TEK MICROCOMPUTER DEVELOPMENT PRODUCTS

USER NEWSLETTER
VOLUME II, ISSUE NO. 4, MARCH, 1984

USER GROUP NEWS

Tektronix®
COMMITTED TO EXCELLENCE

FOREWARD

This issue marks the completion of the second volume of "USER GROUP NEWS". Our intent is to provide useful and timely information to our customers. We need your assistance in assessing the accomplishment of our goal(s) and to set even better goals for the future.

To that end we have included a critique on the last page of this issue. We would appreciate your taking the time to answer the questions and return your answers and suggestions via the enclosed envelope. Please provide as much information as possible.

CALL FOR ARTICLES

If you have application articles or just some good ideas, we would like to print them in the *Applications Section* and/or place them in the *User Group Library Section* of "USER GROUP NEWS". A submittal form is located at the end of the *Applications Section* for your convenience.

ABOUT THE "USER GROUP LIBRARY SECTION"

We have collected a number of application programs since the last issue and we will make the programs reported in the User Group Library available through your Tektronix Applications Engineer. Each issue will report updates to the library and a separate total listing will be produced annually.

John Owens
Editor

PRODUCT INFORMATION SECTION

V-SYSTEMS: 16-BIT MICROPROCESSOR SUPPORT

Tek V-Systems

The V-Systems from Tektronix are systems designed to provide complete hardware and software support for design engineers needing the highest quality design tools available. The V-Systems are configured to integrate with an existing host computer, either an 8560/61 or a VAX* computer and include all the hardware and optionally the software required to do so. Included with the V-Systems are Tektronix's 8540 Integration Unit, 64 K-Bytes of memory, 16-bit emulation support both emulator and probe, Integrated Logic Analysis, and as an option Tektronix's unique LANguage Development System (LANDS) for high level language support. LANDS is available for either Pascal or "C" and includes a Language Directed Editor, Compiler, Integration Control System, High Level Debugger, Assembler, Linker, and for supporting VAX computers, ICOM40.

The Tektronix V-Systems are currently available to support the Motorola 68000, 68008, and 68010, as well as Intel's 8086, and 8088 microprocessors.

The V-System will also support all the other Tektronix emulators, software products, and options allowing expansion and growth to cover your future design projects.

8540 Integration Unit

Tektronix 8540 Integration Unit provides support for Tektronix's entire line of real-time emulators both 8-bit and 16-bit. Code developed on a Tektronix 8560 or 8561, Digital VAX Computer, or other Host computer can be down-loaded to the 8540's program memory, up to 256 K-bytes, for execution on the emulator processor. Execution takes place under control of powerful debug software, and the resulting data can be uploaded for powerful post processing by the host computer. For in-depth analysis of real-time code execution, the Trigger Trace Analyzer includes sophisticated triggering to capture program flow in a high speed memory buffer.

*VAX is a registered trademark of Digital Equipment Corporation.

16-Bit Emulation Family Support

Tektronix V-Systems are designed to support the Motorola 68000 series of processors and the Intel 8086 series of processors. Each of these processor types requires only one emulator and allows you to retarget to other members of the chip family by simply adding a new probe.

Support for 68000, 68008, and 68010

Tektronix support for the 68000, 68008 and 68010 is provided with the 68XXX emulator and the appropriate probe for the selected microprocessor. The 68XXX Emulator System will support real-time operation at clock frequencies up to 12.5 MHz. No wait states are inserted when accessing prototype memory or I/O. When accessing the internal emulator memory, wait states are automatically inserted at some frequencies. For the 68010, the system supports fully transparent operation in 68010 virtual memory environments. No special hardware or software is needed. (For more information, refer to the 68XXX Data Sheet.)

Support for the 8086, and 8088

Tektronix support for the 8086 and 8088 is provided with the 8086/8088 emulator and the appropriate probe for the selected microprocessor. The emulator provides total support of both Min and Max modes allowing full flexibility in the 8086 and 8088 designs. These in-circuit probes also allow support of the 8087 floating point coprocessor for both the 8086 and 8088. The 8087 is integrated with the processor in the probe and can be accessed in all emulation modes. When tracing processor execution, the actual instruction being executed is displayed, not simply instructions entering the queue. (For more information, refer to the 8086/88 Data Sheet.)

Trigger Trace Analyzer*

The Trigger Trace Analyzer is a sophisticated logic analyzer that is integrated into the 8540 to monitor and capture data surrounding real-time events during emulation. The TTA has four triggerable events for triggering purposed, each consisting of a word recognizer and a counter. These events allow the user to set complex triggering points and time sections of code. The four word recognizers are identical in capability. Each will trigger on combinations of addresses, data, and specific control signals of the emulator in use. The address and data comparators provide "equal to," "not equal to," "don't care," "ranging," and "range exclusion" triggering capability. Each counter can count triggers, time events or provide delays.

Microprocessor software support with PASCAL and "C"

Tektronix offers the first high-level microprocessor software design support that gives the programmer true high level coding support, from source code entry through prototype debug. It's called the LANguage Development Systems (LANDS), and it uses four basic tools to elevate the entire design process into high level language for the popular languages of Pascal and "C". These tools bring an unprecedented level of automation to microprocessor software design support and include: a Language Directed Editor, a Compiler with microprocessor enhancements, Integration Control System and High Level Debug.

Language Directed Editor intercepts syntax errors

Tektronix LANDS Language Directed Editor (LDE) actually understands the syntax of the high level language in use. Any syntax errors are brought to your immediate attention during the editing session so they can be easily corrected using the screen editing capability eliminating many costly recompilations.

LANDS Compiler Targets on microprocessor design

LANDS Pascal and "C" compilers are designed to give you full microprocessor coding support right down to the bit level required for microprocessor application programming. You can assign variables to specific addresses, directly access I/O ports and change bit values within a data byte. Interrupt service routines can be written and called entirely in high level language. In addition, large programs can be broken down and independently coded and debugged, which allows a modular approach to complex software development projects.

ICS automatically defines the Hardware/Software interface

The LANDS Integration Control System (ICS) is a unique design tool that reduces hardware/software interface programming to a single, simple interaction with ICS software. Through prompts supplied by the ICS, or through a regular editor, the user simply fills in a brief list of parameters that describes the hardware/software interface. Once this is done, the ICS handles all details connected with implementing the interface, including the generation of low-level code for interrupt handling and hardware initializing/reset. In addition, ICS can automatically

*Optionally includes external interface back panel and 8 lead test probe.

handle the specifics of setting up the code to run under emulation.

High-Level Debug streamlines the development cycle

Tektronix LANDS rounds out high level software development support by providing High Level Debug, which lets you perform debug operations entirely at the compiler source level while your program executes on your prototype in real-time. For instance, you can set breakpoints based on original Pascal or "C" source code statements, line numbers, or procedure/function names. You can obtain the current value of any variable by entering its name as used in the original source. Plus high level debug allows you to trace procedures, examine variable values in different levels of recursion, or modify returned values of functions. Additionally, program structures can be checked for type and variable content on line at any time.

High-level amenities for assembly coding

With Tektronix assemblers/linkers, you get features that are normally only associated with high-level coding. For example, you can create sophisticated macro statements that provide high-level coding power. The INCLUDE directive can be used to include other files containing assembler source, data types, constants and variables. Conditionals, using Boolean expressions, are available to help you control the assembly process. And Tek assemblers all share the same base, which means once you learn a Tek assembler you can move from one microprocessor to another with a minimum of learning time. All the MACRO commands, expression handling and assembler directives are the same.

ICOM40 provides an integrated environment

ICOM40 is a transparent communications environment which allows remote access to 8540's connected to a VAX* Computer with either UNIX** or VMS*** operating systems. The VAX computer operating system and the 8540 operating system can be accessed from any terminal connected to the VAX computer. In this mode, 8540 commands are entered from the keyboard as if they were VAX operating system commands. These commands are recognized by ICOM40 as 8540 commands and sent to the 8540 for processing. The 8540 processes the commands and sends responses back to ICOM40 and then on to the originating Terminal/Process.

Optional Equipment

The following equipment is available as options to the V-Systems:

Option 1 for the 8540 upgrades the standard 64KB memory card to 128KB of memory.

Option 2 adds the Memory Allocation Controller for allocating 4 K-Byte blocks of memory to any address range within the addressable limits of the microprocessor. This option is not available for the 8086, and 8088 emulators as the function is included on the emulator itself.

Option 3 adds the rear interface panel and eight lead probe for the Trigger Trace Analyzer. This allows the monitoring of up to eight external points and provides external outputs for trigger pulses generated off the trigger trace cards four super breakpoints.

Options 1A-1G selects the Pascal Language Development System with support for different systems and media.

Options 2A-2G selects the "C" Language Development System with support for different systems and media.

Options A1-A5 selects the appropriate power cord for the 8540.

The V-System is designed to allow expansion as your needs change. Any of the above options can be added to your V-System at any time. The system can be configured for today's needs and expand to meet your needs tomorrow.

Please contact your local Tektronix sales representative for more information.

Bob Ferguson, MDP Product Marketing

---

*VAX is a registered trademark of Digital Equipment Corporation.

**UNIX is a registered trademark of AT&T Bell Laboratories.

***VMS is a registered trademark of Digital Equipment Corporation.

NEW EMULATOR NEWS

80186 Emulator Now Shipping.

1750A Bus Emulator Starts Shipping This Quarter.

NSC800 Emulator Now Shipping.

78XX Series Emulator Starts Shipping This Quarter.

For more information, contact your local Tektronix sales engineer.

John Owens, Marketing Applications Manager

MUGL - VOLUME III DISK RELEASED!

The third MDP User Group Library (MUGL) disk has just been released and copies may be obtained from your local sales office. This volume contains many new application programs to run on your 8560/61. There are several more impressive 4105 Color Terminal Graphic Screens, as well as a biorhythm program which utilizes the terminal's color capabilities. We have three converter utilities; one to translate 6800 to 6809 assembly source, another which converts Motorola assembly source to Tektronix compatible source, and a utility to convert Whitesmith's object to extended tekhex. John Owens has created a handy disk backup program which create fbr backups on multiple floppies! We have also released four media utilities (including some sources) which permit reading Motorola, Intel, CP/M, and RT-11 floppies on the 8560/1. The most notable submission on this volume is an 8051 simulator package, which, when combined with our assembler, 8751 prom programmer, and 8560/1, provides a complete package for designing with the 8051 chip. For additional information on these and other new submissions, see the MUGL Abstracts section of this issue.

MUGL is provided as a service to MDP users for collecting and distributing user contributed software for all Tektronix Microprocessor Development Products. The program works like this:

- All users are encouraged to submit their creations to MUGL, MDP Marketing, PO Box 4600 MS 92-635, Beaverton, OR 97075. All submissions will be considered and are made with the understanding that the software may be placed in the public domain. Please don't send your only copy, as we are unable to return any submissions, whether accepted or not. For your convenience, a software submission form is included in this issue and on each MUGL disk volume. We must have the author's name to consider a submission, but we will withhold it if you prefer not to be contacted by anyone.

- We will generally check out the programs, but no guarantees of any kind will be made. We prefer to have the object, source, documentation, and manual page (as applicable) submitted on a floppy disk, but we'll take whatever you have. If the program warrants it, we can add the documentation.
- All accepted programs will be archived in MDP Marketing and as soon as we have enough to reasonably fill a disk, a new volume will be released.
- Annually, we will provide a master listing and index of all MUGL software. Each volume will also include a catalog listing and summary of all software included on that disk.

Here's your chance to obtain lots of neat applications software, for free! However, we need your contributions to keep the program going, so send in your programs!

Greg Saville, Software Support Manager

BINARY RIGHT TO COPY LICENSE

A new form of discounting for companies needing more than one copy of a particular piece of software, is now available for VAX and 856X language development software. For 50% of the item's list price, a user can obtain the Binary Right to Copy License, which allows the right to duplicate the selected software package from one machine to another.

When ordering the Binary Right to Copy, the user will receive a license stating permission to copy the software to ONE and ONLY ONE other 8560 or VAX. You will not receive software, manuals, or warranty. The user must do the duplication himself. Manuals can be purchased separately. Since we warrant the first copy, there is no reason to warrant the additional copy. However, Software Subscription Service can be obtained on the software for the new machine. Binary Right to Copy is available for all assemblers, compilers, and LANDS packages, as well as ICOM40 and COLORKEY+ for the VAX. It is not available for the ACE editor, the auxiliary packages for the 856X, or 8550 software.

This new method of discounting multiple copies of software is available at any time, not just at the time of original purchase. If you need more copies of software you already own, you can simply buy the Binary Right to Copy License.

Marilyn Hanson, MDP Product Marketing

ACE CONFIG FILES FOR Z-29 AND HP-2645 TERMINALS

We have recently received ACE configuration files for the Zenith Z-29 and Hewlett Packard HP2645 terminals. Copies (hardcopy listing only) are available from me. Thanks to Gregory Greer and Robert Stone of the NASA Goddard Space Flight Center for offering to share their efforts.

Greg Saville, Software Support Manager

COLORKEY+ IS NOW AVAILABLE FOR THE VAX

The Colorkey+ user Interface is now available for VAX 730/750/780/782 minicomputers with UNIX or VMS operating systems.

Colorkey+ is a special color user interface for the VAX/8540 development system environment. Its advanced use of color coding and other graphics features provide you with an exceptionally fast learning curve and a valuable reference tool.

COLOR

Color has a strong proven track record for bringing increased productivity to computer-based user interfaces. The benefits of coding displayed information in color fall into three main groups. First, color allows quick discrimination between different types of displayed data. Second, color reduces the chances of error due to misinterpretation. Third, color reduces user fatigue.

KEY

Colorkey+ uses a "soft key" format that simplifies command entry and keeps the number of required key strokes to an absolute mininum. The user is presented with a set of "current key labels" across the bottom of the display that are associated with 8 "soft" keys on the keyboard.

PLUS

A good user interface lets you start work on a design project with no advance knowledge of the microcomputer design system. A better interface teaches you the specifics of the system while you interact with it. The best interface allows you to freely intermix interface commands with actual system commands. This way you benefit from the tutorial aspects of the user interface, but can opt for the efficiency of system commands at any time you wish.

Colorkey+ gives you all these capabilities. For instance, if you are using the user interface keys for moving through the file system, Colorkey+ will display the actual system commands that would perform those operations. As you become more familiar with the file system command set, you can bypass Colorkey+ at any time to interact directly with the system, and then use Colorkey+ when you need to.

Colorkey+ also lets you go back and edit previous command sequences made during the current work session. Suppose you have entered a sequence of commands and wish to use the same sequence again but with different parameters. All you have to do is scroll through the command history and edit in the new parameters. You can then execute the original command sequence with the new parameters.

Terminal Requirements

Colorkey+ has been optimized for use with the Tektronix 4105 Color Graphics Terminal. However, it can also be used with the Digital VT100 series terminals, the Tektronix CT8500 and many other ANSI standard terminals.

Diane Wortsmann, MDP Product Marketing

8-BIT HIGH-LEVEL LANGUAGE

Tektronix now offers a high-level language for 8080/8085 and Z80/NSC800 on the 856X Development System. Modular Development Language for Micros, MDL/u, designed specifically for microprocessor-based product development, is now available on the 856X .

It is a given fact that programming in high-level language is faster than writing code in assembly language. But, quite often memory or execution time constraints do not allow the luxury of using a high-level language. However, testing your algorithm with a program written in HLL can save you considerable development time. The program can then be scrutinized for time-critical areas or memory constraints, and portions or perhaps all of the code may be re-written in assembly language. Once you know that your concept will achieve the desired results, portions needing recoding will proceed much faster and with fewer errors. The HLL text becomes a basis of specifying the functionality of the language module.

In addition, there may be times when the coding is not complete but the prototype needs to be tested so hardware development can continue. A quick program can be written in HLL to test your prototype so that development of software and hardware can proceed simultaneously.

A method of automated module testing can also be set up with the I/O capabilities offered, particularly in MDL/u. With the I/O simulation/substitution allowed, modules of code can be extensively tested individually or collectively by replacing prototype I/O with predetermined stimulus data. The results of execution can be stored for comparison to expected results.

MDL/u Programming Language with Rational pre-processor offers you an inexpensive, easy-to-learn high-level-language. MDL/u is a language based on ANSII-Standard BASIC with extensions particularly targeted to microprocessor development. Rational is a pre-processor for the BASIC compiler which gives you program control structures similar to the C Programming Language. Together they give you an efficient method of programming for the 8080/8085 and Z80/NSC800 microprocessors. Tektronix now offers two MDL/u products on the 856X development system: one that generates 8080 assembly code and one that generates Z80 assembly code.

See your local Tektronix representative for ordering information.

Marilyn Hanson, MDP Product Marketing

- -

TTA HIGH LEVEL PROGRAMMING

TTA HLP (High Level Programming) software is now being shipped with Version 2 TTA's (8540F03, 8540 Opt 03) at no extra charge. HLP is a new command language that offers a high level "problem-oriented" approach to programming the TTA. When installed on the 8560, it provides an easy way to use the TTA to its fullest capability.

For example, to count the time between two events:

OLD WAY

```
eve 1 a=02 b=f
eve 2 a=08 b=f
ctr 3 11xx
cou 1 s=ev1 v=1 o=delay
cou 2 s=ev2 v=1 o=timeout
cou 3 s=2usec v=0 g=selt
```

HLP

```
let start = a = 2 b = f
let end = a = 8 b = f
count 2usec after start until end
```

NOTE:

TTA HLP is NOT COMPATIBLE WITH OLD TTA's

This is a new feature offered for new TTA's only (serial number B030000 or higher). HLP runs on the 8560 only.

Roger Crooks, MDP Product Marketing

ICOM40 SOURCE

ICOM40 Source options do not contain ICOM40 binary. The source is meant for users who must modify ICOM40 to make it run on their machine. Users who need source and binary must buy BOTH, or buy the source and recompile it to get the binary.

Diane Wortsmann, MDP Product Marketing

MANUALS LIST

Manuals are listed in the following categories:

- 8560 Users Manuals
- 8550 Users Manuals: DOS/50 V.2
- 8540 Users Manuals
- 8500 MDL Series B Assembler Users Manuals
- 8500 MDL Series Emulator Specifics Manuals
- Other 8500 Series Users Manuals
- 8550 Users Manuals: DOS/50 V.1
- 8500 MDL Series A Assembler Users Manuals
- Host Software
- 8500 Series Installation Manuals

8560 Users Manuals	PART NUMBER	
8560	MUSDU Class C Text Processing Package Users Mnl.	070-4272-00
8560	MUSDU Class C Native Programming Pkg Users Mnl.	070-4271-00
8560	MUSDU Class C Auxiliary Utilities Pkg.Users Mnl.	070-4270-00
8560	MUSDU ACE Reference Card	070-4190-00
8560	OtSDU ACE Users Booklet (version 2)	070-4468-00
8560	MUSDU ACE Screen Editor Users Booklet (Version 3) 4105 Edition	070-4725-00
8560	MUSDU Language-Directed Editor Users Manual	070-4253-00
8560	MUSDU Language-Directed Editor Users Manual 4105M Edition	070-4728-00
8560	MUSDU Language-Directed Editor CT8500-Edition Reference Card	070-4249-00
8560	MUSDU Language-Directed Editor Reference Card 4105 M Edition	070-4727-00
8560	MUSDU Language-Directed Editor Template for CT8500 Keyboard (package of 4 templates)	070-4622-00
8560	MUSDU Pascal Debug 8086/8088 Reference Card	070-4283-00
8560	MUSDU Pascal Debug Z8001/Z8002 Reference Card	070-4464-00
8560	MUSDU Pascal 68000 Compiler Users Manual	070-3875-00
8560	MUSDU Pascal Debug 68000 Reference Card	070-4465-01
8560	MUSDU 8086/8088 Pascal Language Ref. Manual	070-4378-00
8560	MUSDU 8086/8088 Pascal Compiler Users Manual	070-3878-00
8560	MUSDU Z8001/Z8002 Pascal Compiler Users Manual	070-3876-00
8560	MUSDU Pascal Compiler 68000/68010 Users Manual	070-3875-01
8503	Disk Expansion Unit Users Manual	070-4463-00
8560	MUSDU Intel COMM Users Manual	070-4481-00
8560	MUSDU User Information Instruction Sheet	070-4679-00

8561	MUSDU 4-User Upgrade Instruction Sheet	070-1623-00
8561	MUSDU 8-User Upgrade Instruction Sheet	070-1438-00
8561	MUSDU 4-User Upgrade Option User Information	070-4764-00
8561	MUSDU 8-User Upgrade Option User Information	070-4770-00
8560	MUSDU Digital Design Lab Users Manual	070-4550-00
8560	MUSDU UNICOM Users Manual	070-4536-00
8560	MUSDU Magnetic Tape Interface Users Manual	070-4586-00

8550 USERS MANUALS: DOS/50 V.2		PART NUMBER
8550	Microcomputer Dvlpt Lab Users Manual: DOS/50 V2	070-3936-00
8550	Microcomputer Dvlpt Lab Sys Ref Bklt: DOS/50 V2	070-3937-00
8550	MDL System Users Manual DOS/50 Version 2.1A	070-4553-01
8550	Microcomputer Development Lab GUIDE Instl Manual	070-4402-00
8550	Microcomputer Development Lab Editor V4.X Manual	070-3571-00
8550	Microcomputer Dvlpt Lab Editor V4.X Ref Card	070-3572-00
8550-to-8540	Conversion Instruction Sheet	070-4437-02
RTPA	Users Mnl: DOS/50 V2	070-3922-00
8550	MDL ACE Users Booklet (version 2)	070-4365-00
8550	MDL Intel COMM Users Manual	070-4480-00
8550	MDL Pascal 8086/8088 Compiler Users Manual	070-3877-00
8550	MDL Pascal 8080/85 Compiler Users Manual V4.0	070-4336-00
8550	MDL Pascal 8080/8085 Compiler Version 4.02	070-4591-00
8300H01/02	MDL/u Compiler Users Manual	070-3601-00
8300H01/02	MDL/u Compiler Reference Booklet	070-3602-00
8080A	MDL/u Compiler Specifics	070-3598-00
6800/02	MDL/u Compiler Specifics	070-3599-00
8086	Prototype Debug Specifics	070-3603-00
8086	Prototype Debug Reference Card	070-3604-00
8550	MDL RT11/50 Users Manual: Volume 1, System	070-4409-00
8550	MDL RT11/50 Users Manual: Volume 2, System	070-4410-00
8550	MDL RT11/50 Users Manual: Volume 3, System	070-4411-00
8550	MDL RT11/50 Users Manual: Volume 4, FORTRAN IV	070-4412-00
8550	MDL RT11/50 Installation Sheet	070-4404-00

8540 USERS MANUALS	PART NUMBER	
8540	Integration Unit System Users Manual OS/40	070-3939-00
8540	Integration Unit Reference Booklet OS/40	070-3992-00
8540	Integration Unit System Users Manual OS/40	070-4552-01
8540	Integration Unit Intel COMM Users Manual	070-4479-00

8500 MDL SERIES B ASSEMBLER USERS MANUALS		PART NUMBER
8500	MDL Series B Assembler Core Users Manual	070-3856-01
8550	Host Specifics	070-3943-01
8560	Host Specifics	070-3944-01
Z80A	Assembler Specifics	070-3949-00
Z80A	Assembler Reference Card (8560)	070-3950-00
Z8001/2	Assembler Specifics	070-3854-00
Z8001/2	Assembler Reference Card (8550)	070-3973-00
Z8001/2	Assembler Reference Booklet (8560)	070-3958-00
1802	Assembler Specifics	070-4507-00
1802	Assembler Reference Booklet (8560)	070-4506-00
6800/01/02	Assembler Specifics	070-3947-00
6800/01/02	Assembler Reference Card (8560)	070-3948-00
6809	Assembler Specifics	070-3960-00
6809	Assembler Reference Card (8550)	070-4369-00
6809	Assembler Reference Card (8560)	070-3961-00
68000	Assembler Specifics Users Manual for	070-3855-01
68000	Assembler Reference Booklet (8550)	070-3974-00
68000	Assembler Reference Booklet (8560)	070-3959-00
8048family	Assembler Specifics	070-3955-00
8048family	Assembler Ref Card (8560)	070-3956-00
8051	Assembler Specifics	070-4321-00
8051	Assembler Reference Card (8550)	070-4364-00
8051	Assembler Reference Card (8560)	070-4320-00
8080A/8085A	Assembler Specifics	070-3945-00
8080A/8085A	Assembler Reference Card (8560)	070-3946-00
8086/8088	Assembler Reference Booklet (8550)	070-3852-00
8086/8088	Assembler Reference Booklet (8560)	070-3957-00
8086/80186	Specifics Users Manual for B Series	070-3853-01

9900/9989	Assembler Specifics	070-4373-00
9900/9989	Assembler Reference Card (8560)	070-4368-00
9900/9989	Assembler Reference Card (8550)	070-4367-00

8500 MDL SERIES EMULATOR SPECIFICS USERS MANUALS		PART NUMBER
Z80A	Emulator Specifics	070-3964-01
Z8001/2	Emulator Specifics	070-3969-00
6800/6802	Emulator Specifics	070-3963-00
6801/68120	Emulator Specifics	070-3991-00
6809	Emulator Specifics	070-3971-00
68000	Emulator Specifics	070-3970-01
68000	Emulator Processor 8--10 MHz Part No.	070-4798-00
8048family	Emulator Specifics	070-3967-01
8080A	Emulator Specifics	070-3962-00
8085A	Emulator Specifics	070-3966-00
8086/87/88	Emulator Specifics	070-3968-01
9900/9989	Emulator Specifics	070-3965-00
TMS9900	Emulator Specifics	070-4397-00
3870/3872/F8	Emulator Specifics	070-4438-00

OTHER 8500 SERIES USERS MANUALS		PART NUMBER
8500 MDL Series	ACE Screen Editor Reference Manual	070-4726-00
8500 MDL Series	ACE Users Manual (Version 1)	070-3573-01
8500 MDL Series	ACE Reference Manual (Version 2)	070-4361-00
8500 MDL Series	ACE Users Reference Card (Version 1)	070-3574-00
8500 MDL Series	Pascal Debug Users Manual	070-4281-00
8500 MDL Series	Pascal Language Reference Manual	070-3880-00
8500 MDL Series	2716/2732 PROM Programmer Specifics	070-3868-00
8500 MDL Series	2764 PROM Programmer Specifics Users	070-4375-00
8500 MDL Series	8748/etc. PROM Programmer Specifics	070-3869-00
8500 MDL Series	8751 PROM Programmer Specifics Users	070-4414-00
8500 MDL Series	68701 PROM Programmer Specifics Users	070-4413-00
8500 MDL Series	Trigger Trace Analyzer Users Manual	070-3760-01
8500 MDL Series	TTA High-Level Programming Language	070-4947-00
8500 MDL Series	Extended Hex Interface Instructions	070-4478-00
CT8500	Video Display Terminal Operator's Manual	070-3737-00

8550 USERS MANUALS: DOS/50 V.1		PART NUMBER
8550	Microcomputer Developt Lab System Users Manual	070-3457-00
8080A	Emulator Specifics	070-3562-00
6800/02	Emulator Specifics	070-3563-00
Z80A	Emulator Specifics	070-3564-00
TMS9900	Emulator Specifics	070-3565-00
8085A	Emulator Specifics	070-3566-00
3870/3872/F8	Emulator Specifics	070-3567-00
1802	Emulator Specifics	070-3568-00
8048family	Emulator Specifics	070-3569-00
6809	Emulator Specifics	070-3851-00
8550	Microcomputer Dvlpt Lab Sys Ref Bklt: DOS/50 V1	070-3458-00
RTPA	Users Manual: DOS/50 V1	070-2785-01

8500 MDL SERIES	A ASSEMBLER USERS MANUALS	PART NUMBER
Assembler	Users Manual	070-3575-01
8080A/8085A	Assembler Specifics	070-3576-00
8080A/8085A	Assembler Reference Card	070-3577-00
6800/01/02	Assembler Specifics	070-3578-00
6800/01/02	Assembler Reference Card	070-3579-00
Z80A	Assembler Specifics	070-3580-01
Z80A	Assembler Reference Card	070-3581-00
TMS9900	Assembler Specifics	070-3582-00
TMS9900	Assembler Reference Card	070-3583-00
3870/3872/F8	Assembler Specifics	070-3584-00
3870/3872/F8	Assembler Reference Card	070-3585-00
1802	Assembler Specifics	070-3586-00
1802	Assembler Reference Card	070-3587-00
8048family	Assembler Specifics	070-3588-00
8048family	Assembler Reference Card	070-3589-00
8086/8088	Assembler Specifics	070-3592-00

8086/8088	Assembler Reference Card	070-3593-00
Z8000	Assembler Specifics	070-3594-00
Z8000	Assembler Reference Card	070-3595-00
68000	Assembler Specifics	070-3596-00
68000	Assembler Reference Card	070-3597-00
6809	Assembler Specifics	070-3692-00
6809	Assembler Reference Card	070-3693-00

HOST SOFTWARE		PART NUMBER
ICOM40	VAX/UNIX Integrated Communications System	070-4543-00
ICOM40	Integrated Communications System Users	070-4742-00
VAX/UNIX	Host Specifics Users Manual for B Series	070-4741-00
VAX/VMS Host	Assembler Specifics Users Manual for	070-4740-00
Pascal Compiler	68000/68010 Usr. Mnl. for VAX/UNIX	070-4857-00
Pascal Debug	68000/68010 Usr. Mnl. for VAX/UNIX Host	070-4852-00
Pascal LDE	Users Manual for	070-4855-00
Pascal 68000/68010	Debug Usr. Mnl. for VAX/VMS Host	070-4852-00
Pascal LDE	Users Manual for	070-4854-00

8500 SERIES SERVICE MANUALS		PART NUMBER
8301	Microprocessor Development Unit Service Manual	070-2976-01
8301/8540	Conversion Instruction Sheet	070-4447-00
8501	Data Management Unit Service Manual	070-2975-00
8540	Integration Unit Service Manual	070-3920-00
8540	Integration Unit EEPROM Patch Information	070-4287-04
8560	MUSDU Service Manual	070-3900-00
8503	Disk Expansion Unit Service Manual	070-4356-00
8560	MUSDU GPIB Interface Service Manual	070-4475-00
DataTrak	8" Flexible Disc Drive Service Manual	070-4253-00
RTPA	Service Manual	070-2724-01
TTA	Service Manual	070-3762-00
PROM	Programmer Controller Service Manual	070-3757-00
2716/2732	PROM Programmer Module Service Manual	070-3758-00
2764	PROM Programmer Module Service Manual	070-4350-00
8751	PROM Programmer Module Service Manual	070-4352-00
8748family	PROM Programmer Module Service Manual	070-3759-00
68701	PROM Programmer Service Manual	070-4351-00
64K/128K	Program Memory Service Manual	070-3924-00
8500	Modular MDL Series 8086-to-8086/8087	070-4561-00
8500	Modular MDL Series 8088-to-8088/8087	070-4562-00
8500	MDL Series 68000 Emulator Processor 8 to 10 MHz	070-4772-00
Z80A	Emulator Processor Service Manual	070-2715-01
Z8001/2	Emulator Processor Service Manual	070-3772-00
1802	Emulator Processor Service Manual	070-2631-01
3870/3872/F8	Emulator Processor Service Manual	070-2634-01
6500/1	Emulator Processor Service Manual	070-2887-00
68xx	Emulator Processor Service Manual	070-3768-00
68xx	Emulator Processor Field Modification Sheet	070-4458-00
6800/6802	Emulator Processor Service Manual	070-2354-03
6801/68120	Prototype Control Probe Service Manual	070-3864-00
6809	Prototype Control Probe Service Manual	070-3867-00
6809E	Prototype Control Probe Service Supplement	070-4461-00
68000	Emulator Processor Service Manual	070-3770-00
8500	MDL Series 68008 Prototype Control Probe	070-4690-00
8500	MDL Series 68000-A and 68010 Prototype Control	070-4692-00
8048family	Emulator Processor Service Manl	070-2632-01
8080A	Emulator Processor Service Manual	070-2353-03
8085A	Emulator Processor Service Manual	070-2716-01
8086/8088	Emulator Processor Service Manual	070-3774-01
9900	Emulator Processor Service Manual	070-2712-01
9900/9989	Emulator Processor Service Manual	070-4157-00

8500 SERIES INSTALLATION MANUALS		PART NUMBER
8540	Integration Unit Installation Guide	070-3921-00
8550	Microcomputer Development Lab Installation Guide	070-2974-01
8560	MUSDU Installation Guide	070-3899-00
8560	Series MUSDU TNIX Version 2 Installation	070-4496-02
8560/8561	MUSDU Installation Guide	070-4627-00
8503	Disk Expansion Unit Installation Manual	070-4355-00

8560	GPIB Interface Installation Service Manual	070-4476-00
Z80A	Emulator Processor/PCP Installation Manual	070-3665-01
Z8001/2	Emulator Processor/PCP Installation Mnl.	070-3773-00
1802	Emulator Processor/PCP Installation Manual	070-3667-00
3870/3872/F8	Emulator Processor/PCP Installn. Manual	070-3669-00
68xx	Emulator Processor Installation Manual	070-3769-00
68xxx	Emulator Processor with 68000-A/68008/68010	070-4691-00
6800/02	Emulator Processor/PCP Installation Manual	070-3663-00
6801/68120	Prototype Control Probe Installation Manl	070-3865-00
6809	Prototype Control Probe Installation Manual	070-3866-00
6809E	Prototype Control Probe Installatn Svc Suppl	070-4462-00
68000	Emulator Processor/PCP Installation Manual	070-3771-01
68000	Emulator Processor Emulator Board EMU2 Upgrade	070-4833-00
8048family	Emul. Proc./PCP Installn. Manual	070-3671-00
8080A	Emulator Processor/PCP Installation Manual	070-3664-00
8085A	Emulator Processor/PCP Installation Manual	070-3666-00
8086/8088	Emulator Processor/PCP Installation Manual	070-3775-00
9900/9989	Emulator Processor/PCP Installation Manual	070-4158-00
TMS9900	Emulator Processor/PCP Installation Manual	070-3670-00
TTA	Installation Manual	070-3761-00
PROM Programmer	Controller Installation Manual	070-3903-00
64K/128K	Program Memory Installation Manual	070-3923-00
MAC	Installation Manual	070-3925-00

VST LINKER ON THE 8560

The current linker on the 8560/61, Version 2, is capable of linking programs with up to 1200 global symbols. Although this is sufficient for a majority of programs, users writing extremely large programs, especially for chips such as the 68000 and 8086, find that the linker runs out of symbol table space, resulting in Link Error 102: Memory Overflow. A new version of the linker will soon be available which will accept an unlimited number of global symbols. The "virtual symbol table" linker, Version 3, will be distributed with the next version of TNIX, which is scheduled for availability in April. To accommodate users who have reached the limit of the current linker, preliminary copies of the linker are available through the sales offices.

To use the virtual symbol table capacity of the linker, include a -b on the invocation line. Similar to the -b on version 2 ofthe assembler, the vst link is considerably slower, so should be used only when Memory Overflow occurs.

The virtual symbol table version of the linker is available only on the 8560/61 Series. It will not be available on the 8550 and there is no need for a new version on the VAX, as the current version takes advantage of the virtual memory.

Marilyn Hanson, MDP Product Marketing

APPLICATIONS SECTION

BINARY TRANSFERS VIA CU

There are times when it may be desirable to transfer binary files between systems. The unicom utility, cu, provides a convenient mechanism for transferring ascii text files between machines, but is not designed to work with raw binary files. The two following programs provide a quick-and-dirty method of making these kinds of transfers. The program, en.c, can be used to encode a binary image into an ascii representation suitable for transfer with cu's ~ %put or ~ %take command. Once transferred, the complimentary program, un.c, is used to uncode the ascii data back to its original form. These simple utilities have no error checking, they were only intended to provide a quick, easy way to transport a binary image. They are just as reliable as cu normally is for any other ascii transfer. We have used them here in MDP Marketing regularly with no problems. You can always send the file across, bring it back, and "cmp" the original with the doubly transferred copy for verification. These utilities are included in the MDP User's Group Library (MUGL Disk Volume III) if you don't want to type them in.

File: en.c

```
/* en - encode any binary file to an ascii representation suitable
for "~ %take"'ing with cu. Use the complimentary utility "un" to
convert en's output back to its original binary representation.
uses std in & out, example usage: en <binaryin >asciirepout
gas - 11/17/83 */
#include <stdio.h>
main()
{
int c,i=0;
while((c=getchar()) != EOF)
        {
        putchar(((c & 0x0f0) >> 4) + 'A');
```

```
        putchar((c & 0x0f) + 'A');
        i++;
        if ((i % 16) == 0) putchar('\n');
        }
putchar('\n');
}
```

File: un.c

```
/* un - uncode ascii representation file "~ %take"'ed with cu back
to its original binary form. Use the complimentary utility "en"
to ascii encode a binary file for "~ %take"'ing with cu.
Uses std in & out, example usage: un <asciirepin >binaryout
gas - 11/17/83 */
#include <stdio.h>
main()
{
int c;

while ((c=getchar()) != EOF)
        {
        if (c != '\n')
      putchar((((c - 'A') << 4) & 0x0f0) | (getchar(c) - 'A'));
        }
}
```

Greg Saville, Software Support Manager

- -

MULTI DISK FBR AND INCREMENTAL BACKUP

The following files allow a user to perform periodic backup of all files and directories under a specified path. The command first looks for a file ".UPDATE" in the directory defined by the path or in the current directory if no path is provided. If the ".UPDATE" file does not exist all files and directories will be archived on sequential "fbr" command created disks, and the ".UPDATE" file will be created. If the ".UPDATE" file exists, all files and directories newer then the date of the ".UPDATE" file will be archived. When the files have been archived, the date attribute of the ".UPDATE" file is updated to the current date and time.

If more then one disk is used, the user is prompted to change the volume as needed.

METHOD

The file "bkup" shown below, first eases users' fears by sending "Gathering data"; then tests for the existence of ".UPDATE" in the target directory. It might be useful to note the use of the expression "**1{1+/}**.UPDATE". This expression resolves to ".UPDATE" if no parameters are used with the command, and it resolves to "path.dir/.UPDATE" if "path.dir" is the path description passed to the command. After the test a command is used to generate a recursive list of all files and directories contained in the selected directory along with their size and path. The sed command uses the sed script "sedf" shown below. The sed script is used to flag information as to type and remove unnecessary data and lines. The awk command uses the awk script "awkf" shown below. The awk script takes the output of the sed command and translates the data into a command file. The awk program creates a command file that will not exceed the limit of characters on a command line, the number of files that a disk can contain, or the total number of blocks that a disk can contain. The command file also contains the prompting sequences for disk changes. After creating the command file in "/usr/tmp" the mode of the file is changed to permit execution, then the file is executed. After completion of the created command file the file is removed and the attributes of the ".UPDATE" file is changed to current date and time. Prior to exiting the "bkup" command file, a message (DONE) is sent to the user to indicate completion.

/usr/bin/bkup contains:

```
echo "Gathering data"
if test -f $1${1+/}.UPDATE
  then
    find ${1-.} -newer .UPDATE -exec ls -dsl {} \; | \
    sed -n -f /usr/lib/backup/sedf | \
    awk -f /usr/lib/backup/awkf >/usr/tmp/bkup$$
  else
    lr -lsa ${1-.} | \
    sed -n -f /usr/lib/backup/sedf | \
    awk -f /usr/lib/backup/awkf >/usr/tmp/bkup$$
fi
chmod 777 /usr/tmp/bkup$$
/usr/tmp/bkup$$
rm /usr/tmp/bkup$$
touch $1${1+/}.UPDATE
echo "DONE"
```

/usr/lib/backup/sedf contains:

```
s/^\//.\//
/\.\//{
s/:/\//
s/^\.\//# /
p
}
/^....*dr/{
s/\(....\) .* \(.*\)/\& \1 \2/
p
}
/^.....-/{
s/\(....\) .* \(.*\)/% \1 \2/
p
}
```

/usr/lib/backup/awkf contains:

```
BEGIN {SIZ = 0
CNT = 0
AA=""
TRAP = "0"
LLEN = 11
print "echo -n \"Enter return when first disk is ready!\""
print "read ready"
LAST = "fbr -c"}
{LEN = length($3)}
/# /{AA = $2
PLEN = length(AA)}
{if ( $1 != "#" ){
   if ( SIZ + $2 <= 1980 && CNT + 1 <= 255 ) {
      if ( $1 == "%" ) {
         if ( LEN + PLEN + LLEN + 2 >= 1024 ) {
           print LAST
           print "fbr -u \\"
           LAST = AA$3
```

```
            LLEN = PLEN + LEN + 11
            }
          else  {
            LLEN = LLEN + PLEN + LEN + 2
            print LAST" \\"
            LAST = AA$3
            }
          }
          SIZ = SIZ + $2
          CNT = CNT + "1"
       }
    else{
       TRAP = "1"
       }
    }
if (TRAP == 1 ) {
    FIL = FIL + 1
    CNT = 1
    SIZ = $2
       if ( $1 == "%" ) {
          print LAST
          print "echo -n \"Enter return when next disk is ready!\""
          print "read ready"
          print "fbr -c \\"
          LAST = AA$3
          LLEN = LEN + PLEN + 11
          }
    TRAP = "0"
    }
}
END { print LAST }
```

CAUTION

1. bkup does not properly handle very large files. The cause of the problem has not been resolved. Failure will result in fbr producing an archive out of space message. If this occurs, exit with a "control C" and edit the file "/tmp/bkup$$" and remove the offending file reference. The file can then be reexecuted followed by entering a "touch .UPDATE" command.
2. No signals have been trapped. Adding a signal trap may be useful.
3. The problem mentioned in the first caution could be handled in the "bkup" shell script by using the shell "-e" option. The "-e" option causes the "/tmp/bkup$$" shell script to exit on encountering an error. Additionally, the return status could be tested to automate handling the problem.
4. Adding the "-v" parameter to the fbr command in the file "/usr/lib/bkup/sedf" would provide additional feedback to the user. In some applications it might be usefull.

John Owens, Marketing Applications Manager

NUMERIC FOR LOOPS IN THE SHELL

Here is a simple program which can be used in shell scripts. It generates streams of numbers, and is most typically used in shell "for" loops:

```
for i in 'from 1 to 10'
do
        ...
done
```

General syntax is:

```
from nnn to mmm by iii
```

This generates whole numbers starting with nnn, and no greater than mmm, incremented by iii. The "by iii" portion is optional. The normal name of the command is "from", although any other name should work. Additionally, If the command is invoked via the name "to", an implicit "from 1" is assumed. (No other name works like this). I normally have one binary linked into two names, "from" and "to", in my private bin.

Examples...

```
$ from 1 to 10 by 2
1
3
5
7
9

$ to 5
1
2
3
4
5
```

(Source follows).

```
/*
 * [from f] to t [by b]
 *
 *    Prints integers on standard output in the range
 *    f (default = 1) through t, incremented by b (default 1).
 *
 *    Binary may optionally be linked to two names: 'from' and 'to',
 *    allowing the from clause to be "optional".
 */

#include <stdio.h>

#define FROM 1
#define TO 2
#define BY 3

long  From = 1;
long  To;
long  By = 1;
```

```
long atol(), NextNum();

char **Argp;

#define NEXTNUM NextNum(argc, argp); argp++;argc--;

main(argc, argp)
char **argp;
{
        int ToFlag;
        long i;

        Argp = argp;
        ToFlag = 0;

        while(argc--){
                switch(what(*argp)){
                case FROM:
                        From = NEXTNUM;
                        break;
                case TO:
                        To = NEXTNUM;
                        ToFlag = 1;
                        break;
                case BY:
                        By = NEXTNUM;
                        break;
                default:
                        error("'%s' is unrecognizable\n", argp);
                }
                argp++;
        }

        if(!ToFlag)
                error("'to' field required\n");

        if(By == 0)
                By = 1;
        if(By < 0)
                By *= -1;

        if(From <= To)
                for(i=From; i<=To; i += By)
                        printf("%D\n", i);
        else
                for(i=From; i>=To; i -= By)
                        printf("%D\n", i);
        exit(0);
}

what(w)
char *w;
{
        if(strcmp(w, "from")==0)
                return FROM;
        if(strcmp(w, "to")==0)
                return TO;
        if(strcmp(w, "by")==0)
                return BY;
        return FROM+TO+BY;
}

error(f,a,b,c,d,e)
{
        printf("%s: ", Argp[0]);
        printf(f,a,b,c,d,e);
        exit(1);
}

long
NextNum(argc, argp)
char **argp;
```

```
{
        if(argc<1)
                error("Number expected after '%s'\n", argp[0]);
        return atol(argp[1]);

}
```

Jim Besemer, MDP Engineering

FAST PROGRAMMING MOD FOR 2764/27128

It presently takes approximately eight minutes to program a 2764 EPROM and 16 minutes for a 27128. Implementation of the following simple mod will reduce these programming times by approximately a factor of four. New firmware for the programmer (an 8550F33) must be installed, and we recommend that you erase the existing 2732 EPROM on the programmer board and re-program it.

Perform the following steps:

1 Cut edge connector pin 33 away from the 5V plane on the backside of the board. The pins are numbered 1-49 on the backside and the double-width copper connector counts as 21 and 23. The number 49 should appear to the right side of the backside of the board. Make cut close to edge connector.

2 Jumper edge connector pin 41 (backside) and edge connector pins 36 and 40 (component side) to remaining plane (the wide foil plane connected to ZIF socket pin 28). This provides the 6V required by the fast algorithm to Vcc pin 28 of ZIF socket. On the component side the pins are even numbered with pin 2 to the right and the triple-width copper connector counting as 12,14,16.

3 Install new firmware; it should be available at your local sales office.

4 Note that when writing a 2764 (or 27128) EPROM, the wpr command line must now read: wpr 0 2764F/I 0 1fff. Note the new Fast spec: 2764F/I.

NOTE: This mod has been incorporated into production units with serial numbers B02XXXX and higher beginning January 27, 1984.

Ted Benning, Field Applications Engineer

WHERE SHOULD COMMANDS BE LOCATED & WHY!

Commands in TNIX are easily created or modified for the benefit of a user, a group of users, all users, or for users who are working in selected directories.

As provided, most commands in the 8560 are located in /bin. The remaining TNIX commands are in /usr/bin. Starting with TNIX version 2.1, most Tektronix created commands will be located in "/tek" directory.

The TNIX shell variable "PATH" controls the order in which directories are searched for commands. Changing the variable can be done by redefining the variable when desired, or by including a nonstandard initialization in the ".profile" file. The "PATH" provided by TNIX includes the local directory, /bin, and /usr/bin. The system manager can establish a default "PATH" definition for all users by including the definition in "/etc/profile".

New commands when created may be placed in one of many locations. Generally /bin, /tek, and /usr/bin should be left as provided to assure no conflict with new TNIX commands made available in the future, or with commands that use other commands in their execution. For example, the "rm" and "test" commands are frequently used by other commands. It is recommended that user-created commands which replace TNIX commands not be placed in the above directories. Only under carefully evaluated circumstances should a new command be placed in /bin or /usr/bin that has the same name as a standard TNIX command or that replaces a standard TNIX command.

1. **Commands which need to be made available to all users should be placed in /usr/local.** Any new commands should have manual pages created for the benefit of new or infrequent users. The manual pages can be installed in /usr/man/cat9 or /usr/man/local. The "/usr/local" directory must be added since it is not standard. The man command will also require modification to permit the search of cat1 through cat9 and or the local directory. If desired, commands which contain the same command name as standard TNIX commands can be located in this directory but this should be done with caution.

   A new user might initially be better off having only the standard command environment made available. When the user has gained sufficient familiarity with the system, the new commands can be made available by simply modifying the "PATH" variable.

2. **Commands which are needed by a single user should be placed in /usr/<username>/.bin.** The "PATH" variable then would include the string "/usr/<username>/.bin. The name ".bin" was chosen so that it would not be displayed during a listing of the "/usr/<username>" directory and would also infer that it is a commands containing directory.

3. **Commands that need to be used by any user in a specific directory may be located in a ".bin" directory at that location.** If the user's PATH includes ".bin" then the .bin in the local directory will be searched for commands. This allows commands which perform the same function, but with different implementations which meet the needs of a user in that directory, to exist by the same name in various directories where they are needed. Instead of man pages, a command named "cmds" could be put in each of the .bin directories that would explain the commands available in that directory.

4. **Commands can be located in the current directory.** The default "PATH" searches the current directory of executable files, thus the user should not create executable files which are TNIX command names. An executable file with the name "test" is a sure road to problems. Using the current directory to contain commands is useful during their development. After the number of commands grows, the directory will become difficult to use due to the number of files in the listing. Generally after a command is tested and documented, it is moved to one of the locations described above.

The "PATH" variable selects both the directories which are to be searched and the order in which they are searched. For example the authors "PATH" is defined as:

PATH=:.bin:/usr/johno/.bin:/usr/local:/bin:/usr/bin

Which searches
1 The current directory
2 The ".bin" directory in the current directory
3 The "/usr/johno/.bin" directory
4 The "/usr/local" directory
5 The "/bin" directory
6 the "/usr/bin" directory

The command "PATH=:.bin:$HOME/.bin:$PATH" in a user's .profile file will result in the above path definition. This method of adding to your "PATH" is safer than using the above command literally since it incorporates any changes in the default "PATH" provided by the system.

NOTE: The above is a recommendation that, if implemented, will help prevent many problems.

John Owens, Marketing Applications Manager

GROPE - A FUZZY GREP

Occasionally when scanning files with the grep command for a pattern, it would be useful to have a few lines displayed prior to the line matched as well as after the line matched. The following command called "grope" can be used to locate a pattern in a file and display a selected number of lines prior to the line matched as well as a selected number of lines following the line containing the matched string. The command syntax is:

grope n m string <filename

where n is the number of lines prior the matched string line, m is th number of lines following the matched string, and string is the character sequence to be searched for.

The command content follows:

```
: ${1-} ${2-} ${3-}
awk "
BEGIN{ cnt = 0
MAXLN = $1 + $2 + 1          # the number of lines displayed
DLY = $2 + 1                 # one greater then the trailing lines
DELAY = 0 }
{++cnt
LINE[ cnt ] = \$0
}
{if ( DELAY > 1 )
--DELAY
}
{if ( cnt > MAXLN )
for ( i = 1; i <= cnt; i++ )
LINE[ i ] = LINE[ (i + 1 ) ]
cnt = MAXLN
}
/$3/{
DELAY = DLY
}
{if ( DELAY == 1 )
{for ( i = 1; i <= cnt ; i++ )
print LINE[ i ]
DELAY = 0
}
}
END {
if (DELAY >= 1 )
for ( i = DELAY; i <= cnt; i++ )
print LINE[ i ]
}"
```

The above was an exercise to demonstrate the use of arrays in awk.

John Owens, Marketing Applications Manager

IOCTL CALLS UNDER TNIX

With TNIX, users are permitted extensive control of input/output parameters. This degree of control greatly exceeds that of V7 UNIX*, and yet TNIX is also compatible with many programs written for UNIX which control input/output parameters. The way that this compatibility is accomplished is the subject of this article. The handling of the sg_flags component of struct sgttyb will be examined in detail.

Input/output control requests (or IOCTLs) are all made by calling the ioctl function with varying arguments. The argument list for ioctl is:

```
ioctl(channel#, opcode, address)

where channel# is the number of the I/O channel to be operated on
            (channel 1 applies to standard i/o)
      opcode   is a special number used to tell ioctl() what to do
      address  tells ioctl where to put(get) its output(input)
```

In UNIX, the opcodes for ioctl are defined in an include file named sgtty.h (this file was created for the stty and gtty system calls, hence its name). The opcode names are sometimes mnemonic, but it is important to use them rather than their defined integers. This is because the only necessary correspondence that exists between "compatible" systems lies between the ioctl opcode name (defined in the include file) and the action taken. The actual numbers are free to change, as the implementer sees fit. Here is an example of a defined ioctl opcode:

```
TIOCFLUSH - flush previously typed-in characters
```

In this article we are particularly concerned with the RS-232 port characteristics. These are expressed in a structure named sgttyb which looks like:

```
struct sgttyb {
      char  sg_ispeed;   /* the input speed                  */
      char  sg_ospeed;   /* the output speed                 */
      char  sg_erase;    /* the terminal erase character     */
      char  sg_kill;     /* the terminal kill character      */
      int   sg_flags;    /* a flag word for other parameters */
};
```

This structure is also defined in the sgtty.h file.

The sg_flags component is treated by UNIX as a list of individual bits, each of which has an assigned purpose. For example, bit 3 (the '8' position) is used to signify echo. So, if bit 3 is set, the computer echoes characters sent to it; if bit 3 is not set, then characters are not echoed. The way a bit setting actually gets translated into a port action is not important here. To make use of ioctl to control the RS-232 port, all we need to know is:

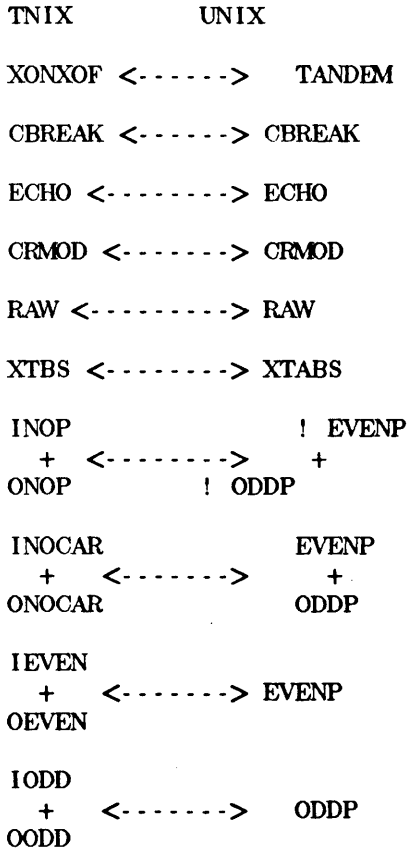
```
how to read the current settings
how to alter the current settings
what the bit positions in sg_flags mean
```

A brief digression. The difference in ioctl between TNIX and UNIX results in a reassignment of the bit positions in sg_flags. Some UNIX parameters exist in TNIX, some don't, and a number of TNIX flags are unique. Compatibility between UNIX and TNIX programs is achieved by maintaining two parallel get/set paths - one for UNIX style parameters, and one for TNIX parameters. The extra ioctl path, and the extra (and redefined) flag bits are defined in the include file tiop.h, which is the TNIX equivalent of sgtty.h. That is, you would normally only have one or the other in any given program. Internally, TNIX uses the TNIX bit positions exclusively. Ioctl takes care of translating from UNIX to TNIX settings when necessary.

To read the current tty settings under TNIX (UNIX), you use TIOPGETP (TIOCGETP), and to set them the command TIOPSETP (TIOCSETP) is used.

NOTE: Although both types of ioctl reading and writing procedures are available under TNIX, the flag settings used are quite different. Be very careful not to mix them.

For a complete list of the defined variables, refer to the include files themselves. Here is a list of equivalent settings in both syntaxes (that is, doing a set of one type may be thought of as equivalent to a set of the other type using the corresponding values):



A bug description.

In TNIX 2.0 and earlier, the translation method used to convert UNIX to TNIX bit positions in ioctl calls is not perfect. In particular, if any bits of the RS-232 port flags are set that are not in the table above, and a user does *any* UNIX-type ioctl set (e.g. ioctl(1,TIOCSETP,&mode)), all bits not in the above table will be cleared. The only ways to avoid this are:

a) don't ever change the RS-232 port settings in a program
b) don't ever set any of these bits, so they can't be changed
c) use system(3) to run stty(1) for setting the bits
d) only use TIOPGETP/TIOPSETP to change the environment in the program

Richard Doty, MDP Engineering

KSH PATCH FOR COPY USER'S SCRIPT

There is an error in the setuser command which occurs when selecting the option to "copy a custom keyshell script from one user to another." To correct the problem, simply edit /bin/setuser and change the line that reads:

cp /usr/$copyname/.ksh $userhome/.ksh

To the following:

cp /usr/$copyname/.ksh $userhome

Greg Saville, Software Support Manager

~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~

FASTER LDE INVOCATION - REVISITED

An article in a previous issue of User Group News (Vol II, Iss 3) explained how to speed up lde invocation by eliminating the help screen in the 4105's graphics plane. This note details how to make the help screen available from within the editor, but only when you request it.

First, make the changes referred to in the previous article. Then edit the first line in the 4105-.init file with lde and remove the <escape> <ctrl-L> sequence. Create a command in /usr/bin called "lde.help" which consists of the following line:

cat /usr/lib/lde/lde.4105.help

This lde.4105.help file contains the original 3rd line from the original .init file. Executing "lde.help" from within lde will paint the graphics help screen on the 4105. The removal of the ctrl-l sequence preserves the graphics plane during lde's terminal reinitialization.

Mark Malinoski, Field Applications Engineer

~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~

QUICK AND DIRTY LOGS WITH 8 BIT CHIPS

Occasionally it is useful to perform fast exponential computations with minimum of hardware, memory, and time. For example: A device that produces an output on an eight bit port that is a function of eight bit input port(s). The function may include (among other things) exponentiation, multiplication, or division. An example of expressing numbers (up to 16 bits) as a base two logarithm (up to 8 bits) follows. Log base two was selected in as much as it is functionally the same as logarithms of any other base, but much easier to convert.

If the log of a number must fit into a byte value, then the upper four bits could be chosen to represent the magnitude (characteristic) of the number and the lower four bits represent the fractional portion (mantissa) of the number. Alternate methods of partitioning the byte or word between characteristic and mantissa can be selected and are easy to implement.

```
Format chosen for the following example:

CHARACTERISTIC | MANTISSA

BITS     7 6 5 4 3 2 1 0

         ? ? ? ? ? ? ? ?
```

The conversion of a number into its log base two value is accomplished in two steps.

The number to be converted:

```
Number

f e d c b a 9 8 7 6 5 4 3 2 1 0

0 0 0 1 0 1 1 1 0 0 1 1 1 0 0 0
            1738
```

First, the upper four bits, characteristic of the resulting value, are set equal to the location of the most significant non-zero bit in the number to be converted. The characteristic is equal to N when the value of the number is in the range 2^N to 2^(N+1) - 1

Find the characteristic:

```
Number                              Log base 2

f e d c b a 9 8 7 6 5 4 3 2 1 0     7 6 5 4 3 2 1 0

0 0 0 1 0 1 1 1 0 0 1 1 1 0 0 0     1 1 0 0 ? ? ? ?
        ^
            1738                        C?
```

The number is then shifted left until the most significant bit is set. The result is a number between 8000 hex and FFFF hex that represents the fractional portion of the number. The representation of the number is changed to 2 to the Nth power times the fraction portion of the number.

Reformat the number:

```
Number                              Log base 2

f e d c b a 9 8 7 6 5 4 3 2 1 0     7 6 5 4 3 2 1 0

1 0 1 1 1 0 0 1 1 1 0 0 0 0 0 0     1 1 0 0 ? ? ? ?
^
            B9C0 * 2 ^ 0C = 1738            C?
```

To find the mantissa portion of the log value, locate the position in the following table one location prior to the table value exceeding the fractional part in the *Mantissa + 1/2 column*. If the fractional part is larger than the highest value given, then the mantissa is equal to zero and the characteristic is incremented. The Number column corresponding to the *mantissa column* could be used, but the results would favor smaller log values on the average. The Number column corresponding to the *mantissa column* would be used to find the antilog value. The table can be implemented in word or byte values. Use the table of 16 values below to compare to the upper byte of the fractional part of the number.

Log base two table

Log Mantissa	Fraction Number Byte	Number (Mantissa + 1/2) Byte
0	80	83
1	86	89
2	8C	8D
3	92	95
4	98	9C
5	9F	A2
6	A6	AA
7	AD	B1
8	B5	B9
9	BD	C1
A	C5	CA
B	CE	D3
C	D7	DC
D	E1	E6
E	EB	F0
F	F5	FB

Find the Mantissa in the log table:

```
Number                              Log base 2

f e d c b a 9 8 7 6 5 4 3 2 1 0    7 6 5 4 3 2 1 0

1 0 1 1 1 0 0 1 1 1 0 0 0 0 0 0    1 1 0 0 ? ? ? ?
^ ^ ^ ^ ^ ^ ^ ^

          B9C0                          C8
```

The table location (mantissa) 8 would be correct for this example. Thus the log base two of 1738 hex is C8 hex.

The choice of two four bit fields was arbitrary and can be changed to suit the needs of the application. For example, if better accuracy is needed, the following table of 256 values can be used to obtain the mantissa portion of the base two log of a number. If a 16 bit log value is used, then the lower eight bits of the number are obtained from the 256 value look-up table below and the upper eight bits again represent the magnitude of the number to convert. The partitioning of the word value between characteristic and mantissa can be selected to meet the needs of the application.

Base two Antilogarithm table for values 00 to FF

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00	8000	8058	80b1	810b	8164	81be	8218	8272	82cd	8328	8383	83de	843a	8495	84f1	854e
10	85aa	8607	8664	86c1	871f	877d	87db	8839	8898	88f6	8955	89b5	8a14	8a74	8ad4	8b35
20	8b95	8bf6	8c57	8cb9	8d1a	8d7c	8ddf	8e41	8ea4	8f07	8f6a	8fce	9031	9095	90fa	915e
30	91c3	9228	928e	92f4	935a	93c0	9426	948d	94f4	955c	95c3	962b	9694	96fc	9765	97ce
40	9837	98a1	990b	9975	99e0	9a4b	9ab6	9b21	9b8d	9bf9	9c65	9cd2	9d3e	9dab	9e19	9e87
50	9ef5	9f63	9fd2	a041	a0b0	a11f	a18f	a1ff	a270	a2e1	a352	a3c3	a435	a4a7	a519	a58b
60	a5fe	a672	a6e5	a759	a7cd	a842	a8b6	a92b	a9a1	aa17	aa8d	ab03	ab7a	abf1	ac68	ace0
70	ad58	add0	ae49	aec2	af3b	afb5	b02f	b0a9	b123	b19e	b21a	b295	b311	b38e	b40a	b487
80	b504	b582	b600	b67e	b6fd	b77c	b7fb	b87b	b8fb	b97c	b9fc	ba7d	baff	bb81	bc03	bc85
90	bd08	bd8b	be0f	be93	bf17	bf9c	c021	c0a6	c12c	c1b2	c238	c2bf	c346	c3ce	c456	c4de
A0	c567	c5f0	c679	c703	c78d	c817	c8a2	c92e	c9b9	ca45	cad2	cb5e	cbec	cc79	cd07	cd95
B0	ce24	ceb3	cf43	cfd2	d063	d0f3	d184	d216	d2a8	d33a	d3cc	d45f	d4f3	d587	d61b	d6af
C0	d744	d7da	d870	d906	d99d	da34	dacb	db63	dbfb	dc94	dd2d	ddc7	de60	defb	df96	e031
D0	e0cc	e168	e205	e2a2	e33f	e3dd	e47b	e51a	e5b9	e658	e6f8	e798	e839	e8da	e97c	ea1e
E0	eac0	eb63	ec07	ecaa	ed4f	edf3	ee99	ef3e	efe4	f08b	f132	f1d9	f281	f329	f3d2	f47b
F0	f525	f5cf	f67a	f725	f7d0	f87c	f929	f9d6	fa83	fb31	fbdf	fc8e	fd3e	fded	fe9e	ff4e

Obtaining the result of 1738 hex divided by the square root would be accomplished in the following steps. The result is the antilog of the log of the square root of 02 subtracted from the log of 1738 The log of 2 (which is 10 hex) divided by two is the log of the square root of two; thus log of the square root of 2 is 08. The log of the result is C8 - 08 hex which is C0. The antilog of C0 is 1000 hex.

The above result was performed using the table of 16 values. An error of less then 4% was introduced, which for many applications is acceptable. If the table of 256 values is used the result is 1065 which reduces the error to less then 1%. When performed with even greater accuracy the result would have been 106B hex.

The "C" language program used to produce the log conversion table follows:

```
#include <math.h>
  unsigned int i, k, n, c ;
  double  j, r ;
  double steps = 256;
main()
  {
     c = 0 ;                                    /* steps counter */
     j = 0.0 ;
                                                /* one loop for each line */
     for ( k = 0 ; k <= steps/16 - 1; k++ )
     {
          printf ( "%2x  ",k );
                                                 /* put 16 values on a line */
          for ( i = 0 ; i <= 15 ; i++ )
          {
          j = c/steps + 15 ;
          r =  pow(2.0,j) ;
          n = (unsigned int) (r + .00001) ;
          printf (" %4x", n ) ;
          c = c + 1 ;
          }
      printf ("\n") ;
      }
  }
```

John Owens, Marketing Applications Manager

MDP PASCAL PROCESSING CAPACITY

This article describes the processing capabilities of the 8560 versions of MDP Pascal and should be useful for judging just how large a program the compiler can handle. Since Tektronix Pascal supports separate compilation, there should be no problem in partitioning your source into small modules which can be combined at link time. Since the new virtual symbol table linker is now available, there is no real limit to the number of modules that may be linked.

The current pascal compilers on the 8560 are limited by the LSI-11 cpu to a maximum memory image of 64K bytes per invocation. Of this 64K, about 7K is reserved for the p-code interpreter which run the compiler (the compiler itself is written in pascal). Another 8K, starting at the high end of memory, is used for the pcode stack and grows downward. Approximately 28K of heap starts just above the resident pcode interpreter and grows upward toward the stack. Between the heap and stack is a movable buffer area which is used to page in the compiler pcode as needed. Under normal conditions, there is about 24K available for the compiler to build its symbol table. As the compiler runs, building its symbol table, the pcode buffer is squeezed and/or moved, causing a page fault. When 1000 page faults have been detected, a message is displayed alerting the user that excessive paging is occurring. If this continues to happen, the compiler will eventually abort, since the resulting thrashing to disk would be intolerable anyhow. When this occurs, the only solution is to divide the source into smaller modules which the compiler can handle.

The compiler capacity measures may be divided into two classes: the declarative oriented measurements and the processing oriented measurements.

In declarative capacity, the storage is tied up as long as the scope of the declared items is active. In the case of the outermost scope, this is the entire compilation unit. Therefore, this capacity can be measured by counting the number of items at the outermost level, leaving the inner scopes empty. An example of this is the number of subroutines (procedures) declared.

In processing capacity, the storage is used briefly and returned to the stack or heap. This capacity must be measured with a specified level of declarative capacity. One example of this is the parenthesis nesting in an arithmetic expression.

Following is a list of the various items with their storage requirements listed in bytes:

```
lidents = 14    length of identifier, less actual text
lconst  =  8    length of constant name
ltypes  = 20    length of type
lvars   = 12    length of var
lfields = 20    length of record field
lsubs   = 34    length of subroutine/function
lparms  = 16    length of subroutine parameter
```

The following figures list the maximum number of declarative measures that may be expected to compile successfully and assumes that the identifiers are exactly 8 characters long, and that every declared item has an identifier. (i.e., The use of var a: array[1..10] of real; requires less storage, since there are no identifiers for the subrange and array types). These are the maximum measures--each assumes that none of the others are used. For example, 800 constant names will be accepted only if no types, variables, record fields, or subroutines are present. In actual use, the program is a mixture of many types of declarative items.

Declarative Measure	Size	Max Number
constant name	30	800
types	42	570
variables	34	705
record fields	42	570
subroutines (no args)	56	428
subroutines (10 args)	436	55

This should help determine the size of program you can expect to be compiled. In some cases, you may need to use separate "include files" containing only the definitions required by the module being compiled since including all definitions can take a considerable amount of storage when not really needed for the single module being

compiled. Also keep in mind, an identifier can be any length, (as long as it fits on an input line), and all characters are significant. Therefore, using very long identifiers will use up much more storage.

Greg Saville, Software Support Manager

SETTING BREAKPOINTS ON REALS IN PASCAL DEBUG

When using 68000 Pascal and setting breakpoints on statements using real numbers, you may not get the results you expect. Since the compiler generates trap instructions rather than normal executable instructions in real number manipulations, breakpoints on them require some special considerations.

For an example, get the Payments program running as described in the Learning Guide Demonstration Run in the Pascal Debug Users Manual. (There is an article in User Group News, Volume II, Issue 2, Pages 27-33 which you may find helpful in getting the Payments program running.) If you set a software breakpoint on the statement that reads: "interest := intrate*principal ..." and "go" from pdb, you will stop at the breakpoint as expected. If you then "clear" the breakpoint, and continue with "go", everything works fine. However, should you wish to leave the breakpoint active for subsequent breaks, you will get runtime error 120, stating that the real instruction is corrupt. This is because pdb's software breakpoints patch your code with a software interrupt trap. When this trap is detected, it vectors off to a routine to halt the emulator and display a message of why it stopped. You can't continue because your original program code was modified. Therefore, if you wish to break on statements referencing reals, be aware that you must clear the break after you hit it the first time before you can continue.

Another approach is to use hardware breakpoints. The emulator and tta breakpoints do not modify your program, so you may run up to the breakpoint, halt, and restart normally. However, you will find that the emulator halts twice for each breakpoint. The first break is detected when you hit the breakpoint. When you continue with "go", the associated trap handler for the real number reads the same memory location again during its emulation, and the hardware breakpoint causes another halt. At this point you can continue with another "go", and the program continues as expected. This poses no problem, other than the minor inconvenience of having to "go" twice.

In summary, there are no bugs associated with breakpoints on math operations with real numbers, just a matter of understanding how they are handled buy the emulator.

Greg Saville, Software Support Manager

ROM PATCHES FOR 8540

Several ROM patches have recently been evaluated and approved for the 8540. ROM patches 51 and 52 deal with the lighting of the LED on the TTA board at the wrong time during diagnostic checking. Rom patches 53 and 54 solve the problem of the rompatch command hanging when there were no empty slots in the romboard. ROM patches 55 and 56 were added to correct the checksum algorithm used by rompatch, which did not compare the lower byte of the checksum.

ROM patches 51 thru 56 are listed below.

```
rompatch 03f41 51 1071 /138801/0 03
rompatch 0a743 52 10d4 /138801/0 02
rompatch 0d6ec 53 067c /DEFLT/ROMPATCH[] 95
rompatch 04699 54 06a2 /DEFLT/ROMPATCH[] c0c00401cc0c0a1b43
rompatch 09077 55 00f4 /DEFLT/ROMPATCH[] 1f2db6
rompatch 06422 56 0db6 /DEFLT/ROMPATCH[] 9c2aec0c0d89ec0d929c2aec1f20f7
```

The above ROM patches should not be entered unless the previous 50 rompatches have been entered into the 8540.

Chuck Smith, MDP Product Marketing

~ ~

FREE 8051 SIMULATOR AVAILABLE!

Travis Marlatte of E.F. Johnson Co. (Johnson Ave, Waseca, MN 56093) has created an outstanding 8051 software simulator package which can be used in the design and debug of 8051 applications on an 8560/61! He has agreed to make this package available through the MUGL library with the provision that he cannot provide any consultation or support on its use. Because of the high quality of the documentation and the availability of the fully commented source, there should be no need for contacting him with questions. However, he would appreciate any feedback regarding enhancements or bugs via mail.

Following is a summary of the simulator's capabilities paraphrased from the supplied documentation.

The environment of the simulator is an 8051, 8751, or 8031 isolated from peripheral hardware. External stimuli can be presented through the command language of the simulator. The user has complete control, with commands to cause single stepping through instructions, commands to cause constant trace output to be produced while simulating the execution of the instructions, commands to simulate a characters received via the serial UART, and full status of the simulated processor is available, plus some of the obscure registers. The 128 byte internal memory is implemented, as well as the 4K of code memory. In addition, 4K of external memory is supported. Commands to examine, block display, repeated set, and fill of all three memories are available. There are 3 types of breakpoints: PC breaks, internal memory breaks, and stackpointer breaks. There are 3 PC breakpoints available, which can optionally be set with an iteration count and/or set to arm another PC breakpoint. There are three memory breakpoints which can be set to halt simulation when an internal memory location changes value. The stackpointer breakpoint can be set to trigger whenever the stackpointer crosses the set boundary. This is useful for detecting and monitoring stack overflow.

Trace output can be directed to a file, as well as the CRT. This permits detailed analysis away from the CRT.

Simulated real time is shown as part of the status display. Instruction sequences can be accurately timed, even when breakpoints interrupt the program. The master oscillator frequency is programmable by the user.

Command summary:

- imem - examine/alter internal memory
- cmem - examine/alter code memory
- xmem - examine/alter external memory
- g - start/restart simulation
- bpmem - memory breakpoint control
- stat - display microprocessor register contents
- sel - select desired register set
- mc - initialize timer
- quit - exit simulator
- ss - enable/disable single step
- bppc - PC breakpoint manipulation
- bpstk - SP breakpoint manipulation
- read - load (Intel) hex application program
- trace - enable/disable short/long trace option
- mosc - set master oscillator to desired value

- reset - reset uP's registers
- set - set variable name value
- seri - simulate receiving a character
- trout - redirect trace output
- chksum - calculate/check checksum for code
- pctrace - view execution trace buffer
- dis - disassemble memory contents

We have run some tests with this program and find it to be a very useful tool for designing and debugging 8051 applications. Execution is very fast, operation is simple, good error messages and documentation are supplied, and the source is well commented. Since the complete C source is included, custom changes can easily be made if desired. Application programs can be written with our 8051 assembler, converted to Intel hex with the "ehex -i" command, and loaded and debugged with the simulator. Final prototype test can be made by programming an 8751 with our prom programmer and using a DAS or 1240 logic analyzer for hardware analysis. This makes a very nice, complete package for designing with the 8051 microprocessor. Best of all, the simulator is available free of charge from your MUGL library!

Here is an example log session demonstrating some of the simulator's capabilities.

```
$ : take a look at our demo program...

$ cat demo.asm
; demo.asm for sim51 - the "standard" learning guide demo program
; adapted for the 8051 uP (gas 1/19/84).

        section coderom
        org     0               ; define starting location
start   mov     r0,#table       ; set table pointer
        mov     r1,#tsize       ; set pass counter
        clr     a               ; clear accumulator
loop    add     a,@r0           ; add byte from table
        inc     r0              ; point to next byte
        djnz    r1,loop         ; decrement, loop if not five passes yet
halt    jmp     halt            ; otherwise loop forever here

        section dataram
tsize   equ     5               ; define table size
        org     50h             ; define table location in internal ram
table   block   tsize           ; reserve block of length tsize

        list    dbg             ; include symbols for lstr
        end     start


$ : assemble it...

$ asm demo.obj demo.lst demo.asm

asm                8051 X02.10-12 Copyright (C) 1983 Tektronix, Inc.
All rights reserved. Licensed Material - Property of Tektronix
*****Pass 2

   19 Lines Read
   19 Lines Processed
    0 Errors
```

```
$ : convert tek object to intel hex...

$ ehex -i demo.obj >demo.hex

$ : run lstr to get symbol table...

$ lstr demo.obj >demo.sym

$ cat demo.sym

0x00000000 S %DEMOOBJ
0x00000000 S CODEROM
0x00000000 S DATARAM
0x00000009 l HALT
0x00000005 l LOOP
0x00000000 l START
0x00000050 l TABLE
0x00000005 l TSIZE

$ : now invoke simulator and run program...

$ sim51
sim51:

: load our program...
sim51: read demo.asm
File access in progress
File access complete

: initialize our data table...
sim51: imem 4f =
50 = ff 01
51 = ff 02
52 = ff 03
53 = ff 04
54 = ff 05
55 = ff
sim51:

: set trace to instructions only...
sim51: trace inst
trace inst

: set breakpoint at "halt"...
sim51: bppc 0 9
sim51: bppc
Current breakpoint configuration:
       #   addr1 depen count
  bppc 0    09     0     1
  bppc 1    off
  bppc 2    off

: now execute...
sim51: g
Starting at addr 0000
0000     78  50       MOV         R0, #50
0002     79  05       MOV         R1, #05
0004     e4           CLR         A
0005     26           ADD         A, @R0
```

```
0006      08          INC         R0
0007      d9  fc      DJNZ        R1, 0005
0005      26          ADD         A, @R0
0006      08          INC         R0
0007      d9  fc      DJNZ        R1, 0005
0005      26          ADD         A, @R0
0006      08          INC         R0
0007      d9  fc      DJNZ        R1, 0005
0005      26          ADD         A, @R0
0006      08          INC         R0
0007      d9  fc      DJNZ        R1, 0005
0005      26          ADD         A, @R0
0006      08          INC         R0
0007      d9  fc      DJNZ        R1, 0005

PROCESSOR STATUS
acc    pc   sp  dptr      time   reg bnk  R0  R1  R2  R3  R4  R5  R6  R7
 0f  0009  07  0000      46us      00    55  00  ff  ff  ff  ff  ff  ff

P0  P1  P2  P3  PSW  TCON  SCON  IE  IP   B  TMOD  TH0  TL0  TH1  TL1
ff  ff  ff  ff   00    00    00  00  00  00    00   00   00   00   00

eP0  eP1  eP2  eP3  mP0  mP1  mP2  mP3  SBUFi  SBUFo
 ff   ff   ff   ff   ff   ff   ff   ff     00     00


Next instruction -
0009      02  00  09   LJMP        0009

* * * pc value break point

: Note, the instruction trace concludes with a full processor status
: dump.  Our calculation sum (1+2+3+4+5 = f) is in the acc register.
: The time for the complete program is 46us as shown in the time field.
: This assumes the default 6 MHz clock frequency, but can easily be
: redefined by the user with the mosc (master oscillator) command.
: At this point, the user could examine or alter any memory location,
: disassemble his code, alter processor registers, etc. and rerun.

: now run again, with full trace enabled...
sim51: trace full
sim51: g 0
Starting at addr 0000
PROCESSOR STATUS
acc    pc   sp  dptr      time   reg bnk  R0  R1  R2  R3  R4  R5  R6  R7
 00  0000  07  0000      46us      00    55  00  ff  ff  ff  ff  ff  ff

P0  P1  P2  P3  PSW  TCON  SCON  IE  IP   B  TMOD  TH0  TL0  TH1  TL1
ff  ff  ff  ff   00    00    00  00  00  00    00   00   00   00   00

eP0  eP1  eP2  eP3  mP0  mP1  mP2  mP3  SBUFi  SBUFo
 ff   ff   ff   ff   ff   ff   ff   ff     00     00


0000      78  50        MOV         R0, #50

PROCESSOR STATUS
acc    pc   sp  dptr      time   reg bnk  R0  R1  R2  R3  R4  R5  R6  R7
 00  0002  07  0000      48us      00    50  00  ff  ff  ff  ff  ff  ff
```

```
P0  P1  P2  P3  PSW  TCON  SCON  IE  IP   B  TMOD  TH0  TL0  TH1  TL1
ff  ff  ff  ff   00    00    00  00  00  00    00   00   00   00   00

eP0  eP1  eP2  eP3  mP0  mP1  mP2  mP3  SBUFi  SBUFo
 ff   ff   ff   ff   ff   ff   ff   ff     00     00


0002     79  05        MOV          R1, #05

PROCESSOR STATUS
acc    pc   sp  dptr       time   reg bnk  R0  R1  R2  R3  R4  R5  R6  R7
 00  0004   07  0000       50us      00    50  05  ff  ff  ff  ff  ff  ff

P0  P1  P2  P3  PSW  TCON  SCON  IE  IP   B  TMOD  TH0  TL0  TH1  TL1
ff  ff  ff  ff   00    00    00  00  00  00    00   00   00   00   00

eP0  eP1  eP2  eP3  mP0  mP1  mP2  mP3  SBUFi  SBUFo
 ff   ff   ff   ff   ff   ff   ff   ff     00     00


0004     e4            CLR          A

PROCESSOR STATUS
acc    pc   sp  dptr       time   reg bnk  R0  R1  R2  R3  R4  R5  R6  R7
 00  0005   07  0000       52us      00    50  05  ff  ff  ff  ff  ff  ff

P0  P1  P2  P3  PSW  TCON  SCON  IE  IP   B  TMOD  TH0  TL0  TH1  TL1
ff  ff  ff  ff   00    00    00  00  00  00    00   00   00   00   00

eP0  eP1  eP2  eP3  mP0  mP1  mP2  mP3  SBUFi  SBUFo
 ff   ff   ff   ff   ff   ff   ff   ff     00     00


0005     26            ADD          A, @R0

PROCESSOR STATUS
acc    pc   sp  dptr       time   reg bnk  R0  R1  R2  R3  R4  R5  R6  R7
 01  0006   07  0000       54us      00    50  05  ff  ff  ff  ff  ff  ff

P0  P1  P2  P3  PSW  TCON  SCON  IE  IP   B  TMOD  TH0  TL0  TH1  TL1
ff  ff  ff  ff   01    00    00  00  00  00    00   00   00   00   00

eP0  eP1  eP2  eP3  mP0  mP1  mP2  mP3  SBUFi  SBUFo
 ff   ff   ff   ff   ff   ff   ff   ff     00     00


0006     08            INC          R0

PROCESSOR STATUS
acc    pc   sp  dptr       time   reg bnk  R0  R1  R2  R3  R4  R5  R6  R7
 01  0007   07  0000       56us      00    51  05  ff  ff  ff  ff  ff  ff

P0  P1  P2  P3  PSW  TCON  SCON  IE  IP   B  TMOD  TH0  TL0  TH1  TL1
ff  ff  ff  ff   01    00    00  00  00  00    00   00   00   00   00

eP0  eP1  eP2  eP3  mP0  mP1  mP2  mP3  SBUFi  SBUFo
 ff   ff   ff   ff   ff   ff   ff   ff     00     00
```

```
0007     d9  fc      DJNZ        R1, 0005

: etc, etc...

: This gives a quick overview of just a few of the capabilities of this
: simulator.  There are many additional functions which are not
: shown here.  See the command summary listed above for more
: information on the capabilities of this fine program.
```

Greg Saville, Software Support Manager

~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~

MDP USER GROUP SOFTWARE LIBRARY/ARTICLE SUBMITTAL FORM

The following form may be used to submit software which you feel might be of interest to other MDP users.

The form and the program(s) should be forwarded to:

Technical Support Manager
Tektronix Inc.
P.O. 4600
Del Sta. 92-635
Beaverton Or.
97075

or if USENET is available:

{uv-beaver,zentel,decvax,...}!tektronix!tekmdp!mdpbug

MDP USER'S GROUP SOFTWARE LIBRARY/ARTICLE SUBMITTAL FORM

1. ABSTRACT. _____

_____

_____

2. Execution CPU_____Primary Language_____

Hardware configuration required _____

_____

Software configuration required (include source if non-Tek) _____

_____

_____

3. Do you want the following to appear in U.G.N.

Author's name_____ O yes _____O no

Company Name _____ O yes _____O no

Area code_____Tel. No. _____ O yes _____O no

Company address _____

_____

_____

4. Program Title_____

Program Function _____

5. Source. If insufficient room is provided, please submit a disk (containing the information requested) attached to this form.

6. I am submitting the program/article described above for possible placement in the MDP User's Group Library. I understand there is no compensation due to me for an accepted program/article. This program/article is of my own design; the data contained in this submittal is not copyrighted and does not break any obligation to another person or organization relating to proprietary or confidential information. Tektronix, Inc. is authorized to distribute (free of charge on customer supplied media) or publish copies of this program to Tektronix MDP users.

Signature _____Date _____

THIRD PARTY SOFTWARE

INTEL-COMPATIBLE 8086/186 ASSEMBLER/LINKER/LOCATOR FOR THE 8540

REX-SMA/186 is an integrated software development package hosted on VAX-11 under VMS for downloading to and symbolic debugging on Tektronix 8540 Integration Unit. Available from Systems & Software, the package includes an assembler, linker, locator, librarian, and Tekhex converter - all Intel-compatible. For more information about REX-SMA/196 contact Systems & Software:

Dr. Y. P. Chien
Systems & Software, Inc.
3303 Harbor Blvd., C-11
Costa Mesa, CA 92626
Phone: (714) 241-8650

This product can also be used with Caine, Farber, and Gordon's 8086 PLM compiler.

For more information about CFG's 8086 PLM compiler, contact:

Kent Gordon
Caine, Farber, and Gordon
750 East Green Street
Pasadena, California 91101
Phone: (213) 449-3070
Telex: 295316 CFG UR

Rodney Bell, Software Product Manager

PLM 8085 DEVELOPMENT SYSTEM - INTEL-COMPATIBLE

The PLM 8085 Development System is now available from Tektronix and Caine, Farber & Gordon through a cooperative marketing arrangement with CFG. This system can replace Intel systems in projects requiring PLM support. With the PLM 8085 Development System, these projects can continue their PLM-based designs ... AND benefit from the increased productivity and broader support of Tektronix systems. The PLM 8085 System consists of CFG's PLM SW, and Tektronix's 6140 8-bit color Microcomputer Development System. Other configurations of the PLM Development System are available to support Z80 and NSC800 development and VAX and UNIX hosts.

For more information about the PLM 8085 Development System, contact your local Tektronix Sales Representative. For information about the 8085 PLM software, you can also directly contact Caine, Farber & Gordon.

Rodney Bell, Software Product Manager

PRODUCT PERFORMANCE SECTION

8086 PASCAL ICS UPDATE

There is an incompatibility in the current 8086 ICS (V01.10-05) and the latest release of the 8086/87/88/186 assembler (V02.04-11). The fix is easy; simply edit the file /lib/8086/ics.mc and change the line which reads:

_____N$$W__ 0

To the following:

_____NOLIST WRN

The next release of 8086 Pascal will incorporate this update.

Greg Saville, Software Support Manager

- - - - - - - - - - - - - - - - - - - -

PASCAL 68000/68010 COMPILER FOR 8560/61

Tektronix intends to provide high-quality software products, and to help you use our products we offer the system described below to keep you informed of minor problems that have been reported so you can avoid them.

- With the optimizer on, use of the $stackck or $list in-source compiler options (to turn stack checking ptimizer to report internal errors. We recommend that the $stackck and $list option be used once at the beginning of the source file. No problems are observed when optimization is suppressed.
- If using the $tagck compiler option (to turn the checking of tag values on or off), you need to turn stack checking off by inserting a $stackck- directive at the beginning of the source file. Without this precaution, incorrect code may be generated.

- If using the Pascal 'with' statement you need to turn stack checking off by inserting a $stackck- directive at the beginning of the source file. Without this precaution, incorrect code may be generated.
- If the number of nested 'with' statements exceeds the available registers, erroneous code may be generated. This should not happen unless the 'with' statements are nested more than five deep.
- There is a problem with packed records that causes incorrect code to be generated whenever enough registers are not available and a temporary variable must be used. We suggest not using packed records.
- When the optimizer is on, one test case which had boolean constants used with relational operators failed during the third phase (code generator). If you get the following error message:

  Phase 3 ...
  Bus error: core dumped

  try compiling that module with the optimizer turned off.
- With the optimizer on, there is a limit to the number of declarations in the current scope. The limit varies with the complexity of the declarations. Hitting the limit generates the following error:

  OPT: Internal Error 602
  or
  CGEN: Internal Error 802

  One workaround is to use only the necessary declarations rather than including all declarations in all modules. If there are no unnecessary declarations, the module will have to be split so that fewer declarations are required in both new modules. Another workaround is to turn the optimizer off for that module.
- Similarly, there is a limitation in the amount of code that can be compiled in one module. The amount of code that can be handled depends on the complexity of each statement. If the limit is exceeded, you get the error:

  PARS: 203 (e) Program or module size exceeds compiler limitations

  The workaround is to split the module.

Pascal 68000/68010 Debug for the 8560

The 'step' command:

- When execution is stopped on a software breakpoint coinciding with a Pascal statement whose first machine instruction is a subroutine call, the 'step' command will skip over the statements in the subroutine, even when defined in the current module. This can happen when the statement is a parameterless procedure call, or involves an expression whose evaluation begins with a parameterless function call. Examples:

  do_something; { do_something call only }

  ch := chr(ord(inchar)+16#20); { inchar called first _____ }

  while not_done do something; { not_done called first }

  The workaround, if 'step' is needed in the parameterless procedure or function, is to enter the subroutine by means of the 'go to ...' command, then continue using 'step'.
- The 'step' command will sometimes produce one or two extra stops just before exiting a subroutine. The statement number given for the extra stop(s) coincides with the last statement number in the subroutine, even though that statement may not have been executed. If the 'tb' command is used following one of these redundant stops, the traceback display may be incorrect; specifically, the calling scope's activation record may be omitted from the index-numbered list.

  This problem does not affect program execution in any way and can be ignored; simply continue single-stepping until the calling scope is reached. Do not use the 'tb' command after a redundant stop of this kind.

Symbolic address arguments in PDB command expressions may be incorrectly pre-processed when passed through to the emulator or TTA via the TNIX shell by means of the '!!' command prefix. An argument of the form &<identifier> might end up translated into a 32-bit ASCII hex address string incorrectly containing a special internal memory-space code in the most significant eight bits, resulting in an error message from the emulator or TTA. The workaround is to query PDB about the address first, then issue the pass-through command using the absolute hex address, not the Pascal identifier.

John Owens, Marketing Applications Manager

MDP BUG BASE

The following product performance reports are contained in our data base. If you have encountered additional problems not listed here or in previous issues, please use the product performance report form provided at the end of this section. We will keep you informed about the progress toward the solution to the problem. We will also try to provide a "work-around" immediately.

John Owens, Marketing Applications Manager

KSH WHERE AM I DEBUG DISPLAY

Configuration

8560 TNIX V2, 8540 OS-40 version 1.0 and keyshell version 2.0

Problem

Get into the Debug-configure-newsetup menu and press where am i-next during each step. When fill is reached, there are 11 where am i lines and several are for unselected menus, including one that is all blank except for "manual" and "done." Even selecting items (instead of pressing next), not all the boxes picked (in "fill" even the current box) are highlighted.

INVALID ERROR ON LINK OF 186 MODULES

Configuration

8560 TNIX V2, 80186 assembler version 2.04-11 and linker V2.08-00

Problem

Assembly language modules created with the new 186 assembler and linked with modules from the 8086 V1 assembler, give a warning message that the microprocessor has been redefined. The link is successful but the relink capability is disabled.

PASCAL DEBUG BREAK POINT ERROR

Configuration

8560 TNIX V2, 8540 OS-40 version 1, 68000 pascal compiler V1.01-14, Pascal debug V1.05-00.

Problem

There is a problem with setting software breakpoints on pascal statements involving real numbers. When running up to the break, the first break occurs normally, but if you continue with pdb's "go" command, pdb gets lost and aborts. If you clear the breakpoint before restarting, pdb continues correctly. Hardware breakpoints can be used instead to get around this problem, but then you always get two breaks before continuing.

REMOTE MAIL SOURCE IDENTIFICATION

Configuration

8560 TNIX V2, Optional UNICOM Package version 1.0

Problem

When sending remote mail, the sending system is always identified as "sneezy" rather than using the name defined in whoami.h.

NO ERROR ON INVALID XCHG OPERAND

Configuration

8560 TNIX V2 and 8086/80186 assembler version 2.04-11

Problem

An error is not reported when the instruction XCHG tries to use a 16-bit register and 8-bit register together.

LDE FINDS TOKENS IN COMMENTS

Configuration

8560 TNIX V1.03 and LDE VAX UNIX V1.02; VAX VMS V1.03

Problem

If the cursor is located inside a comment field, Find or Replace Token does not work.

ACECONFIG CHARACTER LIMITATION

Configuration

8560 TNIX V2.0 and ACE V3.0

Problem

The aceconfig program does not allow the insertion of the "^" (circumflex) character into a configuration file (this is required for some teletype terminals).

COLORKEY ERROR WITH V3, 4105

Configuration

8560 TNIX V2 and 4105 firmware V3

Problem

Colorkey+ hangs when displaying *.pix files on new 4105 version 3 firmware.

Comments

Edit the *.pix files located in /usr/lib/ksh/bin and remove the "enable gin mode" command. An easy way to do this is to invoke lde on the *.pix files and delete the "escape control-z" represented as "^[^Z" near the beginning of the file. Place the cursor on the start of the above sequence and press rubout twice, write the file out, and then exit.

Z80 ASM -b OPTION SECTION PHASE ERROR

Configuration

8560 TNIX V2 and Z80 assembler V2.00-02

Problem

The Z80 assembler won't assemble properly when -b switch (Virtual Symbol Table) is used with forward referencing. (i.e., using a symbol before it is defined.) It gives an undefined operand and section phase errors. The same code works fine with 8086 and Z8000 assembler with or without -b option.

WHEX -I CHECKSUM ERROR

Configuration

8540 OS-40 version 1

Problem

When using the wh (write hex) command with the -i (INTEL format) option, no checksum is produced for the termination block.

68000 PASCAL I/O PORT LIMITATION WITH -i OPTION

Configuration

8560 TNIX V2.0 and 68000 Pascal V02.02-01 or 8086 Pascal V01.10-05

Problem

When using the -i (small integer) compiler option, I/O port addresses can not use the full 16 bit range of values. For the 68000, values greater than 7FFF fail, while values greater than CCBF fail for the 8086. Workaround -- rather than use the -i option at compilation time, define your own; type integer = -32768..32767. Although not as effective as using -i, this does allow full range addressing.

LINKER RESOLUTION OF 8048 OUT-OF-PAGE JUMPS

Configuration

8560 TNIX V2.0, 8048 ASM V2.04-08 LINKER V2.08-00

Problem

The linker does not produce an error when an in-page jump instruction (i.e. jnc) has a destination address resolved to an address outside the current page.

ACE AND CORE DUMPS WITH BREAK KEY

Configuration

TNIX V2.0 ace V3.00-00

Problem

While in command mode of ace V3.00, if the user types a "break" a "memory fault core dump" occurs, (stty cbreak -echo nl). The core dump doesn't always occur on the first "break", but will eventually occur if multiple "breaks" are typed.

UMASK AND LDE WRITE ERROR

Configuration

TNIX V2.0 / LDE V2.0

Problem

If a user inadvertently sets umask to a non-writable value such as 227, LDE will exit normally, but the file will be created empty.

LDE AND MULT. SHELL ESCAPES

Configuration

LDE V2.02 AND TNIX V2.0

Problem

If a large number (approx. 15) of shell commands are executed during a single LDE session, LDE may not have write permission in the current directory even if it should. Subsequent Update commands to LDE will not work.

PROBLEM REPORT

Customer Name _____ Date _____

Company Name _____ Title _____

Company Address _____

Internal Address/Dept _____

City _____ State _____ Zip Code _____

Area code _____ Tel. No. _____ Ext. _____

HARDWARE CONFIGURATION. Include serial number and firmware version numbers.

SOFTWARE CONFIGURATION. Include version numbers for all involved products and operating system.

PROBLEM. Include source, results obtained, and results expected. Please submit the minimum source code required to demonstrate the problem. Complete documentation will enable us to duplicate the problem.

COMMENTS.

Send to:
MDP Technical Support Manager
Tektronix Inc
Del. Station 92-635
P.O. Box 4600
Beaverton, Oregon 97075
or if you have access to USENET
{uw-beaver,zentel,decvax,...}!tektronix!tekmdp!mdpbug

USER GROUP LIBRARY ABSTRACTS

USER GROUP LIBRARY INDEX

Following is an index of the User Group Library, Volumes I, II, and III.

Volumn 1	Command	Function
	admin	login stats
	ascii	ascii converter
	asciitable	ascii table
	asmit	auto filename extension
	ats	at status
	atstats	at statistics
	decimal	decimal converter
	donum	do command x times
	donum	repeated command script
	extx	ascii text extractor
	fdmp	file dump utility
	fman	fast manual page command
	fmt	link list formatter
	help	personal commands help
	hex	hex converter
	ifix	intel fix
	ifix	intel vip utility
	ioc	ioc.form - 'fill out' form
	ioc	nroff utility
	load	object/symbol loader

load	symbolic debug loader
logl	terminal session logger
log	terminal session logger
logger	phone call logger
mailall	mail to all users
mailto	mail to users on remote systems
mdshex	intel mds symbol lister
mlabel	mailing label printer
month	calendar printer
octal	octal converter
patch	file patch routine
prolog	m900 prom programmer communications program
prq	printer queue status
prq	spooler queue status
restore	debug session restore
restore	restore emu status
save	z80 debug session save
tele	telephone number search
tools	personal commands list
tr68000	68k trace filter
trz8002	z8k trace filter
trz80	z80 trace filter
tsplit	tek hex file split
vmore	more paging utility
vmore	'vanilla more'

Volumn 2	Command	Function
	4105defines	4105 definitions library
	4105defines	4105defines.h - c define library
	box	displays graphics checkerboard on 4105
	com1	nec to tek asm source converter editor script
	debug	debug.help - display debug help screen
	dnld	tekhex downloader program
	encode	4105 programming utility
	fraction	convert floating point number to a fraction
	gcat	cat data to 4105 graphics screen
	hilbert	4105 color terminal graphics
	hp	hewlett packard calculator simulator
	ibm	ibm disk reader for the 8560
	ige	4105 graphics generator
	ige	interactive graphics editor
	intelsym	intel object to tekhex converter
	lines	4105 graphics demo
	list	listing header/formatter
	lp1r	modified printer spooler
	maint	mail list maintenance program
	mvul	rename upper to lower case file names
	pictures	pictures.dir - directory of 4105 graphics pictures
	reform	newline/carriage return/linefeed translation
	rmd	modified remove command
	rt11	unix <-> rt-11 file i/o package
	setcolor	modify 4105 character/background colors
	sierpinski	4105 color terminal graphics demo
	telex	special character filter
	thex	whitesmith's object to tekhex converter

tree	print tree structure of a directory
tta	tta.help - display tta help screen
umodem	unix - cp/m modem communications
xtab	expand tabs filter

Volumn 3	Command	Function
	6800to6809	6800 to 6809 source converter
	aototh	whitesmith a.out to tekhex converter
	bio	biorhythm plot program
	bkup	multi disk fbr incremental backup
	cpm60	cp/m disk reader
	en-un	encode/uncode binary file for cu transfer
	hp	hewlett packard calculator simulator
	intel60	intel disk reader
	moto60	motorola disk reader
	rt60	dec rt-11 disk reader
	sim51	8051 microprocessor simulator
	tekfix	motorola to tek source converter

John Owens, Marketing Applications Manager

- -

4105 GRAPHICS DEMO

Graphics Demo **MUGL TNIX Vol II**
856X w/4105 Terminal **4105**

Abstract

This directory contains 23 outstanding color graphic screens demonstrating the capabilities of the 4105 color terminal. Included are examples of pie charts, bar graphs, several maps of various areas, detailed cross sectional views of devices, printed circuit board layouts, and demo screens of the various graphic fill patterns and characters sets available on the 4105. These slides provide a very impressive demonstration of the capabilities of the Unicorn terminal. A shell script is included which automatically cycles through each slide. These files were ported from IDD's 4052 demos.

Adapted by: Doug Morrill, Atlanta FO

6800 TO 6809 - SOURCE CONVERSION

Source Converter MUGL TNIX Vol III
856X sed scrip

Abstract

This directory contains a sed script which can be used to convert Tektronix 6800 assembler source to 6809 compatible source.

Author: Bob Christman, Philadelphia Field Office

AOTOTH - WHITESMITH A.OUT TO TEKHEX CONVERTER

Conversion Utility
856X

MUGL TNIX Vol III
C

Abstract

This directory contains utilities to convert Whitesmith's object files to extended tekhex. Symbols are included according to Tekhex rules (no lower case or leading underscore) as many of Whitesmith's library routines use the underscore. These lines are prefixed with a "W" in the output symbols. A shell script is also included which automates the downloading process into an 8540 emulation station as well as some demonstration files for testing.

Author: Chris Maynard, Tek UK Harpenden

BIO - BIORHYTHM PLOT PROGRAM

Recreation
856X w/4105 Color Terminal

MUGL TNIX Vol III
C

Abstract

This program calculates and plots a nice multi-color Biorhythm chart on a 4105 color terminal. C source included.

Author: William Pfeifer, MDP Design Engineering

BKUP - MULTI DISK FBR INCREMENTAL BACKUP

Shell Script
856X

MUGL TNIX Vol III
Shell, Awk, and Sed

Abstract

This program provides the capability of backing up a set of files and directories starting at any node (directory). The initial backup saves all files and subdirectories. Subsequent backups are based on the modification dates of files. Multi-disk backups are provided and the user is prompted to insert new disks when needed. The program will create a series of "fbr" command created disks, thus files may be easily recovered.

Author: John Owens, Tektronix, Inc.

CPM60 - CP/M DISK READER

Media Utility
856X

MUGL TNIX Vol III
C

Abstract

This utility allows reading standard single-sided, single-density CP/M format disks on the 8560/1. Featuring built in "help" prompting, options are provided for listing CP/M directories, copying binary or text CP/M files to the 856X hard disk, and wildcard specs with optional query. In addition, this updated version now handles extents properly, thus allowing transfer of CP/M files greater than 16K bytes. Note: this program is a reader only, it has no provision at this time for writing to CP/M disks.

Author: Diane Wortsmann, MDP Marketing
Adapted by: Howard Christeller, DC Field Office

EN/UN - ENCODE/UNCODE BINARY FILE FOR CU TRANSFER

Communications Utility
856X/UNICOM

MUGL TNIX Vol III
C

Abstract

These complimentary programs allow transferring binary files with the UNICOM CU program. EN encodes a raw binary image into an ascii representation suitable for "~ %take'ing" or "~ %put'ing" with cu. Once transferred, UN uncodes the ascii representation back to its original binary form. C source included.

Author: Greg Saville, MDP Marketing

HP - HEWLETT PACKARD CALCULATOR SIMULATOR

Utility/Simulator
856X

MUGL TNIX Vol III
C

Abstract

This program simulates a very complete HP Calculator. In addition to the reverse polish style of operations and the standard mathematical functions, other features include: numeric entry similar to the HP-1X series; full range of trigonometric functions, including hyperbolic functions; decimal and analog time conversions; statistical functions including combinations, permutations, correlation, linear regression and estimation; exponential engineering, and fix notations; integer arithmetic in decimal, octal, and hex modes including logical operations AND, OR, XOR, and NOT; 62 continuous memory registers including memory register arithmetic; register exchange functions; polar and rectangular conversions; and more. Features "cbreak" operation, so the return key acts as a true enter key and is not required for line input. Command set includes: enter, clearx, factorial, sigmaplus, sigmaminus, stats, percent, multiply, add, subtract, divide, convtime, convfrom, convto, squarex, clear, stackdisplay, exponential, fix, inverse, lastx, rotatemode, snlog, off, pushpi, recallmem, storex, squareroot, exchange, mod, power, absolute, cosine, rotatedown, fraction, pgamma, hypotenuse, integer, clog, mantissa, nlog, polar, rectangular, sine, tangent, rotateup, xychange, and, or, xor, complement, changesign. Even simulates "continuous memory" by saving entire calculator state when terminating and restoring status upon reinvocation. Executable binary image and manual page only, source code is not available.

This is an updated release from the original version offered in MUGL Volume II which corrects reported bugs.

Author: Ed Morin, MDP Design Engineering

INTEL60 - INTEL DISK READER

Media Utility
856X

MUGL TNIX Vol III
C

Abstract

This utility allows reading single-sided, single-density Intel ISIS format disks on the 8560/1. Featuring built in "help" prompting, options are provided for listing directories, copying binary or text files to the 856X hard disk, and wildcard specs with optional query. Note: this program is a reader only, it has no provision at this time for writing Intel disks.

Author: Diane Wortsmann, MDP Marketing

MOTO60 - MOTOROLA DISK READER

Media Utility — MUGL TNIX Vol III
856X — C

Abstract

This utility allows reading single-sided, single-density Motorola MDOS V2.0 format disks on the 8560/1. Featuring built in "help" prompting, options are provided for listing directories, copying binary or text files to the 856X hard disk, and wildcard specs with optional query. Note: this program is a reader only, it has no provision at this time for writing Motorola disks. C source included.

Author: Diane Wortsmann, MDP Marketing

RT60 - DEC RT-11 DISK READER

Media Utility — MUGL TNIX Vol III
856X — C

Abstract

This utility allows reading single-sided, single-density DEC RT-11 format disks on the 8560/1. Featuring built in "help" prompting, options are provided for listing directories, copying binary or text files to the 856X hard disk, and wildcard specs with optional query. Note: this program is a reader only, it has no provision at this time for writing RT-11 disks.

Author: Diane Wortsmann, MDP Marketing

SIM51 - 8051 MICROPROCESSOR SIMULATOR

Simulator — MUGL TNIX Vol III
8560/61 — C

Abstract

This package implements a simulator which can be used for designing and debugging 8051 microprocessor applications on an 8560/61.

Following is a summary of the simulator's capabilities paraphrased from the supplied documentation.

The environment of the simulator is an 8051, 8751, or 8031 isolated from peripheral hardware. External stimuli can be presented through the command language of the simulator. The user has complete control, with commands to cause single stepping through instructions, commands to cause constant trace output to be produced while simulating the execution of the instructions, commands to simulate a characters received via the serial UART, and full status of the simulated processor is available, plus some of the obscure registers. The 128 byte internal memory is implemented, as well as the 4K of code memory. In addition, 4K of external memory is supported. Commands to examine, block display, repeated set, and fill of all three memories are available. There are 3 types of breakpoints: PC breaks, internal memory breaks, and stackpointer breaks. There are 3 PC breakpoints available, which can optionally be set with an iteration count and/or set to arm another PC breakpoint. There are three memory breakpoints which can be set to halt simulation when an internal memory location changes value. The stackpointer breakpoint can be set to trigger whenever the stackpointer crosses the set boundary. This is useful for detecting and monitoring stack overflow.

Trace output can be directed to a file, as well as the CRT. This permits detailed analysis away from the CRT.

Simulated real time is shown as part of the status display. Instruction sequences can be accurately timed, even when breakpoints interrupt the program. The master oscillator frequency is programmable by the user.

Command summary:

- imem - examine/alter internal memory
- cmem - examine/alter code memory
- xmem - examine/alter external memory
- g - start/restart simulation
- bpmem - memory breakpoint control
- stat - display microprocessor register contents
- sel - select desired register set
- mc - initialize timer
- quit - exit simulator
- ss - enable/disable single step
- bppc - PC breakpoint manipulation
- bpstk - SP breakpoint manipulation
- read - load (Intel) hex application program
- trace - enable/disable short/long trace option
- mosc - set master oscillator to desired value
- reset - reset uP's registers
- set - set variable name value
- seri - simulate receiving a character
- trout - redirect trace output
- chksum - calculate/check checksum for code
- pctrace - view execution trace buffer
- dis - disassemble memory contents

We have run some tests with this program and find it to be a very useful tool for designing and debugging 8051 applications. Execution is very fast, operation is simple, good error messages and documentation are supplied, and the source is well commented. Since the complete C source is included, custom changes can easily be made if desired. Application programs can be written with our 8051 assembler, converted to Intel hex with the "ehex -i" command, and loaded and debugged with the simulator. Final prototype test can be made by programming an 8751 with our prom programmer and using a DAS or 1240 logic analyzer for hardware analysis. This makes a very nice, complete package for designing with the 8051 microprocessor. Best of all, the simulator is available free-of-charge from your MUGL library!

Author: Travis Marlatte, E.F. Johnson Co.
Please do not contact the author regarding general usage, however
feedback regarding enhancements, bugs, etc. is solicited in
writing to:

E.F Johnson Co.
Johnson Ave.
Waseca, MN 56093

TEKFIX - MOTOROLA TO TEK SOURCE CONVERTER

Source Converter
EXORmacs/Versados 3.0/Pascal 2.0

MUGL TNIX Vol III
Pascal

Abstract

This utility can be used to transport assembly language source from a Motorola EXORmacs development system to a Tektronix 8550 or 8560. Assembler directives supported by Motorola are converted to their comparable Tektronix equivalents. There are some items which must be converted manually, most notably macros. This program is written in Motorola Pascal, Version 2.0 for Versados 3.0 or later. All sources are included on this MUGL Volume in 8560 fbr format and must be ported to an EXORmacs for compilation and execution.

Author: Charles A. Brandt, Advanced Mechanization, Inc.

INDEX

USERGROUP NEWS EVALUATION

Customer Name	Company Name
Address	
City	State
Zip	Mail label No. OR

OPTIONAL

Please rate articles presented in "USER GROUP NEWS" on a scale of one to ten. Enter your rating on the following attributes; usefulness[ ], depth,[ ] effectiveness[ ], accuracy[ ], scope[ ], format[ ], or other _____ [ ] Please provide a rating of the following sections of "USER GROUP NEWS"; "Product Information"[ ], "Applications"[ ], "Third Party Software"[ ], "Product Performance"[ ], "User Group Library Abstracts"[ ].

Have you used an application program presented in "USER GROUP NEWS"? Yes[ ], or No[ ]. Have you used a library program? Yes[ ], or No[ ]. How can "USER GROUP NEWS" serve you better? Please provide suggestions below.

_____

_____

_____

Do you feel that an "Answers to Users' Questions" section would be useful? Yes[ ], or No[ ].

What Third Party Software have you used on an 8560? _____
What micros are you using today? _____
What micros are you going to use soon? _____
What micros are under consideration for the future? _____
Will your design include multiple micros? _____
What development system enhancements do you want to see? _____

_____

_____

Send us your answers - we're interested. Please use the postage paid envelope enclosed with this issue or send to:

MDP Technical Support Manager
Tektronix Inc
Del. Station 92-635
P.O. Box 4600
Beaverton, Oregon 97075
or if you have access to USENET
{uw-beaver,zentel,decvax,...}!tektronix!tekmdp!mdpbug

NEW INFORMATION IS AVAILABLE

Subject/product	**I would like to receive the following information:**
8 Bit Standalone Development System	☐
16 Bit Development System	☐
C Language Development System	☐
VAX Software	☐
8500 Series System Brochure	☐
80186 Emulator System	☐
68010/68008 Emulator System	☐
1750A Bus Emulator	☐
NSC800 Emulator	☐
7800 Family Emulator	☐
Third Party Vendor Catalogue	☐
Have a salesperson call	☐
Other_____	☐

IMPORTANT NOTICE

Please complete this card IF:

1. The address label is incorrect, or
2. You are not the person who should receive this newsletter, or
3. You wish to add names to our mail list.

For identification purposes, please place the number on the upper right of your current mailing label here:_____

☐ add to list ☐ change address ☐ remove from list

Company name _____

Company address _____
internal address or department

_____
street & number or P.O. Box

_____
city, state, zip

_____
country

User Name _____

Title _____

Do you want the newsletter label to specify the:

☐ title only ☐ name only ☐ both in future newsletter mailings
date_____