COMPANY CONFIDENTIAL

Engineering News

1 Nov. 1977

Tektronix Enters New Market with 8002 Microprocessor Lab

Tektronix Enters New Market with 8002 Microprocessor Lab

ELECTRO '77 ANNOUNCEMENT

On April 19, at the Electro 77 show in New York, the Logic Development Products group formally introduced the Tektronix 8002 Microprocessor Laboratory (ML).

The 8002 is a "general purpose" design tool because it can be used to develop software for more than one kind of microprocessor (the user adapts the 8002 system to each kind of microprocessor by using an emulator module board specific to that microprocessor). Most other entrants in the microprocessor design aid field are dedicated to one microprocessor or another.

For the microprocessor system designer who uses different kinds of microprocessors from one project to another, the ML is an attractive alternative to buying an additional design aid for each one. (Besides saving the cost of extra equipment, the designer also saves time by not having to learn to operate more than one design aid.) The ML is now available for work with 8080, 6800, Z80, 9900 and 8085 microprocessors. Emulator modules for others will be added to the product line soon.

THE COMPETITION

The market for design aids will be at least $40 million this year, and at least $100 million by 1980. In that market, the 8002 Microprocessor Lab faces two kinds of competition: direct and indirect. The 8002 is a multiple microprocessor support machine; other MDA's on the market are designed for use with a single microprocessor. They cost as much as the 8002 and are less versatile. New generations of microprocessors will quickly obsolete the single chip design aids. The 8002 faces direct competition from Intel, Motorola and Texas Instruments.

Besides the features of the 8002 itself, a big selling point is the fact that the 8002 is the only multiple microprocessor support design aid on the market backed by a service organization as large as Tektronix' worldwide service group.

THE PEOPLE BEHIND THE MDA

The MDA engineering, manufacturing and marketing groups (part of Logic Development Products) live in building 39. There are three engineering groups: hardware, software and human factors engineering.

OPERATION

Following is a description of the hardware and software features of the 8002. This discussion has been abstracted from the first section of the 8002 Microprocessor System User's Manual (part number 070-2313-00).

FOR MORE INFORMATION

If you would like more information about the 8002 Microprocessor Lab, call Ken Yabuki on ext. 6419 or drop by 39-282. Ken is the Logic Development Products marketing operations manager.

BASICS OF THE MICROPROCESSOR DESIGN LAB

INTRODUCTION

Developing microprocessor-based products requires a new kind of design tool: the microprocessor design aid (MDA). The objective of an MDA is to provide all members of a design team with a common working tool that closely approximates the real product they are trying to develop.

An MDA helps designers develop software and hardware, then helps integrate the two into a complete stand-alone microprocessor-based product.

FEATURES

System Programs Can Make A Difference

The 8002 Microprocessor Lab (ML), like most MDA's has three basic elements: a central processing unit, memory and input/output (I/O) facilities. It also contains *system programs* that perform supervisory functions. The editor and user programs help enter microprocessor instructions into the 8002 and assemble these instructions into a meaningful program for the prototype instrument under development.

Debugger programs monitor the user program as it is executing on the ML emulator processor (a microprocessor within the ML, identical to the microprocessor in the prototype). If errors are discovered and corrections must be made, the system programs help make the corrections.

So, "system" programs perform support functions and "user" programs are developed, tested, and debugged on the MDA.

Memory and I/O Capability

The 8002's large memory contains both resident RAM (Random Access Memory) and on-line disc storage which allows information to be stored on disc until it is needed, then quickly transferred into the ML's RAM work space for processing. Efficient memory and I/O capability in an MDA decrease the development cycle turn-around time. The 8002 features a fast and convenient flexible disc operating system with a total dynamic RAM storage capacity of 64k bytes (in Program Memory) and approximately 660k bytes of on-line storage capacity in the flexible disc unit.

Emulation is the method employed by the 8002 to check out a user program. Typically, an MDA that uses emulation contains a hardware model of the prototype microprocessor. This model may use discrete logic, a microprocessor of another type, or a microprocessor identical to the prototype microprocessor. The 8002 uses *substitutive emulation* (the MDA emulator processor is identical to the prototype microprocessor).

THE VALUE OF IN-PROTOTYPE TESTING

Typically, the simplest MDA's do not have the facilities for hardware development and testing. More complete MDA's provide limited signal monitoring functions, but most of these functions can also be handled by conventional hardware test equipment. The 8002 has the ability to swap known-good hardware elements into the prototype hardware and also swap known-good software programs.

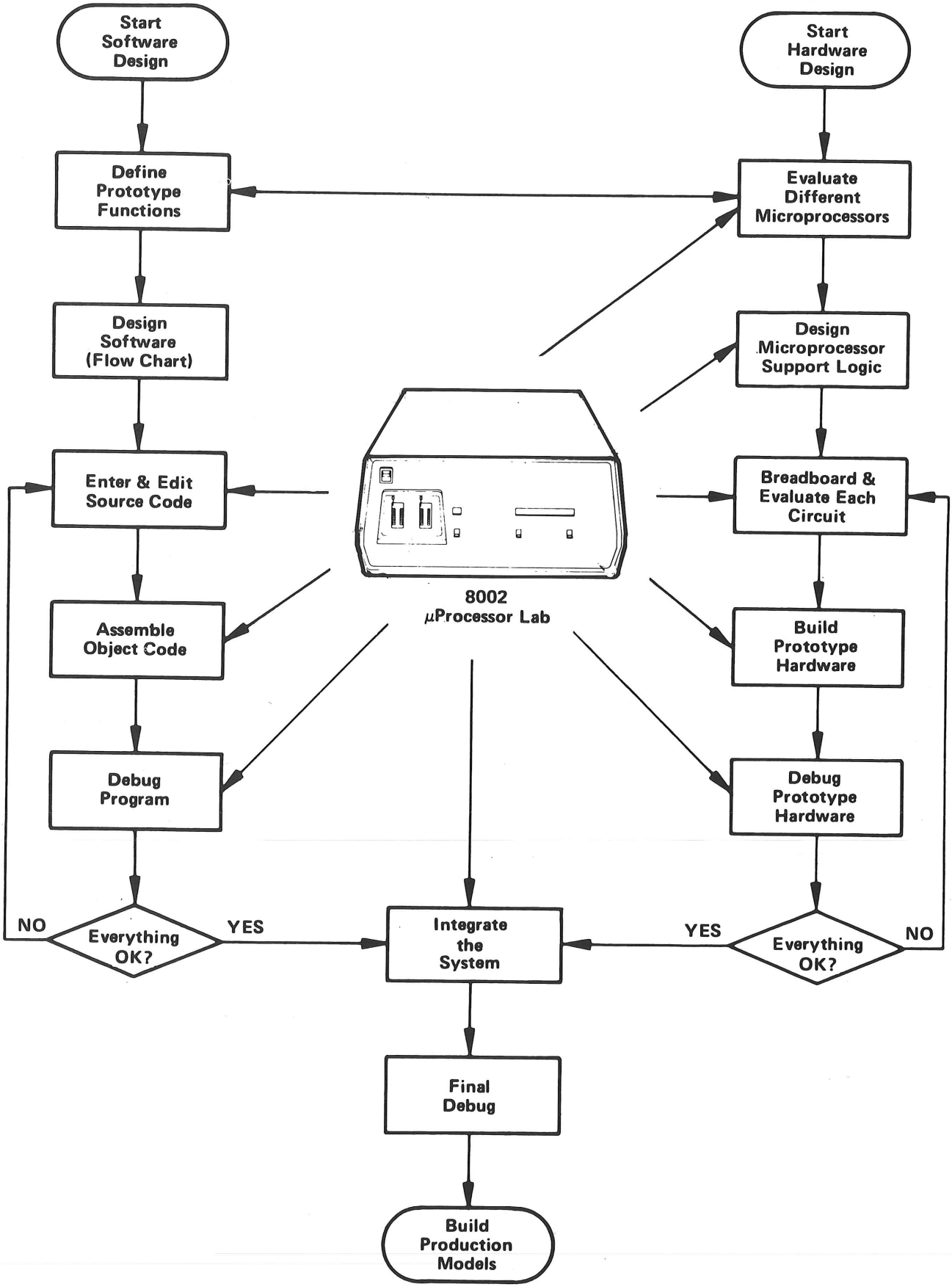
The basic idea is to connect portions of the MDA circuitry to the prototype hardware in the early stages of development so that the two parts can be exercised together as one complete microprocessor-based system. The combined unit then runs under the control of the developmental software while being supervised by the 8002's debug system program. This technique is called *in-prototype testing* and allows the system subcomponents (both hardware and software) to be tested, debugged, and verified as soon as they are complete. So, the entire prototype system is developed from the ground up on verified building blocks. The worry about a total system failure at the end of the development cycle is thus eliminated from the very start. The value of in-prototype testing cannot be over-emphasized.

MICROPROCESSOR DEVELOPMENT CYCLE

Unified hardware/software effort from conception to completion eliminates hard-to-find system integration bugs. This harmony can be achieved when a common environment is available to all members of the design team. The figure on the next page illustrates the microprocessor development cycle when the 8002 is used.

HARDWARE

The center of 8002 internal architecture is a system microprocessor that uses other microprocessors to perform different software and hardware support functions. The 8002 contains 16k bytes of system RAM memory and up to 64k bytes of RAM Program memory. The 8002 also supports two flexible disc drives with approximately 330k bytes on each disc.



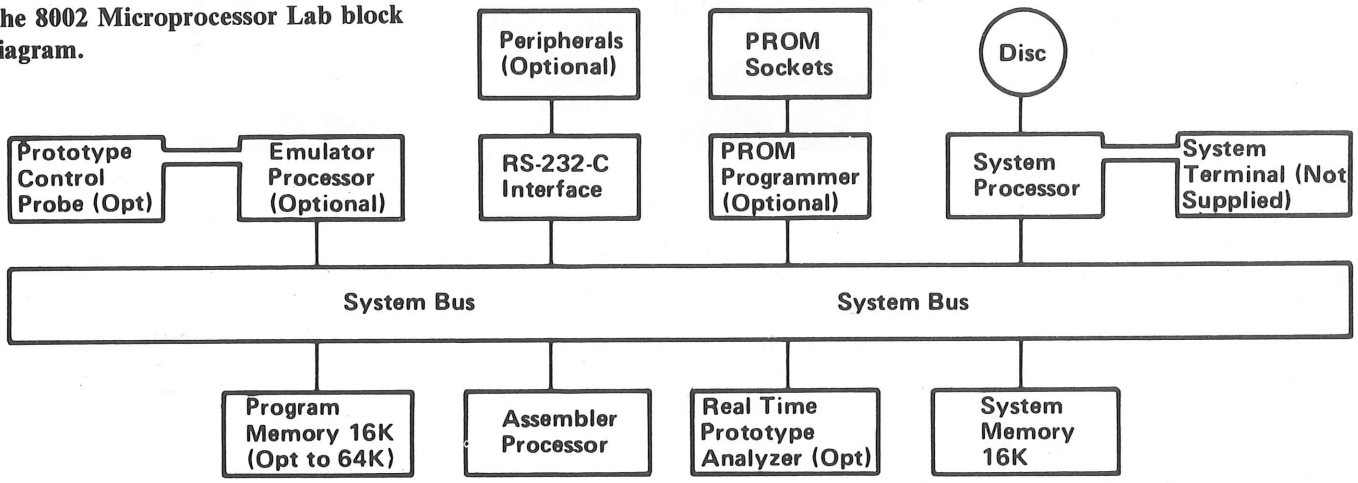
Microprocessor development cycle with 8002 Microprocessor Lab.

An 8002 system block diagram is shown below.

The system contains three microprocessors — the system processor, the assembler processor, and the emulator processor. The microprocessors reside on separate 8002 mainframe plug-in circuit cards that are connected to each other through a system bus. Also residing in the 8002 mainframe is the optional PROM programmer, the RS-232-C interface (with three I/O ports), the 16k byte system memory, and the standard 16k byte program memory (expandable to 64k).

The flexible disc unit communicates with the other system components through the system processor. Other optional system peripherals such as the CT8100 CRT Terminal and the LP8200 Line Printer communicate with the system through the RS-232-C interface.

The 8002 Microprocessor Lab block diagram.



SOFTWARE

TEKDOS

TEKDOS (Tektronix Disc Operating System) is the operating system for the 8002 and is loaded from the system disc into the flexible disc unit each time the system is powered up. TEKDOS contains the supervisory software programs for the system.

The Text Editor

The Text Editor is invoked by the TEKDOS EDIT command to (1) enter new user programs into memory, then store the programs on disc; (2) correct user programs that fail to pass through the Tektronix Assembler; and (3) add, delete, or modify program segments that fail to pass through the debugger system.

The Tektronix Assembler

After a source program has been entered and stored on a flexible disc by the text editor, the user program must be translated into machine-executable object code. This function is performed by the assembler which then stores the assembled object code in another file on the disc.

The assembler is loaded from disc into program memory and runs on the assembler processor. The assembler uses free space in program memory for I/O buffers and symbol tables. A different version of the assembler exists for each microprocessor supported by the 8002. Each version is loaded into the system by inserting a different system disc into the flexible disc unit.

The Linker

The linker software is considered a submodule of the assembler software and is provided with each system disc. The linker is used to join several smaller user program modules into one large program. This feature allows different software engineers to work on program segments independently, and then join the segments into a large workable program.

The Emulator

The emulator software allows user programs to be loaded into the optional emulator processor for testing and debugging.

The Debugger System

Since the assembler software can only detect syntax errors in the user program, some program logic errors may remain undetected until the user program is executed on a real microprocessor. The debugger system monitors user program execution on the emulator processor and the prototype microprocessor. The debugger software allows you to examine, trace, modify, and change portions of your program as the program executes. This feature ensures that the user program will be clean and free of bugs before it is placed in PROMs and plugged into the prototype instrument.

The PROM Programmer

The PROM Programmer software supervises and controls the transfer of user programs between the 8002 program memory and PROM chip plugged into the 8002 front panel. ■

SOFTWARE AND FIRMWARE ARE VALUABLE PROPERTY

It seems so easy to those of us who work with computers or microprocessors: a few keystrokes, a few lines of code, and our system is doing something new and different. Maybe even doing something really clever and interesting. And part of the fun is telling people about it.

Be careful. You may be literally giving away the store!

As microprocessors become increasingly important components in Tektronix products, the firmware which programs them becomes as much a proprietary resource as the CRT construction techniques or our IC masks. But there is one major difference: firmware can be listed, copied, and modified far more easily than hardware can. That means potential competitors can easily use it — if they get their hands on it.

EXAMPLE

A recent example of the problems which can arise occurred with the 4051 firmware. There is no doubt that a great deal of Tektronix money went into developing the system code which has been the basis of the 4051's success ... would you believe $10 a byte?

We recently found that a small California operation is openly offering information on how to modify the firmware, including ways to bypass the secret locks on tapes. As a result, we can no longer assure potential software developers of 4051 applications that their tapes will not be copied.

Larry Mayhew, Group Vice President, Information Display Group

Bill Walker, Group Vice President, Test and Measurements

And that is not the end of the problem. Several companies now offer add-on memory and peripherals for the 4051. Not only do their products deny us the sales (and profit-share) dollars, but they present a problem when the field service technician tries to provide the expected high level of customer support, because the non-Tek memory must be removed.

PREVENTION

Given that outside access to the code must be restricted, what can we do about it? There are several things, the most important of which is to be aware of the legal status.

Firmware is the property of the developer, and has the status of a trade secret. That means that information we have access to in the course of our duties still belongs to Tektronix, and it's illegal to use it for anything else. We have all signed a document in which we not only agree to keep secrets, but also to assign all inventions to Tektronix. That means all of our bright ideas, which relate to our work, belong to Tektronix.

So there's no doubt Tektronix has the right to protect its investment. It is to our own advantage, too, because it helps keep the company a strong and viable competitor which can continue to provide opportunities for employees.

The key to the situation is professional behavior which means at all times considering both the privileges and duties we have as Tektronix employees when we act.

These guidelines may help:

1. Don't talk to anyone who is not a Tektronix employee about the details of our products which are not yet announced. If any doubt exists consult your manager or the product group responsible for the product.

2. If an employee whom you don't know is seeking information, find out why. Tektronix prides itself on being an open company, but there ought to be some real need before proprietary information is discussed.

3. Help maintain building security. It is just courtesy to say "can I help you" to visitors to an area, and it is especially important to know who is in an area nights and weekends.

4. Lock sensitive information in a desk or filing cabinet. Just keeping it out of sight is one of the most important security factors.

5. Keep hobbies and work separated. The hobby world is open and friendly, and no place for information which must be protected.

If we all are aware of how important it is to keep our internal knowledge inside, we will all benefit from the profitable use of that information and from Tektronix' market leadership.

— Larry Mayhew
— Bill Walker

AVOID DUPLICATION: USE THE SPECIAL DESIGN FILE

Every large institution faces the problem of duplication of effort by isolated groups. In an advanced technology company, the problem is aggravated by the need for confidentiality: proprietary or sensitive material can't be widely published even inside the company because it may leak to the outside world. Yet there is still a need for each group to know what others have done because developments in one group may advance the work done in another group, or at least duplication of work may be avoided.

As a step in that direction, the Technical Information Department maintains a file for designs which have not yet found an immediate application. So far, the following people have contributed to the special design file: Bruce Campbell, Keith Parker, Ron Robinder, Gary Spence, and Dick Sunderland.

If you have a design you would like to keep alive but which doesn't have a direct product application in your area, give Technical Information a call. The following is a list of the special designs that have already been received. If you would like a copy of one of the designs listed, give Technical Information a call (ext. 5674) or drop by 50-462.

FILE NO.	TITLE
0001	E.C. Board, Tek 31 Universal Interface
0002	E.C. Board, Stepping Motor Control Logic, E4301X
0003	E.C. Board, Stepping Motor Driver, E4813XA
0004	E.C. Extender Board, 22/44 contacts on 0.156 centers, 7 in. long, E4810X
0005	Power Converter, Power Line to Single Output
0006	Thermoelectric Heater-Cooler, +85°C to +5°C
0007	Differential Instrumentation Analysis Amplifier
0008	Start-Run-Reset-Timer
0009	Vacuum Station Controller and Monitor System
0010	Elevated Grid Unblanking Circuit
0011	Logarithmic Amplifier for UTI 1200 Quadrupole Gas Analyzer
0012	Dual Filament Automatic Cathode Breakdown Unit
0013	High-Speed Photodetector
0014	Process Gas Analysis Program for Tek 31 Calculator
0015	Remote Gain Amplifier
0016	Heat Tape Temperature Controller
0017	BCD Interface for Systron Donner 7005 DVM
0018	T800 Vertical Pulser
0019	10 kHz to 3 MHz Sample Rate CCD Driver
0020	7000 Series Universal Power Plug-in
0021	512 X 512 D-to-A Converter Display with Readout
0022	10μA/volt Current Amplifier
0023	80 MHz Variable Duty Cycle Square Wave TTL Clock Drive
0024	Quick and Dirty Scan Converter Circuit

A SOLUTION TO M36 JITTER PROBLEM

The familiar M36 channel switch has been accused (rightfully in most cases) of causing CRT readout character jitter. The portion of this jitter actually due to the M36 can be explained by the signal distortion produced by thermal effects within the chip (past and present crosstalk, and self-heating).

A very low-jitter M36 can be made by disabling and bypassing two of the problem transistors in the M36 (Q7 and Q8, if you have your IC catalog handy). The penalty is that the designer must now arrange to handle the bias current changes that occur with different modes.

Because of the increasing need to display alphanumerics on CRT screens, a jitter-free channel switch may have wide usage. Call me if you need details and/or to inform us of probable applications. This circuit can be done with a simple mask change, but a new circuit that has all the benefits of an M36 and is also jitter free is another project altogether and not available yet.

— Carl Battjes ext. 5811 (50-316)

ENGINEERING NOTEBOOKS

As Bill Walker pointed out in the last issue of **Engineering News**, *engineering notebooks are valuable because they provide evidence in patent disputes and because they are a convenient reference for engineering information. In the following article, the Patents and Licenses department answers some of the questions frequently asked about engineering notebooks.*

WHY ARE THEY USED?

Engineering notebooks have two main functions. One is to provide the information Tektronix may need to prove that disputed inventions were conceived and reduced to practice by Tektronix employees. The other function is to provide a reference source for engineering concepts and data.

In disputes concerning inventions, time is a critical element. Engineering notebooks can show when the concept of the invention was recorded and when the working model of the invention was made. Evidence of those dates is required by the U.S. Patent and Trademark Office when more than one inventor is attempting to patent the same invention.

WHAT ARE THEY?

Tektronix engineering notebooks are bound volumes of lined and consecutively-numbered pages. Each notebook is assigned a number and is issued to the person who signs for it. If you have a notebook but you are leaving Tektronix, you must return it to the Patents and Licenses Department because the notebook is Tektronix property.

HOW SHOULD THEY BE USED?

When you start a new project or experiment go to Patents and Licenses and pick up a notebook. This book is assigned to you by name and number. As your project develops, include sketches, block diagrams, schematics, theory of operation, test data and anything else that will establish the concept and the working model of the invention. Of course, unnecessary information clutters the book and weakens its value as a reference.

Concepts and data should be recorded as soon as they are available to avoid the inaccuracies and omissions that seem to occur with the passage of time. The pages must be dated and signed by two witnesses. While it isn't necessary to witness every page, witnessing every five or ten pages should include a statement about which pages are being covered.

WHERE DO I GET ONE?

You can obtain an engineering notebook from the Patents and Licenses Department by calling ext. 7787.

MAKING PHYSICISTS MORE VISIBLE

Jim Deer (Component Engineering) is compiling a list of Tektronix employees who are members of the American Physical Society. That information will be used to increase the visibility of physics as a profession at Tektronix. If you are an APS member, please notify Jim on ext. 7711 or drop by 58-299.

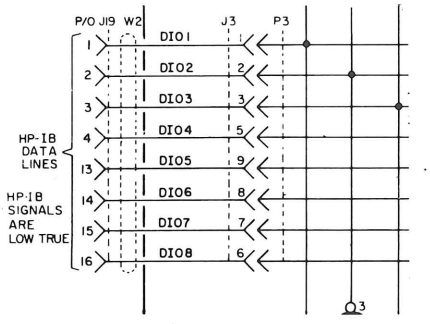
DEMAND HEAVY FOR TEKTRONIX STANDARD

Abbreviations, Acronyms, and Symbols (Tektronix Standard 062-1737-00) brings together most of the "short-forms" used inside Tektronix. This standard also includes the American Standards Institute (ANSI) Y1.1 Standard on Abbreviations.

The Tektronix compiled portion of 062-1737-00 may be obtained from Reprographics by calling ext. 5577. The ANSI portion of the standard may be ordered to your account number through Technical Standards (call ext. 7976); the cost is $12.

WHAT IS THE COMPETITION DOING WITH GPIB?

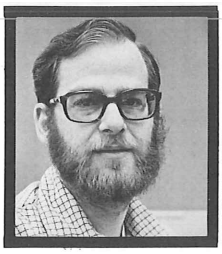
Would you like to see how Hewlett-Packard, Wavetek and Fluke are designing their GPIB instrument interfaces? Maris Graube, the Tektronix corporate interface engineer, has instrument manuals from those three companies and others too. The manuals, which are available as loaners, describe in detail the workings of the instruments and their interfaces (some of the manuals include schematics). If you would like to borrow one, call Maris on ext. 6234.



Part of a Hewlett-Packard schematic showing GPIB data lines.

IN-CIRCUIT EMULATION USING THE TARGET MICROPROCESSOR

Steve presented this paper at ELECTRO/77 in New York in April.



Steve Dum, Human Factors Engineering, ext. 7161.

ABSTRACT

This paper looks at several approaches to developing microprocessor design aids (MDA's) from the user's viewpoint. The paper identifies the features that the user wants in an MDA, and then describes the approaches the MDA designer can take to provide those features. The paper next discusses the factors the designer needs to consider for one of those approaches: in-circuit emulation. The last section looks at an example use of an MDA.

INTRODUCTION

Special tools are required to efficiently develop a microprocessor system. In general, these can be classified as microprocessor design aids (MDA's) The objective of the MDA's is to mimic the action of the microprocessor so the system can be developed in a user-controlled atmosphere. Both emulation and simulation are methods of doing that. However, only a portion of the microprocessor system can be checked out if the MDA does not have in-circuit emulation.

This paper will look at several approaches to designing MDA's. It will identify the features that a user will need in an MDA. The paper describes approaches the MDA designer can take to provide these features. We will then show how in-circuit emulation relates to those approaches. We will also show that in-circuit emulation using the target microprocessor is one of the better approaches to use. Finally, we will give an example of how this can be used in a specific design.

Figure 1 shows an example of a microprocessor design aid system. The target microprocessor system is a prototype of the product the user is designing.

THE FEATURES THE USER DESIRES IN AN MDA

Verification

The primary use of the MDA is for verification of the target microprocessor system functions. For the firmware, each routine must be checked to make sure it produces the proper outputs for all valid inputs. The hardware must be checked to make sure the address and data buses are functional, and that the related logic works.

Input/output circuits must be checked, too. Often this is the most difficult part of the system to check out, because the I/O circuits usually operate asynchronously (relative to the microprocessor). If the target system will be used for event timing or sequence timing, this logic must be verified, too.

Storage

Loading and saving program object code on some external medium is another desirable feature.

Examine

The ability to examine memory locations, registers, and I/O ports of the target microprocessor system is also important.

Change

The user also wants the ability to alter memory locations, registers and I/O ports of the target microprocessor system.

Software Trace

Another feature is executing one instruction at a time and displaying the status of the registers (this allows the designer to verify program execution).

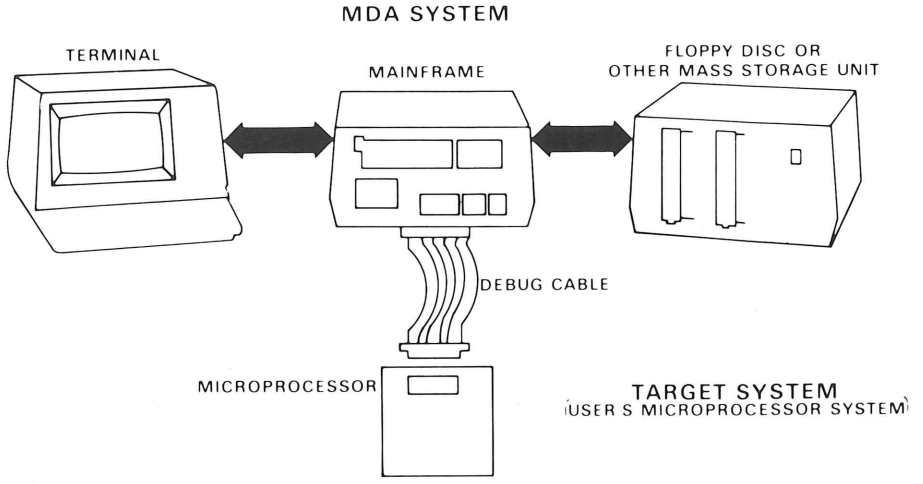


Figure 1. Example microprocessor design aid configuration.

Disassembly

A sixth feature is the ability to disassemble the executed instructions. It's unreasonable to expect the designer to memorize the bit patterns for each microprocessor instruction, especially when the software can do the decoding for him.

Timing Statistics

A seventh feature is the ability to gather statistics on timing. This allows identification of the most time-consuming areas of the user program.

I/O Statistics

Input/output operations usually occur asynchronously. Statistics about I/O performance greatly aid in analyzing and debugging the I/O routines.

Download Capability

The user also wants the ability to load programs from a timeshare computer (preferably directly without resorting to paper tape or other storage media).

Transparency

It is of utmost importance that the user program has full usage of the address and I/O space of the user's microprocessor. If the MDA requires that a block of user address or I/O space not be used, then the MDA itself cannot be used in some otherwise valid user applications.

Breakpoints

Often the user wants to be able to interrupt his program when a set of conditions have been met. A breakpoint is the place where execution of the program is interrupted. The MDA may use either the hardware or the software to detect the breakpoint conditions and stop the user program. Breakpoint conditions may include specific addresses that appear on the bus, the status of the read/write line, microprocessor control signals, or even external user-defined signals. With the program stopped, the user can examine or change the status of the target system. The user can then continue the execution of the program.

Memory Mapping

A general purpose development aid is one that the designer can use throughout the development cycle right down to debugging the etched circuit board prototype with ROMs or PROMs inserted. This means the MDA must be able to replace the microprocessor in the prototype target system with a debug cable and continue to emulate its action in the prototype system. (This is in-circuit emulation, which is described in detail later.) When using the debug cable it is important to be able to use the memory in the prototype system or to be able to disable the memory in the prototype system and substitute for it memory in the MDA system. The MDA should do this without requiring any special circuitry in the prototype target system.

Hardware Trace

The features we have talked about don't allow the designer to completely debug the target system because some of them don't operate in real time. Including a logic analyzer as part of the MDA can correct that problem. The logic analyzer gives the designer a look at program timing and response characteristics. The designer may also use the logic analyzer to store

- selected bus cycles,
- selected op code cycles,
- or all bus cycles.

The logic analyzer's clock may be conditioned so that the logic analyzer will store subroutine calls and returns, jumps, branches, stack operations, or information when the output from a complex comparator is true, or a match is obtained from a content-addressable memory (CAM). The trigger for the logic analyzer can be generated from the same conditions that generate the clock. Normally however, the trigger is either the output of the CAM, or the signal that results from filling the logic analyzer buffer.

Software Development

If the MDA is, in fact, a computer system, then providing the user with software development capability is also reasonable. The user will have a complete microprocessor system development package if the MDA provides an editor and an assembler (and perhaps a PROM programmer).

APPROACHES TO MDA DESIGN

Emulation vs. Simulation

There are two basic modes of MDA operation: simulation and emulation (see figure 2). Simulation is a method of checking out the target microprocessor system firmware. In this mode, the MDA includes a software interpreter. The interpreter "simulates" the action of the object code as though the code were running on the target microprocessor.

Emulation is another way of checking out firmware for the target system. In the emulation mode, the MDA uses a hardware model to mimic the microprocessor. To build the model, the MDA designer may use bit-slice architecture, discrete logic, or even the target microprocessor itself (with some additional support hardware). If the MDA designer uses the target microprocessor as the model, the mode is called "substitutive emulation".

Target System Considerations

Different kinds of microprocessor target systems call for varied emphasis on each of the verification procedures performed by the MDA.

If the microprocessor system is firmware-intensive and makes many analytical or mathematical calculations, then the designer will spend the most time verifying the algorithms and

ironing out miscellaneous bugs in the code. A software simulator may be an adequate verification tool.

On the other hand, if the designer is working on a control application he will spend most of his time analyzing the operation of the target system hardware while it is either controlling the target microprocessor or being controlled by the target microprocessor. Here a hardware emulator with in-circuit emulation is the most effective verification tool.

Since most applications involve both control and analytical calculations to some degree, in-circuit emulation is the ideal approach because it can handle either type of application.

signers tend to use high level languages. If the designer is using a high-level language, there is less concern with the machine instructions actually executed. Program flow and the values of defined variables then become the most important information.

In-Circuit Emulation

An MDA can provide the hardware and software features we have mentioned throughout the development process ... but only if the MDA lets the designer replace the target microprocessor with a debug cable. See figure 3. The debug cable allows the user to perform emulation of the target microprocessor within the prototype system. Thus he can test the target system hardware and software as it will appear in the final product.

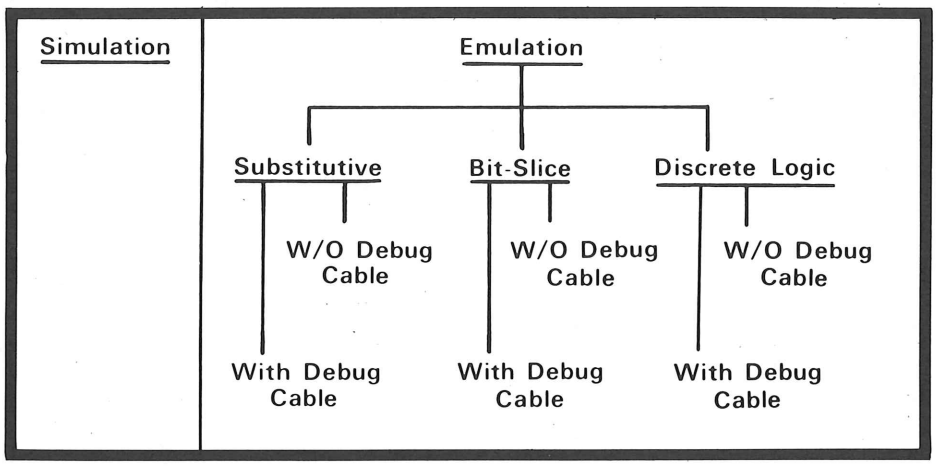


Figure 2. Microprocessor design aid configurations.

The type of source language the designer uses to develop the system also affects the designer's choice of an MDA. Today, most microprocessor system designers use assembly language (a low level language) to develop firmware. For assembly language development, the ability to observe each instruction executed and the contents of the microprocessor registers is very important to the designer.

Microprocessor programs are getting longer (and more expensive to produce); designer's time is becoming more expensive. At the same time, high level translators are becoming more efficient and the price of ROMs is going down. So it's not surprising that more de-

In-circuit emulation can be used with any of the emulation methods described earlier. The purpose of each of the three routes to emulation is to mimic the microprocessor used in the target system. It's difficult to use discrete logic or bit-slice emulation and get an exact mimic. The problem is to guarantee that the circuitry emulates the microprocessor exactly.

Each microprocessor has its own idiosyncrasies. The idiosyncrasies may result either from design faults, oversights, or simply arbitrary decisions by the chip designer. Guaranteeing that discrete hardware emulation, bit-slice emulation, or software simulation will have the same idiosyncrasies as the target microprocessor is virtually impossible. For example, the microprocessor manufacturer may release a new version of a mask to correct bugs in the old design. The trouble is, the new version may contain new bugs. Consider the market for 8080 equivalents. There are microprocessors available that are 99.9 percent the same as the Intel 8080A, however, the equivalents set flags differently than the 8080A under some conditions.

If the development tool cannot account for all of the idiosyncrasies of the target microprocessor, correct emulation isn't assured. There is a way to get that assurance: use the same microprocessor as the target device (substitutive emulation).

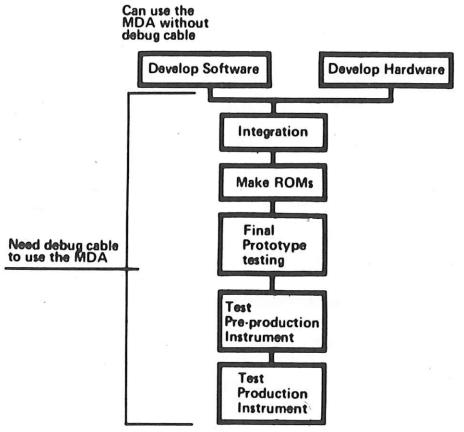


Figure 3. Microprocessor system design process.

DEBUG CABLE DESIGN CONSIDERATIONS

General

The goal of in-circuit emulation is to provide the user with emulation capability at any time he would usually have the microprocessor inserted in the target system. While the debug cable is inserted, the target system circuits should see the same signals from the cable that they would see from the microprocessor. This is not always possible because at times the user wants to execute only one instruction at a time. Executing one instruction at a time forces the MDA to stop the microprocessor perceived by the target system after each instruction. Now, let's look at some of the problems the MDA must solve in trying to match the debug cable responses to those of the target microprocessor.

Stopping the Microprocessor

The MDA must be able to stop the microprocessor. One way the MDA can do this is by giving a Pause or Halt signal to the microprocessor. This technique is useful only if the MDA can examine the status of the microprocessor once it is stopped.

A second method of stopping the microprocessor is to use interrupts. When the microprocessor is interrupted the MDA diverts status information from storage in the user's memory into a special cache in the MDA. The MDA then maps some debug software into the microprocessor's address space. This software, along with any necessary hardware, lets the MDA access any target microprocessor status it needs. The software also provides for any other data transfers necessary to enable the other MDA features. When the user is ready to resume program execution, the MDA will restore the microprocessor's status and then map the special memory out of the microprocessor's address space.

A third way to stop the microprocessor is to force a jump instruction onto the microprocessor's data bus. This enables the MDA to force the target microprocessor into a special software service routine as in the interrupt method above. In order to do this, the microprocessor must provide a signal which shows that the first cycle of an instruction has been fetched. A hardware latch is needed to save the program counter of the microprocessor when the MDA forces the jump instruction onto the bus.

Of the three methods described above, the third provides the least possibility of affecting the target microprocessor's status, while still remaining generally applicable. For this reason it is probably the best choice for a universal MDA designed for today's microprocessors.

Handling Interrupts

The MDA designer must give special attention to interrupts at two points: when the MDA stops the target microprocessor, and when it resumes the operation of the target microprocessor. When the MDA stops the target microprocessor, the MDA must mask out the interrupts so that spurious actions do not occur. However, when the MDA resumes target microprocessor operation, actions consistent with the target system interrupts should occur. For example, if the target microprocessor has a non-maskable interrupt while it is stopped, then the interrupt should appear to occur when the MDA restarts the target microprocessor.

Whenever the MDA runs the target microprocessor in a non-real-time mode, (a software trace mode for example), interrupts will not occur exactly as they would in real time. The user must be aware of this and use real-time debug aids, such as the hardware trace to debug his interrupt handling routines.

Matching Signal Characteristics

The signal characteristics of the debug cable should match the signal characteristics of the microprocessor. When the cable replaces the microprocessor in the target system, delays are introduced by the cable length and associated logic. Also, the loading and the drive power of the debug cable and the microprocessor may not be the same.

Today, debug cables are designed with buffers located close to the microprocessor substitution socket of the cable. This eliminates some of the noise problems of the long debug cable. The buffers may be 15 to 50 cm from the socket in order to allow the target system designer access to the prototype. The prototype breadboard system may be wire-wrapped with a considerable length of wire going to the microprocessor socket. This wire length along with the buffer-to-socket wire length may make noise a serious problem for the debug cable. The noise, loading and drive characteristics of the debug cable are areas the MDA designer needs to focus his attention on in the future.

AN EXAMPLE USING IN-CIRCUIT EMULATION

To illustrate the MDA concepts we've talked about, let's consider an example of an engineer designing a point-of-sale (POS) terminal (see figure 4). At the beginning of the development process, the hardware engineer sketches out the hardware design. The firmware engineer will do the same for the firmware. They will discuss the trade-offs between hardware and firmware. Next, they will implement their designs.

The firmware engineer will use the MDA to develop and debug the software even before the hardware engineer has the hardware working. The hardware engineer will use the MDA to check out the hardware operations of the prototype circuits.

Next, the two engineers integrate their designs. They will use in-circuit emulation to verify that the firmware can drive the hardware. First, they load the firmware into the memory in the MDA and map the firmware into the target system by using the memory mapping features of the MDA.

After they've ironed out the bugs in the POS terminal they will burn the firmware into a PROM for final test of the POS terminal prototype using in-circuit emulation without memory mapping. Finally, the designers will plug the microprocessor in the POS (they now have stand-alone prototype).

At later development stages, the designers can unplug the microprocessor and plug in the debug cable to track down bugs. For example, if mask ROMs arrive from the manufacturer but don't work when they're put into production-stage terminals, the designers can use in-circuit emulation to isolate the problem with the new ROMs.

The in-circuit emulation development tool gives the designer debug capability from the design concept to the production stage. Using substitutive emulation with the MDA gives the designer added assurance that the debug cable will respond in the same manner as the target microprocessor.

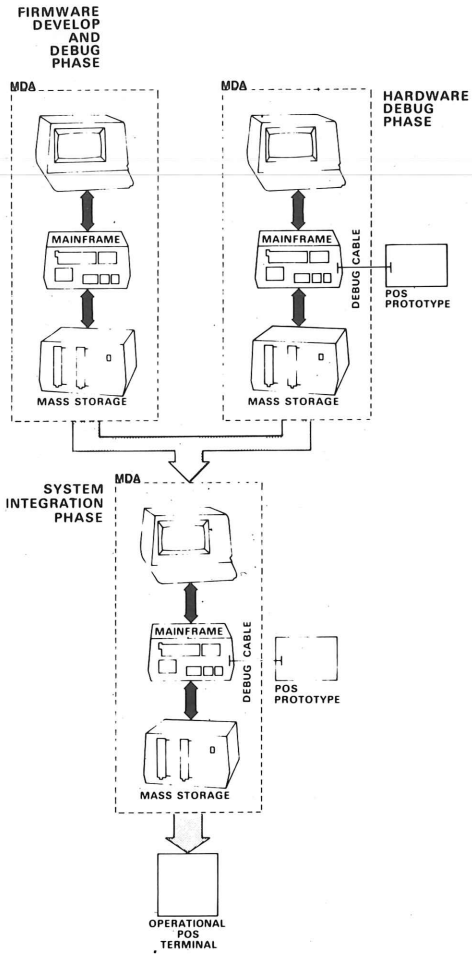


Figure 4. MDA configurations in the design process.

THE ISO

The International Organization for Standardization (ISO) is an agency whose objective is to secure the benefits accruing from agreement on international standards: the expansion of trade, improved product quality, increased productivity, and reduced prices for the consumer.

The ISO works to establish standards in most every area of technology except electrical and electronic which are the special domain of the International Electrotechnical Commission (IEC). The two organizations have been affiliated since 1947.

ISO standards presently have no legal force nationally or internationally, but if several countries adopt them as national standards for inspection and certification then they can influence trade.

The ISO was formed in 1947 as the successor to the International Federation of the National Standardization Associations. The American National Standards Institute (ANSI) has been a member of the ISA and later the ISO in the form of the American Standards Associaton and the United States of America Standards Institute.

If you need more information about the ISO or other standards organizations, give me a call on ext. 7976.

— Chuck Sullivan

Help is on the way for GPIB

It is becoming apparent that the semiconductor manufacturers are as aware of the GPIB (General Purpose Interface Bus) market potential and its attendant problems as are the instrument makers. Motorola is about to introduce a GPIB interface chip (the MC68488). The product preview sheet lists these major features of the MC68488:

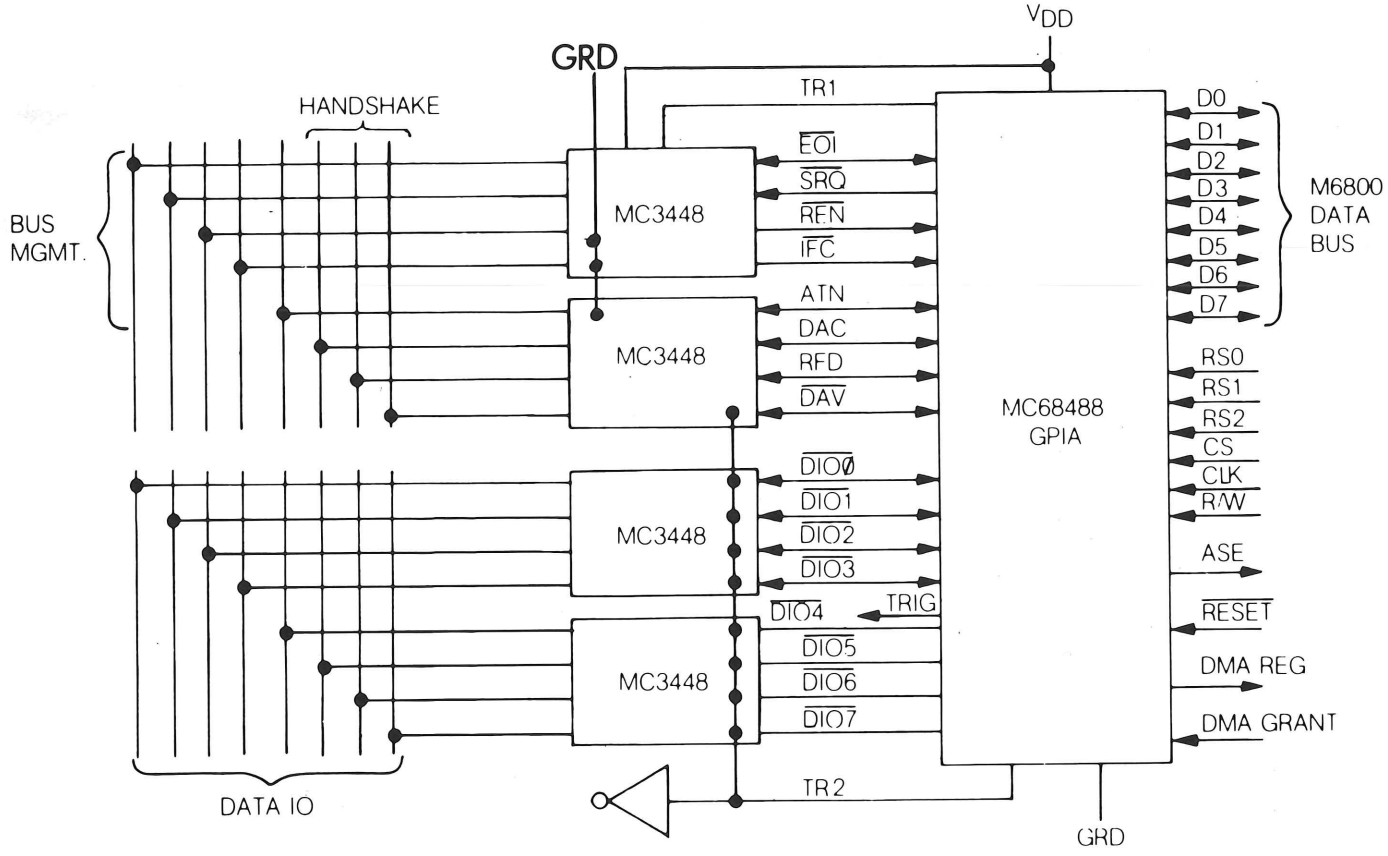
- M6800 bus compatibility
- Single or dual primary address recognition
- Secondary address capability
- Complete source and acceptor handshakes
- Programmable interrupts
- RFD holdoff to prevent data overrun
- Operates with DMA controller
- Serial and parallel polling capability
- Talk-only or listen-only capability
- Selectable automatic features to minimize software
- Synchronization trigger output

We've reprinted the pinout diagram below for your information. On the next page we've also reprinted the general description and system block diagram directly from the Motorola product preview. For more information, call Jim Howe on ext. 5698 or drop by 58-299 (Component Evaluation).

Jim's evaluation of the MC68488 recently appeared in **Component News.**

GENERAL DESCRIPTION

The IEEE 488 instrument bus standard is a bit-parallel, byte-serial bus structure designed for communication to and from intelligent instruments. Using this standard, many instruments may be interconnected and remotely and automatically controlled or programmed. Data may be taken from, sent to, or transferred between instruments. A bus controller dictates the role of each device by making the attention line true and sending talk or listen addresses on the instrument bus data lines; those devices which have matching addresses are activated. Device addresses are set into each GPIA (General Purpose Interface Adapter) from switches or jumpers on a PC



The new Motorola MC68488 GPIA (General Purpose Interface Adapter) works with standard 488-bus driver IC's (MC3448 in this diagram) to meet IEE 488 specifications.

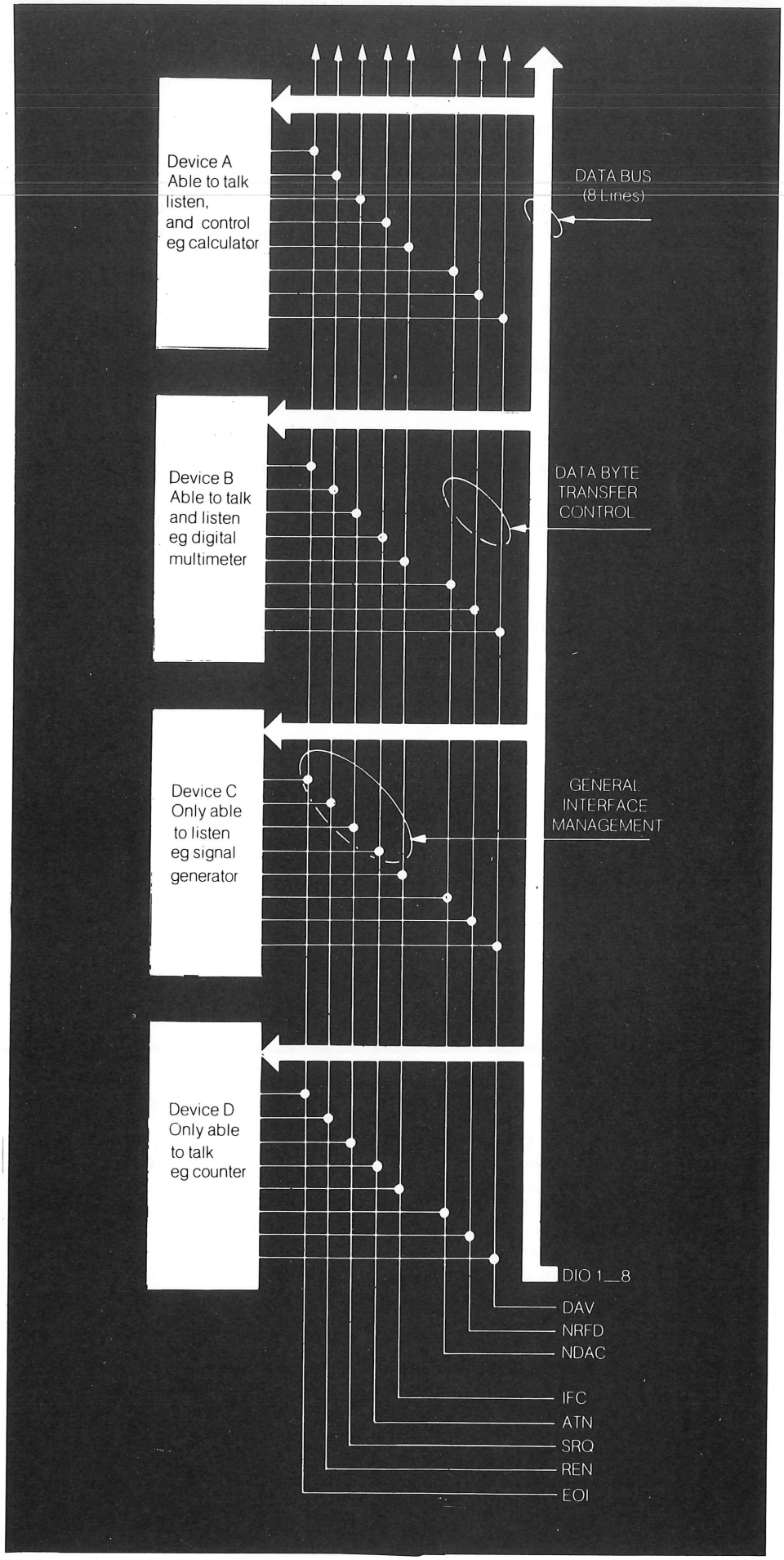
board by a microprocessor as a part of the initialization sequence.

When the controller makes the attention line true, instrument bus commands may also be sent to single or multiple GPIA's.

Information is transmitted on the instrument bus data lines under sequential control of the three handshake lines. No step in the sequence can be initiated until the previous step is completed. Information transfer can proceed as fast as the devices can respond, but no faster than the slowest device presently addressed as active. This permits several devices of different speeds to receive the same data concurrently.

The GPIA is designed to work with standard 488-bus driver IC's (MC3448's) to meet the complete electrical specifications of the IEEE 488 bus. Additionally, a powered-off instrument may be powered-on without disturbing the 488 bus. With some additional logic, the GPIA could be used with other microprocessors.

The MC68488 GPIA has been designed to interface between the MC6800 microprocessor and the complex protocol of the IEEE 488 instrument bus. Many instrument bus protocol functions are handled automatically by the GPIA and require no additional MPU action. Other functions require minimum MPU response due to a large number of internal registers conveying information on the state of the GPIA and the instrument bus.



ENGINEERING SOURCEBOOK TO BE REVISED

The staffs of **Engineering News** and **Component News** are about to begin revising the **Who, What, When, Where, How** engineering sourcebook.

If you have a copy now, we would like to have your input to help us revise the book. Give us a call (ext. 5674) or drop by 50-462. If you prefer writing out your comments, please fill in the questionnaire below.

Have you used the sourcebook often? ☐ Fewer than five times? ☐ Five to ten times? ☐ Ten to twenty times? ☐ More than twenty times?

Is the book more valuable for ☐ the contacts (names, telephone numbers and locations) or for ☐ the description of what each group does?

How would you like to see the book organized? ☐ Alphabetically by name of the group? ☐ Along organizational lines (by business unit, division and department) ☐ By each group's place in the NPI (New Product Introduction) process?

What groups would you like to see included in the sourcebook that aren't in the present version? _____

What would improve the sourcebook for you? _____

Please mail to 50-462. Thank you.

Maureen Key 60 553

Vol. 4, No. 12, 1 Nov. 1977. Editor: Burgess Laughlin, ext. 5468. Art Director: Scott Sakamoto. Published by the Technical Information Dept. (part of Test and Measurement Operations) for the benefit of the Tektronix engineering and scientific community in the Beaverton-Wilsonville area.

Why EN?

Engineering News serves two purposes. Long-range, it promotes the flow of technical information among the diverse segments of the Tektronix engineering and scientific community. Short-range, it publicizes current events (seminars, new services available, and notice of achievements by members of the technical community).

Contributing to EN

Do you have an article or paper to contribute or an announcement to make? Contact the editor on ext. 5468.

How long does it take to see an article in print? That is a function of many things (the completeness of the input, the review cycle and the timeliness of the content). But the *minimum* is three weeks for simple announcements and about five weeks for articles.

The most important step for the contributor is to put his message on paper so that the editor will have something to work with. Don't worry about problems with organization, spelling or grammer. The editor will take care of those when he puts the article into shape for you.