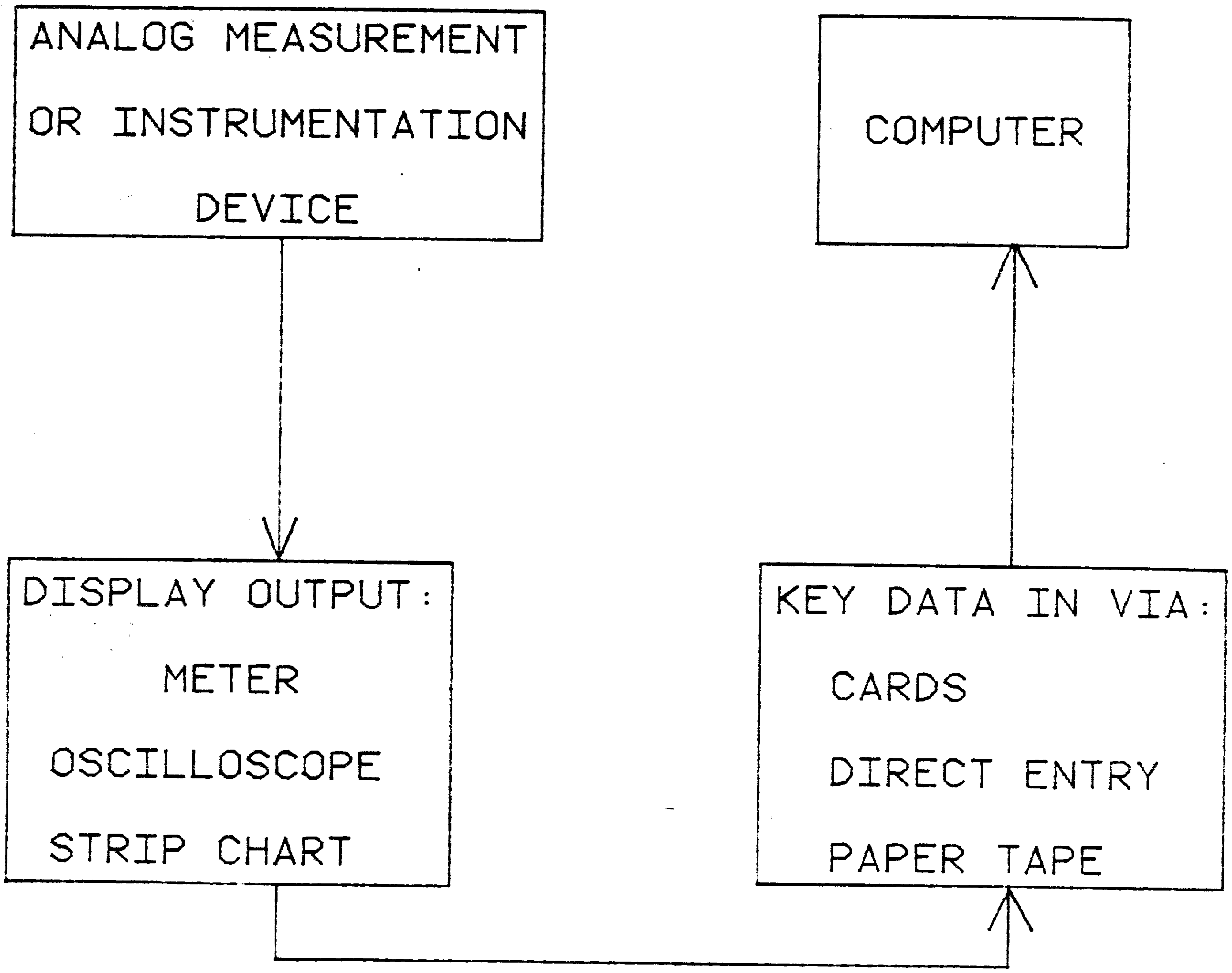
GPIB SEMINAR

AUTOMATING TEST & MEASUREMENT PROCEDURES IN THE 80'S

GPIB

AUTOMATING TEST & MEASUREMENT PROCEDURES IN THE 80'S

DATA CAPTURE BEFORE GPIB



THE NEED FOR A GPIB INTERFACE

* PROLIFERATION OF SMART & FRIENDLY MEASUREMENT/INSTRUMENTATION DEVICES BY DIFFERENT MANUFACTURERS WITH UNIQUE INTERFACING REQUIREMENTS.

* MINIMIZE TIME/EFFORT NEEDED TO INTERCONNECT DEVICES INTO A SYSTEM.

* DESIRE TO CONSTRUCT SYSTEMS WITH A MULTITUDE OF MEASUREMENT/INSTRUMENTATION DEVICES.

CUSTOM INTERFACE PROBLEMS

1. Learn interface of computer and each instrument.

2. Design, build, checkout interface for each instrument.

3. Document each interface.

4. Train maintenance personnel and provide diagnostic tools for each interface.

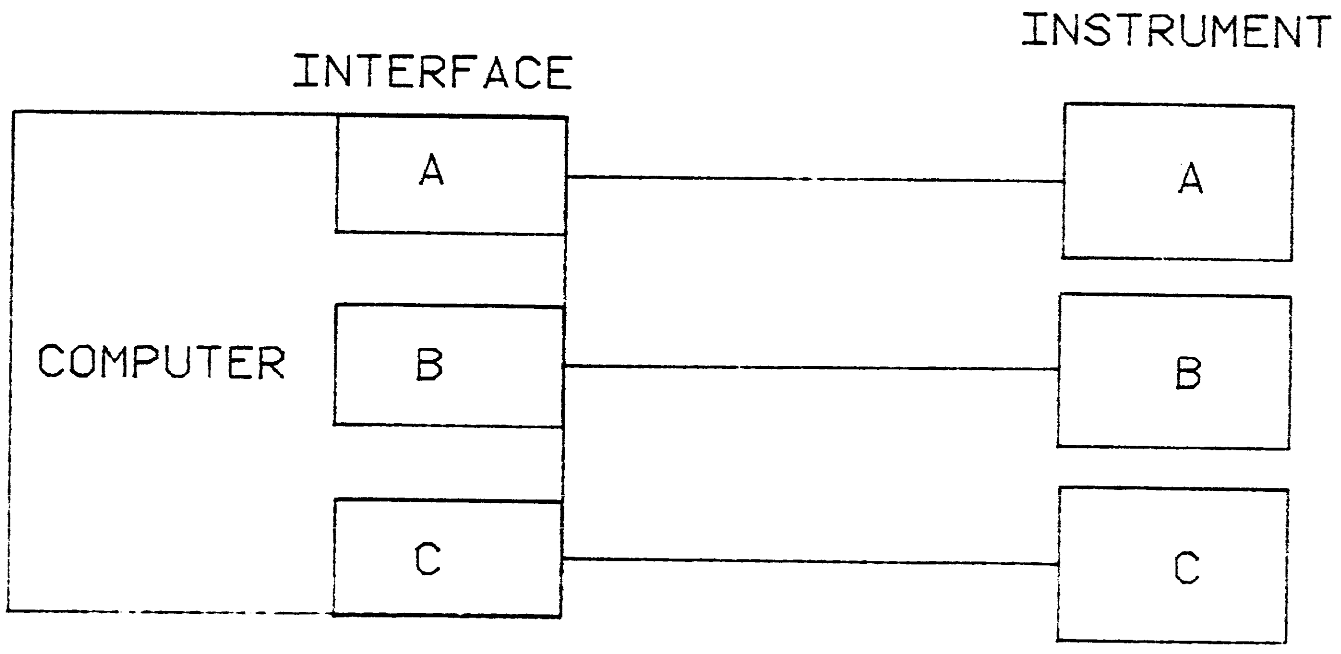
5. If a new instrument is added to system, repeat all of above.

PROBLEMS
(without GPIB)

1. TOO MANY LINES

2. EACH INSTRUMENT'S INTERFACE IS DIFFERENT

3. POINT-TO-POINT CONNECTION



SOLUTIONS
(with GPIB)

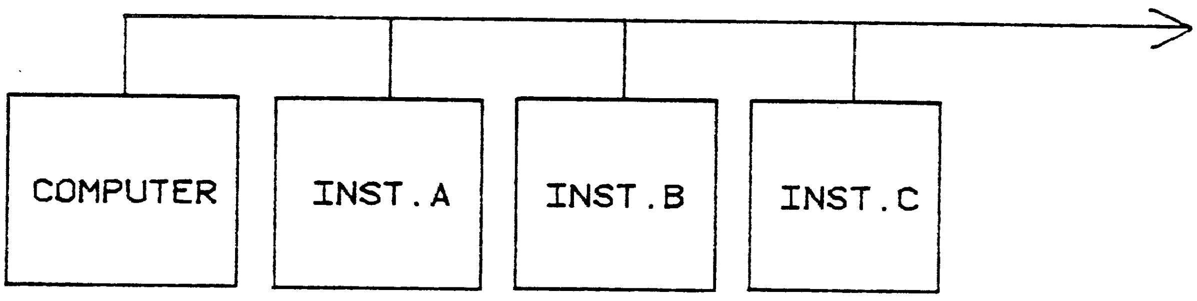
1. ALL DATA IS CARRIED ON 8 DATA LINES:

   DATA TO DEVICES, DATA FROM DEVICES
   PROGRAMMING, COMMANDS, ETC.
   BUS IS BYTE SERIAL, BI-DIRECTIONAL

2. ALL DEVICE INTERFACES ARE THE SAME:

   A. MECHANICAL
   B. ELECTRICAL ---> defined by IEEE-488 std.
   C. FUNCTIONAL

3. MANY DEVICES (14) ON THE SAME BUS:



GPIB

- o Started as the ASCII bus at HP and was later called HP-IB

- o IEC adapted the principal (IEC 625) and initiated a recommendation to IEEE.

- o IEEE established a standard: Standard Digital Interface for Programmable Instrumentation. (IEEE Standard 488-1975)

- o Updated in 1978 to IEEE-488-1978

- o Now commonly known as GPIB.

- o New recommendations December 1981: IEEE 728 "Codes & Formats Conventions"

GPIB DRIVING FORCES

* SHORTAGE OF SKILLED LABOR

* NEED FOR GREATER THROUGHPUT

* COMPLEX TESTS

* DOCUMENTATION

* COST REDUCTIONS

* TEST AND MEASUREMENT CONSISTENCY

* UNATTENDED MONITORING AND DATA ACQUISITION

* COMPUTATION AND ANALYSIS

* PRODUCTIVITY

PHASE I

CUSTOM SYSTEMS

* CUSTOM INSTRUMENTS
* SOFTWARE
* TURN-KEY
* EXPENSIVE

PHASE II

INTEGRATION

* OFF-THE-SHELF INSTRUMENTS
* CUSTOM HARDWARE INTERFACES
* SOFTWARE DRIVERS
* EXPENSIVE

PHASE III

IEEE-488

* STANDARD INSTRUMENTS AND CONTROLLERS
* SYSTEM INTEGRATION
* SOFTWARE DRIVERS SPECIFIC TO INSTRUMENTS
* MODERATE EXPENSE

PHASE IV

SOFTWARE STANDARD

* IEEE-488
* STANDARD CODES AND FORMATS IEEE-728
* MNEMONICS, HIGH-LEVEL COMMANDS
* LEARN MODE
* MODERATE HARDWARE, LESS SOFTWARE

COMPUTER SKILLS

* FEW COMPUTER SPECIALISTS IN TEST EQUIPMENT

* ELECTRONICS WITH COMPUTER INTEREST

* HIGH KNOWLEDGE ON TEST AND MEASUREMENT TECHNIQUES

* KNOW ENOUGH SOFTWARE TO DO THE JOB

SYSTEM USE TODAY

* PRODUCTION TEST AND LABS
  - HIGH THROUGHPUT
  - SPACE
  - AFFORD SOFTWARE DEVELOPMENT

FUTURE NEEDS

* WHERE DRIVING FORCE EXISTS
  - IF EQUIPMENT SMALL
  - IF EQUIPMENT INEXPENSIVE
  - IF SOFTWARE EASIER
  - FIELD, VANS, SHIPS, PLANES, ETC.

ECONOMIC CONCEPTS

* HARDWARE COSTS:

MANUAL	-	$ 6K
PROGRAMMABLE	-	$24K

* TIME ADVANTAGE:

MANUAL/PROGRAM. = 2/1

THUS NEED TWO MANUAL SYSTEMS FOR SAME THROUGHPUT.

* TOTAL EQUIVALENT HARDWARE COSTS:

MANUAL	-	$12K
PROGRAMMABLE	-	$24K

* LABOR COSTS:

MANUAL	-	\$30K X 2	=	\$60K/YR
PROGRAMMABLE	-	\$25K	=	\$25K/YR

* SOFTWARE COSTS:

MANUAL	-	\$0
PROGRAMMABLE	-	\$25K

* CASH FLOW:

MANUAL - PROGRAM. = \$12K - \$49K = -\$37K
(INITIAL INVESTMENT)

MANUAL - PROGRAM. = \$60K - \$25K = +\$35K
(SUBSEQUENT YEARS)

PAY BACK

* THE AMOUNT OF TIME TO PAY BACK THE INITIAL COST

APPROX. 1 YEAR

RATE OF RETURN

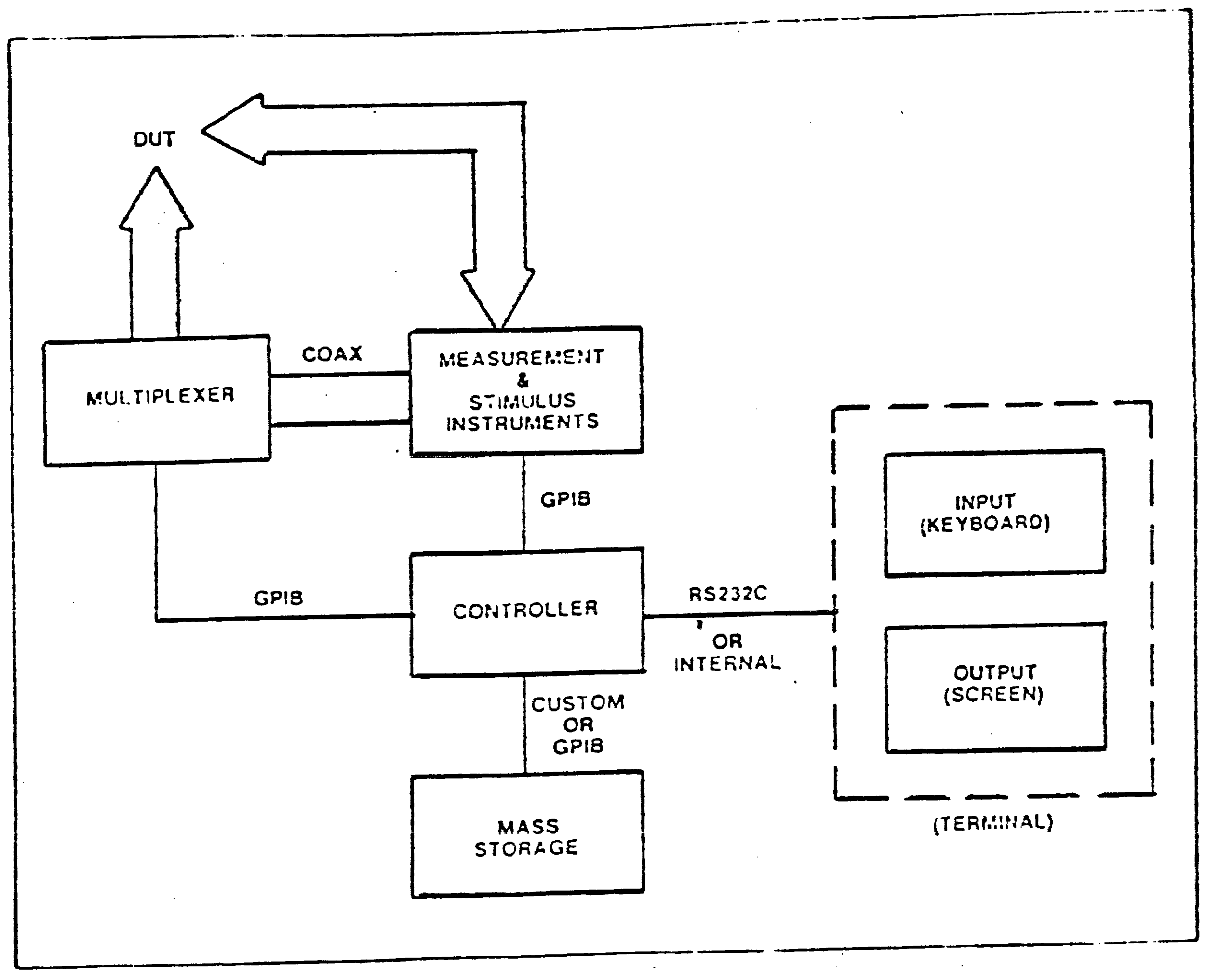
* HOW MUCH INTEREST WOULD YOU HAVE TO EARN TO GET $35K PER YEAR ON AN INITIAL INVESTMENT OF $37K?

= 91% INTEREST RATE

GPIB

MEASUREMENT SYSTEM COMPONENTS

- CONTROLLER
- KEYBOARD
- DISPLAY
- MASS MEDIA
- INSTRUMENTS


DUT
MULTIPLEXER
COAX
MEASUREMENT & STIMULUS INSTRUMENTS
GPIB
GPIB
CONTROLLER
RS232C OR INTERNAL
CUSTOM OR GPIB
MASS STORAGE
INPUT (KEYBOARD)
OUTPUT (SCREEN)
(TERMINAL)

DESIGN OBJECTIVES FOR GPIB

- DEFINE A GENERAL PURPOSE SYSTEM FOR USE IN LIMITED DISTANCE APPLICATIONS.
- SPECIFY THE DEVICE INDEPENDENT MECHANICAL, ELECTRICAL, AND FUNCTIONAL INTERFACE REQUIREMENTS.
- ENABLE THE INTERCONNECTION OF INDEPENDENTLY MANUFACTURED APPARATUS INTO A SINGLE FUNCTIONAL SYSTEM.
- PERMIT APPARATUS WITH A WIDE RANGE OF CAPABILITIES TO BE INTERCONNECTED TO THE SYSTEM SIMULTANEOUSLY.
- PERMIT DIRECT COMMUNICATION BETWEEN DEVICES WITHOUT REQUIRING ALL MESSAGES TO BE ROUTED THROUGH A CONTROL UNIT
- DEFINE A SYSTEM WITH A MINIMUM OF RESTRICTIONS ON THE PERFORMANCE CHARACTERISTICS OF THE DEVICES.
- PERMIT ASYNCHRONOUS COMMUNICATIONS OVER A WIDE RANGE OF DATA RATES.
- DEFINE A SYSTEM THAT IS RELATIVELY LOW COST AND PERMITS THE INTERCONNECTION OF LOW COST DEVICES.
- DEFINE A SYSTEM THAT WORKS AND IS EASY TO OPERATE.

GPIB

IEEE STANDARD 488 LIMITATIONS

- DATA EXCHANGED BETWEEN DEVICES MUST BE DIGITAL
- DATA RATES UP TO ONE MEGABYTE ARE SUPPORTED
- DATA RATE LIMITED TO SLOWEST DEVICE IN SYSTEM
- MAXIMUM TOTAL CABLE LENGTH IS 20 METERS
- MAXIMUM OF 15 DEVICES IN A SYSTEM
- DOES NOT GUARANTEE DEVICES WILL COMMUNICATE
- GPIB DOES NOT GUARANTEE PROGRAMMABILITY OF DEVICES
- PROGRAMMABILITY DOES NOT GUARANTEE GPIB CAPABILITY

GPIB

MECHANICAL SPECIFICATIONS

- CONNECTOR TYPE
- CONNECTOR CONTACT ASSIGNMENTS
- DEVICE CONNECTOR MOUNTING
- CABLE ASSEMBLY
- LINEAR/STAR CONFIGURATION

GPIB

ELECTRICAL SPECIFICATIONS

- DATA RATE
- DRIVER REQUIREMENTS
- RECEIVER REQUIREMENTS
- COMPOSITE LOAD REQUIREMENTS
- CABLE CHARACTERISTICS
- GROUND REQUIREMENTS

GPIB

FUNCTIONAL SPECIFICATIONS

- TEN INTERFACE FUNCTIONS DEFINE USE OF SIGNAL LINES.
- SPECIFIC PROTOCOL BY WHICH INTERFACE FUNCTIONS OPERATE.
- LOGICAL AND TIMING RELATIONSHIPS BETWEEN STATES SPECIFIED.

GPIB

GPIB INTERFACE FUNCTIONS

SYMBOL	FUNCTION
SH	-- Source Handshake
AH	-- Acceptor Handshake
T	-- Talker/Extended Talker
L	-- Listener/Extended Listener
SR	-- Service Request
RL	-- Remote/Local
PP	-- Parallel Poll
DC	-- Device Clear
DT	-- Device Trigger
C	-- Controller

GPIB TALKER

- Device capable of transmitting data over the bus.
- Only one talker at a time.

GPIB

GPIB LISTENER

- Device capable of receiving data over the bus.

- Up to 14 listeners may take part in transfer of data.

GPIB

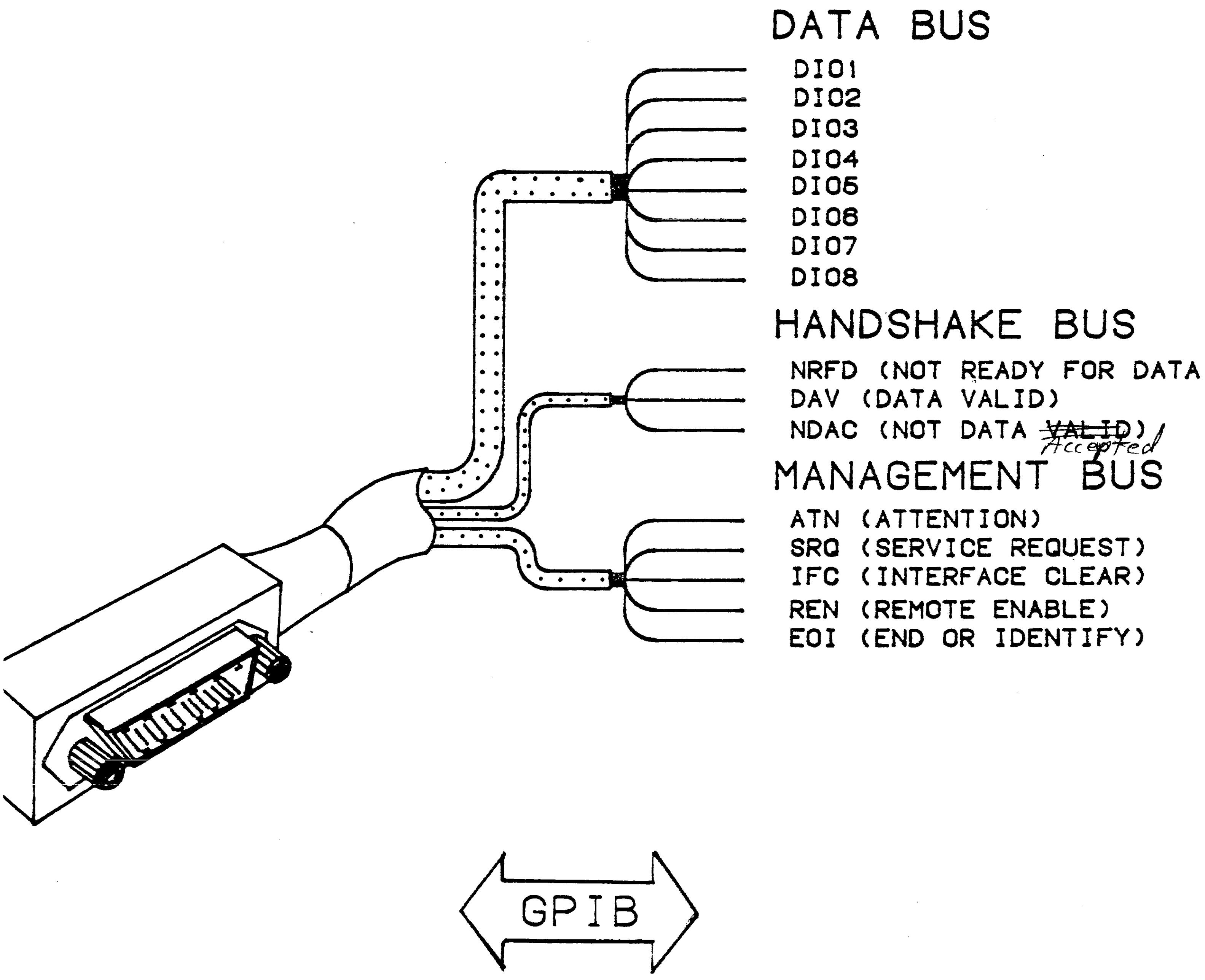
GPIB CONTROLLER

- Assigns other devices as talkers, listeners, or idle devices.
- Responds to Request for Service (SRQ) by other devices.
- Can assign themselves as talker or listener whenever necessary.
- Only one SYSTEM CONTROLLER per system.
- Only one CONTROLLER-IN-CHARGE active at any one time per system.

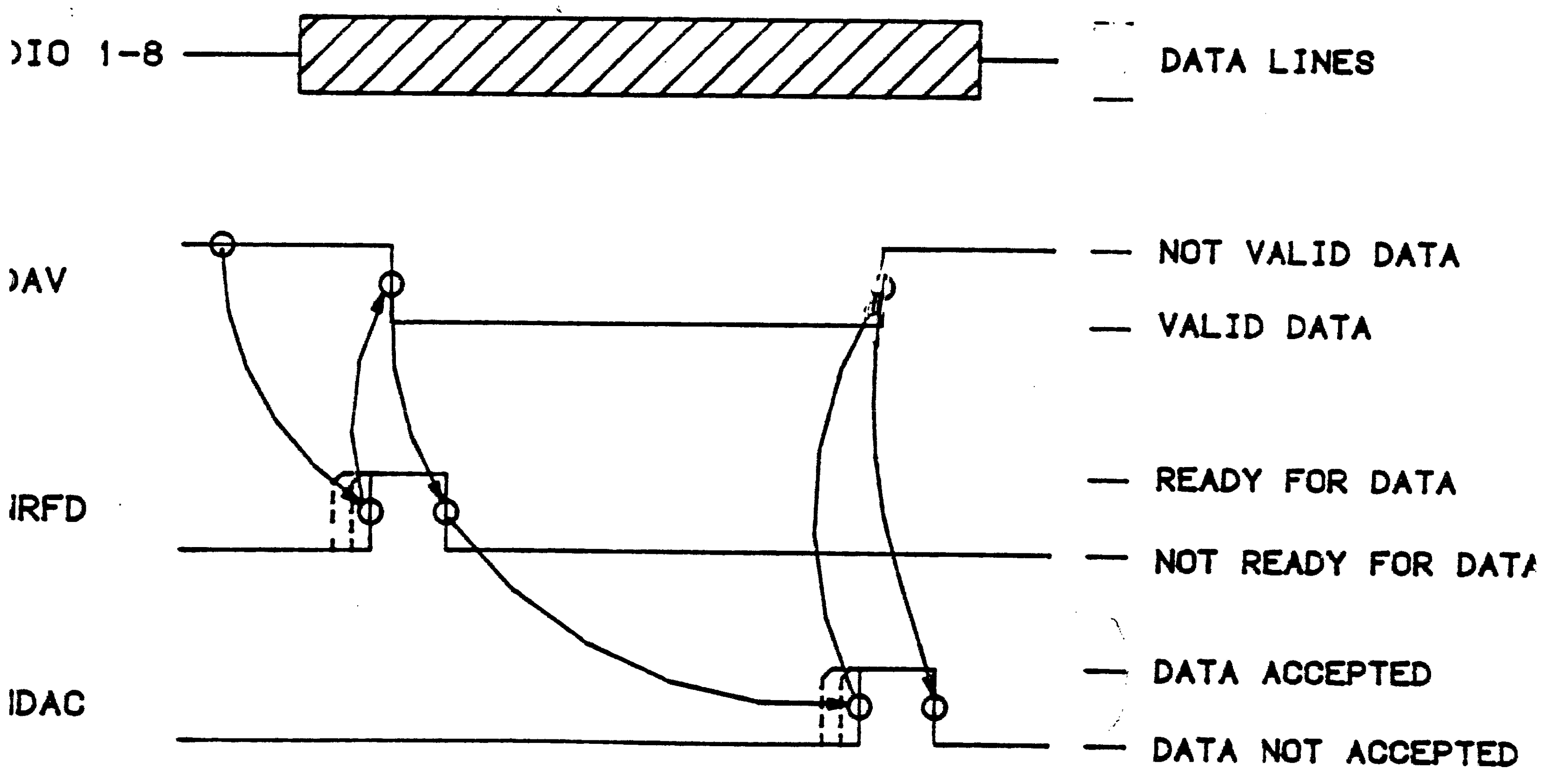
GPIB

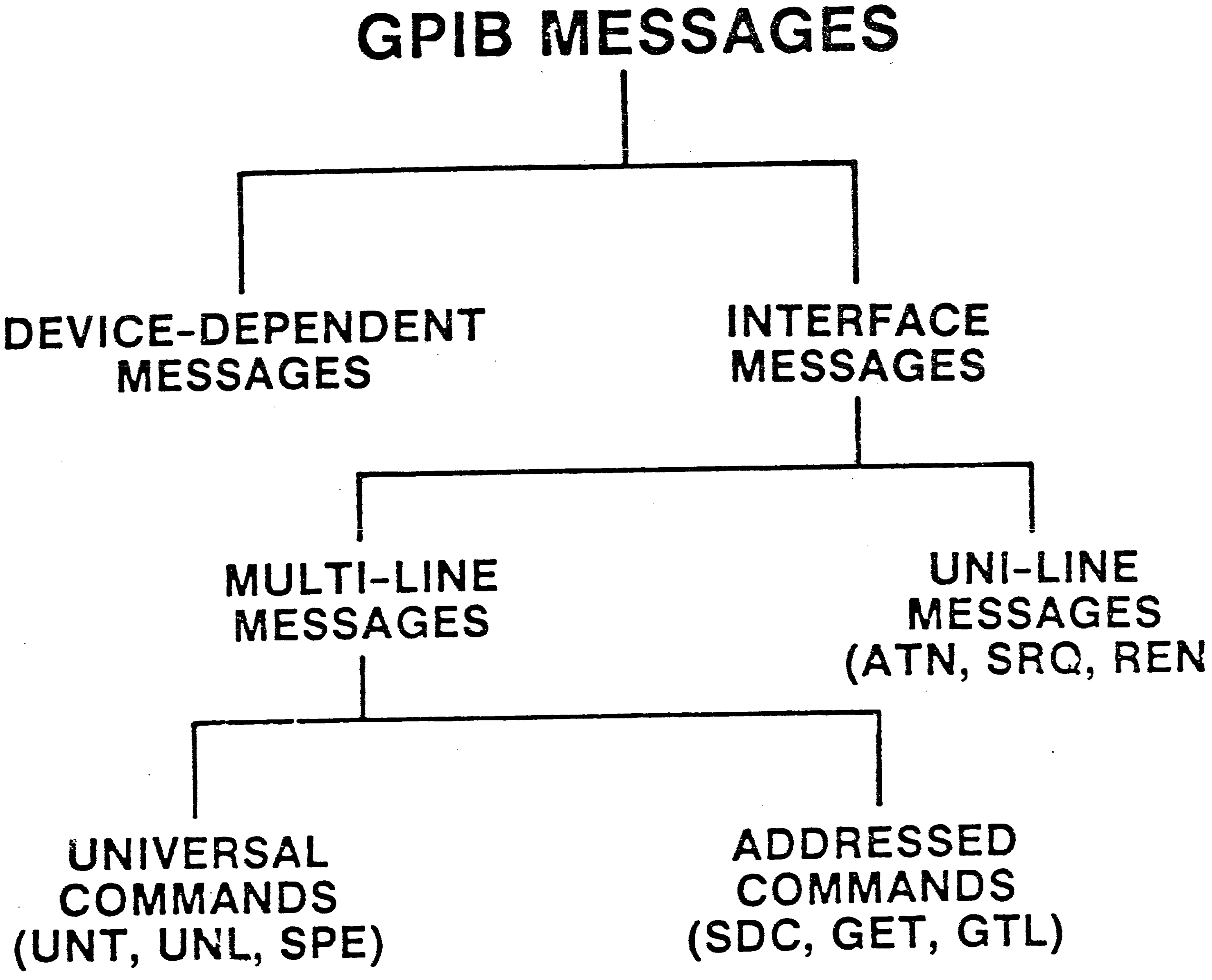
GPIB IDLE DEVICE

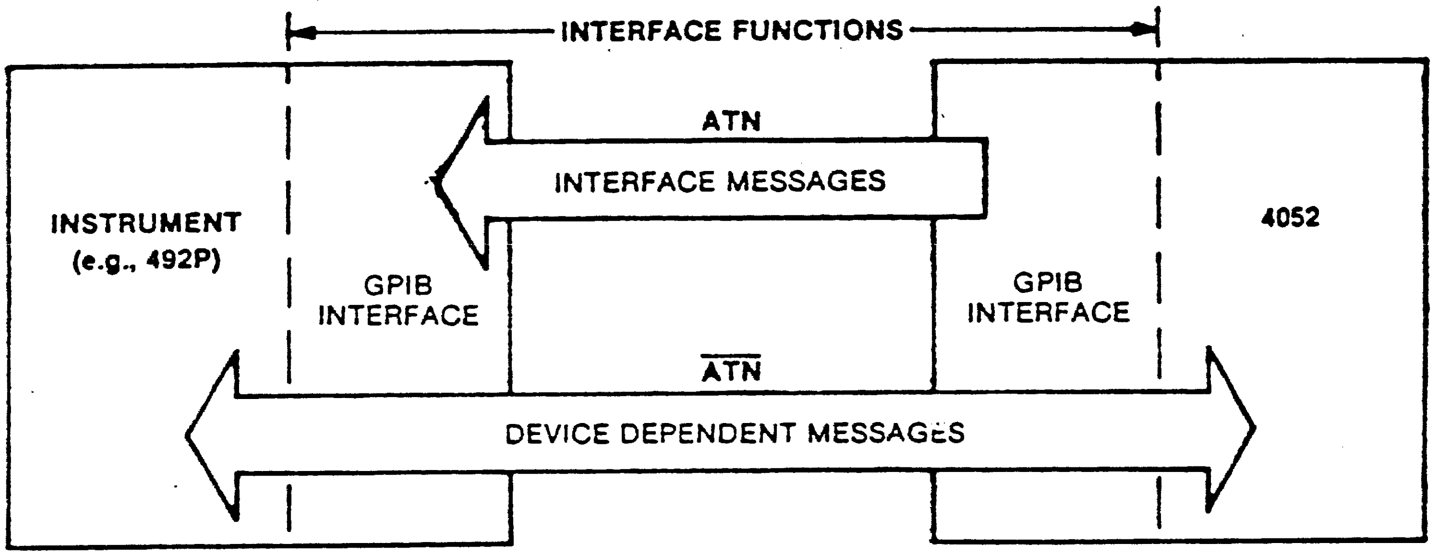
- A device not addressed as a talker or listener.
- Does not slow down data transfer, yet remains available to controller.

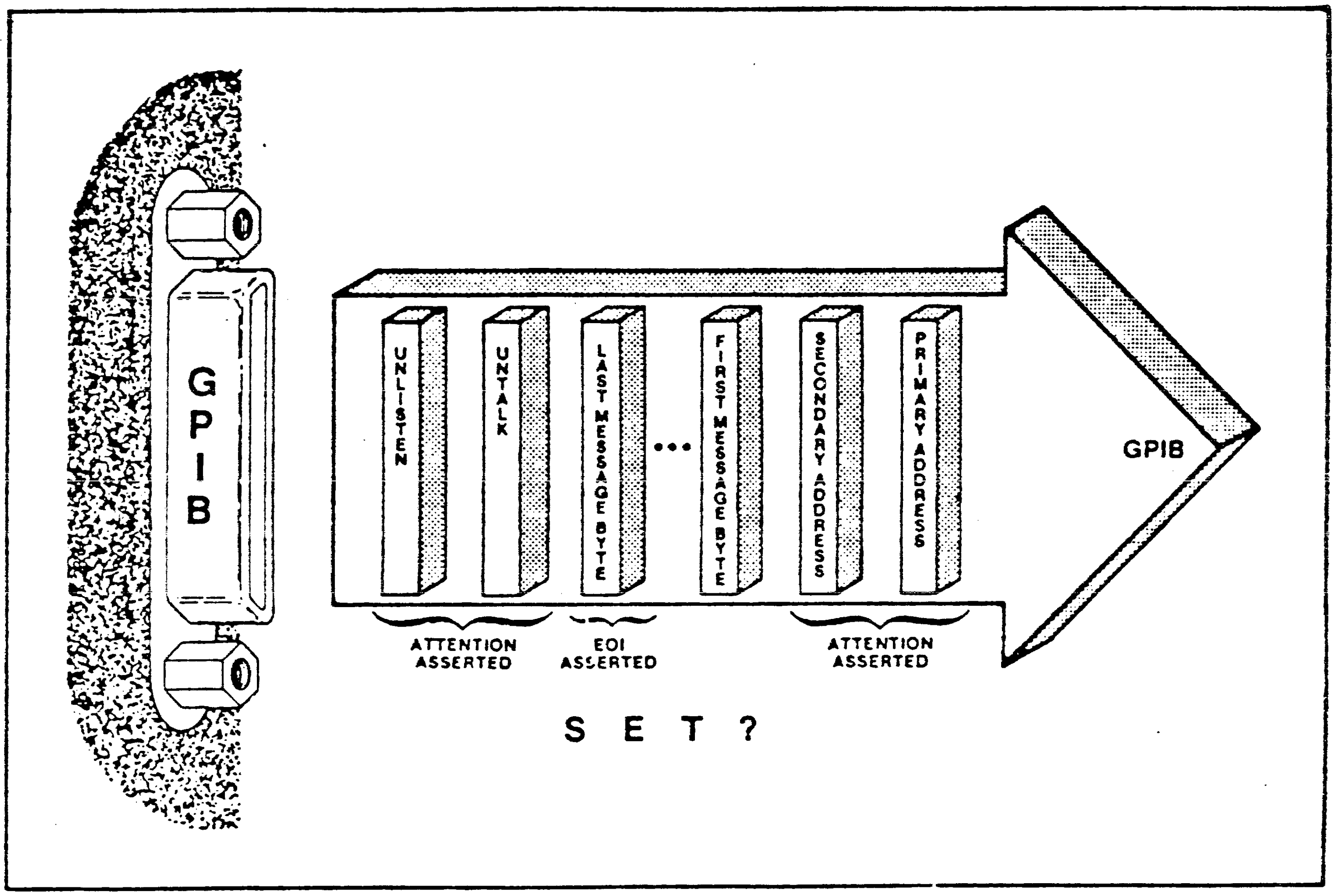

DATA BUS
DIO1
DIO2
DIO3
DIO4
DIO5
DIO6
DIO7
DIO8
HANDSHAKE BUS
NRFD (NOT READY FOR DATA
DAV (DATA VALID)
NDAC (NOT DATA ~~VALID~~ Accepted)
MANAGEMENT BUS
ATN (ATTENTION)
SRQ (SERVICE REQUEST)
IFC (INTERFACE CLEAR)
REN (REMOTE ENABLE)
EOI (END OR IDENTIFY)
GPIB

THE GPIB HANDSHAKE




GPIB MESSAGES
DEVICE-DEPENDENT MESSAGES
INTERFACE MESSAGES
MULTI-LINE MESSAGES
UNI-LINE MESSAGES (ATN, SRQ, REN
UNIVERSAL COMMANDS (UNT, UNL, SPE)
ADDRESSED COMMANDS (SDC, GET, GTL)


INTERFACE FUNCTIONS
ATN
INTERFACE MESSAGES
INSTRUMENT
(e.g., 492P)
GPIB
INTERFACE
4052
GPIB
INTERFACE
$\overline{\text{ATN}}$
DEVICE DEPENDENT MESSAGES


G P I B
UNLISTEN
UNTALK
LAST MESSAGE BYTE
• • •
FIRST MESSAGE BYTE
SECONDARY ADDRESS
PRIMARY ADDRESS
GPIB
ATTENTION ASSERTED
EOI ASSERTED
ATTENTION ASSERTED
S E T ?

GPIB DEVICE DEPENDENT MESSAGES

- MULTI-LINE (DATA BUS SIGNAL, ONE OR MORE BYTES)
- NO RESTRICTIONS IMPOSED BY IEEE 488 STANDARD
- SENT AND RECEIVED ONLY WHEN ATN NOT ASSERTED

GPIB

A TYPICAL BUS CONVERSATION

Using 4050 BASIC, tell the 7854 digitizing oscilloscope to acquire a waveform ten times and display the averaged results:

```
                                    $HEX                    (ADE)
PRINT @1:"AVG10"
                       ATN UNL $3F               REN     (063)
                       ATN LAG  01               REN     (033)
                       ATN SCG  12               REN     (108)
                             A $41               REN     (065)
                             V $56               REN     (086)
                             G $47               REN     (071)
                             1 $31               REN     (049)
                             0 $30               REN     (048)
                            CR $0D EOI           REN    -(013)
                       ATN UNT $5F               REN     (095)
                       ATN UNL $3F               REN     (063)

                             7D01F2 DISPLAY
```

Note the advantages of higher level languages and mnemonic device dependent messages.

Some GPIB mnemonics

PCG primary command group

ACG-addressed commands group

GTL	go to local
SDC	selected device clear
PPC	parallel poll configure
GET	group exececute trigger
TCT	take control

UCG-unaddressed commands group

LLO	local lockout
DCL	device clear
PPU	parallel poll unconfigure
SPE	serial poll enable
SPD	serial poll disable

LAG-listen addressed group

UNL	unlisten

TAG-talk addressed group

UNT	untalk

SCG secondary command group

GPIB

MULTI-LINE INTERFACE MESSAGES

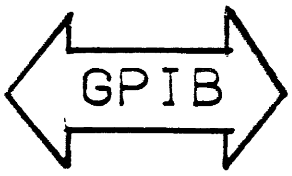
ISO-7 Bit Code Representation

DIO	ASCII	MSG	ASCII	MSG	ASCII	MSG	ASCII	MSG	ASCII	MSG	ASCII	MSG	ASCII	MSG	ASCII	MSG
765 / 4321	ØØØ		ØØ1		Ø1Ø		Ø11		1ØØ		1Ø1		11Ø		111	
ØØØØ	NUL		DLE		SP	Ø	Ø	16	@	Ø	P	16	`	Ø	p	16
ØØØ1	SOH	GTL	DC1	LLO	!		1		A		Q		a		q	
ØØ1Ø	STX		DC2		"		2		B		R		b		r	
ØØ11	ETX		DC3		#		3		C		S		c		s	
Ø1ØØ	EOT	SDC	DC4	DCL	$		4		D		T		d		t	
Ø1Ø1	ENQ	PPC	NAK	PPU	%		5		E		U		e		u	
Ø11Ø	ACK		SYN		&		6		F		V		f		v	
Ø111	BEL		ETB		'		7		G		W		g		w	
1ØØØ	BS	GET	CAN	SPE	(		8		H		X		h		x	
1ØØ1	HT	TCT	EM	SPD	)		9		I		Y		i		y	
1Ø1Ø	LF		SUB		*		:		J		Z		j		z	
1Ø11	VT		ESC		+		;		K		[		k		{	
11ØØ	FF		FS		,		<		L		\		l		|	
11Ø1	CR		GS		-		=	↓	M		]	↓	m		}	
111Ø	SO		RS		.	↓	>	3Ø	N	↓	^	3Ø	n	↓	~	↓
1111	SI		US		/	15	?	UNL	O	15	_	UNT	o	15	DEL	31

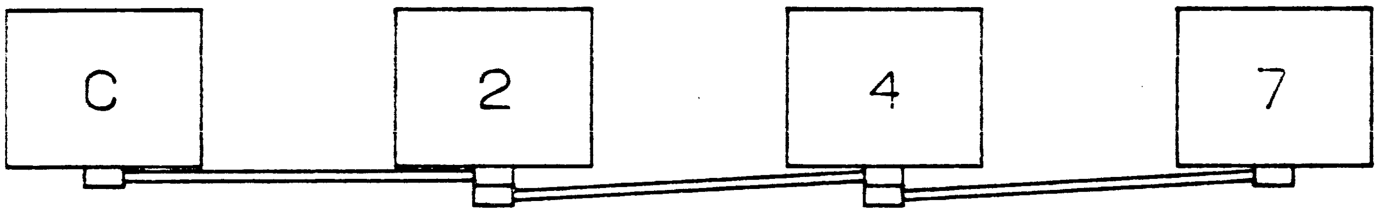
ACG UCG LAG TAG

PCG SCG

SENT AND RECEIVED WITH ATN ASSERTED



SERIAL POLL ACTIVITY FOR MULTIPLE DEVICE SYSTEMS



SRQ becomes asserted. . .which device needs service?

```
250 ON SRQ THEN 800
 .     .
 .     .
 .     .
800 POLL P,S;4;2;7              ATN UNL
 .     .                        ATN SPE
 .     .                        ATN TAG  04
 .     .                              0
990 RETURN                      ATN TAG  02
                                      0
                                ATN TAG  07
                                      65
                                ATN UNT
                                ATN SPD
```

P = POSITION IN LIST OF RESPONDING DEVICE
S = STATUS OF RESPONDING DEVICE

PARALLEL POLL

- UP TO 8 INSTRUMENTS CAN BE CONFIGURED TO RESPOND SIMUTANEOUSLY ON THE DATA BUS
- NO STATUS INFORMATION PROVIDED BY PARALLEL POLL; RESPONDING DEVICES MUST BE RE-ADDRESSED FOR STATUS INFORMATION
- DEVICES PARTICIPATING IN PARALLEL POLL ACTIVITY MUST BE PRECONFIGURED

Sample command:

```
700 PPOLL S;4,1,1;2,2,1;7,3,0
```

Result:

```
Device 4 sets data line 1 to 1 if asserting SRQ
Device 2 sets data line 2 to 1 if asserting SRQ
Device 7 sets data line 3 to 0 if asserting SRQ
```

The resultant byte is stored in variable S

INSTRUMENTATION CONCEPTS

"What do I consider before choosing a GPIB instrument?"

* BASIC FUNCTIONAL CAPABILITIES

"How well does this instrument do its job?"

* INSTRUMENT CAPABILITIES ON GPIB

IMPLEMENTATION OF IEEE-488 INTERFACE FUNCTIONS

* (T)talker,(L)listener
  (SR)service request,etc.

COMMAND SETS

* ENGLISH-LIKE AND FRIENDLY?

* CONSISTENT THOUGHOUT PRODUCT LINE

Example

```
FREQ A      VPOS 10     ID?
PER A       DCV 2       SET?
C1F1E6A2.1
```

QUERY RESPONSE

"How much can this instrument tell you about its present status?"

* NORMAL STATUS
* ABNORMAL STATUS
* COMPLETED OPERATION STATUS
* DETAILED ERROR CONDITIONS

Examples:

COMMAND	INSTRUMENT RESPONSE
"ID?"	Transmits inst. model no.,GPIB address
"SET?"	Transmits all programmable front panel settings,operational parameters
"ERR?"	Transmits specific error status of instrument

LEARN MODE

- Ability of a GPIB instrument to transfer all programmable setting to a controller or other listener

Example:

"SET?" Command returns all front panel settings

PROGRAMMABLE/MANUAL OPERATION

* AVAILABILITY OF ALL INSTRUMENT FUNCTIONS AT FRONT PANEL

* MANUAL INTERVENTION UNDER PROGRAM CONTROL

PENDING SETTINGS BUFFER

* MULTIPLE COMMANDS BUFFERED WITHIN INSTRUMENT

* COMMANDS PRIORITIZED BEFORE EXECUTION

Example:

```
        When sent "LSOUT;VLOG 5.0;ILOG 1.0;"
        "LSOUT" executed last
```

SELF TEST

* Instrument Performs Self-Diagnostic Routine And Returns Appropriate Status

GPIB CONTROLLER CONSIDERATIONS

- PACKAGING
- PERIPHERALS
- I/O CAPABILITIES
- LANGUAGE
- OPERATING SYSTEM
- GRAPHICS
- MEMORY
- PRICE

PACKAGING

- OVERALL SIZE
- RACK MOUNTABLE/DESKTOP ONLY

I/O CAPABILITIES

- GPIB PORT(S) STANDARD OR OPTIONAL
- RS-232 PORT(S) STANDARD OR OPTIONAL
- TTL PORT AVAILABLE
- DMA (DIRECT MEMORY ACCESS)
- I/O SPEED IN ALL MODES

PERIPHERALS INTEGRAL WITH CONTROLLER

- DISPLAY
- KEYBOARD
- PRINTER
- MASS STORAGE; TAPE, DISC, ETC.

LANGUAGE

- OPERATING SYSTEM IN ROM OR SOFT LOAD
- INTERPRETIVE OR COMPILED LANGUAGE
- BASIC, PASCAL, FORTRAN, ETC.
- GPIB LANGUAGE-EASE OF USE AND FLEXIBILITY

GRAPHICS

- GRAPHIC SOFTWARE CAPABILITY
- RESOLUTION
- DISPLAY TYPE-RASTER OR STORAGE

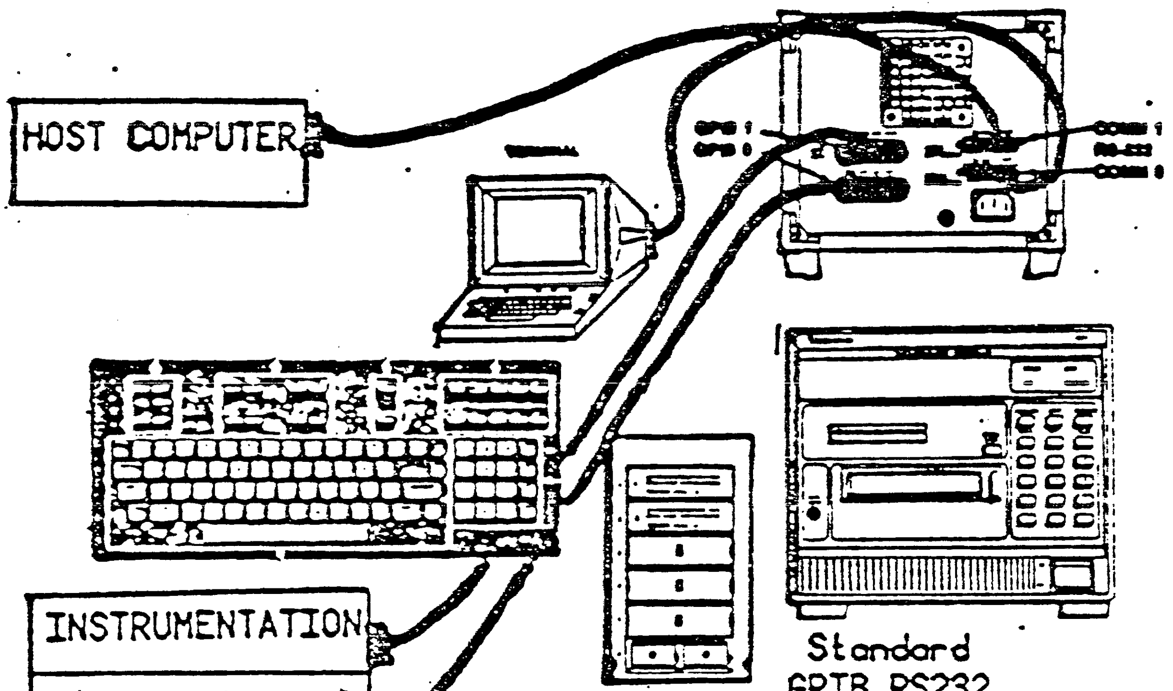
MEMORY

- MEMORY SIZE IN BASE-PRICED UNIT
- USER AVAILABLE MEMORY
- EXPANDABLE TO WHAT SIZE
- MASS STORAGE CAPABILITIES

PRICE

** PRICE MUST BE COMPARED ON **
THE BASIS OF COMPARABLY EQUIPPED
UNITS

4041 INSTRUMENT CONTROLLER
Application Profile



INSTRUMENTATION
(Compatible with)
Mfg "Floor" System

4041 Peripherals IBM RS232 Terminal

Standard
GPIB RS232
32K Memory $_____
64K(Opt.20)$_____
96K(Opt.21)$_____
128K(Opt.22)$_____
160K(Opt.23)$_____
Opt.1 (DMA GPIB+RS232$_____
Opt.2 (TTL) $_____
Opt.30(Prog)$_____
Opt.31(K/B) $_____

GPIB SYSTEM SOFTWARE

1. SOFTWARE COMPONENTS
   * CONTROLLER LANGUAGE
   * APPLICATION SOFTWARE
   * DEVICE-DEPENDENT CODE

2. SOFTWARE STANDARDS
   * CODES & FORMATS
   * HUMAN INTERFACE
   * COMPATIBILITY
   * CONVENTIONS
   * STATUS BYTES
   * QUERIES

ROLE OF SOFTWARE

* INTEGRATES SYSTEM COMPONENTS

* ALLOWS USER CONTROL

* ESTABLISHES SEQUENCE OF ACTIVITY

**SOFTWARE
MAKES THE
SYSTEM WORK**

USER KNOWLEDGE

* KNOW WHAT TASKS THE SYSTEM IS TO PERFORM

* KNOW THE COMPUTER'S LANGUAGE

* KNOW THE KIND OF DATA OR LANGUAGE THE INSTRUMENT REQUIRES

SOFTWARE COMPONENTS

1. CONTROLLER LANGUAGE
   - OPERATING SYSTEM SOFTWARE
   - I/O DRIVERS
   - MATH PACKAGES
   - HI LEVEL PROGRAMS

2. APPLICATION SOFTWARE
   - PERFORMS SPECIFIC TASKS
   - MEASUREMENTS
   - DOCUMENTATION

3. DEVICE-DEPENDENT CODE
   - COMMAND SETS FOR EACH
     GPIB INSTRUMENT

BASIC

* ADVANTAGES

IMMEDIATE MODE
EASY TO LEARN
ACCEPTANCE
HIGHLY INTERACTIVE
EASY TO MODIFY PROGRAMS
STANDARD LANGUAGE

* DISADVANTAGES

NO ABILITY TO REMEMBER MACHINE CODE
RETRANSLATE EACH LINE

COMPILER LANGUAGES

FORTRAN, PASCAL

* ADVANTAGES

TRANSLATE ENTIRE PROGRAM INTO
MACHINE CODE
REDUCE MEMORY REQUIREMENTS
SPEED EFFICIENCY
STANDARD LANGUAGE

* DISADVANTAGES

MORE DIFFICULT LANGUAGE
LEARN EDITORS
MORE TIME CONSUMING PROGRAM EDITING
MORE COSTLY TO IMPLEMENT

ASSEMBLY LANGUAGE

* COMPUTER OR MICROPROCESSOR CODE

* MNEMONICS

MOVE B,A

* ADVANTAGES

SPEED
MEMORY UTILIZATION

*DISADVANTAGES

KNOWLEDGE OF PROCESSOR ARCHITECTURE
TIME-CONSUMING TO WRITE
DIFFICULT TO MODIFY
DETAILED DOCUMENTATION
LITTLE INTERACTION

APPLICATION SOFTWARE

- REAL INVESTMENT

- REQUIREMENTS:

  INTERACTIONS

  MEASUREMENTS

  PASS/FAIL BOUNDARIES

  COMPUTATIONAL REQUIREMENTS

  STATISTICAL DATA

  DOCUMENTATION

  DATA STORAGE

DEVICE-DEPENDENT CODE

* COMMUNICATED BETWEEN A TALKER AND LISTENER DEVICES

* COMPOSED OF ONE OR MORE MESSAGES SEPARATED BY DELIMITERS

```
PRINT @4: "VPOS 15;IPOS .5"
```

* SEND AND RECEIVE ONLY WHEN ATN NOT ASSERTED

* NOT AN INTERFACE MESSAGE

* NOT BASIC

* INSTRUMENT DEPENDENT

COMMANDS

The instrument is controlled by the front panel or via commands received from the controller. These commands are of three types:

Setting commands — control instrument settings
Query-output commands — ask for data
Operational commands — cause a particular action

The instrument responds to and executes all commands when in the remote state. When in the local state, *setting* and *operational commands* generate errors since instrument function are under front panel control; only *query-output commands* are executed.

Each command begins with a header — a word that describes the function implemented. Many commands require an argument following the header — a word or number which specifies the desired state for the function.

FUNCTIONAL COMMAND LIST

Instrument Commands

ILOGIC—Sets logic supply current limit.
ILOGIC?—Returns logic supply current limit.
VLOGIC—Sets logic supply voltage limit.
VLOGIC?—Returns logic supply voltage limit.
INEGATIVE—Sets negative supply current limit.
INEGATIVE?—Returns negative current limit.
VNEGATIVE—Sets negative voltage limit.
VNEGATIVE?—Returns negative voltage limit.
IPOSITIVE—Sets positive current limit.
IPOSITIVE?—Returns positive current limit.
VPOSITIVE—Sets positive voltage limit.
VPOSITIVE?—Returns positive voltage limit.
ITRACK—Sets positive and negative current limits.
VTRACK—Sets positive and negative voltage limits.

Input/Output Commands

OUT ON—Connects supplies to output terminals.
OUT OFF—Disconnects supplies from output terminals.
OUTPUT?—Returns FSOUT and LSOUT ON or OFF.
FSOUTPUT ON—Connects the floating supplies to the output terminals.
FSOUTPUT OFF—Disconnects the floating supplies from the output terminals.
FSOUTPUT?—Returns FSOUT ON or OFF.
LSOUTPUT ON—Connects the logic supply to the output terminals.
LSOUTPUT OFF—Disconnects the logic supply from the output terminals.
LSOUTPUT?—Returns LSOUT ON or OFF.

Instrument Status Commands

REGULATION?—Returns regulation status of all supplies.
LRI ON—Enables logic supply regulation interrupt.
LRI OFF—Disables logic supply regulation interrupt.
LRI?—Returns LRI ON or LRI OFF.
NRI ON—Enables negative supply regulation interrupt.
NRI OFF—Disables negative supply regulation interrupt.
NRI?—Returns NRI ON or NRI OFF.
PRI ON—Enables positive supply regulation interrupt.
PRI OFF—Disables positive supply regulation interrupt.
PRI?—Returns PRI ON or PRI OFF.
RQS ON—Enables generation of service requests.
RQS OFF—Disables generation of service requests.
RQS?—Returns RQS ON or OFF.
USEREQ ON—Enables SRQ when ID button is pushed.
USEREQ OFF—Disables SRQ when ID button is pushed.
USEREQ?—Returns USER ON or OFF.

System Commands

DT SET—Updates hardware after <GET>.
DT OFF—Updates hardware without <GET> message.
DT?—Returns DT SET or OFF.
ERROR?—Returns error code.
ID?—Returns instrument identification and firmware version.
INIT—Initializes instrument settings.
SET?—Returns instrument settings.
TEST—Returns 0 or ROM error code.

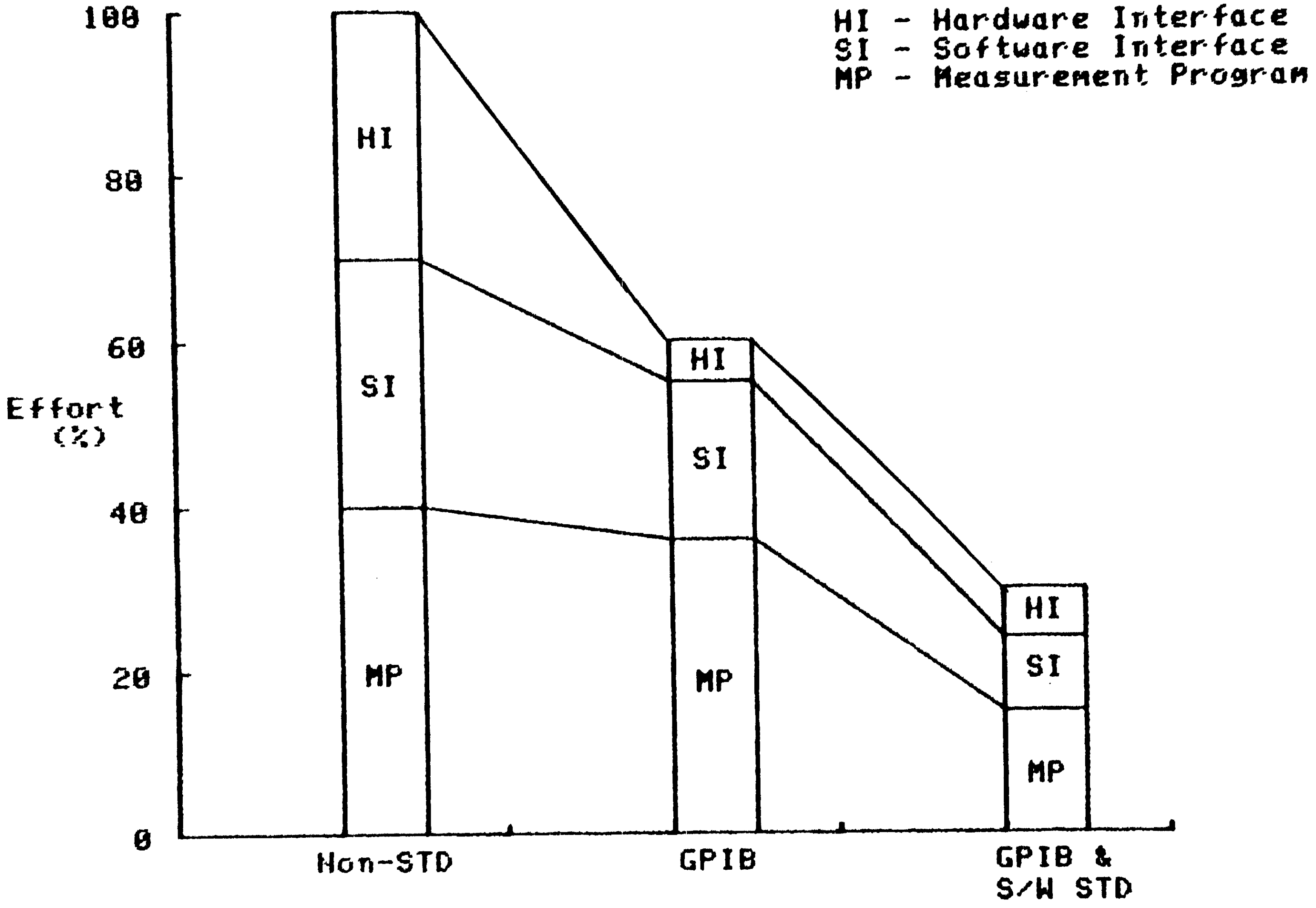
SOFTWARE STANDARDS

- USERS ARE INCREASINGLY INTERACTING WITH SYSTEMS AT THE KEYBOARD RATHER THAN AT THE FRONT OR REAR PANELS OF INSTRUMENTS. THIS MEANS SYSTEMS MUST BE AS FRIENDLY AS POSSIBLE.

- A NEED FOR EASY-TO-LEARN, UNIVERSAL, DEVICE-DEPENDENT CODES, STATUS BYTES, QUERIES and DATA FORMATS

- INSTRUMENT INTELLIGENCE

- IEEE 728 "Codes and Formats Conventions"


HI - Hardware Interface
SI - Software Interface
MP - Measurement Program
100
80
60
40
20
0
Effort (%)
HI
SI
MP
Non-STD
GPIB
GPIB & S/W STD

CODES & FORMATS

-ASSURE SUCCESSFUL SYSTEM OPERATION

-DEVICE-DEPENDENT MESSAGE STANDARD

STANDARD CODES IN STANDARD FORMAT

-PEOPLE COMPATIBLE MESSAGES

-FRIENDLY TO USE

-EASY TO CHANGE OR MODIFY

CHANGE INSTRUMENTS
USE SAME CODE

-COMPATIBILITY AMONG MANUFACTURERS

CODES & FORMATS

(EXAMPLES)

POWER SUPPLY

```
PRINT @ 10: "VPOS 15;OUT ON"
```

DIGITAL MULTIMETER

```
PRINT @ 2: "VDC;RANGE AUTO"
```

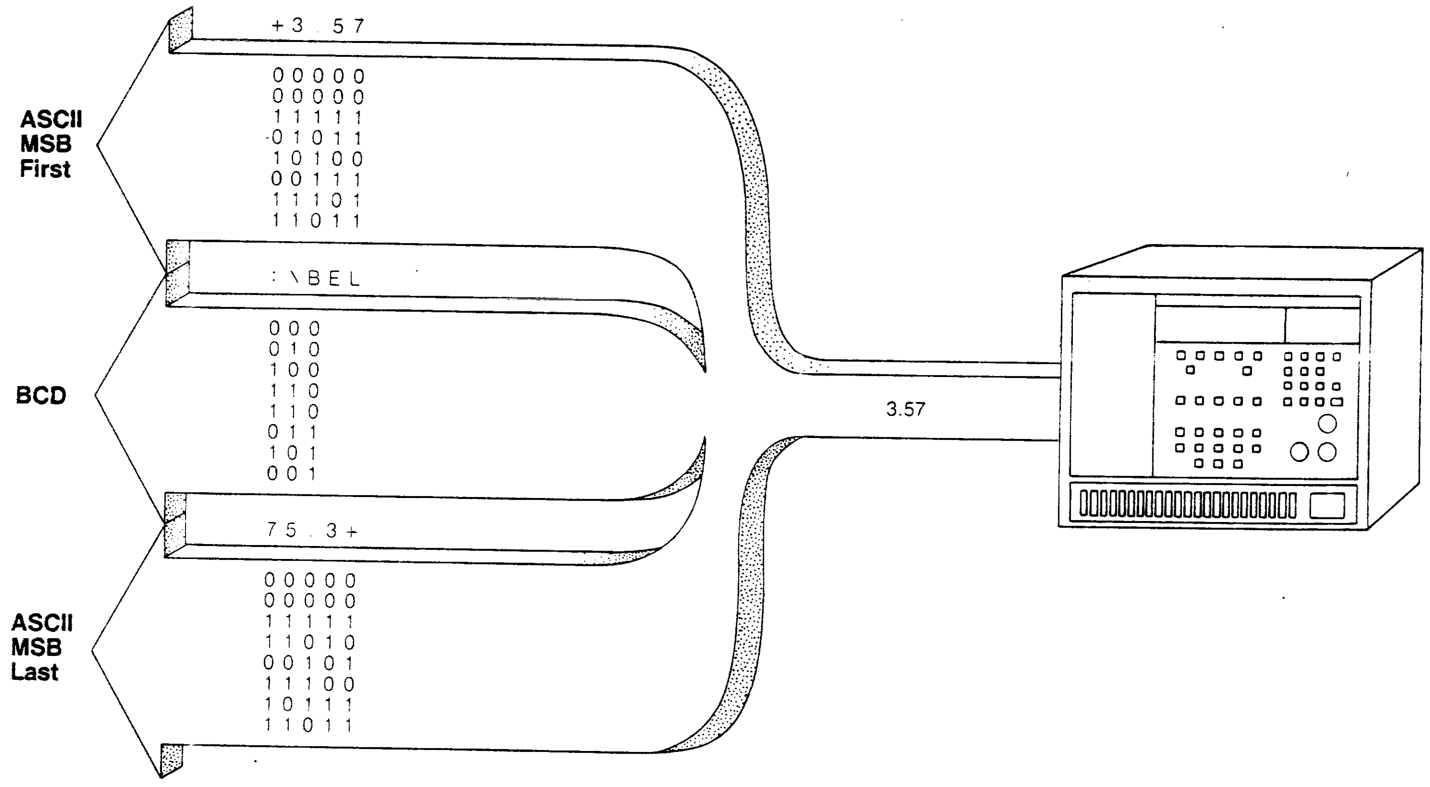
or

```
PRINT @ 7: "F1J1R5T4"
```

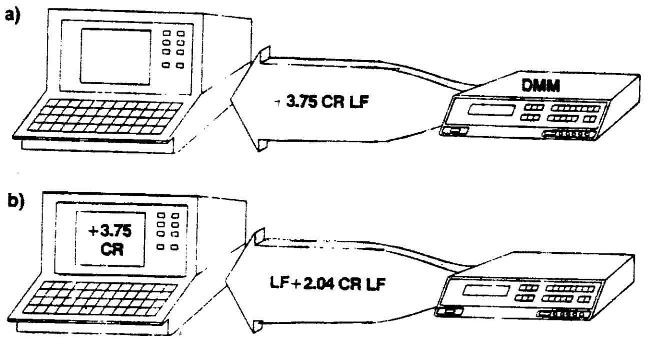
WAVEFORM GENERATOR

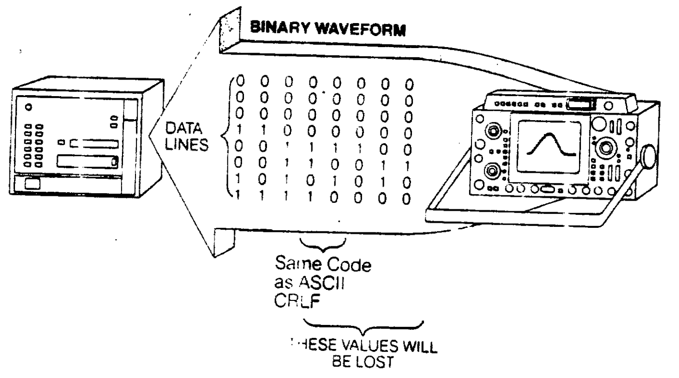
```
PRINT @ 5: "MODE SQUARE;FREQ 2E+6"
```

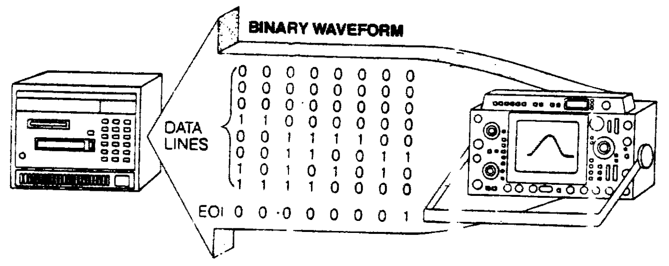
COMPATIBILITY

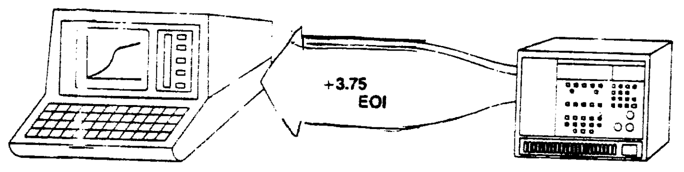


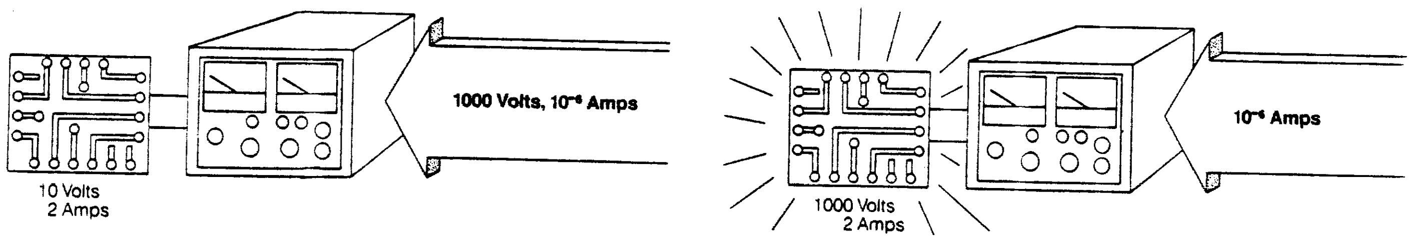
CONVENTIONS











COMMAND ERRORS

will not execute, send SRQ

STATUS BYTES

Abnormal Conditions	Binary	Decimal X = 0	Decimal X = 1
ERR query requested	011X 0000	96	112
Command error	011X 0001	97	113
Execution error	011X 0010	98	114
Internal error	011X 0011	99	115
Power fail	011X 0100	100	116
Execution error warning	011X 0101	101	117
Internal error warning	011X 0110	102	118
Normal Conditions			
No status to report out of the ordinary	000X 0000	0	16
SRQ query request	010X 0000	64	80
Power on	010X 0001	65	81
Operation complete	010X 0010	66	82

QUERIES

* ID?

* SET?

* ERR?

* FUNCTION QUERIES:

  FREQ?
  MODE?
  VPOS?

START-UP CONSIDERATIONS

1. KNOW YOUR INSTRUMENTS
2. CHOOSE THE RIGHT I/O STATEMENT
3. DELIMITERS
4. SYNCHRONIZING
5. INTERRUPTS
6. ASCII vs BINARY
7. IEEE-488 INTERFACE FUNCTIONS
8. COMMON PROBLEMS
9. GPIB SYSTEM PERFORMANCE

KNOW YOUR INSTRUMENTS

* OPERATORS MANUAL
* PROGRAMMERS REFERENCE GUIDE
* COMMAND SET
* COMMAND EXECUTION
  - BUFFERED
  - INTERACTION
* SRQ HANDLING
* ADDRESS SWITCH
* DELIMITER SETTING
* MODES OF OPERATION
* SUBSET IMPLEMENTATION
* DATA FORMATS
  - ONE TYPE
  - MORE THAN ONE
  - SPEED OF TRANSFER

THE RIGHT I/O

* DATA TRANSFER RATES
* CHOOSE I/O STATEMENTS TO MINIMIZE BUS TRAFFIC
* PRINT or OUTPUT
* INPUT or ENTER
* USING/IMAGE STATEMENTS

  A - ASCII CHARACTERS
  K - NUMERIC DATA
  B - BINARY DATA
  W - DOUBLE BINARY WORDS
  # or S - SUPPRESS 'CR'

* RBYTE/WBYTE or SEND 7
* FAST HANDSHAKE

DELIMITERS

CR/LF

EOI

ALTERNATE

```
1 - CONTROL 7,16;128

2 - 10 DIM A(100)
     *
     *
     *
    50 PRINT @4:USING 100:A
     *
     *
     *
   100 IMAGE 100(5D,",")
```

SYNCHRONIZING

- TWO PROGRAMS OPERATING
  - Controller
  - Instrument uP
- HANDSHAKING
- NRFD LINE
- BUS TIMEOUT

INTERRUPTS

ASYNCHRONOUS EVENTS

- TIME CONSUMING

OPC

- MEASUREMENT METHODS:
  1. ON THE FLY
  2. OPC
  3. GET

CONTROLLER INTERRUPTS

```
ON SRQ

ON ERROR

ON IODONE

ON INTERRUPT 7

ON TIMEOUT 7
```

USING INTERRUPTS

```
  10 On srq then call srqhand
  20 Print #1:"WRI ON;ARM A"
  30 Rem *** Continue Program Here
   *
   *
   *
   *
   *
1000 Sub srqhand
1010 Poll status,device;1,0
1020 If status <> 199 then resume
1030 Print #1,0:"READ A"
1040 Input #1,0 using "%":A
1050 Resume
```

ASCII or BINARY

SIMPLICITY vs SPEED

* ASCII

  - SIMPLE
  - FLEXIBLE
    numbers or characters
    various terminators

* BINARY

  - FASTER
  - ONE or TWO BYTES

* TRANSFERS (1000 pts to 4052)

  - BINARY: 350 msec
  - ASCII to string: 8.5 secs
  - ASCII to number: 9.0 secs

INTERFACE FUNCTIONS

FUNCTION	DESCRIPTION	ASSOCIATED MULTILINE (8-bit) INTERFACE MESSAGES
Command and address are merged in these 8-bit messages		
Talker (T)	allows instrument to send data	MTA My Talk Address MSA My Secondary Address
Listener (L)	allows instrument to receive data	MLA My Listen Address MSA My Secondary Address
Source Handshake (SH)	synchronizes message transmission	none
Acceptor Handshake (AH)	synchronizes message reception	none
8-bit messages comprising commands		
Remote-Local (RL)	allows instrument to select between GPIB interface and front-panel programming	GTL (Go To Local) LLO (Local Lockout)
Device Clear (DC)	puts instrument in initial state	DCL (Device Clear) SDC (Selected Device
Device Trigger (DT)	starts some basic operation of instrument	GET (Group Execute Trigger)
Service Request (SR)	request service from controller	SPE (Serial Poll Enable) SPD (Serial Poll Disable)
Parallel Poll (PP)	allows up to eight instruments simultaneously to return a status bit to the controller	PPC (Parallel Poll Configure) PPU (Parallel Poll Unconfigure) PPE (Parallel Poll Enable) PPD (Parallel Poll Disable)
Controller (C)	sends device addresses and other interface messages	UNT (Untalk) UNL (Unlisten) TCT (Take Control)

COMMON PROBLEMS

1. ADDRESSES
   - AUTO POLL or ADDRESS LIST

2. BUS TIMEOUTS
   - DEFAULT
   - INTERRUPTS (ON TIMEOUT)
   - CHANGE TIMEOUT

3. POWER-UP SRQ

4. REMOTE ENABLE
   - INSTRUMENT HANDLER
     drop REN line
   - CONTROLLER HANDLER
     hold REN line
   - GPIB COMMAND

5. GPIB CABLES
   - < 20 METERS
   - DEVICE LOAD 2 METERS

6. POWERED EQUIPMENT
   - MORE THEN 2/3

DATA ACQUISITION

* TIME TO PROCESS COMMANDS
* TIME TO TRIGGER ON EVENT
* TIME TO SETTLE OUT (ACV)
* TIME TO MEASURE
* TIME TO DIGITIZE
* TIME TO OPERATE
* OVERHEAD TIME

OFTEN THE LONGEST TIME

DATA TRANSFER TIME

STATEMENT OVERHEAD
- SYNTAX CHECK
- VARIABLES ANALYSIS
- DEFAULT VALUES
- REDUCE VARIABLES TO CONSTANTS

ADDRESSING SEQUENCE
- BUS OVERHEAD
- ATN LINE
- PRIMARY & SECONDARY ADDRESSES

DATA BURST
- SLOWEST DEVICE ADDRESSED
- ASCII OR BINARY
- AMOUNT OF DATA

BUFFER OVERHEAD
- DATA BUFFER STORAGE (4052 72 CHARACTERS)
- DMA
- FAST TRANSFER/FAST HANDSHAKE

STATEMENT TERMINATION
- CHECK IF TRANSMISSION COMPLETE

UNADDRESSING SEQUENCE
- FINAL BUS TRAFFIC SUCH AS: UNT, UNL, etc.

USE OF HIGH LEVEL OR LOW LEVEL COMMANDS

PROCESSING

I/O PROCESSING -

DEPENDENT ON I/O HANDLER

DATA PROCESSING -

DEPENDENT ON ALGORITHM

$\boxed{\text{DIFFICULT TO ESTIMATE}}$

BEST TO DO ACTUAL TEST

HUMAN INTERACTION

* ENTER PARAMETERS

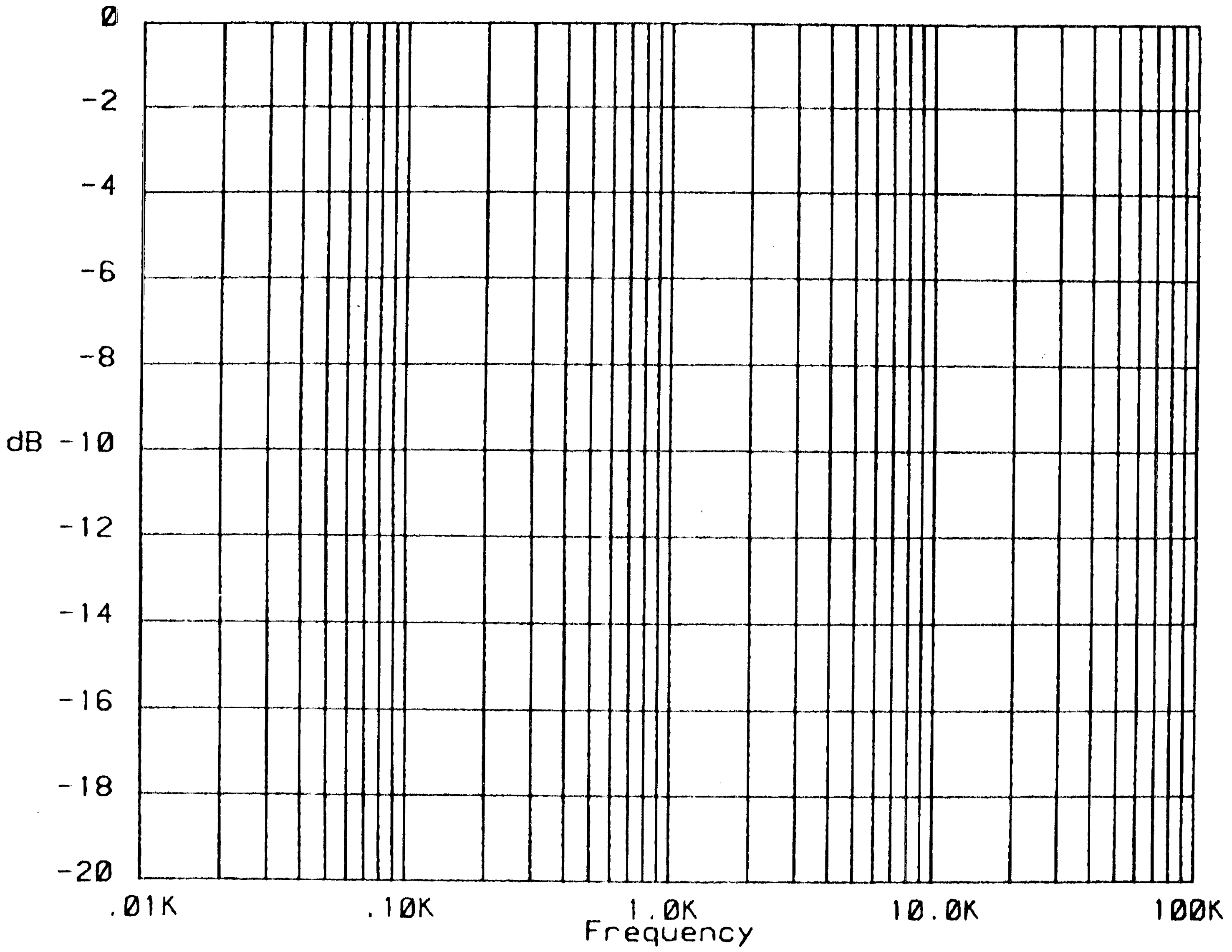
* MAKE ADJUSTMENTS

* SET UP CONFIGURATIONS

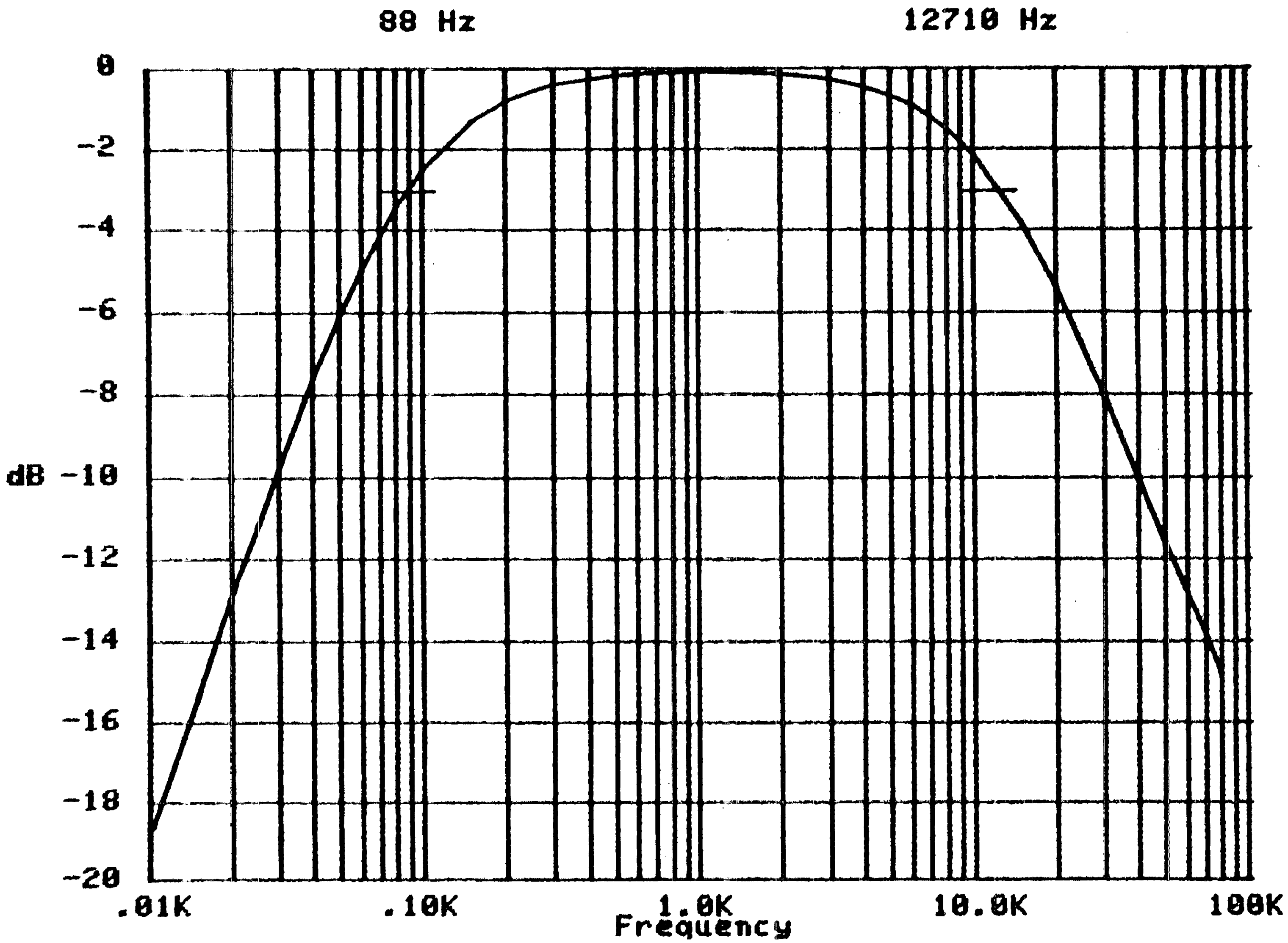
1. CAN BE MINIMIZED BY CAREFUL SYSTEM DESIGN

2. USE FULLY PROGRAMMABLE INSTRUMENTS

3. CAN BE VERY TIME CONSUMING



Ø
-2
-4
-6
-8
dB -1Ø
-12
-14
-16
-18
-2Ø
.Ø1K
.1ØK
1.ØK
1Ø.ØK
1ØØK
Frequency



88 Hz
12710 Hz
0
-2
-4
-6
-8
dB -10
-12
-14
-16
-18
-20
.01K
.10K
1.0K
10.0K
100K
Frequency

Manufacturers of IEEE-488 products

For more information on IEEE-488 products, contact the following manufacturers directly or circle the appropriate numbers on the Information Retrieval Service card. This listing includes all suppliers we could determine provide bus-compatible equipment—computers and peripherals, instruments, integrated circuits, interface boards and systems—for commercial sale. For the convenience of European readers, home-office addresses as well as US sales-office addresses are included for companies whose headquarters are in Europe.

A D Data Systems Inc
200 Commerce Dr
Rochester, NY 14623
716) 334-9649
Circle No 585

Data-acquisition systems, switching systems

ADE Corp
77 Rowe St
Newton, MA 02166
(617) 969-0600
Circle No 586

Semiconductor production equipment

Adret Electronique
12 Av Vladimir Komarov
Trappes 78, Yvelines
France
1) 051 2972
Circle No 587

Signal sources

Analog Devices Inc
Box 280
Norwood, MA 02062
(617) 329-4700
Circle No 588

Automatic device test systems, process-control systems

Analogic Corp
Audubon Rd
Wakefield, MA 01880
(617) 246-0300
Circle No 589

Array processors, data-acquisition systems

Apple Computer Inc
10260 Bandley Dr
Cupertino, CA 95014
(408) 996-1010
Circle No 590

Bus controllers, interfaces

Applied Microsystems Corp
11003 118 Pl Northeast
Kirkland, WA 98033
(206) 823-9911
Circle No 591

In-circuit emulators

Applied Micro Technology
Box 3042
Tucson, AZ 85702
(602) 622-8605
Circle No 592

Interfaces

Auricle Inc
20823 Stevens Creek Blvd
Building C1
Cupertino, CA 95014
(408) 257-9830
Circle No 593

Voice-recognition systems

Autek Systems Corp
3200 Coronado Dr
Santa Clara, CA 95051
(408) 496-0400
Circle No 594

Sampling probes, switching systems, waveform analyzers, waveform recorders

Ballantine Laboratories Inc
Box 97
Booton, NJ 07005
(201) 335-0900
Circle No 595

Calibrators, counters, interfaces, voltmeters

Base^2 Inc
Box 3548
Fullerton, CA 92634
(714) 992-4344
Circle No 596

Printers

Berkeley Nucleonics Corp
1198 Tenth St
Berkeley, CA 94710
(415) 527-1121
Circle No 597

Digital delay generators, interfaces, pulse generators

Boonton Electronics Corp
Box 122
Parsippany, NJ 07054
(201) 887-5110
Circle No 598

Modulation meters, RLC bridges and meters, power meters, signal sources, voltmeters

T G Branden Cop
5565 SE International Way
Portland, OR 97222
(503) 659-9366
Circle No 599

Signal-processing instruments, waveform recorders

Bruel & Kjaer
2850 Naerum
Denmark
02-800500
Circle No 600

Acoustic & vibration measurement instruments & systems

Bruel & Kjaer Instruments Inc
185 Forest St
Marlboro, MA 01752
(617) 481-7000
Circle No 601

Acoustic & vibration measurement instruments & systems

California Computer Systems
250 Caribbean Dr
Sunnyvale, CA 94086
(408) 734-5811
Circle No 602

Interfaces

California Instruments Div
Norlin Industries
5150 Convoy St
San Diego, CA 92111
(714) 279-8620
Circle No 603

Multimeters

Commodore Business Machines
Computer Systems Div
950 Rittenhouse Rd
Norristown, PA 19403
(800) 523-5622
Circle No 604

Bus controllers, interfaces, floppy-disk systems, modems, printers

Computer Automation Inc
Industrial Products Div
2181 DuPont Dr
Irvine, CA 92713
(714) 833-8830
Circle No 605

Automatic board test systems, microcomputers, minicomputers

Computer Data Systems Inc
186-58 Homestead
Morrison, CO 80465
(303) 697-8014
Circle No 606

Switching systems

Comstron/Adret
200 E Sunrise Hwy
Freeport, NY 11520
(516) 546-9700
Circle No 607

Signal sources

Control Logic Inc
9 Tech Circle
Natick, MA 01760
(617) 655-1170
Circle No 608

Interfaces, microcomputers

Data General Corp
Rte 9
Westboro, MA 01581
(617) 366-8911
Circle No 609

Interfaces, microcomputers, minicomputers

Data Laboratories Ltd
28 Wates Way
Mitcham, Surrey CF4 4HR
Great Britain
(01) 640-5321
Circle No 610

(See also **Kontron Electronics Inc**)

Waveform recorders

Data Precision Div
Analogic Corp
Electronics Av
Danvers, MA 01923
(617) 246-1600
Circle No 611

Calibrators, multimeters, oscilloscopes, switching systems

Datatronic AB
Box 42094
S-126 12 Stockholm
Sweden
8-744 59 20
Circle No 612

Interfaces

Datron Electronics Ltd
Meteor Close
Norwich Airport Industrial Estate
Norwich NR6 6HQ
Great Britain
(0603) 412126
Circle No 613

Calibrators, multimeters

Datron Instruments Inc
Laguna Hills Business Park
Laguna Hills, CA 92653
(714) 830-8860
Circle No 614

Calibrators, multimeters

Digital Equipment Corp
Components Group
1 Iron Way
Marlboro, MA 01752
(617) 467-5111
Circle No 615

Interfaces, microcomputers, minicomputers

Dolch Logic Instruments GmbH
Ottostrasse 25
D-6056 Heusenstamm
West Germany
(0 61 04) 64 77-78
Circle No 616

Bus analyzers, logic analyzers

Dolch Logic Instruments Inc
230 Devcon Dr
San Jose, CA 95112
(408) 998-5730
Circle No 617

Bus analyzers, logic analyzers

Dylon Corp
3670 Ruffin Rd
San Diego, CA 92123
(714) 292-5584
Circle No 618

Interfaces, tape drives

Eagle Signal Industrial Controls
Gulf & Western Mfg Co
736 Federal St
Davenport, IA 52803
(319) 326-8111
Circle No 619

Industrial controllers

Eaton Corp
Electronic Instrumentation Div
5340 Alla Rd
Los Angeles, CA 90066
(213) 822-3061
Circle No 620

EMI test instruments & systems, interfaces, signal -sources, spectrum analyzers, synchro/resolver instrumentation

Eaton Corp
Semiconductor Equipment Operations
Memory Test Systems Div
21135 Erwin St
Box 1900
Woodland Hills, CA 91365
Circle No 621

Automatic board test systems, automatic device test systems

EG&G Princeton
Applied Research Co
Box 2565
Princeton, NJ 08540
(609) 452-2111
Circle No 622

Electro-chemical instrumentation, electro-optic instrumentation, signal-processing instruments

EH International Inc
7303 Edgewater Dr
Oakland, CA 94621
(415) 638-5656
Circle No 623

Counters, signal sources, waveform analyzers

EIP Microwave Inc
2731 N First St
San Jose, CA 95134
(408) 946-5700
Circle No 624

Counters

Electro Design Inc
7264 Convoy Ct
San Diego, CA 92111
(714) 277-2471
Circle No 625

Memory simulators

Electro-Metrics Div
Penril Corp
100 Church St
Amsterdam, NY 12010
(518) 843-2600
Circle No 626

Interfaces

Electronic Development Corp
11 Hamlin St
Boston, MA 02127
(617) 268-9696
Circle No 627

Calibrators

Electro Scientific Industries Inc
13900 NW Science Park Dr
Portland, OR 97229
(503) 641-4141
Circle No 628

Laser trimming systems, RLC meters

Elgar Corp
Onan Power Systems Co
8225 Mercury Ct
San Deigo, CA 92111
(714) 565-1155
Circle No 629

Power sources

Epson America Inc
3415 Kashiwa St
Torrance, CA 90505
(213) 539-9140
Circle No 630

Printers

Exact Electronics Inc
Box 347
Tillamook, OR 97141
(503) 842-8441
Circle No 631

Signal sources

Fairchild Semiconductor
464 Ellis St
Mt View, CA 94042
(415) 962-5011
Circle No 632

Bus-interface ICs

Fairchild Test Systems Group
Xincom Div
20450 Plummer St
Chatsworth, CA 91311
(213) 885-1050
Circle No 633

Automatic device test systems

John Fluke Mfg Co Inc
Box C9090
Everett, WA 98206
(206) 342-6300
Circle No 634

Bus controllers, calibrators, counters, data loggers, interfaces, multimeters, printers, signal sources, switching systems, thermometers, voltmeters

GenRad Inc
300 Baker Ave
Concord, MA 01742
(617) 369-4400
Circle No 635

Automatic board test systems, automatic device test systems, RLC bridges & meters

GenRad Semiconductor Test Inc
510 Cottonwood Dr
Milpitas, CA 95035
(408) 946-6960
Circle No 636

Automatic device test systems

Gould Inc Instruments Div
3631 Perkins Ave
Cleveland, OH 44114
(216) 361-3315
Circle No 637

Digital storage scopes, electrostatic recorders

Gould Inc Instruments Div
4600 Old Ironsides Dr
Santa Clara, CA 95050
(408) 988-6800
Circle No 638

Logic analyzers, waveform recorders

Grumman Aerospace Corp
Integrated Logistics Support Dept
ATE Systems—Plant 31
Bethpage, NY 11714
(516) 575-7007
Circle No 639

Automatic board/system test systems, color graphics displays, guided probes, serial bus analyzers, signal analyzers, signal sources, switching systems

Guildline Instruments Inc
2 Westchester Plaza
Elmsford, NY 10523
(914) 592-9101
Circle No 640

Interfaces, voltmeters

Gulton Industries Inc
Measurement & Control Systems Div
Gulton Industrial Park
East Greenwich, RI 02818
(401) 884-6800
Circle No 641

Printers

Hewlett-Packard Co
1507 Page Mill Rd
Palo Alto, CA 94304
Phone local office
Circle No 642

Audio analyzers, automatic board test systems, automatic device test systems, bus analyzers, bus controllers, clocks, computer peripherals, computers, counters, data-acquisition/control instruments and systems, desktop computers, floppy-disk and tape drives, graphics tablets, instrumentation tape recorders, interfaces, logic analyzers, multimeters, network analyzers, oscilloscopes, picoammeters, power meters, power sources, printers, plotters, RLC meters, signal sources, spectrum analyzers, switching and interconnection systems and accessories, telecommunications test equipment, thermometers, timing generators, voltmeters, word generators

Hilevel Technology Inc
14661 Myford Rd
Tustin, CA 92680
(714) 731-9477
Circle No 643

Development systems

ICS Electronics Corp
1620 Zanker Rd
San Jose, CA 95112
(408) 298-4844
Circle No 644

Bus analyzers, bus controllers, bus expanders, clocks, interfaces, OEM interfaces

Information Development & Applications Inc
10759 Tucker St
Beltsville, MD 20705
(301) 937-3600
Circle No 645

Tape drives

Innovative Data Technology
4060 Morena Blvd
San Diego, CA 92117
(714) 270-3990
Circle No 646

Tape drives

Intech Instruments Div
282 Brokaw Rd
Santa Clara, CA 95050
(408) 727-0500
Circle No 647

Logic analyzers

Intel Corp
3065 Bowers Ave
Santa Clara, CA 95051
(408) 987-8080
Circle No 648

Bus-interface ICs

Intel Corp
OEM Microcomputer Systems Operation
5200 NE Elam Young Parkway
Hillsboro, OR 97123
(503) 640-7147
Circle No 649

Interfaces

Interface Technology
150 E Arrow Hwy
San Dimas, CA 91773
(714) 599-0848
Circle No 650

Bus analyzers, bus controllers, word generators

Intersil
10710 N Tantau Ave
Cupertino, CA 95014
(408) 996-5000
Circle No 651

Interfaces

Interstate Electronics Corp
Subsidiary of A-T-O Corp
1001 E Ball Rd
Box 3117
Anaheim, CA 92803
Circle No 652

Signal sources

Ithaco Inc
735 W Clinton St
Ithaca, NY 14850
(607) 272-7640
Circle No 653

Preamplifiers

Keithley Instruments Inc
28775 Aurora Rd
Cleveland, OH 44139
(216) 248-0400
Circle No 654

Ammeters, automatic test systems, electrometers, interfaces, multimeters, switching systems, voltmeters

Kepco Inc
131-38 Sanford Ave
Flushing, NY 11352
(212) 461-7000
Circle No 655

Power sources

Kikusui International Corp
17819 S Figueroa St
Gardena, CA 90248
(213) 515-6432
Circle No 656

Interfaces oscilloscopes, power sources

Kinetic Systems Corp
11 Maryknoll Dr
Lockport, IL 60441
(815) 838-0005
Circle No 657

Interfaces

Kontron Electronics Inc
630 Price Ave
Redwood City, CA 94063
(415) 361-1012
Circle No 658

Bus controllers, counters, multimeters, waveform recorders

Kontron Elektronik GmbH
Breshuer Strasse 2
8057 Eching
West Germany
(089) 31901-1
Circle No 659

(See also **Kontron Electronics Inc**)

Bus controllers, counters, multimeters

Krohn-Hite Corp
Avon Industrial Park
Avon, MA 02322
(617) 580-1660
Circle No 660

Audio instrumentation

Lambda Electronics Corp
Veeco Instruments Inc
515 Broad Hollow Rd
Melville, NY 11747
(516) 694-4200
Circle No 661

Power sources

Leader Instruments Corp
380 Oser Ave
Hauppauge, NY 11787
(516) 231-6900
Circle No 662

Oscilloscopes

LeCroy California
1806 Embarcadero Rd
Palo Alto, CA 94303
(415) 856-1800
Circle No 663

Interfaces, waveform recorders

Marconi Electronics Inc
100 Stonehurst Ct
Northvale, NJ 07647
(201) 767-7250
Circle No 664

Broadcast test equipment, signal sources, telecommunications test equipment

Marconi Instruments Ltd
Longacres
St Albans Herts
(0) 727-59292
Circle No 665

Broadcast test equipment, signal sources, telecommunications test equipment

Matrix Corp
1717 S Saunders St
Raleigh, NC 27603
(919) 833-2837
Circle No 666

Interfaces

MDB Systems Inc
1995 N Batavia St
Orange, CA 92665
(714) 998-6900
Circle No 667

Interfaces

Microcomputer Systems Corp
432 Lakeside Dr
Sunnyvale, CA 94086
(408) 733-4200
Circle No 668

Disk-drive controllers

Micro-Tel Corp
6310 Blair Hill Lane
Baltimore, MD 21209
(303) 823-6227
Circle No 669

Signal sources

Millennium Systems
19050 Pruneridge Ave
Cupertino, CA 95014
(408) 996-9109
Circle No 670

Automatic board test systems, development systems

Motorola Inc
Bipolar Integrated Circuits Group
Box 20912
Phoenix, AZ 85036
Phone local sales office or distributor
Circle No 671

Bus-driver ICs

Motorola Inc
Microsystems Operation
3102 N 56th St
Phoenix, AZ 85018
(602) 244-5714
Circle No 672

Development systems, interfaces

Motorola Inc
MOS Integrated Circuit Group
3501 Ed Bluestein Blvd
Austin, TX 78721
(512) 928-6800
Circle No 673

Bus-interface ICs

National Instruments
8900 Shoal Creek Rd
Austin, TX 78758
(512) 454-3526
Circle No 674

Interfaces

Neff Instrument Corp
1088 E Hamilton Rd
Duarte, CA 91010
(213) 357-2281
Circle No 675

Switching systems

Nicolet Instrument Corp
Oscilloscope Div
5225 Verona Rd
Madison, WI 53711
(608) 271-3333
Circle No 676

Oscilloscopes, waveform recorders

Nicolet Paratronics Corp
2140 Bering Dr
San Jose, CA 95131
(408) 263-2252
Circle No 677

Bus controllers, logic analyzers

Nicolet Scientific Corp
245 Livingston St
Northvale, NJ 07647
(201) 767-7100
Circle No 678

Spectrum analyzers

Norland Corp
Norland Dr
Ft Atkinson, WI 53538
(414) 563-8456
Circle No 679

Waveform analyzers

North Atlantic Industries
60 Plant Ave
Hauppauge, NY 11787
(516) 582-6500
Circle No 680

Angle indicators, voltmeters

Pacific Measurements Inc
488 Tasman Dr
Sunnyvale, CA 94086
(408) 734-5780
Circle No 681

Power meters

Pedersen Instruments
2772 Camino Diablo
Walnut Creek, CA 94596
(415) 937-3630
Circle No 682

Strip-chart recorders

NV Philips Gloeilampenfabrieken
Test & Measuring Dept
TQ III-2
Eindhoven
Netherlands
Phone local office
Circle No 683

(See also **Philips Test & Measuring Instruments Inc**)
(See also **Signetics Corp**)

Bus-interface ICs, counters, logic analyzers, oscilloscopes

Philips Test & Measuring Instruments Inc
85 McKee Dr
Mahwah, NJ 07430
(201) 529-3800
Circle No 684

Counters, logic analyzers, oscilloscopes

Physical Data Inc
8089 SW Cirrus Dr
Beaverton, OR 97005
(503) 644-9014
Circle No 685

Bus controllers, waveform recorders

Pickles & Trout
Box 1206
Goleta, CA 93116
(805) 685-4641
Circle No 686

Interfaces

Plantronics/Zehntel Inc
2625 Shadelands Dr
Walnut Creek, CA 94598
(415) 932-6900
Circle No 687

Automatic board test systems

Polarad Electronics Inc
5 Delaware Dr
Lake Success, NY 11042
(516) 328-1100
Circle No 688

Spectrum analyzers

Precision Filters Inc
303 W Lincoln St
Ithaca, NY 14850
(607) 277-3550
Circle No 689

Filters

Programmed Test Sources Inc
Beaverbrook Rd
Littleton, MA 01460
(617) 486-3008
Circle No 690

Signal sources

Quantrad Corp
19900 S Normandie Ave
Torrance, CA 90502
(213) 538-9800
Circle No 691

Laser trimming systems

Racal-Dana Instruments Inc
18912 Von Karman Ave
Box C-19541
Irvine, CA 92713
(714) 833-1234
Circle No 692
859-8999

Bus analyzers, counters, interfaces, multimeters, power meters, printers, signal sources, switching systems, timing generators, voltmeters

Radiometer Electronics US Inc
31029 Center Ridge Rd
Westlake, OH 44145
(216) 871-7617
Circle No 693

Frequency analyzers, counters

Rair Microcomputer Corp
4101 Burton Dr
Santa Clara, CA 95050
(408) 988-1790
Circle No 694

Microcomputers

RBI Systems
Box 6393
Silver Spring, MD 20906
(301) 949-0430
Circle No 695

Interfaces

Real Time Systems Inc
152 S MacQuesten Parkway
Mt Vernon, NY 10550
(914) 667-0425
Circle No 696

Signal sources

Rockwell International
Electronic Devices Div
3310 Miraloma Ave
Box 3669
Anaheim, CA 92803
Circle No 697

Interfaces, microcomputers

Rod-L Electronics Inc
1185 O'Brien Dr
Menlo Park, CA 94025
(415) 327-5380
Circle No 698

Hi-pot testers

Rohde & Schwarz GmbH & Co
Muehldorfstr 15
D-8000 Munich
West Germany
(089) 4129-1
Circle No 699

Signal sources, telecommunications test equipment

Rohde & Schwarz Sales Co
14 Gloria Lane
Fairfield, NJ 07006
(201) 575-0750
Circle No 700

Signal sources, telecommunications test equipment

Rotek Instrument Corp
220 Grove St
Waltham, MA 02154
(617) 899-4611
Circle No 701

Calibrators

Signetics Corp
Box 409
Sunnyvale, CA 94086
(400) 746-1675
Circle No 702

Bus-interface ICs

SIR-Atlanta Inc
331 Luckie St Northwest
Atlanta, GA 30313
(404) 522-6317
Circle No 703

Automatic test systems

Soltec Corp
11684 Pendleton St
Sun Valley, CA 91352
(213) 767-0044
Circle No 704

Strip-chart recorders

Spectral Dynamics
Box 671
San Diego, CA 92112
(714) 268-7100
Circle No 705

Spectrum analyzers

Scientific Engineering Laboratories
11 Neil Dr
Old Bethpage, NY 11804
(516) 694-3205
Circle No 706

Interfaces

SSM Microcomputer Products
2190 Paragon Dr
San Jose, CA 95131
(408) 946-7400
Circle No 707

Interfaces

Star Micronics Inc
200 Park Ave
New York, NY 10166
(212) 986-6770
Circle No 708

Printers

Summagraphics Corp
35 Brentwood Ave
Fairfield, CT 06430
(203) 384-1344
Circle No 709

Graphics tablets

Systel Computers Inc
538 Oakmead Parkway
Sunnyvale, CA 94086
Circle No 710

Controllers

Systron-Donner Instrument Div
2727 Systron Dr
Concord, CA 94518
(415) 676-5000
Circle No 711

Bus analyzers, bus controllers, calibrators, counters, frequency-response analyzers, multimeters, power sources, printers, signal sources, switching systems

Tektronix Inc
Box 4828
Portland, OR 97208
(800) 547-6711
(800) 452-6773 (in OR)
Circle No 712

Automatic device test systems, bus controllers, calibrators, counters, desktop computers, floppy-disk and tape drives, interfaces, logic analyzers, multimeters, oscilloscopes, plotters, signal sources, spectrum analyzers, testers, waveform recorders

Teledyne Tac
10 Forbes Rd
Woburn, MA 01801
(617) 935-5400
Circle No 713

Device handlers

Texas Instruments Inc
Box 1443, M/S 6404
Houston, TX 77001
(713) 776-6511
Circle No 714

Bus-interface ICs

Three Rivers Computer Corp
160 N Craig St
Pittsburgh, PA 15213
(412) 621-6250
Circle No 715

Bus controllers

Transmagnetics Inc
210 Adams Blvd
Farmingdale, NY 11735
(516) 293-3100
Circle No 716

Synchro/resolver instrumentation

Tri-Data
505 E Middlefield Rd
Mt View, CA 94043
(415) 969-3700
Circle No 717

Floppy-disk systems

Valhalla Scientific Inc
7576 Trade St
San Diego, CA 92121
(714) 578-8280
Circle No 718

Calibrators

Victor Data Products
3900 N Rockwell Ave
Chicago, IL 60618
(312) 539-8200
Circle No 719

Printers

Wang Laboratories Inc
1 Industrial Ave
Lowell, MA 01851
(617) 851-4111
Circle No 720

Interfaces

Wandel u Goltermann GmbH
Box 45
D-7412 Eningen u A
West Germany
(0) 721891-1
Circle No 721

Telecommunications test equipment

W&G Instruments Inc
119 Naylon Ave
Livingston, NJ 07039
(201) 994-0854
Circle No 722

Telecommunications test equipment

Wavetek
9045 Balboa Ave
San Diego, CA 92123
(714) 279-2200
Circle No 723

Interfaces, signal sources

Wavetek Indiana Inc
Box 190
Beech Grove, IN 46107
(800) 428-4424
(317) 787-3332 (in IN)
Circle No 724

Interfaces, signal sources

Wavetek Rockland Inc
Rockleigh Industrial Park
Rockleigh, NJ 07647
(201) 767-7900
Circle No 725

Signal-processing systems, signal sources

Weinschel Engineering
1 Weinschel Lane
Gaithersburg, MD 20760
(301) 948-3434
Circle No 726

Signal sources

Wiltron Co
805 E Middlefield Rd
Mt View, CA 94043
(415) 969-6500
Circle No 727

Network analyzers, signal sources

Wyle Labs
3200 Magruder Blvd
Hampton, VA 23666
(804) 838-0122
Circle No 728

Bus controllers

Ziatech
2410 Broad St
San Luis Obispo, CA 93401
(805) 541-0488
Circle No 729

Bus analyzers, interfaces, OEM interfaces

In 1975, the IEEE defined the Standard Digital Interface for Programmable Instrumentation (IEEE-488-1975). This interface was updated in 1978, becoming IEEE-488-1978, and it is now commonly known as the General-Purpose Interface Bus, or GPIB. Outside North America, the GPIB is known as IEC 625, and some manufacturers also have their own trade names for it.

Many instrument manufacturers are now including the GPIB as an option or standard feature with their instruments. As the use of the GPIB increases, many new or potential users need to understand GPIB operation. This brochure explains how the GPIB actually works.

The Value of GPIB

Before we get into the technical details, let's take a look at why the GPIB was developed.

With the increasing complexity of electronic equipment, manufacturers are finding it necessary to automate their measurement and test systems. This need to automate is occurring in all phases of product development, i.e., research, design, and manufacturing.

One reason for the growing use of automation is that the rapid growth of the electronics industry has resulted in a scarcity of electrical and electronic engineers. Consequently, there is a need to use these engineers more effectively. One way is to automate routine measurement tasks, freeing engineers for more creative (high-payoff) tasks such as design and analysis.

Automated instruments also provide additional benefits with precise, repeatable measurements that can be duplicated on the manufacturing floor and in field service. Audit trails are easily incorporated into such systems for later failure analysis and/or documentation of specific tests for customers and government agencies. The list of benefits goes on and on. Contact your Tektronix sales engineer for other documents discussing the benefits of GPIB instruments.

Before the GPIB was developed, automation of instrument systems was economically feasible only in manufacturing environments where the volume of testing was very high, or where the cost of the system was low compared to the value of the measurement/ test results. The high costs of these systems resulted from the fact that a custom interface had to be designed for each instrument in a system, a costly proposition. These custom interfaces also kept programming and maintenance costs high. Now the GPIB makes systems economical and more practical.

It is now possible to assemble an automated measurement system with relative ease so system costs can be reduced dramatically. Thus, automated measurement systems are now feasible for engineering benches, low-volume production, quality assurance, and many other applications where such systems were previously impractical or too expensive. Now let's take a look at how a GPIB system works.

GPIB System Components

An automated test-and-measurement system usually consists of the following components:

- A number of instruments. These are either stimulus instruments, such as frequency generators, pulse generators, and power supplies, or measurement instruments, such as counters, waveform digitizers, and multimeters.
- A controller that tells the instruments what to do, collects the results, and processes them. This system controller is generally a small computer.
- Computer peripherals, such as tape drives, printers, and plotters that store or display the results of the tests.
- Keyboards that let the user send commands or information to the system.
- Displays for monitoring system operation.

The last three types of system components are often incorporated in the system controller, a desktop computer such as the Tektronix 4052 that is specifically designed for automated instrument systems.

All these components can be easily interconnected if the standard GPIB interface has been built into them. Before the GPIB existed, most measurement systems were operated by controllers that required a

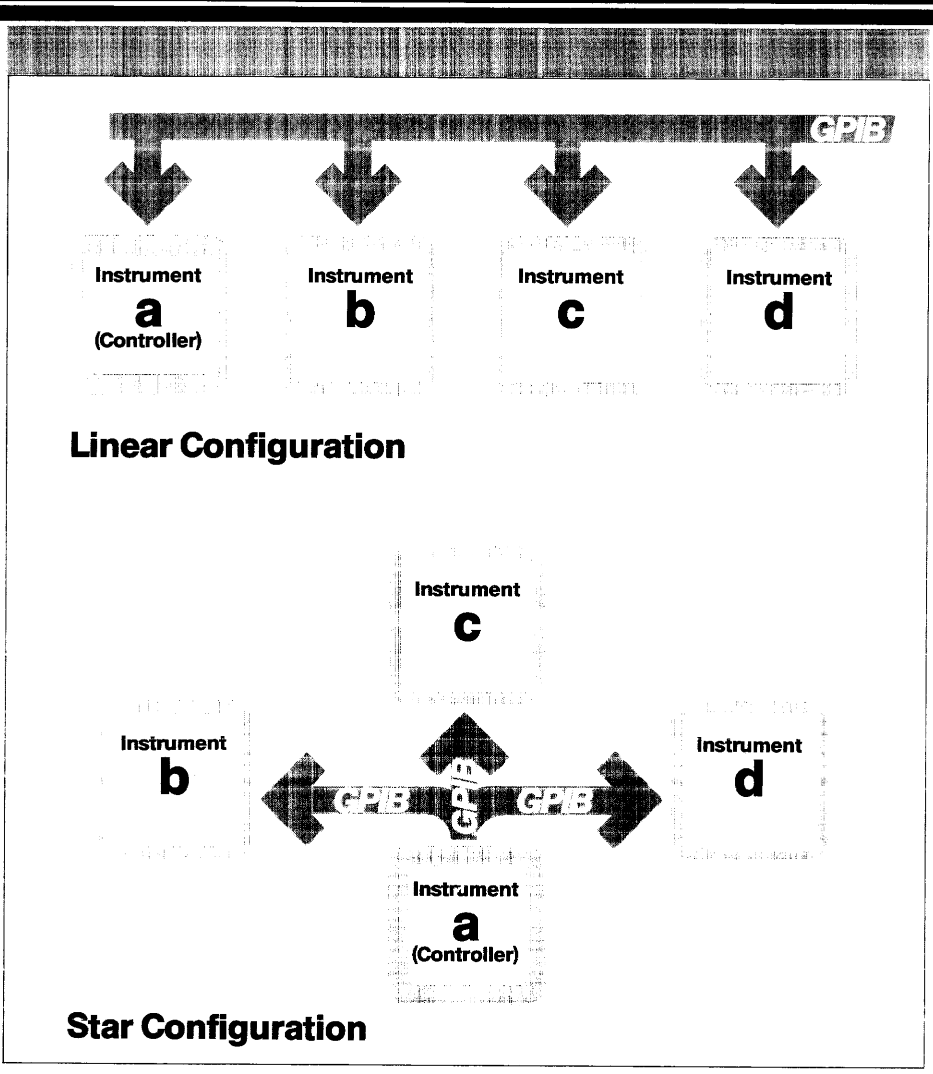


Figure 1. The GPIB, with its standard connector and cable, gives versatility in system setup.

separate connector for each instrument. With the GPIB, controllers no longer require all of these separate ports. Users can directly link up to 14 instruments with the bus, which connects to the controller, and set up the systems in linear or star formations. Figure 1 shows two typical system configurations. Each GPIB instrument or peripheral, called a device, must be assigned a system address.

The user assigns each device its address by appropriately setting switches located on the back panel of the device.

Standard cables are used to connect all the devices as shown in figure 2.

All these devices—the controller, instruments, and peripherals—are known as hardware. But the hardware in a system cannot operate unless it is driven by software, the other part of an automated system. The user of the system must generate the application software for his particular system. The software works through the controller to tell the instruments what signals to generate and what measurements to make, plus it tells the controller what to do with the results.

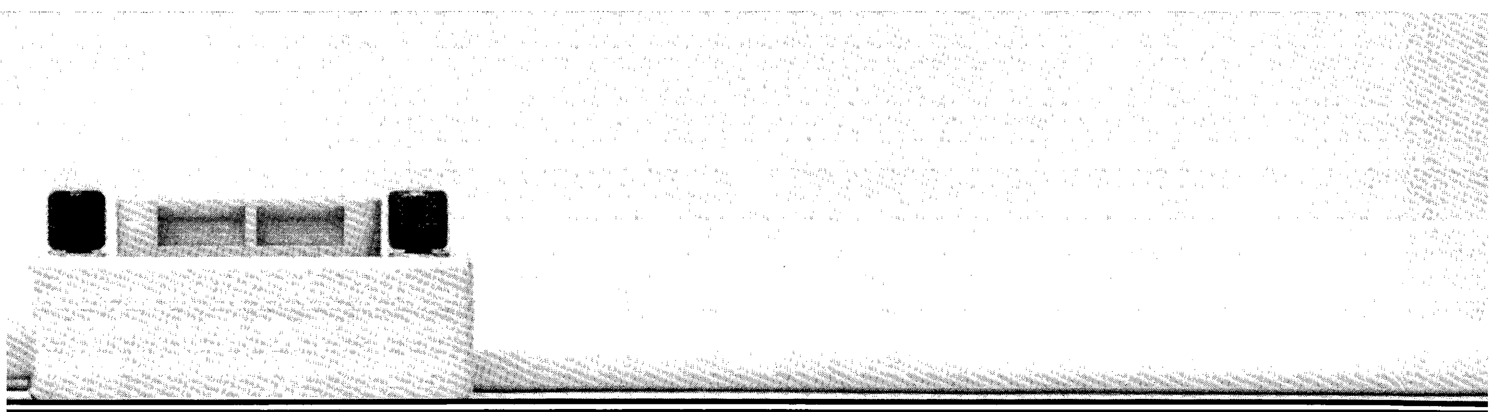
Software Makes the System Work

The software, the program in the controller, makes the system do what the user wants. The GPIB interface lets the user plug system components together, but without software the system can do nothing.

In programmable instrument systems, the term software or program has several meanings:

1) The controller has its own language, such as BASIC or FORTRAN, and users must express their intentions in this language.
2) Within the context of the controller's language, the instrument's commands (or language) have to be sent over the GPIB.
3) The actual GPIB interface must be controlled by the controller based on the user's intentions as expressed in the controller's language.

This software, or program, may sound complicated, but it actually is not. Instrument controllers such as the Tektronix 4052 have built-in capabilities that simplify the job of instrument/system programming. For example, let's look at a typical GPIB-programmable power supply.



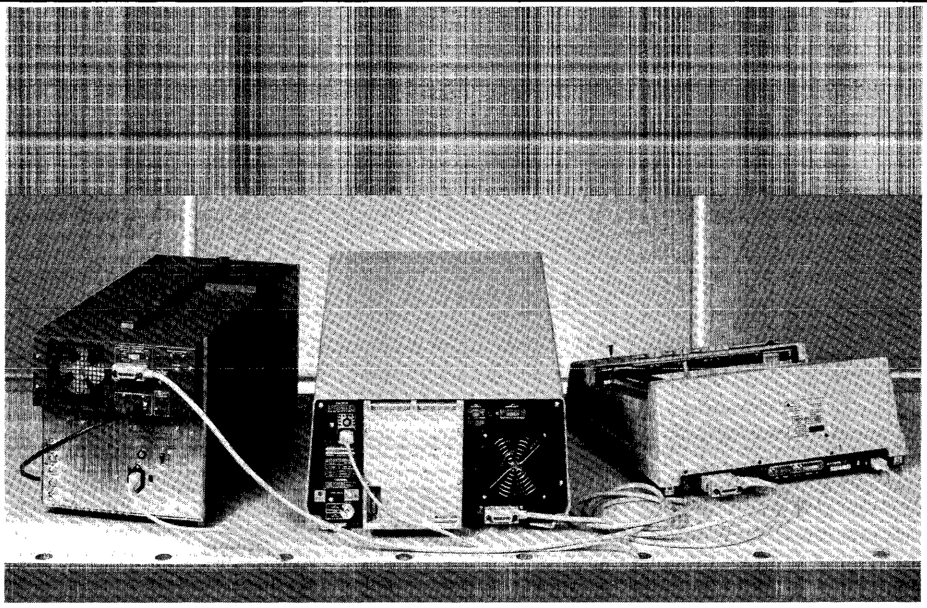


Figure 2. Rear view of a GPIB system composed of a Tektronix 4052 Controller, a 7854 Oscilloscope, and a 4662 Plotter shows simple, uncluttered cabling.

All that's needed to make the power supply put out 15.7 volts is to type into the 4052 the following:

PRINT @ 7: "15.7"

When the controller executes this function, it will automatically perform the GPIB data transfer required to set the proper voltage. PRINT @ 7 addresses device 7 to be a listener. "15.7" sends 15.7 coded in ASCII to the listening device. The user does not need to know anything about the various functions performed by the GPIB. The software makes these functions occur automatically.

A user wanting to learn what actually occurred on the GPIB can link a Tektronix DF2 Logic Analyzer to the bus. The DF2 will show on its screen everything that happened when the instruction was executed. Figure 3 shows what happened when PRINT @ 7: "15.7" was executed.

In order to make the system operate, the user has to:

1) Know what tasks the system is to perform. The system can do nothing by itself.
2) Know the computer's language.
3) Know the kind of data or language the instruments are designed to receive.

But the user does not have to know very much about the operation of the GPIB since the software automatically handles GPIB operations.

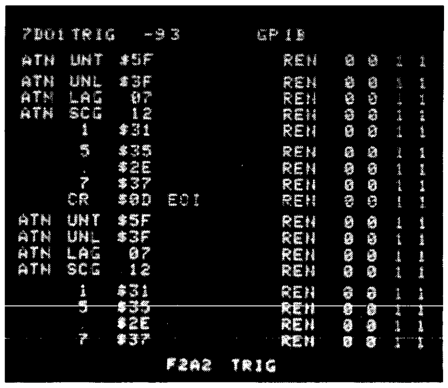
To explore the workings of software further, consider another example. Suppose the user wants to obtain a voltage measurement from a multimeter set at address 2. The user must first program the computer so it will make the multimeter take a voltage reading, and then the user must select the range within which the reading is to be taken. The necessary instruction, as part of a previously stored program typed on the terminal keyboard, would be:

PRINT @2: "VOLTS DC; RANGE AUTO"

When executed, this statement makes the multimeter a listener and sends it the device-dependent message VOLTS DC; RANGE AUTO. (Device-dependent messages are information such as program instructions, measurement data, and test results that are dependent on the nature of a particular device.)

INPUT @2: X

INPUT @2 makes the computer send an interface message instructing the multimeter to be a talker. (Interface messages are commands the controller sends to



```
7D01 TRIG   -93          GPIB
ATN  UNT  $5F                 REN   0 0 1 1
ATN  UNL  $3F                 REN   0 0 1 1
ATN  LAG   07                 REN   0 0 1 1
ATN  SCG   12                 REN   0 0 1 1
       1  $31                 REN   0 0 1 1
       5  $35                 REN   0 0 1 1
       .  $2E                 REN   0 0 1 1
       7  $37                 REN   0 0 1 1
      CR  $0D  EOI            REN   0 0 1 1
ATN  UNT  $5F                 REN   0 0 1 1
ATN  UNL  $3F                 REN   0 0 1 1
ATN  LAG   07                 REN   0 0 1 1
ATN  SCG   12                 REN   0 0 1 1
       1  $31                 REN   0 0 1 1
       5  $35                 REN   0 0 1 1
       .  $2E                 REN   0 0 1 1
       7  $37                 REN   0 0 1 1
              F2A2  TRIG
```

Figure 3. Verifying GPIB activity using a Tektronix Logic Analyzer.

set up device interfaces for data transfers.) The multimeter then sends back, in the computer's language, a number that is assigned to the variable X. The user can now treat variable X in a number of ways: use it in calculations, display it, store it, etc. And if it is desirous to observe what happened on the GPIB, the DF2 Logic Analyzer can be used again.

The point of this second example is to show how the multimeter knew the proper form in which to send its data. The IEEE-488 standard does not specify how data are to be sent; the standard merely provides for eight data lines. In this example, the multimeter sent data as ASCII-coded bytes, the most significant bit first. Fortunately, the computer knew how to accept data of this type. However, if the multimeter had sent another kind of data, say BCD-coded data, or had sent the most significant bit last, the computer would have been confused, and the user would not have obtained a proper reading.

GPIB devices usually send messages encoded in ASCII, a standard binary coding for each character. However, some instruments send data in other coding schemes, such as Binary-Coded Decimal (BCD) code, where each decimal digit is encoded into a four-digit binary code. If a user assembles a system from devices that use different codes, incompatibility will make system operation more cumbersome. The system can operate more efficiently if all the devices use the same codes and formats.

Codes and Formats Assure Successful System Operation

To increase compatibility among devices on the GPIB, Tektronix (among other manufacturers) is using a Codes and Formats standard. This standard defines how data are to be sent and received over the GPIB. Codes and Formats also explains some other conventions that are not clearly stated in the IEEE-488 standard. In the multimeter example above, the multimeter sending data and the computer receiving it were compatible because both followed the Codes and Formats standard.

The Tektronix Codes and Formats standard defines what are for the most part *de facto* standards in the programmable instrument industry. Not all instruments follow these implicit or explicit standards, however. Instrument buyers should therefore be aware that there may be differences with instruments designated as GPIB- or 488-compatible (even among instruments from the same manufacturer).

The Tektronix Codes and Formats standard goes beyond conventional practice to stipulate how programmable instruments are to be compatible not only with controllers but also with people. Compatibility with system users is called friendliness, and the power-supply example demonstrates friendliness clearly. The controller sent "15.7" to the power supply, directing it to put out 15.7 V. However, the IEEE-488 standard does not say that such numbers must be sent. Instead, what might be needed to get an output of 15.7 V from another type of power supply is to send it the characters "2314." But what does 2314 have to do with 15.7 V? A user not knowing the instrument's special programming needs will not know that the 2 represents range 2, 0-50V, and 314 is for 314/1000 of the full range. This power supply is programmable, and it complies with the IEEE-488 standard. But it is not very friendly. The fewer such special conventions a user needs to know, the easier it is to use instruments in a system.

Because users are increasingly interacting with systems at the keyboard rather than at the front or rear panels of instruments, systems must be as friendly as possible. This means, too, that the controller languages should be simple, logical, and easy to interpret or implement.

Details of GPIB Operation

The IEEE-488 standard defines the mechanical, electrical, and functional (what the interface can do) aspects of the General-Purpose Interface Bus. One important mechanical feature of the GPIB is that a standard 24-pin connector is mounted on each instrument. The Bus is defined as a standard cable with the proper mating connectors and 24 wires. Sixteen of those wires carry signals; the remainder are grounds and the cable shield.

Eight of the 16 signal lines are used for transferring data. These data lines carry each byte, or eight-bit coded message, of data over the bus. (Note that a byte commonly represents one character of information.) Five other lines carry interface-management signals. The remaining three lines perform the handshake, which ensures that each byte of data is sent and received correctly. Figure 4 depicts the lines used for the three main functions and shows how a typical system could be configured.

Data. Eight of the 16 signal lines carry data a byte at a time to or from each instrument. These data lines are thus shared; that is why the GPIB is called a bus. It is a dedicated set of lines that carries information to all the devices in the system. Because the lines are shared, the controller designates which instruments are to use the bus at any given time. It assigns one instrument to talk, and send data, and one or more to listen, and receive data.

Handshake. Three lines (NRFD, DAV, and NDAC—see figure 4) are used for a handshake to ensure that data are properly sent and received. Simply put, the handshake works like this:

1) A listener (or listeners, there may be several) indicates on the NRFD line that it is ready to accept data.
2) The talker puts a byte of data on the data lines and indicates on the DAV line that the data are valid.
3) The listener indicates on the NDAC line that it has accepted the data.

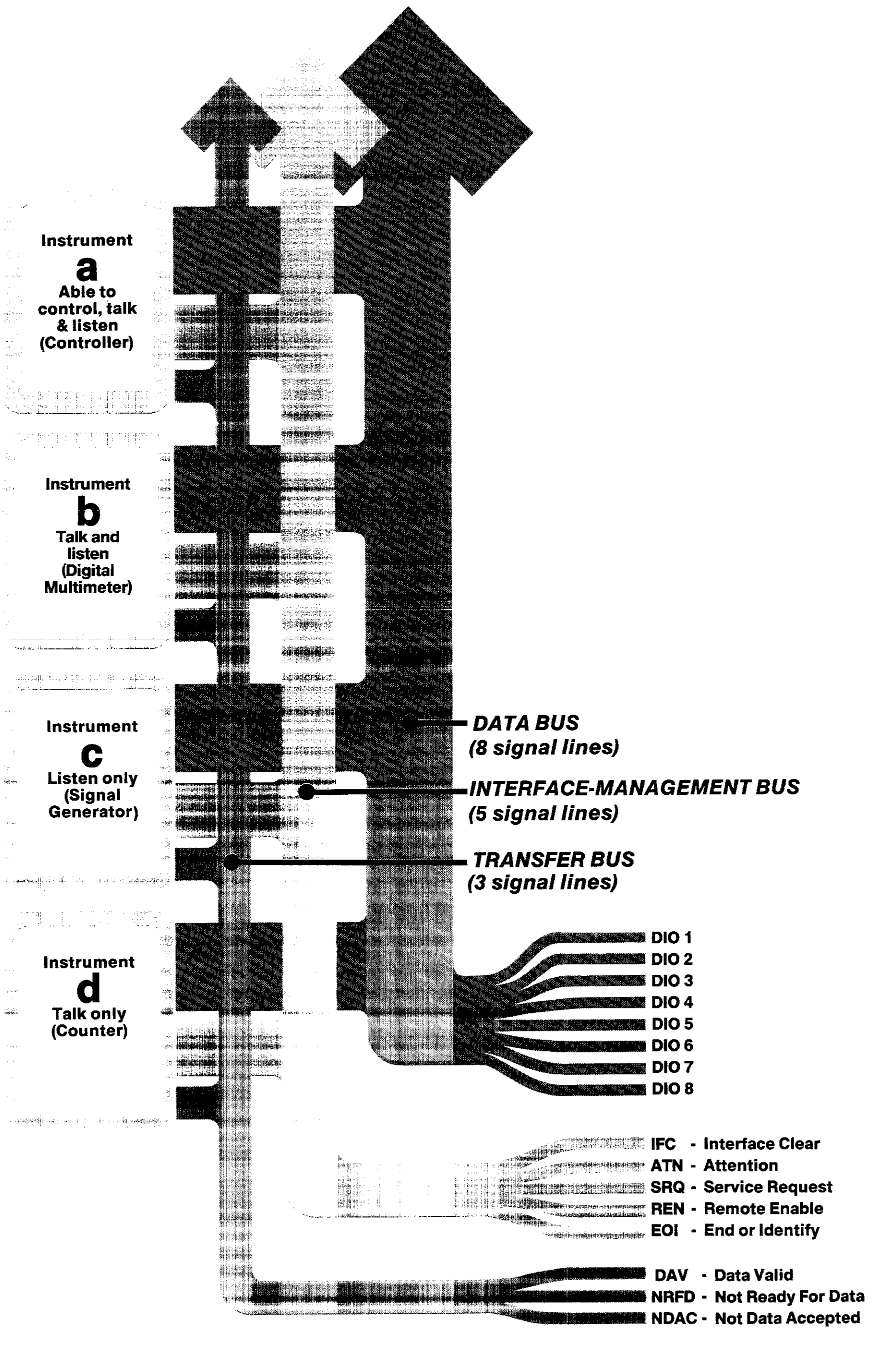


Figure 4. The GPIB is made up of a data bus, a transfer bus, and an interface-management bus.

In other words, the handshake sequence is a dialogue between the listener(s) and the talker that goes like this:

listener(s):	"I am (we are) ready."
talker:	"Here is a byte."
listener(s):	"I've (we've) got it."

The handshake continues until all the data are transferred from the talker to the listener(s).

An advantage of the handshake is that it makes possible the use of both fast and slow devices on the bus (the bus is asynchronous). Since the rate of data transfer is determined by the speed at which the data can be transferred to/from the slowest device involved in the transaction, no data are lost. An additional advantage is that the data rate automatically increases whenever the slower devices are not involved in the transaction.

Management. The remaining five signal lines are used for managing the bus. The most important of these for understanding how the GPIB works is the attention line, ATN.

All the instruments share the GPIB data lines, so the controller has to designate which instruments can use the bus for talking or listening at any given time. The controller does this by sending interface messages to the instruments over the eight data lines. To distinguish these interface messages from the device data that otherwise go over the bus, the controller asserts the ATN line. These interface messages accomplish such tasks as making a device a talker or listener, making all devices stop listening or talking, sending a trigger that initiates action by one or more instruments, and others.

The other bus-management signals are:

- Interface Clear (IFC)—The controller asserts this line to place all the device interfaces in a known quiescent state, e.g., initialization.
- Service Request (SRQ)—Any instrument can assert this line to request service from the controller. The controller then determines which instrument has requested service and why.
- Remote Enable (REN)—The controller asserts this line to control the instruments remotely. This signal line causes the instruments to select between two sources of settings: information from their front panels (local) or information sent to them over the GPIB (remote).
- End or Identify (EOI)—This line is mainly used by any talker to indicate that it is through sending particular device-dependent data.

GPIB Allows Ten Interface Functions

The IEEE-488 standard defines ten functions or tasks that an instrument's interface may perform (see table 1). These functions are activated or deactivated by various interface messages sent by the controller to the instrument interfaces. All these functions are optional, allowing the designer of GPIB-compatible instruments to choose only those suitable for the particular instrument. For example, a simple power supply might only require Listener and Acceptor Handshake functions. With these, it could receive commands from the controller regarding what voltage to put out.

Most GPIB instruments incorporate all but the controller function. However, buyers of GPIB instruments should check to make sure the required functions are incorporated. Many instruments are labeled "488-Compatible," or "GPIB-Compatible," but in the extreme the label may only mean that the instruments have the standard connector. See table 2 for a list of some of the other characteristics of the GPIB.

Further, even though an instrument may be able to perform the general function required, the user needs to know the capabilities of the instrument as it performs that function in the designated application. Within each function, the IEEE-488 standard defines various capabilities. The prospective buyer/user should check with the instrument's vendor to be sure the instrument can perform as required.

How the Bus Handles a Typical Data Transfer

Now that the various signal lines, message types, and functions have been explained, let's see what happens on the bus when, for example, a simple power supply is programmed to put out 15.7 V. First, the controller asserts the attention line and sends out the listen address of

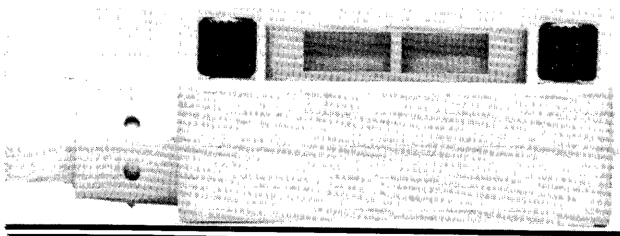


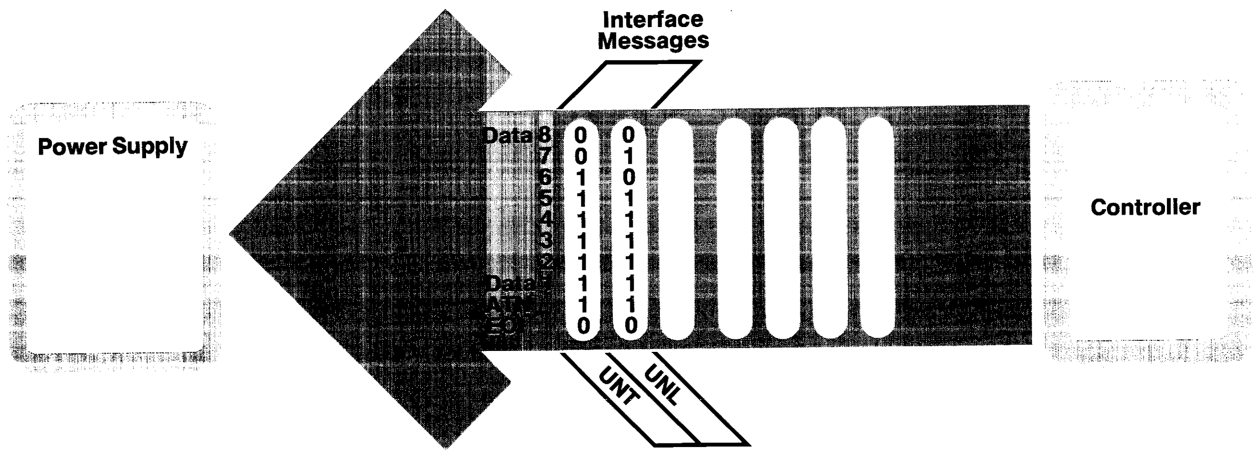
Table 1. **Interface Functions and Corresponding Commands.**

Function	Description	Associated Multiline (8-bit) Interface Messages
Command and address are merged in these 8-bit messages		
Talker (T)	allows instrument to send data	MTA My Talk Address MSA My Secondary Address
Listener (L)	allows instrument to receive data	MLA My Listen Address MSA My Secondary Address
Source Handshake (SH)	synchronizes message transmission	none
Acceptor Handshake (AH)	synchronizes message reception	none
8-bit messages comprising commands		
Remote-Local (RL)	allows instrument to select between GPIB interface and front-panel programming	GTL (Go To Local) LLO (Local Lockout)
Device Clear (DC)	puts instrument in initial state	DCL (Device Clear) SDC (Selected Device Clear)
Device Trigger (DT)	starts some basic operation of instrument	GET (Group Execute Trigger)
Service Request (SR)	requests service from controller	SPE (Serial Poll Enable) SPD (Serial Poll Disable)
Parallel Poll (PP)	allows up to eight instruments simultaneously to return a status bit to the controller	PPC (Parallel Poll Configure) PPU (Parallel Poll Unconfigure) PPE (Parallel Poll Enable) PPD (Parallel Poll Disable)
Controller (C)	sends device addresses and other interface messages	UNT (Untalk) UNL (Unlisten) TCT (Take Control)

Table 2. GPIB Hardware Characteristics.

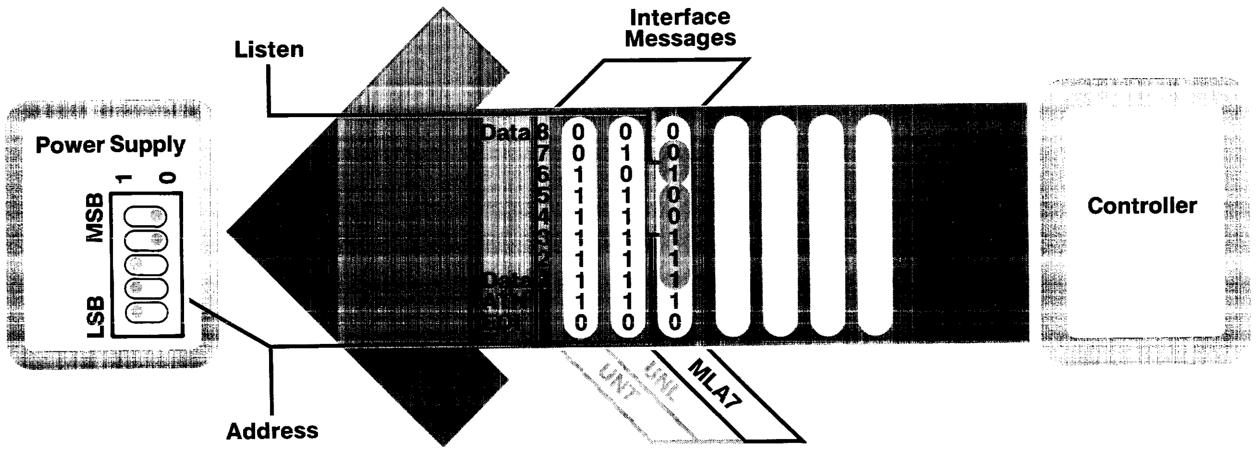
- Cable length of up to 20 meters (approximately 66 feet) with a device load for every 2 meters of cable, i.e., $\leq 2\times$ number of instruments.
- Up to 15 devices (1 controller and 14 instruments) may be connected in linear or star configurations.
- Voltages are generally TTL-compatible.
- GPIB signal and data lines are asserted (or true) when pulled low (+ 0.8V), and released (or false) when pulled high (+ 2.0V). For instance, ATN indicates the ATN line is asserted; $\overline{\text{ATN}}$ that it is unasserted.
- Maximum data rate of up to 250 kilobytes/second over a distance of 20 meters with 2 meters per device, or 1 megabyte/second over a distance of 15 meters, tuned up. Refer to IEEE std 488-1978 for details.

PRINT



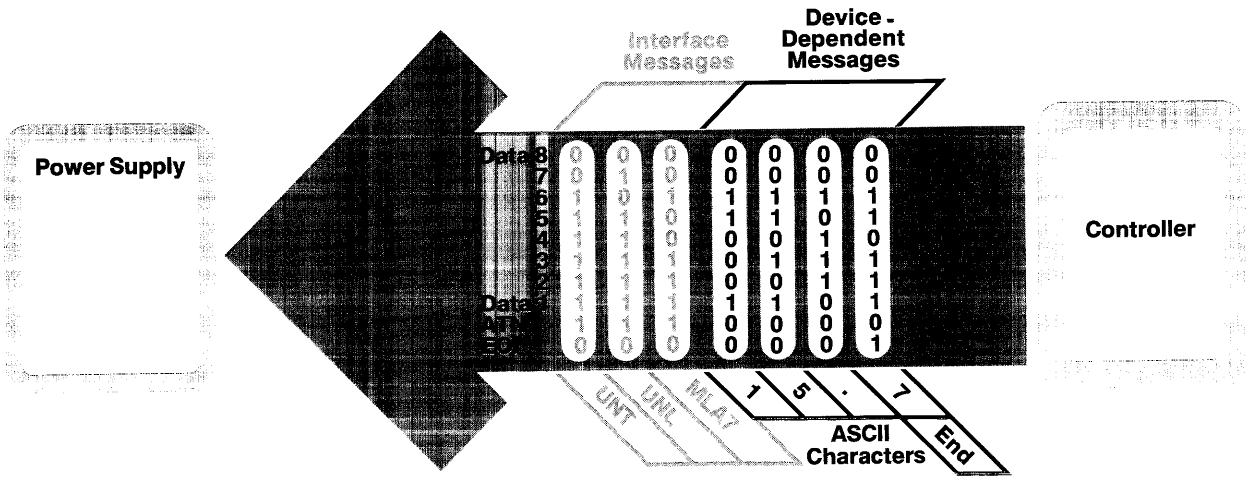
a.

PRINT @ 7:



b.

PRINT @ 7:15.7



c.

the power supply (and remember that the user will have set that address with the switches on the back panel of the power supply). At this point, the controller is sending data, and all the devices are receiving the data by handshaking with the controller. Only the device whose address switches match the listen address sent by the controller will become a listener. Other devices ignore this interface message (see figure 5a,b).

Next, the controller unasserts the attention line, and makes itself a talker. Now, a device-dependent message is sent to the power supply, which at this point is a listener. In this example, the device-dependent message consists of the four characters "15.7" coded in ASCII (figure 5c). Note that the End or Identify line is asserted with the last character, telling the power supply that the message is through and can now be executed. The power supply then puts out 15.7 V. Figure 5 portrays graphically the entire message sequence just described.

The simple program developed by the user (who wanted the power supply to put out 15.7 V) enabled the controller to do all that was required.

How to Select GPIB-Compatible Instruments

Now that you have a basic understanding of the workings of a GPIB system, you're ready to begin planning your own configuration. Selecting GPIB-compatible instruments for use in an automated system requires careful consideration of the specific functional and GPIB capabilities required for the instruments to operate properly in the system. Table 3 lists some instrument capabilities to consider when selecting components for a GPIB-based system. Of course, the instrument should be able to perform these functions over the GPIB.

To assure functional compatibility of system devices, you should obtain from the manufacturers as much information as possible about each instrument considered for use in the system. You should also be certain that the instruments selected are compatible with the system's controller. Devices with compatible interface functions can simplify programming and improve system efficiency. The better informed you are about system components, the easier it will be to assemble friendly, efficient systems that perform all the necessary functions. The result will be fast, cost-effective solutions to your unique test-and-measurement problems.

Table 3. Instrument Capabilities to Consider for GPIB-Compatible Instruments.

Does the instrument:

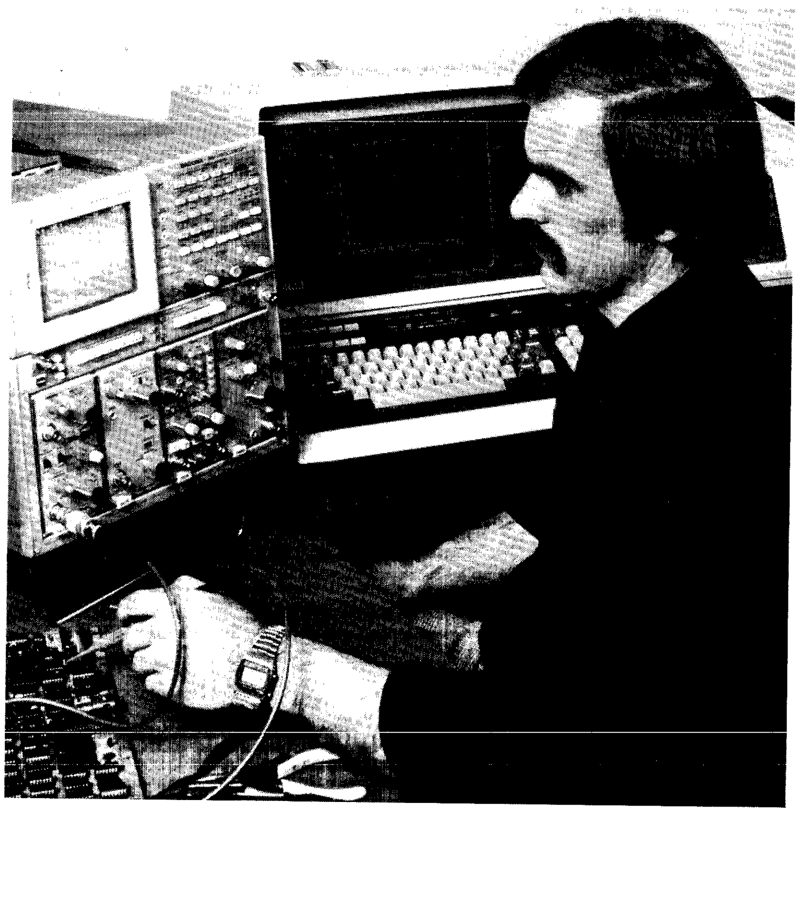
- Execute some or all front-panel functions?
- Request service and respond to status inquiries?
- Load, execute, and report results of self-diagnostic programs?
- Store and execute setup programs to be triggered through the GPIB?
- Process measurement results before sending (i.e., data compression and/or analysis)?
- Accept English-word or English-mnemonic commands and programs?
- Check the syntax of commands before executing them? And, with what constraints?
- Follow the Codes and Formats Standard?
- Check the validity of what the instrument is being programmed to do and then do nothing unusual if it can't execute a command properly?

Figure 5. This sequence shows that first the interface message is sent (a,b) and is then followed by the device-dependent data (c). The combined message illustrates the activity of the ATN and EOI lines.

TEK GPIB IEEE-488 (1978) PLUS CODES AND FORMATS

TEKTRONIX CODES AND FORMATS FOR GPIB INSTRUMENTS

GPIB



Tektronix®
COMMITTED TO EXCELLENCE

Contents

As their measurement needs have grown in number and complexity, many instrument users have realized that their traditional design and test procedures are inadequate. They have new needs which can be satisfied by assembling their individual instruments into interactive and automated systems that will:

- Reduce labor costs.
- Increase the effective use of research and design skills by freeing them for creative work.
- Provide insight into products and processes through coupling analysis with measurements.
- Reduce human errors with precise and repeatable measurements.

Meeting these needs requires instruments, controllers, and peripherals which are easy-to-use with each other.

The first major step toward device compatibility was taken in 1975 when the IEEE published the 488 standard defining an interface for programmable instruments. This bus is usually called the GPIB—General Purpose Interface Bus. Before GPIB, connecting programmable instruments to a computer or to a desktop calculator was a major job because each instrument's interface was different. Now, the IEEE Standard 488 defines an interface that makes it much easier to put together computer-controlled instrument systems.

The IEEE-488 standard defines three aspects of an instrument's interface:

1. Mechanical—the connector and the cable.
2. Electrical—the electrical levels for logical signals and how the signals are sent and received.
3. Functional—the tasks that an instrument's interface is to perform, such as sending data, receiving data, triggering the instrument, etc.

Using this interface standard, instruments can be designed for base compatibility. However, it is only the *first* step toward further standardization for even greater compatibility.

Tektronix has taken the next step by adopting a new standard called Codes and Formats. It is intended to:

- Define device-dependent message formats and codings and thus enhance compatibility among instruments that comply with IEEE Standard 488-1978,
- Reduce the cost and time required to develop system and applications software by making it easier for people to generate and understand the necessary device-dependent coding.

Beyond the Codes and Formats standard, there is also a need for a philosophy of designing instruments to be friendly to the user. They should be controlled over the bus with easily understood commands and should be resistant to operator errors. Since the application of this philosophy is different for each type of instrument, it is not included as a specific standard.

This document explains the details of, and reasons for, the Tektronix Codes and Formats standard; as well as the concept of instruments designed for people.

Setup for Automated Measurement Analysis

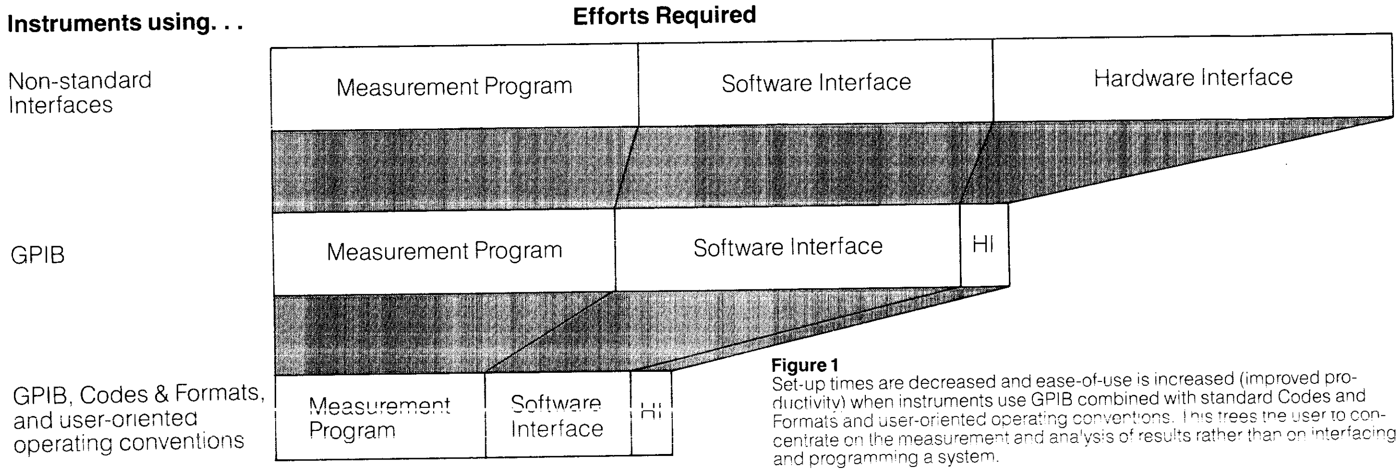


Figure 1
Set-up times are decreased and ease-of-use is increased (improved productivity) when instruments use GPIB combined with standard Codes and Formats and user-oriented operating conventions. This frees the user to concentrate on the measurement and analysis of results rather than on interfacing and programming a system.

TEKTRONIX GPIB IMPLEMENTATION

Compatibility plus Capability

A key factor in determining the productivity of an instrument is its user interface. Previously most interfaces were designed with the instrument logic being the prime concern. Ease-of-use was often sacrificed to keep the instrument hardware simpler (and less costly). Today, with the abundance of low-cost, powerful microprocessors, it is possible to design instruments with ease-of-use as a primary objective without adding significant cost or degrading performance or GPIB compatibility.

Tektronix GPIB instruments demonstrate this crucial difference between GPIB compatibility and capability. They are programmable instruments designed with easily understood commands and with standard data transfer conventions defined by Tektronix Codes and Formats. This can increase your productivity by improving the usability of your instruments.

Using the General Purpose Interface Bus is somewhat like using the telephone system. In both cases, a physical connection can be established between two locations and data can be transmitted—i.e., one person, or instrument, can talk to another. However, on the telephone system, unless both people speak and understand the same language, very little communication can take place. Beyond having a common language, they must also share the same vocabulary for real communication to take place. Similar problems can arise between different instruments not specifically designed to work with others. A common protocol, or "telephone manners," is important, as well—one person should not hang up before the other has finished speaking and has said "goodbye."

The IEEE-488 standard defines a "telephone system" describing how the physical communications system is to be used, but it does not define the "language" sent over the bus or the "manner" in which the physical communications system is to be used. This lack causes incompatibilities. For example, assume that a multimeter has made a measurement of +3.75 volts and now has to transmit this information over the GPIB to a computer. Eight data lines are available for sending information in a byte—serial fashion. The first question is: "What codes should be used to encode the five characters?" The 488 standard gives no recommendations here. The DMM designer is free to choose any code he wishes. If a BCD code is selected but the computer only understands ASCII, the multimeter and computer will be incompatible when connected, even though both devices are "standard."

Suppose the multimeter above does send ASCII code? The question now is, "What Format is the data to be in?" Do we send the character sequence +3.75 right to left, left to right, or some other way? Again, the IEEE-488 standard says nothing regarding the sequence of data bytes. The instrument designer is free to create incompatibility. Figure 2 illustrates three formats for the data from a DMM. For system products, it is important that designers use a common format. Thus, the Tektronix Codes and Formats Standard specifies that ASCII data be sent with the most significant data first, thus reducing the options for instrument designers, and making it easier for users to incorporate instruments into systems.

As more instruments incorporate microprocessors, increase in complexity, and gain intelligence, the potential for incompatibility will become more acute. What is needed is a standard to bring data compatibility to the GPIB in the same way that the IEEE-488 standard brought electrical compatibility to instrument interfaces. Tektronix has adopted such a standard—and has incorporated it in the design of all Tektronix GPIB instruments.

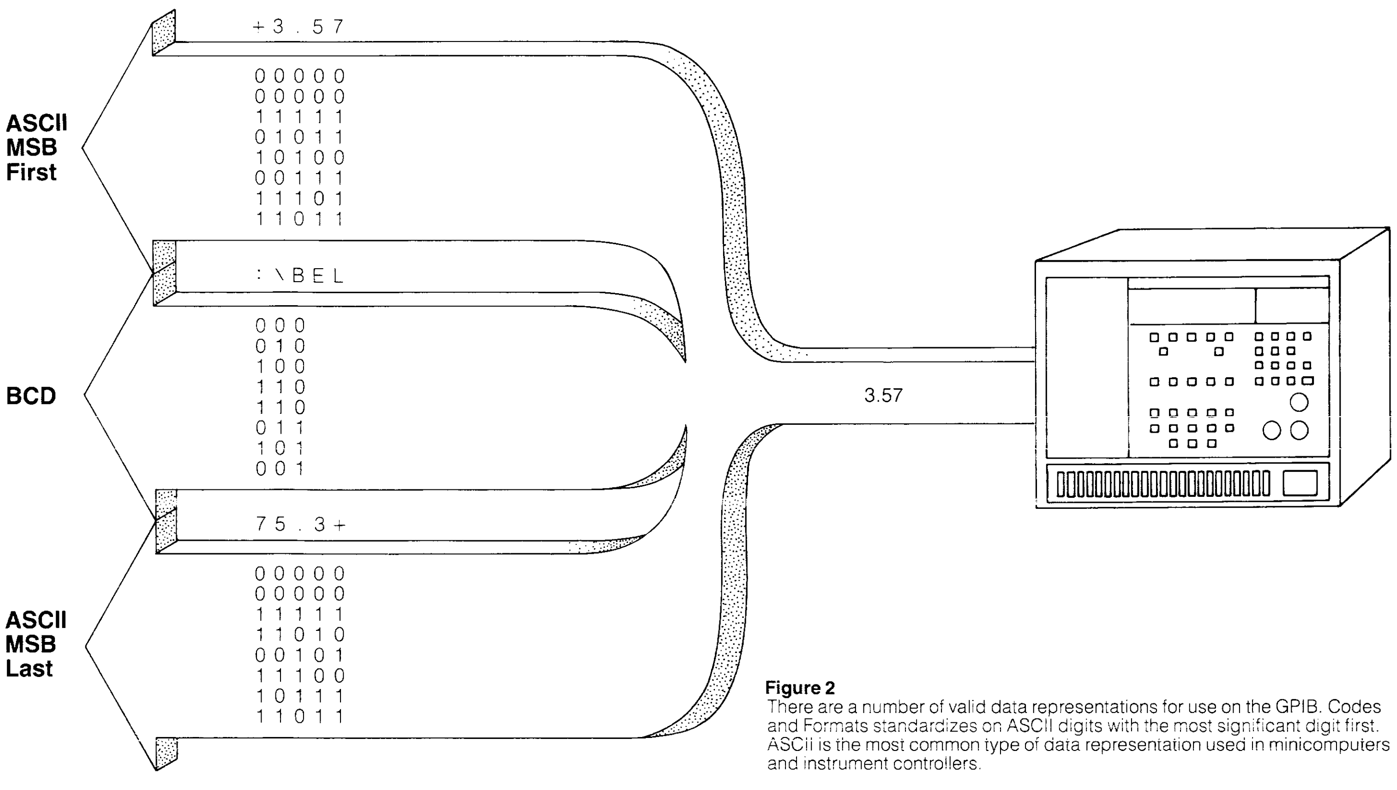


Figure 2
There are a number of valid data representations for use on the GPIB. Codes and Formats standardizes on ASCII digits with the most significant digit first. ASCII is the most common type of data representation used in minicomputers and instrument controllers.

The Tektronix Codes and Formats Standard

This standard:

- establishes a common message structure.
- describes communication elements and how they may be combined.
- defines control protocol.
- standardizes features that are particularly important to test, measurement and analysis systems.

System developers incorporating instruments which comply with the Codes and Formats standard as well as using easily understood commands will find that:

- writing the system software will be easier.
- with a defined language syntax, programs will be more efficient and self-documenting, making software maintenance easier.
- system change, or expansion, will require less software modification; and many support subroutines, such as error handling, won't need to be changed.

Also, Codes and Formats are important from the instrument's point of view. It's relatively easy to tell a person that a particular number format is to be used; most people intuitively understand what a number is.

A microprocessor in an instrument, however, has no such intuition and must be told—explicitly and unambiguously—how to send or how to receive numbers. Otherwise, it can send or receive something that is "obviously" wrong.

Because nearly all of today's GPIB instruments use ASCII-coded characters to send and receive data, Tektronix has chosen ASCII coding as standard.

In addition, nearly all instruments that send or receive numbers use the ANSI X3.42 standard format. This format states in effect that there are three types of numbers—integers, reals, and reals with exponents—and that they should be sent with the most significant character first. Table 1 shows examples of these formats. Unless there are numeric needs that cannot be met by this standard format, present and future Tektronix instruments will also use this format.

Note however, that while a number has been defined, its use has not. The Codes and Formats Standard has placed no restrictions upon the use of the number. It does not matter whether the number is sent from a multimeter, a counter, or a spectrum analyzer. In all cases, the syntax or structure of the number is identical. The semantics or meaning may be as different as VOLTS, PARTICLES, or CENTER FREQUENCY. Having this well-defined format for using a number allows instruments to send and receive numbers without confusion. Clearly, this is a step towards "language" compatibility.

As more instruments incorporate microprocessors, the functions they can perform will become increasingly complex. To have a computer or instrument controller interact with such intelligent devices requires Code and Format conventions more comprehensive than those that simply define numbers.

For instance, suppose that a device makes a group of measurements and is asked to report them. This requires that a group of numbers be sent. To separate one number from the next, a comma is used. For example, the position coordinates from a digitizer might be sent as .732, 1.52.

Furthermore, suppose that another device makes two different types of measurements and is required to report them—for example, FREQUENCY and PHASE. There should be a means for identifying each type. This is to be done by first sending a "header," that is a description of the number. If these headers and numbers are sent consecutively, they must be separated from each other. For this, a semicolon is used. For example: FREQ 1234; PHASE 56 or FREQ 1.234E03; PHASE 56.

These well-defined formats significantly enhance data communication compatibility over the GPIB.

TABLE I

NUMBER FORMATS (ANSI X3.42)

NR1	375 +8960 −328 +0000	Value of "0" must not contain a minus sign.
NR2	+12.589 1.37592 −00037.5 0.000	Radix point should be preceded by at least one digit. Value of "0" must not contain a minus sign.
NR3	−1.51E + 03 +51.2E − 07 +00.0E + 00	Value of "0" must contain a NR2 zero followed by a zero exponent.

The Human Interface

The shape of current and future technology requires people with a wide range of technical skill levels to be intimately involved in the instrument-to-instrument communication process. For example, if a GPIB-programmable power supply needs to be set to 20.0 volts, a person has to write this "need" into the controlling computer's program. The computer becomes an intermediary transmitting the person's intent over the bus to the power supply.

The power supply can be designed in one of two basic ways. The first is with minimal intelligence so that it can accept some "hieroglyphics"—which it in turn can conveniently interpret and execute. For example, some power supplies require the sequence 0 8 E 3 to put out 20 volts. Here the "0" stands for the 0 = 36 volt range, and the "8E3" is the ASCII representation of the hexadecimal commands required as shown in Figure 3.

On the other hand, the power supply can be designed with a microprocessor and intelligence to accept easily understood numbers. In this example, to put out 20 volts, the programmer simply sends the character sequence "VPOS 20." This second method of interacting is obviously a great deal more convenient for people, not only when the computer program is first written but also later, when someone other than the original programmer has to find out what the program is supposed to do. In the future, most devices will be "intelligent" and specifically be designed to interact with people. In order to promote this compatibility, the appropriate Codes and Formats are used.

Numbers in easily read formats are good for both computers and people. It is also necessary to send directions to instruments in a format other than numbers. For example:

TRIG EXT
CLIP ON
PEAK AUTO
FUNC SINE

In these situations, we can simply treat the first word as a header and the second word as a data type that is different from a number.

Other data, called arguments, are useful for various purposes:

String Arguments—for sending text to a display or printer.

Binary Block Arguments—for sending binary data blocks of known length.

Link Arguments—for sending certain types of instrument commands.

End Block—for sending binary data of unknown length or format.

The same general format can be used for all types of communications over the bus—commands to instruments, data from instruments, text to be displayed, and more. This message structure is summarized in Table II.

In basic terms, this is the Codes and Formats Standard adopted by Tektronix. It extends instrument compatibility from the realm of the electrical signals defined in IEEE 488 to the domain of data sent over the GPIB. It extends not only to computers and simple and smart instruments but also to people of widely divergent technical skill levels who have to work with these devices.

To represent the 20 Volts we want from the 36 Volt power supply let the calculator first compute

20/36 = V/4095
V = (20 * 4095)/36 = 2275

Then we need a subroutine to convert 2275 (base 10) to its hexadecimal equivalent (base 16).

The way the calculator will do this is to divide by 16 until the remainder is less than the base 16, each time converting the fractional part of the remainder into a whole number by multiplying the fraction by the base, 16.

2275/16 = 142.1875

To get an integer remainder you multiply:

.1875 * 16 = 3 (3rd character)
142/16 = 18.875

Multiply

.875 * 16 = 14 (2nd character)

8 is less than 16, therefore there is no further division and 8 is the 1st character.

In hex, you would write 8 14 3 as 8E3.

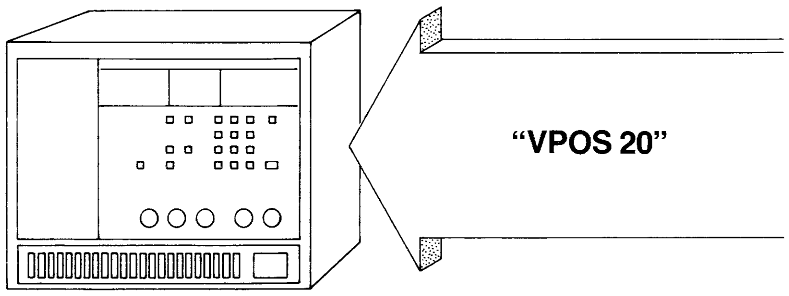


Figure 3
The use of microprocessors in the design of "intelligent" instruments allows instrument commands which are oriented to the user. This allows users to program instruments with understandable commands such as VPOS 20 to set a power supply to a positive 20 volts.

TABLE II

DEVICE-DEPENDENT MESSAGE STRUCTURE

A message represents a given amount of information whose beginning and end is defined. It is communicated between a device functioning as a talker and one or more devices functioning as listeners.

A message begins when the transmitting device is initially addressed to talk and the receiving device(s) is (are) addressed as listener(s).

A message is composed of one or more message units separated by message unit delimiters. A message unit delimiter is a semicolon.

The message ends when the talking device asserts EOI.

There are two message unit types:

1. Mixed Data Message Unit in two acceptable formats are:
   a. Header (in Character Argument format) followed by a space and optional arguments of any type separated by commas. Normally used for programming information.
   b. Non-character argument followed by optional arguments of any type separated by commas. Normally used for measurement data.
2. Query Message Unit consists of a character argument such as "SET," "ID," or "FREQ" followed by a question mark. This is normally used to interrogate a device for data or settings.

Following are definitions of the allowable argument types:

ARGUMENT TYPES AND EXAMPLES	DEFINITION	PURPOSE
Character Argument Trigger	One alphabetic ASCII character optionally followed by any number of ASCII characters excluding space, comma, semicolon, question mark, the control characters and rubout.	Used to transfer alphanumeric data such as message headers, labels, commands, etc.
Non-Character Arguments		
Number −12.3	A numeric value in any of the formats shown in Table 1.	Used to pass numeric values in an ASCII format.
String Argument "Remove Probe"	Opening delimiter (single or double quote) followed by a series of any ASCII characters except for the opening delimiter and a closing delimiter identical to the opening delimiter.	Provides a means for transmitting ASCII text to an output device.
Binary Block Argument $\%\ \underbrace{\text{2 bytes}}_{\text{16 bit binary value}}\ \overbrace{\text{wave-form values}}^{\text{Binary Data}}\ \underbrace{\text{8 bits}}_{\text{checksum}}$	"%" followed by a 2 byte (16 bit) binary integer specifying the number of data bytes plus a checksum byte which follows the data bytes. The checksum is two's complement of the modulo 256 sum of the preceding binary data bytes. This includes the two bytes comprising the 16 bit integer specification.	Used to transfer large arrays of numeric data such as waveforms in a binary format.
Link Argument NR.PT:1024	Character argument (label) followed by ":" and a value represented in any of the above argument types.	Used to attach a name or label to another argument.
End Block Argument @ABCDEFGHIJKL $\left\{\begin{matrix}E\\O\\I\end{matrix}\right\}$	"@" followed by a block of data with EOI set concurrent with the last data byte. End block can only be the last argument in a message and cannot be followed by a message unit delimiter.	Used when a block of data must be sent and neither the amount of data nor its format is known.

Message Conventions

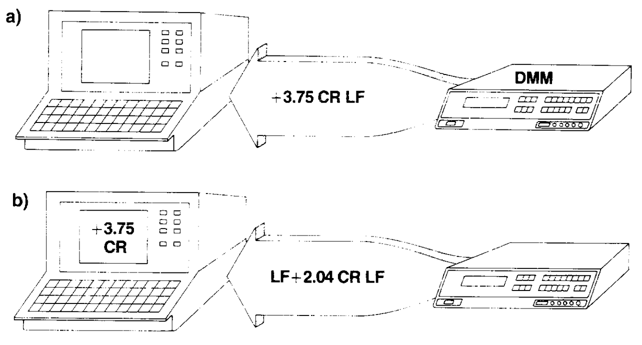
While standardizing this Codes and Formats "language" fosters greater compatibility between devices using the GPIB, it alone does not solve all compatibility problems. Well-defined operational conventions are also needed.

Conventions for using the GPIB are analogous to good manners when using the telephone: one party should not hang up before the other has finished speaking. The following is an example of the lack of good manners in two devices using the GPIB.

Suppose a computer has requested a voltage reading from a multimeter over the GPIB. The multimeter sends the number and terminates the transmission with the characters CR (carriage return) followed by LF (line feed) (Figure 4a). The computer, however, understands that CR by itself terminates a message. The computer "hangs up" after receiving the CR and leaves the LF character in the multimeter unsent (Figure 4b). The next time the computer asks for a multimeter reading, the multimeter sends LF, the character left over from the previous measurement, followed by the measured values (Figure 4c). The computer does not know what to make of a number preceded by a LF character. It stops and indicates an error. Although both devices are GPIB compatible, they do not work together because the IEEE 488 standard has not defined the conventions, "manners," for how the bus is to be used.

To avoid such incompatibilities, a standard way to terminate messages is needed. Two methods are commonly used. The first is to send some printer format characters such as CR or CR LF. The other is to assert the EOI line when the last data byte of a message is sent. The first method was adequate for simple instruments that sent or received only ASCII coded numbers. However, today's more intelligent devices have to send messages representing digitized waveforms or programs for a microprocessor in an instrument. Some of these messages may contain binary data to reduce transfer time. A certain sequence of binary coded bytes (00001101, 00001010), which if interpreted as ASCII, will appear to be a CR LF, and thus be misinterpreted (Figure 5). The second termination method has no such problems. Asserting the EOI line, unambiguously terminates the message.

Thus, the Tektronix Codes and Formats Standard states that instruments sending messages should terminate them by asserting the EOI line concurrently with the last byte of the message (Figure 6).



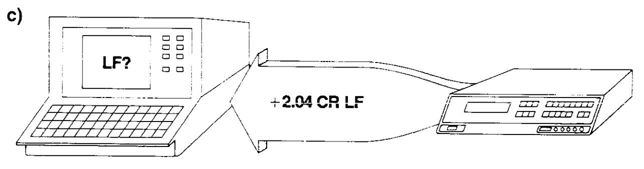
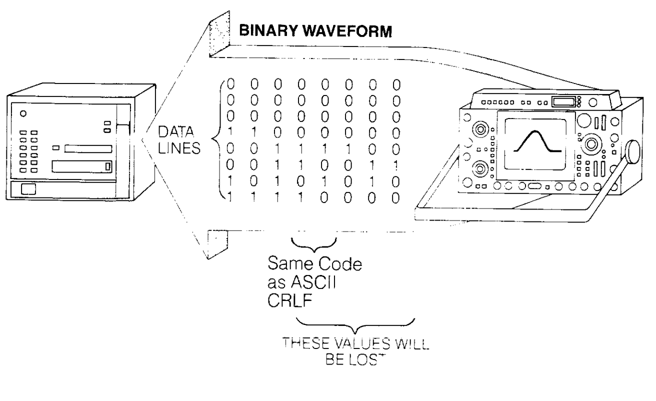


Figure 4
Without a standard message termination, the user must take care that each listener understands the same message termination convention. In this example, (a) the talking instrument uses CR LF as the message terminator, (b) while the listening controller understands CR as the message terminator. (c) When the controller accepts the next message, the first character encountered is the LF left from the last message which the controller may interrupt as an illegal character since it is expecting a numeric value.



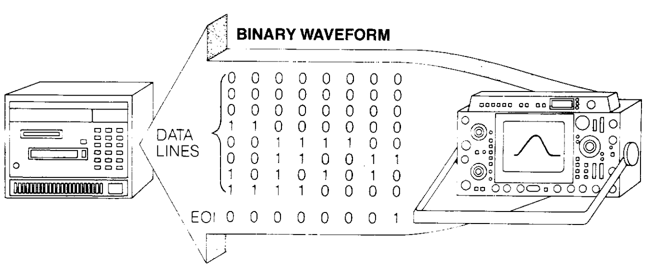


Figure 5
When large amounts of data (e.g., a waveform) are to be transferred over the GPIB, it is often more efficient to code this data in binary rather than ASCII. This can provide more than a 2:1 reduction in the number of bytes to be transferred. However, if CR LF is used by the listener as a terminator, and the binary data coincidentally has the binary equivalents of these characters, the listener will stop listening in the middle of the transmission. Thus, EOI is a better choice for an unambiguous message termination convention.

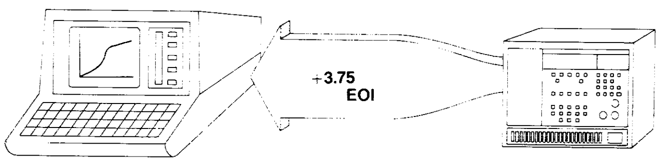


Figure 6
Setting the EOI line concurrent with the last byte of a message provides an unambiguous message terminating convention.

Other problems can be created by instruments which execute each individual command as received. For example, suppose a programmable high-voltage power supply that can be set as high as 1000V has been set to put out 10V and limit current at 2A. Then it is sent the message "VOLTS 1000, CURR 10E – 06," that is, 1000 volts output limited at 10 MA. Lacking good message-handling conventions, the supply goes to 1000V output immediately upon receiving the first part of the message. But because the current limit is still 2A, the value from the previous setting, the supply either crowbars or damages the equipment connected to it (Figure 7). For proper operation, the programmer should have changed the current setting first. Only then should the voltage be changed. It is much easier and safer using a power supply which does not execute any command until the entire message is received and terminated by asserting the EOI line (Figure 8).

This same convention can also prevent misunderstandings between a computer and a measurement instrument. When instructed to send a measurement message, the instrument sends EOI only when the message is complete, and no more data is sent unless directed by the computer to do so. This way the computer knows that no data is lost; the instrument is not inadvertently stopped from talking in the middle of a message.

In short, Tektronix Codes and Formats standard defines a message to be a complete block of information. It begins when a device starts sending data and ends when EOI is sent or received concurrently with the last data byte.

It should be noted, too, that the beginning of a message will need further clarification. An instrument sending a message may be interrupted by the computer taking control, perhaps conducting a serial poll. When the instrument becomes a talker again, it should resume sending the message. Thus, a message begins when a device enters the talker active state for the first time following a reset or a previously sent EOI.

There is a further refinement to the message convention. When a device is made a talker, it should always say something. If it has nothing to say and will have nothing to say for an indefinite period of time, it should send a byte of all ones concurrent with EOI. This lets the listening device know that no meaningful data is forthcoming. Thus, the "talked with nothing to say" byte is a null message. This convention prevents tying up the GPIB while the computer waits for a device to talk that will never send a message.

There are other conventions associated with messages that make life on the bus easier. A listening device should always handshake. It should not stop handshaking just because it does not understand or cannot execute a particular message. After EOI is received, if the listening device is confused, it should send out a service request and, on a serial poll, notify the controller that nonsense has been received. *Under no circumstances should a device execute a message it does not understand.* Some non-Tektronix devices do not follow this convention—with disastrous results. A particular power supply can be sent four letter O's instead of four zeros, a common human mistake, and this supply will put out its maximum voltage instead of the intended zero volts.

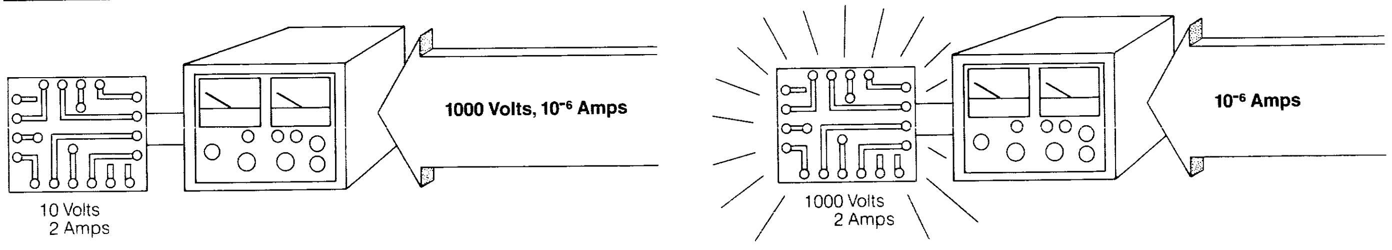


Figure 7
Instruments should not begin execution of any part of a message until the entire message has been accepted. In this example, the settings of a general purpose power supply are to be changed from 10 volts, 2 amps to 1000 volts, 1 milliamp. If the voltage command is executed prior to changing the current setting, the results could be disastrous.

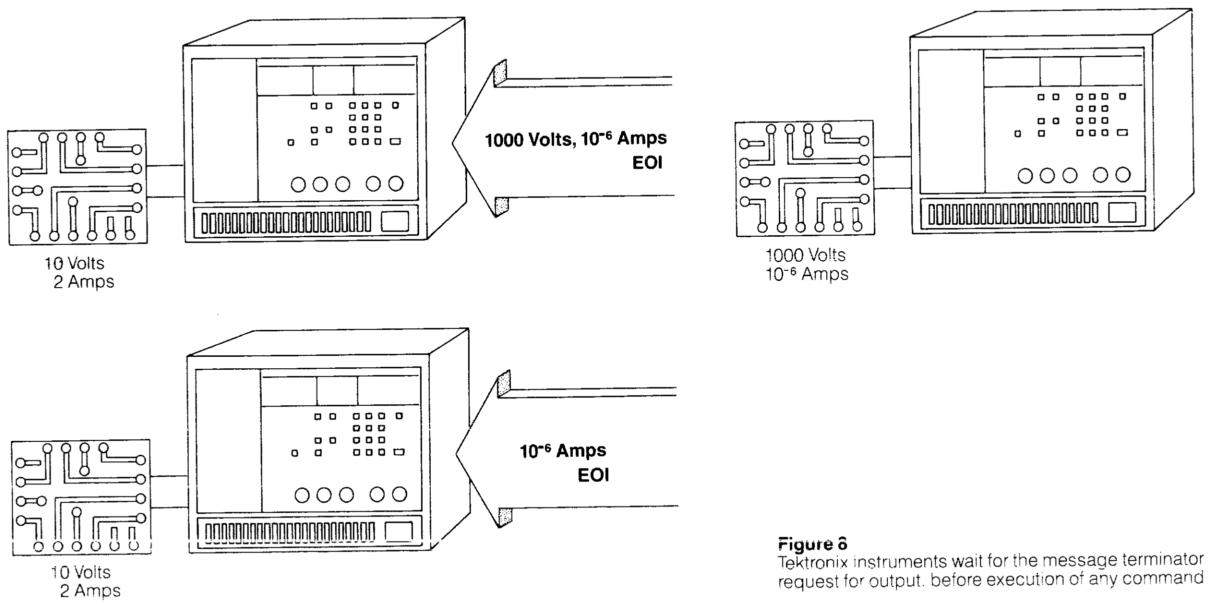


Figure 8
Tektronix instruments wait for the message terminator (EOI), or optionally a request for output, before execution of any command in the sequence.

Status Bytes

The IEEE 488 standard defines a facility for an instrument to send a byte of status data to the computer, but, except for bit 7, the standard does not define the meaning of the bits. The IEEE 488 standard assigns bit 7 to mean that a device is, or is not, requesting service. Thus, bit 7 cannot be used for other purposes.

However, there is a common need for instruments to report certain kinds of status or errors to the computer. So a status byte convention is established for this. One common need is for instruments to report if they are busy or ready. Bit 5 is used for this purpose. Another common need is for instruments to report if they are encountering abnormal conditions. Bit 6 is selected for this.

There are more complex conditions besides busy/ready or normal/abnormal. These are listed in Table III. While these status bytes are generally useful for most purposes, certain instruments may have conditions that are peculiar to them. To report these status states, bit 8 is used to indicate that the status byte is not the common type but particular to an instrument.

Providing a standard coding for the status byte enhances the convenience to the person programming the system. If all the instruments have common status byte codings, then a common status byte handling routine is written for all instruments, not a separate one for each.

TABLE III STATUS-BYTE DEFINITIONS

		Decimal	
Abnormal Conditions	**Binary**	**X = 0**	**X = 1**
ERR query requested	011X 0000	96	112
Command error	011X 0001	97	113
Execution error	011X 0010	98	114
Internal error	011X 0011	99	115
Power fail	011X 0100	100	116
Execution error warning	011X 0101	101	117
Internal error warning	011X 0110	102	118
Normal Conditions			
No status to report out of the ordinary	000X 0000	0	16
SRQ query request	010X 0000	64	80
Power on	010X 0001	65	81
Operation complete	010X 0010	66	82

Definition of the system status Bytes.

ERR query request—This states that a device is reporting an error but is not identifying the error. The controller should send ERR? to find out about the error.

Command error—Results when a message can't be parsed or lexically analyzed.

Execution error—Exists when a message is parsed and analyzed but can't be executed (e.g., setting a device out of its range).

Internal error—Results from an out of calibration error or a malfunction in the device (may be discovered by an instrument self-check).

Power fail—Some instruments can detect a power fail long before the instrument is affected. In those cases a power fail is sent to the controller to save important information or to flag suspect operation.

Execution error warning—The device indicates that it has received and is executing a command but that a potential problem might exist. For example, an instrument may be out of range but is sending a reading anyway.

Internal error warning—The device indicates that it has an internal error but is continuing to function.

No status to report out of the ordinary—When the controller does a serial poll and a device has nothing to say, this byte is reported.

SRQ query request—This is similar to the ERR query request. It simply says "I am requesting service but am not saying why." To find out why, the controller should send SRQ?.

Power on—When a device has finished its "power on" sequence, it may send a "power on" service request. This will let the controller know that a device has just come up on the bus. It should be noted that on a "power on" service request, RSV can not be cleared by a device clear. The only thing that can clear the RSV is handshaking out the status byte to the controller. This eliminates the problem of a device powering up and receiving a device clear before the controller knows it is on the bus. After every power on, this status byte should be the first reported.

Operation complete—This tells the controller some task has been completed. This is not the same thing as the BUSY bit, DIO5, described earlier. The BUSY bit going false indicates that processing and execution of a command has been completed. In a multi-task environment, the execution of a command may cause a long task to start, e.g., data log 10,000 points. At this point the device is ready for another command. When the task is done, the "operation complete" status byte is sent to the controller.

Queries

Even with all the possibilities allowed by status bytes, it is often necessary to send more detailed information from an instrument to a computer. This can be done via "queries."

Normally a measurement instrument sends a measured value when it functions as a talker. To elicit something other than this default message, the computer can first send a query to the instrument. Then, when the instrument is made a talker, it will send the desired information. Queries take the form of a Header followed by a question mark. An example is shown in Figure 9. Here are some queries and their uses:

ERR? is used for investigating detailed error conditions in an instrument. The response an instrument sends back is ERR followed by NR1 numbers that code the particular problem.

SET? requests an instrument to send the computer its present settings and other current state information. Sending this information back to the instrument at a later time returns the instrument to the state it was in when it received SET? This query makes it possible to develop a program using an instrument's front panel as an input to the computer. Using this feature, a programmer never needs to know the instrument's GPIB commands.

ID? makes an instrument identify itself by sending information as instrument type, model number, firmware version, etc. This feature is useful for identifying a particular device in the field and potentially for self-configuring systems.

Defining a standard way to elicit responses from an instrument enhances the convenience to the programmer. When all instruments use the same form to perform similar functions, the programmer has to learn only one convention, not many.

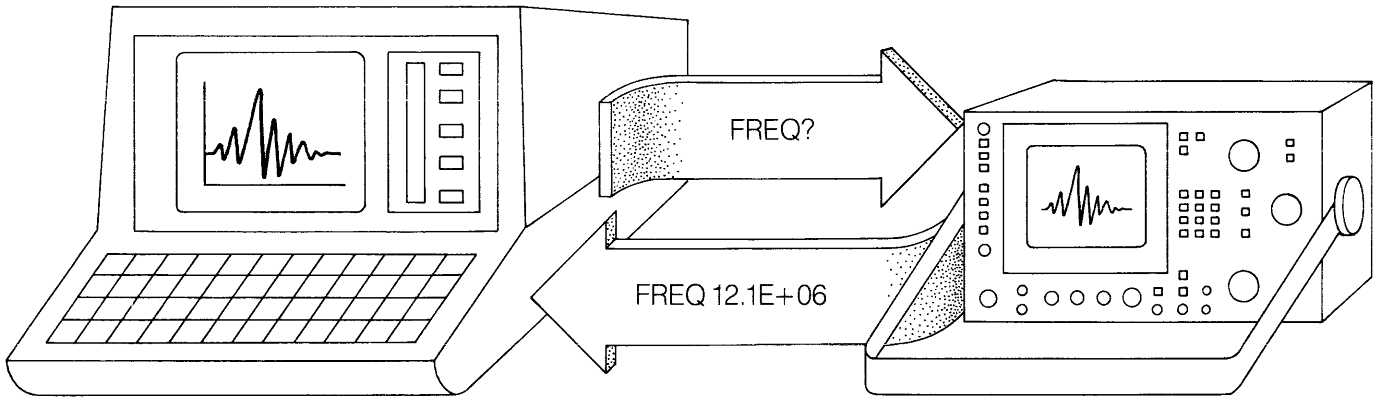


Figure 9
The use of easy-to-remember queries is an important feature of Tektronix GPIB instruments. Many query commands are formed by adding a question mark to the mnemonic for the setting to be queried.

Capabilities and Complexities

Instruments that incorporate queries are more compatible and can be more useful in systems. As instruments get more complex, and in many ways become computers in their own right, more operational conventions are needed. These conventions are more characteristic of communication between a computer and its peripheral processors than what we see today between small desktop calculators and simpleminded instruments. But such conventions need not make intelligent instruments more difficult to use. Quite the contrary, an intelligent instrument should be easier to use if its intelligence is used properly. Here are some examples of conventions adopted by Tektronix to enhance both computer/instrument compatibility and human/system compatibility.

1. While an instrument should always send numbers in the correct format described earlier, it should receive numbers "forgivingly." Specifically:
   a. Negative zero numbers should never be sent, but they should be accepted.
   b. Any number in scientific notation should be sent exactly as defined in ANSI X3.42 standard, i.e., with the decimal point included. Some of today's computers violate this standard and send values without the decimal pont.

      This "illegal" number should be received with an implied decimal point following the least significant digit.
   c. If an instrument receives a number whose precision is greater than the instrument can handle internally, then the number should be rounded off, not truncated. This enhances instrument accuracy.
2. An instrument should recognize both spaces and commas as argument delimiters. Multiple spaces or commas should not be construed as delimiters for null arguments. This convention is useful because some computers gratuitously generate spaces and send them on the GPIB.
3. An instrument should receive headers and character arguments in both upper and lower case and equate them. a=A, b=B, etc. This is useful because some desktop instrument controllers have a problem sending upper-case alpha characters.
4. An instrument sending data about its front panel should use headers and character arguments that correspond to the front panel's nomenclature.

These features make Tektronix instruments "friendly" to a casual or inexperienced programmer and compatible with most computers. There are still other features that are built into Tektronix intelligent instruments. Here are two examples:

The Service Request (SR) function and corresponding status byte reporting are very important. They can alert the controller of programmable instruments to new events or possible malfunctions (Figure 10). Thus, they let the computer controlling an unattended instrument system manage the system effectively or call for help when it can't. Sometimes, however, a person or a computer does not want to be interrupted—for example, when a sensitive or time-dependent measurement is being made. For these cases a message, RQS OFF, can be sent to disable any service requests. To turn the service request capability back on, the message RQS ON is sent.

Another useful convention is related to the Device Trigger (DT) function. Sometimes a command message sent to an instrument should be executed immediately. At other times, the command message should only set up the instrument, and the desired action should be executed when the Group Execute Trigger interface message is sent. To make the instrument execute commands immediately, the message DT OFF is sent. To make the instrument defer execution of commands, the message DT ON is sent.

These and other features of intelligent instruments make them both compatible with computers and friendly to people. Compatibility and friendliness are really the same thing. Given a powerful enough computer, and a clever programmer, most of today's devices that use the IEEE 488 bus can be made to do whatever they were designed to do: compatibility can be forced. Without well-defined codes and formats and without operational conventions and easily understood commands, instruments appear incompatible or unfriendly. With codes and formats and with known operational conventions, devices using the GPIB become friendly as well as compatible, thereby allowing the user to spend more time on the task at hand, rather than figuring out how to make the system work —increased productivity with Tektronix GPIB instruments.

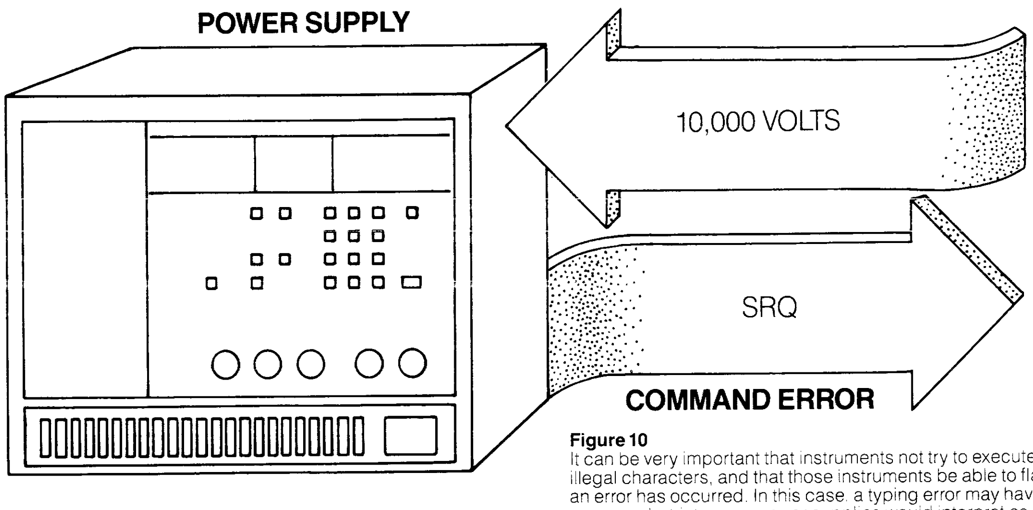


Figure 10
It can be very important that instruments not try to execute commands with illegal characters, and that those instruments be able to flag the controller that an error has occurred. In this case, a typing error may have created an illegal command which some power supplies would interpret as their maximum setting. This should not be allowed to happen!

For further information, contact:

U.S.A., Asia, Australia, Central & South America, Japan
Tektronix, Inc.
P.O. Box 4828
Portland, OR 97208
Phone: 800/547-6711
Oregon only 800/452-6773
Telex: 910-467-8708
Cable: TEKTRONIX

Europe, Africa, Middle East
Tektronix International, Inc.
European Marketing Centre
Postbox 827
1180 AV Amstelveen
The Netherlands
Telex: 18312

Canada
Tektronix Canada Inc.
P.O. Box 6500
Barrie. Ontario L4M 4V3
Phone: 705/737-2700

Tektronix sales and service offices around the world:
Argentina, Australia, Austria, Belgium, Bolivia, Brazil, Canada, Chile, Colombia, Costa Rica, Denmark. East Africa, Ecuador, Egypt, El Salvador, Federal Republic of Germany, Finland, France, Greece, Hong Kong, Iceland, India, Indonesia, Iraq, Israel, Italy, Ivory Coast, Japan, Jordan, Korea, Kuwait, Lebanon, Malaysia, Mexico, Morocco, The Netherlands, New Zealand, Norway, Pakistan. Panama. Peru, Philippines, Portugal, Republic of South Africa, Saudi Arabia, Singapore, Spain, Sri Lanka, Sudan, Surinam, Sweden, Switzerland, Syria, Taiwan, Thailand, Turkey, Tunisia, United Kingdom, Uruguay, Venezuela, Zambia.

Tektronix®
COMMITTED TO EXCELLENCE

99AX-4607 5/81