A Multiple-User Software Development Unit

Microcomputer software development is becoming, more and more, a team effort. And, as with any team-oriented project, there are problems of communication, keeping current on the status of each element in the project, and efficiently integrating the individual elements into a finished product.

The new Tektronix 8560 Multi-User Software Development Unit is a team-oriented design tool. Accommodating up to eight workstations simultaneously, the 8560 provides a powerful, flexible solution to software development problems. The 8560, teamed up with the Tektronix 8540 Integration Unit or the 8550 Microcomputer Development Lab[1], lets you accomplish every phase of software development, from initial design to hardware/software integration, effectively and efficiently. (The 8540 Integration Unit is discussed in an article commencing on page 9 of this issue.)

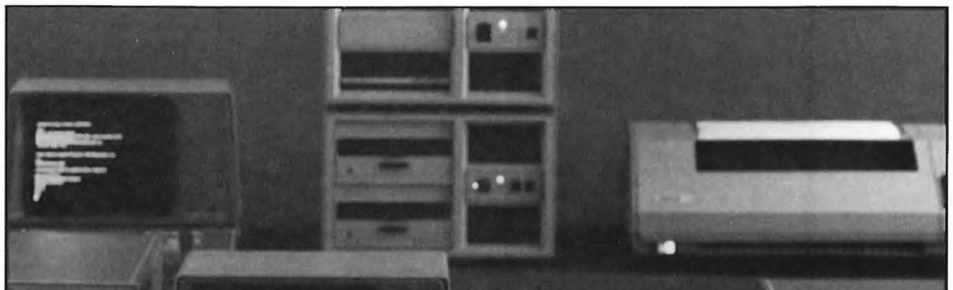


Figure 1. The Tektronix 8560 Multi-User Software Development Unit (lower left) accomodates up to eight workstations, including multiple 8540 Integration Units and line printers.

Hardware overview

The 8560 uses a multiple-processor architecture (see figure 2). The LSI 11/23 main processor executes the operating system, assemblers, compilers, editors, and utility programs. A Z80-based disk controller handles both flexible and hard disk drives. The main processor makes high-level requests and the disk controller handles the details, such as seek optimization, locating the current track and sector, reading and writing blocks of data, and checking for errors.

To relieve the main processor of the burden of handling all of the I/O traffic, an 8088-based processor is provided for each group of four I/O ports. Over 90 percent of terminal I/O is off-loaded from the main processor. The I/O ports are configurable for RS-232 operation up to 9600 baud and RS-422 at 153.6 kilobaud.

Memory in the standard system consists of 128 kilobytes of random access memory (RAM), 35.6 megabytes of hard disk storage, and one megabyte of flexible disk storage for transportable memory. Options expand RAM to 256 kilobytes, with additional hard disk capacity to be available at a later date.

An advanced operating system

The 8560 uses a powerful multitasking operating system called TNIX*. TNIX is a customized version of the popular UNIX** Version 7 operating system, optimized for microprocessor software development. UNIX is a well-established system and includes all of the tools essential for increasing programmer productivity—file management, program development, module build control, interuser communication, system maintenance, and text processing.

The command language chosen for a software development system can be a major contributing factor to software productivity. A well-designed command

[1] "A Microprocessor Development Lab with an Expandable Future," Tekscope Volume 13, Number 1, March 1981.

* TNIX is a trademark of Tektronix, Inc.
**UNIX is a trademark of Bell Laboratories, Inc.

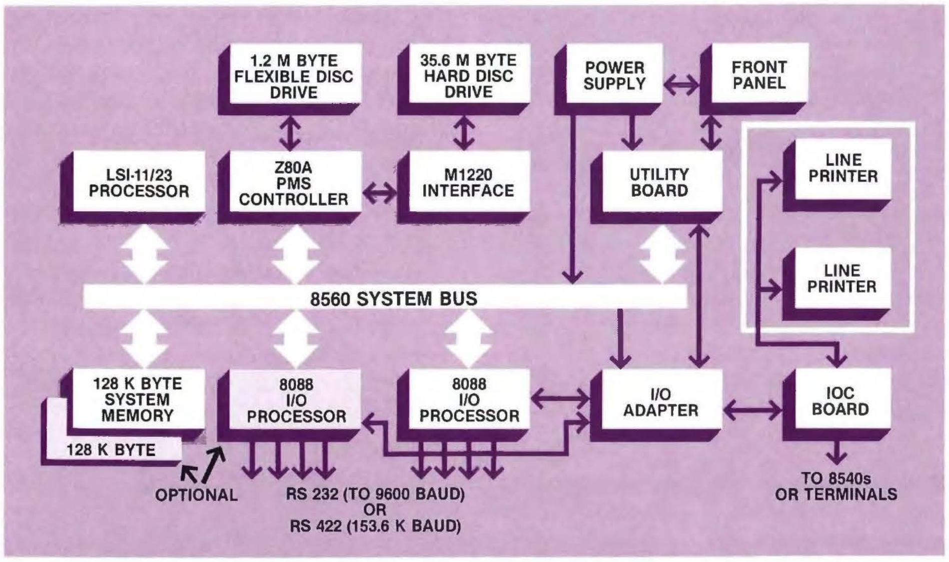


Figure 2. Block diagram of the 8560 Multi-User Software Development Unit. The multiprocessor architecture optimizes performance of the various functions and permits tasks to be carried out in background mode.

language should: be easy to learn, be flexible enough to be customized for individual needs, support powerful command files, and allow unambiguous task processing and information flow.

Learning time can be minimized through the use of a menu-driven command language. However, once the language is mastered, the requirement to use menus can impede productivity (see figure 3).

The TNIX menu-driven program (GUIDE) overcomes the necessity for users to respond to menu-driven queries. As a task proceeds, GUIDE prints out the commands being used, which helps to shorten learning time. However, one doesn't have to use GUIDE for entering commands. Once the system is learned, the user can enter commands directly. Command menus on the 8560 also can be changed, allowing the user to customize the system.

Command files allow multiple commands to be executed quickly and without typographical errors. Most systems allow parameters to be passed to a command file, thereby increasing their flexibility and ease of use. The 8560's command language also allows variables to be defined by the user, and supports structured programming commands like "**if...then... else**", "**for**" loops, "**while**", "**until**", and "**case**" statements. As an example, see figure 4. When these concepts are combined and used in a command file, users

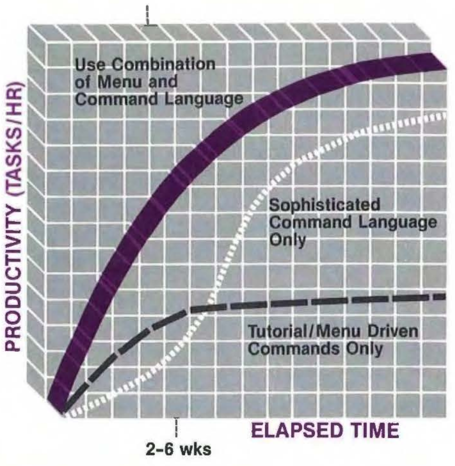
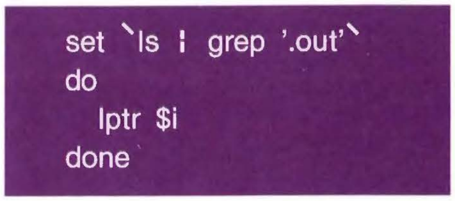


Figure 3. This graph depicts the relative productivity achievable with a system that employs menu-driven commands, with a system that does not, and with one that uses a combination of menu and command language. The 8560 uses the latter approach.



```
set `ls | grep '.out'`
do
  lptr $i
done
```

Figure 4. The above command file checks the current directory for all files with the extension ".out" and outputs them to the lineprinter.

can define or combine commands to perform tasks that would normally require large, complex programs to accomplish.

The 8560 also allows the output of one command to be passed as input to another command, without the use of temporary "holding" files. This "**pipe**" construct allows information to be easily passed through the system while being processed by a series of commands. Large amounts of data can be quickly reduced to a manageable amount and output in a format allowing quick analysis by the user. All I/O devices are handled as ordinary files, thereby eliminating the special programming usually required to handle external devices.

Multitasking increases productivity

A multitasking system increases a software designer's productivity by performing two or more tasks at the same time. With the 8560, time-consuming tasks, such as assemblies or compilation, can be executed in "**background**" mode while the user is free to perform simpler tasks, such as editing or file manipulation, in normal (foreground) mode. The 8560 supports two lineprinter ports, allowing listings to be printed in background mode (otherwise known as lineprinter spooling) while users proceed to perform other tasks. By having two lineprinter ports, users can access either high-speed/medium-quality, or low-speed/high-quality printers on the same system. Short printout jobs don't have to await completion of a lengthy printout, and printout quality does not have to be sacrificed. Background tasks also can be prioritized to minimize the impact of multitasking on other 8560 users.

File management

In a multi-user system, with source code or documentation shared by several users, file management is a critical element. The 8560 uses a hierarchical file structure that permits multilevel directories and controlled access to files. Directories allow a user to quickly locate files of interest without having to peruse the entire list of files on the disk.

Access to files is controlled by assigning each user a unique password. Three groups of users may access each file on the 8560. These include the owners of the file, members of the owner's group, and all others. Each of these groups may be assigned read, write, and execute permission.

Another important 8560 file management feature is the ability to link to files in other directories. The same information can be found under multiple directories, without requiring duplicate files. When a file is updated, the same information is available to someone accessing the file through a different directory. The need to recopy modified files is eliminated.

The capability to directly execute, copy, or link to another's file, with appropriate access control, contributes greatly to productivity in a multi-user environment.

Automated build control

In any major software development task, program management is critical. The large number of interdependent modules generated by the design team must be combined without build errors. Build errors usually result from combining wrong modules or wrong versions of the correct modules.

The 8560 uses an automated approach to the build problem. Using a command called "**make**", programs can be automatically generated from only the most up-to-date source code. "**Make**" utilizes a description file, which defines all intermodule dependencies, and associated commands to generate each module. Upon execution, "**make**" compares the modification date of an output module (i.e. an output file) with the appropriate input module (i.e. source code file). If the modification date of the input module is later than that of the output module, then the output file is recreated. This procedure greatly reduces the time needed to produce an executable program because it is no longer necessary to reassemble or recompile every module.

Interuser communication

Effective comunication between team members is essential for software development to proceed smoothly and with a minimum of problems. The 8560 takes an innovative approach to interuser communication. A command called "**mail**" allows a user to send a message to another user and store it in a private mailbox for that user. If a user has mail, the system automatically notifies the user when he or she logs into the system. The mail can be quickly viewed and then either deleted or retained for future reference. A user can also use the mail system to receive notification when a spooled printer output is completed.

Each user can determine who else is logged into the system and send a message directly to a user by executing the "**write**" command and referencing the user-identification. To avoid interruption of a critical task, each user can decide whether or not to allow direct communication. If necessary, a command requiring special authorization is provided to send a system message to all users regardless of messages being turned off.

Users can also use the 8560's documentation tools with the mail system, to generate memos and so forth.

Optional software expands capability

TNIX includes several optional software packages that allow a user to add capability as needed. An ***auxiliary utilities package,*** containing over 30 programs, enhances operating flexibility. An "**awk**" command allows the user to search a file for a selected pattern and then execute a command upon its occurrence. This is a powerful tool for reducing data from the optional Trigger Trace Analyzer. Another command, "**bc**", provides a binary calculator that lets the user enter arithmetic operations into the system and get results back with unlimited precision. This command also performs base number conversions, such as binary to octal, hexadecimal, etc. Other programs in this package provide batch editing, general preprocessing, and useful file manipulation.

The ***documentation package*** is another extremely useful option. This package greatly simplifies the many tasks involved with producing quality documentation. For example, when text is entered into a file, formatting commands are included to generate the page layout desired. The resulting file is then passed to a formatting utility that produces the document. To change the page layout, only the lines containing format commands need be changed. The formatting utility will automatically generate the revised layout. If the text is to be typeset, as for manuals production, the 8560 can generate output suitable for commercial phototypesetters.

Other time-consuming tasks, such as table generation and typesetting of mathematical equations are efficiently handled by special commands. There are commands to look for spelling errors, generate a permuted index, and so forth. In addition, special reports, manuals, business letters, specifications, and other documents can easily be created on the 8560. Users have complete control over paragraph justification, indentation, underlining, bold-facing, page headers and trailers, footnotes, and character fonts. These "active" documentation tools improve productivity substantially.

The optional ***native programming package*** contains 23 programs which provide high level and assembly language support for the 8560. A C compiler can be used to develop utilities that will enhance 8560 operation. Several supporting programs simplify and extend the use of C.

A "program beautifier" command will clarify program structure by indenting nested loops, procedures, and so forth.

Programs to perform syntax checking and program linking are also provided. In addition, an assembler is included for developing special purpose routines which can execute faster.

BASIC, another high level language, is also provided for development of a variety of applications.

The auxiliary utilities, text processing, and native programming packages are provided to allow the user to tailor the operating system to a particular need. As category C software, they carry a low priority for updating; however, these packages have been under development for several years and typically are error-free.

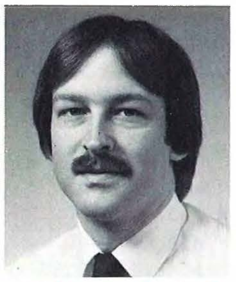
Summary

A software development system should enhance individual and team efforts in producing a reliable, quality product. It should eliminate many of the tedious programming, documenting, and manual software management tasks that design teams encounter.

The 8560 Multi-User Software Development Unit meets all of these requirements, and more. The innovative interuser communications system facilitates sharing of design information. A hierarchical file system with controlled access allows files to be organized and accessed in a manner that maximizes team productivity. Automatic build control and user-programmable command files save hours of processing time and keyboard entry. And the companion 8540 Integration Unit allows software and hardware to be integrated in a controlled manner, with effective tools for rapidly isolating and resolving any problem that may exist.

The TNIX operating system includes several commands designed to maximize efficiency when using the 8540 Integration Unit with the 8560. For example, the operating system recognizes those commands that are uniquely 8540 and passes them directly to the 8540. The system also can selectively access up to eight 8540s connected to a single 8560. The following article discusses the 8540 Integration Unit.

Acknowledgements
The following people were instrumental in the development of the 8560: Tom Clark was the 8560 project manager, with Al Baker as hardware manager, and Bob Wood as software project leader. Errol Crary was the program manager. Bob Tice, mainframes manager for the 8500 Series, also made valuable contributions to the project. Mechanical design was coordinated by Phil Sheeley. Our thanks to these and the many others who made the 8560 project a success. ■



Chuck Smith, Marketing Product Line Manager for the 8560, has been with Tek six years—most of the time as a part of the Information Display Division. He was associated with 4050 Series Marketing and was Advanced Products Manager for the Graphic Computing Systems group. Chuck received his BSCS from Michigan State University in 1973 and currently is working on his MBA at the University of Portland. In his leisure time Chuck enjoys black and white photography, softball, volleyball, and personal computer projects.

A Powerful New Tool for Integrating Microcomputer Hardware and Software

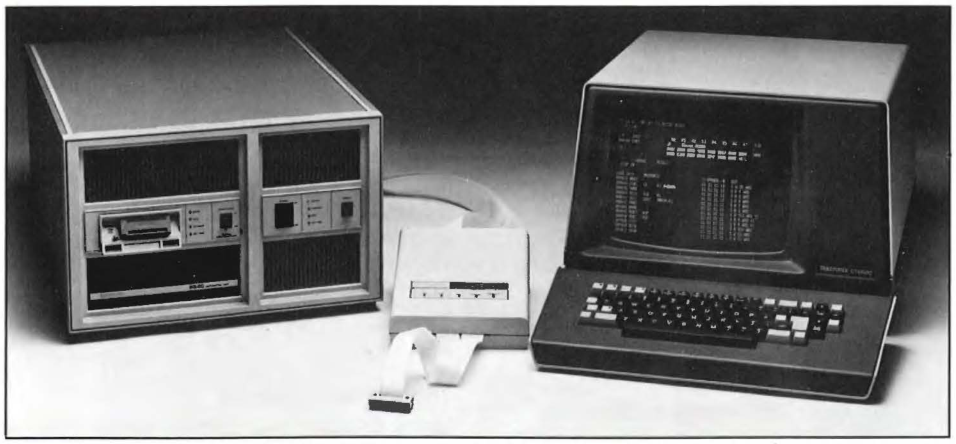


Figure 1. The 8540 Integration Unit, pictured with the CT8500 CRT Terminal and prototype control probe, provides complete coverage of the hardware/software integration process during microcomputer design.

Integrating newly-written software and prototype hardware can easily consume as much time as writing the software itself. The new Tektronix 8540 Integration Unit turns this difficult task into an orderly, efficient process.

The 8540, with an 8560 Multi-User Software Development Unit (or other host computer) and a system terminal, forms a complete microcomputer development system. The system provides a powerful set of tools for testing microcomputer programs and prototype hardware, with full-emulation and PROM programming capabilities.

Software is developed on the 8560 (which also provides mass storage and file management) and then is downloaded to the 8540 via the built-in high-speed (153.6 kilobaud) interface.

For host computers other than the 8560, an optional communication interface is available. The major interface parameters of this interface are software selectable through the 8540's operating system, so the communications package can be tailored to individual host situations.

Using commonly available RS-232-C ports and the optional communications interface package, the 8540 can be interfaced to most host computers in a matter of minutes. All communications parameters, such as parity, echo, and turnaround delay, can be set directly from the keyboard.

Three basic operating modes are available. The *object code transfer* mode permits transferring object code modules between the host and the 8540's program memory, with full error checking and recovery during the process.

Terminal mode allows the user's terminal to communicate directly with the host computer. The terminal is physically connected to the 8540; however, a single command routes the terminal directly to the host, making the 8540 transparent to the user.

Local mode provides direct communication between the terminal and the 8540, for controlling the emulation and debugging process.

Emulator support

The 8540 uses interchangeable emulator modules to allow you to configure the 8540 to your application. The 8540 supports both 8-bit and 16-bit microprocessors including those listed in Table I.

Table 1
Chips The 8540 I.U. Supports

16-bit	8-bit	
Z8001	6809	8088
Z8002	6800	8048
8086	6808	8039
68000	6802	8039-A
TMS9900	Z80A	8035
SBP9900	8080A	8021
SBP9989	8085A	8022
	8049	8041A

Using an emulator processor identical to that targeted for the prototype, the 8540 provides real-time emulation. This means that the prototype code can be executed at the specified operating speed of the target processor, while under control of the 8540's debug system.

Three modes of real-time emulation

Emulation takes place in three progressive modes that allow gradual introduction of hardware and software. In mode 0 (system mode), the software is executed on the 8540's emulator processor. Program input and output can be simulated using system resources in the console display, keyboard, or 8560 file system. Thus, you can begin debugging your program before the prototype hardware is available, or continue debugging should the hardware become inoperable.

In the next phase, Mode 1, the 8540 emulator connects directly to the prototype hardware via the prototype control probe. In this mode, the program resides in 8540 memory and can be transferred to prototype memory in sections as small as 128 bytes. This technique, called mapping, allows the program to be gradually transferred into the prototype on a step-by-step basis. The program can interact with prototype I/O, development system I/O, or both.

In Mode 2, all of the code resides in the prototype memory. This mode is used to make a final check with the actual prototype memory devices (such as ROM or PROM) in place. The control probe remains in the microprocessor socket on the prototype to provide continual debugging control during program execution.

During all three modes of emulation, prototype code execution is under control of the 8540's powerful debug software. For easy reference, key breakpoints may be entered using mnemonic labels (symbols) instead of numeric addresses. At each breakpoint, the status of all of the processor's key registers, flags, and status bits is displayed. You can also display the processor's register status and associated code execution on a cycle-by-cycle basis. Any register or memory location can be modified right from the keyboard.

The 8540 debug commands are integrated into TNIX, the 8560 Software Development Unit's operating system, so the user can control both 8540 and 8560 resources with a unified, compatible syntax. This capability also allows the 8540 to use 8560 resources, such as file I/O and data reduction, to enhance the debugging operation.

Trigger trace analyzer

Many debugging situations require detailed analysis of real-time code execution and the effect on other key points in the hardware. The trigger trace analyzer (TTA), a modular option to the 8540, provides a complete facility to acquire real-time data in both 8-bit and 16-bit processor-based systems. Up to 255 bus transactions occurring before, during, or after a specified event can be captured. An 8-channel data acquisition probe allows you to select and monitor up to eight points in the prototype hardware. For further details on the trigger trace analyzer see the article commencing on page 12.

System overview

The 8540 operating system (OS/40) is similar to DOS/50 Version 2, the operating system developed for the 8550 Microcomputer Development Lab[1]. A few commands are different, but the key difference is that commands execute much faster in the 8540 as they are stored in PROM memory rather than on disk.

[1] "A Microprocessor Development Lab with an Expandable Future," Tekscope Volume 13, Number 1, March 1981.

A functional block diagram of the 8540 is shown in figure 2. A dual-processor architecture (in a master/slave arrangement) enables the 8540 to support several different microprocessors, using the same operating system. In this configuration, the system processor serves as the master, and the emulator processor as the slave. The system processor and emulator processor have completely separate memory space so that system program and user (prototype) programs do not conflict.

The 8540 contains a 100-line system bus structure that provides most of the connections to the plug-in modules and options housed in the mainframe. The emulator controller board separates those control and signal lines that are dedicated to either the system section or the program section. Both the system processor and emulator processor share the basic bus structure, with the emulator controller serving as arbiter under the direction of the system processor.

The system processor resides on the system controller board and provides overall control of the 8540. It directs all I/O activity for the system peripherals, performs all system utility functions, and executes the debug program—controlling the emulator processor through separate debug hardware.

To allow you to configure the 8540 for your specific application, the emulator processors are designed as plug-in modules and assigned option status. An emulator option includes both hardware and software for the target microprocessor or microcomputer. The emulator processor interfaces with the prototype hardware via a prototype control probe. Advanced probe design makes the emulator processor

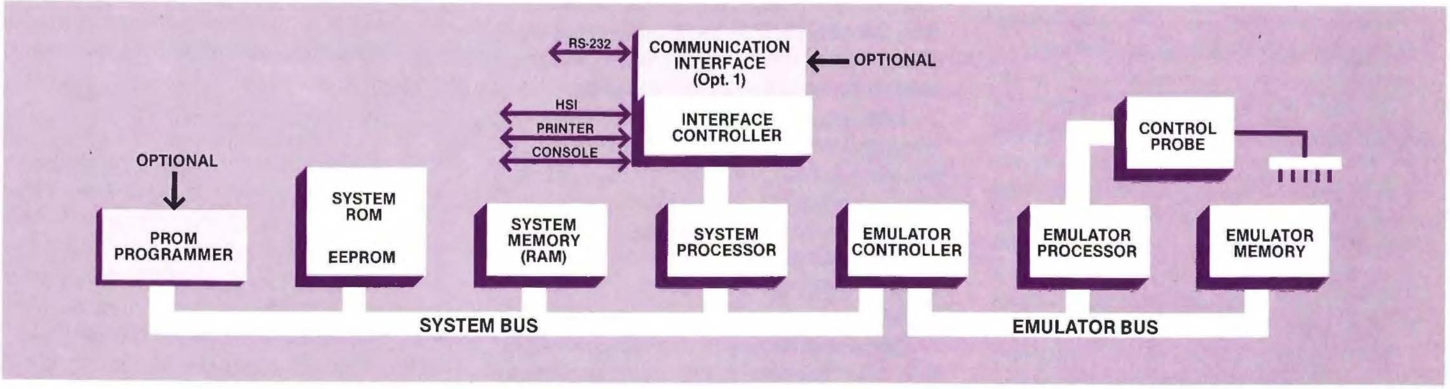


Figure 2. Functional block diagram of the 8540. A wide selection of options allow you to configure the 8540 to your design needs.

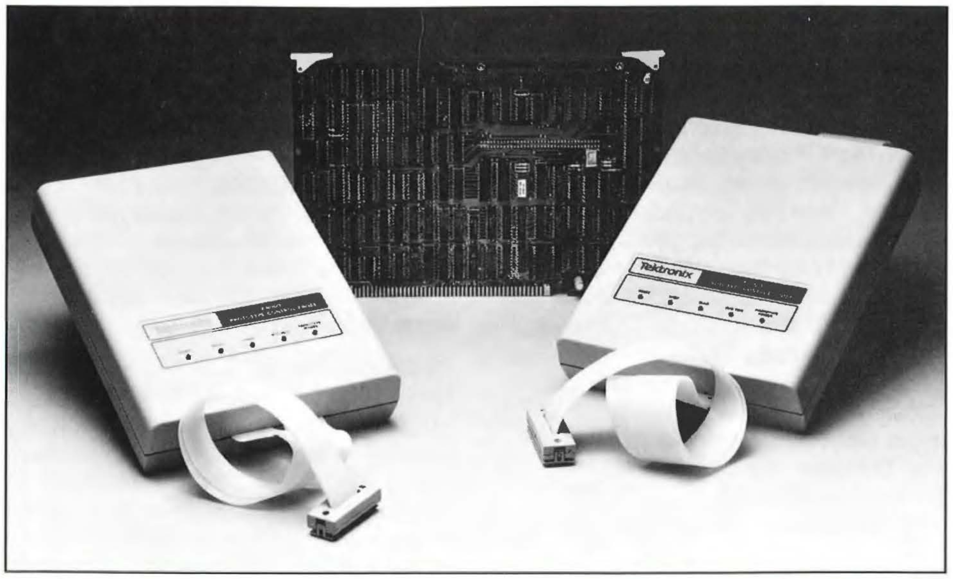


Figure 3. Emulators and prototype control probes for the 8540 feature state-of-the-art design that allows your programs to run at the full operating speed of the target microprocessor.

practically transparent to the prototype and allows prototype code to be executed at the full operating speed of the target processor, without adding wait states or stretching clock pulses (see figure 3).

Versatile memory manipulation

Memory in the 8540 consists of two major sections—system memory and program memory. System memory includes 240K bytes of ROM, which contain the operating system and software for optional equipment. Also resident on the ROM board are 4K bytes of EEPROM used for updating the operating system and for storing unique user-developed command strings. The contents of the EEPROM can be changed from the system terminal keyboard.

The operating system is loaded from system ROM and executed in the 64K-byte system RAM. User symbol table information employed in symbolic debugging is also stored in system RAM.

Program (emulator) memory consists of 32K bytes of static RAM, optionally expandable to 256K bytes. It is used for storing prototype code downloaded from the 8560 or the host computer.

The system processor has control of both system memory and program memory. As previously mentioned, in emulation mode 1, program memory can be mapped into prototype memory in 128-byte blocks, allowing orderly transfer of proven program segments to the prototype.

When working with devices such as the Z8001/Z8002, 68000, and 8086, whose addressing capabilities exceed the 8540's program memory, the Memory Allocation Controller (optional with the Z8001/2 and 68000 emulators) can be used to allocate program memory in 4K-byte blocks over an address space of up to 64M bytes.

PROM programming

Once the prototype code is debugged, it can be put into firmware using the optional PROM programmer available for the 8540. The PROM programmer consists of a controller board, front-panel assembly, and a characteristic module to adapt the programmer to whatever PROM family you require. The 8540 currently supports 2716, 2732, 8748, 8741A, and 8755 PROMs.

System diagnostics

When attempting to integrate software and prototype hardware, it is essential to know that your integration tools are working properly. The 8540 has two resident diagnostic test programs for verifying system operation.

The power-up diagnostic tests are run automatically during power-up or restart conditions. These tests verify the circuitry within the 8540 that is required to boot and transfer the operating system from ROM into the 8540's system memory.

Should a fault occur that prevents the operating system from booting or prohibits ROM-resident diagnostics from running, a program called Critical Function Monitor (CFM) is automatically entered. The CFM contains several test routines and a limited set of user commands that are entered from the system terminal. This program, in conjunction with a series of LEDs located on the system controller and system RAM boards, will usually isolate the source of trouble.

The ROM-resident diagnostics provide a means of verifying system performance, and a tool for troubleshooting in the event that a failure is detected during the running of a test. The menu-driven diagnostics are easy to use, and run automatically after being initiated by the user.

Summary

The 8540 Integration Unit is designed to help you accomplish the entire software/hardware integration process in an orderly, efficient manner. The 8540 can be easily interfaced to most host computers or any of the 8000 Series of Tektronix microcomputer development units, such as the 8560, 8550, and 8001. State-of-the-art emulators allow your programs to run at full speed, while the advanced trigger trace analyzer captures up to 255 bus transactions and select logic operations for analysis. The 8540 supports most popular 8- and 16-bit microprocessors and microcomputers.

Acknowledgments

The 8540 design team included Tom Clark as engineering manager; Dennis Stolarski, hardware project leader; Roger Crooks, software manager; and Bruce Stofer, software project leader. The hardware evaluation manager and project leader were Norm Dodge and Dave Marsh, respectively, with Dave Loney and Ray Epperson performing similar roles for software. ■

Author:
William G. Bevan
Marketing Product Manager

A New Real-Time Debugging Tool for the 8500 Series MDL

The Trigger Trace Analyzer (TTA) is a modular option for the 8540 and 8550 that allows you to monitor the buses and selected control signals in prototype hardware, while your program executes at normal speed. The TTA provides precise control of the selection of data to be stored and analyzed. Up to 255 bus transactions and logic signals from various points on the prototype can be captured and stored in the TTA's acquisition memory and displayed for analysis.

The TTA monitors 64 bits of information that you can select in any combination (using software commands) to define a trigger signal for acquiring data or for other purposes. The 64 bits of information monitored include:

- the address bus (up to 24 bits)
- the data bus (8 or 16 bits)
- the data acquisition probe (8 bits)
- the emulator-dependent bus signal interface (up to 11 bits)
- the external event qualifier (1 bit)
- counter output signals (4 bits)

All of these signals are input to an event comparator, which functions as a word recognizer (see figure 2). The output of the event comparator is ANDed with the output of a programmable general-purpose counter, to generate a trigger signal. There are four such trigger channels in the TTA. These triggers can be used independently or interactively to construct a powerful data acquisition trigger. The outputs of the four trigger channels also are available externally (via BNC connectors on the TTA interface panel) for triggering external equipment.

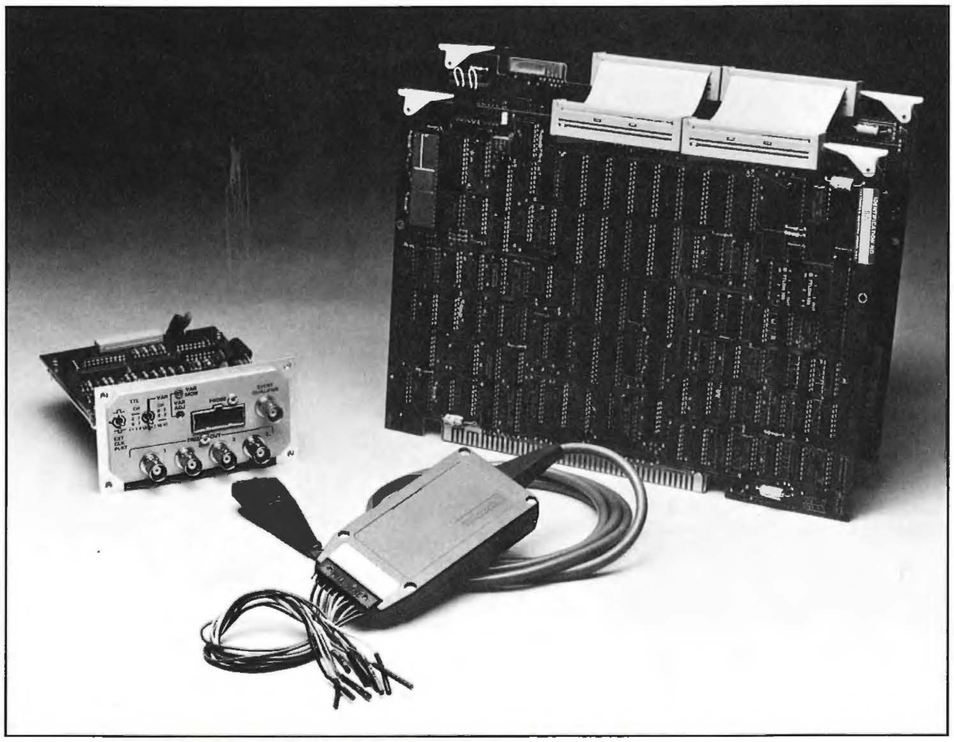


Figure 1. The Trigger Trace Analyzer option includes two plug-in modules, bus interconnecting cables, an 8-channel signal acquisition probe, and an interface panel that includes the four trigger channel outputs.

Defining an event

To better comprehend the flexibility the TTA offers in defining a trigger point, let's consider some of the event control commands used to specify which input data constitutes an event. There is a separate command for each of the event comparator input sources. There is also one command that you can use to specify all inputs—the "**eve**" command. The "**ad**" command is used to define a specific address or range of addresses as an event. The commands

```
ad 1 105E
ad 2 500 530
```

specify that event 1 occurs whenever the program accesses address 105E, and that event 2 occurs whenever the program accesses an address within the range 500 to 530, inclusive. The "**ad**" command can include a "**-n**" command modifier that defines the event as anything other than the value specified. For example,

```
ad -n 4 1000 10FF
```

defines event 4 as any address outside the range 1000 to 10FF.

Another event command, "**ctr**", defines an event as a pattern of the outputs of the four counters associated with the event comparators. The pattern can include 1's, 0's, or X's (don't cares). For example, the command

```
ctr 1 10X0
```

causes event 1 to occur when counter 1 is high and counters 2 and 4 are low.

In addition to triggering on individual events, it is possible to trigger on the occurrence of multiple events. By using the "**cons**" command, events can be linked together so that the occurrence of one event arms the comparator of the following event. All of the events within a sequence must occur on consecutive cycles of the specified type. The "**cons**" command requires you to select one bus

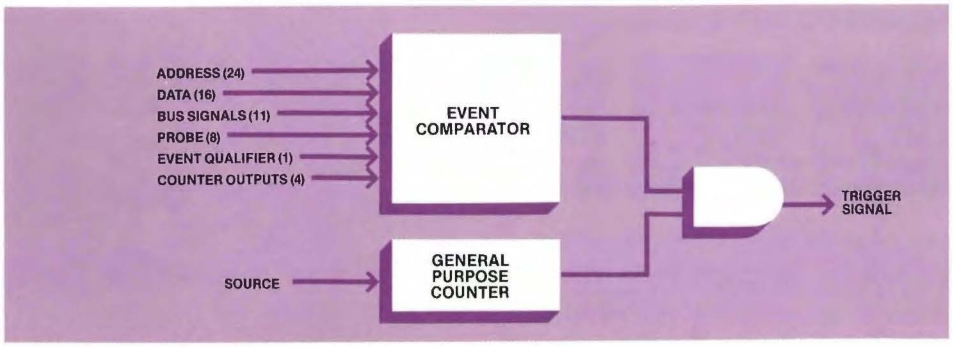


Figure 2. Each trigger channel has its own 64-input event comparator and programmable general-purpose counter. You can select from several event control commands to specify which input data constitutes an event. The four trigger channels can be linked together to provide almost unlimited trigger selections.

mode in which all of the events are considered. The bus modes are: **cyc**—all bus cycles are allowed; **fet**—only fetch cycles are considered; and **emu**—only emulator-dependent bus cycles are considered.

The general-purpose counters

We discussed, previously, the ability to specify the output of the four general-purpose counters as inputs to the event counter, to construct an event. The counters also can be used singly or together for other purposes. Let's take a look at their capability.

The "**cou**" command defines the counter operation. This command selects a value to be counted, a source that is counted, a gate signal that will enable or disable the counting process, and the kind of signal that will be output when the counting operation is completed. For example,

```
cou 2 v=4  s=ev1  o=delay
```

programs counter 2 to be asserted after the fourth occurrence of event 1.

The "**v**" or value parameter may be set anywhere between 1 and 65,536, with the value assumed to be decimal unless specified otherwise.

The source parameter, "**s**", options include counting of: clock intervals from 200 ns to 2 ms in decimal steps; occurrences of event signals for channel 1, 2, 3, or 4; occurrences of trigger signals for channel 1, 2, 3, or 4; the number of bus transactions; the number of emulator cycles; the emulator's clock signal; and the event qualifier signal. Only one of the latter three may be selected at one time. However, each of the four counters may operate on the selected signal.

A gate parameter, "**g**", places a restriction on the indicated counter and specifies those conditions during which the counter may count. Most of the conditions involve the output state of the next lower number counter, so the "**gate**" parameter is only valid for counters 2, 3, and 4.

The "**restart**" parameter allows you to have the counter reloaded with its initial "**value**" when the "**gate**" source is asserted. The options are ON and OFF.

The last counter parameter to be considered is "**output**". As the name implies, this parameter controls the output of the counter. There are five options: when "**arm**" is specified, the counter output remains high; "**disarm**"—the output remains low. When "**pulse**" is specified, the counter output is low, pulses high when counting is complete, then goes low again. In "**delay**", the output is initially low, and goes high after counting is complete. "**timeout**" produces the reverse of "**delay**".

The breakpoint command

The breakpoint command controls the effects of an event's trigger signal. For each trigger, this command can set a breakpoint, clear a breakpoint, and enable or disable the "**continue**" function.

The breakpoint, if enabled, causes a program to halt execution when an event and its associated trigger signal occur. A trace line is displayed on the system terminal and control is returned to the operating system. The "**cont**" function, if enabled, interrupts the program when the event and its trigger signal occur, and a trace is displayed. However, control is returned to the program, which continues execution at full speed.

The breakpoint parameters "**stop**" and "**cont**" can be set as a parameter in most of the event and counter commands.

The acquisition memory

Now that we have discussed how thoroughly we can define *when* data will be captured, let's look at *what* data can be captured. The acquisition memory is a 255-by-62-bit buffer. The input data available for storage includes that monitored by the event comparators with the exception of the counter outputs (see figure 3).

The "**acq**" command specifies what data is to be stored when the trigger signal occurs. "**acq all**" stores all of the most recent 255 bus transactions, which can include the eight inputs from the P6451 data acquisition probe. "**acq ev4**" stores only those transactions defined as event 4. A parameter called "**for expression source**" allows you to specify acquisitions at some point other than the end of a program. The *expression* must evaluate to some number between 1 and 65536. The *source* portion of this parameter identifies a specific kind of bus transaction, with the options available identical to the source parameters used with the "**cou**" command. An "**aftertrig 4**" parameter disables the counting of the source until trigger 4 occurs.

A typical acquisition command may appear as this:

```
acq all for 10 cyc aftertrig4
```

which would store all bus transactions until the occurrence of the tenth cycle after the occurrence of trigger 4.

The display command, "**disp**", allows you to select the portion of acquisition memory to be displayed on the system terminal. You may display all of the bus transactions stored, or display only some number of transactions you want to see.

The information displayed when you enter the display command includes an address, data, an opcode mnemonic, the states of the eight data acquisition probe signals, and symbols representing the type of bus operations that occurred.

A typical application

Now let's consider a typical application that involves using two channels of the TTA.

Problem: Provide timing for an interrupt routine located at 1000H to 1024H.

Solution: Trigger channel one is used to

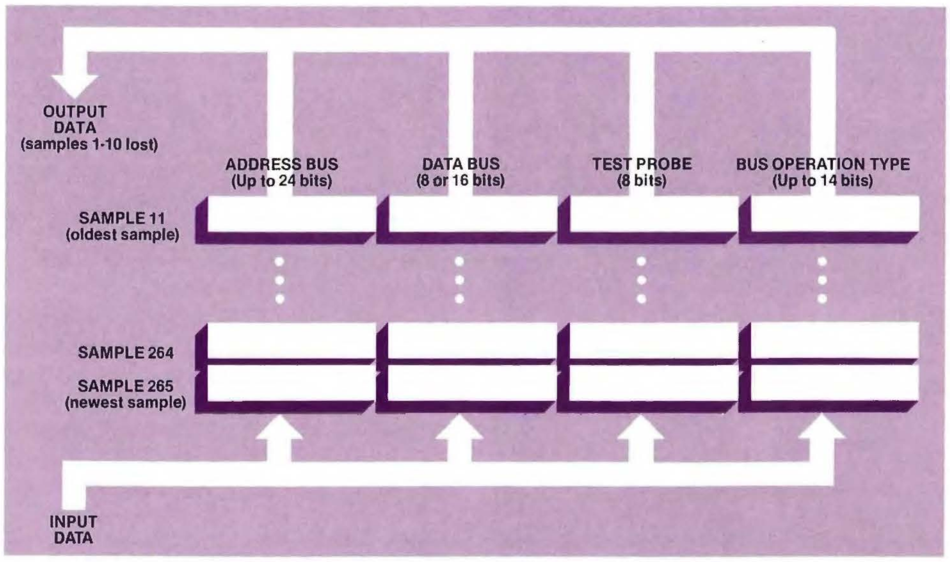

OUTPUT DATA (samples 1-10 lost)
ADDRESS BUS (Up to 24 bits)
DATA BUS (8 or 16 bits)
TEST PROBE (8 bits)
BUS OPERATION TYPE (Up to 14 bits)
SAMPLE 11 (oldest sample)
SAMPLE 264
SAMPLE 265 (newest sample)
INPUT DATA

Figure 3. The acquisition memory is similar to the buffer memory of a logic analyzer. This figure shows the contents of the acquisition memory after 265 samples of input data have been taken. Only the most recent 255 samples are retained.

detect the start of the interrupt routine and activate channel two's counter. When the interrupt routine is completed, channel two's word recognizer is used to stop the counter. The following command sequence is entered:

```
ad 1 1000
ad 2 1024
cou 2 v=0 s=200NSEC o=ARM
g=SEQH-s
```

Where: ad 1000 enters the hexadecimal value 1000 into the address portion of the channel one word recognizer.

ad 1024 enters 1024H into the channel two word recognizer

cou 2 selects the channel two counter

v=0 puts the channel two counter's initial value at zero

s=200NSEC selects 200 nanoseconds as the counting unit

o=ARM sets up EVENT 2 to cause the breakpoint

g=SEQH selects channel one's trigger output as the source that will enable the counter

-s indicates that a breakpoint will occur when the channel two trigger goes active

This command sequence produces a channel one trigger at the start of the interrupt routine, 1000H. This trigger then activates the channel two counter which begins counting in 200 nanosecond increments. Channel two's word recognizer produces a trigger when the interrupt routine is completed at 1024H. This second trigger causes a breakpoint to occur that automatically stops the counter. The resulting counter value is then read by calling up the trigger status display, which will show the counter's value at the time of the breakpoint.

Conclusion

The trigger trace analyzer option for the 8540 and 8550 is a powerful real-time debugging tool. You have almost unlimited capability to specify the trigger conditions for acquiring data while your program executes at full speed. Bus transactions, plus logic states from eight selected points in the prototype hardware, can be captured and stored for analysis. The TTA is a valuable adjunct to the 8540 and 8550 in facilitating software and hardware prototype integration. ■