The Pickles & Trout IEEE-488/IEEE-696 Bus Converter

by Richard S. Newrock

Microcomputers are rapidly finding their way into the laboratory for numerical analysis, data reduction, experimental control and data taking. For the last two, there are three options: direct I/O via parallel ports (and occasionally, serial ports), analog I/O via the appropriate A/D and D/A converters, and the IEEE-488 instrument bus (GPIB or HPIB). This last is a most important method, as most state-of-the-art test and measurement instruments come with a 488-bus interface. If microcomputers are to be useful in the laboratory, they must be able to control such instruments. When I decided to switch my laboratory from minis to micros, the availability of a good S-100/488 bus converter was an important factor. This article reviews the bus converter I purchased, the Pickles & Trout 488 Bus Interface (P&T-488).

This review discusses the hardware and software aspects of the P&T-488. My conclusions can be summarized briefly: the P&T-488 bus converter is an excellent product. The hardware is well-designed and executed. It is simple to program in high-level languages or in assembler (when bus speed is important), and it comes with a useful and complete software package. The driver software, while slow, is good for most purposes, and adequate information is provided to help you write faster routines. Many good examples of software are displayed. Unfortunately, as is often the case in this field, the manual is poor.

In addition, and of great importance to me, the software driver routines are relocatable; they can be called from high-level languages. In particular, I can call them from Fortran programs, a considerable timesaver since nearly all of my research software is written in Fortran, as are the libraries of scientific subroutines I use.

The board is available for $450, directly from Pickles & Trout, P.O. Box 1206, Goleta, CA 93116. The price includes the board, software, test plug, and a ribbon cable with a 488 metric connector. The 488 connector is mounted on the rear panel of the computer and the ribbon cable is run between it and the board; a 488-bus cable is not supplied.

The hardware

The P&T-488 board is glass-epoxy and is solder-masked with silk-screened labels. Each component, the jumper area (for interrupts), and the address switch is clearly labeled. The soldering is cleanly done and all components are carefully mounted. Sockets are provided for the ICs. The ribbon cable connector on the board is of good quality; it is keyed and has connector ejectors. The only thing I didn't like was the 488 connector. The one provided looks as if it will not stand up to repeated use. The manufacturer assures me that it will, and that they tested several other types and found them wanting.

The board power supply, which consists of a single five-volt regular, looks to be more than sufficient. There is an adequate number of 0.1μF bypass capacitors distributed about the board to suppress switching transients.

The P&T-488 can generate interrupts, and provision has been made (via jumpers) to select NMI, pINT, or one of the vectored interrupt lines. Eight conditions can cause an interrupt: a change on any one of the three handshake lines (DAV, NRFD, and NDAC), four of the bus managment lines (IFC, ATN, SRQ, and REN) or POC/RESET on the S-100 bus. They are "or-ed" onto the selected line. Note that it is not necessary to use interrupts to operate the P&T-488.

Pickles & Trout does not make any claims about the compatibility of their board with the IEEE-696 standard; indeed, the pin labels on their schematic are the old S-100 names. To make certain it is compatible with the standard, I checked each pin assignment; there were no conflicts. The board does not use any of the undefined (NDEF) or reserved (RFU) lines, and there are no problems with the new ground lines. The P&T-488 does not support 16-bit data transfers (SIXTN* and SXTRQ* are not implemented), but that is unimportant for a device that uses ASCII codes. It is only addressable at the first 256 I/O ports, and, in that respect, does not meet the IEEE standard, but that is not critical.

Insofar as the IEEE-488 standard is concerned, extensive checks weren't necessary. The P&T-488 works at the data transfer rates for which it was designed, so we can assume that the bus timing, pin assignments, etc., are correct. I did check the line drivers; they are open collector, as they should be for a fully operational controller.

Registers. The user accesses the bus through four 8-bit registers that appear at four consecutive I/O addresses. The port is addressed, via a DIP switch, to any location which is an integral multiple of four (0,4,8, . . .). The board comes addressed at 7C and the software provided expects that address. A special routine allows you to use the software with a different board address. The use of these registers is straightforward, and they make the P&T-488 easy to use. They are worth further discussion.

Register 3, a write-only register, stores the parallel poll response byte. The CPU inserts a byte into this register to be placed on the data lines in response to such a poll.

Register 2 is the data line register; it is a read/write register connected to the 488-bus data lines. To read the data lines, the CPU reads the byte in

Richard S. Newrock, Dept. of Physics, Univ. of Cincinnati, Cincinnati, OH 45221

this register. It can also write a byte to this register to be transferred to the data lines. Whenever an external controller takes over the bus, or when POC/RESET occurs on the S-100 bus, flags are set that disable the P&T-488's output buffers. If so, whatever is contained in register 2 cannot be output.

Register 1, the command line register, is a read/write register that allows the user to set or sense the bus management and handshake lines. Again, if an external controller is active, the interface is inhibited. If an external interface clear (XIFC) is sensed, the P&T-488 will not set any bus management or handshake lines. If external attention (XATN) is set, no lines except "not-ready-for-data" (NRFD) and "service request" (SRQ) can be set. NRFD is made true to prevent an external controller from sending commands to the P&T-488 until its host CPU is ready. SRQ is set if the SRQ bit in register 2 is low; it permits the host CPU to signal the external controller that it wants service.

The read section of register 0 is the interrupt status register. The bits in this register change in response to changes in the state of the bus management and handshake lines, and to POC. This byte is used by the CPU to monitor bus status. In particular, since the board uses only one interrupt line, the CPU must read this register to determine the cause of the interrupt. Two of the status bits are the flags XATN and XIFC. The first of these flags, mentioned above, is set whenever an external controller takes over the bus; the second whenever an external controller issues an IFC.

The write section of register 0 is for interrupt reset. The upper six bits of this register are used to reset the status bits. Bit 1 is used to instruct the controller to be a listener or a talker. Bit zero enables or disables the interrupt.

To someone familiar with 488-bus operation, it should be clear that these registers are all that is needed to control the bus: register three is to respond to parallel polls; register two is to send or receive data; register one is to assert the handshake and bus management lines; and register zero is for status. Pickles & Trout provides examples of assembly language routines for source handshaking, acceptor handshaking, initialization, etc. The best way to understand the operation of the bus and the use of the registers is to examine these routines carefully. In addition, they are an excellent starting place for writing your own drivers.

Since Pickles & Trout supplies driver subroutines as part of the P&T-488 package, why would you want to write your own drivers? The answer is simple: speed. Pickles & Trout note that their software, with an 8080 CPU running at 2MHz with no memory wait states, will transfer data at 3KB/sec. This is rather slow. One reason is that the software continually checks for things that may be nonexistent (or unnecessary) in your system. For example, the software checks for the presence of another controller, for POC on the S-100 bus, for time limits on the handshake cycle, etc. Eliminating these checks (and others) by writing your own software will speed up the data transfer rate considerably. I have not measured the increase, but, according to the manufacturer, the maximum transfer rate should be about 9KB/sec with a 2MHz 8080 (and, perhaps, 22-23KB/sec with a 5Mz 8085).

The software

The software provided by Pickles & Trout can be divided into four parts. The package includes a routine to test bus operation; MSOFT, a package of Basic subroutines to operate the bus; three utility programs; and the aforementioned assembler source and acceptor handshake routines. I found all of it useful, if only for informational purposes.

Test program. The function-test program is a nice touch; I wish more manufacturers would supply such a routine. The program performs seven tests of the board and cable, which allow the purchaser to check the P&T-488 when it is received and at any time thereafter. When planning a new experiment, I often need to order new equipment; naturally, deliveries aren't simultaneous. I have often been in the position of having an instrument's warranty period elapse while waiting for something necessary to test it. The self-test program alleviates this for the P&T-488.

The first four tests are performed with nothing connected to the bus; with the last three a special test plug is used. The first four are simple and check the registers. They consist of writing a byte to the appropriate register and checking the P&T-488's response. If any of these tests fail, it is reported on the system console. Upon (successful) completion of these the operator is prompted to connect the test plug, which connects the data lines to the bus management and handshake lines. This allows the cable to be tested for shorts and continuity, and allows the P&T-488 to talk to itself to test the response to external IFC and ATN.

The tests are simple to use and take little time; my P&T-488 passed with no problems. Two versions of the test are supplied, because of recent revisions to the board. Be sure to check the board serial number and use the correct test routine. The test routine assumes the factory standard address; I found it simplest to test the P&T-488 there and make address changes later.

Utilities. Three utility programs are supplied: BUSMON, 488TODSK, and DSKTO488. The latter two send data from the bus to a disk file or send a disk file over the bus. I generally analyze data as it comes in and create files in my control

The Pickles & Trout 488 is an excellent product, well designed and executed. It comes with a useful and complete software package.

and analysis routines. As such, I don't need these utilities and did not test them. They should be particularly useful for sending data directly to a printer or plotter, for communication between computers, and collecting large amounts of data rapidly.

BUSMON monitors and reports all bus transactions. It reports in two forms: with no special character handling and with all control codes replaced by printable characters. BUSMON stops the processing on three conditions: the occurrence of LF, CR, or on every byte. When the processing stops, the user can enter bus commands from the keyboard, restart, and observe the results of the command. All instructions sent to the controller, and all data sent or received, are displayed on the console. Messages which indicate the occurrence of XIFC, XATN, etc., changes in SRQ, POC and REN, as well as identifiers for the various addressed and universal commands, are also displayed.

I found BUSMON to be useful for troubleshooting instrumentation systems, for learning about newly purchased instruments, and for general error checking. It probably is a useful learning tool for someone new to 488-bus operation.

MSOFT. I have two software packages for the P&T-488: MSOFT, and a set of routines called "CP/M-488." The CP/M-488 package came with the bus when I ordered it. When used, it alters CP/M's I/O routines to allow the P&T-488 to substitute directly for the console keyboard and display. A software switch allows the user to determine where the I/O goes: to the normal console or to the P&T-488. I found these routines clumsy, and the instructions unclear. When I called Pickles & Trout to get some assistance, they told me about the new MSOFT routines; they can be purchased for about $50. You now have a choice when you purchase the package; make sure you get MSOFT, as it is significantly better and easier to use than CP/M-488.

MSOFT is an interface program between P&T-488 and Microsoft Basic. It consists of two parts: MSOFT.COM and MSOFT.REL. The .COM file is used with interpreter basic; the .REL file, a library of relocatable subroutines, is meant to be used with compiled languages. A typical applications program has two parts: a Basic (or other high level language) program plus MSOFT. In a compiled language, the MSOFT routines are inserted at link-time; in interpreter Basic they are called before MBasic (i.e., at the prompt, one enters MSOFT MBASIC MYPROG.BAS).

The MSOFT package defines 11 communications variables and 13 communication functions, four set-up functions and one configuration function.

The variables are for communication to and from MSOFT. The user can choose any names he wishes, but he must tell MSOFT what they are. Several of these variables have obvious uses: the INPUT and the OUTPUT strings; the string LENGTH, POLL RESPONSE and BUS STATUS integers; and the input and output ECHO bytes.

The user should be aware of an important point concerning MSOFT's output. MSOFT always writes data into the same buffer. If the user wishes to save the data in that buffer he must move it before asking MSOFT to get more. This is a subtle point. The MSOFT routine LSTN(A$) tells the P&T-488 to become a listener. Data is read from the bus and stored in the buffer; the string A$ points to that buffer. That is, the string descriptor (described below) of A$ contains the address of MSOFT's buffer. If you now tell MSOFT to get more data, perhaps with LSTN(B$), the string descriptor for B$ will also contain the address of MSOFT's buffer, i.e., both string descriptors now point to the same buffer and therefore both strings contain the same data; whatever was contained in the buffer after the first LSTN command has been lost. To save the data you must move the string between calls to LSTN. For example, between calls to LSTN, use the Basic statement S$=A$. It will store the contents of MSOFT's buffer (pointed to by A$'s string descriptor) in a new buffer, pointed to by S$'s string descriptor.

Three of the variables have special uses:

ERROR CODE. This integer variable indicates if errors occurred during a bus operation. Each bit represents a different type of error.

TIMEOUT. This is the amount of time within which a handshake must occur, or an error will result. TIMEOUT can take any value between 0 and 255; if it equals 255, no check is made. It is important that a 488-bus operating system have a time limit, particularly in systems where the controlled instruments can be many meters away, and under local control. According to Pickles & Trout, with a 2MHz system clock, TIMEOUT=254 corresponds to about 6.5 seconds. This (maximum) time is much too short, more so with a 5MHz processor. Initializing some digital plotters, for example, can take 8-10 seconds.

EOT and EOS. These variables allow the user to set string terminators and other string parameters. This is a necessary feature; many older instruments do not follow the new standard for communications over the 488-bus, IEEE 728-1982.

The communications functions are used to operate the bus. There are three main communication routines which allow the user to control the bus, to listen and to talk. Two routines are provided for each of these functions. The first clears the error byte, performs the function, and then updates the

The CP/M-488 routines are clumsy and the instructions unclear. The MSOFT interface package, now available, is significantly better and easier to use.

error byte. The second performs the function and updates the error byte. The difference is quite important as it gives the user the option of calling a series of subroutines and checking for errors after the series is complete. This speeds up bus transactions considerably. The other communication functions are simpler. Some of them allow the user to reset the bus, clear the interface, enable or disable remote, and update the bus status variable. Others are for parallel and serial polls.

There are four set-up functions used to initialize MSOFT; they tell it the variable and function names. One, SETUP, is only used with interpreter Basic, where you must inform MSOFT.COM of the names of each the communication functions. This is not necessary in compiled languages, where the linker inserts relocatable subroutines where they are needed. The other set-up routines pass variable names to MSOFT, are needed by both MSOFT versions, and must be called at the beginning of all application programs.

The two most important set-up functions are IOSET and PROTCL. IOSET tells MSOFT the name you've chosen for the error code, the timeout value, the poll response byte and the bus status byte. PROTCL sets up the data transfer protocol, including the string lengths and string terminators. These variables may have to be initialized, depending on the language used. (Remember, Basic initializes all variables to zero; other languages may not.) In any case, IOSET and PROTC initialize time limit and string length to 254.

The last set-up function, ECHO, tells MSOFT the name of the byte which determines if bus I/O is echoed on the console. It defaults to no echo.

The last routine is the configuration function. It is called at the beginning of each program if MSOFT has to be informed of a change in the P&T-488's address (from 7C).

Sample programs. Pickles & Trout provides several sample programs. Four of these programs (BISAMPL.BAS, BCSAMPL.BAS, MTSAMPLE.PAS and FSAMPLE.FOR) allow the user to connect any 488-controllable instrument to the bus and play with it. They are menu driven: the user is asked what bus function he would like to perform and is prompted for the necessary parameters. I found these programs to be very useful. If you are unfamiliar with the 488-bus and its commands, these routines will allow you to play with the system, controlling one or more instruments, sending commands and collecting data, until you gain familiarity with the operation of the bus. The programs allow you to try a new instrument, testing it and learning about its programming quirks. Finally, and perhaps most important, the programs present many examples of the software necessary to operate the P&T-488. Unfortunately, the only place much of this information is presented is in these programs.

In addition to the four sample programs, Pickles & Trout provides examples of application pro-

grams written in interpreter and compiler Basic, Fortran, assembler, Pascal and C. These programs, which control a Hewlett-Packard 59309 digital clock, also contain many valuable examples of the use of MSOFT's functions. Although they were very informative, I think they would be even more useful if they referred to a more commonly available 488 instrument, such as a digital voltmeter.

Parameter conversion. The MSOFT communication functions are relocatable subroutines. Since MSOFT is designed to interface to Microsoft Basic, it passes parameters to such subroutines in the same manner as Basic: CALL SUBPROG(P1,P2,.....Pn) passes the parameters P1.....Pn. However, Basic passes parameters differently from other high-level languages. This means that an assembler program is necessary to convert from Basic's parameter passing convention to whatever convention your language requires.

One important difficulty occurs with strings. Basic stores strings in two parts, the string itself and the string descriptor. This last, a three-byte block, contains the number of characters in the string in the first byte and the address of the first character in the string in the second and last bytes. When Basic passes a string, it passes the memory address of the string descriptor. The called subprogram must look into that descriptor block to find the string address.

MSOFT works the same way. When a non-Basic program wants to pass a string to MSOFT, it must first convert the string to Basic's format; i.e., a string descriptor must be created. Similarly, when MSOFT returns a string (e.g., data) it must be converted to the form required by the calling program. The manufacturer provides several routines to perform and illustrate these conversions.

For assembly language programmers, Pickles & Trout wrote CLOCK.MAC. It illustrates how the addresses of passed parameters (strings and integers) are to be placed in the various registers and tables for MSOFT's use. For users of PASCAL/MT+, MT488.MAC is supplied to perform the parameter passing conversion. PASCAL passes addresses on the stack and expects the called routine to remove them from the stack. MT488 does this and places the addresses into the appropriate registers and tables for MSOFT. Several assembly language programs are provided for users of Fortran: STRIN.MAC, STRXFR.FOR and STRSET.FOR. STRIN collects strings from the keyboard and creates the string descriptor for BASIC. STRXFR copies strings from MSOFT's input buffer into a Fortran array. STRSET generates a string descriptor block for a Fortran array. For C a routine is provided to create a string descriptor.

I did not attempt to test all of the sample programs, but looked only at the ones in interpreter and compiler Basic, assembler, and Fortran. All of

the routines worked well. (In Fortran, I did not use STRIN.MAC, preferring to use a canned string-handling program The STRING BIT.) I wrote several application programs to control various instruments in my laboratory and found it to be a straightforward process. The software provided by Pickles & Trout works, and it works well.

The manual

Finally, a few comments about the manual. It is poorly written and extracting information from it is difficult. There are four major problems.

First, it is "schizophrenic." The authors obviously cannot decide on their audience. Very detailed explanations, suitable for the experienced user, are intermixed with material clearly of value only to the novice. While there is nothing wrong with writing a manual for both audiences, it must be done carefully; the advanced material must be well separated from the elementary and clearly identified, or the novice will get lost quickly.

I think that some of the simple material is just plain silly: for example, an entire page is devoted to a table showing all possible settings of the address switches. I would imagine that if a user is unable to address the P&T-488 without this table, he is probably unable to use it at all. However, Pickles & Trout tell me that they received many telephone calls regarding addressing before they included that table; now they receive none. They conclude that such "silly" details are of great help to many users. Perhaps they are correct; after all, experience is the best teacher. But such details belong in an appendix.

Second, the authors have difficulty in going between the general and the specific. They often start a general discussion of some aspect of the bus only to get bogged down in specific, nonilluminating details. For instance, in the middle of a description of uniline and multiline commands, using serial and parallel polls as examples, they go off on a tangent describing parallel poll instrument assignments. While this is important, it is completely out of place, and the reader quickly loses his train of thought about uniline and multiline commands.

Third, many of the important features and functions of the IEEE-488 bus are either inadequately described or not described at all. Important instructions and comments about a particular function are often located in three or four different places in the manual. One has to search to get a complete description of a function or to get a "recipe" for its use. Sometimes the information is in the text and sometimes it is buried in programs. This is exacerbated by the lack of an index. Surely, if you are going to scatter important information about in the text, an index should be provided to help you find it. I spent an inordinate amount of time searching for remarks I recalled reading, but couldn't locate. It would be best if each and every 488-bus command and function were individually

described with notations as to how to best implement the operation in software.

Fourth, and of less importance, the P&T-488 is meant to be used by microcomputer owners, a group that (probably) has little previous 488 experience. Therefore, a (reasonably) complete description of the bus should be given. An attempt to do so is made, but more care and detail are needed. A glossary of bus terms (taken from the IEEE standards document) is provided, but it is terse to the point where an inexperienced user will find it worthless.

Miscellaneous

Finally, there are two miscellaneous items:

—An interesting section of the manual contains a discussion of various "quirks, oddities and gotcha's." You should be aware of these when programming. Look here when your "bugfree" software crashes.

—I contacted the Pickles & Trout people several times while trying to get their software running; they were unfailingly polite and always helpful.

For more information contact:

Pickles & Trout
P.O. Box 1206
Goleta, CA 93116
(805) 685-4641