Tektronix®
COMMITTED TO EXCELLENCE

Please Check for
CHANGE INFORMATION
at the Rear of this Manual

4050 SERIES

R14 GPIB ENHANCEMENT ROM PACK

INSTRUCTION MANUAL

Tektronix, Inc.
P.O. Box 500
Beaverton, Oregon 97077

MANUAL PART NO. 070-4316-01
PRODUCT GROUP 14

First Printing MAR 1982
Revised JAN 1983

WARNING

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

This instrument, in whole or in part, may be protected by one or more U.S. or foreign patents or patent applications. Information provided on request by Tektronix, Inc., P.O. Box 500, Beaverton, Oregon 97077.

MANUAL REVISION STATUS

PRODUCTS: 4051R14, 4052R14, and 4052R14 Option 1A GPIB Enhancements ROM Pack.

This manual supports the following versions of these products: Serial Numbers B010100 and up.

	DESCRIPTION
MAR 1982	Original Issue
JUL 1982	Revised pages: 2-60 and 2-63.
JAN 1983	Revised: Information added for 4052R14 Option 1A.

CONTENTS

Section 2 (cont) **Page**

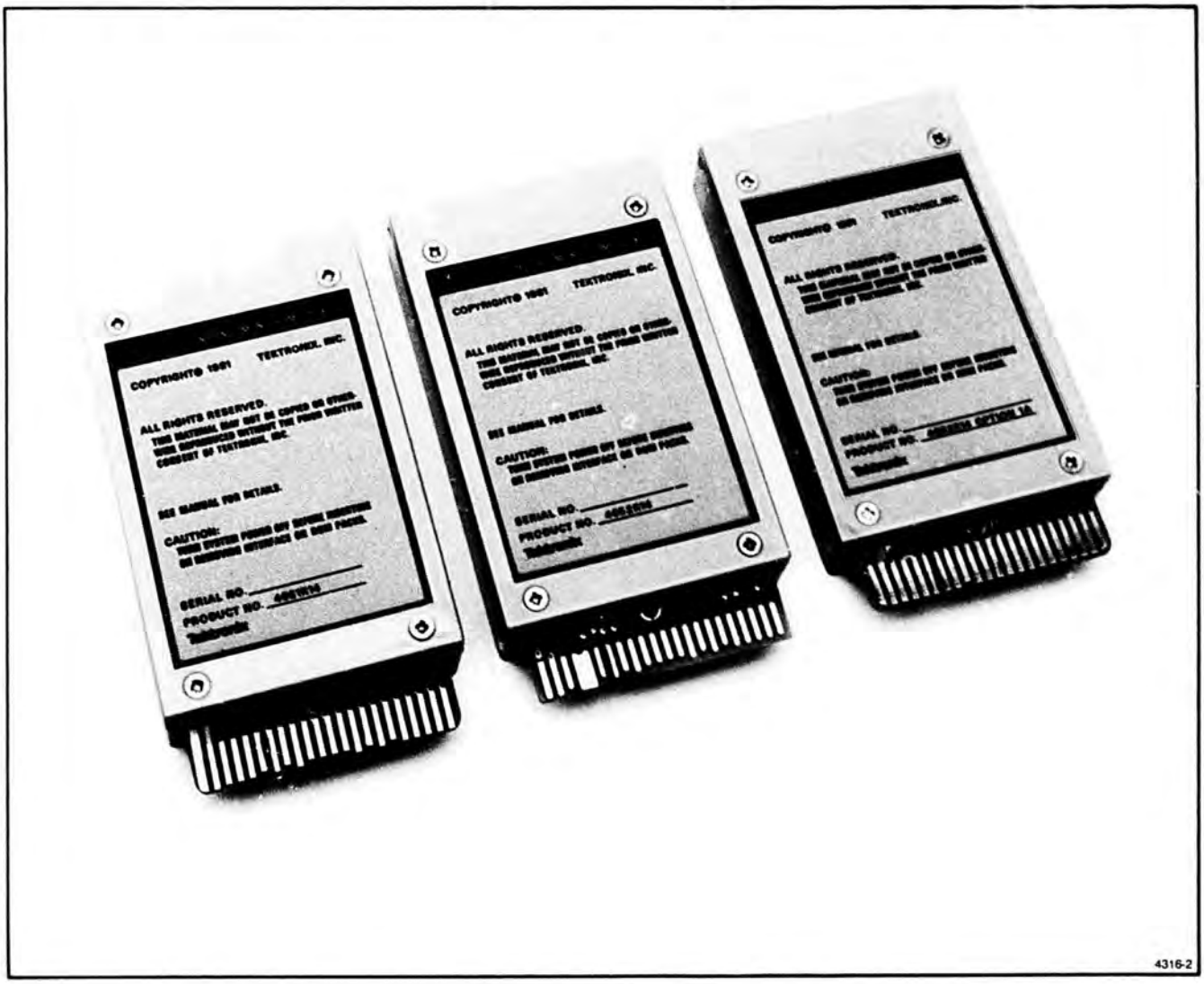


Figure 1-1. 4051R14, 4052R14, and 4052R14 Option 1A GPIB Enhancement ROM Packs.

Section 1

GENERAL DESCRIPTION

INTRODUCTION

The Tektronix 4051R14, 4052R14, and 4052R14 Option 1A GPIB Enhancement ROM Packs (Figure 1-1) are Read Only Memory (ROM) devices to be used with 4050 Series Graphic Computing Systems.

- The 4051R14 ROM Pack is used ***only*** with the 4051 Graphic Computing System.
- The 4052R14 ROM Pack is used ***only*** with either the 4052 or 4054 Graphic Computing Systems.
- The 4052R14 Option 1A ROM Pack is used ***only*** with either the 4052A or 4054A Graphic Computing Systems.

These ROM packs provide routines which are accessed with the 4050 Series BASIC 'CALL' statement.

The routines are designed to enhance the Graphic Computing System's performance as an instrument controller for instrumentation systems using the IEEE Standard 488-1978 Standard Digital Interface for Programmable Instrumentation.[1]

The 4050 Series R14 routines can be functionally divided into three groups:

- Routines which improve the GPIB serial poll and add parallel poll capabilities.
- Routines which issue standard GPIB interface messages (GTL, DCL, IFC, etc.).
- Additional general purpose routines which provide for input/output of binary data, number base conversion, bit comparison, and other functions useful in a GPIB programming environment.

[1] **Published by The Institute of Electrical and Electronics Engineers, Inc., 345 East 47th Street, New York, NY 10017. Tektronix' term for this interface is General Purpose Interface Bus or GPIB.**

This manual provides the following information:

- Section 1 contains the ROM pack specifications and installation instructions.
- Section 2 contains an overview of the 4050 Series R14 routines, some technical details on parameter specification, and a detailed description of each routine. Routines are listed alphabetically.
- Section 3 contains the parts list for each ROM pack.
- Section 4 contains the schematic and exploded view diagram for each ROM pack.
- Appendix A includes several sample programs which illustrate the use of the various routines. These programs may also be used to verify that the ROM pack routines are functioning correctly.
- Appendix B contains a summary of the ROM pack routines, listed alphabetically.
- Appendix C contains a summary of SRQ interrupt set-ups.
- Appendix D is an overview of error handling using ERRLIST, ASKERR, CLRREP, and RETRY.

SERVICE INFORMATION

This instruction manual contains parts lists and schematics for the ROM packs but does not contain service procedures. If necessary, refer to:

- *4051 Service Manual Vol. 1* for the 4051R14 ROM Pack.
- *4052/4054 and 4052A/4054A Technical Data Service Manual* for the 4052R14 and 4052R14 Option 1A ROM Packs.

These manuals contain a general description of ROM pack circuitry. The ROM packs do not require any adjustments or preventive maintenance.

SPECIFICATIONS

The terms "performance requirement" and "supplemental information" are used as follows to describe ROM pack specifications:

Performance Requirement: A statement that defines a characteristic in quantitative terms of performance, usually in limit form.

Supplemental Information: Statements that amplify or supplement performance requirements or that provide typical performance information.

Electrical Characteristics

Power requirements by the 4051R14, 4052R14, and 4052R14 Option 1A ROM packs from the Graphic System power supplies are as follows:

Characteristics	Performance Requirement
Input voltage	+5V
Maximum current (normal) 4051	
Operating	250 mA
Non-Operating (ROM pack inserted but no routines being executed)	150 mA
Maximum current (normal) 4052/4054, 4052A/4054A	
Operating	180 mA
Non-Operating (ROM pack inserted but no routines being executed)	90 mA

Environmental Characteristics

The 4051R14, 4052R14, and 4052R14 Option 1A GPIB Enhancement ROM Packs meet all environmental specifications of the 4050 Series Graphic Computing System.

Characteristics	Performance Requirements
Temperature	
Operating	+10°C to +40°C (+50°F to +104°F)
Non-Operating	−40°C to +65°C (−40°F to +149°F)
Altitude	
Operating	4,572 m (15,000 ft) maximum
Non-Operating	15,240 m (50,000 ft) maximum
Humidity	
Operating	0% to 80% non-condensing
Non-Operating	0% to 95% non-condensing

Physical Characteristics

Characteristics	Supplemental Information
Length (nominal)	119 mm (4.7 in) including edge-board connector
Width (nominal)	66.5 mm (2.62 in)
Depth (nominal)	22.2 mm (0.875 in)
Weight (nominal)	about 230 gm (8 oz)

REFERENCE DOCUMENTATION

Articles

"Codes and Formats Standard Adds Compatibility and Capability to IEEE-488 Instruments." *Tekscope*, Vol. 13, No. 3, Sept. 1981.

Manuals/Booklets

All GPIB communications must conform to the IEEE-488 standard:

IEEE Standard Digital Interface for Programmable Instrumentation (ANSI/IEEE Std 488-1978). The Institute of Electrical and Electronics Engineers, Inc., 345 East 47th Street, New York, NY 10017.

In addition, a complete list of relevant Tektronix documentation can be found in the *4050 Series Graphic Computing System Operator's Manual*.

INSTALLING THE ROM PACK

1. Be sure the Graphic System power switch is off.

CAUTION

Inserting the ROM pack while the power is ON may cause damage to the ROM pack. The Graphic System memory contents may also be lost.

2. Insert the GPIB Enhancement ROM pack into one of the slots in the rear of the Graphic System as shown in Figure 1-2. Press down and gently rock the plastic case until the ROM pack edge connector is firmly seated in the backpack.

3. Turn the power switch ON and wait a few seconds for the system to power-up.

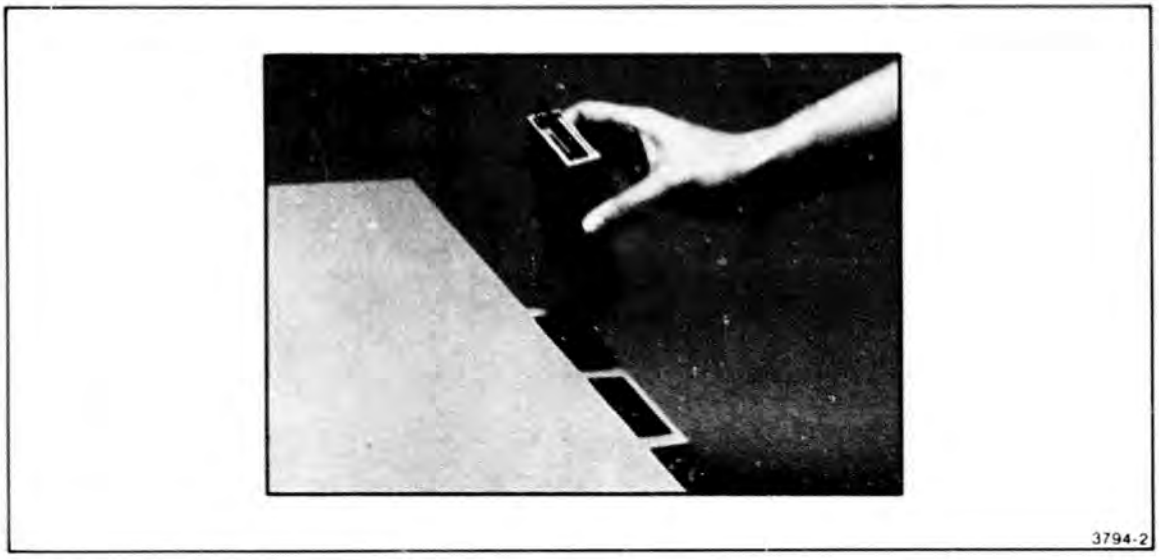


Figure 1-2. ROM Pack Installation.

Section 2

COMMAND DESCRIPTIONS

INTRODUCTION

This section contains an overview of the 4050 Series R14 routines by functional groupings, as well as a detailed description of each routine. These explanations assume the reader is familiar with the operation of Tektronix 4050 Series Graphic Computing Systems.

The explanations include basic information about the use of the commands and, where necessary, some background information about the algorithms used. Examples and sample programs are included to help use the commands.

This section also includes syntax rules for address arrays and address lists.

Sample programs which exercise and verify the ROM pack routines are included in Appendix A. A Routine Summary is provided in Appendix B for quick reference.

Notation and Terminology

In this manual, hexadecimal numbers are preceded with an upper case X and enclosed in quotes. For example:

X'FF3A' means FF3A in hexadecimal.

When compared with hexadecimal, decimal numbers are identified by the prefix "DEC"; for example:

DEC 23 means a decimal value of 23.

The term "error code" refers to a Tektronix Codes and Formats error code. Reference "Codes and Formats Standard Adds Compatibility and Capability to IEEE-488 Instruments," *Tekscope*, Vol. 13, No. 3, Sept 1981.

OVERVIEW OF THE 4050R14 ROM PACK ROUTINES

General Summary

There are 48 routines total between the three ROM packs and most of the routines are common to all three. But there are some differences:

- The following routines are in the 4052R14 Option 1A ROM Pack only: ERRLIST, ASKERR, CLRREP, RETRY, TAPEIN, TAPEAPP, TRIM.
- The following routines have a slightly different version implemented in the 4052R14 Option 1A ROM Pack: SRQON, SRQOFF, NEWTAPE, VLIST, LAST, THEADER, TNAME, BININ, BINOUT
- The routines MTPACK and WAIT are implemented in the 4051R14 ROM Pack only; the 4051 Graphic Computing System does not provide these functions internally. However, they are implemented internally in the 4052/4054 and 4052A/4054A Graphic Computing Systems and so were not implemented in the 4052R14 or 4052R14 Option 1A ROM Packs.

GPIB Polling Routines

There are ten routines designed to improve the Graphic System's GPIB polling capabilities.

- Two routines give the user direct control over SRQ interrupts:

  SRQOFF — Prevents SRQ from generating an interrupt.

  SRQON — Enables SRQ to generate an interrupt.

  SRQOFF is automatically executed at power on. This allows the Graphic System to initialize a GPIB applications program without having to respond to SRQ from a system of instruments connected to the GPIB.

- Two routines enhance the serial poll processing capability:

  CONFIG — Determines the active devices on the bus and returns the device addresses in an array.

  POLL — Performs a serial poll on the specified devices. This POLL routine differs from the 4050 Series 'POLL' routine in that it can time out if no device responds to the poll.

- Four routines implement the parallel poll function. Each routine requires the IEEE-488 function subsets shown in brackets at the right:

PPD	— Unconfigures selected instruments.	[PP1 req'd]
PPE	— Configures instruments for parallel poll.	[PP1 req'd]
PPOLL	— Performs a parallel poll.	[PP1 or PP2 req'd]
PPU	— Unconfigures all instruments	[PP1 req'd]

- Two routines decode the standard instrument error codes according to the Tektronix Codes and Formats standard.

  ERRHLP — Returns general information about Codes and Formats event or error codes.

  STBHLP — Returns general information about Codes and Formats status byte definitions.

Information About Variables

ARSIZE — Returns the currently dimensioned size of an array.

UNDEF — Used to determine whether a variable is defined.

GPIB Mnemonics Routines

These twelve routines allow the user to issue standard GPIB interface messages through "CALL" statements using the corresponding IEEE-488 mnemonics. They also provide additional capabilities not easily available through the "WBTYE" command. The IEEE-488 function subsets required for an instrument to respond to each routine are also listed.

Routine Name	Routine Description	IEEE-488 Function Subsets Required
DCL	Device Clear	DC1 or DC2
GET	Group Execute Trigger	DT1
GTL	Go To Local	RL1 or RL2
IFC	Interface Clear	T1 — T8, TE1 — TE8, L1 — L8 or LE1 — LE8
LISTEN	Makes one or more instruments into listeners.	L1 — L4 or LE1 — LE4
LLO	Local Lockout	RL1
LOCS	Puts all devices in the local state (LOCS).	RL1 or RL2
RWLS	Puts specified instruments in RWLS (remote with lockout state).	RL1
SDC	Selected Device Clear	DC1
TALK	Makes an instrument a talker.	T1 — T8 or TE1 — TE8
UNL	Sends the UNListen message.	L1 — L4 or LE1 — LE4
UNT	Sends the UNTalk message.	T1 — T8 or TE1 — TE8

Error Handling

The following four 4052R14 Option 1A routines enhance error handling in 4052A/4054A Graphic Computing Systems:

ERRLIST	Enables specified errors to be trapped and handled by a user's error handling routine.
ASKERR	Determines which error was trapped.
CLRREP	Sets error repetition counter to zero.
RETRY	Returns program control from an error handler back to the line in which the error occured.

String Input

The following three 4052R14 Option 1A routines aid in the duplication of ASCII data or program files on 4052A/4054A Graphic Computing System's internal mag-tape units:

TAPEIN	Reads and stores an entire ASCII file from the internal mag-tape unit into a character string variable.
TAPEAPP	Reads and appends an entire ASCII file from the internal mag-tape unit to a defined character string in a string variable.
TRIM	Changes the logical length of a defined string in a string variable.

General Purpose Routines

These seventeen general purpose routines provide additional capabilities valuable in a GPIB programming environment, including binary input/output of data, number conversion, and file management routines. The routines are:

BININ	Inputs data in binary block format.
BINOUT	Outputs data in binary block format.

DECHEX	Converts a decimal number to ASCII HEX.
HEXDEC	Converts an ASCII HEX number to decimal.
LAST	Finds the last file. Optionally returns the file number or header.
MTPACK'	Winds and rewinds the tape to correct poor tape spooling that may occur with some cartridges.
NEWTAP	Finds the present file location and returns the tape cartridge status.
PRISTR	Prints a string to a device on the IEEE-488 interface without EOI being transmitted with the last byte and with no carriage return following the string.
RBIN	Inputs binary data.
THEADER	Finds and opens the file where the tape head is currently positioned. Returns the file header in a string or prints it on the screen.
TNAME	Assigns a name to a tape file.
VARCLR	Clears specified bits in a variable.
VARSET	Sets specified bits in a variable.
VARTST	Tests specified bits in a variable.
VLIST	Lists all current BASIC variables and their values on the screen.
WAIT'	Waits a specified amount of time.
WBIN	Outputs binary data.

This routine is implemented in the 4051R14 ROM Pack only. The routine is implemented internally in the 4052 and 4054 Graphic Systems.

ROUTINE SYNTAX

The following symbols are used to define the routine syntax and address syntax in this manual:

Symbol	Meaning
< >	Defined element.
: =	"Is defined as."
[ ]	Optional. The parameter may be omitted.
{ }	Choose one from the given list.
|	Exclusive OR
...	Repeat as needed.

The quotation marks around the routine name and the commas and semicolons within the routine syntax must be entered. For example, the PRISTR syntax is:

$$\text{[line number] CALL "PRISTR", } \left\{ \begin{array}{l} \text{<string constant>} \\ \text{<string variable>} \end{array} \right\} \text{ [; address list]}$$

An example call showing the actual characters entered in the program statement is:

```
250 CALL "PRISTR", A$; 3, 2; 5; 7
```

SYNTAX RULES FOR ADDRESS ARRAYS

The ROM pack routines allow the use of arrays to store primary or primary/secondary addresses for communications with devices on the GPIB. The following conventions govern the use of address arrays:

- One-dimensional arrays contain only primary addresses; in such an array, A(i) is the primary address of device i. Such arrays may be used for addressing devices that implement the IEEE-488 TALKER (T) or LISTENER (L) functions.

- Two-dimensional arrays may be used for addressing devices implementing the IEEE-488 TALKER (T), LISTENER (L), EXTENDED TALKER (TE), or EXTENDED LISTENER (LE) functions. In such arrays, the first column is used to store the primary address, and the second column is used to store the secondary address. That is, element A(i,1) is the primary address of device i, and A(i,2) is the secondary address. All other columns of the array are ignored.

- The values in the address array should be in the range 1 through 30 for primary addresses, and in the range 0 through 31 for secondary addresses.
  - Primary address 0 is reserved for use by the Graphic System controller.
  - A primary address of 31 or above generates Graphic System error message 12.
  - A secondary address of 32 is ignored.
  - A secondary address of 33 or above generates Graphic System error message 12.

  The address values are automatically converted to Talk, Listen, or Secondary addresses, as determined by the routine.
- Negative values in the address array are ignored. An undefined address value in an address array generates Graphic System error message 36. In a two-dimensional address array, if the primary address is negative, neither the primary nor the secondary address is transmitted.
- Noninteger values specified as addresses are first rounded to integers.

ADDRESS LISTS

Some routines require or allow an address parameter or an address list parameter. If an address or address list is used, the bus configuration is cleared upon completion of the routine by sending the UNT and UNL interface messages.

NOTE

The exceptions to this rule are the TALK and LISTEN routines. Their functions would be defeated by clearing the bus configuration.

If an optional address list is omitted, it is assumed that the talker and/or listener is already configured. They will not be unconfigured upon completion of the routine.

The syntax for the address list parameter is as follows:

<address list>	::=	<address> [;<address>] [;<address>] . . .
<address>	::=	<primary address> | <extended address> | <address array> | <extended address array>
<primary address>	::=	numeric expression
<extended address>	::=	numeric expression, numeric expression (primary address, secondary address)
<address array>	::=	one-dimensional array
<extended address array>	::=	two-dimensional array
<numeric expression>	::=	any expression that yields (evaluates to) a single numeric value.

A sample address list is:

2;4,11;6;12

Here:

2, 6, and 12 are primary addresses

4,11 is an extended address, with primary address 4 and secondary address 11

CALL STATEMENTS

Each ROM pack routine are accessed by specifying the routine name in a 4050 Series BASIC CALL statements. A routine name can be specified as a string constant (e.g., 200 CALL "DCL") or as a string variable (e.g., 300 CALL B$, where B$ = "DCL"). Refer to the *4050 Series Graphic System Reference Manual* for more information on CALL statements.

DETAILED ROUTINE DESCRIPTIONS

The remainder of this section gives detailed routine descriptions, including syntax and descriptive forms, purpose, examples, and a detailed explanation of the action of the routine and of each of the routine arguments.

Routines are arranged alphabetically, by the routine name.

ARSIZE

Purpose

This routine returns the currently dimensioned size of an array.

Syntax Form

[line number] CALL "ARSIZE" , {<array> / <numeric variable>},
<numeric variable>, <numeric variable>

Descriptive Form

[line number] CALL "ARSIZE", variable name, target for row dimension, target for column dimension

Examples

```
100 CALL "ARSIZE", A, B, C
```

Explanation

If the first argument is an array name, this routine returns its row and column dimensions in the second and third arguments, respectively.

If the first argument is the name of a one-dimensional array, a value of 0 is returned in the third argument.

If the first argument is an undimensioned numeric variable, a 0 value is returned in both the second and third arguments.

This command is useful in determining the size of an array after a CONFIG (Configure) command is executed, because the second argument indicates the number of instruments on the bus.

ASKERR [4052R14 Option 1A only]

Purpose

ASKERR supplies information on which error caused a trap and the number of times the error has occured on a consecutive basis.

Syntax Form

[line number] CALL "ASKERR" { [, <numeric variable>, <numeric variable>] } { [, <string variable>] }

Descriptive Form

[line number] CALL "ASKERR" { [, error index, repetition count] } { [, error message string] }

Examples

```
100 CALL "ASKERR" ! Print index and count on screen.

110 CALL "ASKERR",IX,RC ! Get index and count.

120 CALL "ASKERR",A$ ! Get error status in string variable.
```

Explanation

NOTE

This routine is included only in the 4052R14 Option 1A ROM Pack.

Appendix A contains an example of usage for ERRLIST, ASKERR, and RETRY. Appendix D contains a summary of how ERRLIST, ASKERR, and RETRY are meant to work together in "A" Graphic Systems (4052A and 4054A).

The ASKERR routine is used in conjunction with the ERRLIST routine, and with ON SIZE and ON TIMEOUT handlers, to return an error list index and a repetition count. The information can be either a pair of numbers or an ASCII string. ERRLIST must be called for both ON SIZE and ON TIMEOUT for the error interrupt intercept and ASKERR to work properly.

Index for ERRLIST/ON SIZE. The index value returned will be an integer from 1 through the number of elements in the error list. For instance, error list 12;19,32,55 sets traps for errors 12 through 19, 32, and 55. Since there are 10 possible errors, then the index will range from 1 to 10. In this example, if error 15 occured, the index would be 4.

Index for ERRLIST/ON TIMEOUT. The index value returned wil' always be 255; there is no formal error code associated with TIMEOUT. ASKERR, when used with ON TIMEOUT, is primarily used with the repetition count. However, ERRLIST must be called with at least one element (dummy error) to properly enable the error intercepts and repetition counts.

Repetition Count. The second parameter is a repetition count. The repetition count refers to the number of consecutive times the same error occurs in the same line number. This count is used primarily with the RETRY routine to determine when the user's program has "retried" a function or process a reasonable number of times; it avoids getting the program stuck in an "infinite loop." The repetition count is set to zero (0) whenever one of the following occur:

- A call to ERRLIST is executed.
- An error code with a new (different) error code occurs.
- An error occurs in a statement with a different line number than that which caused the last (previous) error.
- A call to CLRREP (clear repetition count) is executed.

The repetition count is affected by all errors, not just those specified by ERRLIST.

String Variable. If the argument to ASKERR is a string variable, then an error message string is returned and stored in the string variable. The format of the string is:

ERROR < EN > OCCURRED IN LINE < LN >, REPETITION COUNT = < RC >

EN is the error number, LN is the line number which caused the trap, and RC is the repetition count.

No Argument. If no argument is specified when calling ASKERR, then the error string (message) is printed on the screen.

ASKERR Error. If ASKERR is called before a trap occurs, then error message 90 (no information available) is printed on the screen and the program halts.

BININ

Purpose

This routine inputs binary block data from a device on the bus.

Syntax Form

[line number] CALL "BININ" $\left\{\begin{matrix}[,\ \text{string constant}] \\ [,\ \text{string variable}\ ]\end{matrix}\right\}$ $\left\{\begin{matrix}<\text{array}> \\ <\text{string variable}>\end{matrix}\right\}$
<numeric variable> [, <address>]

Descriptive Form

[line number] CALL "BININ" [, input mode] , target for data,
target for error code [, talk address]

Examples

```
100 CALL "BININ" , "UNPK", Q, E; 5

200 CALL "BININ" , P$, A$, E; 3, 2

300 CALL "BININ" , "PACK, UNSI", Q, E; 1
```

Explanation

The first argument to the BININ routine is optional. If used, it specifies how binary data is assigned to the destination array. This argument may have the following values:

UNPK	Unpacked mode. In this mode, one byte (an 8-bit positive integer) is assigned to each array element.
PACK	Packed mode. In this mode, two bytes are assigned per array element. The high-order byte is the first byte received. The two bytes are assumed to be a 16-bit two's complement number.

UNSI Unsigned. May be used with packed mode to specify 16-bit unsigned numbers.

NDLM NDLM (NO DELIMITER) is only applicable to the 4052R14 Option 1A ROM Pack. The BININ routine in the 4052R14 Option 1A ROM Pack has been programmed so its default mode is to read the message unit delimiter (if one is present) at the end of a binary block. Specifying NDLM supresses the reading of the message unit delimiter by the BININ routine. If a message unit delimiter is present, and NDLM has been specified, it must be read or dealt with by using some other BASIC statement.

This argument is ignored if the target for data is a string variable. If the argument is omitted and the target is an array, UNPK is assumed.

The target for data specifies the variable where the data is to be stored. It may be either an array or string variable. If it is an array, the array is automatically dimensioned to the corresponding binary block size.

NOTE

The Graphic System allows the user to redimension strings and arrays to a smaller size than the original dimension. For arrays, this routine uses the original dimensioned value (not the current dimension) to determine the maximum number of data values allowed in storage.

When a string argument is specified, the data is stored in the string in ASCII HEX format. Two ASCII characters in the string are used to store the value of each byte transmitted over the bus. These characters represent the value of the high- and low-order four bits of each 8-bit byte. If the dimensioned length of the target is 1, Graphic System error message 12 is issued.

The next argument is a target for an error code. This error code indicates the following conditions upon completion of the command.

0 No errors.

3 The beginning of a binary block (the "%" character) was not found in the message.

4 Checksum error occurred.

5 Block of data input is larger than data space available.

6 EOI received before end of binary block.

7 Odd number of bytes in block with packed mode specified. An additional byte with all bits set to 0 is added to the end of the block.

8 Binary block byte count equals zero.

9 Null binary block - array not valid.

The final argument is also optional. It specifies the address of the device that is transmitting the binary data. If this argument is omitted, the routine assumes that the device is already addressed as a talker. It does not UNTalk the device when the data transfer is completed. If the talk address is specified, the routine UNTalks the device after data input. Because there can be only one talker on the bus at a time, only the first row entry in an array is transmitted.

If an error in one of the argument specifications is detected, the execution of the command is canceled and Graphic System error message 12 is issued.

The input message is scanned for the beginning of the binary block (the % character) before input begins. Any characters in the input string before the % character are ignored. If the binary block data does not fill the target array variable, the array is redimensioned to the size required by the block. Two-dimensional arrays are changed to a single dimension. If the data area is filled before the end of the block is reached, error code 5 is returned and the remainder of the binary block is read but not stored.

BINOUT

Purpose

This routine sends data in binary block format from the Graphic System to devices on the bus

Syntax Form

[line number] CALL "BINOUT" { [, <string constant>] / [, <string variable>] } , { <array> / <string variable> } , <numeric variable> [; <address list>]

Descriptive Form

[line number] CALL "BINOUT" [, output mode] , source of data for block, target for error code [; listen addresses]

Examples

```
100 CALL "BINOUT" , "UNPK", Q, E; 5, 3; 11; 24

200 CALL "BINOUT" , PS, A$, E; 3, 2

300 CALL "BINOUT" , "PACK, NEOI", B$, E; J
```

Explanation

The first argument is optional. It specifies how binary data is transmitted on the bus. If the argument is not specified, UNPK mode is assumed. This argument may have the following values:

UNPK — Unpacked mode. One byte is generated for each array element. The array element is assumed to be an integer value in the range 0 through 255 (noninteger values are rounded). Values less than zero or larger than 255 cause error code 10 to be returned.

PACK — Packed mode. Two bytes are generated for each array element. The array element value is assumed to be a 16-bit integer. The high-order byte is transmitted first. Values less than −32768 or greater than 65535 cause error code 10 to be returned.

NEOI — No EOI. The last data byte is not transmitted with EOI. If this argument is omitted, the last byte is transmitted with EOI.

NDLM — NDLM (NO DELIMITER) is only applicable to the 4052R14 Option 1A ROM Pack. The BINOUT routine in the 4052R14 Option 1A ROM Pack has been programmed so its default mode is to output a semicolon (;) character at the end of a binary block as the message unit delimiter. Specifying NDLM suppresses the output of the semicolon (specifies no message unit delimiter to be output by the 4052R14 Option 1A version of the BINOUT routine).

NEOI may be used with either UNPK or PACK by putting a comma between the parameters: e.g., "UNPK, NEOI".

The source argument specifies the variable from which the data is taken. This argument may be either an array or a string variable. If an array argument is specified, the optional input mode argument may be used to define the data format. No data will be transmitted if an element of the source argument is out of the range allowed by the specified input mode.

When a string argument is specified, the data is assumed to be in ASCII HEX format. That is, two bytes in the string are used to store the value of each byte transmitted over the bus. If the length of the source string is odd, error code 11 is returned and the last character in the string is not transmitted.

The next argument is a variable set to indicate the following conditions on completion of the command. The variable has the following values:

0 No errors.

10 Data in source variable out of range. The block is not transmitted and the instrument address UNT and UNL are sent to clear the bus.

11 Source string length is odd.

The final argument specifies the address(es) of the device(s) receiving the binary data. This address list is optional. If it is specified, the routine sends the UNTalk and UNListen messages after the data has been transmitted. If the list is not specified, BINOUT assumes that the listening devices have been previously addressed and therefore does not send UNTalk and UNListen after sending the data.

If an error in any argument is detected, the execution of the routine is canceled and Graphic System error message 12 is issued.

CLRREP [4052R14 Option 1A only]

Purpose

To set the error repetition counter to zero (0).

Syntax Form

[line number] CALL "CLRREP"

Descriptive Form

[line number] CALL "CLRREP"

Example

```
100 CALL "CLRREP"
```

Explanation

NOTE

This routine is included only in the 4052R14 Option 1A ROM Pack.

The CLRREP routine is called to set the error repetition counter to zero (0). This counter is incremented each time the same error (trap) occurs in the same program line as the previous error (trap).

Refer to the descriptions of the ASKERR and RETRY routines for a more complete description of how the repetition counter might be used.

CONFIG

Purpose

This routine determines the addresses of the devices on the bus and returns these addresses in the array specified as an argument.

Syntax Form

[line number] CALL"CONFIG" [, <numeric expression>] , <numeric variable> ; <array>

Descriptive Form

[line number] CALL "CONFIG" [, time out] , target for error code; target for device addresses

Examples

EXAMPLE 1

```
110 DIM A(15)
110 REM CONFIGURE FOR PRIMARY ADDRESS ONLY
120 CALL "CONFIG", 5, E; A
130 IF E THEN 500
```

EXAMPLE 2

```
200 REM CONFIGURE SYSTEM WITH PRIMARY AND SECONDARY ADDRESSES
210 DIM B(15, 2)
220 CALL "CONFIG", E; B
230 IF E THEN 4000
```

Explanation

This routine determines the active devices on the bus and returns their addresses in the array specified as the final argument.

The first argument is optional and specifies the time out value in milliseconds. This argument has a range of 0 to 65535. The default is 1 millisecond.

The method used to determine whether a device is present on the bus at any given address is to send the UNL message followed by a listen address. If the NDAC line remains asserted after the time out period has elapsed, the routine assumes that a device is present at this address.

If the specified time out value is not adequate for a given device, the routine assumes that the device responds to all addresses. If this happens, simply provide a time out value greater than the time it takes the device to respond to the ATN false condition.

The second argument is the target for the error code. It has the following values:

0 No errors.

1 The array was not large enough to hold all of the addresses. Delete the array and redimension it, or increase the time out value.

2 No device responded. The array is left unchanged.

The last argument specifies an array which is used to return the addresses of the devices found to be active on the bus.

- If a one-dimensional array is specified, only primary addresses are checked.
- If a two-dimensional array is specified:
  - If a device responds to a given primary address, then no secondary addresses for this primary address are checked. The secondary address entry in the array is set to -1, and the routine proceeds to check the next primary address.
  - If no device responds to a given primary address, then all possible primary/secondary address combinations for this primary address are checked. When all combinations have been checked or a device has responded, the routine proceeds to the next primary address.

Addresses of responding devices are stored in consecutive rows of the array: first responding device in row 1, second responding device in row 2, etc.

If the array specified is not completely filled with addresses, its dimension is modified automatically to correspond to the number of devices found. Only the number of rows is modified for a two-dimensional array.

NOTE

The CONFIG routine operates only with devices that include talker (T or TE) and listener (L or LE) functions. It will not detect talk-only devices. It will not operate properly with listen-only devices on the bus because the listen-only devices will answer all addresses.

DCL

Purpose

This routine clears all devices with a DC1 or DC2 function subset on the bus.

Syntax Form

[line number] CALL "DCL"

Descriptive Form

[line number] CALL "DCL"

Example

```
100 CALL "DCL"
```

Explanation

CALL "DCL" sends the IEEE-488 Device Clear (DCL) interface message (ATN asserted, with DEC 20 or X'14').

If no devices are currently on the bus, the Graphic System issues error message 69.

DECHEX

Purpose

This routine converts the decimal value of an expression to its hexadecimal equivalent in ASCII characters.

Syntax Form

[line number] CALL "DECHEX" , <numeric expression> , <string variable>

Descriptive Form

[line number] CALL "DECHEX", decimal value, target for HEX representation

Examples

```
100 CALL "DECHEX", 278, H$

200 CALL "DECHEX", C, I$
```

Explanation

The first argument defines the decimal value to be converted to HEX. Noninteger values are rounded to integers before the conversion is performed. The range of decimal values allowed is −32768 to 65535. If the value specified is outside of this range, Graphic System error message 96 is issued. Negative values are treated as 16-bit two's complement numbers. For example:

−32768 is converted to X'8000'

−1 is converted to X'FFFF'

The second argument specifies the string in which the result is returned. The string returned is a four-character ASCII HEX representation of the value of the first argument. If the dimensioned length of the target string is less than four, DECHEX returns as many least significant HEX digits as fit in the string, with leading zeros added, if necessary. If the non-zero digits of the number do not fit in the string, Graphic System error message 21 is issued.

ERRHLP

Purpose

This routine returns information about the Tektronix Codes and Formats standard error codes.

Syntax Form

[line number] CALL "ERRHLP" , <numeric expression> , <string variable>

Descriptive Form

[line number] CALL "ERRHLP", target for error code, target for ASCII information

Examples

EXAMPLE 1

```
100 CALL "ERRHLP", 201, E$
```

EXAMPLE 2

```
200 CALL "ERRHLP", E, E$
210 IF E$ < >" " THEN 250
220 REM EVENT NOT FOUND IN TABLE
250 PRINT E$
```

Explanation

This routine returns ASCII explanations of the Codes and Formats error or event codes. If a matching code is not found, a null string ("") is returned.

The error code and ASCII information are:

Error Code	ASCII Information
100	COMMAND ERROR
101	COMMAND HEADER
102	HEADER DELIMITER
103	COMMAND ARGUMENT
104	ARGUMENT DELIMITER
105	NON-NUMERIC ARGUMENT
106	MISSING ARGUMENT
107	INVALID MESSAGE UNIT DELIMITER
108	CHECKSUM ERROR
109	BYTECOUNT ERROR
200	EXECUTION ERROR
201	COMMAND NOT EXECUTABLE IN LOCAL MODE
202	SETTINGS LOST DUE TO rtl
203	I/O BUFFERS FULL
204	SETTINGS CONFLICT
205	ARGUMENT OUT OF RANGE
206	GROUP EXECUTE TRIGGER IGNORED
300	INTERNAL ERROR
301	INTERRUPT FAULT
302	SYSTEM ERROR
400	SYSTEM EVENT
401	POWER ON
402	OPERATION COMPLETE
403	USER REQUEST
404	POWER FAIL
405	REQUEST CONTROL
406	PASSED CONTROL
500	EXECUTION WARNING
600	INTERNAL WARNING
700	DEVICE DEPENDENT EVENT

ERRLIST [4052R14 Option 1A only]

Purpose

To enable specified errors to be trapped and handled by a user's ON SIZE error handling routine and to enable the repetition count.

Syntax Form

[line number] CALL "ERRLIST" [,<error list>]

The following describes syntactically what an <error list> is:

<error list> = <error> [,<error>]...
<error> = <number> | <one dimensional array> | <range>

<number> = <scaler variable> | <scaler expression>

<range> = <number>:<number>

The symbols <, >, =, ., , , [, and] are explained at the beginning of Section 2.

Descriptive Form

[line number] CALL "ERRLIST" [, error code list]

Examples

```
100 CALL "ERRLIST" ! Restore ON SIZE to its normal list.

110 CALL "ERRLIST",3,5,A,18,23 ! scalers, single dim array, range.
```

Explanation

NOTE

This routine is included only in the 4052R14 Option 1A ROM Pack.

Appendix A contains an example of usage for ERRLIST, ASKERR, and RETRY. Appendix D contains a summary of how ERRLIST, ASKERR, and RETRY are meant to work together in "A" Graphic Systems (4052A and 4054A).

The argument(s) to the ERRLIST routine are a list of error codes that the user wishes to trap. As noted in the syntax specification, these error codes may be specified as individual codes or as a range.

The ERRLIST routine uses this list of errors to expand the function of the "ON SIZE THEN < line number > " statement. After executing the ERRLIST statement, the ON SIZE statement performs the error trapping function for any error specified in the error code list (not limited to just SIZE errors anymore). Only the errors specified in the < error list > will cause control to transfer to the line number specified in the ON SIZE statement. Other errors are printed on the screen and stop program execution.

Each value in the < error list > must be an integer value from 1 to 127. Values outside of this range will cause Graphic System error 12.

Exceptions. The following error codes may be specified in the < error list >, but do not cause transfer of control.

Error Code	Error Condition
39	Memory full
40	Program interrupted
41	No ON SIZE handler specified
42	No ON FULL handler specified
43	No ON SRQ handler specified
44	No ON EOI handler specified
45	No ON EXTERNAL INTERRUPT 1 handler specified
46	No ON EXTERNAL INTERRUPT 2 handler specified
47	No ON EXTERNAL INTERRUPT 3 handler specified
48	No ON EOF handler specified
91	STOP statement
92	ABORT
110	GPIB timeout (No ON TIMEOUT)

The "No ON" errors, including "GPIB timeout," are better handled by having a specific ON statement and a matching handler.

Calling the ERRLIST routine with no arguments (no < error list >), restores the ON SIZE function to its normal (predefined) list of errors. If you want to disable all ON SIZE trapping, you must use the OFF SIZE statement.

Whenever a new < error list > is specified using ERRLIST, the repetition count, < error list > index, and error line number variables are set to zero (0). Refer to the description of the ASKERR routine for a description of these variables.

GET

Purpose

This routine sends the Group Execute Trigger interface message.

Syntax Form

[line number] CALL"GET" [; <address list>]

Descriptive Form

[line number] CALL"GET" [; address(es)]

Examples

EXAMPLE 1:

```
100 CALL"GET";2, 5, 8, 13, 4
```

EXAMPLE 2:

```
200 REM TRIGGER DEVICES ALREADY SET UP AS LISTENERS
210 CALL"GET"
```

Explanation

The GET routine sends the listen addresses of the devices specified in the address list. Then it sends the Group Execute Trigger message (<GET> : ATN asserted, with DEC 8 or X'8') and, finally, the UNTalk and UNListen interface messages.

If the address list is omitted only the GET message is transmitted. The devices previously addressed as listeners remain listen addressed.

GTL

Purpose

The GTL (Go To Local) routine returns the devices specified in the address list to either Local State (LOCS) or Local With Lockout State (LWLS).

Syntax Form

[line number] CALL"GTL" [; <address list>]

Descriptive Form

[line number] CALL"GTL" [; address(es)]

Examples

EXAMPLE 1:

```
100 REM DEVICES AT ADDRESSES 3, 5, 7 ARE SENT TO LOCAL
110 CALL"GTL"; 3; 5; 7
```

EXAMPLE 2:

```
200 REM ALL DEVICES CURRENTLY ADDRESSED ARE SENT TO LOCAL
210 CALL"GTL"
```

EXAMPLE 3:

```
300 REM SEND ALL ACTIVE DEVICES TO LOCAL
310 CALL"CONFIG"; E; A
320 CALL"GTL"; A
```

Explanation

The GTL routine sends the listen addresses of the devices specified in the address list. It then sends the Go To Local interface message (<GTL>: ATN asserted, with DEC 1 or X'11') and, finally, the UNTalk and UNListen messages.

If the address list is omitted, only the GTL interface message is transmitted. The devices previously addressed as listeners remain listen addressed.

HEXDEC

Purpose

This routine converts an ASCII HEX number representation to decimal.

Syntax Form

[line number] CALL"HEXDEC" , { <string constant> / <string variable> } , <numeric variable>

Descriptive Form

[line number] CALL"HEXDEC", HEX representation, target for decimal value

Examples

```
100 CALL"HEXDEC", "F078", T
```

Explanation

The first argument defines a string which represents an unsigned integer value in ASCII hexadecimal. The length of the string must be 4 characters or less.

The second argument is the target where the decimal equivalent of the HEX string is returned. The HEX characters are converted until the end of the string is reached. If no legal HEX characters are found, a non-HEX character is encountered, or more than four HEX digits are specified, then Graphic System error message 96 is issued. If a null string is found, Graphic System error message 12 is issued.

IFC

Purpose

This routine clears the GPIB interface by asserting Interface Clear.

Syntax Form

[line number] CALL"IFC"

Descriptive Form

[line number] CALL"IFC"

Examples

```
150 CALL"IFC"
```

Explanation

This clears the GPIB interface by pulsing the IFC control line for approximately five milliseconds.

LAST

Purpose

This routine finds, opens, and optionally returns the file number or header of the last file on the tape.

Syntax Form

[line number] CALL"LAST" { [, <string variable>] / [, <numeric variable>] }

Descriptive Form

[line number] CALL"LAST" [, target for file number or header]

Examples

```
100 CALL"LAST", X

200 CALL"LAST", H$

300 CALL"LAST"
```

Explanation

The LAST routine finds the last file on the tape. If the optional argument:

- is not specified, the file header is printed on the screen.
- is a string variable, the entire file header is returned in the string. The header is not printed on the screen.
- is a numeric variable, the file number is returned. The file number is not printed on the screen.

If the last file is not found, Graphic System error message 52 is issued, and the number or header of the file found is returned. If a mag tape file is open, this routine closes it before it finds the LAST file. If the "NO HEADER" mode was selected in the mag tape environmental parameters, this routine issues Graphic System error message 60 for 4051R14 and 4052R14 ROM Packs, and issues message 61 for the 4052R14 Option 1A ROM Pack.

LISTEN

Purpose

This routine makes the devices specified in the address list into listeners.

Syntax Form

[line number] CALL"LISTEN" [; <address list>]

Descriptive From

[line number] CALL"LISTEN" [; address(es)]

Example

EXAMPLE 1:

```
100 CALL"LISTEN"; 5; 27, 3
```

EXAMPLE 2:

```
200 REM LISTEN ALL ACTIVE DEVICES, RESPONDING TO A PRIMARY ADDRESS
210 DIMENSION A(15)
220 CALL "CONFIG", E; A
230 CALL"LISTEN"; A
```

Explanation

This routine converts the instrument address into the instrument listen address and sends it over the bus with ATN asserted. Negative values are not transmitted.

LLO

Purpose

This routine puts all instruments on the bus into either Local With Lockout State (LWLS) or Remote With Lockout State (RWLS). Whether the instruments actually enter a lockout state or not is also determined by the state of the Remote Enable (REN) control line. (See the LOCS routine discussion for a description of when REN is asserted.)

Syntax Form

[line number] CALL "LLO"

Descriptive Form

[line number] CALL "LLO"

Example

```
100 CALL "LLO"
```

Explanation

This routine sends the Local Lockout (LLO) interface message (ATN asserted, with DEC 17 or X'11').

NOTE

Because LLO is a universal GPIB command, it places all devices in a lockout mode. If a device is addressed as a listener by a subsequent command, the front panel is locked out until the device is specifically sent to a local state with GTL or LOCS.

LOCS

Purpose

This routine puts all devices in the LOCAL State.

Syntax Form

[line number] CALL "LOCS"

Descriptive Form

[line number] CALL "LOCS"

Examples

```
100 CALL "LOCS"
```

Explanation

This routine returns all devices to the LOCAL state by unasserting the REN (Remote Enable) bus control line for approximately 280 microseconds.

The REN line is reasserted after the LOCS routine is executed because the Graphic System controller asserts REN when the INIT or any I/O statement is executed. REN is unasserted when the controller becomes not BUSY (IDLE) or when the LOCS routine is executed.

NOTE

Setting REN false is the only way to remove a device from a lockout state.

MTPACK

Purpose

This routine is included to help correct poor tape spooling that may occur with some tape cartridges.

Syntax Form

[line number] CALL "MTPACK"

Descriptive Form

[line number] CALL "MTPACK"

Example

```
100 CALL "MTPACK"
```

Explanation

The MTPACK routine runs a tape forward to the end and back again. This procedure may help correct poor tape spooling that sometimes occurs with a tape cartridge. Any new tape cartridge should be respooled with this routine before it is used.

NOTE

This routine is implemented in the 4051R14 ROM pack only. The routine is implemented internally in the 4052 and 4054 Graphic Systems.

NEWTAP [Version 1: 4051R14 and 4052R14 only]

Purpose

This routine finds the present file location and returns the tape cartridge status as a numeric code.

Syntax Form

[line number] CALL "NEWTAP" [, <numeric variable>]

Descriptive Form

[line number] CALL "NEWTAP" [, target for cartridge status]

Examples

EXAMPLE 1

```
100 CALL "NEWTAP"
```

EXAMPLE 2

```
200 CALL "NEWTAP", C
210 GOSUB C + 1 of 300, 400, 500
300 REM HANDLE NO TAPE INSERTED CONDITION
400 REM PREVIOUSLY ACCESSED TAPE
500 REM FIRST TIME ACCESSED TAPE
```

Explanation

NOTE

This explanation is applicable to Version 1: 4051R14 and 4052R14 ROM Packs only.

NEWTAP is used to get the status of a tape cartridge in the internal mag-tape unit.

The variable, if present, reports the following status in numeric form:

Status	Meaning
0	No tape cartridge present.
1	Tape inserted and at least one previous tape operatoin has been performed.
2	This call to NEWTAP is the first tape operation performed on the present cartridge (newly inserted tape).

The status values 1 and 2 do not refer to whether the tape has information or is blank; they refer to whether there has been any tape access/operation such as REWIND, FIND, MARK, INPUT, etcetera since the tape has been inserted in the mag-tape slot.

NEWTAP itself is considered an operation since it positions the tape at a logical resting place — beginning of present file, Beginning-of-Tape for a blank tape, etc. For instance, if the tape is presently in the middle of file 5 and NEWTAP is called, NEWTAP will position the tape at the beginning of file 5 and leave the file open — same as if FIND 5 had been executed.

NEWTAP [Version 2: 4052R14 Option 1A only]

Purpose

NEWTAP positions the tape (if one is inserted) at the beginning of the present file and returns the tape cartridge status and write protect status as numeric codes.

Syntax Form

[line number] CALL "NEWTAP" [,numeric variable [,numeric variable]]

Descriptive Form

CALL "NEWTAP" [,target for cartridge status [,target for write protect status]]

Examples

EXAMPLE 1 position tape, no status.

```
100 CALL "NEWTAP"
```

EXAMPLE 2 general tape handler.

```
100 CALL "NEWTAP",C
110 GOSUB C + 1 OF 300,400,500


300 REM Handle no tape inserted condition.


400 REM Handle previously accessed tape.


500 REM Handle first time accessed tape.
```

Explanation

NOTE

This explanation is applicable to Version 2 4052R14 Option 1A.

NEWTAP is used to get the status of a tape cartridge in the internal mag-tape unit.

The first variable, if present, reports the following status in numeric form:

Status	Meaning
0	No tape cartridge present.
1	Tape inserted and at least one previous tape operation has been performed.
2	This call to NEWTAP is the first tape operation performed on the present cartridge (newly inserted tape).

The status values 1 and 2 do not refer to whether the tape has information or is blank, they refer to whether there has been any tape access/operation such as REWIND, FIND, MARK, INPUT, etcetera since the tape has been inserted in the mag-tape slot.

NEWTAP itself is considered an operation since it positions the tape at a logical resting place — beginning of present file, Beginning-of-Tape for a blank tape, etc. For instance, if the tape is presently in the middle of file 5 and NEWTAP is called, NEWTAP will position the tape at the beginning of file 5 and leave the file open — same as if FIND 5 had been executed.

The second variable returns write protect status. Zero (0) means enabled for writing; one (1) means write protected. If there is no tape inserted, the value is also a one (1). However, the second variable should not be checked if the first variable is zero (0) — no tape inserted.

POLL

Purpose

The POLL routine is used to serial poll the devices specified in the address list. It returns information identifying the device requesting service (if any) and the status byte returned by the device requesting service or the last device polled.

This routine is similar to the 4050 BASIC POLL statement, with the following exceptions. The 4050R14 ROM Pack POLL routine:

- may time out if a device does not respond to a serial poll
- returns a status byte whether or not a device requested service.
- may take an address array as an argument to specify addresses, using the conventions for addressing described at the beginning of this section.

Syntax Form

[line number] CALL "POLL" [, < numeric expression >],
< numeric variable > , < numeric variable > , < address list >

Descriptive Form

[line number] CALL "POLL" [, time out], target for device identifier, target for status byte, address(es)

Examples

EXAMPLE 1

```
100 DIM A(15)
110 CALL "CONFIG", E, A
120 ON SRQ THEN 500
130 CALL "SRQON"

200 REM MAIN PROGRAM
```

```
500 REM SRQ INTERRUPT HANDLER
510 CALL"POLL", 5, I, S, A
520 IF I<>0 THEN 560
530 REM RE-CONFIGURE
540 CALL"CONFIG", E, A
550 CALL"POLL", 1000, I, S, A
560 REM SERVICE INTERRUPT FROM DEVICE I

700 RETURN
```

EXAMPLE 2

```
1000 REM THIS POLL RETURNS THE STATUS FROM DEVICE 3
1010 CALL"POLL", 5, I, S, 3
```

Explanation

The first argument to the POLL routine is optional. It specifies the time out value in milliseconds. The range is from 0 to 65535 milliseconds, with 0 indicating an infinite wait. The default value is 5 milliseconds. If the routine does not receive a response from a device within the specified length of time, the routine "times out." This allows the program to continue even when a device fails to send a status byte in response to the serial poll.

The second argument identifies the device which sends the status byte. It is an index for the address list and has a value of 0 if no device responds to the poll by returning a status byte. If no device returns a status byte with RQS asserted, this routine returns the status byte from the last device that responded to the poll, and the index is negative.

The third argument is the variable in which the status byte is returned.

After the [illegible] the arguments specify addresses of the device(s) to be POLLed. These should follow the conventions for address lists given at the beginning of this section.

PPD

Purpose

This routine unconfigures selected devices. It instructs selected devices to no longer respond to parallel poll.

Syntax Form

[line number] CALL "PPD" [; <address list>]

Descriptive Form

[line number] CALL "PPD" [; address(es)]

Examples

```
100 CALL "PPD"; A

200 CALL "PPD"; 3, 5, 7
```

Explanation

The PPD routine first sends addresses specified in the address list. It then sends the Parallel Poll Configure (PPC) interface message (ATN asserted, with X'5'). Next, the PPD message (ATN unasserted, with X'70') is transmitted. Finally, if an address list was specified, the UNTalk and UNListen messages are sent.

PPE

Purpose

This routine configures the devices that are to respond to a parallel poll.

Syntax Form

[line number] CALL"PPE" <numeric expression> ,
<numeric expression>[; <address list>]

Descriptive Form

[line number] CALL"PPE", sense of parallel poll response, line on which device responds [; addresses]

Examples

```
100 CALL"PPE", 1, 4; A

200 CALL"PPE", 0, 7; 3, 2
```

Explanation

The PPE routine first sends addresses specified in the address list, then the Parallel Poll Configure (PPC) interface message. The PPE message is then transmitted, encoded with the parallel poll response sense (0 or 1) and the bus line number on which the response is to be generated (1 to 8). If the arguments are not in range, Graphic System error message 18 is issued. Finally, if an address list was specified, the UNTalk and UNListen commands are sent.

PPOLL

Purpose

This routine performs a parallel poll of the devices previously configured for parallel poll.

Syntax Form

[line number] CALL"PPOLL" , < numeric variable>

Descriptive Form

[line number] CALL"PPOLL", target variable for Parallel Poll response

Examples

```
100 CALL"PPOLL", A
```

Explanation

The PPOLL routine asserts the ATN and EOI lines on the IEEE-488 interface and reads the data lines. The value returned in the target variable is the decimal equivalent of the binary values defined by the parallel poll response on the data lines.

PPU

Purpose

This routine unconfigures all devices. It instructs all devices not to respond to parallel poll.

Syntax Form

[line number] CALL"PPU"

Descriptive Form

[line number] CALL"PPU"

Example

```
100 CALL"PPU"
```

Explanation

The PPU routine sends the Parallel Poll Unconfigure interface message (ATN asserted, with X'15').

PRISTR

Purpose

This routine outputs a string to the GPIB Interface without transmitting EOI with the last byte. It also does not transmit the carriage return sent by some forms of the Graphic System PRINT command.

Syntax Form

[line number] CALL"PRISTR", { <string constant> / <string variable> } [; <address list>]

Descriptive Form

[line number] CALL"PRISTR" , string to be output [; address(es)]

Examples

```
100 CALL"PRISTR","LLSET"; 5

110 CALL"BINOUT", Q, E; 5
```

Explanation

If the address list is specified, PRISTR first sends these items as listen addresses. It then sends the string, the last byte being sent WITHOUT EOI.

If the address list is specified, then UNTalk and UNListen are sent to complete the execution of the routine.

RBIN

Purpose

This routine inputs binary data from a device on the bus until a byte with EOI asserted is received or until the destination variable is filled.

Syntax Form

[line number] CALL"RBIN" { [, <string constant>] / [, <string variable>] } , { <array> / <string variable> } , <numeric variable> [; <address>]

Descriptive Form

[line number] CALL"RBIN" [, output mode] , target for data, target for error code [, talk address]

Examples

```
100 CALL"RBIN", Q, E; 5

200 CALL"RBIN", AS, E; 3, 2

300 CALL"RBIN", "PACK, UNSI", Q, E; 1
```

Explanation

The first argument to the RBIN routine is optional. It indicates how the binary data is assigned to the destination array. The argument is ignored if the destination argument is a string variable. This argument may have the following values:

UNPK — Unpacked mode. One byte (an 8-bit positive integer) is assigned to each array element. This is the default mode.

PACK — Packed mode. Two bytes per array element. The high-order byte is received first. The two bytes taken together define a 16-bit two's complement number.

UNSI — Unsigned. Used with packed mode to specify 16-bit unsigned integer.

The target for data argument is the variable in which the data is stored. This argument may be an array or a string variable. If an array argument is specified, packed mode may be specified to define the data format.

When a string argument is specified, the data is stored in the string in ASCII HEX format. Two bytes in the string store the value of each byte received over the bus.

The third argument is a variable which RBIN sets to indicate the following conditions on completion of the command.

0 — No errors

7 — Odd number of bytes was received in PACKed mode. If this occurs, an additional byte with all bits set to zero is added as the last byte of the target string.

The last argument is optional. It specifies the address of the device that is transmitting the binary data. When the address argument is omitted, the routine assumes the device is already addressed as a talker. It does not UNTalk the device when the data transfer is completed. If the talk address is specified in the argument list, the RBIN routine UNTalks the device after the data has been input. Because there can be only one talker on the bus at a time, only the first row entry in an array is used for the address.

If an error in one of the argument specifications is detected, the execution of the routine is canceled and Graphic System error message 12 is issued.

The input of data from the bus is terminated by a byte received with EOI asserted. If the binary data does not fill the target array, the array is redimensioned to the size required by the data. Two-dimensional arrays are changed to one-dimensional arrays. If the data area is filled before the end of the data is reached, the remainder of the data is not read from the active talker.

NOTE

The Graphic System allows the user to redimension strings and arrays to a smaller size than the original dimension. For arrays, this routine uses the original dimensioned size (not the current dimension) to determine the maximum number of bytes input. Therefore, to input a specific number of bytes, the user should DELETE the array and dimension it to the required size.

RETRY [4052R14 Option 1A only]

Purpose

To return program control from an error handler back to the line in which the error occured.

Syntax Form

[line number] CALL "RETRY"

Descriptive Form

[line number] CALL "RETRY"

Example

```
100 CALL "RETRY"
```

Explanation

NOTE

This routine is included only in the 4052R14 Option 1A ROM Pack.

Appendix A contains an example of usage for ERRLIST, ASKERR, and RETRY. Appendix D contains a summary of how ERRLIST, ASKERR, and RETRY are meant to work together in "A" Graphic Systems (4052A and 4054A).

The RETRY routine performs a function similar to the RETURN statement. RETURN transfers control to the line *following* the line that caused the transfer. RETRY transfers control back to the "beginning" of the *same* line that caused the transfer. RETRY is used when transfer of control was caused by an interrupt (error trap) to an error handler and the "program" wants to change or look at something and then "try" again.

The RETRY routine may be called only in the highest level of an error handler (either a SIZE or TIMEOUT handler). In other words, there must be no other return addresses on the "stack" on top of (later than) the trap return address. For instance, if an error handler calls a subroutine (executes a GOSUB) and the subroutine then calls RETRY (instead of executing a RETURN), then Graphic System error 90 will be issued and program execution will be stopped.

RWLS

Purpose

This routine puts the devices specified in the address list into the Remote With Lockout State (RWLS). Note that whether the devices are in the Lockout State also depends upon the state of the Remote Enable control line (REN), controlled by the Graphic System. (See the LOCS routine description for details of REN assertion.)

Syntax Form

[line number] CALL"RWLS" [; <address list>]

Descriptive Form

[line number] CALL"RWLS" [; address(es)]

Example

EXAMPLE 1

```
100 DIM A(15)
110 CALL"CONFIG", E; A
120 CALL"RWLS"; A
```

EXAMPLE 2

```
200 CALL"RWLS", 1; 4, 5
```

Explanation

This routine sends the LLO interface message (ATN asserted, with DEC 17 or X'11') followed by the device listen addresses specified in the address list.

NOTE

Since LLO is a universal GPIB command, it puts all devices in a lockout state. If a device is later addressed as a listener, its front panel is locked out until the device is sent to a local state with GTL or LOCS (i.e., REN is unasserted).

SDC

Purpose

The SDC (Selected Device Clear) routine clears the devices specified in the address list.

Syntax Form

[line number] CALL"SDC" [, < address list>]

Descriptive Form

[line number] CALL"SDC" [, address(es)]

Examples

```
100 CALL"SDC", 2, 6, 25

200 CALL"SDC", A
```

Explanation

This routine sends the listen addresses of the devices specified in the address list, followed by the Selected Device Clear (SDC) interface message (ATN asserted, with DEC 4 or X 4), then by UNTalk and UNListen.

If the address list is omitted, only the SDC interface message is transmitted. The devices previously addressed as listeners remain listen addressed.

SRQOFF [Version 1: 4051R14 and 4052R14 only]

Purpose

Calling SRQOFF disables recognition of SRQ interrupts by the Graphic Computing System.

Syntax Form

[line number] CALL "SRQOFF"

Descriptive Form

[line number] CALL "SRQOFF"

Example

100 CALL "SRQOFF"

Explanation

NOTE

This explanation is applicable to Version 1: 4051R14 and 4052R14 ROM Packs only.

This routine is executed automatically at power-on and as part of the INIT statement. This allows turning on the system, loading programs, initializing programs, and activating SRQ support (via the ON SRQ statement) before SRQ interrupts are enabled using the SRQON routine.

Appendix C has a more detailed explanation of the SRQ interrupt processes.

SRQOFF [Version 2: 4052R14 Option 1A only]

Purpose

Calling SRQOFF, in a 4052A or 4054A Graphic Computing System, performs the same function as the OFF SRQ statement.

Syntax Form

[line number] CALL "SRQOFF"

Descriptive Form

[line number] CALL "SRQOFF"

Example

```
100 CALL "SRQOFF"
```

Explanation

NOTE

This explanation is applicable to Version 2: 4052R14 Option 1A ROM Pack only.

In a 4052A or 4054A Graphic Computing System, the OFF SRQ statement disables SRQ interrupts as well as "turns off" the ON SRQ action. Therefore, the SRQOFF routine is only there to provide compatibility with programs that were written for 4051, 4052, and 4054 Graphic Computing Systems. If you are writing programs for 4052A or 4054A systems only, then you do not need to use SRQOFF; just use OFF SRQ instead.

Appendix C has a more detailed explanation of the SRQ interrupt processes.

SRQON [Version 1: 4051R14 and 4052R14 only]

Purpose

Calling SRQON enables recognition of SRQ interrupts by the Graphic Computing System.

Syntax Form

[line number] CALL "SRQON"

Descriptive Form

[line number] CALL "SRQON"

Example

```
100 ON SRQ THEN 200
110 CALL "SRQON"
```

Explanation

NOTE

This explanation is applicable to Version 1 4051R14 and 4052R14 ROM Packs only.

The ON SRQ statement, when an SRQ interrupt is acknowledged by a Graphic Computing System, determines where to branch to for servicing the SRQ interrupt. The SRQON routine determines whether or not SRQ interrupts are acknowledged. SRQ interrupts are not acknowledged if they have been disabled by calling SRQOFF; see the SRQOFF routine Version 1 description.

It takes both statements, as shown in the example, to properly set the system up to handle SRQ interrupts.

Appendix C has a more detailed explanation of the SRQ interrupt process.

SRQON [Version 2: 4052R14 Option 1A only]

Purpose

The SRQON routine, in 4052A and 4054A Graphic Computing Systems, performs no active function. It is there to provide compatibility for programs originally written for non "A" 4050 Series Systems or for programs that must run on the full 4050 Series of Graphic Computing Systems.

Syntax Form

[line number] CALL "SRQON"

Descriptive Form

[line number] CALL "SRQON"

Example

```
100 CALL "SRQON"
```

Explanation

NOTE

This explanation is applicable to Version 2: 4052R14 Option 1A ROM Pack.

In a 4051, 4052 or 4054 Graphic Computing System, the SRQON routine enables SRQ interrupts and the ON SRQ statement determines where to branch to if an SRQ interrupt is received. However, in a 4052A or 4054A system, the ON SRQ statement enables SRQ interrupts as well as determines the branch line number for servicing SRQ interrupts; it does both. Therefore, there is no direct need for an SRQON function in 4052A or 4054A systems.

It is there for compatibility with 4051, 4052, and 4054 systems, if needed.

Appendix C has a more detailed explanation of the SRQ interrupt processes.

STBHLP

Purpose

This routine returns information about the Tektronix Codes and Formats standard status bytes. For more information concerning the Tektronix Codes and Formats standard, see reference 1 in Appendix C.

Syntax Form

[line number] CALL"STBHLP" , < numeric expression> , < string variable>

Descriptive Form

[line number] CALL"STBHLP", status byte, target for ASCII information

Examples

100 CALL"STBHLP", 97, S$

200 CALL"STBHLP", S, S$

Explanation

The first argument is the value of the status byte. If a matching status byte is not found, a null string ("") is returned. Bit 5 (the busy bit) is masked and the code is matched against a table stored in the ROM Pack. Values relevant to the ROM pack operation are shown in the following table.

The status bytes and corresponding ASCII information are:

Status Byte	ASCII Information
0	NULL STATUS
16	NULL STATUS
65	POWER ON
66	OPERATION COMPLETE
67	USER REQUEST
68	REQUEST CONTROL
69	PASSED CONTROL
81	POWER ON
82	OPERATION COMPLETE
83	USER REQUEST
84	REQUEST CONTROL
85	PASSED CONTROL
97	COMMAND ERROR
98	EXECUTION ERROR
99	INTERNAL ERROR
100	POWER FAIL
101	EXECUTION WARNING
102	INTERNAL WARNING
113	COMMAND ERROR
114	EXECUTION ERROR
115	INTERNAL ERROR
116	POWER FAIL
117	EXECUTION WARNING
118	INTERNAL WARNING

TALK

Purpose

This routine makes a talker of the device at the specified address.

Syntax Form

[line number] CALL"TALK", <address>

Descriptive Form

[line number] CALL"TALK"; talk address

Examples

```
100 CALL"TALK"; K

200 CALL"TALK"; 5
```

Explanation

This routine converts the instrument address to an instrument talk address, as follows:

primary address transmitted	=	primary address + 64
secondary address transmitted	=	secondary address + 96

These values are transmitted over the GPIB with ATN asserted. If any value is out of range (from 1 through 30 for primary addresses, and from 0 through 31 for secondary addresses), Graphic System error message 12 is issued.

NOTE

Since only one talker is allowed on the bus at a given time, only the first row entry of an array is transmitted:

A(1)	*for a one-dimensional array*
A(1,1)	*for a two-dimensional array, and primary address only*
A(1,1) A(1,2)	*for a two-dimensional array, with both primary and secondary addresses*

TAPEAPP [4052R14 Option 1A only]

Purpose

To input and append an entire ASCII file, using the internal mag-tape, to the string in the specified string variable

Syntax Form

[line number] CALL "TAPEAPP", < string variable >

Descriptive Form

[line number] CALL "TAPEAPP",target for mag-tape file

Example

```
100 FIND 1
110 CALL "TAPEAPP",A$
```

Explanation

NOTE

This routine is included only in the 4052R14 Option 1A ROM Pack.

The TAPEAPP routine inputs an ASCII data or program file using the internal mag-tape unit and appends the characters to those already in the target string. If the target string variable is undefined (not used before) or a null string (LEN 0), then the operation is the same as TAPEIN; in other words, it does the equivalent of an "assignment" since there is nothing to append to.

TAPEAPP does not terminate input when it sees a carriage return character. Input is terminated when the End-of-File is encountered or the string variable becomes full.

Appendix A has a program example using TAPEAPP.

If the string variable is not found in the CALL statement, then Graphic System error 12 will be issued.

TAPEIN [4052R14 Option 1A only]

Purpose

To read an entire ASCII file into a character string using the internal mag-tape unit.

Syntax Form

[line number] CALL "TAPEIN", < string variable >

Descriptive Form

[line number] CALL "TAPEIN",target for mag-tape file

Example

```
120 FIND 1
130 CALL "TAPEIN",File$ ! Input ASCII file into File$.
```

Explanation

NOTE

This routine is included only in the 4052R14 Option 1A ROM Pack.

The TAPEIN routine inputs an ASCII data or program file using the internal mag-tape unit and stores it into the string variable. TAPEIN does not terminate input when it sees a carriage return character. Input is terminated when the End-of-File is found or the string variable becomes full.

Appendix A has a program example using TAPEIN.

If a string variable is not found in the CALL statement, then Graphic System error 12 will be issued.

THEADER

Purpose

This routine finds and opens the file where the tape head is currently positioned and returns the file header or number.

Syntax Form

[line number] CALL"THEADER" { [, <numeric variable>] / [, <string variable>] }

Descriptive Form

[line number] CALL"THEADER" [, target for header or file number]

Examples

```
100 CALL"THEADER"

200 CALL"THEADER", T$

300 CALL"THEADER", F
```

Explanation

If the argument is

- not specified, the file header is printed on the screen.
- a string variable, the entire file header is returned in the variable (or as much of it as will fit in the string). The header is not printed on the screen.
- a numeric variable, the file number is returned in the variable. The file number is not printed on the screen.

The THEADER routine will close a mag tape file that has been opened before reading the file header. If the "NO HEADER" mode has been selected in the mag tape environmental parameters, the THEADER routine issues Graphic System error message 60 for 4051R14 and 4052R14 ROM Pack, and issues message 61 for the 4052R14 Option 1A ROM Pack.

TNAME

Purpose

This routine assigns a name to a file on mag tape.

Syntax Form

[line number] CALL"TNAME" , { <string constant> / <string variable> }

Descriptive Form

[line number] CALL"TNAME" , file name

Examples

```
100 CALL"TNAME", "PROM BURN"

200 CALL"TNAME", T$
```

Explanation

The TNAME routine writes a file name to the header of the file where the tape head is currently positioned. The file to be named must first be opened with a FIND command. A file identified as LAST cannot be named; a file identified as NEW cannot be named until data is stored in the file. TNAME will close a mag tape file that has been opened before entering the file name.

The argument specifies a string variable or string constant containing a name to be entered in the tape header.

- If a "SECRET" file is to be named, the name is truncated to 5 characters and left-justified in the 11-character field.
- Non-SECRET file names are truncated to 11 characters, if necessary, and right-justified in the 11-character field.

The reader is advised to limit file names to 6 characters or less. This will prevent the name from being overwritten by subsequent commands or routines that write data on the tape (e.g., SAVE, PRINT, or WRITE).

The name may contain any characters except those found in numbers (digits, decimal point, +, or -). If an "S" is found as the sixth from the last character in the name of a non-SECRET file, it is changed to an "s", since an "S" in that position makes the file secret.

The following Graphic System error messages may be issued in connection with the TNAME routine:

Message Number	Message
12	An incorrect (or no) argument was specified.
36	The variable is undefined.
55	The mag tape file is not open.
60	The name contains a digit, decimal point, + or -, OR the mag tape system has been set to "NO HEADER" mode.
61	The file is NEW or LAST.

TRIM [4052R14 Option 1A]

Purpose

To change the current logical length of a string variable without altering or deleting any of the data within the string variable.

Syntax Form

[line number] CALL "TRIM", <string variable>, <numeric expression>

Descriptive Form

[line number] CALL "TRIM", string to be altered, new length

Example

EXAMPLE 1 General example:

```
100 CALL "TRIM",AS,S*13
```

EXAMPLE 2 Specific changes (see text for more explanation):

```
100 INIT
110 A$ = "HELLO THERE"
120 CALL "TRIM",AS,5
130 CALL "TRIM",AS,11
140 CALL "TRIM",AS,50
```

Explanation

NOTE

This routine is included only in the 4052R14 Option 1A ROM Pack.

The TRIM routine is used to change the current logical length (as opposed to the dimensioned length) of a defined string variable. The range for the new length is between 1 and the dimensioned length of the string variable.

In example 2, A$ has an initial logical length after line 110 of 11 characters (LEN(A$) = 11). After line 120, A$ has a length of 5 and contains the characters "HELLO". However, none of the character positions have been altered. Line 130 restores A$ to 11 characters and they still have the value "HELLO THERE". Line 140 pushes the logical length past 11 characters to a logical length of 50. The original 11 characters are still there, but characters 12 through 50 are undefined, that is, they are whatever happens to be in memory at the time line 140 is executed. It is not intended that TRIM will be used to push the logical length of a string variable past "defined" limits.

A practical use for TRIM might be to output large amounts of string data when memory is in short supply for a given application. For instance, in the following examples, A$ has a logical length between 25,000 and 26,000 characters. It is desired to print the first 5000 characters OF A$ to file 1, using the internal mag-tape unit, without altering the data in A$. And, there is only 2000 bytes of memory available. Method 1 will give a MEMORY FULL error because of line 100, but method 2 will work.

METHOD 1: Use the SEG function

```
100 DIM B$(5000)
110 B$ = SEG(A$,1,5000)
120 FIND 1
130 PRINT @33 B$
```

METHOD 2: Use the TRIM routine

```
100 L = LEN(A$)
110 CALL "TRIM",A$,5000 ! Trim A$ to 5000 characters
120 FIND 1
130 PRINT @33 A$
140 CALL "TRIM",A$,L ! Restore A$ to its original length
```

Appendix A has a program example using TRIM.

Improper use of TRIM will cause errors. If the length parameter specified in the CALL statement is greater than the dimensioned length of the string variable, then error 12 results. If the specified string variable is undefined, then error 36 results.

UNDEF

Purpose

This routine determines whether a variable is defined without issuing an error message or interrupting system operation.

Syntax Form

[line number] CALL "UNDEF" { <array> / <string variable> / <numeric variable> } , <numeric variable>

Descriptive Form

[line number] CALL "UNDEF", variable to be tested, target for the result

Example

```
1 CALL "UNDEF", A, B
2 IF NOT(B) THEN 200
3 GOTO 100
100 REM CODE EXECUTED FIRST TIME ONLY
200 REM MAIN PROGRAM
```

Explanation

The first argument specifies the variable to be tested. This may be a numeric variable, a string variable, an array, or an array element.

The second argument is a variable that returns the result. It is set to:

0 if the variable is defined.

1 if the variable is undefined.

An array is undefined if any element in the array is undefined.

This function is useful in programs that are executed many times, but in which initialization is only desired the first time (for example, to prevent loss of data from previous executions).

UNL

Purpose

This routine UNListens all devices on the bus.

Syntax Form

[line number] CALL 'UNL'

Descriptive Form

[line number] CALL 'UNL'

Example

```
100 CALL 'UNL'
```

Explanation

This routine sends the unlisten (UNL) interface message (ATN asserted, with DEC 63 or X'3F') to all devices on the bus.

UNT

Purpose

This routine untalks all devices on the bus

Syntax Form

[line number] CALL "UNT"

Descriptive Form

[line number] CALL "UNT"

Examples

150 CALL "UNT"

Explanation

This routine sends the untalk (UNT) interface message (ATN asserted, with DEC 95 or X 5F) to all devices on the bus

VARCLR

Purpose

The VARCLR routine clears certain bits in the target variable or array specified by the first argument. The bits to be cleared are defined by the binary value of the second argument.

Syntax Form

[line number] CALL"VARCLR", { <array> / <numeric variable> }, <numeric expression>

Descriptive Form

[line number] CALL"VARCLR", variable to be cleared, bits to clear

Examples

EXAMPLE 1

```
100 REM CLEAR THE"BUSY" BIT IN A STATUS BYTE
110 CALL"VARCLR", A, 16
```

EXAMPLE 2

```
200 CALL"VARCLR", B(I), 127
```

EXAMPLE 3

```
300 A=123
310 C=12
320 CALL"VARCLR", A, C
330 PRINT A

    A=115
```

Explanation

The first argument specifies the variable or array in which bits are to be cleared. If an array is specified, the operation is performed on all elements of the array (all array elements must be defined). The second argument contains a numeric value which is used as a 16-bit mask. The complement of the mask is logically ANDed with the 16-bit value of the target and the result is stored in the target variable.

Any value from −32768 to 65535 can be used for input. Output is in the range 0 through 65535

Common values to test for in GPIB serial poll status bytes are shown as follows.

CONDITION	BITS	Variable Value	
		DECIMAL	HEX
Busy	5	16	10
Abnormal	6	32	20
RQS	7	64	40
Device Dependent	8	128	80
RQS and Busy	5,7	80	50
RQS and Device Dependent	7,8	192	C0

VARSET

Purpose

This routine sets certain bits in the target variable or array, as specified in the first argument. The bits to be set are defined by the binary value of the second argument.

Syntax Form

[line number] CALL "VARSET", { <array> / <numeric variable> }, <numeric expression>

Descriptive Form

[line number] CALL "VARSET", variable to be set, bits to set

Examples

EXAMPLE 1

```
100 REM SET BIT 4 IN THE BINARY VALUE OF A
110 CALL "VARSET", A, 8
```

EXAMPLE 2

```
200 REM SET BITS 2 and 3 IN THE VALUE OF B(I)
210 CALL "VARSET", B(I), 6
```

Explanation

The first argument specifies the variable or array that serves as the target for the bit-set operation. If an array is specified, the operation is performed on all elements of the array. (All elements of the array must be defined.)

The second argument defines a numeric value whose binary equivalent is used as the bit mask. The mask is logically inclusive-ORed with the 16-bit binary value of the target variable, and the result is stored in the target variable.

Any value from −32768 to 65535 can be used for input. Outputs are in the range from 0 to 65535.

EXAMPLE

```
CALL "VARSET", A, B
```

Values of the Variables

VARIABLE	BINARY	HEX	DECIMAL
A	0000 0010 1011 0100	02B4	692
B	0000 0010 1001 0001	0291	657
NEW A	0000 0010 1011 0101	02B5	693

Note that in variable B, only bit 1 was not already set (compared to variable A). Accordingly the value of NEW A only increased by 1.

VARTST

Purpose

This routine makes a bit-by-bit comparison between the values of the first and second arguments. It then returns the result of this comparison in the third argument.

Syntax Form

[line number] CALL"VARTST" ,< numeric variable> , < numeric expression>, < numeric variable>

Descriptive Form

[line number] CALL"VARTST", variable to be tested, bits to be tested, target for test result

Examples

EXAMPLE 1

```
100 CALL"VARTST", A(7), 4, Z

110 IF Z THEN 200
```

EXAMPLE 2

```
200 REM ROUTINE FOR COMPARISON
500 CALL"VARTST", A, 3, Z
510 IF Z= 3 THEN 530
600 J= 693
610 B= 17
620 CALL"VARTST", J, B, M
630 PRINT M

    M= 17
```

Explanation

The first argument contains the value to be tested; the second argument defines the bits to be tested and the third argument returns the results.

This routine makes two 16-bit masks from the decimal values of the first and second arguments. These masks are logically ANDed and the result of the operation is returned in the third argument.

Common values to test for in GPIB serial poll status bytes are as follows.

CONDITION	Variable Value		
	BITS	DECIMAL	HEX
Busy	5	16	10
Abnormal	6	32	20
RQS	7	64	40
Device Dependent	8	128	80
RQS and Busy	5,7	80	50
RQS and Device Dependent	7,8	192	C0

VLIST [Version 1: 4051R14 and 4052R14 only]

Purpose

This routine lists all current BASIC variables and their values on the screen.

Syntax Form

[line number] CALL"VLIST"

Descriptive Form

[line number] CALL"VLIST"

Example

```
100 CALL "VLIST"
```

Explanation

This routine lists the current variables with the following information:

Variable Type	Information Listed
Numeric	Value, if defined.
	Not listed if not defined.
Array	Dimension and ALL DEFINED if every element is defined.
	Dimension and UNDEF if any element is undefined.
String	If defined, includes the dimension, the bytes currently used, and the first 32 bytes of the string.
	UNDEF, if undefined and accessed.
	Not listed if undefined and not accessed.

VLIST [Version 1: 4051R14 and 4052R14 only]

Purpose

This routine lists all current BASIC variables and their values on the screen.

Syntax Form

[line number] CALL"VLIST"

Descriptive Form

[line number] CALL"VLIST"

Example

```
100 CALL "VLIST"
```

Explanation

This routine lists the current variables with the following information:

Variable Type	Information Listed
Numeric	Value, if defined.
	Not listed if not defined.
Array	Dimension and ALL DEFINED if every element is defined.
	Dimension and UNDEF if any element is undefined.
String	If defined, includes the dimension, the bytes currently used, and the first 32 bytes of the string.
	UNDEF, if undefined and accessed.
	Not listed if undefined and not accessed.

Explanation

NOTE

This explanation is applicable to Version 2: 4052R14 Option 1A ROM Pack only.

The VLIST routine generates the following information:

Variable Type	Information Listed
Numeric	Value, if defined; not listed if not defined.
Array	Dimension and "ALL DEFINED" if every element is defined; dimension and "UNDEF" if any element is undefined.
String	If defined, includes the dimensioned size, the current length, and a calculated number of characters of the string (number of characters depends on string length, screen size, and font size); dimension and "UNDEF" if undefined and dimensioned; no information if undefined and undimensioned.

The generated information is output to the screen if the optional string variable is omitted; it is stored in the string variable and not output to the screen if the variable is present. If the string variable is not dimensioned large enough to hold all the information, no error results, but information is lost.

The algorithm used to generate the information is based on output to the screen, independent of whether the information is going to the screen or to the variable.

WAIT

Purpose

This routine delays the execution of the program by a specified length of time.

Syntax Form

[line number] CALL"WAIT" [, <numeric expression>]

Descriptive Form

[line number] CALL"WAIT" [, delay time]

Examples

```
100 CALL"WAIT", 100
```

Explanation

The argument specifies the number of seconds to delay. The range of this argument is from 0 to 1E07 seconds, with a resolution of 0.01 seconds. The default WAIT value is 1E07 seconds, the maximum value. If 0 is specified, the program continues without interruption. If the argument is out of range, Graphic System error message 12 is issued.

The WAIT routine is terminated if an interrupt occurs.

NOTE

This routine is implemented in the 4051R14 ROM Pack only. The routine is implemented internally in the 4052 and 4054 Graphic Systems.

WBIN

Purpose

The WBIN routine sends binary data from the Graphic System to devices on the bus.

Syntax Form

[line number] CALL"WBIN" { [, <string constant>] / [, <string variable>] } , { <array> / <string variable> } , <numeric variable> [; <address list>]

Descriptive Form

[line number] CALL"WBIN" [, output mode] , source of data, target for error code [; listen address(es)]

Examples

```
100 CALL "WBIN", Q, E; 5

200 CALL "WBIN", "NEOI", A$, E; 3, 2; 7; 14

300 CALL "WBIN", "PACK,NEOI", C, E; 24
```

Explanation

The first argument to the WBIN routine is optional. If used it indicates how the binary data is transmitted on the bus. This argument may have the following values:

UNPK — Unpacked mode. One byte is output for each array element. The array element is assumed to be an integer value between 0 and 255. Non-integer values are rounded. Values less than zero or larger than 255 cause error code 10 to be returned and no data to be transmitted.

PACK Packed mode. Two bytes are output for each array element. The high-order byte is transmitted first. Negative values are assumed to be 16-bit two's complement numbers. Values less than −32768 or greater than 65535 cause error code 10 to be returned and no data to be transmitted.

NEOI No EOI. Indicates that the last data byte should not be transmitted with EOI. If this argument is omitted, the last byte is transmitted with EOI.

The user may specify both UNPK and NEOI in one routine as "UNPK,NEOI".

The source argument specifies the variable from which the data is taken. This argument may be either an array or a string variable. If an array is specified, one or two bytes are output for each array element, depending on the output mode selected. The array element is assumed to be an integer value between −32768 and 65535 for packed mode, and 0 to 255 for unpacked mode. Non-integer values are rounded.

If a string argument is specified, the data is assumed to be in ASCII HEX format. Two bytes in the string store the value of each byte transmitted over the bus. If the length of the source string is odd, the last (odd) byte is ignored and error code 11 is returned.

The next argument is a variable which indicates the following conditions on completion of the routine.

0 No errors.

10 Data in source variable out of range. The block is not transmitted. Instead, the address is sent, followed by UNT and UNL, to clear the bus.

11 Source string length is odd.

The last argument specifies the address(es) of the device(s) receiving the binary data. This address list is optional. If the list is specified, WBIN sends UNTalk and UNListen after sending the data. If the address list is omitted, WBIN assumes that the listening devices have been previously addressed and does not transmit UNT and UNL after sending the data.

If an error in one of the argument specifications is detected, the execution of the routine is canceled and Graphic System error message 12 is issued.

Section 3

REPLACEABLE PARTS

PARTS ORDERING INFORMATION

Replacement parts are available from or through your local Tektronix, Inc. Field Office or representative.

Changes to Tektronix instruments are sometimes made to accommodate improved components as they become available, and to give you the benefit of the latest circuit improvements developed in our engineering department. It is therefore important, when ordering parts, to include the following information in your order: Part number, instrument type or number, serial number, and modification number, if applicable.

If a part you have ordered has been replaced with a new or improved part, your local Tektronix, Inc. Field Office or representative will contact you concerning any change in part number.

Change information, if any, is located at the rear of this manual.

SPECIAL NOTES AND SYMBOLS

X0000 Part first added at this serial number.

0X0 Part removed after this serial number.

FIGURE AND INDEX NUMBERS

Items in this section are referenced by figure and index numbers to the illustrations.

INDENTATION SYSTEM

This mechanical parts list is indented to indicate item relationships. Following is an example of the indentation system used in the description column.

```
1  2  3  4  5                    Name & Description

Assembly and or Component
Attaching parts for Assembly and or Component
.
   Detail Part of Assembly and or Component
   Attaching parts for Detail Part
   .
      Parts of Detail Part
      Attaching parts for Parts of Detail Part
      .
```

Attaching Parts always appear in the same indentation as the item it mounts, while the detail parts are indented to the right. Indented items are part of, and included with, the next higher indentation. The separation symbol "." indicates the end of attaching parts.

Attaching parts must be purchased separately, unless otherwise specified.

ITEM NAME

In the Parts List, an Item Name is separated from the description by a colon (:). Because of space limitations, an Item Name may sometimes appear as incomplete. For further Item Name identification, the U.S. Federal Cataloging Handbook H6-1 can be utilized where possible.

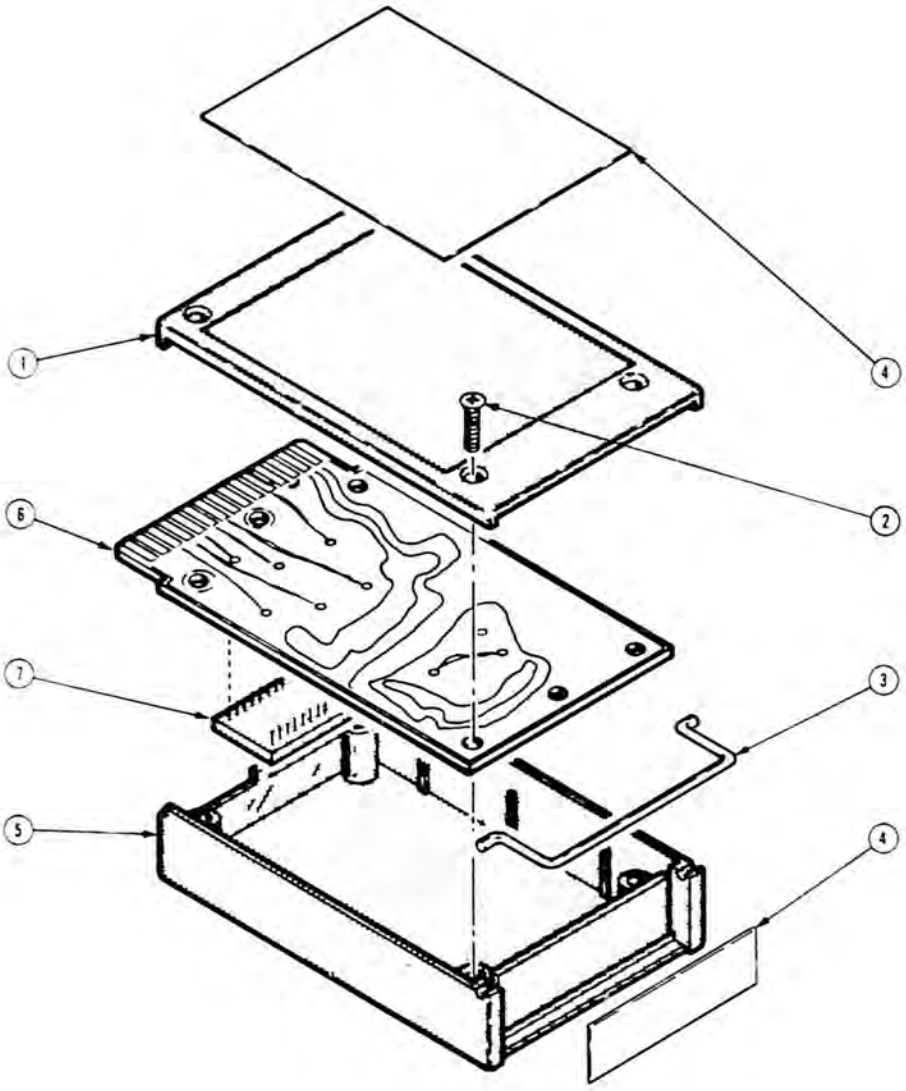
ABBREVIATIONS

ASSEMBLY	ASSEMBLED	ELECTRON	ELECTRONIC	NUT	NUT	ASSEMBLY	ASSEMBLED
ATTACHING	ATTACHING	ELECTRICAL	ELECTRICAL	NYLON	NYLON	ELECTRON	ELECTRON
ADAPTER	ADAPTER	ELECTROLYTIC	ELECTROLYTIC	NYLAND	NYLANDERSENT	SEMICONDUCTOR	SEMICONDUCTOR
ALUMINUM	ALUMINUM	ELEM	ELEMENT	OSCILLATOR	OSCILLATOR	SHIELD	SHIELD
ASSEMBLY	ASSEMBLY	INTERNAL	INTERNAL	OVAL	OVAL	SHOULDERED	SHOULDERED
AMPLIFIER	AMPLIFIER	ELECTRICAL PARTS LIST	ELECTRICAL PARTS LIST	OVERALL	OVERALL	SOCKET	SOCKET
ANGLE	ANGLE	EQUIPMENT	EQUIPMENT	OVERHEAD	OVERHEAD	SOLDER	SOLDER
ASSEMBLY	ASSEMBLY	EXTERNAL	EXTERNAL	OXIDE	OXIDE	SPECIAL	SPECIAL
ASSEMBLIES	ASSEMBLIES	FILLISTER HEAD	FILLISTER HEAD	PACKING	PACKING	SPRING	SPRING
ATTENUATOR	ATTENUATOR	FLEXIBLE	FLEXIBLE	PANEL	PANEL	SQUARE	SQUARE
BALL	BALL	FLAT HEAD	FLAT HEAD	PLASTIC	PLASTIC	STAINLESS STEEL	STAINLESS STEEL
BEARING	BEARING	FILLISTER	FILLISTER	PART NUMBER	PART NUMBER	STEEL	STEEL
BRACKET	BRACKET	FRAME ASSEMBLY	FRAME ASSEMBLY	PAN HEAD	PAN HEAD	SWITCH	SWITCH
BRASS	BRASS	FASTENER	FASTENER	PHOSPHOR BRONZE	PHOSPHOR BRONZE	TUBE	TUBE
BEVEL	BEVEL	FIBER	FIBER	PLAIN	PLAIN	TERMINAL	TERMINAL
BOARD	BOARD	GASKET	GASKET	PLATE	PLATE	THREAD	THREAD
BRONZE	BRONZE	GLASS	GLASS	PLASTIC	PLASTIC	THICK	THICK
CABINET	CABINET	HANDLE	HANDLE	PART NUMBER	PART NUMBER	TENSION	TENSION
CADMIUM	CADMIUM	HEXAGON	HEXAGON	PAN HEAD	PAN HEAD	TERMINAL	TERMINAL
CAPACITOR	CAPACITOR	HEXAGONAL HEAD	HEXAGONAL HEAD	POWER	POWER	THICK	THICK
CERAMIC	CERAMIC	HEXAGONAL SOCKET	HEXAGONAL SOCKET	RECTANGULAR	RECTANGULAR	TRUSS HEAD	TRUSS HEAD
CHASSIS	CHASSIS	HEXAGONAL SOCKET HEAD	HEXAGONAL SOCKET HEAD	RECEPTACLE	RECEPTACLE	TOLERANCE	TOLERANCE
CIRCUIT	CIRCUIT	HEX HD EXTRA THICK	HEX HD EXTRA THICK	RIVET	RIVET	VOLTAGE	VOLTAGE
CLAMP	CLAMP	HEX HD EXTRA WIDE	HEX HD EXTRA WIDE	ROUND	ROUND	VARIABLE	VARIABLE
COMPOSITION	COMPOSITION	HEX SOCKET	HEX SOCKET	RUBBER	RUBBER	WITH	WITH
CONNECTOR	CONNECTOR	HEX HD EXTENSION	HEX HD EXTENSION	SCREW	SCREW	WASHER	WASHER
CERAMIC	CERAMIC	HIGH VOLTAGE	HIGH VOLTAGE	SECTION	SECTION	WITHOUT	WITHOUT
COPPER	COPPER	INTEGRATED CIRCUIT	INTEGRATED CIRCUIT	SELECTED	SELECTED	WOOD	WOOD
CAPACITOR	CAPACITOR	INSIDE DIAMETER	INSIDE DIAMETER	SEMICONDUCTOR	SEMICONDUCTOR	WIRE	WIRE
CERAMIC	CERAMIC	IDENTIFICATION	IDENTIFICATION	SHIELD	SHIELD	WIREWOUND	WIREWOUND
CARBON	CARBON	MICA	MICA	SILICON	SILICON	XFORMER	XFORMER

Tektronix Part No	Serial/Model No Eff	Dscont	Qty	1 2 3 4 5	Name & Description	Mfr Code	Mfr Part Number
380-0384-01			1		HSG HALF ROM PK LID ABS	80009	380-0384-01
					..... (ATTACHING PARTS)		
211-0707-00			4		SCREW MACHINE 4-40 X 0.500 FLH STL	83385	ORD BY DESCR
					..... (END ATTACHING PARTS)		
367-0180-00			1		HANDLE BOW 2.62 L SST	91260	OBD
			1		MKR SET IDENT		
380-0343-01			1		HSG HALF PTR INNER ABS	80009	380-0343-01
			1		CKT BOARD ASSY (SEE A1, A2, OR A3 REPL)		
136-0761-00			2		SKT PL-IN ELEK MICROCKT 24 PIN (4051R14 ONLY)	09922	DILB24P108
136-0761-00			4		SKT PL-IN ELEK MICROCKT 24 PIN (4052R14 & R141A ONLY)	09922	DILB24P108
					STANDARD ACCESSOIRES		
070-4316-01			1		MANUAL TECH INSTRUCTION	80009	070-4316-01
070-4315-01			1		REF 4050R14 GPIB	80009	070-4315-01

Component No.	Tektronix Part No.	Serial/Model No. Eff	Dscont	Name & Description	Mfr Code	Mfr Part Number
				4051R14		
A1	670-7631-00			CKT BOARD ASSY 8K PROM PACK	80009	670-7631-00
A1C115	283-0422-00			CAP FXD CER DI 0.047UF +80-20% 50V	04222	DG015E473Z
A1C205	283-0422-00			CAP FXD CER DI 0.047UF +80-20% 50V	04222	DG015E473Z
A1C305	283-0422-00			CAP FXD CER DI 0.047UF +80-20% 50V	04222	DG015E473Z
A1C315	283-0422-00			CAP FXD CER DI 0.047UF +80-20% 50V	04222	DG015E473Z
A1C401	290-0804-00			CAP FXD ELCTLT 10UF +50-10% 25V	55680	ULA1E100TEA
A1C415	283-0422-00			CAP FXD CER DI 0.047UF +80-20% 50V	04222	DG015E473Z
A1U105	156-0916-02			MICROCIRCUIT DI 8-2 INP 3-STATE BFR BURN	27014	DM81LS97A
A1U115	160-1636-00			MICROCIRCUIT DI 4096 X 8 EPROM PRGM	80009	160-1636-00
A1U205	156-0916-02			MICROCIRCUIT DI 8-2 INP 3-STATE BFR BURN	27014	DM81LS97A
A1U305	156-0480-02			MICROCIRCUIT DI QUAD 2 INP & GATE	01295	SN74LS08NP3
A1U315	160-1637-00			MICROCIRCUIT DI 4096 X 8 EPROM PRGM	80009	160-1637-00
A1U405	156-0469-02			MICROCIRCUIT DI 3 8 LINE DCDR	01295	SN74LS138NP3
				4052R14		
A2	670-7632-00	B010100	B010281	CKT BOARD ASSY CHARACTER SYMBOL ROM PACK	80009	670-7632-00
A2	670-7632-01	B010282		CKT BOARD ASSY CHARACTER SYMBOL ROM PACK	80009	670-7632-01
A2C[illegible]	283-0422-00			CAP FXD CER DI 0.047UF +80-20% 50V	04222	DG015E473Z
A2C[illegible]3	283-0422-00			CAP FXD CER DI 0.047UF +80-20% 50V	04222	DG015E473Z
A2C[illegible]01	290-0804-00			CAP FXD ELCTLT 10UF +50-10% 25V	55680	ULA1E100TEA
A2C[illegible]15	283-0422-00			CAP FXD CER DI 0.047UF +80-20% 50V	04222	DG015E473Z
A2R[illegible]	315-0202-00	B010100	B010281	RES FXD CMPSN 2K OHM 5% 0.25W	01121	CB2025
A2U[illegible]	160-1638-00			MICROCIRCUIT DI 4096 X 8 EPROM PRGM	80009	160-1638-00
A2U[illegible]	160-1639-00			MICROCIRCUIT DI 4096 X 8 EPROMPRGM	80009	160-1639-00
A2U[illegible]	156-0469-02			MICROCIRCUIT DI 3 8 LINE DCDR	01295	SN74LS138NP3
				4052R14A		
A3	670-8122-00			CKT BOARD ASSY GPIB ENHANCEMENT ROM PACK	80009	670-8122-00
A3C[illegible]	283-0422-00			CAP FXD CER DI 0.047UF +80-20% 50V	04222	DG015E473Z
A3C[illegible]3	283-0422-00			CAP FXD CER DI 0.047UF +80-20% 50V	04222	DG015E473Z
A3C101	290-0804-00			CAP FXD ELCTLT 10UF +50-10% 25V	55680	ULA1E100TEA
A3C[illegible]	283-0422-00			CAP FXD CER DI 0.047UF +80-20% 50V	04222	DG015E473Z
A3U[illegible]	156-0469-02			MICROCIRCUIT DI 3 8 LINE DCDR	01295	SN74LS138NP3
A3[illegible]	160-2026-00			MICROCIRCUIT DI 4096 X 8 EPROM PRGM	80009	160-2026-00
A3[illegible]	160-2027-00			MICROCIRCUIT DI 4096 X 8 EPROM PRGM	80009	160-2027-00
A3[illegible]	160-2043-00			MICROCIRCUIT DI 4096 X 8 EPROM PRGM	80009	160-2043-00

Fig. & Index No.	Tektronix Part No.	Serial/Model No. Eff	Dscont	Qty	1 2 3 4 5	Name & Description	Mfr Code	Mfr Part Number
1	380-0384-01			1		HSG HALF,ROM PK LID,ABS	80009	380-0384-01
						·····(ATTACHING PARTS)·····		
2	211-0102-00			4		SCREW,MACHINE 4-40 X 0.500",FLH,STL	83385	ORD BY DESCR
						·····(END ATTACHING PARTS)·····		
3	367-0189-00			1		HANDLE,BOW 2.62 L,SST	91260	OBD
4				1		MKR SET,IDENT		
5	380-0343-01			1		HSG,HALF,PTR,INNER,ABS	80009	380-0343-01
6				1		CKT BOARD ASSY (SEE A1, A2, OR A3 REPL)		
*	136-0751-00			2		SKT,PL-IN ELEK,MICROCKT,24 PIN	09922	DILB24P108
						(4051R14 ONLY)		
	136-0751-00			4		SKT,PL-IN ELEK,MICROCKT,24 PIN	09922	DILB24P108
						(4052R14 & R141A ONLY)		
						STANDARD ACCESSOIRES		
	070-4316-01			1		MANUAL,TECH INSTRUCTION	80009	070-4316-01
	070-4315-01			1		REF,4050R14 GPIB	80009	070-4315-01



Section 4

SCHEMATICS

Symbols and Reference Designators

Electrical components shown on the diagrams are in the following units unless noted otherwise.

Capacitors — Values one or greater are in picofarads (pF).
Values less than one are in microfarads (μF).

Resistors — Ohms (Ω).

Graphic symbols and class designation letters are based on ANSI Standard Y32.2-1975.

Logic symbology is based on ANSI Y32.14-1973 in terms of positive logic. Logic symbols depict the logic function performed and may differ from the manufacturer's data.

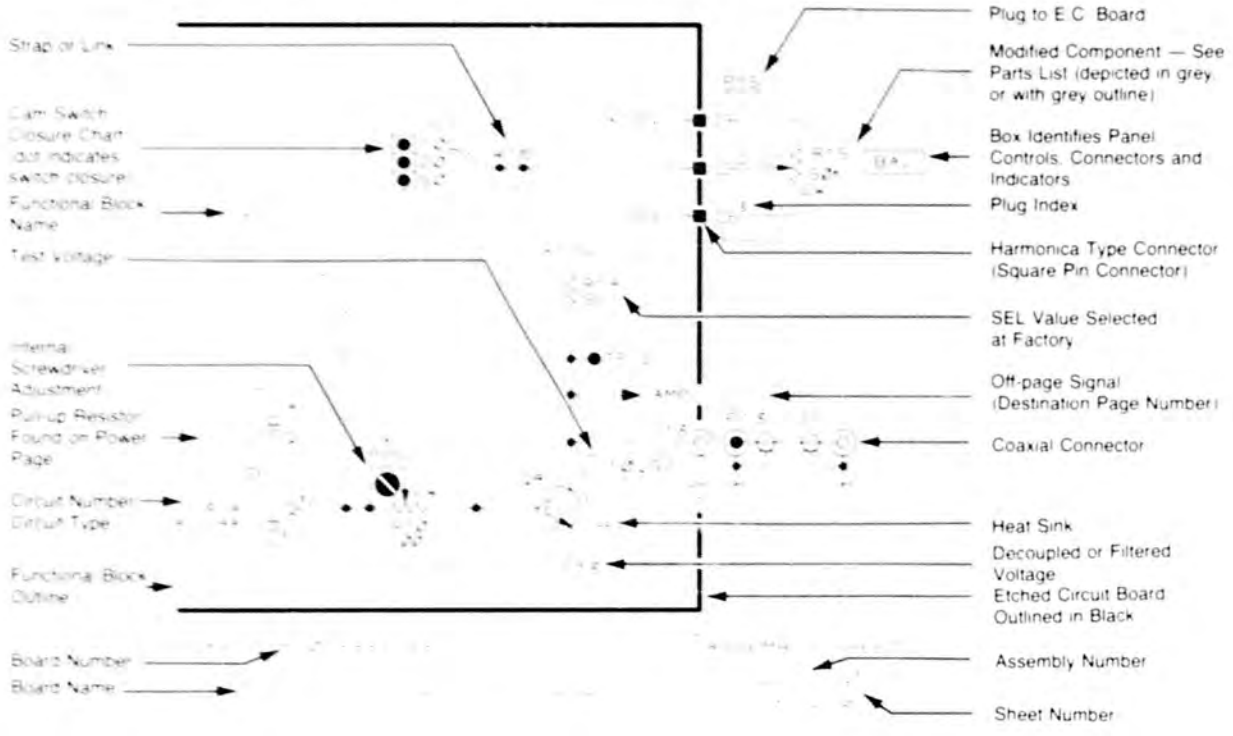
Abbreviations are based on ANSI Y1.1-1972. Other ANSI standards that are used in the preparation of diagrams by Tektronix, Inc., are:

Y14.15-1966	Drafting Practices
Y14.2-1973	Line Conventions and Lettering
Y10.5-1968	Letter Symbols for Quantities Used in Electrical Science and Electrical Engineering

The following prefix letters are used as reference designators to identify components or assemblies on the diagrams.

A	Assembly, separable or repairable (circuit board, etc.)	H	Heat dissipating device (heat sink, heat radiator, etc.)	S	Switch or contactor
AT	Attenuator, fixed or variable	HR	Heater	T	Transformer
B	Motor	HY	Hybrid circuit	TC	Thermocouple
BT	Battery	J	Connector, stationary portion	TP	Test point
C	Capacitor, fixed or variable	K	Relay	U	Assembly, inseparable or non-repairable (integrated circuit, etc.)
CB	Circuit breaker	L	Inductor, fixed or variable	V	Electron tube
CR	Diode, signal or rectifier	M	Meter	VR	Voltage regulator (zener diode, etc.)
DL	Delay line	P	Connector, movable portion	W	Wirestrap or cable
DS	Indicating device (lamp)	Q	Transistor or silicon-controlled rectifier	Y	Crystal
E	Spark Gap, Ferrite bead	R	Resistor, fixed or variable	Z	Phase shifter
F	Fuse	RT	Thermistor		
FL	Filter				

The following special symbols may appear on the diagrams:



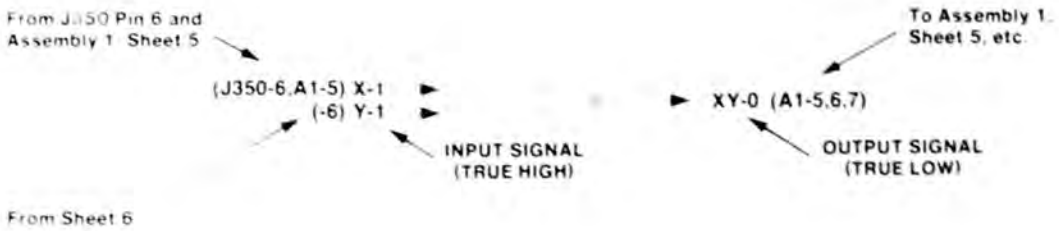
1. True High and True Low Signals

Signal names on the schematics are followed by -1 or a -0. A TRUE HIGH signal is indicated by -1, and a TRUE LOW signal is indicated by -0.

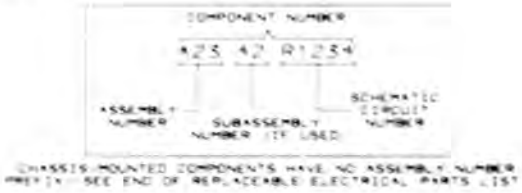
SIGNAL -1 TRUE HIGH
SIGNAL -0 TRUE LOW

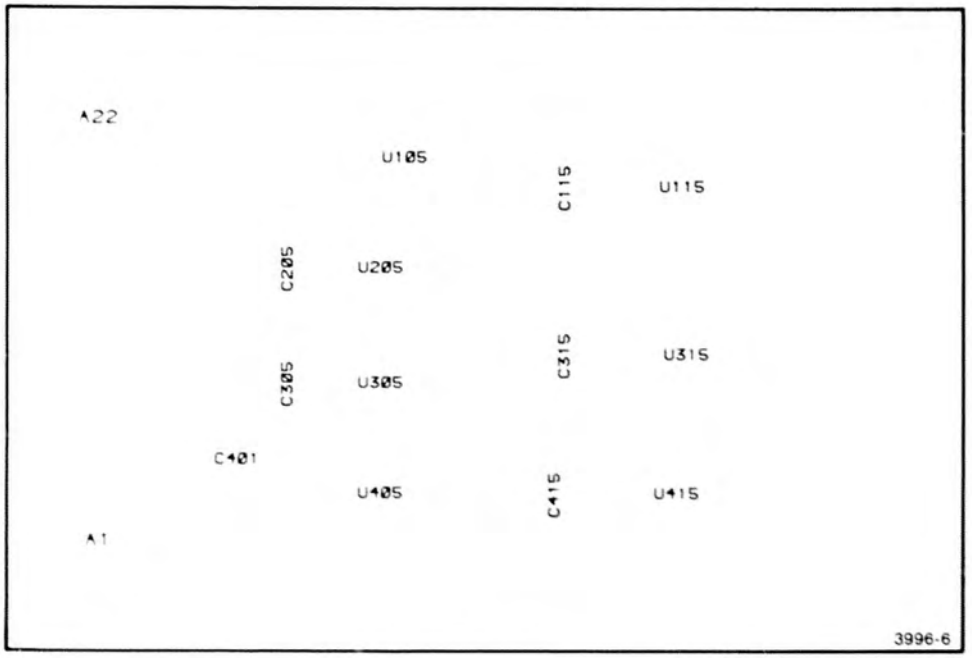
2. Cross-References

Schematic cross-references (from-to information) are included on the schematics. The "from" reference only indicates the signal source, but the "to" reference lists all loads where the signal is used. All from-to information will be enclosed in parentheses.

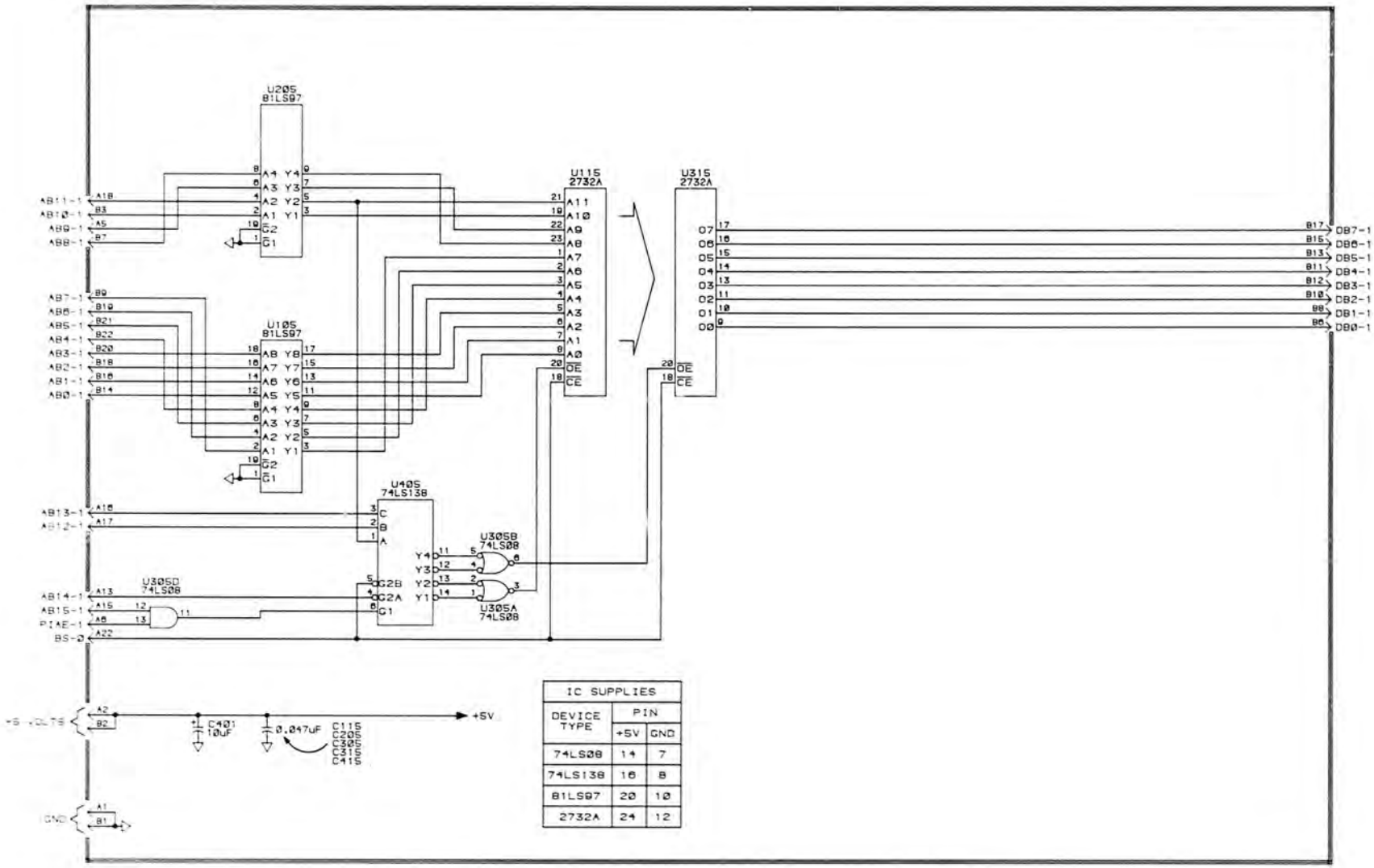


3. Component Number Example





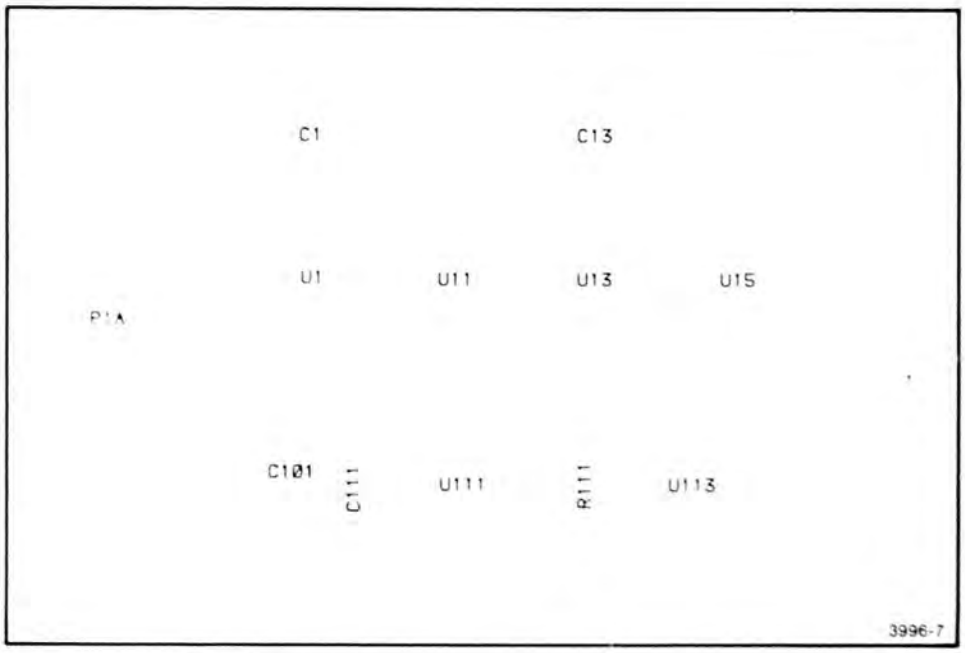
4051R14 (670-7631-00) Component Locations.



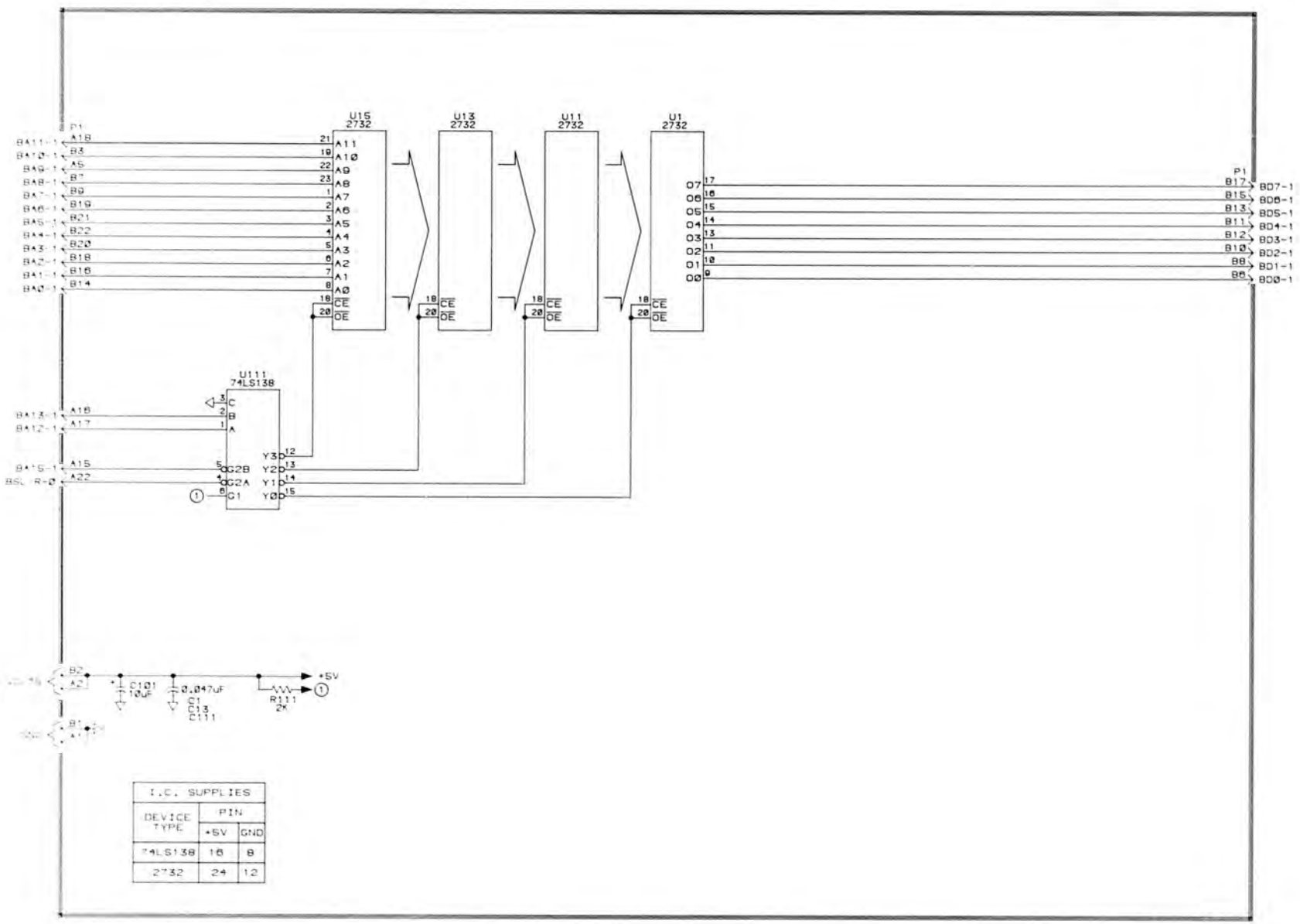
IC SUPPLIES		
DEVICE TYPE	PIN	
	+5V	GND
74LS08	14	7
74LS138	16	8
81LS97	20	10
2732A	24	12

INSTRUMENT:	NOTES:	BOARD: 670-7631-00 8K ROM PACK	ASSEMBLY-SHEET:
4051R14	REV. JAN 1983 4316-S1	GPIB ENHANCEMENT ROM PACK	A1-1 (1 OF 1)

4051R14 670-7631-00 A1-1



4052R14 (670-7632-00,01) Component Locations.



I.C. SUPPLIES		
DEVICE TYPE	PIN	
	+5V	GND
74LS138	16	8
2732	24	12

NOTES: KEYSLOT LOCATED BETWEEN PINS A6 & A7.

4316-S2

BOARD: 670-7632-00 16K ROM PACK

GPIB ENHANCEMENT ROM PACK

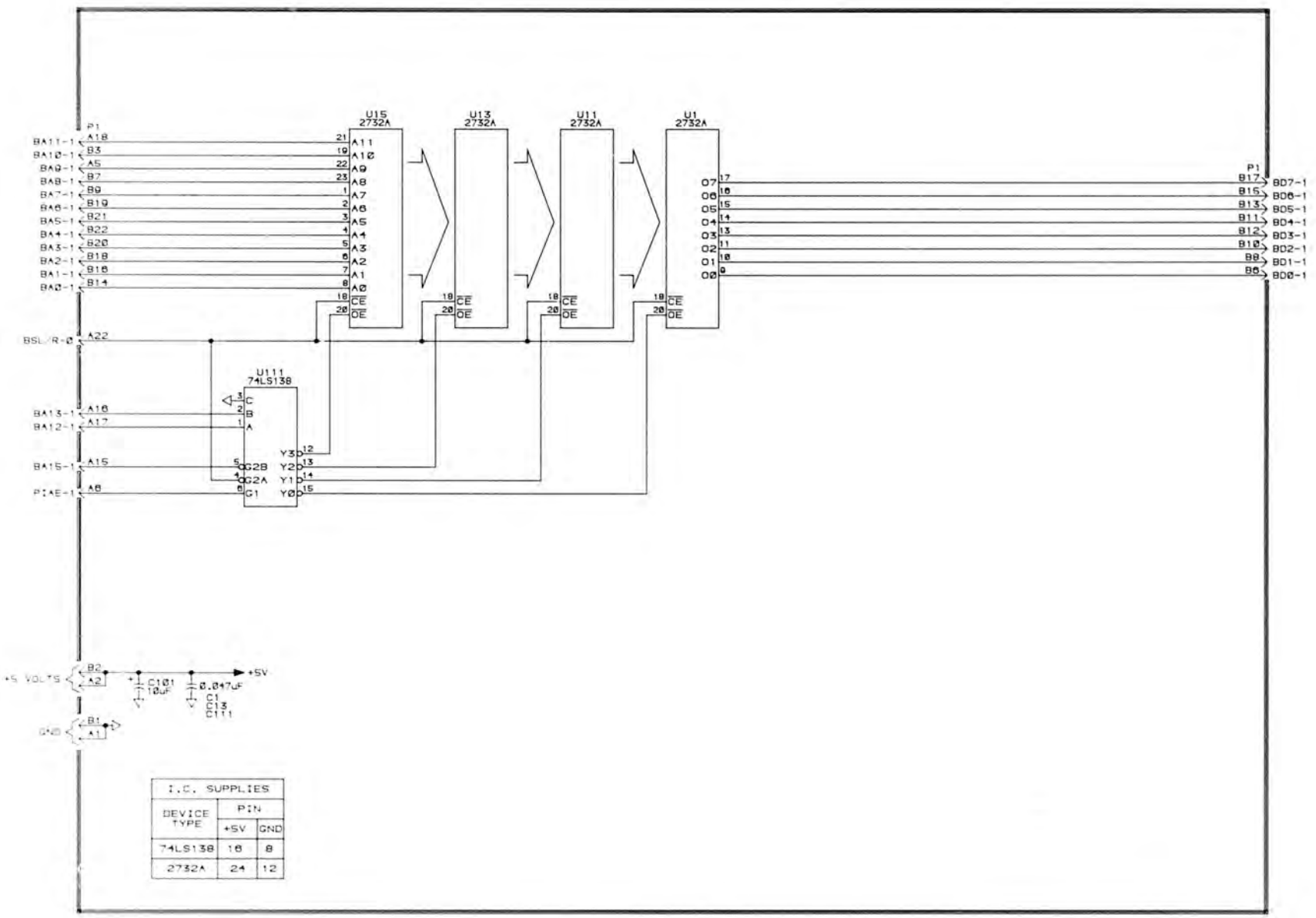
ASSEMBLY-SHEET:

A2-1

(1 OF 1)

4052R14
670-7632-00

A2-1

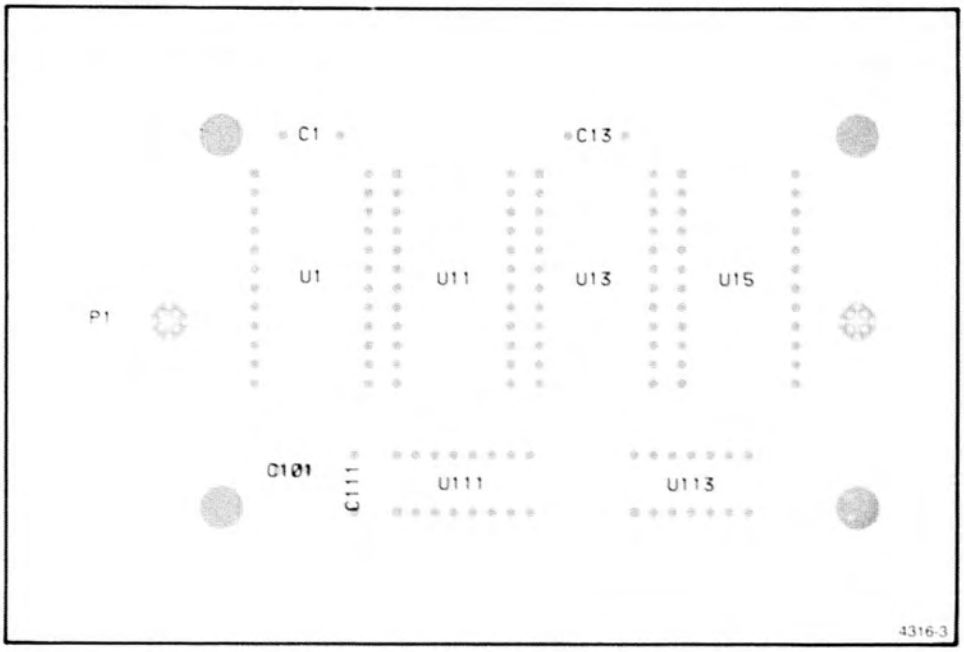


I.C. SUPPLIES		
DEVICE TYPE	PIN	
	+5V	GND
74LS138	16	8
2732A	24	12

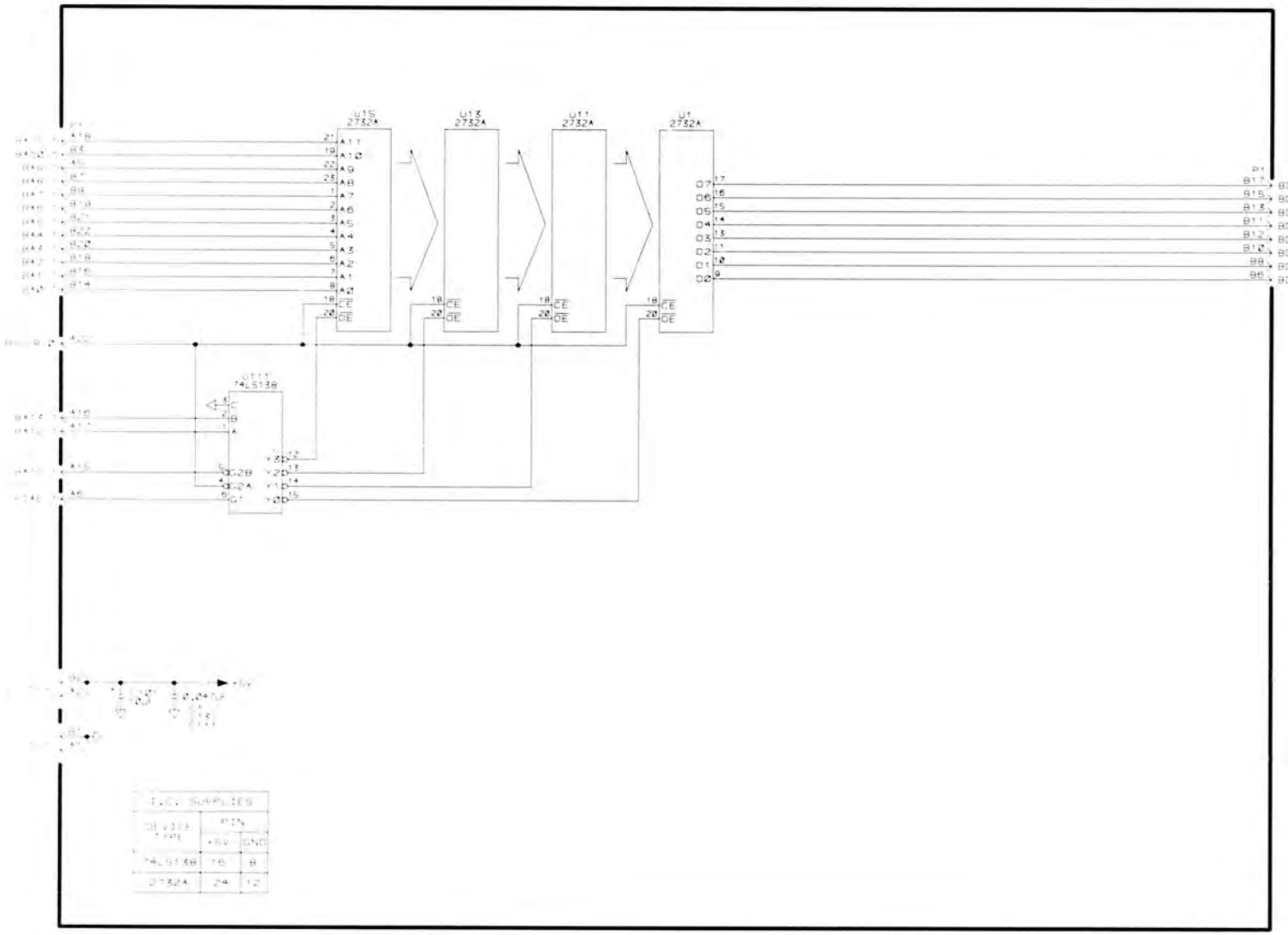
4052R14	NOTES: KEYSLOT LOCATED BETWEEN PINS A6 & A7. ADD: JAN 1983 4316-S4	BOARD: 670-7632-01 16K ROM PACK GPIB ENHANCEMENT ROM PACK	ASSEMBLY-SHEET: A2-1 (1 OF 1)

4052R14
670-7632-01

A2-1



4052R14 Option 1A (670-8122-00) Component Locations.



I.C. SUPPLIES		
DEVICE TYPE	PIN	
	+5V	GND
74LS138	16	8
2732A	24	12

NOTES: 1. EXPANDED INDICATES BETWEEN PINS A6 & A7.

ADD: JAN 1983
4516-SS

BOARD: 670-8122-22 16K ROM PACK

GPIB ENHANCEMENT
ROM PACK

ASSEMBLY SHEET
A3-1

Appendix A

SAMPLE PROGRAMS

INTRODUCTION

The following programs exercise the routines in the 4050 Series GPIB ROM Pack. The programs illustrate the use and syntax of the routines provided by the ROM pack. If you encounter errors while using the ROM pack, the programs may be used to verify that the ROM pack is functioning properly.

These programs do not use any of the parallel poll routines. Use the example programs given in Section 2 to verify that these routines function correctly.

To run a program, install the ROM pack as discussed in Section 1. Then enter the program as shown. Refer to the appropriate 4050 Series Graphic Computing System manual if you need more information.

Program 1: IEEE-488 Mnemonics Routines

The following program uses a bus monitor to show the bus traffic generated by each IEEE-488 standard mnemonic routine. Any bus monitor that allows either single step control or total traffic display will function with this program.

Examples: 7D01/DF2, 7D02, 8000 Series, DAS 9000 Series, and 067-1027-00 manual keyboard.

```
1 REM(GPIB STD MESSAGES WITH KEY BOARD)
2 REM USE KEYBOARD(067-1027-00) SET-M,SET-L,TTY OFF-TO MONITOR
3 GO TO 100
4 REM SENDS H14(DCL) WITH"ATN"AND"REN"(NOTE H=HEX)
5 CALL "DCL"
6 REM CLEAR"REN"GREEN LIGHT WITH RESET-R
7 END
8 REM SENDS H30(L16),H36(L22),H3B(L24),H08(GET),H5F(UNT),H3F(UNL)
9 CALL "GET":16:22:24
10 REM THIS MESSAGE IS SENT WITH "ATN" AND "REN" ASSERTED
11 END
12 REM SENDS H34(L20),H30(L16),H2B(L8),H01(GTL),H5F(UNT),H3F(UNL)
13 CALL "GTL":20:16:8
14 REM THIS MESSAGE IS SENT WITH "ATN" AND "REN" ASSERTED
15 END
16 REM PULSES THE "IFC" AND "REN" LINES
17 CALL "IFC"
18 REM CLEAR "IFC" GREEN LIGHT WITH RESET-I
19 END
20 REM SENDS H36(L22),H3B(L24),H3A(L26)
21 CALL "LISTEN":22:24:26
22 REM THIS MESSAGE IS SENT WITH "ATN" AND "REN" ASSERTED
23 END
24 REM SENDS H11(LLO) WITH "ATN" AND "REN"
25 CALL "LLO"
27 END
28 REM PULSES THE "REN" LINE ONLY
29 CALL "LOCS"
31 END
32 REM SENDS H11(LLO),H21(L1),H23(L3),H25(L5),H5F(UNT),H3F(UNL)
33 CALL "RWLS":1:3:5
34 REM THIS MESSAGE IS SENT WITH "ATN" AND "REN"
35 END
36 REM SENDS H23(L3),H25(L5),H27(L7),H04(SDC),H5F(UNT),H3F(UNL)
37 CALL "SDC":3:5:7
38 REM BOTH "ATN" AND "REN" ARE ASSERTED WITH THIS MESSAGE
39 END
40 REM SENDS H56(T22) WITH "ATN" AND "REN"
41 CALL "TALK":22
42 REM ONLY ONE MESSAGE IS SENT BECAUSE ONLY ONE TALKER IS ALLOWED
43 END
44 REM SENDS H3F(UNL) WITH "ATN" AND "REN"
45 CALL "UNL"
47 END
48 REM SENDS H5F(UNT) WITH "ATN" AND "REN"
49 CALL "UNT"
51 END
```

Program 2: Bit Pattern Manipulations

This program exercises the bit manipulation routines of the ROM Pack: (1) VARSET, (2) VARCLR, (3) VARTST, (4) HEXDEC, and (5) DECHEX.

The initial section, lines 1 to 300, sets up the operating conditions for the rest of the program and must be run first.

```
1 REM(BIT PATTERN MANIPULATIONS)
3 GO TO 100
4 REM * DECHEX * CHANGE DECIMAL NUMBERS INTO HEX NUMBERS
7 GO TO 1000
8 REM * HEXDEC * CHANGE HEX NUMBERS INTO DECIMAL NUMBERS
11 GO TO 2000
12 REM * VARSET * BIT SET DEMONSTRATION
15 GO TO 3000
16 REM * VARCLR * BIT CLEARING DEMONSTRATION
19 GO TO 4000
20 REM * VARTST * BIT PATTERN TEST
23 GO TO 5000
24 REM DECIMAL TO BIT PATTERN
26 GO TO 6000
100 PAGE
110 INIT
120 DIM A$(4),Z(16),U(16)
130 DATA 32768,16384,8192,4096,2048,1024,512,256
140 DATA 128,64,32,16,8,4,2,1
150 READ Z
160 N=0
170 PRINT "UNIT INITIALIZED FOR BIT PATTERN MANIPULATION"
180 PRINT
190 PRINT " KEY 1---CONVERTS DECIMAL TO HEX"
200 PRINT
210 PRINT " KEY 2---CONVERTS HEX TO DECIMAL"
220 PRINT
230 PRINT " KEY 3---SETS BITS"
240 PRINT
250 PRINT " KEY 4---CLEARS BITS"
260 PRINT
270 PRINT " KEY 5---TESTS ONE BIT PATTERN AGAINST ANOTHER"
280 PRINT
290 PRINT " KEY 6---CONVERTS DECIMAL NUMBERS TO BIT PATTERNS"
300 END
```

Lines 1000 to 1060 convert decimal numbers input from the keyboard and print out the HEX values on the screen. This routine may be used to set an inital value for the VARSET, VARCLR, and VARTST routines.

```
1000 PRINT "INPUT DECIMAL VALUE ";
1010 INPUT N
1020 IF N>65535 THEN 1000
1030 CALL "DECHEX",N,A$
1040 IF F1=1 THEN 4140
1050 PRINT A$
1060 END
```

Lines 2000 to 2040 convert HEX to decimal numbers.

```
2000 PRINT "INPUT HEX VALUE ";
2010 INPUT A$
2020 CALL "HEXDEC",A$,S3
2030 PRINT S3
2040 END
```

Lines 3000 to 3130 display bit patterns, set bits specified by the operator, and display the results. The bit pattern display subroutine in lines 4140 to 4220 is called by this routine.

```
3000 PRINT "INITIAL VALUE ";
3010 J=N
3020 GOSUB 4140
3030 PRINT "WHICH BIT DO YOU WISH SET? ";
3040 INPUT A
3050 PRINT
3060 CALL "VARSET",N,2^(A-1)
3070 J=2^(A-1)
3080 PRINT "INCREASE      ";
3090 GOSUB 4140
3100 J=N
3110 PRINT "FINAL VALUE   ";
3120 GOSUB 4140
3130 END
4140 FOR I=15 TO 0 STEP -1
4150 CALL "VARTST",J,2^I,Z1
4160 IF NOT(Z1) THEN 4180
4170 Z1=1
4180 PRINT Z1;
4190 NEXT I
4200 PRINT " =";J
4210 PRINT
4220 RETURN
```

Lines 4000 to 4130 clear specified bits from the pattern.

```
4000 PRINT "INITIAL VALUE ";
4010 J=N
4020 GOSUB 4140
4030 PRINT "WHICH BIT DO YOU WISH TO CLEAR? ";
4040 INPUT A
4050 PRINT
4060 PRINT "DECREASE      ";
4070 J=2^(A-1)
4080 GOSUB 4140
4090 CALL "VARCLR",N,2^(A-1)
4100 J=N
4110 PRINT "FINAL VALUE   ";
4120 GOSUB 4140
4130 END
```

Lines 5000 to 5090 test specified bits for a match.

```
5000 PRINT "WHICH BIT DO YOU WISH TO TEST? ";
5010 INPUT A
5020 J=2^(A-1)
5030 GOSUB 4140
5040 J=N
5050 GOSUB 4140
5060 CALL "VARTST",N,2^(A-1),W
5070 J=W
5080 GOSUB 4140
5090 END
```

Lines 6000 to 6160 convert decimal numbers input by the user to bit patterns that may be used as input to the VARSET, VARCLR, and VARTST routines. The range for input is 0 to 65535, decimal.

```
6000 REM DECIMAL TO BIT PATTERN
6010 PRINT "INPUT DECIMAL VALUE ";
6020 INPUT N
6030 PRINT
6040 IF N>65535 THEN 6010
6050 M=N
6060 FOR I=1 TO 16
6070 IF M<Z(I)-1 THEN 6110
6080 U(I)=0
6090 NEXT I
6100 GO TO 6140
6110 U(I)=1
6120 M=M-Z(I)
6130 GO TO 6090
6140 PRINT USING 6160;U
6150 RETURN
6160 IMAGE 16(D,X)
```

Program 3: Command Checkout

The following program checks out the following routines:

(1) ARSIZE, (2) UNDEF, (3) ERRHLP, (4) STBHLP, (5) LAST, (6) THEADER, (7) VLIST, (8) TNAME, (9) WAIT, (10) MTPACK, and (11) NEWTAP.

Lines 1 to 99 are the user key entry points to start each routine.

```
1 REM (MISC.GPIB COMMAND CHECKOUT)
3 END
4 REM COMMANDS * ARSIZE * UNDEF  *
7 GO TO 100
8 REM COMMAND  * ERRHLP *
11 GO TO 1000
12 REM COMMAND * STBHLP *
15 GO TO 1110
16 REM COMMANDS*  LAST  * THEADER * VLIST *
19 GO TO 1200
20 REM COMMANDS*  LAST  * THEADER * TNAME *
23 GO TO 1300
28 REM COMMANDS* NEWTAPE * THEADER * WAIT *
31 GO TO 3000
32 REM COMMAND * MTPACK * WAIT * NEWTAPE *
35 GO TO 4000
99 END
```

User key 1 initiates the array size test in lines 100 to 350. The "ARSIZE" and "UNDEF" functions are checked out in these routines.

```
100 REM   ARRAY SIZE CHECK OUT
110 INIT
120 PAGE
130 DIM A(12)
140 PRINT "ARSIZE TEST"
150 CALL "ARSIZE",A,B,C
160 PRINT "THE VALUES SHOULD BE 12 AND 0. THEY ARE ";
170 PRINT B;"  ";C
180 DELETE A
190 DIM A(3,23)
200 CALL "ARSIZE",A,B,C
210 PRINT "THE VALUES SHOULD BE 3 AND 23. THEY ARE ";
220 PRINT B;"  ";C
230 CALL "UNDEF",A,B
240 PRINT "UNDEF TEST"
250 PRINT "THE VALUE SHOULD BE 1. IT IS ";
260 PRINT B
270 A=1
280 CALL "UNDEF",A,B
290 PRINT "THE VALUE SHOULD BE 0. IT IS ";
300 PRINT B
310 DELETE A
320 CALL "UNDEF",A,B
330 PRINT "THE VALUE SHOULD BE  1. IT IS ";
340 PRINT B
350 END
```

User key 2 prints out the "ERRHLP" messages; user key 3 prints out the "STBHLP" messages. The supporting program is in lines 1000 to 1180.

```
1000 REM  ERRHLP CHECK OUT
1010 INIT
1020 FOR I=100 TO 800 STEP 100
1030 FOR J=1 TO 10
1040 CALL "ERRHLP",I+J,A$
1050 IF LEN(A$)=1 THEN 1070
1060 PRINT I+J;"  ";A$
1070 NEXT J
1080 NEXT I
1090 END
1110 REM  STBHLP CHECKOUT
1120 INIT
1130 FOR I=0 TO 255
1140 CALL "STBHLP",I,A$
1150 IF LEN(A$)=1 THEN 1170
1160 PRINT I;"  ";A$
1170 NEXT I
1180 END
```

Lines 1200 to 1480 include the programs for a number of file-oriented routines. User key 4 demonstrates the "LAST", "THEADER", and "VLIST" commands. User key 5 demonstrates the "LAST", "THEADER", and "TNAME" commands.

```
1200 REM  FILE ORIENTED COMMAND CHECKOUT
1210 INIT
1220 CALL "LAST",Z
1230 CALL "THEADER",A$
1240 FIND Z-1
1250 CALL "THEADER",X
1260 FIND X+1
1270 CALL "THEADER",B$
1280 CALL "VLIST"
1290 END
1300 REM  TNAME CHECKOUT
1310 INIT
1320 DIM A$(11),C$(1)
1330 CALL "LAST",C
1340 FIND C-1
1350 SAVE
1360 CALL "THEADER"
1380 PRINT "WHAT NAME DO YOU WANT ON THIS FILE?"
1390 INPUT A$
1400 IF A$="" THEN 1380
1410 FIND C-1
1420 CALL "TNAME",A$
1430 CALL "THEADER"
1450 PRINT "IS THIS CORRECT? "
1460 INPUT C$
1470 IF C$="N" THEN 1380
1480 END
```

Lines 3000 to 3330 contain the program which demonstrates the "NEWTAP" routine.

```
3000 REM
3010 PAGE
3020 INIT
3030 PRINT "ENTER A TAPE FILE NUMBER ABOVE 5 AND BEFORE THE LAST"
3040 INPUT A1
3050 CALL "NEWTAPE",A2
3060 IF A2=0 THEN 3290
3070 FIND A1
3080 PRINT "THIS IS THE HEADER FOR THAT FILE. "
3090 PRINT
3100 CALL "THEADER"
3110 PRINT "REMOVE THE TAPE FROM THE 4050 AND THEN REINSERT IT."
3120 CALL "NEWTAPE",C
3130 GOSUB C+1 OF 3150,3190,3230
3140 STOP
3150 REM NO TAPE CONDITION
3160 PRINT "REINSERT THE TAPE"
3170 CALL "WAIT",2
3180 GO TO 3120
3190 REM OLD TAPE CONDITION
3200 PRINT " PLEASE REMOVE THE TAPE AND THEN REINSERT IT."
3210 CALL "WAIT",4
3220 GO TO 3120
3230 REM NEW TAPE CONDITION
3240 PRINT "THANK YOU. WE WILL NOW PROCEED WITH THE TEST."
3250 PRINT
3260 PRINT "THIS IS THE FILE WHERE THE TAPE IS NOW POSITIONED."
3270 CALL "THEADER"
3280 END
3290 PRINT "THIS TEST REQUIRES THAT A MULTI-FILE TAPE BE INSTALLED ";
3300 PRINT "IN THE 4050 BEFORE WE CAN PROCEED."
3310 CALL "WAIT",5
3320 GO TO 3020
3330 END
```

Lines 4000 to 5000 contain the demonstration program for the "MTPACK" routine.

```
4000 REM
4010 INIT
4015 PAGE
4020 PRINT "THIS ROUTINE WILL CYCLE THE TAPE FROM END TO END FOR THE";
4030 PRINT " PURPOSE OF RE-ALIGNING THE TAPE.  REMOVE THE TAPE FROM ";
4040 PRINT "THE UNIT AND INSPECT IT FOR RIDGES WHERE EARLIER FILES";
4050 PRINT " WERE READ. AFTER YOU HAVE DONE THIS REINSERT THE TAPE."
4060 CALL "NEWTAPE",C
4070 GO TO C+1 OF 4100,4200,4300
4080 STOP
4100 PRINT "PLEASE REINSERT THE TAPE"
4110 CALL "WAIT",5
4120 GO TO 4060
4200 PRINT "PLEASE REMOVE THE TAPE"
4210 CALL "WAIT",5
4220 GO TO 4060
4300 CALL "NEWTAPE"
4310 CALL "MTPACK"
4320 PRINT "NOW INSPECT THE TAPE FOR RIDGES AND REPEAT THE TEST IF";
4330 PRINT " YOU THINK NECESSARY OR IF IT IS A NEW TAPE."
5000 END
```

Program 4: SRQ Commands

The following program demonstrates the operation of the SRQ-related ROM pack routines.

```
1 REM (SRQ-POLL-CONFIG TEST)
3 GO TO 100
4 REM COMMAND * CONFIG *
5 INIT
6 F1=1
7 GO TO 1000
8 REM COMMAND * CONFIG *
9 INIT
10 F1=2
11 GO TO 2000
12 REM COMMAND * STD POLL *
13 INIT
14 F1=3
15 GO TO 3000
16 REM COMMAND * CALL"POLL" *
17 INIT
18 F1=4
19 GO TO 4000
20 REM      AUTO POLLING
21 INIT
22 F1=5
23 GO TO 5000
24 REM * TURN ON SRQ SENSITIVITY
25 CALL "SRQON"
27 END
28 REM * BLOCK SRQ INTERUPTS
29 CALL "SRQOFF"
31 END
40 REM INITIAL INSTRUCTIONS
41 INIT
42 PAGE
43 GO TO 220
99 END
100 REM
110 PAGE
120 LIST 130,210
130 REM:::
140 REM:::THIS PROGRAM WAS WRITTEN TO DEMONSTRATE
150 REM:::SOME OF THE FEATURES OF THE 4050R14 GPIB ENHANCEMENT
160 REM:::ROM PACK. THERE ARE FIVE PROGRAMS IN THIS SERIES
170 REM:::   (1)IEEE-488 MNEMONICS COMMANDS
180 REM:::   (2)BIT MANIPULATION COMMANDS
190 REM:::   (3)FILE ORIENTED COMMANDS
200 REM:::   (4)SRQ RELATED COMMANDS  ****
210 REM:::   (5)PARALLEL POLL COMMANDS
220 PRINT
230 PRINT "THIS PROGRAM DEMONSTRATES THE OPERATION OF THE ""SRQ""";
240 PRINT "RELATED COMMANDS."
250 PRINT
260 PRINT "FULL OPERATION OF THIS PROGRAM REQUIRES THAT A NUMBER ";
270 PRINT "OF INSTRUMENTS BE CONNECTED TO THE SYSTEM. EACH ";
280 PRINT "SHOULD BE SET TO A DIFFERENT ADDRESS."
290 PRINT
300 PRINT " KEY-1 CONFIG COMMAND FOR PRIMARY ADDRESSES ONLY "
310 PRINT
320 PRINT " KEY-2 CONFIG COMMAND FOR PRIMARY AND SECONDARY ADDRESSES"
330 PRINT
340 PRINT " KEY-3 STANDARD POLL COMMAND OPERATION"
350 PRINT
360 PRINT " KEY-4 CALL""POLL"" OPERATION"
370 PRINT
380 PRINT " KEY-5 AUTO OR BLIND POLLING "
390 PRINT
400 PRINT " KEY-6 SRQON "
410 PRINT
420 PRINT " KEY-7 SRQOFF "
430 PRINT
440 PRINT " KEY-10 REPEAT THESE INSRUCTIONS "
450 PRINT
460 PRINT "PRESS ONE OF THE KEYS TO START THE PROGRAM"
470 END
```

```
1000 REM  CONFIGURE COMMAND FOR PRIMARY ADDRESSES ONLY
1010 PAGE
1020 PRINT "THE CONFIGURE AND THE ARRAY SIZE COMMANDS ARE USED IN ";
1030 PRINT "THIS PROGRAM."
1040 PRINT "THIS ROUTINE AUTOMATICALLY FINDS OUT WHO IS ON THE BUS."
1050 OPTION A
1060 DIM A(15)
1070 CALL "CONFIG",5,E;A
1080 IF E=0 THEN 1160
1090 CALL "ARSIZE",A,Z3,Z4
1100 PRINT "THERE ARE ";Z3;" DEVICES ON THE BUS."
1110 PRINT "THE ADDRESS LIST IS ";
1120 FOR I=1 TO Z3
1130 PRINT A(I);"  ";
1140 NEXT I
1150 END
1160 DELETE G$
1170 DIM G$(1)
1180 IF E=1 THEN 1250
1190 PRINT "NO DEVICES RESPONDED TO THE POLL. CHECK THE INSTRUMENTS."
1200 PRINT
1210 PRINT "DO YOU WISH TO REPEAT THE CONFIGURATION CHECK ? ";
1220 INPUT G$
1230 IF G$="Y" THEN 1000
1240 END
1250 PRINT
1260 PRINT "THE ARRAY SPACE DESIGNATED WAS NOT LARGE ENOUGH  FOR ";
1270 PRINT "THE NUMBER OF RESPONSES RECEIVED."
1280 PRINT
1290 PRINT " DO YOU WISH TO REPEAT THE TEST ? ";
1300 INPUT G$
1310 IF G$="N" THEN 1350
1320 IF G$="Y" THEN 1000
1330 PRINT "PLEASE REPLY YES OR NO ."
1340 GO TO 1290
1350 END
```

```
2000 REM CONFIGURE COMMAND FOR SECONDARY AND PRIMARY ADDRESSES
2010 PAGE
2020 PRINT "THE CONFIGURE AND THE ARRAY SIZE COMMANDS ARE USED IN ";
2030 PRINT "THIS PROGRAM."
2040 PRINT "THIS ROUTINE AUTOMATICALLY FINDS OUT WHO IS ON THE BUS."
2050 DELETE B
2060 DIM B(15,2)
2070 CALL "CONFIG",5,E;B
2080 IF E>0 THEN 2220
2090 CALL "ARSIZE",B,Z3,Z4
2100 PRINT "THERE ARE ";Z3;" DEVICES ON THE BUS."
2110 S1=0
2120 FOR I=1 TO Z3
2130 IF B(I,2)<0 THEN 2150
2140 S1=S1+1
2150 NEXT I
2160 PRINT S1;" OF THESE DEVICES HAD SECONDARY ADDRESSES."
2170 FOR I=1 TO Z3
2180 PRINT "THE ADDRESSES ARE: ";B(I,1);" PRIMARY ";B(I,2);" SECONDARY"
2190 PRINT
2200 NEXT I
2210 END
2220 DELETE G$
2230 DIM G$(1)
2240 IF E=1 THEN 2310
2250 PRINT "NO DEVICES RESPONDED TO THE POLL. CHECK THE INSTRUMENTS."
2260 PRINT
2270 PRINT "DO YOU WISH TO REPEAT THE CONFIGURATION CHECK ? ";
2280 INPUT G$
2290 IF G$="Y" THEN 2000
2300 END
2310 PRINT
2320 PRINT "THE ARRAY SPACE DESIGNATED WAS NOT LARGE ENOUGH  FOR ";
2330 PRINT "THE NUMBER OF RESPONSES RECEIVED."
2340 PRINT
2350 PRINT " DO YOU WISH TO REPEAT THE TEST ? ";
2360 INPUT G$
2370 IF G$="N" THEN 2410
2380 IF G$="Y" THEN 2000
2390 PRINT "PLEASE REPLY YES OR NO ."
2400 GO TO 2350
2410 END
```

```
3000 REM  STANDARD POLL ROUTINE
3010 REM INITIALIZATION DECISIONS ENTRY
3020 DELETE V
3030 PRINT "HOW MANY DEVICES DO YOU HAVE ON THE BUS ?"
3040 INPUT X1
3050 DIM V(X1)
3060 FOR I=1 TO X1
3070 PRINT "WHAT IS THE ADDRESS OF NUMBER ";I;"?  ";
3080 INPUT V(I)
3090 IF V(I)>1 AND V(I)<31 THEN 3120
3100 PRINT "THE VALUE ENTERED IS OUT OF RANGE. ENTER THE PROPER VALUE.";
3110 GO TO 3080
3120 NEXT I
3130 PRINT "THE ADDRESS LIST IS ";
3140 FOR I=1 TO X1
3150 PRINT V(I);"  ";
3160 NEXT I
3170 DELETE G$
3180 DIM G$(1)
3190 PRINT
3200 PRINT "IT IS ESSENTIAL THAT THIS LIST BE CORRECT BECAUSE IF ";
3210 PRINT "ANY OTHER DEVICE IS PRESENT THEN THE SYSTEM WILL ";
3220 PRINT "HANG UP OR ERROR OUT WITH A SYSTEM ERROR   MESSAGE.";
3230 PRINT "IS THE ADDRESS LIST COMPLETE AND CORRECT ?";
3240 INPUT G$
3250 IF G$="N" THEN 3020
3260 IF G$<>"Y" THEN 3230
3270 REM
3280 REM SET UP TEST OF YOUR ADDRESSES
3290 REM
3300 PRINT "IN ORDER TO TEST THE POLL ROUTINE YOU WILL NEED TO ";
3310 PRINT "GENERATE SRQ'S.IF THE DEVICES CONNECTED ARE ALL TM5000 ";
3320 PRINT "FAMILY,YOU MAY GENERATE SRQ'S BY USE OF THE USER ";
3330 PRINT "REQUEST FUNCTION. DO YOU WISH ";
3340 PRINT "TO DO SO ?";
3350 INPUT G$
3360 IF G$<>"Y" THEN 3400
3370 FOR I=1 TO X1
3380 PRINT @V(I):"RQS ON;USER ON;"
3390 NEXT I
3400 REM ACTIVATE THE POLL ROUTINE
3410 ON SRQ THEN 3470
3420 REM TURN ON SRQ SENSING
3430 CALL "SRQON"
3440 REM
3450 REM MAIN BODY OF YOUR PROGRAM GOES HERE
3460 REM END OF PROGRAM
3470 REM STANDARD SERIAL POLL TO AN ADDRESS LIST
3480 DELETE Y
3490 DIM Y(X1)
3500 FOR I=1 TO X1
3510 POLL X2,X3;V(I)
3520 Y(I)=X3
3530 NEXT I
3540 REM STANDARD PROCEDURE IS TO PRINT OUT THE FOLLOWING MESSAGE
3550 PRINT "STATUS BYTE ";X3;" AT BUS ADDRESS ";X2
3560 REM EXPANDED PRINT OUT METHOD
3570 FOR I=1 TO X1
3580 CALL "STBHLP",Y(I),Y$
3590 PRINT "ADDRESS ";V(I);" REPORTS ";Y(I);" ";Y$

3600 NEXT I
3610 END
```

```
4000 REM POLL ROUTINE USING THE CALL"POLL" COMMAND
4010 REM INITIALIZATION DECISIONS ENTRY
4015 PAGE
4020 DELETE V
4030 PRINT "HOW MANY DEVICES DO YOU HAVE ON THE BUS ?"
4040 INPUT X1
4050 DIM V(X1)
4060 FOR I=1 TO X1
4070 PRINT "WHAT IS THE ADDRESS OF NUMBER ";I;"?  ";
4080 INPUT V(I)
4090 IF V(I)>1 AND V(I)<31 THEN 4120
4100 PRINT "THE VALUE ENTERED IS OUT OF RANGE. ENTER THE PROPER VALUE.";
4110 GO TO 4080
4120 NEXT I
4130 PRINT "THE ADDRESS LIST IS ";
4140 FOR I=1 TO X1
4150 PRINT V(I);"  ";
4160 NEXT I
4170 DELETE G$
4180 DIM G$(1)
4190 PRINT
4200 PRINT " IT IS DESIRABLE THAT THE LIST BE CORRECT, BUT ERRORS ARE ";
4210 PRINT "NOT FATAL TO  THE PROGRAM. THE SYSTEM WILL TIME OUT AT ";
4220 PRINT "NON RESPONDING ADDRESSES. IS THE LIST COMPLETE AND ";
4230 PRINT "CORRECT ?";
4240 INPUT G$
4250 IF G$="N" THEN 4020
4260 IF G$<>"Y" THEN 4230
4270 REM
4280 REM SET UP TEST OF YOUR ADDRESSES
4290 REM
4300 PRINT
4310 PRINT "IN ORDER TO TEST THE POLL ROUTINE YOU WILL NEED TO ";
4320 PRINT "GENERATE SRQ'S.IF THE DEVICES CONNECTED ARE ALL TM5000 ";
4330 PRINT " FAMILY INSTRUMENTS YOU MAY GENERATE  SRQ'S BY ";
4340 PRINT "ACTIVATING THE USER REQUEST FUNCTION. DO YOU WISH ";
4350 PRINT "TO DO SO ?";
4360 INPUT G$
4370 IF G$<>"Y" THEN 4410
4380 FOR I=1 TO X1
4390 PRINT #V(I):"RQS ON;USER ON;"
4400 NEXT I
4410 REM ACTIVATE THE POLL ROUTINE
4420 ON SRQ THEN 4460
4430 REM TURN ON SRQ SENSING
4440 CALL "SRQON"
4450 REM
4460 REM MAIN BODY OF YOUR PROGRAM GOES HERE
4470 REM END OF PROGRAM
4480 REM THE CALL"POLL"ROUTINE TO AN ADDRESS LIST
4490 DELETE Y
4500 X9=0
4510 DIM Y(X1)
4520 FOR I=1 TO X1
4525 X3=0
4530 CALL "POLL",5,X2,X3;V(I)
4535 PRINT X2,X3,V(I),I
4540 IF X2=0 THEN 4560
4550 X9=X9+1
4560 Y(I)=X3
4570 NEXT I
4580 IF X9=X1 THEN 4600
4590 PRINT "ONLY ";X9;" DEVICES RESPONDED TO THE POLL."
4600 REM EXPANDED STATUS BYTE PRINT OUT METHOD
4610 FOR I=1 TO X1
4620 CALL "STBHLP",Y(I),Y$
4630 PRINT "ADDRESS ";V(I);" REPORTS ";Y(I);" ";Y$
4640 NEXT I
4650 END
```

```
5000 REM AUTO OR BLIND POLLING
5010 PRINT "WHAT IS THE MOST INSTRUMENTS YOU WILL HAVE ON THE BUS ?";
5020 INPUT Z1
5030 IF Z1>30 THEN 5010
5040 DELETE C
5050 DIM C(Z1,2)
5060 PRINT "WHAT IS THE SLOWEST RESPONSE TIME ON THE BUS ?";
5070 INPUT Z2
5080 IF Z2>1 THEN 5100
5090 Z2=1
5100 IF Z2<1000 THEN 5120
5110 Z2=1000
5120 CALL "CONFIG",Z2,E;C
5130 IF E>0 THEN 5400
5140 CALL "ARSIZE",C,Z3,Z4
5150 PRINT "THERE ARE ";Z3;" DEVICES ON THE BUS. "
5160 PRINT "THE ADDRESSES ARE: ";
5170 FOR I=1 TO Z3
5180 PRINT C(I,1);"  ";
5190 NEXT I
5200 PRINT
5210 ON SRQ THEN 5270
5220 Z9=0
5230 REM NEED CALL"SRQON"
5240 CALL "SRQON"
5250 REM MAIN PROGRAM HERE
5260 REM  ACTUAL POLL ROUTINE AFTER AUTO-CONFIGURE
5270 REM Z2=TIMEOUT Z5=DEVICE INDICATOR Z6=STATUS BYTE
5280 CALL "POLL",Z2,Z5,Z6;C
5290 IF Z5>0 THEN 5340
5300 IF Z9=1 THEN 5320
5310 PRINT "NO SRQ WAS FOUND AT ANY INSTRUMENTS ON THE BUS."
5320 CALL "SRQOFF"
5330 END
5340 PRINT
5350 PRINT " AN SRQ WAS FOUND AT ";C(Z5,1)
5360 Z9=1
5370 CALL "STBHLP",Z6,J$
5380 PRINT "IT HAS REPORTED A STATUS BYTE OF: ";Z6;" WHICH IS ";J$
5390 RETURN
5400 DELETE G$
5410 DIM G$(1)
5420 IF E=1 THEN 5490
5430 PRINT "NO DEVICES RESPONDED ON THE BUS.CHECK THE INSTRUMENTS."
5440 PRINT
5450 PRINT "DO YOU WISH TO REPEAT THE AUTO-CONFIGURE TEST ?";
5460 INPUT G$
5470 IF G$="Y" THEN 5000
5480 END
5490 PRINT
5500 PRINT "THE ARRAY SPECIFIED WAS NOT LARGE ENOUGHT FOR THE ";
5510 PRINT "NUMBER OF RESPONSES   RECIEVED."
5520 PRINT
5530 PRINT "DO YOU WISH TO REPEAT THE TEST ?"
5540 INPUT G$
5550 IF G$="N" THEN 5590
5560 IF G$="Y" THEN 5000
5570 PRINT "PLEASE REPLY YES OR NO"
5580 GO TO 5520
5590 END
```

Program 5: Error Handling

4052R14 Option 1A Only. This program has examples of ERRLIST, ASKERR, CLRREP, and RETRY

```
100 CALL "ERRLIST",69  ! 'NO DEVICES ON GPIB' error code
110 ON SIZE THEN 2000
120 ON TIMEOUT THEN 1000
130 _timout=0.01
140 CALL "TIMSET",_timout  ! Set time-out to 10mS
150   !
160   !
170   !
500 PRINT @5:"MEASURE VOLTS"
510 INPUT @5:Voltage  ! Timeoout may occur here.
520 CALL "CLRREP"  ! Reset repetition counter to zero
530   !
540   !
550   !
1000 REM Timeout error handler
1010 CALL "ASKERR",X,Rpcount  ! Get the repetition count
1020 IF Rpcount 10 THEN 1060
1030 _timout=_timout+0.01  ! Increase time-out value by 10mS
1040 CALL "TIMSET",_timout
1050 CALL "RETRY"  ! Try to read measurement again
1060 PRINT "Instrument time-out on device 5"
1070 STOP
1080   !
1090   !
1100   !
2000 REM 'NO DEVICES ON GPIB HANDLER' error handler.
2010 PRINT "Instrument 5 has been powered down."
```

Program 6: TAPEIN Example

4052R14 Option 1A Only.

```
100 REM  Copy file 1 to file 2
110 DIM File$(MEMORY-500)
120 FIND 1
130 CALL 'TAPEIN',File$  ! Read ASCII file into the string File$.
140 PRINT 'Size of file =';LEN(File$)
150 FIND 2
160 PRINT @33:File$;  ! Write ASCII file data to file 2.
170 CLOSE
```

Program 7: NEWTAPE Example

4052R14 Option 1A Only.

```
100 CALL 'NEWTAPE',Tap_status,Wr_protect
110 IF Tap_status=0 THEN 100
120 IF Wr_protect THEN
130    PRINT 'Mag-tape is write protected, please change SAFE switch'
140    CALL 'NEWTAPE',Tap_status,Wr_protect
150    GO TO Tap_status+1 OF 140,140,100
160 END IF
```

Program 8: TAPEAPP and TRIM Example

4052R14 Option 1A Only.

```
100 REM  Copy files 7 thru 12 to files 1 thru 6
110 DIM File$(MEMORY-500),Fsize(6)
120 File$=""
130 Lastlen=0  ! Init length of all files to zero
140 FOR I=1 TO 6  ! Loop to read in all the files to copy
150     FIND I+6
160     CALL "TAPEAPP",File$  ! Read in a file and append to string
170     Fsize(I)=LEN(File$)-Lastlen  ! Calculate size of file
180     Lastlen=LEN(File$)  ! Remember length of all files
190 NEXT I
200 REM  Now begin to output the files from File$
210 FOR I=1 TO 6  ! Loop to write out files
220     FIND I
230     MARK 1,Fsize(I)  ! Create a file of appropriate size
240     Lastlen=LEN(File$)  ! Save the length of all files
250     CALL "TRIM",File$,Fsize(I)  ! Trim off all but the first file
260     FIND I
270     PRINT @33:File$;  ! Send file data out to mag-tape
280     CLOSE  ! Close the file
290     CALL "TRIM",File$,Lastlen  ! Restore string length
300     File$=REP("",1,Fsize(I))  ! Delete first file from string
310 NEXT I
```

Appendix B

ROUTINE SUMMARY

Routine	Example	Purpose
ARSIZE	CALL "ARSIZE", A, B, C	ARSIZE returns the currently dimensioned size of an array.
ASKERR	CALL "ASKERR",IX,RC	ASKERR supplies information on which error caused a trap and the number of times the error has occured on a consecutive basis.
BININ	CALL "BININ", P$, A$, E; 3	BININ inputs binary block data from a device on the bus.
BINOUT	CALL "BINOUT", P$, A$, E; 3	BINOUT sends data in binary block format from the Graphic System to devices on the bus.
CLRREP	CALL "CLRREP"	CLRREP sets the error repetition counter to 0.
CONFIG	CALL "CONFIG", E; A	CONFIG determines the active devices on the bus and returns the addresses in the array specified as an argument.
DCL	CALL "DCL"	DCL clears all devices on the bus.
DECHEX	CALL "DECHEX", 278, H$	DECHEX converts the decimal value of an expression to its equivalent in ASCII HEX.
ERRHLP	CALL "ERRHLP", 201, E$	ERRHLP returns information about the Tektronix Codes and Formats error codes.
ERRLIST	CALL "ERRLIST",3,5:9	ERRLIST enables specified errors to be trapped and handled by a user's ON SIZE and ON TIMEOUT error handling routines.

4052R14 Option 1A ROM Pack only

Routine	Example	Purpose
GET	CALL "GET"; 2; 5, 8; 13, 4	GET sends Group Execute Trigger to devices at addresses listed.
GTL	CALL "GTL"; 3, 5; 7	GTL returns devices specified in the address list to either Local State (LOCS) or Local With Lockout State (LWLS).
HEXDEC	CALL "HEXDEC", "F078", X	HEXDEC converts an ASCII HEX number representation to decimal.
IFC	CALL "IFC"	IFC pulses the GPIB IFC line to clear the GPIB interface.
LAST	CALL "LAST", X	LAST finds and returns the file number of the last file on the tape.
LISTEN	CALL "LISTEN"; 5; 27	LISTEN makes the devices specified in the address list listeners.
LLO	CALL "LLO"	LLO puts all instruments on the bus into either Local With Lockout State (LWLS) or Remote With Lockout State (RWLS).
LOCS	CALL "LOCS"	LOCS puts all devices in the LOCAL state.
MTPACK	CALL "MTPACK"	MTPACK winds and rewinds the tape to correct poor tape spooling that may occur.
NEWTAP	CALL "NEWTAP", X	NEWTAP finds the present file location and returns the tape cartridge status in a numeric code.
POLL	CALL "POLL" 5, I, S; A	POLL is used to serial poll the devices specified in the address list.
PPD	CALL "PPD"; 3; 5; 7	PPD unconfigures selected devices.

4051 ROM Pack only. This routine is internally implemented in the 4052 and 4054 Graphic Systems.

Routine	Example	Purpose
PPE	CALL "PPE", 1, 4; A	PPE configures the devices that are to respond to a parallel poll.
PPOLL	CALL "PPOLL", A	PPOLL performs a parallel poll of the devices previously configured.
PPU	CALL "PPU"	PPU unconfigures all devices.
PRISTR	CALL "PRISTR", A$; B	PRISTR outputs a string to the GPIB interface without EOI being transmitted with the last byte.
RBIN	CALL "RBIN", A$, E; 3	RBIN inputs binary data from a device on the bus.
RETRY	CALL "RETRY"	RETRY returns program control from an error handler back to the line in which the error occured.
RWLS	CALL "RWLS"; 1, 4; 5	RWLS puts the devices specified in the address list into the Remote With Lockout State (RWLS).
SDC	CALL "SDC"; 2, 6; 25	SDC clears the devices specified in the address list.
SRQOFF	CALL "SRQOFF"	SRQOFF disables Graphic System response to SRQ interrupts.
SRQON	CALL "SRQON"	SRQON enables recognition of SRQ interrupts by the Graphic System.
STBHLP	CALL "STBHLP", 97, S$	STBHLP returns ASCII information about the Codes and Formats standard status bytes.
TALK	CALL "TALK"; 5	TALK makes a talker of the device at the specified address.

4052R14 Option 1A ROM Pack only.

Routine	Example	Purpose
TAPEAPP*	CALL "TAPEAPP",A$	TAPEAPP inputs and appends an entire ASCII file, using the internal mag-tape, to the string in the specified string variable.
TAPEIN*	CALL "TAPEIN",A$	TAPEIN reads an entire ASCII file, using the internal mag-tape unit, into the specified string variable.
THEADER	CALL "THEADER", T$	THEADER finds and opens the file where the tape head is currently positioned and returns the file header or number.
TNAME	CALL "TNAME", "PROM BURN"	TNAME assigns a name to a mag tape file.
TRIM*	CALL "TRIM",A$,500	TRIM changes the current logical length of a string variable.
UNDEF	CALL "UNDEF", X, D	UNDEF determines whether a variable is defined.
UNL	CALL "UNL"	UNL unlistens all devices on the bus.
UNT	CALL "UNT"	UNT untalks all devices on the bus.
VARCLR	CALL "VARCLR", X, 127	VARCLR clears the bits specified by the binary value of the second argument in the variable or array specified as the first argument.
VARSET	CALL "VARSET", X, 6	VARSET sets the bits specified by the second argument in the variable or array specified as the first argument.
VARTST	CALL "VARTST", A, 4, Z	VARTST tests the bits specified by the second argument in the value of the first argument.

* 4052R14 Option 1A ROM Pack only.

Routine	Example	Purpose
VLIST	CALL "VLIST"	VLIST lists all current BASIC variables and their values on the screen.
WAIT	CALL "WAIT", 100	WAIT delays the execution of the program by the specified number of seconds.
WBIN	CALL "WBIN", Q, E; 5	WBIN sends binary data from the Graphic System to devices on the bus.

4051 ROM Pack only. This routine is internally implemented in the 4052 and 4054 Graphic Systems.

Appendix C

SRQ INTERRUPT SET-UP

Some routines have 2 versions. In the case of SRQ handling there is an SRQOFF version 1, SRQOFF version 2, SRQON version 1, and an SRQON version 2. The version 1 routines work with the 4051R14 and 4052R14 ROM Packs; the version 2 routines work with the 4052R14 Option 1A ROM Pack. The reason for the different versions is that SRQ handling works somewhat differently between the 4051, 4052, and 4054 Graphic Computing Systems and the 4052A/4054A Graphic Computing Systems (non-"A" versions [version 1] versus "A" versions [version 2]).

4051, 4052 AND 4054 SRQ INTERRUPT SET-UP

There are two situations we are concerned with here: without a 4051R14 or 4052R14 ROM Pack (GPIB ROM Pack) installed, and with a 4051R14 or 4052R14 ROM Pack (GPIB ROM Pack) installed.

Without a GPIB ROM Pack Installed

Without a GPIB ROM Pack installed, interrupts are always enabled. ON/OFF SRQ statements do not turn on and off the SRQ interrupt system; they only control whether there is a line number specified to branch to if an SRQ interrupt occurs. When an SRQ interrupt occurs, the Graphic System checks to see if there is a line number to branch to (as specified by an "ON SRQ" statement). If there is, the program is directed to that line number and processing continues. If not, then the Graphic System prints the "no ON SRQ" error message and halts. The reasons there may not be a line number specified as a branch address at the time of an SRQ interrupt are as follows:

- No "ON SRQ" statement has been executed.
- An "ON SRQ" was subsequently disabled by an "OFF SRQ" statement.
- There is an address, an SRQ interrupt has occured, the program has branched to the SRQ handler, and now the SRQ branch address has been "stacked" (is not available until a RETURN from the handler is executed). This represents a "window" within a handler where after a POLL statement has been executed, and the program is still in the handler, a subsequent (new) SRQ will cause a halt. Again, this is because the branch address has been temporarily disabled (it is "stacked" and not replaced until the RETURN is executed from the handler back to the main program).

In many cases, it is undesirable for the Graphic System to halt on a "No SRQ ON" condition. For instance, when the Graphic System is being powered on, when programs are being loaded in and getting things set up to handle interrupts, etc.

There are also times when, even though an "ON SRQ" statement has been executed and a handler is in place, it is desirable to disable interrupts. Without a GPIB ROM Pack, this is not possible in non-"A" systems.

With a GPIB ROM Pack Installed

The GPIB ROM Pack adds two SRQ routines: SRQON and SRQOFF. They are there to enable and disable SRQ interrupts respectively. When a Graphic System is being powered on, an automatic call to SRQOFF is made to prevent SRQ interrupts from being recognized (acknowledged). This gives user programs time to get things initialized and set up to handle SRQ interrupts, including executing an "ON SRQ" statement, before interrupts are enabled by a call to SRQON.

SRQ interrupts can be disabled anytime by calling SRQOFF. In general, there is never any reason to use the "OFF SRQ" statement with a GPIB ROM Pack installed in a non-"A" Graphic System.

SRQ Statement Summary

- The "ON SRQ" statement determines (specifies) the SRQ handler branch address (line number).
- CALL "SRQON" enables SRQ interrupts.
- CALL "SRQOFF" disables SRQ interrupts.

There is an example in Appendix A using SRQON and SRQOFF routines for non-"A" Graphic Systems (4051, 4052, and 4054).

4052A AND 4054A SRQ INTERRUPT SET-UP

In the "A" Graphic Systems, the "ON SRQ" statement has been extended — it specifies the SRQ branch line number for the SRQ handler *and* enables SRQ interrupts.

The "OFF SRQ" statement has been extended so it disables SRQ interrupts.

The "A" Graphic Systems power on with SRQ interrupts disabled. Since it is the "ON SRQ" statement that enables interrupts, it is now impossible for an SRQ interrupt to be recognized until (unless) an "ON SRQ" statement has been executed.

SRQON and SRQOFF in "A" Systems

There is now no direct reason for having SRQON and SRQOFF routines in the 4052R14 Option 1A ROM Pack. However, they are there to maintain compatibility (prevent errors) with programs originally written for non-"A" Graphic Systems using either the 4051R14 or 4052R14 ROM Packs. They differ in the following way:

- SRQON is a "no-op," meaning it performs no action.
- SRQOFF does the same thing as the "OFF SRQ" statement does (disables SRQ interrupts from being acknowledged); it is redundant as far as "A" systems are concerned.

Appendix D

4052A/4054A ERROR HANDLING

There are four new routines added to the 4052R14 Option 1A ROM Pack to aid in error handling in 4052A and 4054A Graphic Computing Systms. The routines are ERRLIST, ASKERR, CLRREP, and RETRY. They work in conjunction with the "ON SIZE" and "ON TIMEOUT" statements.

ERRLIST, ASKERR, and CLRREP work together to trap errors, identify errors, and keep track of how many times a given error has occured consecutively in the same program line.

RETRY allows the program to make changes after a line has caused an error, or after a timeout has occured, and then "RETRY" the line again.

Detailed Information. Syntax descriptions and detailed explanations for ERRLIST, ASKERR, CLRREP, and RETRY can be found in Section 2 of this manual. Information on the "ON SIZE" and "ON TIMEOUT" statements can be found in the relevant Graphic Computing System reference manual. Appendix A in this manual has an example of error handling using the above listed routines and statements.

ON SIZE. The "ON SIZE" statement, without ERRLIST, can trap 6 errors. When the trap occurs, there is no way to tell which error of the 6 caused the trap. ERRLIST and ASKERR expand the scope of "ON SIZE" to trap and identify errors.

ERRLIST. ERRLIST is used to specify which errors can cause an "ON SIZE" trap; the range is from error code 1 to error code 127.

ASKERR/Error Codes. When an "ON SIZE" trap occurs, ASKERR is used to find out which error caused the trap.

ASKERR/Repetition Count. There is an internal repetition count that is incremented each time an error occurs or each time a GPIB timout (as defined by "ON TIMEOUT,") occurs. The value of the repetition count can be queried by ASKERR. The repetition count is reset to zero when one of the following occurs

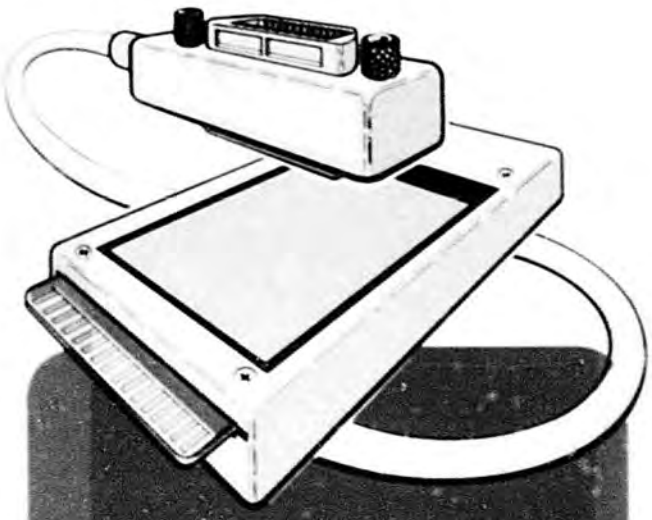
- A call to CLRREP (clear repetition count) is executed.
- A call to ERRLIST is executed.
- A trap occurs with a different error code than that which caused the last trap.
- A trap occurs in a statement with a different line number than that which caused the last trap

The repetition count counts all errors, not just those specified in ERRLIST.

RETRY. RETRY is used to exit from (return from) an "ON SIZE" or "ON TIMEOUT" handler. It works similar to the RETURN statement except it returns to the beginning of the statement in which the trap occured (RETURN returns to the statement *following* the statement that caused the trap)

INDEX

Tektronix



4050R14

GPIB ENHANCEMENT ROM PACK

REFERENCE GUIDE

MANUAL PART NO. 070-4314-01 First Printing JUN 1982
PRODUCT GROUP 14 Revised JAN 1983

ABOUT THIS GUIDE

This reference guide summarizes the extensions to 4050 Series BASIC provided by the 4051R14, 4052R14, and 4052R14 Option 1A GPIB Enhancement ROM Packs.

A cross-reference listing of routines by functional groups is provided.

The Routine Summary contains user information for each routine:

- The descriptive form of the routine and any parameters (omitted if same as example).
- The purpose of the routine.
- A brief example of the routine.

The routines are arranged in alphabetical order in this summary.

Refer to the Instruction Manual for complete information on the routines.

NOTE

Some routines have two versions. Version 1 routines are found in 4051R14 and 4052R14 ROM Packs. Version 2 routines are found in 4052R14 Option 1A ROM Packs only.

Refer to the Instruction Manual for syntax symbols and rules, for address arrays, address lists, and error lists.

FUNCTIONAL CROSS-REFERENCE OF ROUTINES

GPIB Polling Routines

These routines enhance the 4050 Series GPIB polling capabilities.

SRQ Interrupts

SRQOFF
SRQON

Serial Poll

CONFIG
POLL

Parallel Poll

PPD
PPE
PPOLL
PPU

Standard Instrument Error Message Decoding

ERRHLP
STBHLP

Arrays

ARSIZE
UNDEF

GPIB Mnemonics Routines

These routines allow the use of "CALL" statements for standard GPIB interface messages.

DCL
GET
GTL
IFC
LISTEN
LLO
LOCS
RWLS
SDC
TALK
UNL
UNT

General Purpose Routines

The routines allow the user additional GPIB programming capabilities.

BININ
BINOUT
DECHEX
HEXDEC
LAST
MTPACK
NEWTAP
PRISTR
RBIN
THEADER
TNAME
VARCLR
VARSET
VARTST
VLIST
WAIT
WBIN

4051 ROM Pack only. These routines are internally implemented in the 4052 and 4054.

Error Handling

The following four 4052R14 Option 1A routines enhance error handling in 4052A/4054A Graphic Computing Systems.

ERRLIST
ASKERR
CLRREP
RETRY

String Input

The following three 4052R14 Option 1A routines aid in the duplication of ASCII data or program files on 4052A/4054A Graphic Computing System's internal mag tape units.

TAPEIN
TAPEAPP
TRIM

ROUTINE SUMMARY

CALL "ARSIZE",variable name,target for row dimension,target for column dimension

Returns the currently dimensioned size of an array.

100 CALL "ARSIZE",A,B,C

CALL "ASKERR",error index,repetition count
or
CALL "ASKERR",target for error message

4052R14 Option 1A ROM Pack only. Supplies information on which error caused a trap and the number of times the error has occurred on a consecutive basis.

100 CALL "ASKERR",Err,Rep
110 CALL "ASKERR",A$

CALL "BININ",input mode,target for data,target for error code;talk address

Inputs binary block data from a device on the bus.

100 CALL "BININ","UNPK",Q,E,5

CALL "BINOUT",output mode,source of data for block,target for error code; listen addresses

Sends data in binary block format from the 4050 Series System to devices on the bus.

100 CALL "BINOUT","UNPK",Q,E,5,3,11,24

CALL "CLRREP"

4052R14 Option 1A ROM Pack only. Sets the error or timeout repetition counter to zero (0).

100 CALL "CLRREP"

CALL [illegible] ;timeout,target for error code,target for device addresses

Determines the active devices on the bus and returns these addresses in the array specified as an argument

100 DIM A(15)
110 REM CONFIGURE FOR PRIMARY ADDRESS ONLY
120 CALL CONFIG 5,E,A
130 IF E THEN 500

CAL [illegible]

Clears all devices on the bus

100 ALL DC

CALL [illegible] decimal value,target for HEX representation

Converts the decimal value of an expression to its equivalent in ASCII HEX

100 ALL DECHEX 279,H$

CALL [illegible] numeric expression,string variable

Returns information about the Tektronix Codes and Formats error codes

200 ALL ERRHLP E,E$
210 E$ THEN 250
220 REM EVENT NOT FOUND IN TABLE
250 PRINT E$

Refer to the Instruction Manual for error codes and ASCII information

CALL [illegible] ,error list

4052R14 Option 1A ROM Pack only. Enables specified errors or timeouts to be trapped and handled by a user's ON SIZE or ON TIMEOUT statement parsing routines

100 ALL ERRLST 15,E list 18,23

CALL "GET";address(es)

Sends Group Execute Trigger to devices at addresses listed

100 CALL "GET",2,8,13,4

CALL "GTL";address(es)

Returns devices specified in the address list to either Local State (LOCS) or Local With Lockout State (LWLS)

100 REM DEVICES AT ADDRESSES 3,5,7 ARE SET
110 REM TO LOCAL
120 CALL "GTL",3,5,7

CALL "HEXDEC",HEX representation,target for decimal value

Converts an ASCII HEX number representation to decimal

100 CALL "HEXDEC","F078",T

CALL "IFC"

Pulses the GPIB IFC line to clear the GPIB interface

100 CALL "IFC"

CALL "LAST",target for file number or header

Finds and returns the file number or header of the last file on the tape

100 CALL "LAST",X
200 CALL "LAST",H$
300 CALL "LAST"

CALL "LISTEN";address(es)

Makes the devices specified in the address list into listeners.

100 CALL "LISTEN",5,27

CALL "LLO"

Puts all instruments on the bus into either Local With Lockout State (LWLS) or Remote With Lockout State (RWLS)

100 CALL "LLO"

CALL "LOCS"

Puts all devices in the LOCAL state.

100 CALL "LOCS"

CALL "MTPACK"

Winds and rewinds the tape to correct poor tape spooling that may occur.

100 CALL "MTPACK"

CALL "NEWTAP",target for cartridge status

Version 1. Finds the present file location and returns the tape cartridge status in a numeric code.

100 CALL "NEWTAP",A

CALL "NEWTAP",target for cartridge status,target for write protect status

Version 2. Positions the tape at the beginning of the present file (current position) and returns the tape cartridge status and write protect status as numeric codes.

Cartridge Status

0 – no tape inserted
1 – tape inserted, not first tape operation
2 – tape inserted, first tape operation

Write Protect Status

1 – enabled for writing
0 – write protected

100 CALL "NEWTAP",C,W

CALL "POLL",timeout,target for device identifier,target for status byte;address(es)

Used to serial poll the devices specified in the address list.

100 DIM A(15)
110 CALL "CONFIG",E,A
120 ON SRQ THEN 500
130 CALL "SRQON"
200 REM MAIN PROGRAM
500 REM SRQ INTERRUPT HANDLER
510 CALL "POLL",S,I,S,A
520 REM SERVICE INTERRUPT
700 RETURN

CALL "PPD";address(es)

Unconfigures selected devices.

200 CALL "PPD",3,7

CALL "PPE",sense of parallel poll response,line on which device responds;address(es)

Configures the devices that are to respond to a parallel poll.

200 CALL "PPE",1,7;3,16,23

CALL "PPOLL",target variable for parallel poll response

Performs a parallel poll of the devices previously configured.

100 CALL "PPOLL",A

CALL "PPU"

Unconfigures all devices.

100 CALL "PPU"

CALL "PRISTR",string to be output;address(es)

Outputs a string to the GPIB interface without EOI being transmitted with the last byte.

100 CALL "PRISTR","LLST",5
110 CALL "BINOUT",Q,E,5

CALL "RBIN",output mode,target for data,target for error code;talk address

Inputs binary data from a device on the bus.

300 CALL "RBIN","PACK,UNSI",Q,E,1

CALL "RETRY"

4052R14 Option 1A ROM Pack only. Returns program control from an error handler back to the line in which the error occurred.

100 CALL "RETRY"

CALL "RWLS",address(es)

Puts the devices specified in the address list into the Remote With Lockout State (RWLS).

200 CALL "RWLS",1,4

CALL "SDC",address(es)

Clears the devices specified in the address list.

100 CALL "SDC",2,16,25

CALL "SRQOFF"

Disables 4050 Series System Controller response to SRQ interrupts. See Instruction Manual for differences between Versions 1 and 2.

100 CALL "SRQOFF"

CALL "SRQON"

Enables recognition of SRQ interrupts by the 4050 Series System. See Instruction Manual for differences between Versions 1 and 2.

100 ON SRQ THEN 200
110 CALL "SRQON"

CALL "STBHLP",status byte,target for ASCII information

Returns ASCII information about the Codes and Formats standard status bytes obtained from a serial poll.

100 CALL "STBHLP",97,S$

Refer to the Instruction Manual for status byte and ASCII information.

CALL "TALK",talk address

Makes a talker of the device at the specified address.

200 CALL "TALK",5

CALL "TAPEAPP",target for mag tape file

4052R14 Option 1A ROM Pack only. Inputs and appends an entire ASCII file, using the internal mag tape, to the string in the specified string variable.

100 FIND 1
110 CALL "TAPEAPP",A$

CALL "TAPEIN",target for mag tape file

4052R14 Option 1A ROM Pack only. Inputs an entire ASCII file into a character string using the internal mag tape unit.

100 FIND 1
110 CALL "TAPEIN",A$

CALL "THEADER",target for header or file number

Finds and opens the file where the tape head is currently positioned and returns the file header or number.

100 CALL "THEADER"
200 CALL "THEADER",T$
300 CALL "THEADER",F

CALL "TNAME",file name

Assigns a name to a mag tape file.

100 CALL "TNAME","ROMS"

Refer to the Instruction Manual for error messages.

CALL "TRIM",string to be altered,new length

4052R14 Option 1A ROM Pack only. Changes the current logical length of a string variable without altering or deleting any of the data within the string variable.

100 CALL "TRIM",A$,5

CALL "UNDEF",variable to be tested,target for the result

Determines whether a variable is defined.

91 CALL "UNDEF",A,B
92 IF NOT(B) THEN 200
93 GOTO 100
100 REM INITIALIZING CODE
200 REM MAIN PROGRAM

CALL "UNL"

Unlistens all devices on the bus.

```
100 CALL "UNL"
```

CALL "UNT"

Untalks all devices on the bus.

```
100 CALL "UNT"
```

CALL "VARCLR",variable to be cleared,bits to clear

Clears the bits specified by the binary value of the second argument in the variable or array specified as the first argument.

```
100 CALL "VARCLR" A,16
```

CALL "VARSET",variable to be set,bits to set

Sets the bits specified by the second argument in the variable or array specified as the first argument.

```
100 REM SET BIT 3 IN THE BINARY VALUE OF A
110 CALL "VARSET",A,8
```

CALL "VARTST",variable to be tested,bits to be tested,target for test result

Tests the bits specified by the second argument in the value of the first argument.

```
100 CALL "VARTST",A(7),4,Z
110 IF Z THEN 200

200 REM ROUTINE FOR COMPARISON
500 CALL "VARTST",A,3,Z
510 IF Z = 3 THEN 630
600 J = 693
610 B = 17
620 CALL "VARTST",J,B,M
630 PRINT M
```

CALL "VLIST"

Version 1: Lists all current BASIC variables and their values on the screen.

```
100 CALL "VLIST"
```


CALL "VLIST",target to receive information

Version 2: Lists all the current BASIC variables and their values on the screen or stores them in the string variable.

```
100 CALL "VLIST" ! list to screen
110 CALL "VLIST",A$ ! store in A$
```

CALL "WAIT",delay time

Delays the execution of the program by the specified number of seconds.

```
100 CALL "WAIT",100
```

CALL "WBIN",output mode,source of data,target for error code;listen address(es)

Sends binary data from the 4050 Series System Controller to devices on the bus.

```
200 CALL "WBIN","NEOI",A$,E;3,7,14
```

Tektronix

COMMITTED TO EXCELLENCE

ASCII & GPIB CODE CHART

BITS B4 B3 B2 B1	CONTROL		NUMBERS SYMBOLS		UPPER CASE		LOWER CASE	
0 0 0 0	NUL	DLE	SP	0	@	P	`	p
0 0 0 1	SOH	DC1	!	1	A	Q	a	q
0 0 1 0	STX	DC2	"	2	B	R	b	r
0 0 1 1	ETX	DC3	#	3	C	S	c	s
0 1 0 0	EOT	DC4	$	4	D	T	d	t
0 1 0 1	ENQ	NAK	%	5	E	U	e	u
0 1 1 0	ACK	SYN	&	6	F	V	f	v
0 1 1 1	BEL	ETB	'	7	G	W	g	w
1 0 0 0	BS	CAN	(	8	H	X	h	x
1 0 0 1	HT	EM	)	9	I	Y	i	y
1 0 1 0	LF	SUB	*	:	J	Z	j	z
1 0 1 1	VT	ESC	+	;	K	[	k	{
1 1 0 0	FF	FS	,	<	L	\	l	|
1 1 0 1	CR	GS	-	=	M	]	m	}
1 1 1 0	SO	RS	.	>	N	^	n	~
1 1 1 1	SI	US	/	?	O	_	o	DEL
	ADDRESSED COMMANDS	UNIVERSAL COMMANDS	LISTEN ADDRESSES		TALK ADDRESSES		SECONDARY ADDRESSES OR COMMANDS (PPE)	(PPD)

KEY

Tektronix, Inc.
P.O. Box 500
Beaverton, Oregon 97077