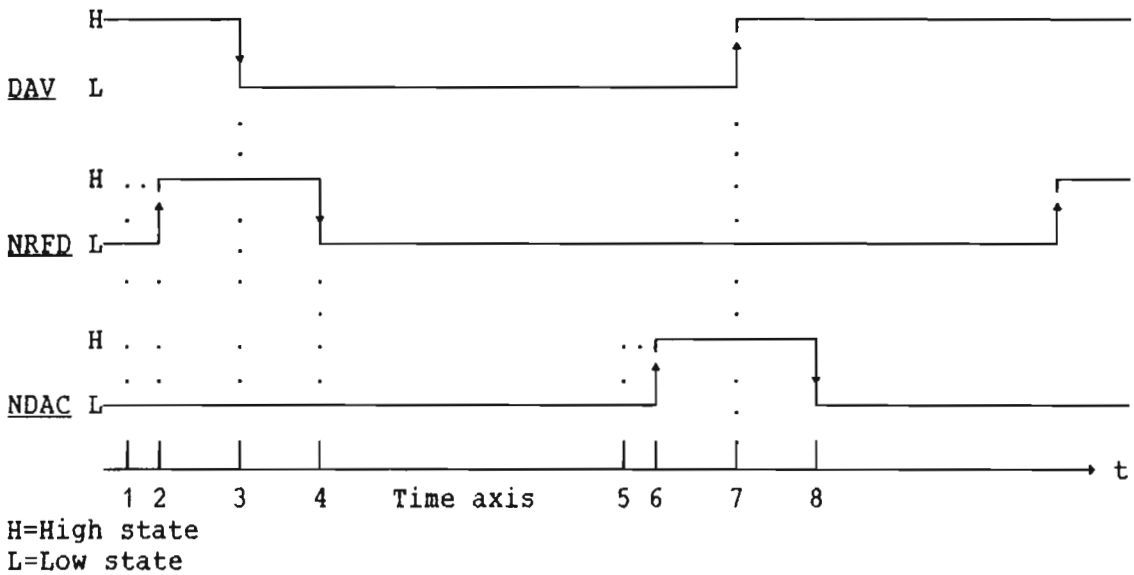


Fig. 5. Handshake timing Diagram

Handshake cycle as shown in fig. 5 :

1) DAV is false, indicating no data valid. NDAC is true. Some listeners set their NRFD false; they are ready to accept data.

2) The NRFD is false; all listeners are ready to accept data.

3) The talker sets DAV true; valid data is on the bus.

4) The NRFD is true; listeners are reading data.

5) Some listeners set their NDAC false; they have accepeted data.

6) The NDAC is false; all listeners have accepted data.

7) The talker sets DAV false; data transfer is completed.

8) The NDAC is true. A new data transfer can start.

2.2.3 Interface Management Bus

These signals are used to manage the flow of information across the interface.

Attention - ATN

This signal is sent from the the device acting as the controller-in-charge, and tells devices how to respond to the data received on the data lines (DIO1 - DIO8).

End Of Identify - EOI

This signal is used by a device acting as talker, to indicate the end of a multiple data transfer. Together with ATN it is the signal used by the controller-in-charge to execute a parallel poll.

Service Request - SRQ

This signal is sent to the device acting as the controller-in-charge, and tells that a device needs service.

Interface Clear - IFC

This signal can only be sent from the device acting as the system controller. It puts all device interfaces on the bus in a predefined quiescent state.

Remote Enable - REN

This signal is only sent from the device acting as the system controller, and selects whether the devices are operated by remote or by local controls. Remote controls mean that a device can receive data and commands over the bus. Local controls mean that a device it operated from controls located on the device, for example an operators panel. When the REN line is reset by the system controller, no other messages can be exchanged between devices on the bus.

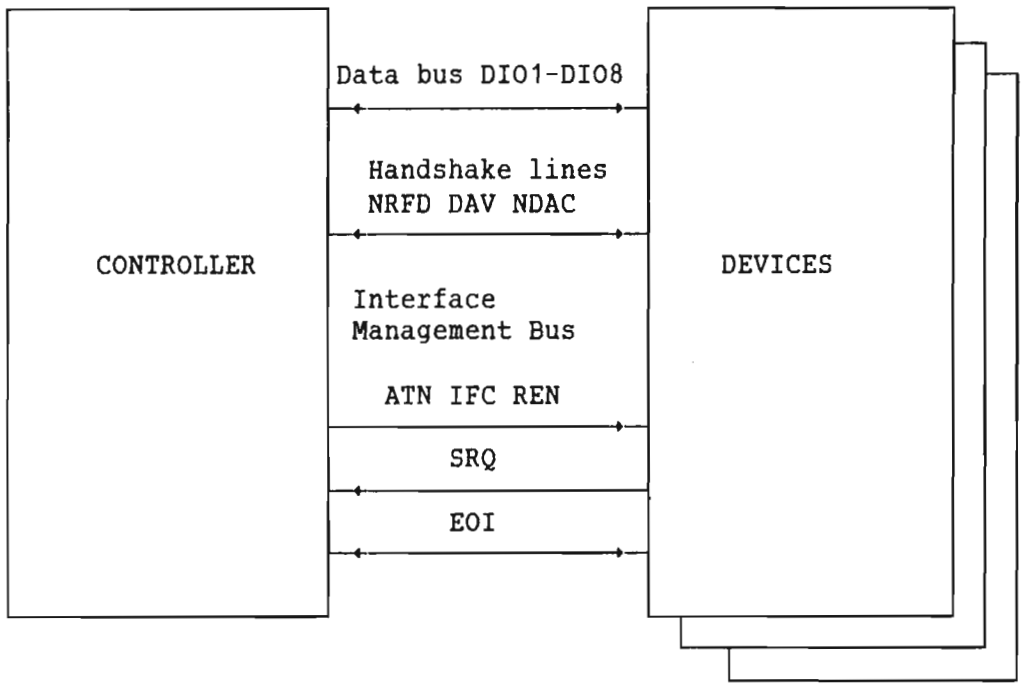


Fig. 6. Bus Signals

2.3 Functions within a Device

Functions within a device can be divided into three functional groups (see also fig. 7):

- Device functions.
- Interface functions.
- Message coding.

2.3.1 Device functions

The device functions are device-dependent. They depend upon external functions which a device is performing, ie., measuring analogous parameters, making analogous outputs, etc.

2.3.2 Interface functions

An interface function is the basic element that must exist to be able to receive, process and send messages.

The total number of interface functions, and how they work, is defined in the IEEE- and IEC-standard. The interface functions in a specific device will be a subset of these, and not necessarily all of them. Only the main set of interface functions are shown in fig. 7.

Each interface function can further be divided into defined states. However, this is not described in this manual.

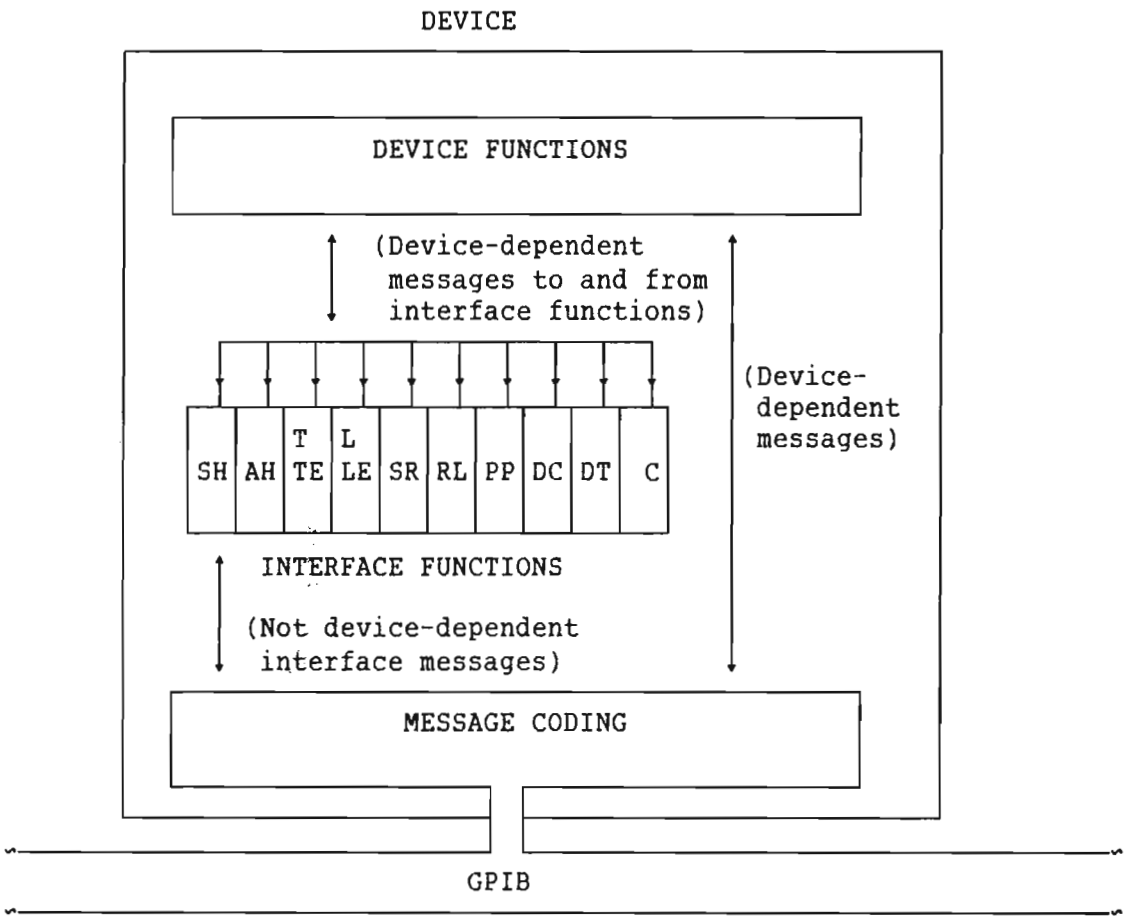


Fig. 7. Functions within a Device

2.3.3 Messages

A message is defined to be a quantity of information transferred inside a device or between devices. It can consist of a signal sent on one or more lines. Messages sent on one line are called single-line messages. Messages sent on several lines are called multi-line messages. The messages are further classified as:

- Local messages which are sent within a device (between device and interface functions).

- Remote messages which are sent between devices. They can either be interface messages or device-dependent messages.

The ATN line on GPIB has a special purpose for decoding of multi-line messages. When the ATN line is false, messages are treated as device-dependent. When the ATN line is true, messages are treated as interface messages.

The interface messages can be classified as:

1) Universal commands (can be sent to all devices independent of interface function states).

2) Addressed commands (can be sent to all devices previously addressed to listen).

3) Addresses (can be sent to all devices).

4) Secondary addresses or commands (can be sent to all devices enabled by an earlier address or command).

Some of the messages are referred to in the next sections, and in the description of the software implemented in ND's system controller. The effect these messages have on devices will then be commented. Appendix E in this manual lists all messages that can be transferred over the data bus with ND's GPIB system.

2.4 Short Description of each main Interface Function

2.4.1 Interface Function Overview

The main interface functions and their mnemonics are listed in the table below. The name of a interface function is obtained by appending a number after the main name. For example C1 means subset 1 of the controller function.

Interface function	Mnemonics
Acceptor Handshake	AH
Source Handshake	SH
Talker (or Talker Extended)	T (TE)
Listener (or Listener Extended)	L (LE)
Service Request	SR
Remote or Local	RL
Parallel Poll	PP
Device Clear	DC
Device Trigger	DT
Controller	C

2.4.2 Source Handshake - SH

The SH function enables a device to guarantee the proper transfer of multi-line messages. An interlocked handshake sequence between the SH function and one or more Acceptor Handshake functions, (each contained within separate devices), guarantees asynchronous transfer of each multi-line message. The SH function controls the initiation and the termination of the transfer of a multi-line message byte. This function uses the DAV, RFD and DAC messages (sent on the handshake lines) to transfer each byte. The handshake timing was described in section 2.2.2.

2.4.3 Acceptor Handshake - AH

The AH function enables a device to guarantee proper reception of remote multi-line messages. An interlocked handshake sequence between an SH function and one or more AH functions, (each contained within separate devices), guarantees asynchronous transfer of each message byte. An AH function may delay either the initiation or the termination of a multi-line message transfer, until prepared to continue with the transfer process. The AH function uses the DAV, RFD and DAC messages (sent on the handshake lines) to transfer each byte.

2.4.4 Talker - T (or TE)

The T function enables a device to send device- dependent data (including status information during a serial poll sequence) over the interface bus, to other devices. The capability only exists when the device has previously been addressed to talk.

There are two alternative versions of the functions: one with, and one without address extension. The normal T function uses a one-byte address. The T function with address extension, TE, uses a two-byte address. In all other conditions, the capabilities of both versions are the same.

Only one of the two alternatives, T or TE, can exist in a given device.

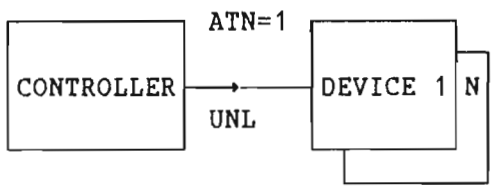
2.4.5 Listener - L (or LE)

The L function enables a device to receive device- dependent data (including status information) over the interface bus, from other devices. This capability only exists when the device has previously been addressed to listen.

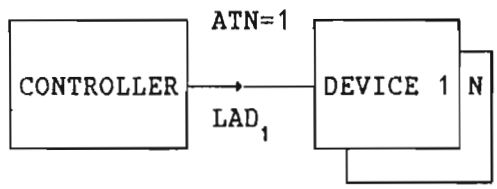
There are two alternative versions of the function: one with, and one without address extension. The normal L function uses a one-byte address. The L function with address extension, LE, uses a two-byte address. In all other conditions, the capabilities of both versions are the same.

Only one of the two alternatives, L or LE, can exist in a given device.

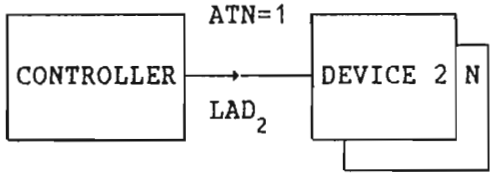
Example of a data transfer:



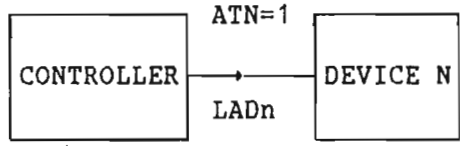
1. The controller sends the unlisten command to inhibit all previously enabled listeners.



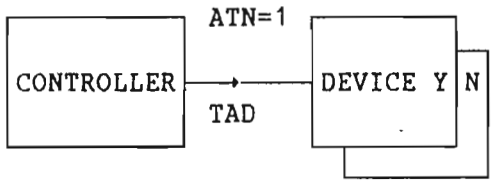
2. The controller enables the first device for listening.



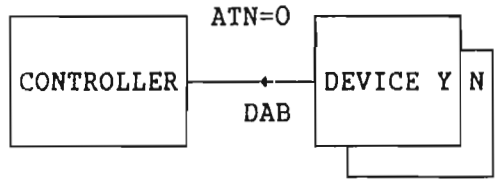
3. The controller enables for the second listener.



4. The controller enables for the n'th listener.



5. The controller addresses device Y for talking.



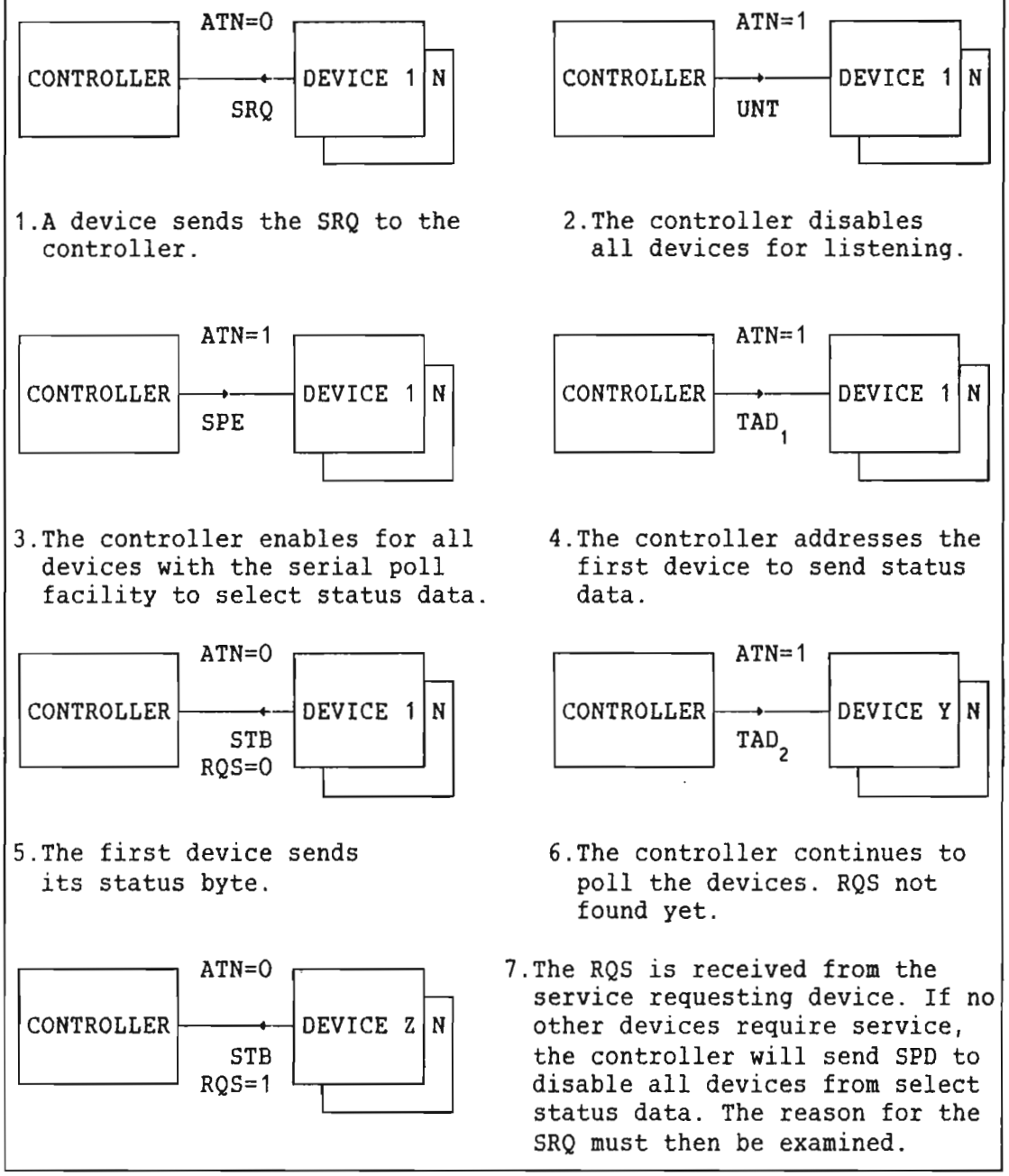
6. The talker sends the first data byte. This sequence can be repeated until the talker terminates the transfer by setting EOI-line or sending EOS, or until the controller sets ATN-line to 1.

2.4.6 Service Request - SR

The SR function enables a device to request service asynchronously from the controller-in-charge of the interface bus.

It also synchronizes the transfer of the RQS (Request Service) message of the composite status byte during a serial poll, so that the SRQ (Service Request) message can be removed from the interface bus once the RQS message is received true by the controller-in-charge.

Example of a serial poll:



2.4.7 Remote or Local - RL

The RL function enables a device to select between two sources of input information. The function indicates to the device whether input information shall be received from local controls or from the interface bus. The local controls can, for instance, be located on the device's front panel.

2.4.8 Parallel Poll - PP

The PP function enables a device to send a PPR (Parallel Poll Response) message to the controller-in-charge, without having previously been addressed to talk, but enabled by a IDY (Identify) message. The IDY message is EOI and ATN set to true. This was also mentioned in section 2.2.3.

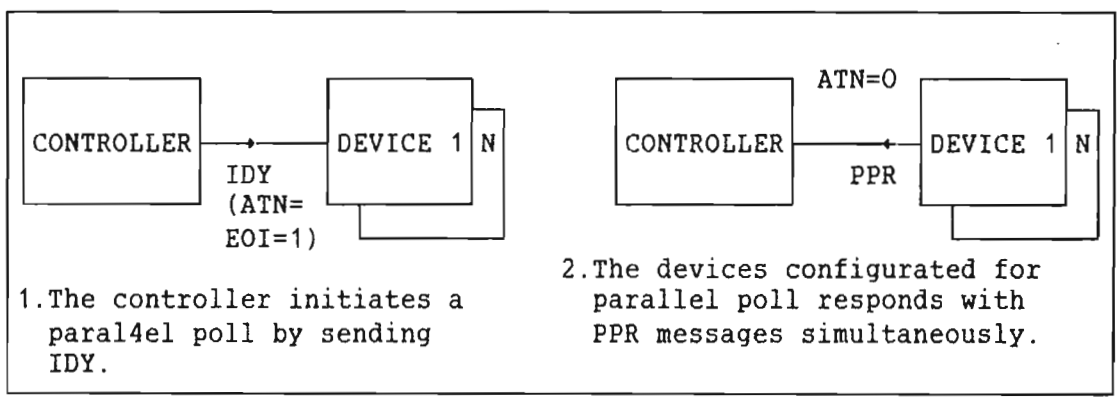
The data lines on the interface bus (D1-D8) are used to transfer the device's status information during a parallel poll. A device shall only send the PPR message as a single-line message. This makes it possible for as much as eight devices to have their own line. Additional devices must share a line.

The use of the parallel poll facility within a system requires that the controller-in-charge respond with collecting status information from the devices sending the PPR valid (true or false, depending on the configuration of the devices).

The parallel poll facility can be used to indicate a request for service. This capability differs from the use of the SRQ message in the following way:

- A controller-in-charge initiates a parallel poll sequence, but any device requests the initiation of a serial poll sequence.
- A parallel poll enables the transfer of status information from multiple devices simultaneously, but a serial poll sequentially collects status information from each device.

Example of a parallel poll:



2.4.9 Device Clear - DC

The DC function enables a device to be cleared (initialized), either individually, or as a part of a group of devices. The group may be either a subset of, or all previously addressed devices on the bus.

2.4.10 Device Trigger - DT

The DT function enables a device to have its basic operation(s) started either individually, or as a part of a group of devices. The group may be either a subset of, or all previously addressed devices on the bus.

2.4.11 Controller - C

The C function enables a device to send device addresses, universal commands and addressed commands to other devices, over the interface bus. It also provides the possibility of conducting parallel polls to determine which devices require service.

A C function can only execute these facilities when sending the ATN message over the interface bus.

If more than one device connected to the interface bus has a C function, all but one of them should be in a function state called the Controller Idle State (CIDS) at any given time. The only device containing the C function which is not in the CIDS state, is the controller-in-charge. Devices with the C function are allowed to take turn as the controller-in-charge of the interface bus, if the system controller can pass this control.

The C function in the only device on the interface bus which can exist in a function named the System Control Active State (SACS). It shall remain in this state throughout operation on the bus, and is the only device with the capability to send IFC and REN whether or not it is the controller-in-charge. This device is the system controller.

2.5 Interface Functions on ND system Controller

The interface functions implemented on ND's system controller are listed in the following table.

Mnemonic	Interface function subset	Capabilities
SH1	Source Handshake 1	Control of the DAV-line.
AH1	Acceptor Handshake 1	Control NRFD- and NDAC-line.
T8	Talker 8	Basic talker, unaddress if MLA.
L4	Listener 4	Basic listener, unaddress if MTA.
SR0	Service Request 0	No capability.
RL0	Remote or Local	No capability.
PP0	Parallel poll	No capability.
DC0	Device Clear 0	No capability.
DT0	Device Trigger 0	No capability.
C1	Controller 1	System controller
C2	Controller 2	Send IFC and take charge.
C3	Controller 3	Send REN
C4	Controller 4	Respond to SRQ
C26	Controller 26	Send interface messages, execute parallel poll.

The implemented interface functions only make it possible for the controller to be both system controller and controller-in-charge. It cannot pass any control over to another controller. The controller has all other available controller functions implemented.

The system controller cannot act as an ordinary device on the interface bus.

3 SYSTEM INITIATION

3.1 Introduction

The system initiation will usually be performed by the person responsible for the computer system (system supervisor). This chapter is meant to be a guideline for this person, but also contains some information that could be useful for the ordinary user.

3.2 Switch-setting on the Controller Card

Whether the controller is active or not is determinded by a main switch on the card. The setting of this red switch has the meaning:

SYC	-	The Controller card is active SYSTEM CONTROLLER.
DEV	-	The Controller card is NOT ACTIVE at all.

NOTE There must NOT be any other devices acting as SYSTEM CONTROLLERS on the interface bus.

The thumbwheel switch on the card determines which device number, and thereby which controller number, the card shall have. The numbered position of the switch is equal to the controller number (between 0 and 7).

3.3 How to start and stop the GPIB Driver

This instruction assumes that you have SINTRAN III version J.

3.3.1 How to start the GPIB Driver

- Log on as user SYSTEM.
- Enter the SINTRAN-SERVICE program.
- If XMSG not is started, execute the command START-XMSG. (The program should respond with: OK: XMSG STARTED.)

- Execute the command START-GPIB in this way:

	Comments:
*START-GPIB ↵	Command name entered.
CONTROLLER NUMBER: 0 ↵	Enter the number for the controller you want to start.
OK, CONTROLLER STARTED	Controller is available. If you start more controllers, repeat the command with a new controller number each time.
*EXIT ↵	Back to SINTRAN.

NOTE If a controller is not started, and you get the message: "GPIB-startup-error <error code>", this means a fatal error has occurred. You can fetch the meaning of this error code by logging on the MONITOR and executing the commmand GET-ERROR-MESSAGE. (Look at chapter 5 for explanation.) You get error messages directly for all other error conditions that can occur when trying to start the GPIB Driver.

3.3.2 How to stop the GPIB Driver

- Log on as user SYSTEM.
- Enter the SINTRAN-SERVICE program.
- Execute the command STOP-GPIB.

This command follows the same sequence as for the command START-GPIB. You stop the controllers one by one. The meaning of error codes that could appear can be fetched with the GET-ERROR-MESSAGE command in the GPIB MONITOR.

3.4 How to change the GPIB Buffersize

As a little introduction, some comments are given on the communicating system in GPIB. This is a help for you to understand when and why the buffersize sometimes ought to be changed.

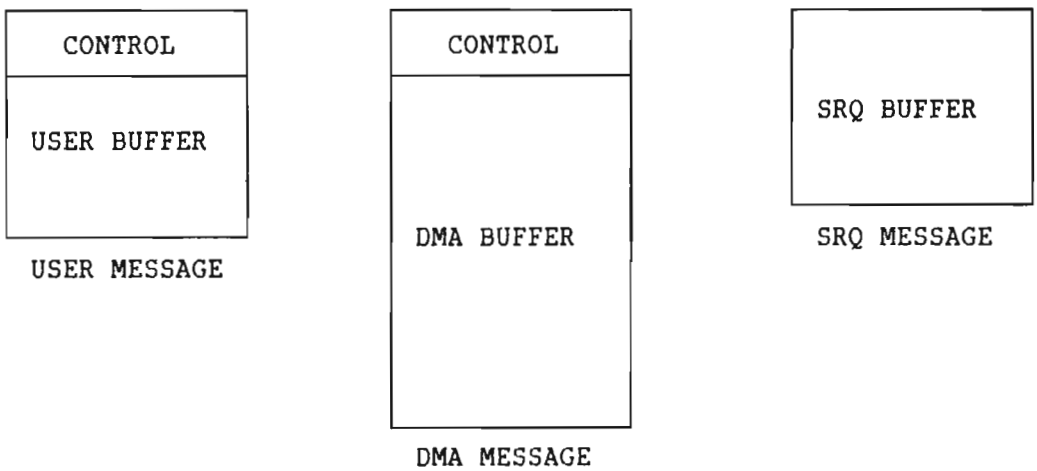
The controller is communicating with the users through the XMSG system. Each of the GPIB controllers (0-7) has two types of buffers: DMA buffer and User buffer.

The nest figure shows this communication between the user and the controller via XMSG.

The information flow between the building blocks in XMSG can be regarded as data packets. There are three types of messages used:

- User Message (one for each user).
- DMA Message (one for each controller).
- SRQ Message (one for each controller).

The format of the message types is:



Only one data packet is exchanged between two blocks at a time. The User buffer and DMA buffer are the media that transport the data packets.

The ports are software interfaces between the blocks in the GPIB system and XMSG. All ports, except the SRQ port, are bidirectional. The SRQ (Service Request) port handles all SRQ from the controller via the GPIB Driver. The command port sends and receives messages in the User Message. The DMA port send and receives messages in the DMA Message. The only difference between the use of the User and the DMA buffer is that the DMA buffer is larger, and therefore can transport a larger quantity of data.

The GPIB Driver is a part of the controller. SMESS is a routine in the LIBRARY that handles the messages sent to and from the user.

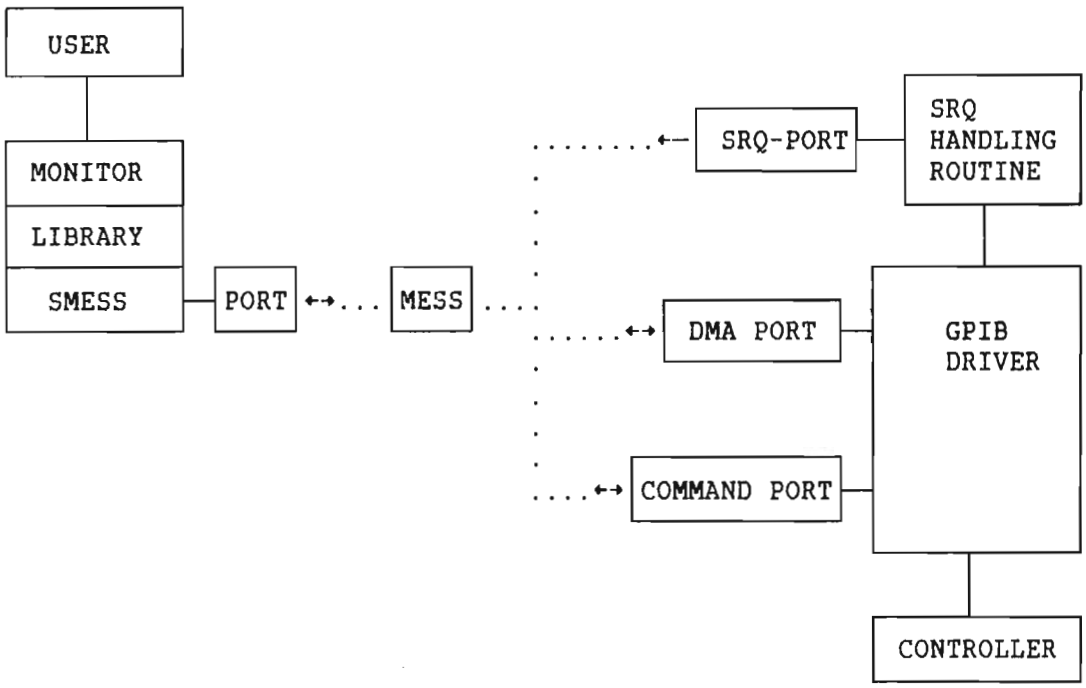
When the bytecount is too large to be sent in the User buffer from SMESS to the GPIB Driver, these things happen:

- Only the control part of the User Message is filled. No data is put in the User Buffer.

- The User Message is sent to the GPIB Driver, which will find out that SMESS demands the DMA buffer.

- The DMA Message is sent to SMESS, which will put the data in the DMA buffer and return it.

- DMA output from the controller is then performed, and the User Message is returned with status.



NOTATION: ..... means data route, —— means other communication linkage, ←— or —→ means data flow direction, MESS means Message, SMESS means System Message Handler.

Fig. 8. The GPIB Software Communication System in XMSG

If the size of your User buffer is smaller than the size of the messages sent to your device(s), and this is usually (or often) so, this results in several more calls to XMSG. These extra calls take more time to execute, and your LIBRARY routines/MONITOR commands will therefore also use more time. It would then be useful to change the size of the User buffer. The DMA buffer must be large enough to send the largest message you will ever send to any device.

The size of the User buffer and DMA buffer is described in the data field. The variables in the data field describing the buffers are called USIZE and DSIZE (User buffersize and DMA buffersize).
The variables have these default values after the GPIB system is initiated:

- USER BUFFERSIZE - 400_8 bytes

- DMA BUFFERSIZE - 2000_8 bytes

The maximum buffersize depends on your XMSG configuration. The variables in the data field are stored in the file XMSG-SYS-DEF. See the manual SINTRAN III - Communication Guide, for a closer description of the maximum buffersize you can have.

Changing the data field (for each controller) is done in the following way:

- Log on as user SYSTEM.

- Enter the SINTRAN SERVICE program, then:

Execute the command CHANGE-GPIB-BUFFERSIZE:

	Comments:
*CHANGE-GPIB-BUFFERSIZE ↵	Command name entered.
CONTROLLER NO.:<CONTROLLER NUMBER> ↵	Enter the number of the controller.
MEMORY? Y ↵	
IMAGE? Y ↵	Data fields in memory and image area will be changed.
SAVE-AREA? N ↵	Memory field is then entered and the buffers' old values displayed and new values typed sequentially.

	MEMORY	IMAGE	
USER BUFFER SIZE (OCT)	<OLD V.>	<OLD V.>	<NEW VALUE> ↵
DMA BUFFER SIZE (OCT)	<OLD V.>	<OLD V.>	<NEW VALUE> ↵

*EXIT ↵	Back to SINTRAN from SINTRAN-SERVICE

3.5 How to change the Timeout Value on the GPIB

The timeout function allows the controller to detect devices on the interface bus which do not respond when addressed. Sometimes we want to use devices which not respond as quickly as the initiated timeout value. The timeout value must then be altered. The timeout value must be changed individually for each controller.

The initiated timeout value is 6 [sec].

The timeout value is changed in the SINTRAN-SERVICE program. The new value must be entered with negative sign. This number will then be stored as a two-complement in the memory. A timeout value at for example 10 [sec.] is entered as -14_8 , and is stored as 177766_8 in the memory.

Changing of the timeout value is done in the following way:

- Log on as user SYSTEM.
- Enter the SINTRAN SERVICE program, then:

Execute the command CHANGE-DATAFIELD:

	Comments:
*CHANGE-DATAFIELD ↵	Command name entered.
LOG.UNIT.NO: <CONT. LUN> ↵	<CONT. LUN> - controller's logical unit number.
INPUT OR OUTPUT? I ↵	
MEMORY? Y ↵	Data fields in memory and image area will be changed.
IMAGE? Y ↵	
SAVE-AREA? N ↵	
MEMORY	Memory field entered.
TTMR/<OLD VALUE> <NEW VALUE>	Old timeout value displayed - new timeout value typed.
TTMR/<NEW VALUE> .	New timeout value displayed - memory field left when full stop entered.
*EXIT ↵	Back to SINTRAN from SINTRAN-SERVICE

3.6 How to select privileged User

It is possible to select which SINTRAN user that shall be privileged user. This is done by modifying a routine in the LIBRARY. This routine is stored separately on the file GPCHK:SYMB. The MONITOR and the standard LIBRARY are delivered in BRF-format.

Is it also possible to allow all SINTRAN users to do privileged commands/routines. This is done by allowing all SINTRAN users to log on as user 0 on the controller. The GPIB configuration file is always placed under the user area of the privileged user. If all users are given privileged priority, they must have read and write access to this file.

> NOTE There can only be one privileged user at a time on the controller.

In the standard version of the LIBRARY user SYSTEM is selected as privileged user. Only this user is privileged. If this is suitable for you, do as follows:

```
                                    Comments:
@NRL ↵                              Enter Nord Relocating Loader from SINTRAN.
                                    Declare the file GPIB-MONITOR as program
                                    file.

*PROG-FILE GPIB-MONITOR ↵

                                    Load the files GPIB-MONITOR:BRF and
                                    GPIB-LIBRARY:BRF together with the library
                                    FORT-1BANK-B.

*LOAD GPIB-MON,GPIB-LIB,FORT-1BANK-B ↵

*EXIT ↵                             Go back to SINTRAN.
                                    You now have the standard GPIB-version.
```

If you want to select another privileged user, or want to give all users priority, do as follows:

```
                                        Comments:

@PED GPCHK ↵                    Read the file GPCHK:SYMB into the
                                program editor.
                                The program part of the file,
                                without comments, is:

 SUBROUTINE GPCHK (STATUS,USERNAME)
 CHARACTER*(*) USERNAME
 CHARACTER*16 PUSER
 INTEGER STATUS,PSTAT
 LOGICAL CHECK

 DATA PUSER /'SYSTEM          '/   Substitute "SYSTEM" with the user
                                   name you want to be privileged.

 DATA CHECK /.TRUE./               Substitute "TRUE" with "FALSE"
                                   if all users are allowed to do
 USERNAME=PUSER                    privileged commands/routines.
 IF (CHECK.EQ..TRUE.) THEN
    CALL GSYCK (PSTAT,PUSER)
       STATUS=PSTAT
 ELSE
    STATUS=1
 ENDIF
 RETURN
 END
 PED:W ↵↵                          Store the modified file.
```

After the routine is modified do this:

```
                                        Comments:

@FORTRAN-100-B ↵                Call the FORTRAN compiler.

FTN:COMPILE GPCHK,,"GPCHK:BRF" ↵ Compile the file GPCHK:SYMB, and
                                store the result as the BRF-format
                                of the file.

FTN: EXIT ↵                     Go back to SINTRAN if no errors
                                occurred.

@BRF-MONITOIR ↵                 Enter the BRF-MONITOR.

*EXCHANGE-UNITS GPCHK,GPIB-LIB ↵ Substitute the old routine in the
                                LIBRARY with the modified one.

*EXIT ↵                         Back to SINTRAN.
```

You must now load the BRF-files GPIB-MONITOR and GPIB-LIBRARY together, with the library FORT-1BANK-B. Dump the result as the GPIB-MONITOR program file. How to do this was explained at the beginning of the section.

4 DEVICE CONNECTION ON THE BUS

4.1 Restrictions

The following limitations are an implemented part of the GPIB standard outlined by IEEE and IEC with respect to device connection on a bus.

- The total number of devices connected to an interface bus is 15.
- The total cable length in the system must neither exceed 20 meters nor 2 meters times the number of devices interconnected (the controller included). This means: If only one device is connected to the the controller, the cable length is allowed to be 4 meters.

These limitations are made with respect to general considerations about noise and time constants for signal transmission.

The restrictions on cable lengths would normally only make it possible to have the instrumentation system located very close to the computer. This problem is solved for ND's system controller. In addition to having the instruments connected directly to the controller, called local option, a remote option is available.

The remote option makes it possible to have devices located at a greater distance from the computer, which in many cases will be an advantage. The remote option requires an additional box, the remote box. The devices are connected to the remote box in this option, and the cable between the remote box and the controller card is available in 50 and 100 meters lengths.

To satisfy both IEEE and IEC compatible devices, the GPIB contact panels on the computer and the remote box are equipped with two connectors. These are electrically connected to the same bus.

The limitations for cable lengths listed above are only valid when either referring to the devices connected to the contact panel on the computer, or to the devices connected to the contact panel on the remote box. Fig. 9 shows the limitations on device connection in the local option.

The local and the remote option can be used simultaneously. A total number of 27 devices can be connected to the same interface bus if both options are used. Fig. 10 illustrates this.

The additional restriction with the remote option is:

- All devices acting as listeners at the same time either have to be in the remote or in the local part of the interface bus.

The possibility to have up to eight controllers located in the computer makes considerable expansion of the GPIB-system possible. How many controllers and devices that ought to be handled by a computer depends on how powerful it is, and what other tasks it is performing.

The following paragraphs are valid for ND's system controller:

- 13 devices can be connected to a local part of the bus.
- 14 devices can be connected to a remote part of the bus.
- Until 8 controller cards can be located in a computer.

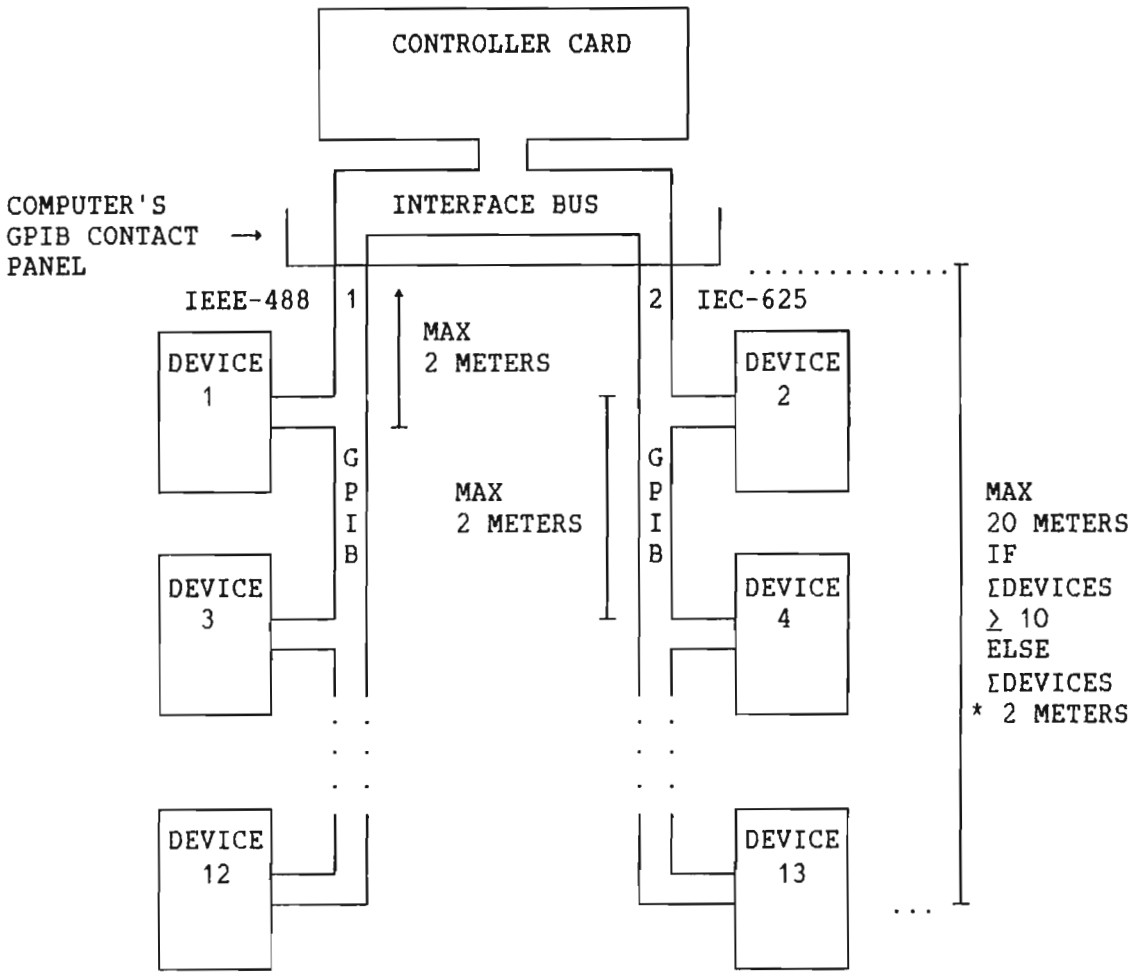


Fig. 9. Restrictions on Device Connection in Local Option

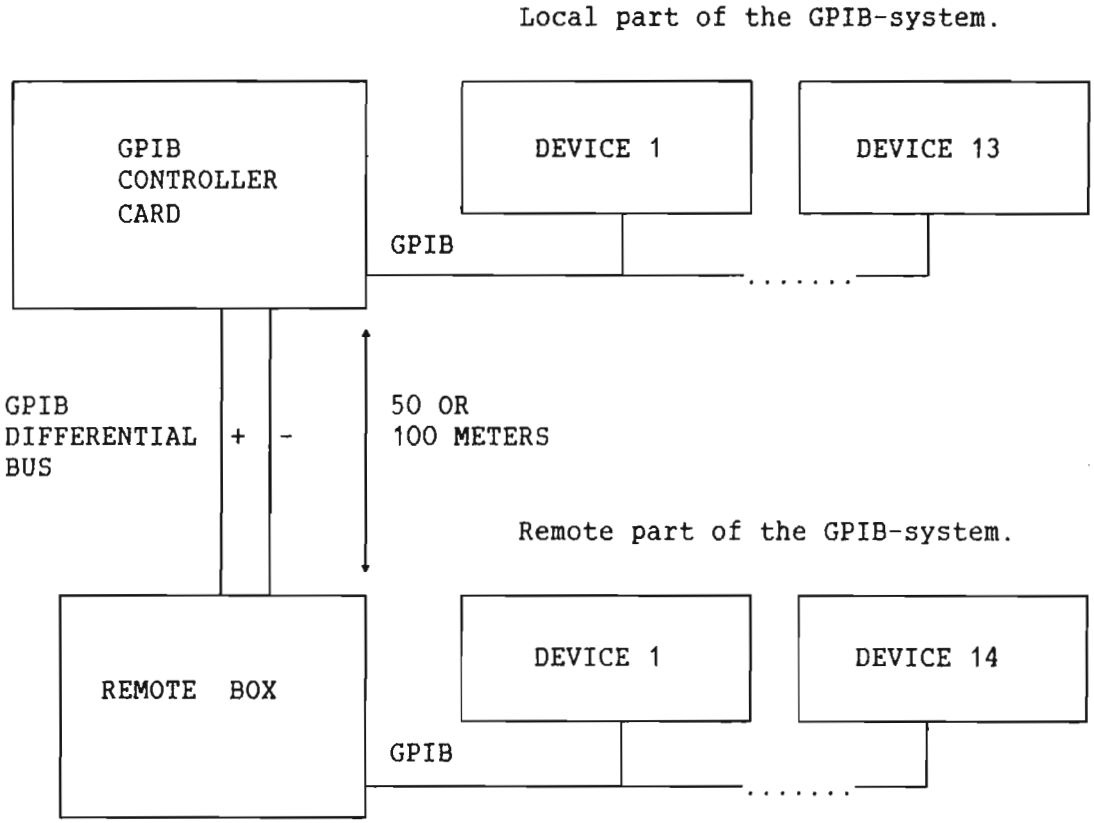


Fig. 10. Local and Remote Option

4.2 How to connect the remote Box to the Controller

Referring to figures 11 and 12:

- Insert the remote cable's connectors in one end of the cable into the corresponding D-connectors on the computer's GPIB contact panel (called Remote 1 and 2 in fig. 11).

- Insert the cable's connectors in the other end into the corresponding D-connectors on the remote box (named Remote 1 and 2 in fig. 12).

- Remote 1 on the computer should now be electrically connected to Remote 1 on the remote box, and Remote 2 equally connected.

- The power cable should be connected to a main 220 V ac power supply.

- To activate the remote box, set the switch on the front panel in upper position. The indicator marked POWER should now light up. If not, check the fuse on the back panel.

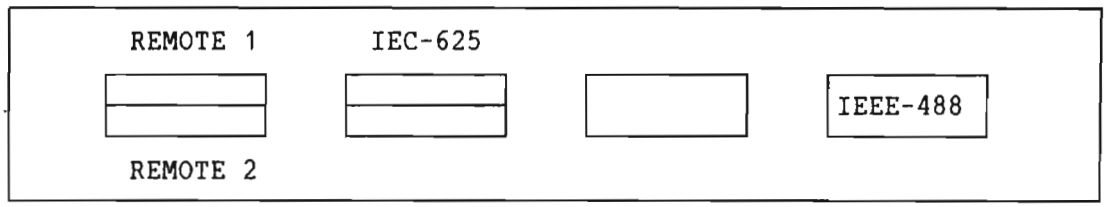


Fig. 11. The Computer's GPIB Contact Panel

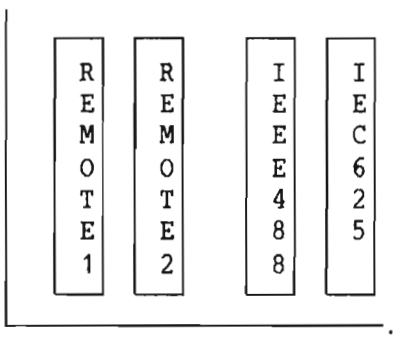


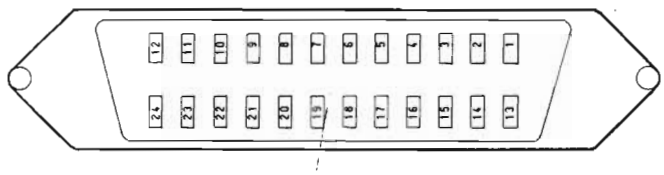
Fig. 12. The Remote Box' GPIB Contact Panel

4.3 The GPIB Connectors Pin assignment

The pin assignment of the IEEE-488 and IEC-625 connectors are listed in the following table. Figure 13 shows the two connector types.

Pin number	Signal name IEEE-488	Signal name IEC-625	Pin number	Signal name IEEE-488	Signal name IEC-625
1	DIO1	DIO1	13	DIO5	SHIELD
2	DIO2	DIO2	14	DIO6	DIO5
3	DIO3	DIO3	15	DIO7	DIO6
4	DIO4	DIO4	16	DIO8	DIO7
5	EOI	REN	17	REN	DIO8
6	DAV	EOI	18	GND 5	GND 5
7	NRFD	DAV	19	GND 6	GND 6
8	NDAC	NRFD	20	GND 7	GND 7
9	IFC	NDAC	21	GND 8	GND 8
10	SRQ	IFC	22	GND 9	GND 9
11	ATN	SRQ	23	GND 10	GND 10
12	SHIELD	ATN	24	GND 11	GND 11
			25		GND 12

IEEE-488



IEC-625

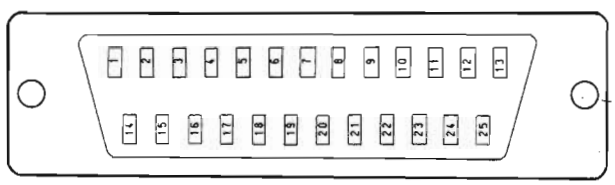


Fig. 13. The IEEE-488 and IEC-625 Connectors

4.4 How to connect Devices on the Bus

The GPIB-connectors are of the piggy-back type, ie., one connector can be stacked upon the other. The connector stack should not be made too high. The interconnected devices should only have small differences in frame potential.

NOTE	Never set the power on a device when the system is running! This can cause faulty operation in the system.

Device connection referring to fig. 11 and 12:

- All IEEE-488 compatible devices must be connected to the Ribbon connector local or remote (called IEEE-488).

- All IEC-compatible devices must be connected to the D-connector local or remote (called IEC-625).

There are different ways of inteconnecting devices based on either linear configuration or star configuration. The linear configuration is symbolically shown in fig. 9 and fig. 10.

4.5 Specifications for the Remote Box

20 pairs of differential lines are used for extending the GPIB. The receivers, type 26LS31, and line drivers, type 26LS32, are used for digital transmission over balanced differential lines. These are standard EIA RS-422 signal drivers and receivers. There are 17 pairs of lines for the standard signals in the GPIB. The EOI-line has two pairs of lines, one for each direction. There are two pairs of control lines, one out and one in. The last two lines are static; RPRES indicating that the remote box is cabled to the computer, and RPON for remote box power-on check.

4.6 Examples of Data Transfer Rates

The maximum data transfer rate is 175 000 bytes/s in the local part of GPIB. There is some overhead in initiating and ending the transfer. This overhead is mostly caused by the XMSG-system used by the GPIB-driver and the code for the microprocessor (Z80) on the controller card. If the device is located in the remote part, additional 2 µs are used for each byte transferred (remote box is 100 meters away from the computer).

The time spent on a typical data input is:

Sending talk address and activating DMA (Direct memory access) transfer	0,4 ms	)[1]
Sending 10 000 bytes	65,0 ms	)[2]
Sending unlisten message	0,2 ms	)[1]
Sending untalk message	0,4 ms	)[1]
TOTAL	66,0 ms	

)[1] Depending on the slowest device.
)[2] Depending on the slowest addressed device.

A remote device would required an additional overhead for 20 ms.

When a SRQ is received, the controller sends an interrupt to the computer's CPU. The GPIB driver then initiates a serial poll from the controller. The controller polls the devices enabled for serial polling, starting with the device with the lowest number. When the RSQ message is received the controller sends user and device number and poll status to the driver. In this way you are informed about poll status and device number. The time spent on a typical serial poll of two local devices is:

Time used between SRQ received and SPE (serial poll enable)	3,2 ms
Polling of first device	0,8 ms
Polling of second device	0,8 ms
SPD (serial poll disable) and UNT (untalk)	0,6 ms
TOTAL	5,4 ms

The additional overhead in the GPIB-driver and XMSG-system is between 10 and 40 ms.

5 GPIB_MONITOR

5.1 INTRODUCTION

The GPIB MONITOR is a subsystem where the different calls to the GPIB LIBRARY are implemented as commands.

The subsystem for the GPIB MONITOR is called by writing "GPIB-MONITOR", or an unambiguous abbreviation of this name, in SINTRAN command mode. The MONITOR responds to this call by displaying its name, microprogram version and version date. Afterwards it displays "ENTER COMMAND:" to tell that commands can be accepted. This is the MONITOR's prompt.

Example of GPIB MONITOR call:

```
@GPIB-MONITOR ↵

GPIB-MONITOR ND-10768 revision A00 84-08-17

ENTER COMMAND:
```

The next step will usually be to log on a controller. Read the description of the commands LOG-ON, LOG-OFF, SET-DEFAULT-CONTROLLER and EXIT in the section 5.2 before you do this.

The command can be entered on one or several lines, similar to SINTRAN commands.

The MONITOR will usually only prompt the user for missing parameters if no default values exist.

Abbreviation of the command name and its parameters is legal if the abbreviation is not ambiguous. This is also valid for device names.

The command and the parameters are separated with spaces or commas.
Format: <COMMAND NAME> <PARAMETER-1> <PARAMETER-2> ... <PARAMETER-N>

If a "@" is entered as a parameter, the MONITOR ignores the command and asks for the next. This can be a very useful facility. It can, for example, prevent your being trapped in a parameter specification input routine in a command. If you give an illegal value as parameter, the MONITOR will keep asking for this parameter until the input is legal. If you do not know what to answer, and want to continue with another command, just enter "@" and avoid being trapped.

Example of breaking a command:

```
                                          Comments:
ENTER COMMAND: GET-FUNCTION-CODE  ↵       Command name entered.

FUNCTION CODE: 100  ↵                     This answer is illegal.

                                          The MONITOR keeps asking
FUNCTION CODE OUT OF RANGE                for the same parameter.
FUNCTION CODE: @  ↵                       "@" entered, and trap
ENTER COMMAND:                            avoided.
```

The radix can be specified by appending B for octal, D for decimal or H for hexadecimal after a number. The radix indicator must be separated from the number with full stop. I.e., "15.D" is equal to "F.H" is equal to "17.B".)

The radix indicator can be specified with a default value. The default value is set by the command SET-DEFAULT-RADIX. The default value is octal after system initiation.

SINTRAN commands are executed directly from the MONITOR by appending "@" to the MONITOR's prompt and then entering the SINTRAN command. Be very careful with the SINTRAN commands you give to the MONITOR, they are not checked by it. You can, by accident, call for another subsystem or log off the computer. This causes you to log off the GPIB subsystem as well, and all your reserved devices are released.

Example of entering a SINTRAN command:

```
                                          Comments:
ENTER COMMAND: @WHO-IS-ON  ↵              Suppose you are logged in as user
                                          RT in SINTRAN and give this
                                          command from the MONITOR to
                                          SINTRAN. Then this could be the
                                          result:
               48 LAURA-LARA
               50 SYSTEM
      ——→     545 RT                      Your user name and terminal number
```

5.2 COMMANDS

5.2.1 LOG-ON

Explanation

This command logs you on to a GPIB controller as user. If the command is executed from a user in SINTRAN defined to be privileged when the system was initiated, it asks whether the user wants to be privileged or not. If yes, it tries to log on as user 0_8. If the command is executed from any other user, it tries to log on as nonprivileged user. Microprogram version and your user number are returned by the MONITOR. If the controller is not active, or does not succeed in logging on as privileged user, an error message is displayed.You are allowed to be logged on to several controllers simultaneously.

Parameters

1. <CONTROLLER NUMBER> A number between 0_8 and 17_8.

2. <PRIVILEGED USER> Y or N.

Example:

	Comments:
ENTER COMMAND: LOG-ON ↵ MICROPROGRAM VERSION: 066 REV.: K YOUR USER NUMBER IS: 01B	You log as nonprivileged user and get user number 01_8.

5.2.2 LOG-OFF

Explanation

This command logs you off as user on a controller. Your reserved devices on the controller are released.

Parameter

1. <CONTROLLER NUMBER> A number between 0_8 and 17_8.

Example:

ENTER COMMAND: LOG-OFF ↵	Comments: You log off default controller.

5.2.3 EXIT

Explanation

This command logs you off the MONITOR. You logs off all controllers where you are logged on, and all your reserved devices are released. This command has no parameters.

Example:

ENTER COMMAND: EXIT ↵	Comments: You log off the MONITOR.

5.2.4 SET-DEFAULT-RADIX

Explanation

This command sets the default radix indicator. Octal is set as default radix when you enter the MONITOR.

Parameter

1. <RADIX> B, D or H. B sets octal as default, D sets decimal as default and H sets hexadecimal as default radix indicator.

Example:

ENTER COMMAND: SET-DEFAULT-RADIX ↵ RADIX (B=OCTAL D=DECIMAL H=HEX): D ↵	Comments: You select decimal as default radix indicator.

5.2.5 SET-DEFAULT-CONTROLLER

Explanation

This command sets the default controller number. This number will be used as default controller number, instead of the system asking you for this parameter. When you enter the MONITOR, controller number 0_8 is set as default controller number.

Parameter

1. <CONTROLLER NUMBER> A number between 0_8 and 17_8.

Example:

```
ENTER COMMAND: SET-DEFAULT-CONTROLLER A.H ↵

Comments: You choose controller number 10 as default.
```

5.2.6 RESET-DEFAULT-CONTROLLER

Explanation

This command resets the default controller number. It means that the MONITOR will always ask you for controller number where needed. This command has no parameters.

Example:

```
ENTER COMMAND: RESET-DEFAULT-CONTROLLER ↵
ENTER COMMAND: LOG-ON 5 ↵

Coments: You reset default controller and log on controller 5.
```

5.2.7 RESERVE-CONTROLLER

Explanation

This command reserves a controller for you. The command can be used both by the privileged and by the nonprivileged user. When the controller is reserved by a user, no other users are able to log on the controller.

Parameter

1. <CONTROLLER NUMBER> A number between 0_8 and 17_8.

Example:

```
                                        Comments:
ENTER COMMAND: RESERVE-CONTROLLER 3 ↵   You reserve controller 3.
```

5.2.8 RELEASE-CONTROLLER

Explanation

This command releases a controller reserved earlier. Only the user who reserved the controller is allowed to release it.

Parameter

1. <CONTROLLER NUMBER> A number between 0_8 and 17_8.

Example:

	Comments:
ENTER COMMAND: RELEASE-CONTROLLER 9.D ↵	You release controller 9.

5.2.9 CONFIGURATE-DEVICE

Explanation

This command configurates a new device on a controller's interface bus. The configuration of the device is written on the GPIB configuration file, and also into the controller. Default values in the command are displayed between slashes. Device number is returned by the MONITOR after configuration. This command can only be given by the privileged user.

Parameters

1. <CONTROLLER NUMBER> A number between 0_8 and 17_8.

2. <DEVICE NAME> A string of maximum 20_8 characters.

3. <LISTEN ADDRESS> A number between 40_8 and 76_8.

   This parameter is the listen address to the device. The address 77_8 is the GPIB UNL (unlisten) message.

4. <TALK ADDRESS> A number between 100_8 and 136_8.

   This parameter is the talk address to the device. The address 137_8 is the GPIB UNT (untalk) message.

5. <SECONDARY ADDRESSES ON DEVICE> Y or N.

   If the device is to be addressed by a secondary address, this must in commands. always be specified after the device name. Secondary addresses make it possible to choose between different sources within a device.

6. <TIMEOUT ENABLED ON INPUT> Y or N.

   If the device uses more than 16 [msec] to send or receive the first byte after it has been addressed the answer should be N.

7. <LOCAL OR REMOTE BUS> L or R.

   The answer is L if the device is connected to the local bus, and R if it is connected to the remote bus.

8. <SERIAL POLL ALLOWED> Y or N.

   If the device has the service request (and serial poll) function (SR1), the answer should be Y. If the answer is N, the controller will not poll the device if a SRQ is received.

NOTE A device which sends a SRQ (service request) message and is <u>not</u> polled can jam the controller.

9. <SINGLE OR MULTI-USER> S or M.

   If only one user is allowed to use the device, the answer should be S. If several users are allowed to use the device, the answer should be M.

10. <EOI WITH, AFTER OR NONE EOI> W, A or N.

    This parameter affects the way the EOI-line is set to true by the controller at the end of a data transfer. If EOI should be set with the last byte transfer, the answer is W. If EOI should be set after the last byte transfer, the answer is A. If EOI not should be used to indicate the end of a data transfer, the answer is N.

11. <EOS-CHARACTER USED> Y or N.

    If the device uses an EOS-character when exchanging data the answer should be Y, otherwise it should be N. If the answer is N, the MONITOR will not ask for the next two parameters.

12. <EOS CHECK ON INPUT> Y or N.

    If the device uses the EOS-character to indicate end of data transfers from the device to the controller, the answer should be Y, otherwise it should be N. The controller will always use EOS-character to indicate end of data transfers when sending data to the device, if the answer was Y on previous parameter.

13. <EOS-CHARACTER VALUE> A number between 0_8 and 377_8.

    The ASCII value of the EOS-character.

Example:

	Comments:
ENTER COMMAND: CONFIGURATE-DEVICE ↵	
LISTEN ADDRESS//: 56.B ↵	
TALK ADDRESS/116B/: ↵	Default talk adress is always Listen address + 40_8.
SECONDARY ADDRESSES ON DEVICE/NO/: ↵	Some of the default values are well suited, and you select them.
TIMEOUT ENABLED ON INPUT/YES/: ↵	
LOCAL OR REMOTE BUS/LOCAL/: ↵	
SERIAL POLL ALLOWED/YES/: ↵	
SINGLE OR MULTI-USER/SINGLE/: M ↵	
EOI WITH, AFTER OR NONE EOI/WITH/: ↵	
EOS-CHARACTER USED/NO/: Y ↵	The device uses EOS-character.
EOS-CHECK ON INPUT//: Y ↵	The device sends EOS-character to the controller, and the value of this is 012_8.
EOS-CHARACTER VALUE//: 012 ↵	
DEVICE NUMBER IS 03B	The device got the device number 3_8 on the controller.

5.2.10 CHANGE-DEVICE-CONFIGURATION

Explanation

This command changes the device configuration of a device which has been configurated previously. The MONITOR uses the old parameters from the previous device configuration as default values. This command can only be executed by the privileged user.

Parameters

1. <CONTROLLER NUMBER> A number between 0_8 and 17_8.

2. <DEVICE NAME> A string of maximum 20_8 characters.

3. <LISTEN ADDRESS> A number between 40_8 and 76_8.

   This parameter is the listen address to the device. The address 77_8 is the GPIB UNL (unlisten) message.

4. <TALK ADDRESS> A number between 100_8 and 136_8.

   This parameter is the talk address to the device. The address 137_8 is the GPIB UNT (untalk) message.

5. <SECONDARY ADDRESSES ON DEVICE> Y or N.

   If the device is to be addressed by a secondary address, this must always be specified in commands after the device name. Secondary addresses make it possible to choose between different sources within a device.

6. <TIMEOUT ENABLED ON INPUT> Y or N.

   If the device uses more than 16 [msec] to send or receive the first byte after it has been addressed, the answer should be N.

7. <LOCAL OR REMOTE BUS> L or R.

   The answer is L if the device is connected to the local bus, and R if it is connected to the remote bus.

8. <SERIAL POLL ALLOWED> Y or N.

   If the device has the service request (and serial poll) function (SR1), the answer should be Y. If the answer is N, the controller will not poll the device if a SRQ is received.

NOTE A device which sends a SRQ (service request) message and is <u>not</u> polled can jam the controller.

9. <SINGLE OR MULTI-USER> S or M.

   If only one user is allowed to use the device, the answer should be S. If several users are allowed to use the device the answer, should be M.

10. <EOI WITH, AFTER OR NONE EOI> W, A or N.

    This parameter affects the way the EOI-line is set to true by the controller at the end of a data transfer. If EOI should be set with the last byte transfer, the answer is W. If EOI should be set after the last byte transfer, the answer is A. If EOI not should be used to indicate the end of a data transfer, the answer is N.

11. <EOS-CHARACTER USED> Y or N.

    If the device uses an EOS-character when exchanging data the answer should be Y, otherwise it should be N. If the answer is N, the MONITOR will not ask for the next two parameters.

12. <EOS CHECK ON INPUT> Y or N.

If the device uses the EOS-character to indicate end of data transfers from the device to the controller, the answer should be Y, otherwise it should be N. The controller will always use EOS-character to indicate end of data transfers when sending data to the device, if the answer was Y on previous parameter.

13. <EOS-CHARACTER VALUE> A number between 0_8 and 377_8.

The ASCII value of the EOS-character.

5.2.11 RECONFIGURATE

Explanation

This command reads the GPIB configuration file into the controller. This file is found under the user area of the privileged user.The file is called GPIB-CONFIG-XX:SYMB, where XX denotes a variable that is equal to the controller number. This command is privileged.

Parameter

1. <CONTROLLER NUMBER> A number between 0_8 and 17_8.

Example:

```
ENTER COMMAND: RECONFIGURATE 1 ↵

Comments: GPIB-CONFIG-01:SYMB is read into controller 1.
```

5.2.12 SEND-PARALLEL-POLL-CONFIGURATION

This command sends a parallel poll configuration byte to the specified device. This byte tells how the device shall respond with its PPR (Parallel Poll Response) message during a parallel poll.The parallel poll configuration byte is called PPE (Parallel Poll Enable) in the GPIB standard. See also the description of parallel poll in chapter 2. If several devices share a single data line for the PPR, they must get the same configuration byte.

Parameters

1. <CONTROLLER NUMBER> A number between 0_8 and 17_8.

2. <DEVICE NAME> A string of maximum 20_8 characters.

3. <CONFIGURATION BYTE> A number.

The configuration byte is calculated in this way:

PPE = 140_8 + ST + B,

where S and B are variables. ST=0_8 makes the device send PPR false, and ST=10_8 makes it send PPR true. The variable B says which data line PPR^8 shall be sent on, and thereby which corresponding bit in the byte the message shall affect. B is 0_8 makes the device send PPR on DIO1, and B=7_8 makes it select DIO8.

Example:

ENTER COMMAND: SEND-PARALLEL-POLL-CONF 0,HP-DVM,153.B

Comments:
The configuration byte 153_8 is sent to HP-DVM. This byte makes the device send PPR true on DIO4.

5.2.13 RESET-REN-LINE

Explanation

This command resets the REN (Remote ENable) line on a controller's interface bus. The REN-line is set until this command is executed. The next executed MONITOR command that makes use of the interface bus sets the REN-line again. When the REN-line is reset, the devices can be operated from their local controls. This command is privileged.

Parameter

1. <CONTROLLER NUMBER> A number between 0_8 and 17_8.

Example:

ENTER COMMAND: RESET-REN-LINE ↵

Comments: You reset REN-line on default controller

5.2.14 SET-REN-LINE

Explanation

This command sets the REN-line again after it has been reset by the RESET-REN-LINE. The devices are then operated by the controller. It is not necessary to execute this command to be able to make use of other MONITOR commands. This command is privileged.

Parameter

1. <CONTROLLER NUMBER> A number between 0_8 and 17_8.

Example:

```
ENTER COMMAND: SET-REN-LINE 0 ↵

Comments: You set REN-line on controller 0.
```

5.2.15 SET-CONTROLLER-ADDRESS

Explanation

This command changes the GPIB controller's talk and listen addresses. This command is privileged. When the GPIB controller has been cleared, the listen address is set to 65_8 and the talk address is set to 125_8. Only the talk address is specified by the user. The listen address is automatically calculated to:

Listen address = Talk address - 40_8.

Parameters

1. <CONTROLLER NUMBER> A number between 0_8 and 17_8.

2. <TALK ADDRESS> A number between 100_8 and 136_8.

Example:

```
ENTER COMMAND: SET-CONTROLLER-ADDRESS 135 ↵

Comments:
Controller's talk address is specified to 135₈ , and listen address
is calculated to 75₈ by the controller.
```

5.2.16 RESERVE-DEVICE

Explanation

This command reserves the specified device(s) for you. Before any operation can be performed on a device, it must be reserved. Single user devices can only be reserved by one user, and multi-user devices by many users simultaneously. If the desired device cannot be reserved, you will get an error message. Whether a device is for single users or multi-users is decided when the device is configurated.

Parameters

1. <CONTROLLER NUMBER> A number between 0_8 and 17_8.

2. <DEVICE NAME(S)> A string of maximum 20_8 characters for each dev.

Example:

ENTER COMMAND: RESERVE-DEVICE INSTRUMENT-1,INSTRUMENT-2 ↵

Comments: You reserve the devices INSTRUMENT-1 and INSTRUMENT-2.

5.2.17 RELEASE-DEVICE

Explanation

This command releases the specified device(s) for you. This command can be given by the user who reserved the specified device(s).

Parameters

1. <CONTROLLER NUMBER> A number between 0_8 and 17_8.

2. <DEVICE NAME(S)> A string of maximum 20_8 characters for each dev.

Example:

ENTER COMMAND: RELEASE-DEVICE HP-1610 ↵

Comments: You release the device HP-1610.

5.2.18 DELETE-DEVICE

Explanation

This command deletes the device from the controller's device list and the GPIB configuration file. This command can only be given by the privileged user.

Parameters

1. <CONTROLLER NUMBER> A number between 0_8 and 17_8.

2. <DEVICE NAME> A string of maximum 20_8 characters.

Example:

```
ENTER COMMAND: DELETE-DEVICE LOGIC-ANALYZER ↵

Comments: You delete the device LOGIC-ANALYZER.
```

5.2.19 TRIGGER-DEVICES

Explanation

This command sends a trigger command to the specified device(s). This command is named GET (Group Execute Trigger) in the GPIB standard. The trigger command allows you to have basic operations in different devices started simultaneously.

Parameters

1. <CONTROLLER NUMBER> A number between 0_8 and 17_8.

2. <DEVICE NAME(S)> A string of maximum 20_8 characters for each dev.

Example:

```
ENTER COMMAND: TRIGGER-DEVICES DEVICE-A,DEVICE-B ↵

Comments: You trig the devices DEVICE-A and DEVICE-B simultaneously
```

5.2.20 CLEAR-CONTROLLER

Explanation

This command clears the the GPIB controller into a predefined quiescent state, but all information about users and device configuration will remain unchanged. This command is privileged.

Parameter

1. <CONTROLLER NUMBER> A number between 0_8 and 17_8.

Example:

```
ENTER COMMAND: CLEAR-CONTROLLER ↵
Comments: You clear default controller.
```

5.2.21 CLEAR-DEVICES

Explanation

This command sends a SDC (Selected Device Clear) command to the specified device(s). The specified device(s) is set in a device-dependent cleared state.

Parameters

1. <CONTROLLER NUMBER> A number between 0_8 and 17_8.

2. <DEVICE NAME(S)> A string of maximum 20_8 characters for each dev.

Example:

```
ENTER COMMAND: CLEAR-DEVICE TEK-400,TEK-1800 ↵

Comments: You clear the devices TEK-400 and TEK-1800.
```

5.2.22 INTERFACE-CLEAR

Explanation

This command sends the IFC pulse on the GPIB, and all the devices' interfaces are put into a predefined quiescent state. This command is privileged.

Parameters

1. <CONTROLLER NUMBER> A number between 0_8 and 17_8.

Example:

```
ENTER COMMAND: INTERFACE-CLEAR ↵

Comments: You clear all devices on the default controller's bus.
```

5.2.23 LOCAL-DEVICES

Explanation

This command sends the GTL (Go To Local) command to the specified device(s). The device(s) is then operated by the local controls.

Parameters

1. <CONTROLLER NUMBER> A number between 0_8 and 17_8.

2. <DEVICE NAME(S)> A string of maximum 20_8 characters for each dev.

Example:

```
ENTER COMMAND: LOCAL-DEVICES PER,TEK,LISA ↵

Comments: You set the devices PER, TEK and LISA in local mode.
```

5.2.24 SEND-ASCII-STRING

Explanation

This command sends an ASCII-string to the specified device(s). The ASCII-string is device-dependent data for the device(s).

Parameters

1. <CONTROLLER NUMBER> A number between 0_8 and 17_8.

2. <DEVICE NAME(S)> A string of maximum 20_8 characters for each dev.

3. <ASCII-STRING> A string of a restricted number characters.

   If the ASCII-string is entered on the same line as the device name(s), it must be preceded by a space or a comma and a "@". The command-string must not exceed 60 characters on a line. (This means that the sum of the characters in the command-name, the device name(s) and ASCII-string is limited to 60 characters.) If the string is longer than this, enter "↵" after parameter no 2. The MONITOR responds with : "ENTER STRING", and is it now possible to enter another 80 characters.

Example 1

```
ENTER COMMAND:SE-ASC-S HP-1,HP-2 @MN5 ↵

Comments: The string "MN5" is sent to the devices HP-1 and HP-2.
```

Example 2

```
ENTER COMMAND:SEND-ASCII OSC-1,OSC-2 ↵
ENTER STRING
F1.103MHZ ↵

Comments: The string "F1.103MHZ" is sent to OSC-1 and OSC-2.
```

5.2.25 SEND-DATA

Explanation

This command sends data from the databuffer in the MONITOR to the specified device(s).

Parameters

1. <CONTROLLER NUMBER> A number between 0_8 and 17_8.

2. <DEVICE NAME(S)> A string of maximum 20_8 characters for each dev.

3. <NUMBER OF BYTES> A number between 1_8 and 10000_8.

   The number of bytes to transfer, maximum 8 k bytes. If number of bytes is entered on the same line as the device names, it must be preceded by a space or a comma and a "@".

Example:

```
ENTER COMMAND: SEND-DATA CDX @5.D ↵

Comments: 5 bytes are sent from the databuffer to the device CDX.
```

5.2.26 RECEIVE-DATA

Explanation

This command receives data from a talking device. The data received is placed in the MONITOR's databuffer. The MONITOR displays the number of bytes received.

Parameters

1. <CONTROLLER NUMBER> A number between 0_8 and 17_8.

2. <DEVICE NAME(S)> A string of maximum 20_8 characters for each dev.

   The device last specified is the talker, the others are listeners.

3. <NUMBER OF BYTES> A number between 1_8 and 10000_8.

   The number of bytes supposed to receive, maximum 8 k bytes. If the number of bytes is entered on the same line as the device names, it must be preceded by a space or a comma and a "@". If the EOI-line is set true, or an EOS-character is detected before the maximum number of bytes have been received, the transfer is terminated.

Example 1:

```
ENTER COMMAND: RECEIVE-DATA FLUKE @1A.H ↵

Comments: The controller will receive max 26 bytes from FLUKE.
```

Example 2:

Suppose you have a device which responds with device-dependent data after it is enabled by a device-dependent command. Then you can send this command by the SEND-ASCII-STRING command. The device-dependent data can then be received by the RECEIVE-DATA command.

```
ENTER COMMAND: SEND-ASCII HP-1610 @CP ↵
ENTER COMMAND: RECEIVE-DATA HP-1610 @5 ↵
NUMBER OF BYTES RECEIVED : 000002B

Comments: HP-1610 is enabled to send device-dependent data.
          It sent two bytes to the controller.
```

5.2.27 DUMP-DATABUFFER

Explanation

This command writes a specified part of the databuffer on your terminal. The databuffer is divided into bytes, and can contain maximum 8 k bytes (8192 bytes). The output file is divided into three parts. The left part is the databuffer presented as words (two bytes are equal to one word). The center part is the databuffer presented as bytes. The right part is the databuffer presented as symbolic ASCII-characters. Non-printable characters are substituted with full stops. The databuffer is dumped in groups of eight bytes, and the first byte dumped is always a number which remains as integer when divided with the constant 8.

Parameters

1. <FIRST BYTE> A number between 0_8 and 17777_8.

   The first byte in the databuffer to be displayed.

2. <NUMBER OF BYTES> A number between 1_8 and 10000_8.

   The number of bytes in the databuffer to be displayed.

3. <RADIX> B, D or H.

   B for octal, D for decimal or H for hexadecimal. The radix indicator in this command is specified by an actual parameter, and not by just appending it to the number as usual. The MONITOR will not prompt you for this parameter if it is missing.

Example:

```
ENTER COMMAND: DUMP-DATABUFFER 1,12 D ↵

Comments:

The part of the databuffer is displayed with the matrix:

BYTE        WORDS (DEC)                    BYTES (DEC)              ASCII
NO
0000: WORD1 WORD2 WORD3 WORD4   BYTE1 BYTE2 .. BYTE8   CH1 CH2 ...CH8
0010: WORD5 WORD6 WORD7 WORD8   BYTE9 BYTE10 ...BYTE16 CH9 CH10 ..CH16
```

5.2.28 SAVE-DATABUFFER

Explanation

This command saves the databuffer to a file in the SINTRAN file system. You must have write access to the SINTRAN user where the file is to be saved. If the file does not exist, it must be created with two double quotes around the file name. Then you must also have directory access to the file.

Parameter

1. <OUTPUT FILE> A SINTRAN file name.

Example:

```
ENTER COMMAND: SAVE-DATABUFFER (SYSTEM)MEASUREMENTS:DATA ↵

Comments:
A copy of the databuffer is placed in the file  MEASUREMENTS:DATA
on user SYSTEM.
```

5.2.29 LOAD-DATABUFFER

Explanation

This command reads a specified file in the SINTRAN file system into the databuffer. You must have read access to the file.

Parameter

1. <FILE-NAME> A SINTRAN file name.

Example:

ENTER COMMAND: LOAD-DATABUFFER DEVICE-DEPENDENT ↵

Comments: The file DEVICE-DEPENDENT is read into the databuffer.

5.2.30 CLEAR-DATABUFFER

Explanation

This command sets the controller's databuffer to zeros only. The command has no parameters.

Example:

ENTER COMMAND: CLEAR-DATABUFFER ↵

Comments: The controller's databuffer is set to zeros.

5.2.31 TRANSFER-DATA

Explanation

This command transfers data between specified devices on the bus, and the controller is not listening to the transfer.

Parameters

1. <CONTROLLER NUMBER> A number between 0_8 and 17_8.

2. <DEVICE NAME(S)> A string of maximum 20_8 characters for each dev.

   The device last specified will be the talker, and the others will be listeners.

3. <NUMBER OF BYTES> An arbitrary number.

   The maximum number of bytes to transfer between the devices. If the EOI-line is set true, or an EOS-character is detected before the maximum number of bytes has been received, the transfer will be terminated.

Example:

ENTER COMMAND: TRANSFER-DATA PRINTER,VOLT 6 ↵

Comments: Max. 6 bytes are transferred from device VOLT to PRINTER.

5.2.32 LIST-SYSTEM-DEVICES

Explanation

This command lists all or a group of the devices on the controller's bus that have been configurated.

Parameters

1. <CONTROLLER NUMBER> A number between 0_8 and 17_8.

2. <DEVICE NAME> A string of maximum 20_8 characters.

   The device name, or the abbreviation of the device name(s), to be listed. Default value is all configurated devices on the bus. The MONITOR will not prompt you for this parameter.

3. <OUTPUT FILE> A SINTRAN file name.

   The SINTRAN file name for an output file or device where data is to be written. You must have write access to this file. Default value is your own terminal. The MONITOR will not prompt you for this parameter.

Example 1:

```
ENTER COMMAND: LIST-SYSTEM-DEVICES F ↵

Comments:
You list all devices on the bus with names that start with an F.
The displayed result of this command has the matrix:

DEV. DEV.  LAD  TAD  SA TO SP R/L M/S  EOI  EOS EOSC  RESERVED BY
NO.  NAME                                            USER(S)

The entries in the matrix are: DEV.NO.- Device Number, DEV. NAME -
Device name, LAD - Listen address, TAD - Talk address,
SA - Secondary Address, TO - Timeout enabled, SP - Serial poll
allowed, R/L - Remote or local bus, S/M - Single or Multi-user,
EOI - EOI with, after or none EOI, EOS - EOS check on input,
EOSCH - EOS-character (See description of these parameters in
the command CONFIGURATE-DEVICE.) RESERVED BY USER(S) - the entries
are the user numbers.
```

Example 2:

```
ENTER COMMAND: LIST-SYSTEM-DEVICS F,DEVICE-LIST ↵

Comments:
The devices with names that start with an F are written to the
file named DEVICE-LIST.
```

Example 3:

```
ENTER COMMAND: LI-SYS-DEV,,"(PRH)DEVICE-LIST)" ↵

Comments:
All system devices are written to a new file named DEVICE-LIST on
user name PRH. The file is created by putting double quotes
around the file name.
```

5.2.33 LIST-USER-DEVICES

Explanation

This command lists all or a group of the devices on the bus that have been reserved by a you.

Parameters

1. <CONTROLLER NUMBER> A number between 0_8 and 17_8.

2. <DEVICE NAME> A string of maximum 20_8 characters.

   The device name, or the abbreviation of the device name(s), to be listed. The MONITOR will not prompt you for this parameter.

3. <OUTPUT FILE> A SINTRAN file name.

   The SINTRAN file name for the output file, where data is to be written. Default value is your own terminal. The MONITOR will not prompt you for this parameter.

Example:

```
ENTER COMMAND: LIST-USER-DEVICES ↵

Comments: All your reserved devices are displayed on the terminal
```

5.2.34 HELP

Explanation

This command lists all or a subset of the commands available in the MONITOR on your terminal.

Parameter

1. <COMMAND> A string of characters.

The command name, or the abbreviation of a set of commands, to be listed. This is similar to the HELP command in SINTRAN. The MONITOR will not prompt you for this parameter.

Example:

```
ENTER COMMAND: HELP SEND ↵                    Comments:
SEND-ASCII-STRING                        All commands that start
SEND-DATABUFFER                          with "SEND" are listed
SEND-PARALLEL-POLL-CONFIGURATION         on your terminal.
```

5.2.35 WHO-IS-ON

Explanation

This command lists all users on a controller on your terminal. The list is divided into three parts. The left part consists of the GPIB user numbers. The center part consists of the SINTRAN user names. The right part consists of the users' logical device numbers from SINTRAN.

Parameter

1. <CONTROLLER NUMBER> A number between 0_8 and 17_8.

Example:

```
      ENTER COMMAND: WHO-IS-ON ↵                    Comments:

      00B   USER : SYSTEM  ON TERMINAL NO. 001B
----> 01B   USER : BJARTE  ON TERMINAL NO. 060B    Your user name
                                                   and status is in
                                                   the coloumn
                                                   indicated by an
                                                   arrow.
```

5.2.36 GET-ERROR-MESSAGE

Explanation

This command displays the error message corresponding to the given error code. The error codes are also listed in Appendix C.

Parameter

1. <ERRORCODE> A number between 1_8 and 437_8.

Example:

```
ENTER COMMAND: GET-ERROR-MESSAGE 23 ↵        Comments:

ERROR CODE :  23B                          You fetch the error-message
CONTROLLER DETECTED ERROR                  corresponding to the error-
PRIVILEGED COMMAND                         code 23(8).
```

5.2.37 GET-FUNCTION-CODE

Explanation

This command displays the driver function corresponding to the given function code. The function code tells what the MONITOR was doing when an error occurred.

Parameter

1. <FUNCTION CODE> A number between 0_8 and 21_8.

Example:

```
ENTER COMMAND: GET-FUNCTION-CODE 21 ↵        Comments:

FUNCTION CODE:  21B                        You fetch the meaning of the
                                           function code 21(8).
LOG ON UNPRIVILEGED
```

5.2.38 EXECUTE-PARALLEL-POLL

Explanation

This command executes a parallel poll of the devices configurated for this. The result of the parallel polling is displayed on your terminal. The parallel poll is frequently used by a device to indicate that it needs some kind of service. The parallel poll is executed by sending an IDY (Identify) command on the bus, and then receiving PPR (Parallel Poll Response) messages from one ore more devices.You must decode the meaning of the poll status yourself. As the expression for the PPE message (explained in section 5.2.12), the PPR message sent from a device can be given the following equation:

$PPR = ST * 2^B$.

$ST=0_8$ - PPR is sent false. $ST=10_8$ - PPR is sent true. B denotes the bit in the byte affected, and thereby which data line the PPR is sent on. $PPR = 0_8$ - PPR sent on DIO1. $PPR = 7_8$ - PPR is sent on DIO8. Which device is sending the PPR can also be figured out by looking at

the bit pattern.

Parameter

1. <CONTROLLER NUMBER> A number between 0_8 and 17_8.

Example:

```
ENTER COMMAND: EXECUTE-PARALLEL-POLL ↵

POLLSTATUS :  253B

Comments:
Suppose all devices are configurated to send the PPR true. The
pollstatus can be written as: 253(8) = 2^7 + 2^3 + 2^2 + 1. That means
PPR messages were received on DIO8, DIO4, DIO3 and DIO2.
```

5.2.39 PERMANENT-SRQ-INTERRUPT-ENABLE

Explanation

This command permanently establishes service request interrupt on the specified device(s). The command tells the GPIB-driver how RQS (Request Service) messages received from these devices are to be t eated. The serial poll sequence itself is automatically handled by the controller. You are able to receive all RQS messages from the specified devices after the controller has executed a serial poll. You disable yourself from getting these messages by executing the command SRQ-INTERRUPT-DISABLE or by releasing the devices.

Parameters

1. <CONTROLLER NUMBER> A number between 0_8 and 17_8.

2. <DEVICE NAME(S)> A string of maximum 20_8 characters for each dev.

Example:

```
ENTER COMMAND: PERMANENT-SRQ-INTERRUPT-ENABLE DA-CONVERTER ↵

Comments:
The device DA-CONVERTER is permanently enabled for service request.
```

5.2.40 TEMPORARY-SRQ-INTERRUPT-ENABLE

Explanation

This command temporarily establishes service request interrupt enable on the specified device(s). The command tells the GPIB-driver how RQS (Request Service) messages received from these devices are to be treated. The serial poll sequence itself is automatically handled by the controller. You are only able to receive the first RQS message sent from one of the specified devices. The SRQ-interrupt is then disabled on the specified device(s). You can also disable this interrupt by executing the SRQ-INTERRUPT-DISABLE command, or by releasing the device(s).

Parameters

1. <CONTROLLER NUMBER> A number between 0_8 and 17_8.

2. <DEVICE NAME(S)> A string of maximum 20_8 characters for each dev.

Example:

```
ENTER COMMAND: TEMPORARY-SRQ-INTERRUPT-ENABLE CALCULATOR ↵

Comments: The device CALCULATOR is temporarily enabled for service
request.
```

5.2.41 SRQ-INTERRUPT-DISABLE

Explanation

This command disables the SRQ-interrupt on the specified device(s). This means that none of the possible RQS messages from your specified device(s) are sent to you after they have been received by the controller.

Parameters

1. <CONTROLLER NUMBER> A number between 0_8 and 17_8.

2. <DEVICE NAME(S)> A string of maximum 20_8 characters for each dev.

Example:

```
ENTER COMMAND: SRQ-INTERRUPT-DISABLE SPACE-LAB ↵

Comments:
You do not want to receive RQS-messages from SPACE-LAB.
```

5.2.42 CHECK-FOR-INTERRUPT

Explanation

This command checks whether any RQS messages has been received since last time this command was executed. If any RQS messages has been received from your reserved device(s) you will get status information about this, and you must use this to figure out why the device sent the RSQ message. This command has no parameters.

Example:

```
ENTER COMMAND: CHECK-FOR-INTERRUPT ↵
SRQ-INTERRUPT RECEIVED
CONTROLLER NO. 000B  INSTRUMENT NO.001B
    POLL STATUS. 100B

Comments:
You examine whether any of your devices are requiring service.
RQS is received from the device with device number 1(8), belonging
to controller 0(8). The poll status received, 100(8), means that only
the RQS message was sent, without any additional status
information.
```

5.2.43 WAIT-FOR-INTERRUPT

Explanation

This command checks whether any RQS messages have been received since last time this command was executed. If any RQS messages have been received from your reserved device(s), you will get status information about this. If not, the MONITOR waits a predefined time to see whether any RSQ messages will be received. The time to wait is specified by the next command. The command has no parameters.

Example:

```
ENTER COMMAND: WAIT-FOR-INTERRUPT ↵

Comments:
You examine whether any of your devices are requiring service
or will require service during a predefined period.
```

5.2.44 CHANGE-INTERRUPT-WAIT-TIME

Explanation

This command changes the MONITOR's wait time for an SRQ-interrupt. The current value is displayed when the command is executed.

Parameters

1. <TIME UNIT> B, S, M or H.

   The parameter indicates what kind of time unit to use, together with the time value. B for basic time unit (20 [ms]), S for second, M for minute or H for hour.

2. <NUMBER OF TIME UNITS> An arbitrary number.

Example:

```
ENTER COMMAND: CHANGE-INTERRUPT-WAIT-TIME ↵     Comments:
CURRENT VALUE: 0012B SECOND(S)                  Old value is 10 seconds.
TIME UNIT (B/S/M/H): S ↵                        You select 20 seconds as
NUMBER OF TIMEUNITS: 20.D ↵                     the new value.
```

5.2.45 MODE

Explanation

This command makes it possible to run a set of MONITOR commands directly in the MONITOR. The mode file must already have been created in SINTRAN. The mode files which shall run in the MONITOR are made in the same way as a mode file created in SINTRAN, with one exception. The MONITOR's prompt is not written before the MONITOR commands.

It is also possible to run SINTRAN commands in the MONITOR's mode file. The SINTRAN commands must, as usual, be preceded by "@".

Parameter

1. <INPUT FILE> A SINTRAN file name

Example:

```
ENTER COMMAND: MODE DATA-RECEPTION ↵

Comments: You get the MONITOR to execute the mode file
DATA-RECEPTION. The file could contain the mode jobs:

LOG-ON
RESERVE-DEVICE LISA
RECEIVE-DATA LISA,5
SAVE-DATABUFFER DATABUFFER:DATA
@PED DATABUFFER:DATA
```

6 GPIB_LIBRARY

6.1 INTRODUCTION

The GPIB LIBRARY is a set of FORTRAN callable routines. This library allows you to perform the same functions as the commands you give directly to the MONITOR. In this way, the LIBRARY is an effective aid for developing testing and measuring procedures.

The parameters in the routines are described with the help of mnemonics. They consist of input and output parameters. Input parameters are specified by you in the program. The output parameters are returned by the routine itself.

The parameters' full names are written in the description of the parameters. All parameters are described in each routine, although many of them appear in several routines. It is hoped that this makes it easier to use the manual the first time. For experienced users Appendix B, which lists the routines briefly, will be interesting.

6.2 ROUTINES_CALLABLE_BY_ALL_USERS

6.2.1 LOG_ON_AS_NON-PRIVILEGED_USER

Format

GENT (STAT,MVERS,cntno,USNO)

Explanation

This routine logs you on to a specified GPIB controller, as non-privileged user. If the specified controller is not active, an error message is returned by the routine. If the log on procedure succeeds, status code 0_8 indicating no errors is returned together with microprogram version and user number.

Input parameter

1. <cntno> Controller number. An integer between 0_8 and 17_8.

Output parameters

1. <STAT> Error code. An integer between 0_8 and 437_8.

2. <MVERS> Microprogram version. A character string. An integer.

3. <USNO> User number. An integer between 1_8 and 17_8.

6.2.2 LOG_OFF_CONTROLLER

Format

GEXIT (STAT,cntno)

Explanation

This routine logs you off a controller. All your reserved devices are released.

Input parameter

1. <cntno> Controller number. An integer between 0_8 and 17_8.

Output parameter

1. <STAT> Error code. An integer between 0_8 and 437_8.

6.2.3 RESERVE_DEVICE(S)

Format

GRES (STAT,cntno,nodev,devar)

Explanation

This routine reserves the specified device(s) for you. Before any operation can be performed on a device, it must be reserved by you. Single-user devices can only be reserved by one user, and multi-user devices by many users simultaneously. If the desired device cannot be reserved, you get an error message. Whether a device is for single or multi-users are decided when the device is configurated.

Input parameters

1. <cntno> Controller number. An integer between 0_8 and 17_8.

2. <nodev> The number of specified device numbers. An integer.

3. <devar> Array containing the device numbers. An integer.

Output parameter

1. <STAT> Error code. An integer between 0_8 and 437_8.

6.2.4 RELEASE_DEVICE(S)

Format

GREL (STAT,cntno,nodev,devar)

Explanation

This routine releases the specified device(s) for a user. This routine can only be performed by the user who reserved the specified device(s).

Input parameters

1. <cntno> Controller number. An integer between 0_8 and 17_8.

2. <nodev> The number of specified device numbers. An integer.

3. <devar> Array containing the device numbers. An integer.

Output parameter

1. <STAT> Error code. An integer between 0_8 and 437_8.

6.2.5 TRIGGER_DEVICE(S)

Format

GTRGD (STAT,cntno,nodev,devar)

Explanation

This routine sends a trigger command to the specified device(s). The basic operations in the device(s) are then started (trigged).

Input parameters

1. <cntno> Controller number. An integer between 0_8 and 17_8.

2. <nodev> The number of specified device numbers. An integer.

3. <devar> Array containing the device numbers. An integer.

Output parameter

1. <STAT> Error code. An integer between 0_8 and 437_8.

6.2.6 SET_DEVICE(S)_IN_LOCAL

Format

GLOCD (STAT,cntno,nodev,devar)

Explanation

This routine sets the specified device(s) in local mode. In local mode a device is only operated from its local controls.

Input parameters

1. <cntno> Controller number. An integer between 0_8 and 17_8.

2. <nodev> The number of specified device numbers. An integer.

3. <devar> Array containing the device numbers. An integer.

Output parameter

1. <STAT> Error code. An integer between 0_8 and 437_8.

6.2.7 CLEAR_DEVICE(S)

Format

GCLRD (STAT,cntno,nodev,devar)

Explanation

This routine sends a SDC (Selected Device Clear) command to the specified devices. A cleared device is in a predefined , device-dependent quiescent state.

Input parameters

1. <cntno> Controller number. An integer between 0_8 and 17_8.

2. <nodev> The number of specified device numbers. An integer.

3. <devar> Array containing the device numbers. An integer.

Output parameter

1. <STAT> Error code. An integer between 0_8 and 437_8.

6.2.8 SEND_DATA_TO_DEVICE(S)

Format

GWRIT (STAT,cntno,nodev,devar,bytco,datad)

Explanation

This routine sends data from the controller to the specified device(s).

Input parameters

1. <cntno> Controller number. An integer between 0_8 and 17_8.

2. <nodev> The number of specified device numbers. An integer.

3. <devar> Array containing the device numbers. An integer.

4. <bytco> The number of specified bytes to send. An integer.

5. <datad> Array containing the data to send.

Output parameter

1. <STAT> Error code. An integer between 0_8 and 437_8.

6.2.9 READ_DATA_FROM_DEVICE

Format

GREAD (STAT,cntno,nodev,devar,bytco,DATAD,RBYTCO)

Explanation

This routine receives data from a talking device. There can be other devices listening to the transfer. The device last specified is the talker, and the other devices are listeners. If the EOI-line is set true, or an EOS-character is detected before the maximum number of bytes has been received, the transfer will be terminated.

Input parameters

1. <cntno> Controller number. An integer between 0_8 and 17_8.

2. <nodev> The number of specified device numbers. An integer.

3. <devar> Array containing the device numbers. An integer.

4. <bytco> The maximum number of bytes to receive. An integer.

Output parameters

1. <STAT> Error code. An integer between 0_8 and 437_8.

2. <DATAD> Array containing the received data.

3. <RBYTCO> The number of bytes read in the transfer. An integer.

6.2.10 TRANSFER DATA BETWEEN DEVICES

Format

GTRNS (STAT,cntno,nodev,devar,bytco)

Explanation

This routine transfers data between specified devices. The device last specified is the talker, and the others devices are listeners. If the EOI-line is set true or an EOS-character is detected before the maximum number of bytes has been received, the transfer will be terminated.

Input parameters

1. <cntno> Controller number. An integer between 0_8 and 17_8.

2. <nodev> The number of specified device numbers. An integer.

3. <devar> Array containing the device numbers. An integer.

4. <bytco> The maximum number of bytes to transfer. An integer.

Output parameter

1. <STAT> Error code. An integer between 0_8 and 437_8.

6.2.11 LIST THE SYSTEM DEVICE(S)

Format

GSYSD (STAT,cntno,RBYTCO,DATAD)

Explanation

This routine returns information about all configurated devices on a controller's interface bus.

Input parameter

1. <cntno> Controller number. An integer between 0_8 and 17_8.

Output parameters

1. <STAT> Error code. An integer between 0_8 and 437_8.
2. <RBYTCO> The number of configurated devices. An integer.
3. <DATAD> Array containing returned device information. Integers.

   The returned device information consists of 6 bytes for each device. The array has the format:

   ←— Bit numbers
   15 14 8 7 0

P	T	Device number	Configuration byte
Listen address			Talk address
Reserved by user(s)			

   The fields in the array mean:

   - P - SRQ-interrupt enabled permanently.
   - T - SRQ-interrupt enabled temporarily.
   - Listen and talk addresses as described in the configurate device routine.
   - Device number as described in the get device number routine.
   - Configuration byte as further described.

The configuration bits have the meaning (the same parameters described in the configurate device routine. False - 0. True - 1.):

- Bit 0 - EOS terminates input.
- Bit 1 - EOS-character used.
- Bit 2 - EOI after last byte.
- Bit 3 - EOI with last byte.
- Bit 4 - Multi-user device.
- Bit 5 - SRQ allowed.
- Bit 6 - Remote bus.
- Bit 7 - Timeout enabled.

The field Reserved by user(s) has the meaning:

Every user is assigned an own bit. For each user who has reserved the device this bit is set. The bit number assigned for a user corresponds to the user number.

6.2.12 LIST_THE_USER_DEVICE(S)

Format

GSYSD (STAT,cntno,RBYTCO,DATAD)

Explanation

This routine returns information about all your configurated devices on a controller's interface bus. The format and meaning of the returned information were explained in the previous routine.

Input parameter

1. <cntno> Controller number. An integer between 0_8 and 17_8.

Output parameters

1. <STAT> Error code. An integer between 0_8 and 437_8.
2. <RBYTCO> The number of your reserved devices. An integer.
3. <DATAD> Array containing returned device information. Integers.

6.2.13 GET_DEVICE_NUMBER

Format

GGTNO (STAT,cntno,devnam,DEVNO)

Explanation

This routine returns the device number of the specified device name. A device automatically gets a device number by the controller when the device is configurated. You should execute this routine to get the device number to the other routines where it is needed.

Input parameters

1. <cntno> Controller number. An integer between 0_8 and 17_8.

2. <devnam> Device name. A string of maximum 20_8 characters. An integer

Output parameters

1. <STAT> Error code. An integer between 0_8 and 437_8.

2. <DEVNO> Device number. An integer between 0_8 and 37_8.

6.2.14 PARALLEL_POLL_CONFIGURATE_DEVICE

Format

GPPCD (STAT,cnto,devno,conf)

Explanation

This routine sends a parallel poll configuration byte to the specified device. This byte tells how the device shall respond with its PPR (Parallel Poll Response) message during a parallel poll. The parallel poll configuration byte is called PPE (Parallel Poll Enable) in the GPIB standard. See also the description of parallel poll in chapter 2.

Input parameters

1. <cntno> Controller number. An integer between 0_8 and 17_8.

2. <devno> Device number. An integer between 0_8 and 37_8.

3 <conf> Configuration byte.

The configuration byte is calculated in this way:

PPE = 140_8 + ST + B,

where S and B are variables. ST=0_8 makes the device send PPR false, and ST=10_8 makes it send PPR true. The variable B says which data line the PPR shall be sent on, and thereby which corresponding bit in the byte the message shall affect. B is 0_8 makes the device send PPR on DIO1, and B=7_8 makes it select DIO8.

Output parameter

1. <STAT> Error code. An integer between 0_8 and 437_8.

6.2.15 EXECUTE_PARALLEL_POLL

Format

GEXPP (STAT,cnto,PPSTAT)

Explanation

This routine executes a parallel poll of the devices configurated for this. The result of the parallel polling is returned to the output parameter PPSTAT. The meaning of the parallel poll status must be decoded by your program.

Input parameter

1. <cntno> Controller number. An integer between 0_8 and 17_8.

Output parameters

1. <STAT> Error code. An integer between 0_8 and 437_8.

2. <PPSTAT> Parallel poll status.

You must decode the meaning of the poll status yourself. As the expression for the PPE message (explained in section 5.2.12), the PPR message sent from a device can be given the following equation:

PPR = ST * 2^B.

ST=0_8 - PPR is sent false. ST=10_8 - PPR is sent true. B denotes the bit in the byte affected, and thereby which data line the PPR is sent on. PPR =0_8 - PPR sent on DIO1. PPR=7_8 - PPR is sent on DIO8. Which device is sending the PPR can also be figured out by looking at the bit pattern.

6.2.16 WRITE_ERROR_MESSAGE_AND_STOP

Format

GSERM (stat)

Explanation

This routine writes a short error message on the terminal, corresponding to the given error code, and the program terminates. If the routine is executed in a RT program, the error message is written in the format:

HH.MM.SS ERROR 50 IN <RT-name> AT <adr>; USER ERROR. SUBERROR: <stat>

where <RT-name> is the name of the RT-program, <adr> is an address inside the GSERM routine and <stat> is error code as decimal number.

Input parameter

1. <stat> Error code. An integer between 0_8 and 437_8.

6.2.17 WRITE_ERROR_MESSAGE_AND_CONTINUE

Format

GCERM (stat)

Explanation

This routine writes a short error message on the terminal, corresponding to the given error code, and it returns to the calling program. If the routine is executed in a RT-program, the error message is written in the same format as explained in the previous routine.

Input parameter

1. <stat> Error code. An integer between 0_8 and 437_8.

6.2.18 ENABLE_SRQ-INTERRUPT_PERMANENTLY

Format

GPESR (STAT,cntno,nodev,devar)

Explanation

This routine permanently establishes service request interrupt on the specified device(s). The routine tells the GPIB-driver how RQS (Request Service) received messages from these devices are to be treated. The serial poll sequence itself is automatically handled by the controller. You are able to receive all RQS messages from the specified devices after the controller has executed a serial poll. You disable yourself from getting these messages by executing the SRQ-interrupt disable routine, or by releasing the devices.

Input parameters

1. <cntno> Controller number. An integer between 0_8 and 17_8.

2. <nodev> The number of specified device numbers. An integer.

3. <devar> Array containing the device numbers. An integer.

Output parameter

1. <STAT> Error code. An integer between 0_8 and 437_8.

6.2.19 ENABLE_SRQ-INTERRUPT_TEMPORARILY

Format

GTESR (STAT,cntno,nodev,devar)

Explanation

This routine temporarily establishes service request interrupt enable on the specified device(s). The routine tells the GPIB-driver how RQS (Request Service) received messages from these devices are to be treated. The serial poll sequence itself is automatically handled by the controller. You are only able to receive the first RQS message sent from one of the specified devices. The SRQ-interrupt is then disabled on the specified device(s). You can also disable this interrupt by executing the SRQ-interrupt disable routine, or by releasing the device(s).

Input parameters

1. <cntno> Controller number. An integer between 0_8 and 17_8.

2. <nodev> The number of specified device numbers. An integer.

3. <devar> Array containing the device numbers. An integer.

Output parameter

1. <STAT> Error code. An integer between 0_8 and 437_8.

6.2.20 DISABLE_SRQ-INTERRUPT

Format

GDISR (STAT,cntno,nodev,devar)

Explanation

This routine disables the SRQ-interrupt on the specified device(s). This means that none of the possible RQS messages from your specified device(s) are sent to you after they have been received by the controller.

Input parameters

1. <cntno> Controller number. An integer between 0_8 and 17_8.

2. <nodev> The number of specified device numbers. An integer.

3. <devar> Array containing the device numbers. An integer.

Output parameter

1. <STAT> Error code. An integer between 0_8 and 437_8.

6.2.21 CHECK_FOR_SRQ-INTERRUPT

Format

GCSRQ (STAT,CNTNO,DEVNO)

Explanation

This routine checks whether any RQS messages have been received since last time this routine was executed. If any RQS messages have been received from your reserved device(s) you will get status information about this, and you must use this to figure out why the device sent the RSQ message.

Output parameters

1. <STAT> Pollstatus. An integer between 0_8 and 377_8.

2. <CNTNO> Controller number. An integer between 0_8 and 17_8.

3. <DEVNO> Device number. An integer between 0_8 and 37_8.

6.2.22 WAIT_FOR_SRQ-INTERRUPT

Format

GWSRQ (STAT,CNTNO,DEVNO,timeunit,timeno)

Explanation

This routine checks whether any RQS messages have been received since last time this routine was executed. If any RQS messages have been received from your reserved device(s) you will get status information about this. If not, the MONITOR waits a predefined time to see whether any RSQ messages will be received. The time to wait is specified in the routine.

Input parameters

1. <time unit> An integer between 1 and 4.

   This number indicates the time unit to wait. 1 for basic time unit (20 [ms]), 2 for second, 3 for minute and 4 for hour.

2. <timeno> The number of time units to wait. An arbitrary integer.

Output parameters

1. <STAT> Pollstatus. An integer between 0_8 and 377_8.

2. <CNTNO> Controller number. An integer beetween 0_8 and 17_8.

3. <DEVNO> Device number. An integer between 0_8 and 37_8.

6.3 ROUTINES CALLABLE BY THE PRIVILEGED USER

6.3.1 LOG ON AS PRIVILEGED USER

Format

GPENT (STAT,MVERS,cntno)

Explanation

This routine logs you on to a specified controller as privileged user. That means you get user number 0_8. The program calling this routine must be running under a user name in SINTRAN which is allowed to perform this. If the log on procedure succeeds, status code 0_8, indicating no errors, is returned together with the microprogram version.

Input parameter

1. <cntno> Controller number. An integer between 0_8 and 17_8.

Output parameters

1. <STAT> Error code. An integer between 0_8 and 437_8.

2. <MVERS> Microprogram version. A character string. An integer.

6.3.2 CONFIGURATE DEVICE

Format

```
GCDEV (STAT,cntno,DEVNO,devnam,laddr,taddr,secad,timeout,remote,serpo,
       siuser,eoiaft,eoiwit,eos,eosi,eosch)
```

Explanation

This routine configurates a new device on a controller's bus. The device gets listen address, talk address, etc., according to the parameters in the call.

Input parameters

1. <cntno> Controller number. An integer between 0_8 and 17_8.

2. <devnam> Device name. A string of maximum 20_8 characters.
An integer.

3. <laddr> Listen address. An integer between 40_8 and 76_8.

This parameter is the device's listen address. The number 77_8 is the GPIB unlisten message.

4. <taddr> Talk address. An integer between 100_8 and 136_8.

This parameter is the device's talk address. The number 137_8 is the GPIB untalk message.

5. <secad> Secondary address? Logical.

If the device has a secondary address, it must always be specified after the device number when we address the device.

6. <timeout> Timeout enabled? Logical.

If the device uses more than 16 [msec] to send or receive the first byte after it has been addressed, the answer should be false.

7. <remote> Remote bus? Logical.

The answer is true if the device is connected to the remote bus, and false if it is connected to the local bus.

8. <serpo> Serial poll allowed? Logical.

If the device has the service request (and serial poll) function (SR1), the answer should be true. If the answer is false, the controller will not poll the device if a SRQ is received.

NOTE A device which sends a service request and is <u>not</u> polled can jam the controller.

9. <siuser> Single-user? Logical.

If only one user is allowed to use the device, the answer should be true. If several users are allowed to use the device, the answer should be false.

10. <eoiaft> EOI after last byte? Logical.

This and the next parameter affect the way the EOI-line of the controller is set at the end of a data transfer. If EOI should be set after the last byte, the answer should be true, otherwise it should be false.

11. <eoiwit> EOI with last byte? Logical.

    IfEOI should be set with the last byte, the answer should be true, otherwise it should be false. The combination of false, false in parameter 10 and 11 indicates that the device does not need EOI at all. The combination true, true is not legal.

12. <eos> EOS-character used? Logical.

    If the device uses an EOS-charater when exchanging data the answer should be true, otherwise it should be false.

13. <eosi> EOS-character terminates interrupt? Logical.

    If a detected EOS-character from the device should terminate the input of data to the controller the answer should be true, otherwise it should be false.

14. <eosch> EOS-character. An integer between 0_8 and 377_8.

    The ASCII value of the EOS-character, if any.

Output parameters

1. <STAT> Error code. An integer between 0_8 and 437_8.

2. <DEVNO> Device number. An integer between 0_8 and 37_8.

6.3.3 DELETE_DEVICE

Format

GDEL (STAT,cntno,devno)

Explanation

This routine deletes a device from the controller's device list and the GPIB configuration file.

Input parameters

1. <cntno> Controller number. An integer between 0_8 and 17_8.

2. <devno> Device number. An integer between 0_8 and 37_8.

Output paramter

1. <STAT> Error code. An integer between 0_8 and 437_8.

6.3.4 SET_CONTROLLER_ADDRESS

Format

GDDEV (STAT,cntno,taddr)

Explanation

This routine changes the specified controller's talk and listen addresses. After system initiation the controller's listen address is 65_8 and the talk adress is 125_8. Only the new talk address is specified in the routine. The listen address is automatically calculated in this way:

Listen address = Talk address - 40_8.

Input parameters

1. <cntno> Controller number. An integer between 0_8 and 17_8.

2. <taddr> New talk address. An integer between 0_8 and 136_8.

Output paramter

1. <STAT> Error code. An integer between 0_8 and 437_8.

6.3.5 SEND_INTERFACE_CLEAR

Format

GSIFC (STAT,cntno)

Explanation

This routine puts all the devices' interfaces on the bus into a predefined quiescent state.

Input parameter

1. <cntno> Controller number. An integer between 0_8 and 17_8.

Output parameter

1. <STAT> Error code. An integer between 0_8 and 437_8.

6.3.6 CLEAR_CONTROLLER

Format

GCLR (STAT,cnto)

Explanation

This routine clears the the GPIB controller into a predefined quiescent state, but all information about users and device configuration will remain unchanged.

Input parameter

1. <cntno> Controller number. An integer between 0_8 and 17_8.

Output parameter

1. <STAT> Error code. An integer between 0_8 and 437_8.

A P P E N D I X A

MONITOR COMMANDS (in alphabetical order)

MONITOR COMMANDS (IN ALPHABETICAL ORDER)

Command name	Parameters

```
CHANGE-DEVICE-CONFIGURATION [cntno,devnam,laddr,taddr,secad,timeout,
                             loc/rem,serpo,sing/mult,eoi-w/a/n,eos,
                             eosi,eosch]
CHANGE-INTERRUPT-WAIT-TIME <timeunit,timeno>
CHECK-FOR-INTERRUPT
CLEAR-CONTROLLER [cntno]
CLEAR-DATABUFFER
CLEAR-DEVICES <cntno,devnam>
CONFIGURATE-DEVICE [cntno,devnam,laddr,taddr,secad,timeout,
                    loc/rem,serpo,sing/mult,eoi-w/a/n,eos,
                    eosi,eosch]
DELETE-DEVICE [cntno,devnam]
DUMP-DATABUFFER <fbyte,bytco,{radix}>
EXECUTE-PARALLEL-POLL <cntno>
EXIT
GET-ERROR-MESSAGE <stat>
GET-FUNCTION-CODE <stat>
HELP <command>
INTERFACE-CLEAR <cntno>
LIST-SYSTEM-DEVICES <cntno,{devnam},{outfile}>
LIST-USER-DEVICES <cntno,{devnam},{outfile}>
LOAD-DATABUFFER <infile>
LOCAL-DEVICES <cntno,devnam>
LOG-OFF <cntno>
LOG-ON <cntno,[priv]>
MODE <infile>
PERMANENT-SRQ-INTERRUPT-ENABLE <cntno,devnam>
RECEIVE-DATA <cntno,devnam,bytco>
RECONFIGURATE [cntno]
RELEASE-CONTROLLER <cntno>
RELEASE-DEVICE <cntno,devnam>
RESERVE-CONTROLLER <cntno>
RESERVE-DEVICE <cntno,devnam>
RESET-DEFAULT-CONTROLLER
RESET-REN-LINE [cntno]
SAVE-DATABUFFER <outfile>
SEND-ASCII-STRING <cntno,devnam,string>
SEND-DATA <cntno,devnam,bytco>
SEND-PARALLEL-POLL-CONFIGURATION <cntno,devnam,conf>
SET-CONTROLLER-ADDRESS [cntno,taddr]
SET-DEFAULT-CONTROLLER <cntno>
SET-RADIX <radix>
SET-REN-LINE [cntno]
SRQ-INTERRUPT-DISABLE <cntno,devnam>
TEMPORARY-SRQ-INTERRUPT-ENABLE <cntno,devnam>
TRANSFER-DATA <cntno,devnam,bytco>
TRIGGER-DEVICES <cntno,devnam,>
WAIT-FOR-INTERRUPT
WHO-IS-ON <cntno>
```

EXPLANATION OF PARAMETERS USED

The parameters used to describe the commands are written as mnemonics. The brackets around the parameters are significant: <> mean commands or parameters that can be used by all users. {} mean parameters that can be used by all users, but which the MONITOR does not asks for. [] mean commands or parameters that can only be used by the privileged user.

The parameters used are listed in the following table:

Mnemonic	Parameter name	Legal values (octal)
BYTCO	Number of bytes	Arbitrary
CNTNO	Controller number	0 - 17
CONF	Configuration byte	See PPE in App. E
DEVNAM	Device name(s)	Max 20 characters
EOI-W/A/N	EOI with, after or none?	W,A,N
EOS	EOS-character used?	Y,N
EOSCH	EOS-character value	0 - 377
EOSI	EOS terminates interrupt?	Y,N
FBYTE	First byte	0 - 17777
INFILE	Input file	SINTRAN syntax
LADDR	Listen address	40 - 76
LOC/REM	Local or remote bus	L,R
OUTFILE	Output file	SINTRAN syntax
PRIV	Privileged user?	Y,N
RADIX	Radix indicator	B,D,H
SECAD	Secondary address?	Y,N
SERPO	Serial poll allowed?	Y,N
SING/MULT	Single or multi user?	S,M
STAT	Error code	0 - 437
TADDR	Talk address	100 - 136
TIMENO	Number of time units	Arbitrary
TIMEOUT	Timeout enabled?	Y,N
TIMEUNIT	Time unit	B,S,M,H

A P P E N D I X B

LIBRARY ROUTINES (in alphabetical order)

LIBRARY ROUTINES (IN ALPHABETICAL ORDER)

Routine name	Routine call name	Parameters
CHECK FOR SRQ	GCSRQ	(STAT,CNTNO,DEVNO)
CLEAR CONTROLLER	GCLRC	[STAT,cntno]
CLEAR DEVICES	GCLRD	(STAT,cntno,nodev,devar)
CONFIGURATE DEVICE	GCDEV	[STAT,cntno,DEVNO,devnam,laddr, taddr,secad,timeout,remote,serpo, siuser,eoiaft,eoiwit,eos,eosi,eosch]
DELETE DEVICE	GDEL	[STAT,cntno,devno]
DISABLE SRQ	GDISR	(STAT,cntno,nodev,devar)
ENABLE SRQ PERMANENTLY	GPESR	(STAT,cntno,nodev,devar)
ENABLE SRQ TEMPORARILY	GTESR	(STAT,cntno,nodev,devar)
EXECUTE PARALLEL POLL	GEXPP	(STAT,cntno,PPSTAT)
GET DEVICE NUMBER	GGTNO	(STAT,cntno,devnam,DEVNO)
LIST SYSTEM DEVICES	GSYSD	(STAT,cntno,RBYCO,DATAD)
LIST USER DEVICES	GUSED	(STAT,cntno,RBYCO,DATAD)
LOG OFF CONTROLLER	GEXIT	(STAT,cntno)
LOG ON PRIVILEGED	GPENT	[STAT,MVERS,cntno]
LOG ON UNPRIVILEGED	GENT	(STAT,MVERS,cntno,USNO)
PARALLEL POLL CONFIGURATE	GPPCD	(STAT,cntno,devno,conf)
READ DATA FROM DEVICES	GREAD	(STAT,cntno,nodev,devar,bytco, DATAD, RBYTCO)
RELEASE DEVICES	GREL	(STAT,cntno,nodev,devar)
RESERVE DEVICES	GRES	(STAT,cntno,nodev,devar)
SEND DATA TO DEVICES	GWRIT	(STAT,cntno,nodev,devar,bytco,datad)
SEND INTERFACE CLEAR	GSIFC	[STAT,cntno]
SET CONTROLLER ADDRESS	GDDEV	[STAT,cntno,taddr]
SET DEVICES IN LOCAL	GLOCD	(STAT,cntno,nodev,devar)
TRANSFER DATA BETWEEN DEV.	GTRNS	(STAT,cntno,nodev,devar,bytco)
TRIGGER DEVICES	GTRGD	(STAT,cntno,nodev,devar)
WAIT FOR SRQ	GWSRQ	(STAT,CNTNO,DEVNO,TIMEUNIT,TIMENO)
WRITE ERR. MESS. AND CONT.	GCERM	(stat)
WRITE ERR. MESS. AND STOP	GSERM	(stat)

EXPLANATION OF PARAMETERS USED

The parameters used to describe the routines are written as mnemonics. The brackets around the parameters are significant: () mean routines that are available for all users. [] mean routines that can only be used of the privileged user. Input parameters written by the user are written in lowercase. Output parameters returned by the Library are written in uppercase.

The parameters used are listed in the following table:

Mnemonic	Parameter name	Type
BYTCO	Number of bytes	Integer
CNTNO	Controller number	Integer
CONF	Configuration byte	Integer
DATAD	Data array	Integer
DEVAR	Device number array	Integer
DEVNAM	Device name	String
DEVNO	Device number	Integer
EOIAFT	EOI after?	Logical
EOIWIT	EOI with?	Logical
EOS	EOS-character?	Logical
EOSCH	EOS-character	Integer
EOSI	EOS ends interrupt?	Logical
LADDR	Listen address	Integer
MVERS	Microprogram version	Integer
NODEV	Number of devices in DEVAR	Integer
PPSTAT	Parallel poll status	Integer
REMOTE	Device on remote bus?	Logical
SECAD	Secondary address?	Logical
SERPO	Serial poll allowed?	Logical
SIUSER	Single user?	Logical
STAT	Error code or pollstatus	Integer
TADDR	Talk address	Integer
TIMENO	Number of time units	Integer
TIMEOUT	Timeout enabled?	Logical
TIMEUNIT	Time unit	Integer
USNO	User number	Integer

A P P E N D I X C

ERROR CODES (in octal values)

ERROR CODES (IN OCTAL VALUES)

Controller detected errors (00B-77B):

00B - NO ERROR

The controller card is neither the system controller nor, the controller-in- charge:

01B - ILLEGAL FUNCTION CODE for nonprivileged user, with parameters.
02B - ILLEGAL FUNCTION CODE for nonprivileged user, without parameters
03B - ILLEGAL FUNCTION CODE for privileged user, with parameters.
04B - ILLEGAL FUNCTION CODE for privileged user, without parameters.

The controller card is controller-in-charge, but not the system controller:

05B - ILLEGAL FUNCTION CODE for nonprivileged user, with parameters.
06B - ILLEGAL FUNCTION CODE for nonprivileged user, without parameters
07B - ILLEGAL FUNCTION CODE for privileged user, with parameters.
10B - ILLEGAL FUNCTION CODE for privileged user, without parameters.

The controller card is the system controller, but not the controller-in-charge:

11B - ILLEGAL FUNCTION CODE for nonprivileged user, with parameters.
12B - ILLEGAL FUNCTION CODE for nonprivileged user, without parameters
13B - ILLEGAL FUNCTION CODE for privileged user, with parameters.
14B - ILLEGAL FUNCTION CODE for privileged user, without parameters.

The controller card is the system controller and controller-in-charge:

15B - ILLEGAL FUNCTION CODE for nonprivileged user, with parameters.
16B - ILLEGAL FUNCTION CODE for nonprivileged user, without parameters
17B - ILLEGAL FUNCTION CODE for privileged user, with parameters.
20B - ILLEGAL FUNCTION CODE for privileged user, without parameters.

21B - PRIVILEGED COMMAND when the controller card is neither system controller, nor controller-in-charge.
22B - PRIVILEGED COMMAND when the controller card is controller-in-charge, but not the system controller.
23B - PRIVILEGED COMMAND when the controller card is system controller, but not controller-in-charge.
24B - PRIVILEGED COMMAND when the controller card is the system controller and controller-in-charge.

This command can not be executed (or wrong parameters):

40B - DEVICE NOT DEFINED when trying to reserve it.
41B - DEVICE ALREADY IN USE when trying to use a single user device
42B - USER IS NOT DEFINED.
43B - DEVICE NOT RESERVED when trying to send listen address.
44B - DEVICE NOT RESERVED when trying to send talk address.
45B - DEVICE NOT RESERVED when trying to send parallel poll configuration.
46B - DEVICE NOT RESERVED when trying to enable SRQ.
47B - DEVICE NOT ALLOWED when trying to enable SRQ.
50B - LISTEN ADDRESS NOT OK (bit 6=0, bit 5=1).
51B - TALK ADDRESS NOT OK (bit 6=1, bit 5=0).
52B - SECONDARY ADDRESS NOT OK (bit 6=1, bit 5=1)
53B - ILLEGAL COMBINATION OF LISTENERS.
54B - DEVICES HAVE DIFFERENT EOS.
55B - DEVICE NOT RESERVED when trying to release a device.
56B - REMOTE BOX POWER OFF.
57B - REMOTE BOX NOT CONNECTED.

These are additional fatale errors reported from the controller card:

60B - HANDSHAKE ERROR
61B - UNRECOGNISED SECONDARY COMMAND
62B - PARALLEL POLL ERROR
63B - TAKE CONTROLL ERROR
65B - ERROR REGISTER 0 INTERRUPT
66B - ERROR REGISTER 1 INTERRUPT (Z 80)
67B - FATAL ERROR IN RAM OR PROM
70B - DEVICE ASKING FOR SRQ IS NOT DEFINED
72B - TIMEOUT - GPIB-BUS HANGUP.
73B - NO DATA FROM DEVICE,timeout - GPIB-bus hangup.

Driver detected errors (100B-177B):

100B - ILLEGAL FUNCTION CODE.
101B - MEMORY ADDRESS REGISTER ERROR.
102B - PRIVILEGED COMMAND.
103B - MAXIMUM NO. OF USERS ALREADY LOGGED ON.
104B - PRIVILEGED USER ALREADY LOGGED ON.
105B - TIME OUT, (software).
106B - DATA BUFFER TOO SMALL.
107B - CONTROLLER NOT ACTIVE.
110B - WRONG MICROPROGRAM VERSION.

Library detected errors (200B-277B):

200B - NO SUCH DEVICENAME.
201B - PRIVILEGED COMMAND.
202B - CONTROLLER NUMBER OUT OF RANGE.
203B - YOU ARE NOT LOGGED ON THIS CONTROLLER.
204B - XMSG NOT STARTED.
205B - NO DEVICES CONFIGURATED FOR THIS CONTROLLER.
206B - DEVICE ALLREADY EXISTS.
207B - ILLEGAL TRANSFER LENGTH FOR READ/WRITE.
210B - DEVICE NOT DEFINED.
211B - CALL NOT ALLOWED IN SINTRAN-RTP.

XMSG detected errors (300B-377B):

300B - NO MORE XT-BLOCKS FREE
301B - NON-LOCAL REMOTE PORT ILLEGAL HERE
303B - TASK IS NOT ALLOWED ANY MORE MEMORY
304B - FACILITY IS NOT YET IMPLEMENTED
305B - ILLEGAL MESSAGE BUFFER POINTER
306B - MESSAGE BUFFER NOT YOURS
307B - ILLEGAL SERVICE PROGRAM CALLING
310B - NO MORE PORTS AVAILABLE
311B - FUNCTION NOT AVAILABLE TO DRIVERS
312B - NO DEFAULT MESSAGE
313B - MESSAGE IS ALLREADY CHAINED
314B - MESSAGE IS IN A QUEUE
315B - XMSG KERNEL ALREADY INITIALISED
316B - XMSG CRASH (INFO IN XMSG BASEFIELD)
317B - WRITE NOT ALLOWED (INDIRECT BUFFER)
320B - NO VALID INDIRECT BUFFER DEFINED
321B - ILLEGAL FUNCTION CODE IN MONITOR CALL
322B - INVALID MAGIC NUMBER
323B - MESSAGE SPACE FULL
324B - ILLEGAL MESSAGE SIZE
325B - ILLEGAL PORT NUMBER
326B - PRIVILEGED FUNCTION CALLED WITHOUT PRIVILEGE
327B - PRIVILEGE REQUEST REFUSED
330B - REMOTE MACHINE NOT AVAILABLE
331B - REMOTE TASK SPACE OVERFLOW
332B - MESSAGE ALREADY HAS XMSG BUFFER (XFDUB)
333B - XMSG LOCKED
334B - NO PORT OPEN (SO "DEFAULT PORT" PARAM INVALID)
335B - ILLEGAL TRANSFER LENGTH FOR READ/WRITE
336B - ILLEGAL DISPLACEMENT IN READ/WRITE
337B - ILLEGAL USE OF REENTRANT SEGMENT IN XFDIB
340B - INDIRECT BUFFER NOT ON VALID SEGMENT
341B - NETWORK SEQUENCING ERROR
342B - REMOTE MACHINE NOT DEFINED
344B - XMSG NOT STARTED

XROUT detected errors (400B-477B):

400B - ILLEGAL SERVICE NUMBER
401B - NO OPEN PORT HAS THIS NAME
402B - ANOTHER PORT ALEADY HAS THIS PORT
403B - NO SPACE LEFT FOR NAMES
404B - ILLEGAL PARAMETER TYPE
405B - MISSING MANDATORY PARAMETER
406B - UNKNOWN MAGIC NUMBER
407B - RESULTING MESSAGE TOO LONG
410B - STANDARD MESSAGE FORMAT NOT HANDLED
411B - CALLER WAS NOT PRIVILEGED
412B - ILLEGAL MACHINE NUMBER PARAMETER
413B - CANNOT ACCESS REMOTE XROUT
414B - ILLEGAL CLUSTER NUMBER PARAMETER
415B - ILLEGAL PIOC NUMBER PARAMETER
416B - INVALID SERVICE REQUEST - NO MULTI-MC XMSG

417B - ILLEGAL/RESERVED LOG. UNIT NO. FOR LINK
420B - NO MORE XL-BLOCKS (LINK DESCRIPTORS)
421B - NOT ENOUGH XD-BLOCKS FOR LKINI
422B - NO TRACE GENERATED
423B - TRACE ALREADY ACTIVE
424B - TRACE PASSIVE
425B - TRACE FILE OPEN ERROR (SE PARAM 1)
426B - TRACE RT-PROG (XTRACE) NOT FOUND
427B - ILLEGAL SYSTEM NUMBER
430B - BAD LINK - OPEN UNSUCCESSFUL
431B - ATTEMPT TO REDEFINE LOCAL MACHINE NO.
432B - LOCAL MACHINE NUMBER NOT YET DEFINED
433B - TOO MANY REMOTE NAMES TO THIS MACHINE
434B - OLD LETTER CALLS (SERVICE 2) CANNOT USE XMSG
435B - ALL CONNECTIONS WITH THIS NAME BUSY
436B - THIS IS NOT A CONNECT PORT
437B - REMOTE PORT STATICALLY DECLARED

A P P E N D I X D

FUNCTION CODES (in octal values)

FUNCTION CODES (IN OCTAL VALUES)

00B - DMA INPUT
01B - DMA OUTPUT
02B - READ STATUS
03B - CLEAR DEVICE
04B - PIO INPUT
05B - PIO OUTPUT
06B - ENABLE SRQ INTERRUPT
07B - DISABLE ALL INTERRUPTS
10B - EXECUTE SERIAL POLL
11B - EXECUTE PARALLEL POLL
12B - READ SYSTEM DEVICE LIST
13B - READ USER DEVICE LIST
14B - SEND CONTROL STRING
15B - SEND COMMAND BYTE WITH DEVICE LIST
16B - EXECUTE MICROPROGRAM CONTROLLER TEST
17B - LOG OFF
20B - LOG ON PRIVILEGED
21B - LOG ON UNPRIVILEGED

A P P E N D I X E

IMPLEMENTED MESSAGES (in alphabetical order)

EXPLANATION

All messages that you are authorized to be sent on the interface data bus (DIO1-DIO8), either by a device or by the controller card itself, are listed. The legal values of the messages and the message type are also shown. The coding of the handshake lines and interface management bus, which must exist to transfer these messages, are not shown.

Abbreviations used to describe message types:

- AC - Adressed command
- AD - Address
- UC - Universal command
- SE - Secondary address or command
- DD - Device-dependent
- ST - Status

All variables are integers. The meaning of the variables is:

Δx means a variable which does not affect the meaning of the message.

ΔA means a variable which specifies the legal address space.

DAB means device-dependent data.

EOS means the value of the EOS-character.

STB means device-dependent status information.

In the PPE and the PPR messages some special variables are used: ST says how the PPR message is sent; ST=0_8 → PPR is sent false. ST=10_8 → PPR is sent true. The variable B says which single data line the PPR message shall be sent on, and thereby which corresponding bit in the byte the message shall affect. The selected bit number in the byte is equal to B, where 0 is the least significant bit and 7 is the most significant bit.

IMPLEMENTED MESSAGES (IN ALPHABETICAL ORDER)

Mnemonic	Message name	Type	Legal Values (Octal)
ACG	Addressed command group	AC	0 + Δx 0≤Δx≤17
DAB	Data byte	DD	0≤DAB≤377
DCL	Device clear	UC	24
EOS	End of string	DD	0≤EOS≤377
GET	Group execute trigger	AC	10
GTL	Go to local	AC	1
LAG	Listen address group	AC	40 + Δx 0≤Δx≤37
LLO	Local lockout	UC	21
MLA	My listen address	AD	40 + ΔA 0≤ΔA≤36
MSA	My secondary address	SE	140 + ΔA 0≤ΔA≤37
MTA	My talk address	AD	100 + ΔA 0≤ΔA≤36
NUL	Null byte	DD	0
PPC	Parallel poll configure	AC	5
PPE	Parallel poll enable	SE	140 + ST + B ST {0,10} 0≤B≤7
PPD	Parallel poll disable	SE	160
PPR	Parallel poll response	ST	$ST*2^B$ 0≤B≤7
PPU	Parallel poll unconfigure	UC	25
RQS	Request service	ST	100 + Δx 0≤Δx≤77 200≤Δx≤277
SCG	Secondary command group	SE	140 + Δx 0≤Δx≤37
SDC	Selected device clear	AC	4
SPD	Serial poll disable	UC	31
SPE	Serial poll enable	UC	30
STB	Status byte	ST	STB + Δx 0≤STB≤77 200≤STB≤277 Δx {0,100}
TAG	Talk address group	AD	100 + Δx 0≤Δx≤37
UCG	Universal command group	AD	20 + Δx 0≤Δx≤17
UNL	Unlisten	AD	77
UNT	Untalk	AD	137

Index

********** SEND US YOUR COMMENTS!!! **********

Are you frustrated because of unclear information in this manual? Do you have trouble finding things? Why don't you join the Reader's Club and send us a note? You will receive a membership card — and an answer to your comments.

Please let us know if you

- find errors
- cannot understand information
- cannot find information
- find needless information

Do you think we could improve the manual by rearranging the contents? You could also tell us if you like the manual!

********** HELP YOURSELF BY HELPING US!! **********

Manual name: General Purpose Interface Bus (GPIB) User Guide

Manual number: ND-12.023.02

What problems do you have? (use extra pages if needed) _____

_____

_____

_____

_____

Do you have suggestions for improving this manual ? _____

_____

_____

_____

_____

_____

Your name: _____ Date: _____

Company: _____ Position: _____

Address: _____

_____

What are you using this manual for ? _____

_____

NOTE!
This form is primarily for documentation errors. Software and system errors should be reported on Customer System Reports.

Send to:
Norsk Data A.S
Documentation Department
P.O. Box 25, Bogerud
Oslo 6, Norway

→
Norsk Data's answer will be found on reverse side

Answer from Norsk Data: _____

Answered by _____ Date _____

Norsk Data A.S
Documentation Department
P.O. Box 25, Bogerud
Oslo 6, Norway

Systems that put people first

NORSK DATA A.S OLAF HELSETS VEI 5 P.O. BOX 25 BOGERUD 0621 OSLO 6 NORWAY
TEL.: 02 - 29 54 00 - TELEX: 18284 NDN