DV-488

IEEE-488 Interface Board

Keithley MetraByte Corporation

A Subsidiary of Keithley Instruments, Inc.
440 Myles Standish Boulevard
Taunton, Massachusetts 02780

Part Number: 24821

First Printing: June 1989

Keithley MetraByte Corporation
440 Myles Standish Boulevard
Taunton, Massachusetts 02780

WARNING
Keithley MetraByte Corporation assumes no liability for damages consequent to the use of this product. This product is not designed with components of a level of reliability suitable for use in life support or critical applications.

WARRANTY INFORMATION

All products manufactured by Keithley MetraByte are warranted against defective materials and workmanship for a period of one year from the date of delivery to the original purchaser. Any product found to be defective within the warranty period will, at the option of Keithley MetraByte, be repaired or replaced. This warranty does not apply to products which have been damaged by improper use.

TABLE OF CONTENTS

1.0 INTRODUCTION

General Description (MBC-488 & uCMBC-488)

MetraByte's MBC-488 and uCMBC-488 are General Purpose Instrument Bus (GPIB) interface cards conforming to the IEEE-488 standard (November 1978 and later. See Appendix A). The MBC-488 or uCMBC-488 plug directly into any IBM PC/XT/AT expansion slot (including the PC/XT short slot J8) as well as virtually any other PC bus compatible or the IBM PS/2. The (uC)MBC-488 is supplied with a standard IEEE-488 interface connector. It can be configured as a controller, talker or listener and supports the 12 interface messages applicable to the IEEE-488 standard (See Appendix B). Full software support (DV488PC.SYS & DV488UC.SYS), via DOS device drivers, is supplied with the both boards (PC and uChannel Bus compatible boards). The DV488 software allows interfacing with up to 14 other devices and provides system timing and interrupt control (when applicable) for complete GPIB operation.

The hardware and supporting software allow Programmed and Direct Memory Access (DMA) data transfer with data transfer rates of 2 KBytes/Sec in programmed mode and 400 KBytes/Sec in DMA mode. The supplied device drivers are general purpose in nature and may be used to interface with virtually any high and low level computer language supporting device drivers. Example routines in various languages (C, FORTRAN, BASIC, TurboPASCAL, and QuickBASIC, etc.) are provided and may be listed and/or copied as desired.

(uC)MBC-488 configuration and installation is quick and easy with the IBM PC version being configured with hardware switches and the PS/2 version being configured via the PS/2's Setup and Installation programs supplied.

Product Returns and RMA Procedure

MetraByte's warranty and product return policies are fully described in the MetraByte Product catalog. In the unlikely event that you wish to return any MetraByte product, whether for repair, replacement, or any other reason, you must first call the Technical Support Department (508-880-3000) between the hours of 9:00 AM and 4:30 PM to receive a Return Materials Authorization (RMA) number. This assures that your account will be properly credited (for product returns) and/or allows MetraByte to track your products to better serve you in the event that you require support or service.

2.0 INSTALLATION

MBC-488 & the IBM PC/XT/AT

MetraByte's MBC-488 board uses 16 consecutive address locations in the IBM PC's I/O address space. Some I/O address locations will be occupied by internal I/O and other peripheral cards. The base address of the MBC-488 can be set by the Base Address DIP switch to any 16-bit boundary. This address space extends from decimal 256-1023 (Hex 100-3FF). The table below summarizes the usual address assignments and is reproduced from the "IBM Technical Reference Manual":

ADDRESS (Hex)	DEVICE	ADDRESS (hex)	DEVICE
000-0FF	Internal system	378-37F	LPT1:
200-20F	Game	380-38C	SDLC comm.
210-217	Expansion unit	380-389	Binary comm. 2
220-24F	Reserved	3A0-3A9	Binary comm. 1
278-27F	Reserved	3B0-3BF	Mono dsp/LPT1:
2F0-2F7	LPT2:	3C0-3CF	Reserved
2F8-2FF	COM2:	3D0-3DF	Color graphics
300-31F	Prototype card	3E0-3E7	Reserved
320-32	Hard disk	3F0-3F7	Floppy disk
3F8-3F	COM1:		

This list covers the standard I/O options. You may, however, have other peripherals e.g. hard disk drives, graphics boards, etc. that use I/O address space. Check the manuals that came with your other peripherals to avoid address conflicts. Review the above list and choose an address space 16-bits wide and set it via the Base Address DIP switch. The DV488 device driver (supplied) supports 2 MBC-488 boards. If you are using more than one MBC-488 board, separate Base Addresses are required to avoid address conflicts.

A second DIP switch (marked DMA/INT) is to the left of the Base Address switch and controls selection of DMA level, interrupts and operation in the J8 short slot of an IBM PC/XT. The first slider selects the DMA level. There are 4 DMA levels provided by the internal (PC) 8237 DMA controller. Level 0 is used internally by the system for memory refresh and is not available. Levels 1, 2 and 3 are available, with level 2 usually being used by the floppy disk drive. Level 1 is usually available, and level 3 is available on standard IBM PC (without hard disk) or IBM PC/AT. On the XT model, level 3 is used by the hard disk. Select the DMA level desired and set the switch accordingly. The MBC-488 can be operated in non-DMA modes in the rightmost short slot (J8) of a PC/XT by setting switch J8 ON (otherwise it should be OFF). This slot is normally intended for driving the expansion chassis interface and has slightly different signals from the other slots. If the DMA mode of operation is disabled, both DMA and interrupts will be disabled and their settings will be meaningless.

There are 6 interrupt levels on the bus (2 thru 7). Interrupts are used for Direct Memory Access (DMA) data transfer only. With the exception of level 6 which is used by the floppy disk adapter, most of the other levels may be available. The MBC-488 is shipped with its DIP switches set for a base I/O address of Hex 300, interrupt level 5, DMA level 1, and the J8 slot feature disabled. These are usually good default values, and you may not need to alter them. If you want to check them or change them before you install the board in your computer, insert the software disk in your floppy drive and enter:

A> INSTALL

The INSTALL.EXE program is a self-explanatory program (INSTALL.EXE) that gives you a pictorial view the correct switch settings on the MBC-488 for any combination of addresses, interrupt level, DMA level etc. Simply set the switches the way you see them on the screen and press <ESC> to exit to DOS. You will also see warning messages for possible conflicting addresses. If you receive a warning for a device that is not in your computer, it can safely be ignored. These cautions apply strictly for IBM standard devices (although the same mapping is followed by most compatibles) and may not be totally foolproof as far as non-IBM peripherals are concerned. If your MBC-488 does not appear to work correctly, or interferes in some way with other devices on your computer e.g. disk drives etc. or your computer will not boot up, remove the MBC-488 and try a different I/O address, interrupt or DMA level. Once you have set the base I/O address, make a note of its value as you will need to use it in the SYSCON initializing command in your programs. All the other switch settings are read by the driver (software) so you can forget the interrupt and DMA level settings. Prior to installing the MBC-488 in your computer, SHUT OFF THE POWER and discharge any static electricity that you may be carrying. The MBC-488 will fit in any of the regular full depth slots of the IBM PC/XT/AT or the "half" slots of the IBM XT or Portable computer. You may feel some resistance when pushing the IEEE-488 connector through the rear panel of your computer as the slots are close to a clearance fit for this type of connector. The base of the connector plate has been machined with a slight undercut, so once seated in the slot it should slide up and down freely. When you mate a standard IEEE-488 cable to the connector, you may find it blocks adjacent slot access to some extent due to its width. This can be avoided by placing the MBC-488 in an end slot, or adjacent to a board that has no connector on it. If the connector becomes a problem, MetraByte's IE-488 board avoids this difficulty through the use of a special adapter cable. If you later remove the MBC-488 board, MetraByte recommends that you retain the special electrostatically shielded packaging and use it for storage.

Backing up the DV488 Diskette

It is important to make a back-up copy of the software supplied with MBC-488. The software is not copy protected so you may make as many copies as desired. It is supplied on a DOS 2.1 and higher double sided (360K) floppy disk format.

uCMBC-488 & the PS/2

The uCMBC-488 adheres to the design philosophy of the IBM PS/2 Programmable Option Select (POS) rules and, therefore, has no jumpers or switches. The uCMBC-488 has been assigned a Board Identifier Number (BIN) of 5018 (hex) which is registered with IBM Corp. Actual configuration of the uCMBC-488 board is done in software and requires a copy of the IBM Reference Diskette and several files supplied by MetraByte. Prior to plugging the board into the PS/2 expansion bus, you must install its BIN onto the reference disk (special ID numbers may be obtained from MetraByte's Tech Support Dept. 508-880-3000). This will then be used only once to install the system configuration onto battery backed RAM. The reference diskette will not be used again unless you reconfigure your system. It should be stored in a safe place after use. The uCMBC-488 has several static sensitive components so that caution should be used when removing the board from it's shipping container.

1) Make a back-up copy of both the PS/2 Reference Disk and the MetraByte uCMBC-488 Utility Disk. Now copy the @5018.ADF file from the MetraByte Utility disk onto the copy of the PS/s Reference disk.

2) Shut OFF the power to the PS/2 and any option boards that are externally powered. Remove the cover from the computer and install the board.

3) Place the copy of the IBM Reference Disk in drive A: and repower the computer.

4) After the computer runs it's power-up check, the error code **165 ... Adapter Configuration Error** will appear, followed by two beeps.

5) After the IBM logo appears, press <Enter> and <PgDn> after the first page of text.

6a) If you intend to use the default Base Address of 300 (HEX), you can automatically configure your system by typing <Y> at the prompt **Run Automatic Configuration.**

6b) If you are not using the default Base Address for the uCMBC-488, type <N> at the prompt. This will return you to the Main Menu where you will select: **3. Set Configuration**. A Configuration Menu will appear. Select: **2. Change Configuration**. Use the cursor keys to through the configuration list until you see the **MetraByte uCMBC-488** in the appropriate expansion slot. Use the <F5> and <F6> keys to choose a new Base Address. The press <F10) to install this new configuration in RAM.

7) You will not need the copy of the Reference Diskette again unless or until you reconfigure your system. It should be stored in a safe place.

Modifying the @5018.ADF File

IBM's Configuration program has 16 possible choices for each selection. In the case of Base Address, the uCMBC-488 supports 255 possible addresses. Use the GENADF.EXE file to modify @5018.ADF from 300 (HEX) to any other address you prefer. Be sure the file (GENADF.EXE) is on the copy of the working (reference) diskette and then proceed as above to change the configuration.

PS/2 Installation Errors

Several common errors may be encountered if you did not follow the above instructions explicitly. They are:

Error Message	Meaning	Solution
165	Unrecognized Adapter Board	Reconfigure the System.
* Conflict	Two or more adapter boards share the same address.	Select a new address for one of the boards.
Set-up cannot find appropriate .ADF file.	.ADF file does not appear on working disk.	Copy .ADF to the working disk.
Set-up cannot read .ADF file.	Unreadable .ADF file on working disk.	Use **GENADF.EXE** to recreate correct .ADF file.

3.0 DEVICE DRIVER (DV488)

Introduction and General Overview

DV488 is the overall name given to MetraByte's (uC)MBC-488 device driver and example routine software package. It is supplied free with the purchase of either the MBC-488 or the uCMBC-488 GPIB interface boards. The software contains many example programs. The examples illustrate serial and parallel data transfer in various formats including HIGH, LOW, PACKED byte, and STRING data. Additionally, single/dual board configurations, (uC)MBC-488 usage as talker/listener, and DMA/programmed data transfer modes are shown. The examples are done in BASICA, C, FORTRAN, TurboPASCAL, and QuickBASIC. The two Device Drivers are; DV488PC.SYS and DV488UC.SYS for the IBM PC/XT/AT and PS/2, respectively. (See Appendix C for a complete file listing and brief description of each).

Installing DV488 Device Driver

Prior to operation of either the MBC-488 or uCMBC-488, the appropriate device driver (DV488PC or DV488UC) must be installed. This is done by running DVSETUP.EXE (supplied on your diskette). To run DVSETUP.EXE, type:

C:>DVSETUP<cr>

Once invoked, DVSETUP will prompt you for the (sub)directory containing DV488. Note that VIPARSE.SYS must be in the same (sub)directory. CONFIG.SYS is a readable file. It should look something like this.

```
BUFFERS = 20
FILES = 20
DEVICE = (path:)VIPARSE.SYS /HK=ALT H /MK=ALT M /SK=ALT TAB
DEVICE = (path:)DV488PC.SYS
```

If VI.SYS was selected (for PCIP instruments, see Appendix C) as the device driver, add the following line to CONFIG.SYS. It must be added before the line selecting VI.SYS.

```
DEVICE = (path:)ANSI.SYS
```

CONFIG.SYS is automatically executed at system (computer) start-up and installs the various device drivers required.

PROGRAM Mode of Operation

MetraByte's (uC)MBC-488 may be operated in one of two data transfer modes; a Programmed Mode and a DMA mode. The Programmed mode requires that DMA and Interrupt levels be disabled on the MBC-488 whereas the uCMBC-488 operates entirely via software so that nothing need be done in order to operate in either mode.

A set of image specifiers are used in conjunction with the ENTER/OUTPUT commands to format data prior to transfer to strings, integer variables, etc. These image specifiers also allow the addition of parity to ASCII data and control characters <cr> & <lf> for string data termination.

DIRECT MEMORY ACCESS (DMA) Mode of Operation

Both the MBC-488 and the uCMBC-488 in conjunction with the supplied device driver software are capable of DMA data transfer at very high speed. Data transfer speeds on the order of 400Kbaud are possible on the PS/2 and similar increases are possible on the IBM PC/XT/AT. This tremendous increase in transfer speed is possible by essentially bypassing the CPU, program variables and other associated program overhead. The DMA mode of operation is interrupt driven and it's ultimate transfer speed is subject to various interrelated factors, each of which contributes to a final top end. Rarely, will actual data transfer speeds attain the maximum possible. During DMA data transfer, data is dumped (or retrieved) directly to/from sequential memory locations. Generally, the performance of the computer's DMA controller far exceeds the ability of the device or instrument transferring data to/from the GPIB or the speed at which the instrument is capable of collecting data. For a more detailed discussion concerning DMA, see either chapter 5 (MBC-488) or Chapter 6 (uCMBC-488).

In order to implement DMA data transfers, remember that the (uC)MBC-488 must be configured for interrupt implementation. See the hardware configuration section of this manual for a complete discussion.

Data Transfer Sequence

Regardless of the Mode of Operation (Programmed or DMA), the command sequence is the same:

OUTPUT	ENTER
1. UNTALK	1. UNTALK
2. UNLISTEN	2. UNLISTEN
3. Computer's TALK Address	3.Computer's LISTEN Address
4. Device LISTEN Address(es) (secondary address(es))	4.Device TALK Address(es) (secondary address(es))
	5. UNTALK

Once data transfer is complete, the bus is untalked if in control. When communicating with devices using secondary addressing, the device's extended address is specified by separating the primary address with a period (.).

When the computer is not the active controller on the bus, no other device may be addressed by the computer (controller commands). However, the computer may transfer data as a talker/listener if configured as such by the controller. During an ENTER command, the computer waits for the active talker on the bus to finish data transmission prior to it's transmission if it has been addressed by the talker/listener. Address codes must be satisfied during GPIB activity or error(s) will occur. DOS handles such errors, but often in a rather cryptic fashion. Most High and Low level languages are capable of handling DOS errors allowing the user to perform WAIT loops or write error handling routines thus allowing a high degree of debugging and increasing overall program efficiency. The OUTPUT statement waits until the controller addresses the computer to talk and the ENTER command waits until the controller addresses the computer to listen. The computer may assume control if the controller in charge sends a Take Control Message after it programs the computer as a talker. During DMA data transfer the computer waits until it is addressed before transferring data. A driver "Device Time-out" error may occur if data is not successfully transferred. The Time-out delay is adjustable using the TIMEOUT command.

4.0 GPIB PROGRAMMING with DV488

Programming and Device Driver Access

Data transfer operations to or from either the MBC-488 or the uCMBC-488 via the GPIB are defined by a set of commands. The syntax or structure of these commands is critical. Each element within a command must be separated by a single space and, normally, the first two letters of a command must be capitalized. Trailing arguments are not case dependent (may be upper or lower case). Since all data, whether transmitted or received, must be formatted prior to GPIB activity these data must pass through the driver. This requires that the device driver be accessed. For data transmission, the driver is "OPENed" for OUTPUT whereas data reception requires the driver to be "OPENed" for INPUT as follows:

OPEN "$DV488UC" FOR OUTPUT AS #1

and/or

OPEN "$DV488UC" FOR INPUT AS #2

Once the driver has been accessed, data may be transmitted or received as follows:

To XMIT: **PRINT #1, "OUTPUT 4 $ E +"**

where: 4 is the device #
$ is a String variable containing data
E is Even Parity
+ is addition of <cr><lf> to the end of the data

and

To RCV: **PRINT #1, "ENTER 4 $ +"**
INPUT #2, DVMDT$

NOTE: The driver (DV488) provides for various errors (syntax, etc) that may occur. It does not provide for error handling, however, since this is done via DOS (Disk Operating System). DV488 errors may be accessed from the (uC)MBC-488 as shown in Chapter 4 (GPIB Programming with DV488). It is left to the user to assure that all required conditions have been met prior to GPIB activity.

Image Specifiers (Data Types) and Terminators

The image specifiers define data types as falling into two separate types with four categories. These are listed below along with the specifiers for each:

$	STRING
$ count	STRING with optional count (ENTER only)
B	BYTE packed bytes of an integer array
RB	BYTE packed bytes of an integer array in REVERSE Order
BL	BYTE low byte of integer array (WORD)
BH	BYTE high byte of integer array (WORD)

Image terminators are required for the devices on the GPIB to recognize the End Of Data transfer. Each terminator acts somewhat differently so that you should be aware of the subtleties. The image terminators operate the same way in all transfer modes. However, during DMA transfers, the last data byte plus the image terminators, if requested, are sent via the system CPU in order to clear the (DMA) interrupt for other computer operations (the interrupt service routine handles this last byte plus terminators prior to returning to normal operation).

Programmed Terminator	Terminator Sent (OUTPUT command)	Terminator Sensed (ENTER command)
%	NONE	EOI
#	EOI	EOI
+	EOI and <cr><lf>	<lf> or EOI
!	EOI	<cr> or EOI
NONE	EOI	EOI

Command Structure & Syntax

Data transfer operations to/from the (uC)MBC-488 are defined by a set of commands. Case is unimportant. For example:

EN		en
ENT	or	ent
ENTE		ente
ENTER		enter

Language peculiarities may require a detailed understanding of the particular language in order to transfer data. However, the examples provided with the diskette should be adequate for the majority of data I/O. For example, transferring an array (bytes) of data requires the segment, offset and byte count. Since BASICA does not have a VARSEG statement, a driver semaphore must precede the VARPTR (data segment) statement:

```
OUTPUT 12 B + &HFFFF varptr(A%(0)) 25
```

However, QuickBASIC supports the VARSEG statement:

```
OUTPUT 12 B + varseg(A%(0)) varptr(A%(0)) 25
```

In both cases, "A%" is the is the array containing the data to be transferred to device "12". The data is in byte "B" format with "25" bytes being transferred.

Device Driver Commands

COMMAND	*USAGE*
BRD2	PRINT #1, "BRD2 REQUEST"

FUNCTION

This argument is used to direct commands to the second board in the system. It is position sensitive and must be used in the first position of the command line.

COMMAND	*USAGE*
ABort	PRINT #1, "ABORT"

FUNCTION

Command to CLEAR the GPIB. This command will only have effect if the device sending the command is in control of the bus. It's use will halt DMA data transfer currently in progress.

COMMAND	*USAGE*
BUfferclear	OPEN #1 "$DV488PC" FOR OUTPUT AS #1 PRINT #1, "BUFFERCLEAR"

FUNCTION

Used to CLEAR INPUT buffer of the computer prior to accepting data. Normally, it is used immediately after OPENing the DV488 file for output. This is not a GPIB command.

COMMAND	*USAGE*
CLear	PRINT #1, "CLEAR 12 5"

FUNCTION

Used to clear up to 14 devices (6 secondary) to their power-up/default state. When used with a device number, it clears only that device. If device numbers are not specified, all devices on the bus are cleared.

COMMAND	*USAGE*
ENter	PRINT #1, "ENTER 7 $"

FUNCTION

Used to receive data from the GPIB. As mentioned previously, there are five types of data that may be transferred.

STRING data "$" Strings are often the most convenient format for simple data. String data is limited to about 4000 characters and is also the slowest method of data transfer. A count may be used with this format, but it cannot exceed 4000 bytes (default). Data transfer ends at 4000 bytes (or at specified count) or when a valid terminator or EOI is recognized.

Packed byte "B" Data is received and automatically packed into a one dimension integer array (low byte first).

Reverse order "RB" Data is received into the type of array specified and packed in Reverse Order statrting at the objetc's address plus the count. For single precision real arrays (variables), the count must be a multiple of four. Generally used for entering IEEE REAL numbers that are sent by an instrument with the sign being sent first.

HIGH byte "BH" Data is received and placed into every second position (High Byte position) of an integer array.

LOW byte "BL" Data is received and placed into every first position (Low Byte position) of an integer array.

DMA bytes "DMA High speed method of receiving data (up to 400 Kbytes/Sec). Data is received in bytes and is packed into an integer array (low byte first).

NOTE: All byte format transfers will accept up to 64000 characters (bytes). A count is used and data transmission ends when the specified count has been reached or when a valid terminator or EOI has been recognized. Programmed terminators in byte format data transfers may be a problem since hex D <cr> and hex A <lf> may be real data. Each device must be specified if it is to receive data.

COMMAND	*USAGE*
LOcal	PRINT #1, "LOCAL 3"

FUNCTION

Used to cancel the effect of REMote command (by turning off the remote line on the GPIB) when no device(s) are given. If devices are specified, the remote line will not be turned OFF and only the named device will return to local.

COMMAND	*USAGE*
LOCKout	PRINT #1, "LOCKOUT 5"

FUNCTION

Used to disable the front panel LOCAL function for devices supporting full REMOTE/LOCAL operating modes. If the device is not in remote, LOCKOUT will cause the device to enter the local state with lockout, disabling the local function while maintaining all other front panel functions.

LOCKOUT sends the GPIB LLO message

Device address(es) on the GPIB may be specified which will assert the REMOTE line and place those devices in REMOTE with LOCKOUT state.

COMMAND	*USAGE*
OUTput	PRINT #1 "OUTPUT 7 $ +", CMD$

FUNCTION

Used to transfer data to the bus. OUTPUT supports all five data format types.

STRING data "$" Strings are often the most convenient format for sending simple data or commands. String data is limited to about 4000 characters and is also the slowest method of data transfer. No count is used with this type of data format.

PACKED byte "B" Data is transmitted in bytes packed in integer arrays (low byte first).

HIGH byte "BH" Data is transmitted from every second position (High Byte position) of an integer array only.

LOW byte "BL" Data is transmitted from every first position (Low Byte position) of an integer array.

DMA bytes "DMA Highest speed method of transmitting data (up to 400 Kbytes/Sec). Data is transmitted from bytes packed in a integer array (low byte first). The last byte and terminators, if requested, are sent via normal I/O thru the system CPU. This is done in the DMA interrupt service routine which then clears the interrupt line and returns the computer to normal operation.

NOTE: All byte format transmissions will send up to 64000 characters (bytes). A count is used and data transmission ends when the specified count has been reached or when a valid terminator or EOI has been recognized. Programmed terminators in byte format data transfers may be a problem since hex D <cr> and hex A <lf> may be real data. Each device must be specified if it is to transmit data.

COMMAND	*USAGE*
PARpol	PRINT #1, "PARPOL" INPUT #2, STAT%

FUNCTION

Used to return Parallel Poll status byte from the GPIB. No device addresses are used.

COMMAND	*USAGE*
PASctl	PRINT #1, "PASCTL 7"

FUNCTION

Used to transfer control from the present controller to another device on the bus that is capable of control functions. The command sends the TCT GPIB BUS message and waits for a time-out period. A second PASCTL command will verify that control has been passed (error #10 returned). Alternatively, the REQUEST command will return the status of the (uC)MBC-488 board which contains the CIC status bit. If the bit is 0, the board has transferred control.

COMMAND	*USAGE*
PPConf	PRINT #1, "PPCONF 12 PCONF=107"

FUNCTION

Allows the controller to program a device for a specific response to the parallel poll command. The targeted device is programmed to respond to a specific GPIB line (DIO 1-8) and what it's response should be (0 or 1). Only one device may be specified per PPCONF command. P1-P3 is the binary code for the GPIB line and S is the state of the response;

msb							lsb
B7	B6	B5	B4	B3	B2	B1	B0
X	1	1	0	S	P3	P2	P1

COMMAND	*USAGE*
PPucf	PRINT #1, "PPUCF 12"

FUNCTION

Used to disable/clear device(s) for parallel poll response. If a device has been programmed for a specific response to parallel poll, this command clears the device of its response. If no devices address is given, it clears the entire bus of pre-programmed parallel poll configurations.

COMMAND	*USAGE*
REMote	PRINT #1, "REMOTE 7"

FUNCTION

Command used by the controller to establish a remote state so that the device(s) may be controlled by a computer. Device(s) may be addressed or not. If device address(es) are not given, the GPIB asserts the REMOTE line. Once the REMOTE line has been asserted, any device that is subsequently addressed will be set to remote status and remain in that state until a LOCAL command is sensed.

COMMAND	*USAGE*
RXctl	PRINT #1, "RXCTL"

FUNCTION

RXCTL may be used by a device expecting the GPIB controller to transfer control. The command will cause a wait for the time-out period and return an integer indicating status of the transfer (true if control was received, false if not received). No devices are specified.

COMMAND	*USAGE*
REQuest	PRINT #1, "REQUEST" INPUT #2, RQST%

FUNCTION

REQUEST is used to return the hardware status of the (uC)MBC-488 controller board. The status is returned as an integer.

msb							lsb	
B15	B14	B13	B12	B11	B10	B9	B8	
CIC	SRQI	x	x	LA	TA	DMA BUSY	DMA ERR	
B7	B6	B5	B4	B3	B2	B1	B0	Serial Poll
S8	PEND	S6	S5	S4	S3	S2	S1	Register (Read)

COMMAND	*USAGE*
REQuest 2	PRINT #1, "REQUEST 2" INPUT #2, LEVELS%

FUNCTION

REQUEST 2 returns DMA & Interrupt levels

msb							lsb
B7	B6	B5	B4	B3	B2	B1	B0
D3	D2	D1	D0	D3	D2	D1	D0
INTERRUPT LEVEL				DMA LEVEL			
B15	B14	B13	B12	B11	B10	B9	B8
1/0	1/0	1/0	1/0	1/0	1/0	1/0	1/0
		EXTENDED SIGN					

COMMAND	*USAGE*
REQuest 1	PRINT #1, "REQUEST 1", RESPONSE%

FUNCTION

REQUEST 1 is used to write a response pattern to the Serial Poll Register. This pattern is then output by the (uC)MBC-488 when it is serially polled by bus controller (this pattern is also called the Serial Poll Byte).

msb							lsb
B7	B6	B5	B4	B3	B2	B1	B0
S8	rsv	S6	S5	S4	S3	S2	S1

When the rsv bit is 1, a service request is issued on the GPIB. The value of the others bits may be set by the user for whatever purpose is desired.

COMMAND	*USAGE*
SYScon	PRINT #1, "SYSCON MAD1=12 CIC1=1 BA1=&H300 CLK=8"

FUNCTION

Used to initialize the (uC)MBC-488 board(s) and must be executed in the user's program prior to other GPIB command issuances. ONE or TWO boards may be initialized with this command. SYSCON requires three parameters (Board #2 parameters are used only if there is a second board).

"MAD1 <MAD2>" GPIB Module address. Allows two boards to have independent GPIB addresses, if desired.

"CIC1 <CIC2>" Controller In Charge. There are four possible levels as follows:

"CICx=0" Addressed Talker/Listener. In this mode, the user's board is not in control and must wait for the GPIB controller to address it as a Talker or Listener. The address used is the one programmed by the MADx parameter. The REQUEST command is used to determine the addressed state by testing the LA & TA bits. The user's program may switch between the LA and TA modes at will.

"CICx=1" Controller In Charge. This command gives complete control of the GPIB to the (uC)MBC-488.

"CICx=2" Non-addressed Talker/Listener. Used to communicate with devices such as printers, plotters, and other Talk Only/Listen Only devices not supporting GPIB addressing. If the OUTPUT command is issued the driver will implement the TALK function and broadcast on the GPIB disrupting any other GPIB traffic. If the ENTER command is issued the LISTEN function is forced and the driver will accumulate data from the bus. Note that the user's program may not arbitrarily switch between a TALKER and LISTENER mode without reconfiguring the bus via the SYSCON command.

"BA1 <BA2>" Base Address. This command sets the (uC)MBC-488 base address in the computer's I/O address space. It is left to the user to determine that the address assigned is a valid address that does not interfere with other computer processes/devices. If two boards are present, the driver only checks that they are 16-bytes apart. A test for the board at the given address is also done.

"CLK=c" PC/XT/AT Bus Clock Speed. This command is specifically for the MBC488 boards without on-board crystal clocks. It is important that the correct bus clock speed is passed to the driver since it is this speed that determines GPIB timing parameters. If no value is specified, 4 MHz is assumed. Valid values for <c> are 4, 8, 12, 16, 20, 24, 28, 32. Non-valid values will cause an error. This parameter is ignored by the PS/2.

COMMAND	*USAGE*
STatus	PRINT #1, "STATUS 7" INPUT #2, STAT%

FUNCTION

The STATUS command performs a serial poll. The contents of the serial poll register for the specified device is extracted and returned as a long signed integer.

COMMAND	*USAGE*
TIMEOUT	A% = 100 PRINT #1, "TIMEOUT", A%

FUNCTION

Used to set one time-out period for GPIB command and data transfer activity. An integer multiplier is specified which sets the timeout period to a multiple of 56 milliseconds. If the TIMEOUT command is never executed, the default timeout period is about 2 seconds. No devices are specified.

COMMAND	*USAGE*
TRigger	PRINT #1, "TRIGGER 7 12"

FUNCTION

Sends the GPIB "GET" message (Group Executive Trigger). The device(s) must have been programmed to perform a specific task on GET or the message is ignored. One or more devices must be specified.

DV488 ERROR MESSAGES

All errors, whether BASICA or DV488, are handled by DOS. When a device driver error occurs, the board sends three messages to the computer as mentioned above. These errors and messages are listed below. Error numbers not listed are reserved.

ERROR #	MESSAGE	CAUSE
1	Unknown character encountered	Misspelled DV488 command
2	Space expected ...	Improperly spaced command elements.
3	DMA page wraparound	An array address or byte count was passed for DMA transfer that results in crossing a page boundary. (See Section 5).
4	DMA level error	Incorrect or disconnected DMA level.
6	DMA channel busy	Bus request issued while DMA transfer in progress.
8	Hardware Failure	Error during internal procedure call or the IBC chip encountered an error on a write.
9	Device time out	If GPIB transfer failure occurred, driver retries transfer for time out period.
10	GPIB BOARD not in control	Issued command requires (uC)MBC-488 control.
11	GPIB BOARD is in control	Issued command requires (uC)MBC-488 NOT in control.
12	System not initialized	SYSCON not issued prior to GPIB command.
13	Configuration Error	One or more SYSCON parameters invalid. In 2 board systems, Base Addresses are not 16 bytes apart or the interrupt & DMA levels are the same (IBM PC/XT) or the DMA levels are the same (PS/2). Board not found at specified Address.

14	Undefined command	Invalid or missing DV488 command.
15	Syntax error in command line	Most often, missing spaces between command parameters, misspelled commands or incomplete (missing) SYSCON parameters.
16	Undefined image	Missing [$, B, RB, BL, BH, or DMA]
17	Device range error	Invalid GPIB address. Must be between 0 - 30.
18	TOO many devices	Specified command contained too many device addresses.
20	Command / Data out of range	Byte count passed must be between 1 - 65535.
21	Command requires device	No address(es) passed with GPIB command requiring it.
24	GPIB BOARD must be talker or Listener	Normally occurs when CICx = 0 and the IBC chip has not been addressed. Use REQUEST command to check address status in TA and LA bits.

Programmed Data Transfer

In a normal or Programmed transfer, the data travelling to or from the 7210 controller passes through the accumulator register (AX) of the computer's CPU. This allows flexibility in data handling but also involves a good deal of CPU overhead (the CPU continually polls the 7210 for new data which must be moved to/from the AX register and then move the AX register to/from memory). This takes time and is dependent upon CPU speed. Even with this overhead, transfer rates in programmed mode are usually around 1000-2000 bytes/second (also dependent upon GPIB device response speed). Speed is usually not a major factor during instrument control where command strings are generally short and response times long. But it may become a more serious problem with devices that move a lot of data in blocks over the GPIB.

BASICA Example Program

The following is an example program, with comments, illustrating an MBC-488 board and a typical device (Digital Multimeter with GPIB capability). It is written in BASICA as implemented on the IBM PC/XT. It uses Serial Polling to ensure data validity. The program itself (DVSPOL.BAS) is supplied on the diskette that came with your board, but this example is exhaustively commented for new GPIB programmers.

```
100 ' IEEE488 Device Driver Example Program
110 ' BASICA
120 ' (DVSPOL.BAS)
130 ' Serial Poll with STRING data Transfer
140 ' MetraByte Corp  1/17/89
150 '
160 '
170   CLS:KEYOFF:LOCATE 25,1:PRINT" Press any key to exit program";
180   LOCATE 1,1
190 '
200 ' This is an example of reading data from device 12, in this case
210 ' a Keithly 196 system DMM.
220 ' To run with a different device number, change command strings in
230 ' those lines referring to DEVICE 12.
240 '
250 ' #### Establish Communication with device driver ####
260 '
270   OPEN "$DV488" FOR OUTPUT AS #1
280   PRINT #1, "BUFFERCLEAR"
290   OPEN "$DV488" FOR INPUT AS #2
```

Line 370 opens an output file (#1) for data transmission via $DV488. Line 380 clears the output file (buffer) of extraneous data, if any. Line 390 opens an input file (#2) for receiving data via $DV488.

```
300   ON ERROR GOTO 1040
```

This is a simple error trapping routine for both BASICA and DV488 errors that may occur during execution of the program.

```
310 '
320 ' #### Initialize MBC-488 board using SYSCON command.
330 '
340   PRINT #1, "SYSCON MAD1=3 CIC1=1 BA1=&H300"
```

MAD1=3 assigns board #1 (MBC-488) a GPIB device address of 3. CIC1=1 assigns board #1 as controller in charge. BA1=&H300 assigns board #1 a Base Address of 300 (hex). Also transmits the GPIB message IFC (Interface Clear).

```
350 '
360 ' ##### Set GPIB device (DVM) into REMOTE state. ####
370 '
380   PRINT #1, "REMOTE 12"
```

Set GPIB device at address #12 to remote state for computer control.

```
390 '
400 ' #### Set Time-out period (timeout = 0.056 x A%) ####
410 '
420   A% = 100
430   PRINT #1, "TIMEOUT", A%
440 '
450 ' #### Set Mode to TRIGGER on GET ####
460 '
470   CMD$ = "T3F0R3S0M8X"
480   PRINT #1, "OUTPUT 12 $ +" ,CMD$
```

The CMD$ is a DVM specific command for selecting various functions and ranges on the DVM. Line 570 outputs this STRING (CMD$) command to device # 12 and appends a <cr>, <lf>, and asserts the EOI line with the <lf>. Certain devices require a long time initial setup (5 seconds). A time delay loop such as the following is common practice to avoid a bus lock-up.

```
490 '
500 LAP = 50'                5 second delay for instr to process config data
510 INCREMENT = 0.1
520 T0 = TIMER : D$=DATE$
530 T1 = T0
540 WHILE LAP <> 0
550         WHILE (T1 < (T0 + INCREMENT))
560         T1 = TIMER : ID D$<>DATE$ THEN T1=T1+86400!
570         WEND
580 D$=DATE$:T0=T0+INCREMENT:IF T0>=86400! THEN T0=T0-86400!:
T1=T1-86400!
590 LAP = LAP -1
600 WEND
610 '
620 #### Send TRIGGER message ####
630 '
640 PRINT #1, "TRIGGER 12"
```

Trigger Device #12 (DMM) for data measurements.

```
650 '
660 '#### Wait for SRQ (Service Request) ####
670 '
680 PRINT #1, "REQUEST"
690 INPUT #2, REQ%
700 PRINT "Waiting for Service Request"
710 PRINT "REQUEST word = &H";HEX$(SPOLL%):PRINT
720 IF (REQ% AND &H4000) <> &H4000 THEN 680
```

This is a status request that will return data in a long, signed integer format. The information concerns various registers of the 7210 (IBC chip) as well as the MBC-488 registers. No GPIB activity is initiated since the required data comes only from the computer and board. The bit of interest is the SRQI bit in the 7210 Interrupt Status Register 1. This bit is true when the GPIB SRQ line is asserted. The SRQ line is a "WIRE OR" line meaning that one or more devices on the GPIB is requesting service. This type of routine may be used as a "check for GPIB activity" subroutine and may be entered and exited periodically to check for bus activity (data transfers). However, the driver does not support interrupt handling, as such, in this condition.

```
730 '
740 ' #### Read Serial Poll Byte ####
750 '
760   PRINT #1, "STATUS 12"
770   INPUT #2, SPOLL%
780   PRINT "Testing Serial Poll Byte for 'RSV'"
790   PRINT "Serial Poll Byte = &H";HEX$(SPOLL%):PRINT
800   IF (SPOLL% AND 8) <> 8 THEN PRINT "No request in Serial Poll Byte":STOP
```

The Serial Poll (STATUS 12) is used to find out if device # 12 is the device requested service since the SRQ line only indicates that one or more devices require service. Data is returned in a long, signed integer. The contents of the DMM. serial poll register are read back to the computer (via STRING variable SPOLL$) where it is evaluated for activity (set high if true). The 8 is was used previously during the DMM. setup (M8) enabling the DMM. to issue an SRQ under certain conditions.

```
810 '
820 ' #### Read data from device 12 (DMM.) ####
830 '
840   PRINT #1, "ENTER 12$"
850   INPUT #2, DVM$
860   PRINT "DVM Data = ";
870   PRINT DVM$
880   PRINT:PRINT:PRINT
```

Line 810 instructs device 12 to transmit (TALK) STRING data.
Line 820 receives this data into the STRING variable DVM$.
Line 830 displays the received data on the computer display.

```
890   LAP = 20                    '2 second delay for viewing data
900   INCREMENT = 0.1
910   T0 = TIMER : D$=DATE$
920   T1 = T0
930   WHILE LAP <> 0
940         WHILE (T1 < (T0 + INCREMENT))
950         T1 = TIMER : ID D$<>DATE$ THEN T1=T1+86400!
960         WEND
```

```
  970   D$=DATE$:T0=T0+INCREMENT:IF T0>=86400! THEN T0=T0-86400!:
T1=T1-86400!
  980   LAP = LAP - 1
  990   WEND
  1000  '
  1010  KX$ = INKEY$:IF K$ = "" THEN 640
  1020  CLOSE                                   'close all files for I/O
  1030  STOP                                    'halt program
  1040  IF ERR <> 68 AND (ERR<> 57) THEN PRINT "BASIC ERROR # ";ERR;" IN LINE
";ERL:STOP
  1050  INPUT #2, E$
  1060  PRINT "$DV488 driver returned error number - ", E$
  1070  INPUT #2, E$
  1080  PRINT E$
  1090  INPUT #2, E$
  1100  PRINT E$
  1110  END
```

Lines 890 thru 960 are simply checking for any error and returning these errors whether BASIC errors or DV488 errors to STRING variable E$ for display. Enhancing this routine could be done by branching to various other routines for certain errors and correcting them.

Error #57 is a BASIC "Device I/O Error"

Error #68 is a BASIC "device unavailable" error indicating that $DV488 returned the error. If this is the case, three strings are issued from the driver as follows:

1) Device Driver Error Number
2) Original Command String
3) Position (in command) where error occurred and plain english error message.

NOTE: BASICA does not support long integers (32 bits), but the $DV488 driver returns a long integer with only the low byte or word containing information. In all cases, the upper byte or word is the extended sign. However, BASICA will correctly read the long signed integer returned into a BASIC integer.

DMA Command Syntax

```
[BRD2] ENTER DEV DMA[term] SEGMENT OFFSET COUNT
[BRD2] OUTPUT DEV DMA[term] SEGMENT OFFSET COUNT
```

Example:

```
DMASEG% = &H8000 : DEFSEG = DMASEG%
PRINT #1, "ENTER 12 DMA +", DMASEG%, "0", "50"
```

Determining the Status of a DMA Operation

Once a DMA transfer has started, it will remain active until the terminal count is reached or a valid EOI is received. DMA is really the hardware equivalent of an interrupt and is a background operation. As such, your program may continue run in the foreground. Since this is true, your program should be written to check DMA status. The status of a DMA operation can be determined by the REQUEST command (see Commands). REQUEST returns a 16 bit integer, bit 9 is set when the DMA is active and cleared when done. If required, a simple polling loop (check bit 9) can be used to hold program execution until the DMA transfer is complete. If any errors occur during the DMA last byte transfer, bit 8 of the REQUEST 1 return integer will be set.

QuickBASIC Example:

```
PRINT #1, "ENTER 12 DMA +", VARSEG(DMA%(0)), VARSEG(DMA%(0)), "50"
```

Transferring to a STRING$:

```
S$ = SPACE$(1024)     'allocate memory for 1024 characters
' calculate address offset of S$ in memory
offs% = (PEEK(VARPTR(S$) + 2) + (256 * (PEEK(VARPTR(S$) + 3))))
PRINT #1, "ENTER 12 DMA +", VARSEG(S$), offs%, LEN(S$)
```

QuickBASIC Example Program

The following QuickBASIC example illustrates the use of DMA data transfer to/from a specific memory address. It is also supplied on the diskette that was supplied with your uCMBC-488 under the file name QBDVDMA.BAS.

```
' ============================================
'QuickBASIC 4.0
'(QBDV2DMA.BAS)
'Serial Poll with DMA transfer on TWO Boards
'MetraByte Corporation
'============================================
'
'This is an example of reading data from device 12, in this case
'a Keithly 196 system DMM.
'To run with a different device number, change command strings in
'lines referring to DEVICE 12.
'
DECLARE SUB ADDRESS (s%, o%, count%, addr%, byteOff%)
'
CLS:KEY OFF:LOCATE 25,1: PRINT "Press any key to exit program";
LOCATE 1,1

'BOARD #1 array allocation
'Calculate address within DV%() that lies on a page boundary and
'return byte offset to that element

DIM dv%(200)
count% = LEN(dv%(0)) * 25                    'allow 50 bytes for transfer
CALL ADDRESS(VARSEG(dv%(0)), VARPTR(dv%(0)), count%, addr%, byteOff%)

'calculate true beginning element for this array type
element% = byteOff% / LEN(dv%(0))

BOARD #2 array allocation
'Calculate address within DV2%() that lies on a page boundary and
'return byte offset to that element

DIM dv2%(200)
count% = LEN(dv2%(0)) * 25                   'allow 50 bytes for transfer
CALL ADDRESS(VARSEG(dv2%(0)), VARPTR(dv2%(0)), count%, addr2%, byteOff%)

'calculate true beginning element for this array type
element2% = byteOff% / LEN(dv2%(0))

'#### Establish communication with device driver ####

OPEN "$DV488" FOR OUTPUT AS #1
PRINT #1, "BUFFERCLEAR"
OPEN "$DV488" FOR INPUT AS #2
ON ERROR GOTO ersvc
```

```
'Initialize uCMBC-488 board using "SYSCON"command
'
PRINT #1, "SYSCON MAD1=3 CIC1=1 BA1=&H300 MAD2=3 CIC2=1 BA2=&H400"

'#### Set DMM into REMOTE #######

PRINT #1, "REMOTE 12"
PRINT #1, "BRD2 REMOTE 12"

'######### SET TIMEOUT (timeout time = 0.056 x A%)
A%=100
PRINT #1, "TIMEOUT", A%

'######### SET MODE TO TRIGGER ON "GET"
'
DV$ = "T3R3M8X"

PRINT #1, "OUTPUT 12 $ +", dv$
PRINT #1, "BRD2 OUTPUT 12 $ +" dv$

LAP = 50                                        'wait 5 Seconds for set-up finish
T0 = TIMER: D$=DATE$
T1 = T0
WHILE LAP <> 0
  WHILE (T1< (T0 + INCREMENT))
  T1 = TIMER : IF D$ <> DATE$ THEN T1=T1 + 86400
  WEND
D$=DATE$ : T0=T0 + INCREMENT : IF T0>=86400 THEN T0=T0-86400:T1=T1-86400
LAP=LAP-1
WEND
'######## Send trigger message ########
'
KX$ = ""
WHILE KX$ = ""
PRINT #1, "TRIGGER 12"
'
'####### Wait for SQR #######
'
REQ% = 0
REQ2% = 0
'
  WHILE (REQ% AND &H4000) <> &H4000
    PRINT #1, "REQUEST"
    INPUT #2, REQ%
    PRINT "Waiting for Service Request from DVM on board #1"
    PRINT "REQUEST WORD = &H"; HEX$(REQ%):PRINT
  WEND
```

```
'######## Read Serial Poll byte ########
'
PRINT #1, "STATUS 12"
INPUT #2, SPOLL%
PRINT "Testing Serial Poll byte for 'RSV'"
PRINT "Serial Poll Byte = &H"; HEX$(SPOLL%): PRINT
IF (SPOLL% AND 8) <> 8 THEN PRINT "No request in Serial Poll Byte": STOP
'
  WHILE (REQ2% AND &H4000) <> &H4000
    PRINT #1, "BRD2 REQUEST"
    INPUT #2, REQ2%
    PRINT "Waiting for Service Request from DVM on board #2"
    PRINT "REQUEST word = &H"; HEX$(REQ2%):PRINT
  WEND

'######## Read Serial Poll byte ########
'
PRINT #1, "BRD2 STATUS 12"
INPUT #2, SPOLL2%
PRINT "Testing Serial Poll byte for 'RSV'"
PRINT "Serial Poll Byte = &H"; HEX$(SPOLL%): PRINT
IF (SPOLL2% AND 8) <> 8 THEN PRINT "No request in Serial Poll Byte #2": STOP
'
'############ Read data from DVM ############
'
PRINT #1, "ENTER 12 DMA", addr%, "0", 16        ' start DMA on board #1
PRINT #1, "BRD2 ENTER 12 DMA", addr2%, "0", 16   ' board #2
'
'######### Wait for DMA to finish #########
'
REQ% = &H200
'
  WHILE  (REQ% AND &H200) <> &H200
    PRINT #1, "REQUEST"
    INPUT #2, REQ%
    PRINT "Waiting for DMA ENTER to finish on board #1"
    PRINT "REQUEST word = &H"; HEX$(REQ%):PRINT
  WEND

PRINT "Data form DVM on board #1 = ";
FOR offs = element% to element%+14                'recover stored DMA data
'print high byte then low byte of each integer

PRINT CHR$(dv%(offs)) AND &HFF); CHR$(dv%(offs) / 256);
NEXT offs
PRINT:PRINT:PRINT
```

```
LAP = 20                                        '2 Second dealy for viewing
INCREMENT = 0.1
T0 = TIMER:D$ = DATE$
T1 = T0
WHILE LAP <> 0
  WHILE (T1 < T0+INCREMENT))
  T1 = TIMER : IF D$ <> DATE$ THEN T1 = T1 + 86400
  WEND
D$=DATE$:T0=T0+INCREMENT: IF T0>=86400 THEN T0=T0-86400:T1=T1-86400
LAP = LAP -1
WEND
PRINT
'
REQ2% = &H200
'
  WHILE  (REQ2% AND &H200) <> &H200
    PRINT #1, "BRD2 REQUEST"
    INPUT #2, REQ2%
    PRINT "Waiting for DMA ENTER to finish on board #2"
    PRINT "REQUEST word = &H";HEX$(REQ2%):PRINT
  WEND
'
PRINT "Data form DVM on board #2 = ";
FOR offs = element2% to element2%+14            'recover stored DMA data
'print high byte then low byte of each integer

PRINT CHR$(dv2%(offs)) AND &HFF); CHR$(dv2%(offs) / 256);
NEXT offs
PRINT:PRINT:PRINT

LAP = 20                                        '2 Second dealy for viewing
INCREMENT = 0.1
T0 = TIMER:D$ = DATE$
T1 = T0
WHILE LAP <> 0
  WHILE (T1 < T0+INCREMENT))
  T1 = TIMER : IF D$ <> DATE$ THEN T1 = T1 + 86400
  WEND
D$=DATE$:T0=T0+INCREMENT: IF T0>=86400 THEN T0=T0-86400:T1=T1-86400
LAP = LAP -1
WEND
PRINT

KX$ = INKEY$
'
WEND
CLOSE
STOP
```

```
  ersvc:
  '
  IF (ERR<>68) AND (ERR<>57) THEN PRINT "BASIC ERR # ";ERR;" IN LINE ";
ERL:STOP
  INPUT #2, E$
  PRINT "$DV488 driver returned error number - "; E$
  INPUT #2, E$
  PRINT E$
  INPUT #2, E$
  PRINT E$
  END

  SUB ADDRESS (s%, o%, count%, addr%, byteOff%)
  '
  '         Subroutine to Allocate Space for DMA Transfers ...
  '
  DIM a AS LONG
  DIM b AS LONG

    s = s%                                          'get segment address
    o = o%                                          'get segment offset
    IF o<0 THEN o = o + 65536
    IF s<0 THEN s = s + 65536
    a = s * 16 + o
    arrayptr = a                                    'save org ptr for element calc
    page = INT (a / 65536)

    REM Check for DMA wraparound

  b = a - (page * 65536)
  b = b + count%
  IF b > 65535 THEN
    a = a (page = 1) * &H1000:                      'here if page wrap would occur
    IF a > 32767 THEN addr% = a - 65536 ELSE addr% = a

       ELSE REM here if no page wrap
           a = a / 16
           IF a >32767 then addr% = a - 65536
           IF a <= 3277 THEN addr% = a
       ENDIF
       IF addr% < 0 THEN  c = addr% + 65536 ELSE c = addr%
       c = c * 16                                   ' new ptr to actual xfer element
  'return byte offset where DMA will transfer in the named array

  bytOff% = (c - arrayptr)
  'note the byteOff% value must be divide by the length of the array type

  END SUB
```

5.0 IBM PC/XT SPECIFIC PROGRAMMING STRUCTURE

DMA Data Transfer

The NEC 7210 controller on the MBC-488 is capable of performing transfers of data directly from/to the GPIB to/from memory using direct memory access (DMA). In DMA mode, the 7210 generates a DMA request to the 8237 DMA controller on the IBM PC system board. In turn, the DMA controller issues a hold request to the 8088 CPU which releases the internal PC bus as soon as it completes its current instruction. The 8088 issues a hold acknowledge to the DMA controller which then takes control of the bus, placing a valid memory address on the address bus and issuing either a simultaneous memory read and I/O write (or memory write and I/O read) together with a DMA acknowledge to the peripheral requesting transfer (MBC-488). This enables transfer of data directly to/from the 7210 controller from/to memory. Depending upon the interrupt level chosen, the data transfer process will continue until all data has been transferred, as signaled by a valid EOI or the terminal count is reached. If, however, a higher priority interrupt level requests service while (DMA) data is being transfered, the transfer is put on hold until the higher priority request is serviced. Whereupon, data transfer resumes. For this reason, the interrupt level should be chosen carefully so that it is assigned a priority level corresponding to its overall system importance.

The installation of the MBC-488 DMA interrupt handler on IBM PC/XT class machines is accomplished only once but no chaining mechanism is available (unlike the PS/2). The user is cautioned not to use the DMA interrupt level for any other purpose (since a interrupt level corresponds to a vector number which, in turn, points to an address in memory where the handler resides). If the same level is used for another purpose, the DMA interrupt handler should be reinstalled prior to MBC-488 requests for DMA data transfer.

The ultimate speed at which the DMA controller can handle DMA requests on a standard 4.77MHz IBM PC is about 400,000 bytes/second (it is faster still on an IBM PC/AT). In practice, the actual transfer rate will depend upon the GPIB device(s) response speed. DMA transfers are less flexible in passing data directly to variables and in general require more setup than normal (programmed) transfers. However when speed is paramount, DMA is the best way to use the controller.

IBM PC/XT Memory Page Boundary

The DMA controller is unable to transfer data across page boundaries. DV488 checks for page boundary overlap. It is left to the user to ensure that memory has been allocated correctly for data during DMA transfers.

There are sixteen legal pages within IBM PC memory so that it is unlikely that any array, unless very large, will overlap a memory page. The general procedure for determining the segment boundary for a legal page is shown in the QuickBASIC example below. The array (addr%), which is twice the required size, the transfer count (count%), the segment in s%, and the offset in o% are passed to the procedure. The legal segment is returned in ADDR% as well as the byte count to the adjusted address from the element 0 in byteoff%. This integer variable could then be used as a data pointer in the OUTPUT or ENTER commands as follows:

```
PRINT #1, "OUTPUT 12 DMA +", addr%, "0", count%
```

```
SUB ADDRESS (s%, o%, addr%, byteOff%)
'       Subroutine to allocate space for DMA transfer ...
'
DIM a AS LONG
DIM b AS LONG
        s = s%                                  'get Segment Address
        o = o%                                  'get Segment Offset
IF o < 0 THEN o = o + 65536
IF s < 0 THEN s = s + 65536
a = s * 16 + o
arrayptr = a                'save orig ptr for element calc

page = INT(a / 65536)

REM Check for DMA wraparound
b = a - (page * 65536)
b = b + count%

IF b > 65536 THEN
        a = (page + 1) * &H1000:  REM here if Page wraparound occurs
        IF a > 32767 THEN addr% = a - 65536 ELSE addr% = a

ELSE REM here if no page wrap
        a = a / 16
        IF a > 32767 THEN addr% = a - 65536
        IF a <= 32767 THEN addr% = a
ENDIF
IF addr% < 0 THEN c = addr% + 65536 ELSE c = addr%
c = c * 16                  ' new ptr to actual transfer element

' return byte offset where DMA will transfer in the named array
byteOff% = (c - arrayptr)
' Note the byteOff% value must be divided by length of array type

END SUB
```

DMA Transfers in BASIC

The MBC-488 DMA data transfer hardware allows the user to transfer data anywhere within the IBM PC's 20 bit address range with a word count of up to 32767 (64 Kbyte).

IF THE SEGMENT IS SET TO &HFFFF THEN BASIC's DATA SEGMENT IS USED. This allows the user to transfer data directly to a BASIC variable.

This is the value used as a reference starting point. If you choose to transfer data directly into a BASIC string or array variable, they must be declared prior to VARPTR declaration. Once DMA data transfer has begun, NO simple variables may be declared. Failure to comply with this requirement may generate unpredictable errors. The reason for this is that BASIC stores array variables above simple variables in memory, and on declaring a new simple variable, BASIC relocates all array variables and string descriptors. If DMA data transfer to a string or array is in progress, then suddenly relocated, disaster ensues!

STRING array space must be pre-allocated by assigning some data to the string e.g. A$=SPACE$(80) prior to using the ENTER command. Do not try to move data into a null string.

NOTE DMA channel 1 shares the same page register as DMA channel 0 (memory refresh). Altering the page register normally causes no problems except on early IBM PC models with 64K memory on the system board (using 16K chips). On these machines a parity check error will occur (due to a design fault in the hardware) and the computer will have to be re-booted. This problem can be avoided by disabling the memory parity check before commencing the DMA operation:

```
OUT &HA0,0    'Disables NMI & parity check
OUT &HA0,&H80 'Re-enables NMI & parity check
```

DMA channel 3 is used by and cannot be shared with the hard disk in XT computers. In floppy only machines it is unused and available. In the PC/AT both levels 1 and 3 are available despite the presence of the hard disk. The availability of a suitable DMA level is a system limitation of the IBM-PC. It is the user's responsibility to insure that the MBC-488 DMA level is correctly selected on the DIP switch and will not conflict with the use of the DMA level by another peripheral (usually a hard disk controller or LAN card)

DMA data transfers are aborted with the ABORT command.

No check is made on overrunning the limits of the array. It is your responsibilty to avoid this condition

6.0 PS/2 SPECIFIC PROGRAMMING STRUCTURE

DMA Data Transfer

The NEC 7210 controller used on the uCMBC-488 is capable of performing data transfers directly from/to the GPIB and to/from memory using direct memory access (DMA). In DMA mode, the 7210 generates a preempt to the DMA controller on the PS/2 system board. The PS/2's DMA controller is similar to the Intel 8237 Programmable DMA Controller used on the IBM PC/XT/AT. The major difference is that the PS/2's DMA controller supports the MicroChannel Architecture and employs serial data transfer whereas the 8237 (used in the PC) uses a parallel type DMA interface for data transfer. All this is, however, transparent to the user (since DV488.sys takes care of it) so that you need not concern yourself with the details. DMA enables transfer of data directly to/from the controller from/to memory at very high rates. The ultimate speed at which the DMA controller can handle data is about 400,000 bytes/second. In practice, the actual data transfer rate will depend upon the speed of response of the device(s) on the GPIB that is involved in the DMA transfer. DMA transfers are less flexible in passing data directly to variables and, in general, require more setup than normal transfers, however when speed is paramount, they are the best way to use the controller.

DMA Data Transfers in BASIC

The uCMBC-488 DMA data transfer hardware allows the user to transfer data anywhere within the PS/2's 20 bit address space with a word count of up to 32767 (64 Kbyte).

IF THE SEGMENT IS SET TO &HFFFF, BASIC's DATA SEGMENT IS USED.
This allows the user to transfer data directly to a BASIC variable. If VAR%(0) is set equal to &HFFFF, then BASIC's data segment will automatically be used. This is useful when directly transferring to/from BASIC variables.

All simple variables must be declared prior to the VARPTR statement and cannot be declared after memory has been set aside for DMA transfer. Failure to comply with this requirement may generate unpredictable errors. The reason for this is that BASIC stores array variables above simple variables in memory, and on declaring a new simple variable, BASIC relocates all array variables and string descriptors. If DMA data transfer to a string or array is in progress, sudden reallocation of memory may be undesirable. Also, on an ENTER command the string space must be pre-allocated by assigning some data to the string, e.g. Do not try to move data into a null string.

Data transfers may be made to/from memory outside BASIC's workspace or directly to an integer array variable or string variable within BASIC. This allows you to store block data up to the available limits of RAM.

NOTE: The TIMEOUT command has no effect in DMA mode. A DMA transfer will remain active until either Terminal COUNT has been reached or a valid EOI terminator has been recognized. To abort a DMA transfer in progress use the ABORT command.

7.0 MBC-488 HARDWARE STRUCTURE

MBC-488 Register I/O Map

The MBC-488 uses 16 consecutive I/O addresses. The first 8 addresses are used by the NEC 7210 controller and the following I/O address drives registers on the MBC-488 that control generation of DMA requests and interrupts. The I/O address map is as follows:

ADDRESS		READ	WRITE
Base	+ 0	Data byte in	Data byte out
	+ 1	Interrupt Status 1	Interrupt Mask 1
	+ 2	Interrupt Status 2	Interrupt Mask 2
	+ 3	Serial Poll Status	Serial Poll Mode
	+ 4	Address Status	Address Mode
	+ 5	Command Pass Through	Auxiliary Mode
	+ 6	Address 0	Address 0/1
	+ 7	Address 1	End of String
	+8-11 *	Interrupt/DMA status	Interrupt/DMA control
	+12-15	Not used	Not used

* See next section (MBC-488 Interrupts & DMA) for explanation.

The Base Address is set by the DIP switch on the MBC-488 board and may be on any 16 bit boundary in I/O address space. The NEC 7210 controller chip register functions are explained in the NEC 7210 data sheet. A more useful and expanded description of the controller chip functions is in a 90 page application manual titled "uPD7210 General Purpose Interface Bus Manual" available from any NEC Electronics Inc. sales office. This information is not required by the high level programmer using the MBC-488 driver, but would be of value to the assembly language programmer interested in writing his own drivers. Any register may be accessed by the appropriate OUT DX, AL or IN AL, DX instruction (equivalent to Basic's OUT & INP()).

MBC-488 Interrupts & DMA

The MBC-488 can generate an interrupt on any of the available IBM PC bus levels 2 - 7 as selected on the interrupt level DIP switch. The interrupt is the logical "OR" of the interrupt from the 7210 controller and the 8237 DMA controller Terminal Count (T/C). The standard driver, DV488.SYS, makes use of interrupts and DMA requests only in DMA mode, the interrupt outputs and DMA REQUEST are tristated (high impedance) in normal program mode. This means several MBC-488 or other peripherals can share DMA/interrupt levels as long as they use them in turn and tristate their outputs when disabled. It is possible to program the controller to generate an interrupt on up to 13 different conditions including the obvious ones of a GPIB service request (SRQ) or an EOI or a specific end of string character (EOS). The hardware will support propagation of this interrupt to the IBM PC/XT, but it is the user's responsibility to provide a suitable interrupt service routine to handle the interrupt. Since BASIC and most other high level languages do not support interrupts (there is no ON INTERRUPT statement in Basic for instance) there is no support for interrupts in the driver (DV488.SYS), however an assembly language programmer can take advantage of the hardware's ability to generate interrupts by writing their own interrupt service handler.

Reading Base Address + 8 (or 9, 10 or 11) returns the following data:

ADDRESS	D7	D6	D5	D4	D3	D2	D1	D0
Base + 8	DMA	x	Interrupt Level			DMA LVL	DMA ENA	INT ENA

Bit D7 is set by a double write to the interrupt/DMA control register (D1, below) with DMA enable = 1 and is cleared by T/C from the 8237 DMA controller.

Bits D5 thru D3 are derived from the board DIP switches:

010	Level 2
011	Level 3
100	Level 4
101	Level 5
110	Level 6
111	Level 7

Bit D2 is derived from a DIP switch and indicates DMA level 1 (1) or level 3 (0).

Bits D1 and D0 correspond to bits in the interrupt/DMA control register. (1) enabled (0) disabled.

Writing to Base Address + 8 (or 9, 10 or 11) enables interrupts and/or DMA as follows:

ADDRESS	D7	D6	D5	D4	D3	D2	D1	D0
Base + 8	x	x	x	x	x	x	DMA	INT

Bits D1 and D0 indicate enable (1) or disable (0)

MBC-488 Specifications

Electrical

+5v power	470mA typical / 600mA , max
-5v, +12v, -12v power	Not used
Controller chip	NEC uPD7210
Bus drivers	SN75160AN & SN75162AN or SN75160BN or SN75162BN (these are socketed for easy replacement)
I/O address	Can be set on any 16 bit boundary from Hex 100 to Hex 3F0
PC bus loading	1 74LS TTL load on all inputs
Interrupts	Can be set on any level 2-7 (active only during DMA transfer)
DMA level	Selectable level 1 or 3 (active only during DMA transfers)
Operation in J8 slot	Possible in all non-DMA modes of PC/XT (J8 slot is half slot near power supply, usually used for expansion interface)

Interface Repertoire

T6	Basic talker, serial poll, unaddressed if MLA
TE0	No extended talker function
L4	Basic listener, unaddressed if MTA
LE0	No extended listener function
SH1	Complete source handshake capability
AH1	Complete acceptor handshake capability
SR1	Complete service request capability
PP1	Parallel poll remote configuration capability
RL1	Complete remote/local capability
DC1	Complete device clear capability
DT1	Complete device trigger capability
C1,C2,C3,C4,C5	Controller states

Bus Performance

Data transfer rate	2 Kbyte/sec (normal program mode) 450 Kbyte/sec (DMA) These rates assume zero response time from addressed devices. In practice, transfer speeds may be lower, especially in DMA.
Devices on GPIB	15 (max, including MBC-488)
Secondaries	2 (max, addressed from same driver)
Bus length	20 meters (max). Not to exceed 2 meters per device. 5 devices - 10 meters 12 devices - 20 meters
Connector	On rear plate 24 pin female micro-ribbon (Std. IEEE-488) with metric hardware.

Mechanical & Environmental

Card size	5" long x 3.9" high 1/2 slot size
Weight	4.25 oz. (0.125 Kg)
Operating Temp	0 to 50 deg. C.
Storage Temp	-40 to 100 deg. C.
Humidity	0 - 90% non-condensing

8.0 uCMBC-488 HARDWARE STRUCTURE

uCMBC-488 I/O Register Map

The uCMBC-488 has 16 I/O registers These 16 registers are accessed by reading and writing 16 consecutive I/O addresses (starting at the uCMBC-488's Base Address) in the PS/2's memory workspace. The first 8 addresses are used by the NEC 7210 controller while the last 8 are used to control interrupt requests and DMA.

ADDRESS		READ	WRITE
Base	+ 0	Data byte in	Data byte out
	+ 1	Interrupt Status 1	Interrupt Mask 1
	+ 2	Interrupt Status 2	Interrupt Mask 2
	+ 3	Serial Poll Status	Serial Poll Mode
	+ 4	Address Status	Address Mode
	+ 5	Command Pass Through	Auxiliary Mode
	+ 6	Address 0	Address 0/1
	+ 7	Address 1	End of String
	+8-11 *	Interrupt/DMA status	Interrupt/DMA control
	+12-15	Not used	Not used

* See next section for STATUS/CONTROL map.

STATUS/CONTROL Registers (Base Addr 8 - 11)

Hardware Register 8 RETURN STATUS

B7	B6	B5	B4	B3	B2	B1	B0
DMA ACTIVE	EOT INTR PEND	x	x	ARB3	ARB2	ARB1	ARB0

Hardware Register 8 WRITE CONTROL

B7	B6	B5	B4	B3	B2	B1	B0
DMA CONTROL	x	x	x	x	x	x	x

Register 8 Bit Description:

DMA ACTIVE 1 = ENABLE 0 = DISABLE
EOT INT PEND 1 = DMA TC REACHED REQUESTING INTERRUPT
ARB 0 - 3 I/O READBACK OF POS ARBITRATION LEVEL
DMA CTL SET = 1 TO ENABLE DMA

Hardware Register 9 RETURN STATUS							
B7	B6	B5	B4	B3	B2	B1	B0
IL3	IL2	IL1	IL0	IRQ PEND	7210 INT PEND	7210 INT ACTV	EOT INT ACTV

Hardware Register 9 WRITE CONTROL							
B7	B6	B5	B4	B3	B2	B1	B0
x	x	x	x	x	x	7210 INT CTL	EOT INT CTL

NOTE: Writing to Register 8 clears EOT latch.

Register 9 Bit Description:

IL0 - 3	I/O READBACK of INTERRUPT LEVEL
IRQ PEND	1 = OR of INTERRUPT PENDING BITS
7210 INT PEND	1 = 7210 CONTROLLER REQUESTING INTERRUPT
7210 INT ACTIVE	1 = BUS IRQ INT IS ENABLED
EOT INT ACTIVE	1 = BUS IRQ ON DMA EOT IS ENABLED
7210 INT CTL	SET = 1 TO ENABLE IRQ ON 7210 INTERRUPT
EOT INT CTL	SET = 1 TO ENABLE BUS IRQ DMA TERMINAL COUNT

The Base Address may be set on any 16 bit boundary in I/O address space via the PS/2's POS (Programmable Option Select) register. The NEC 7210 controller chip register functions are explained in the NEC 7210 data sheet. A more useful and expanded description of the controller chip functions is in a 90 page application manual titled "uPD7210 General Purpose Interface Bus Manual" available from NEC Electronics Inc. sales office. This information may be of value to the assembly language programmer interested in writing drivers for the uCMBC-488. Any register may be accessed by the appropriate OUT DX, AL or IN AL, DX instruction (equivalent to Basic's OUT & INP()).

uCMBC-488 Interrupts & DMA

The uCMBC-488 can generate an interrupt on any of the available MicroChannel levels, 3, 4, 5, 6, 7, 11, 15 as selected by the interrupt level POS Register. The interrupt is the logical "OR" of the request from the 7210 controller and the DMA controller Terminal Count (T/C). The uCMBC-488 driver, DV488UC.SYS makes use of interrupts and DMA requests only in DMA mode, the interrupt outputs and DMA REQUEST are tristated (high impedance) in normal program mode. However, the requested interrupt level will be installed in the first use of the "SYSCON" command as a chained handler. The handler will remain in the system until a reboot occurs. It is possible to program the 7210 controller to generate an interrupt on up to 13 different conditions including the obvious ones of a GPIB service request (SRQ) or an End Or Identify (EOI) or a specific End Of String (EOS) character. The hardware will support propagation of this interrupt into the PS/2, but it is the user's responsibility to provide a suitable interrupt service routine to handle the interrupt. Since BASIC and most other high level languages do not support interrupts (there is no ON INTERRUPT statement in Basic for instance) there is no support for interrupts in DV488UC.SYS. However, an assembly language programmer can take advantage of the hardware capabilities by writing drivers.

In a programmed data transfer the data travelling to or from the 7210 controller is transferred through the accumulator AX register of the 80286 (80386) CPU in the PS/2. This allows flexibilty in data handling but also involves a good deal of CPU overhead (the CPU continually polls the 7210 for new data which must be moved to/from the AX register and then move the AX register to/from memory). This takes time and is dependent upon CPU speed. Even with this overhead, transfer rates in programmed mode are usually around 1000-2000 bytes/second (also dependant upon GPIB device response speed). Speed is usually not a major factor in instrument control where command strings are short and response times long, but it may become a more serious problem with devices that have to move a lot of data in blocks across the GPIB.

uCMBC-488 Specifications

Electrical

+5v power	900mA max.
-5v, +12v, -12v	Not used
Controller chip	NEC uPD7210
Bus drivers	SN75160AN & SN75162AN or SN75160BN or SN75162BN (these are socketed for replacement)
I/O address	Can be set on any 16 bit boundary
PC bus loading	1 74LS TTL load on all inputs
Interrupts	Can be set on level 3,4,5,6,7,11,15 (only active in DMA transfer, tri-state otherwise). This is a shared handler.
DMA level	Selectable level 0-7 (only active in DMA transfer, tri-state otherwise)

Interface Repertoire

T6	Basic talker, serial poll, unaddressed if MLA
TE0	No extended talker function
L4	Basic listener, unaddressed if MTA
LE0	No extended listener function
SH1	Complete source handshake capability
AH1	Complete acceptor handshake capability
SR1	Complete service request capability
PP1	Parallel poll remote configuration capability
RL1	Complete remote/local capability
DC1	Complete device clear capability
DT1	Complete device trigger capability
C1,C2,C3,C4,C5	Controller states

Bus Performance

Data transfer rate	2 Kbyte/sec (normal program mode) 400 Kbyte/sec (D.M.A.) These rates assume zero response times for devices and that only one adapter arbitrates. In practice, transfer speeds may be lower, especially in D.M.A.
Devices on Bus	15 (max, including uCMBC-488)
uCMBC-488 Boards	2 (max, per driver or computer)
Maximum bus length	Two meters times the number of devices, not exceeding 20 meters
Connector	On rear plate 24 pin female micro-ribbon (Std. IEEE-488) with metric hardware.

Mechanical & Environmental

Card size	12.5" long x 3.12" high Full slot size
Weight	4.25 oz. (0.125 Kg)
Operating Temp	0 to 50 deg. C.
Storage Temp	-40 to 100 deg. C.
Humidity	0 - 90% non-condensing

APPENDIX A

IEEE-488 STANDARD

GPIB OPERATION DESCRIPTION

The IEEE Standard 488-1978 defines the IEEE-488 bus which is by far the most popular for electronic instrumentation. This bus was first introduced by the Hewlett Packard Company as the HP-IB (Hewlett Packard Interface Bus), later used by other manufacturers as the GPIB (General Purpose Interface Bus) and finally standardized by the IEEE as the IEEE-488 bus. All these buses are electrically identical. In Europe, the IEC-625 standard is electrically similar although a different connector system is used. The IEEE-488 standard has allowed many different manufacturers to build their devices so that they can easily be used together. A system can be set up simply by plugging devices together. Now that the hardware standard for the GPIB is complete, the IEEE is continuing its work on standard conventions in GPIB command syntax and structure. IEEE Std. 728 - 1982 is an example of this.

Controller, Talker, and Listener

There are 3 types of devices that can be connected to the GPIB, listeners, talkers and controllers. Some devices include more than one of these functions. The bus standard allows a maximum of 15 devices and a maximum cable length of 2 times the number of devices in meters or 20 meters whichever is less. A minimum system consists of one controller and one talker or listener device e.g. an IBM PC with MBC-488 and a digital voltmeter. It is possible to have several controllers on the bus but only one may be active at any given time. The active controller may pass control to another controller which in turn can pass it back or on to another controller etc. A listener is a device that can receive data from the bus when instructed by the controller and a talker likewise transmit data on to the bus when instructed. Apart from controller to device communications, the controller can set up a group of listeners as well as a talker so that it is possible to send data between groups of devices as well.

GPIB Control Signals

The bus is organized into 3 sets of signals: 8 data lines, 3 data byte transfer control lines and 5 general interface management lines. The specification defines the pin-out and type of connector hardware. Data is transmitted in a bit parallel, byte serial asynchronous form on the data lines. Logic levels are active low (negative true) and employ standard TTL voltage levels. Bus transceivers are arranged so that unpowered devices present a passive load to the bus and do not interfere with the operation of active devices.

IEEE-488 Connector pin Assignments

The GPIB connector is a 24 pin connector with signals assigned as below:

D1	1	13	D5
D2	2	14	D6
D3	3	15	D7
D4	4	16	D8
EOI	5	17	REN
DAV	6	18	GND
NRFD	7	19	GND
NDAC	8	20	GND
IFC	9	21	GND
SRQ	10	22	GND
ATN	11	23	GND
SHIELD (GND)	12	24	LOGIC GND

Signal Functions

D1-D8	Data input/output are the 8 bidirectional data lines. Used to carry all interface and device dependent messages.
EOI	End or identify. Used by a talker to indicate the end of a multi-byte transfer sequence, or by controller with ATN high to perform a parallel poll.
DAV	Data valid. Asserted when valid data on 8 data lines. Part of handshake.
NRFD	Not Ready for Data. Asserted by any device that is not yet ready to receive data. Part of handshake.
NDAC	Not Data Accepted. Released by any device that has accepted data. Part of handshake
IFC	Interface Clear. Master reset of GPIB.
SRQ	Service Request. Similar to an interrupt. Asserted by a device that requires service from the controller.
ATN	Attention. Used by controller to specify how data on GPIB is interpreted - data, commands or addresses.
REN	Remote Enable. Asserted by controller to take remote command of a device.
GND	Ground. Signal return, shield, logic and frame grounds. These are usually common although the functions are differentiated.

BUS Operation

The controller or talker may initiate data transfer using the Handshakle Process as follows:

1. Source initializes DAV to high
2. Acceptor(s) initialize NRFD to low and set NDAC TO LOW
3. Source checks for error condition then sets data on DIO lines
4. Source delays to allow data to settle on DIO lines
5. Acceptors have all indicated readiness to accept 1st data byte (NRDF goes high)
6. Source, upon sensing NRDF high, sets DAV low to indicate that data on DIO lines is settled and valid
7. First acceptor sets NRDF low to indicate that it is no longer ready, then accepts the data. Other acceptors follow at their own rates.
8. First acceptor sets NDAC high to indicate that it has accepted the data. (NDAC remains low due to other acceptors driving NDAC low)
9. Last acceptor sets NDAC high to indicate that it has accepted data; all have now accepted and the NDAC line remains high.
10. Source, having sensed that NDAC is high, sets DAV high. This indicates to the acceptors that data on the DIO lines must now be considered invalid.
11. Source changes data on DIO lines
12. Source delays to allow data to settle on DIO lines
13. Acceptors, upon sensing DAV high set NDAC low in preparation for next cycle. NDAC line goes as the first acceptor sets it low.
14. First acceptor indicates that it is ready for the data byte by setting NRDF high. (NRDF remains low due to other acceptors driving it low).
15. Last acceptor indicates that it is ready for the data byte by setting NRFD high; It remains high
16. Source, sensing NRDF high, setd DAV low to indicate that data on the DIO lines is settled and valid.
17. First acceptor sets NRDF low to indicate that it is no longer ready, then accepts the data.
18. First acceptor sets NDAC high to indicate that it has accepted data.
19. Last acceptor sets NDAC high to indicate that it has accepted data.
20. Source senses NDAC high, sets DAV high (as in 10).
21. Source removes data byte from DIO lines after setting DAV high.
22. Acceptors sense DAV high and, in turn, set NDAC low in preparation for next cycle.
23. All three handshake lines are now at initialized state.

Uniline Commands

	Command	D8	D7	D6	D5	D4	D3	D2	D1	ATN	EOI	SRQ	IFC	REN
ATN -	attention	x	x	x	x	x	x	x	x	1	x	x	x	x
IDY -	identify	x	x	x	x	x	x	x	x	x	1	x	x	x
SRQ -	service req	x	x	x	x	x	x	x	x	x	x	1	x	x
IFC -	i/f clear	x	x	x	x	x	x	x	x	x	x	x		x
REN -	remote enable	x	x	x	x	x	x	x	x	x	x	x	x	1

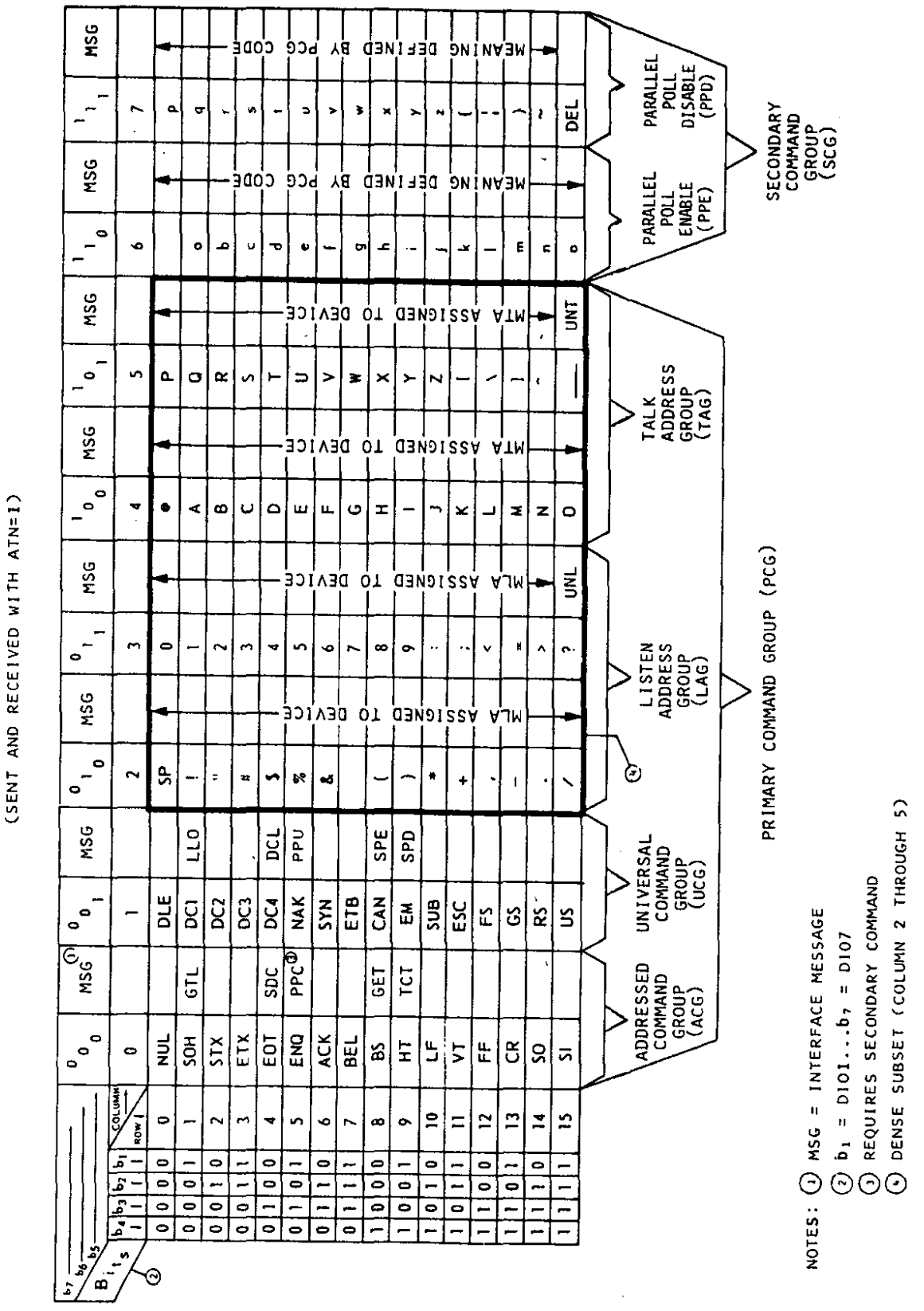
Multiline Commands

	Command	D8	D7	D6	D5	D4	D3	D2	D1	ATN	EOI	SRQ	IFC	REN
DAB -	data byte	B8	B7	B6	B5	B4	B3	B2	B1	0	x	x	x	x
GTL -	go to local	x	0	0	0	0	0	0	1	1	x	x	x	x
SDC -	sel dev clr	x	0	0	0	0	1	0	0	1	x	x	x	x
PPC -	par poll cfg	x	0	0	0	0	1	0	1	1	x	x	x	x
GET -	trigger	x	0	0	0	1	0	0	0	1	x	x	x	x
TCT -	take control	x	0	0	0	1	0	0	1	1	x	x	x	x
DCL -	device clear	x	0	0	1	0	1	0	0	1	x	x	x	x
LLO -	local lockout	x	0	0	1	0	0	0	1	1	x	x	x	x
PPU -	p poll uncfg	x	0	0	1	0	1	0	1	1	x	x	x	x
SPE -	ser poll en	x	0	0	1	1	0	0	0	1	x	x	x	x
SPD -	ser poll dis	x	0	0	1	1	0	0	1	1	x	x	x	x
MLA -	listen addr	x	1	0	L5	L4	L3	L2	L1	1	x	x	x	x
UNL -	unlisten	x	0	1	1	1	1	1	1	1	x	x	x	x
MTA -	talk address	x	1	0	T5	T4	T3	T2	T1	1	x	x	x	x
UNT -	untalk	x	1	0	1	1	1	1	1	1	x	x	x	x
PPE -	par poll en	x	1	1	0	S	P3	P2	P1	1	x	x	x	x
PPD -	p poll dis	x	1	1	1	D4	D3	D2	D1	1	x	x	x	x
PPR1-	p poll resp	x	x	x	x	x	x	x	1	1	1	x	x	x
	. . responses 1 - 8 have corresponding data bit set.													
PPR8-	p poll resp	1	x	x	x	x	x	x	x	1	1	x	x	x
MSA -	secondary ad	x	1	1	S5	S4	S3	S2	S1	1	x	x	x	x

Note how the GPIB relies for many control functions on certain specialized byte codings asserted with the ATN line high (Multiline commands). These codings are differentiated mainly by the state of the 6Th. and 7Th. data bits (the 8Th. bit, D8, is usually irrelevant). The bulk of the the control commands are asserted with D6 & D7 low and ATN high. D7 low and D6 high and ATN high imply a listen address or unlisten command, and D7 high and D6 low and ATN high imply a talk address or untalk command. If both D6 and D7 are high and ATN high this implies a secondary address or if EOI is also asserted, a parallel poll command. All data is transmitted after the commands with the ATN line low. Whether data, commands or addresses are being transmitted, the "three wire handshake" is always used to transfer the data except in the case of the parallel poll response. Not all devices are designed to respond to all possible commands. The capabilities of devices are usually designated by the manufacturer as a series of codings consisting of a letter or letters followed by a number. If the number is 0 then the device does not have that capability

Multiline Interface Messages:ISO Code Representation

The following chart has been reproduced for your convenience from the "IEEE Standard Digital Interface for Programmable Controllers" book mentioned at the end of this section.



Interface Functions

Function	Symbol	Levels
Source Handshake		SH 0,1
Acceptor Handshake		AH 0,1
Talker or Extended Talker	T or TE	0 - 8
Listener or Extended Listener	L or LE	0 - 4
Service Request	SR	0,1
Remote/local	RL	0 - 2
Parallel Poll	PP	0 - 2
Device Clear	DC	0 - 2
Device Trigger	DT	0,1
Controller	C	0 - 28

A full explanation of these codings is supplied in the IEEE-488 specification.

Sequence of Operations

A typical sequence of commands used by a controller talking to a programmable power supply might be as follows:

1) Controller sends IFC true to clear & initialize interface.

2) Controller sends DCL true to clear individual devices.

3) Controller sends MLA of power supply.

4) Controller enables REN to remote enable power supply.

5) Controller sends data byte(s) to program output (DAB).

6) Controller sends UNL unlisten command to end communication with power supply and unlisten all listeners.

7) Controller sends MLA or MTA for next device etc. Note that the controller must always enable a device as a listener or talker by transmitting the listen or talk address command before data can be transferred. When data transfer is complete, the controller tells all devices to unaddress themselves by sending the untalk and/or unlisten commands. Devices are in effect "toggled" on and off the bus by this sequence. It is possible to set up several devices to share the same data by addressing them and enabling them before placing data on the bus. The IEEE-488 bus supports both serial and parallel polling to determine which device may be requesting service. A talker initiates a serial poll by asserting the SRQ line. If the device has been enabled for serial polls (SPE), the controller can then perform a read of the each device's status byte in turn, if it detects a true state on D7 it means that that device generated the service request. A parallel poll after SRQ is set up by the PPE so that a particular device will activate a particular bit D1 - D7 when responding to a parallel poll request (PPRn). The bits returning set will then indicate which device(s) are requesting service. This is faster than a serial poll.

Adherence to IEEE-488 Standard

Not all manufacturers of devices adhere completely to the standard. For instance, it was the original intent of the standard that the last byte of a message would be signalled by asserting the EOI line, so that the controller or receiving device would know that the message was at an end and could then act upon it. Some manufacturers use a carriage return, line feed sequence and omit the EOI. With suitable image terminators, the MBC-488 interface will both respond to and generate this type of message ending.

Although unusual, manufacturers occasionally design devices that perform non-standard bus operations which are not so easily accommodated. This might include transmitting data with the ATN line high. This sort of situation may be difficult to handle with the standard MBC-488 commands that are designed to perform operations in accordance with the specification. In this case you may have to resort to programming the NEC 7210 controller on the MBC-488 directly. Information on doing this is provided in Section 8.

If you are interested in becoming more familiar with the IEEE-488 bus, the following references are among the many sources of information now available.

1. "IEEE Standard Digital Interface for Programmable Instruments". ANSI/IEEE Std. 488-1978. Published by the IEEE Inc., 345 East 47Th. St., New York, N.Y., 10017.

2. "488 together with Code & Format Conventions" Includes ANSI/IEEE Std. 488-1978 and 728-1982 in hardbound form. ISBN 471-80786-9. Published by John Wiley & Sons Inc.

APPENDIX B
IEEE-488 INTERFACE MESSAGES

The IEEE-488 standard (November 1978) defines 12 interface messages. Below is a listing of these messages. MetraByte's DV488 (device driver) supports all 12 of these message types.

MESSAGE	FUNCTION
The DATA Message	The actual data sent from the talker to one or more listeners on the GPIB.
The Trigger Message	Causes listener device(s) to perform a device dependent action when addressed.
The Clear Message	Causes addressed device(s) to return to a predefined state.
The Remote Message	Causes the addressed device(s) to switch from local (front panel) control to remote program control.
The Local Message	Causes addressed device(s) to clears the remore message.
Local Lockout Message	Prevents an operator from manually returning to local state via ther front panel controls.
Clear Lockout (Local Message)	Clear all devices on the GPIB and sets Local Mode
Service Request Message	Any device on the GPIB may send this message requesting service from the controller. Message is cleared by sending device status byte when service is no longer required.
Status Byte Message	A data byte representing status of the sending device. Bit 6 is set if the device sent a service request message with the remaining bits being device dependent.
The Status Bit Message	A data byte representing the condition (or status) of a group of devices on the GPIB with each bit representing a single device. This is a typical response to a parallel poll operation.
Pass Control Message	Allows transfer of bus management (control) to another device on the bus. There can be only one active bus controller at a time.
The Abort Message	The system controller takes unconditional control from the active controller, if different. This message also terminates communications on the bus and sends a Clear All message.

APPENDIX C
DV488 FILE LISTING for MetraByte's MBC-488

The following is a complete listing of the files on the 5.25" diskette supplied with your MBC-488 board (also available on 3.5" diskette). This listing is also in the file "FILES.DOC" on the diskette.

Documentation and Utility Files

FILES	DOC	Directory of DV488 files on diskette
README	DOC	File containing update history
LPT	DOC	Operating instuctions for Printer/Plotter device driver
DVSETUP	EXE	Automatic setup of user's CONFIG.SYS file
INSTALL	EXE	Aid in setting up MBC-488 Base Address DIP switches

System Files

DV488PVC	SYS	GPIB DOS Device Driver for IBM PC/XT/AT & PS/2 25,30
VIPARSE	SYS	Virtual Image Support system (Parse File Only)
LPT	SYS	Printer/Plotter Device Driver to be loaded by CONFIG.SYS
VI	SYS	Complete support DOS DEVICE DRIVER for parsing and use with pop up screen instruments. This file is included to supply the latest version to current users of PCIP instruments.

BASIC Language Demonstration/Example Programs

DVDMA	BAS	DMA Data Transfer
DVBL	BAS	Serial Poll with LOW BYTE Transfer
DVBH	BAS	Serial Poll with HIGH BYTE Transfer
DVSPOL	BAS	Serial Poll with STRING Transfer
DVB	BAS	Serial Poll with PACKED BYTE Transfer
DVFLUKE	BAS	Serial Poll with a FLUKE 8840A DMM
PLTRTST	BAS	Control of Plotter DOS DEVICE DRIVER
HPEXAMPL	BAS	Example Plot File that can be sent to an HP7475A Plotter from DOS using HOT keys and the TYPE command via LPT.SYS

QuickBASIC Language Demo/Example Files

QBDVB	BAS	Serial Poll with PACKED BYTE Transfer
QBDVDMA	BAS	DMA Data Transfer
QBDVBH	BAS	Serial Poll with HIGH BYTE Transfer
QBDVSPOL	BAS	Serial Poll with STRING Transfer
QBDVPPOL	BAS	Parallel Poll - Power Supply Control
QBDVBL	BAS	Serial Poll with LOW BYTE Transfer
QBDVMSTR	BAS	Dual Computer Communications {MASTER}
QBDVSLAV	BAS	Dual Computer Communications {SLAVE}
QBDVPCTL	BAS	Passing Control to Other Device
QBDVRCTL	BAS	Receiving Control
QBDV2DMA	BAS	Dual Board Simultaneous DMA
QBDVMSTS	BAS	Dual boards (1 Control, 1 Slave)
QBDVREQ2	BAS	"REQUEST 2" returns DMA & INT levels
QBDVTON	BAS	TALK ONLY Mode
QBDVLON	BAS	LISTEN ONLY Mode
QBDVDUAL	BAS	DUAL BUS Serial Poll
QBDVHP	BAS	Entering REAL numbers with an HP3458A DMM
QBDVBIG	BAS	Talk only of 64Kbytes

PASCAL, C, FORTRAN Language Specific Examples

CDVSPOL	C	C EXAMPLE of Serial Poll with STRING Transfer
CDVB	C	C EXAMPLE of Serial Poll with PACKED BYTE Transfer
CDVSPOL	MAK	C Make file - Serial Poll with STRING Transfer
CDVB	MAK	C Make file - Serial Poll with PACKED BYTE Transfer
FDVSPOL	FOR	FORTRAN EXAMPLE of Serial Poll with STRING Transfer
FDVB	FOR	FORTRAN EXAMPLE of Serial Poll with PACKED BYTE Transfer
PDVSPOL	PAS	TurboPASCAL EXAMPLE of Serial Poll with STRING Transfer
PDVB	PAS	TurboPASCAL EXAMPLE of Serial Poll w/ PACKED BYTE Transfer

Subdirectory (3.5") or second disk (5.25")

MBC488	Callable libraries and routines included for users with existing software (NOT RECOMMENDED FOR NEW APPLICATIONS)

APPENDIX D
DV488 FILE LISTING for MetraByte's uCMBC-488

The following is a complete file listing contained on the 3.5" diskette supplied with your uCMBC-488 board. This same listing is contained in "FILES.DOC" on the diskette.

System Files

DV488UC	SYS	GPIB Device Driver for PS/2 Models 50 thru 80
VIPARSE	SYS	Parse only support DOS DEVICE DRIVER
LPT	SYS	Printer/plotter Device Driver loaded by CONFIG.SYS
VI	SYS	Complete support DOS DEVICE DRIVER for parsing and use with Pop up screen instruments. This file is included to supply the latest version to current users of PCIP instruments.

Documentation, Utility and Executable Files

FILES	DOC	Directory of DV488 files on diskette
README	DOC	File containing update history
LPT	DOC	Operating instructions for printer/plotter driver
DVSETUP	EXE	Automatic setup of user's CONFIG.SYS file
@5018	ADF	PS/2 POS Configuration file
GENADF	EXE	Utility used to modify uCMBC-488 Base Adddress selection

BASIC Language Demo/Example Files

DVDMA	BAS	DMA Data Transfer
DVBL	BAS	Serial Poll with LOW BYTE Transfer
DVBH	BAS	Serial Poll with HIGH BYTE Transfer
DVSPOL	BAS	Serial Poll with STRING Transfer
DVB	BAS	Serial Poll with PACKED BYTE Transfer
DVFLUKE	BAS	Serial Poll with FLUKE 8840A DMM
PLTRTST	BAS	Control of Plotter DOS DEVICE DRIVER
HPEXAMPL	PLT	Example plot file that can be sent to a HP7475A plotter from DOS using HOT keys and the TYPE command via LPT.SYS

QuickBASIC Language Demo/Example Files

QBDVB	BAS	Serial Poll with PACKED BYTE Transfer
QBDVDMA	BAS	DMA Data Transfer
QBDVBH	BAS	Serial Poll with HIGH BYTE Transfer
QBDVSPOL	BAS	Serial Poll with STRING Transfer
QBDVPPOL	BAS	Parallel Poll - Power Supply Control
QBDVBL	BAS	Serial Poll with LOW BYTE Transfer
QBDVMSTR	BAS	Dual Computer Comm {MASTER}
QBDVSLAV	BAS	Dual Computer Comm {SLAVE}
QBDVPCTL	BAS	Passing Control to Other Device
QBDVRCTL	BAS	Receiving Control
QBDV2DMA	BAS	Dual Board Simultaneous DMA
QBDVMSTS	BAS	Dual boards (1 Control, 1 Slave)
QBDVREQ2	BAS	"REQUEST 2" returns DMA & INT levels
QBDVTON	BAS	TALK ONLY Mode
QBDVLON	BAS	LISTEN ONLY Mode
QBDVDUAL	BAS	DUAL BUS Serial Poll
QBDVHP	BAS	Entering REAL numbers with an HP3458A DMM
QBDVBIG	BAS	Talk Only Transfer of 64K bytes

PASCAL, C, FORTRAN Language Specific Examples

CDVSPOL	C	C EXAMPLE of Serial Poll with STRING Transfer
CDVB	C	C EXAMPLE of Serial Poll with PACKED BYTE Transfer
CVSPOL	MAK	C Make file - Serial Poll with STRING Transfer
CDVB	MAK	C Make file - Serial Poll with PACKED BYTE Transfer
FDVSPOL	FOR	FORTRAN EXAMPLE of Serial Poll with STRING Transfer
FDVB	FOR	FORTRAN EXAMPLE of Serial Poll w/ PACKED BYTE Transfer

PDVSPOL	PAS	TurboPASCAL EXAMPLE of Serial Poll w/ STRING Transfer
PDVB	PAS	TurboPASCAL EXAMPLE of Serial Poll w/ PACKED BYTE Transfer

Subdirectory (3.5")

UCMBC488	Callable libraries and routines included for users with existing software (NOT RECOMMENDED FOR NEW APPLICATIONS)

APPENDIX E
INSTALLING the LINE PRINTER DRIVER

LPT is a DOS installable driver for the uCMBC-488. LPT allows an IEEE-488 compatible device to be accessed as the systems line printer. LPT is a stand alone driver and does not require any other GPIB driver.

LPT configures the host machine as a talk only device, and the line printer as the listener. The GPIB is not a random access bus. LPT assumes that it is the only talker/controller on the bus. Other bus traffic, especially traffic involving the host machine, is likely to be disrupted.

To install the printer driver, modify the ***config.sys*** file to include:

DEVICE = <path>LPT.SYS B<bbb> A<pp> [/<ss>] [K<k>]

where;

<path>	is the directory path to the LPT.SYS file
<bbb>	is the Base Address (hex) of the uCMBC-488
<pp>	is the optional decimal primary base address of the printer
<ss>	is the optional decimal secondary of the printer
<k>	is the optional Hot Key selection number (below)

Choice	Key Sequence Plotter ON	Key Sequence Plotter OFF
1	<Alt> <A>	<Alt> <Z>
2	<Lft Shift><Y>	<Lft Shift><N>
3	<Rt Shift><U>	<Rt Shift><D>
NONE	<Ctrl><+>*	<Ctrl><->*

* The <+> and <-> keys are the grey keyss on the numeric keypad.

When the Plotter ON hat key is pressed, one beep is sounded. Likewise, when the Plotter OFF hot key is pressed, two beeps are sounded.

Installing LPT.SYS

The following example assumes;

1. LPT.SYS resides in the \DV\488 directory
2. uCMBC-488 has a Base Address of 300 (hex)
3. Primary Address = 5
4. No secondary address
5. Hot Key sequence is <Ctrl><+> for Plotter ON
   <Ctrl><-> for Plotter OFF

To install the LPT device driver , simply add the following line to your CONFIG.SYS file:

DEVICE = \DV\488\LPT.SYS B300 A5.