ORION

UniLab II™ Universal Development System

LOW COST SUPPORT OF SOFTWARE AND HARDWARE DEBUGGING
FOR 150 DIFFERENT 8 AND 16 BIT MICROPROCESSORS

Thank you for your interest in the latest tool for improving microprocessor system
development productivity.

The UniLab II is actually four instruments in one:

1. a real-time, In-Circuit Emulator which uses an actual target microprocessor for
   absolute transparency,

2. an Advanced 48 Channel Bus-State Analyzer for locating software and hardware
   bugs,

3. a Stimulus Generator for automated testing, and

4. an EPROM Programmer for popular devices.

All in a single package!

PROGRAM PERFORMANCE ANALYSIS™

Optimize your code’s speed with UniLab’s Program Performance Analysis option. At
just $495 you can graphically display just the information you need.

YOU AND YOUR DESIGNS CAN BE A WINNER WITH UniLab!

The UniLab’s non-intrusive analysis technique lets your hardware and software run at full
speed, without modification. Its exceptionally powerful triggering language allows you to
specify precise trigger points based on symptoms you observe, and let the UniLab locate
the causes for you. The UniLab II also supports all conventional debugging techniques
including multiple breakpoints and single stepping.

EASY SUPPORT FOR TOUGH MICROCS

UniLab’s portfolio of Personality Paks™ makes it easy to select exactly the cables,
adapters, and software you need to work with nearly any target microprocessor. The
cost of processor specific support: typically under $600.

You can even get an Orion MicroTarget™, a functioning minimum target system, for
many popular microprocessors. In-Place Emulation Modules plug right into your
microprocessor’s socket for easy hook up.

Call Orion’s Sales Engineers today, toll free, 1-800-245-8500 or 415-361-8883
(within California) for more information about the exciting UniLab. Better yet, call to
reserve your own UniLab for A.S.A.P. shipment.

Best Regards,

Orion Instruments, Inc.
The UniLab II is the latest generation of state-of-the-art low-cost development systems. With support for virtually any 8 or 16 bit microprocessor, the UniLab II offers you ease of use and a range of advanced features not found even in more expensive, but less capable alternatives.

Supporting more than 150 different microprocessor types, the UniLab II gives you the ultimate in versatility while providing all the essential tools in a single integrated system. For ease of ordering precisely what you need to support a particular microprocessor, Orion now offers Personality Paks configured especially for your target processor. See the Personality Pak price list and ordering configuration guide for the chart of microprocessors supported.

The UniLab II requires an IBM PC, XT, AT or compatible PC with PC-DOS 2.1 or later. A minimum of 320K RAM (512K with Program Performance Analyzer option) and one floppy disk drive is required, but a second disk drive (floppy or Winchester) is recommended. An RS-232C port is required for connection of the UniLab. VAX based systems are available. Please contact our Sales Engineers for further information.

**PRICES:**

**STANDARD UNILAB II**
(395 ns minimum bus cycle time, 195 ns minimum ROM access time)

<table>
<thead>
<tr>
<th>PART NO.</th>
<th>DESCRIPTION</th>
<th>PRICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>84101</td>
<td>UniLab II (32K emulation memory)</td>
<td>$4980.00</td>
</tr>
<tr>
<td>84102</td>
<td>UniLab II (64K emulation memory)</td>
<td>$5380.00</td>
</tr>
<tr>
<td>84103</td>
<td>UniLab II (128K emulation memory)</td>
<td>$5580.00</td>
</tr>
</tbody>
</table>

**HIGH-SPEED UNILAB II**
(297 ns minimum bus cycle time, 150 ns minimum ROM access time)

<table>
<thead>
<tr>
<th>PART NO.</th>
<th>DESCRIPTION</th>
<th>PRICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>84201</td>
<td>UniLab II (32K high-speed emulation memory)</td>
<td>$5380.00</td>
</tr>
<tr>
<td>84202</td>
<td>UniLab II (64K high-speed emulation memory)</td>
<td>$5780.00</td>
</tr>
<tr>
<td>84203</td>
<td>UniLab II (128K high-speed emulation memory)</td>
<td>$5980.00</td>
</tr>
</tbody>
</table>

System hardware is both memory and speed upgradeable. (Upgrade performed at factory only.)

**ACCESSORIES INCLUDED:**

- User's Manual with Tutorial Section
- Comprehensive Reference Manual
- Quick Reference Card
- Stimulus Generator Cable
- Jumper Cable Wiring Tool
- 16-pin IC Clip
- Component Clip Adaptor Probes (2)

**PROGRAM PERFORMANCE ANALYZER:**

The Program Performance Analyzer (PPA) is a time-saving software tool which generates a graphical time or address domain display of your program's execution. This useful enhancement to the UniLab II system gives you the ability to uncover invisible bugs, eliminate unneeded code and increase program efficiency at a remarkably low price.

PART NO: 87002 ................................................................. $ 495.00
WARRANTY:
All Orion products are covered against defects in workmanship and materials for a period of 90 days from date of purchase. Defective items returned to the factory during the warranty period will be repaired or replaced at Orion's option, and returned to the customer. Customer pays freight in, Orion pays freight out. Detailed warranty statement available.

SUPPORT SERVICES OPTION INCLUDING EXTENDED WARRANTY
During our 90-day warranty, you are entitled to unlimited telephone Applications Engineering support from Orion. You can continue to receive unlimited Applications Engineering telephone support and an extended product warranty for $500 per year per system, if you purchase this agreement within the normal warranty period. The extended warranty begins at 90 days and covers all purchased accessories for an additional twelve (12) months (cables are not covered after initial 90-day warranty). Support Services Option subscribers also receive free of charge all software updates, and the Orion Express customer newsletter. After the normal 90-day warranty period, the extended warranty is available for $700. Consulting services are available at a special 50% discount rate to Support Services Option subscribers.

DISCOUNTS:
Please contact Orion for information on quantity and educational discounts.

PAYMENT TERMS:
Open account payment terms are available to rated firms. MasterCard, VISA, and American Express cards also accepted. C.O.D. shipments payable only in cash or cashier's check. Prices and specifications subject to change without notice.

ORIGINAL UNIVERSAL DEVELOPMENT LABORATORY
The UDL, our original model development system, is still available for CP/M and DOS applications. Prices from $2995.00.

Orion Instruments, Inc.
ORION
702 Marshall Street
Redwood City, California 94063
Telex: 530942
FAX: 415-361-8970

All Orion products are sold with a no-risk 15-day money-back guarantee. Order Now!

Call Toll-Free 1-800-245-8500
In California: 415-361-8883
ORION UDL™ — Universal Development Laboratory

Orion’s original Universal Development Laboratory, the UDL, is perfect for the smaller budget. With support for virtually any 8 or 16-bit microprocessor, the UDL gives you a basic integrated development system at the lowest cost. The UDL lets you get started inexpensively, but is fully upgradeable to the advanced features and ease of use of Orion’s new UniLab II. Ask our Sales Engineers for further information.

For ease of ordering precisely what you need to support a particular microprocessor, Orion now offers Personality Paks® configured especially for your target processor. See the Personality Pak Order Configuration guide for the chart of microprocessors supported.

The UDL runs on most CP/M (Version 2.2 or 3.0, 64K RAM required) 5¼” and 8” formats and MS-DOS (Version 2.1 or later, 128K RAM required). A minimum of one floppy disk drive is required, but a second disk drive (floppy or Winchester) is recommended. An RS-232C port is required for connection of the UDL to your computer.

The UDL comes with a User’s Manual, a stimulus generator cable, a jumper cable wiring tool, a 16-pin IC clip, and component clip adaptors.

### PRICES:

#### STANDARD UDL

(395 ns minimum cycle time, 195 ns minimum ROM access time)

<table>
<thead>
<tr>
<th>PART NO.</th>
<th>Description</th>
<th>PRICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>82101</td>
<td>Universal Development Laboratory (32K emulation memory)</td>
<td>$ 2995.00</td>
</tr>
<tr>
<td>82102</td>
<td>Universal Development Laboratory (64K emulation memory)</td>
<td>$ 3390.00</td>
</tr>
<tr>
<td>82103</td>
<td>Universal Development Laboratory (128K emulation memory)</td>
<td>$ 3590.00</td>
</tr>
</tbody>
</table>

#### HIGH-SPEED UDL

(297 ns minimum cycle time, 150 ns minimum ROM access time)

<table>
<thead>
<tr>
<th>PART NO.</th>
<th>Description</th>
<th>PRICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>82201</td>
<td>Universal Development Laboratory (32K high speed emulation memory)</td>
<td>$ 3430.00</td>
</tr>
<tr>
<td>82202</td>
<td>Universal Development Laboratory (64K high speed emulation memory)</td>
<td>$ 3855.00</td>
</tr>
<tr>
<td>82203</td>
<td>Universal Development Laboratory (128K high speed emulation memory)</td>
<td>$ 4085.00</td>
</tr>
</tbody>
</table>

### WARRANTY:

All Orion products are covered against defects in workmanship and materials for a period of 90 days from date of purchase. Defective items returned to the factory during the warranty period will be repaired or replaced at Orion’s option, and returned to the customer. Customer pays freight in, Orion pays freight back. Detailed warranty statement available.

### PAYMENT TERMS:

Net open account terms available to rated firms. MasterCard, VISA, and American Express cards also accepted. C.O.D. shipments payable only in cash or cashier’s check. Prices and specifications subject to change without notice.

Orion Instruments, Inc.

702 Marshall Street
Redwood City, California 94063
Telex: 530942
FAX: 415-361-8970

All Orion products are sold with a no-risk 15-day money-back guarantee. Order Now!

Call Toll-Free 1-800-245-8500
In California: 415-361-8883

Rev. 7-87
Printed in U.S.A.
Why companies that can afford any I² development system choose the one that costs only $2995.

The UDL™ is four powerful instruments ingeniously boxed in one.
So why would you want to spend $2995 on our development system when you could buy something really expensive?

To begin with, the UDL—for Universal Development Laboratory—turns almost any computer into a powerful integrated workstation for software and hardware debugging of almost any target microprocessor.

Internal proprietary software neatly integrates an advanced 48-channel bus state analyzer, an 8/16-bit in-circuit emulator, an EPROM programmer, and an input stimulus generator.

When they see the UDL in action for the first time and learn the principal behind its design, engineers ask, "why didn’t anybody think of that before?" What can we say?

One more thing. Our unique real-time emulation technique gives perfect transparency, yet allows you to work with 36 different target processor types without buying special hardware adapters.

In sum, the UDL lets you do the work of a $30,000 I² development system for less than the cost of the usual personality module. Even companies that already own “boat-anchor” development systems are buying UDLs by the bundle as an economic way to add I² capability.

For complete details on the UDL—or for a 10-day no obligation engineering evaluation program—please call or write.

After all, it’s not just any development system.

ORION Instruments
702 Marshall St., 6th Floor
Redwood City, CA 94063
Tel. (415) 361-8883
Telex 530942
TELEPHONE REFERENCE GUIDE

SALES DEPARTMENT

VP SALES
Bill White

SALES ENGINEERS
John Bugee' (Eastern Region) ext.238
Bill Leach (Central Region) ext.241
Jan Liband (Western Region) ext.240

SALES COORDINATORS
Karen Cash ext.228
Claudia Holt ext.235

TECHNICAL SUPPORT

TECH SUPPORT SUPV.
Bob Bowes ext.227

TECHNICAL LIASION
Scott Gilliland ext.247

TECH SUPPORT ENGINEERS
Chuck Gorman ext.231
Marina Zago ext.215

CUSTOMER SERVICE

CUSTOMER SERVICE REPS
Katy Morkner (Central/Intl) ext.221
Marcia Smith (Eastern/Western) ext.221

Telephone: 415-361-8883
800-245-8500

Fax: 415-361-8970

Telex: 530942

6.9.87/TRG
ORION

DOMESTIC SALES REPRESENTATIVES, AUGUST 1987

WESTERN REGION

SO. CALIFORNIA

Advanced Product Sales - APS
32184 Oak Shore Drive
Westlake Village, CA 91361

Telephone: 818/706-8351

Lucky Balin

APS
847 Triunfo Canyon Rd.
Westlake Village, CA 91361

Telephone: 805/496-7355

Mort Katz

APS
13130 Decant Drive
Poway, CA 92064

Telephone: 619/530-0200

Mike Knapp

ROCKY MOUNTAIN

Scientific Devices Inc. - SDI
1006 Depot Hill Road
Broomfield, CO 80020

Telephone: 303/469-5145
Fax: 303/465-2980

Brian Clark
(also temporarily covering Utah)

SDI
3920 E. Indian School Rd.
Phoenix, AZ 85018

Telephone: 602/957-0401
Fax: 602/956-9359

Jeff Gipe
SDI
5215 Los Arboles NE
Albuquerque, NM 87110

Telephone: 505/883-3668
Fax: 303/465-2980 — (CO office)

SDI**
16882 Balsa Chica Street
Huntington Beach, CA 92649

Telephone: 714/840-3501

** Billing address only.

NORTHWEST

Syntek - SYN
11410 N.E. 124th Street
Kirkland, WA 98034

Telephone: 206/488-0394
Fax: 206/821-7726

NO. CALIFORNIA

Western Digital, Inc. - WDI
655 Sky Way, Suite 123
San Carlos, CA 94070

Telephone: 415/591-6535
CENTRAL REGION

SOUTHWEST

Scientific Devices Southwest - SSW
1200 East Collins Blvd.
Suite 110
Richardson, TX 75081 (Dallas & Ft. Worth)

Telephone: 214/231-8106
*OK,AR,LA

Harvey Evans
Dan Evans
Carl Geller
Charles Farris
Jerry Bissell
Teri Campion

Rick Morgenstern

SSW
16720 Stuebner Airline Road
Suite 222
Spring, TX 77379 (Houston)

Telephone: 713/376-8666

Terry Bissell

SSW
605 Broken Bow
Roundrock, TX 78681 (Austin & San Antonio)

Telephone: 512/388-3982 - Austin
512/736-3164 - San Antonio

WATER WORLD

Sector Engineering Sales Inc. - SES
8705 Port Washington Road
Milwaukee, WI 53271

Telephone: 414/352-8280

Ron Zimm
Cathy Zimm

SES
8441 Wayzata Blvd
Suite 104
Bloomington, MN 55426

Telephone: 612/593-1809

Mike Baum
TRISTATE – BUCKEYE

Sellenraad & Associates
770 South Adams
Suite 111
Birmingham, MI 48011

Telephone: 313/645–2640
Fax: 313/540–4393

Wil Sellenraad
Terry Wing
Brian Clapp
## EASTERN REGION

### NEW YORK

**Dick Brown Associates - DBA**  
1728 Rue Mirador  
Point Pleasant, NJ 08742  

Telephone: 201/295-1357

---

### KEYSTONE STATES

**John Bell Associates - JBA**  
The Limekiln  
Rt. 202 - #3  
Holicong, PA 18928  

Telephone: 215/794-7075  
Fax: 215/794-0744

---

### CHESAPEAKE

**Scientific Associates, Inc. - SAI**  
9512A Lee Highway  
Fairfax, VA 22031  

Telephone: 703/385-0600  
Fax: 703/359-7341

---

### SOUTHEAST

**Teqspec - TCS**  
3211A Alt. US 19 N  
Palm Harbor, FL 34273-0399  

Telephone: 813/785-2276

---

TCS  
1325 N. Atlantic Ave. #192  
Cocoa Beach, FL 32931  
Telephone: 305/783-3444
TCS
PO Box 11060
Ocala, FL 32673
Telephone: 904/347-1707

TCS
PO Box 2786
Norcross, GA 30071-2786
Telephone: 404/242-7215

TCS
1019 A Old Monrovia RD
Suite 132
Huntsville, AL 35806
Telephone: 205/883-6699

TCS
3919 Quiling Road
Winston Salem, NC 27104
Telephone: 919/765-5553

NORTHEAST

Tri-Logic – TLI
187 Ballardvale Street
Ballardvale Park
Willington, MA 01887

Telephone: 617/658-3800

Brad Paul
Rich Paul
Dave Paul
Rich Salmon
Dave Starrett
Kamie Rienhart
Joe Socha

7.28.87 REPLIST
Host Computer Interface
RS-232C connector, 19,200 or 9,600 baud, switch selectable.

Diskette Formats
CP/M 8" IBM format, many 5¼" formats
IBM PC 5¼", MS-DOS
ISIS
Apple CP/M, 80 col

Emulator
Download time: 1 second for 2K bytes including
16-bit block error check (2 sec on some systems)
195 ns max access time ROM emulation.
(145 ns optional)
32K X 8-bit or 16K X 16-bit standard. By cable, program option.
Expandable to 128 K bytes with optional plug-in board.
20-bit enable address decoding.
Individual 2k segments can be selected in any combination within 17-bit field.
Stand-alone operation possible as a ROM emulator.
16-bit idle register loops target CPU allowing loading of emulation RAM and resumption of program execution.
Optional, target-processor-specific, software gives full debugging capability including register and target memory display and change, breakpoints, and single stepping.
Program loading software: from hex or binary disk files, hex serial download, memory image, ROM read.

Bus-State Analyzer
48 data inputs. Two groups of 8 can be separately clocked.
6 clock signal inputs. Gated to form one bus clock.
Clock edge filter prevents re-trigger before 100 ns.
395 ns minimum bus cycle (10 MHz 68000)
297 ns with optional high-speed option.
Address demultiplexing latches included—also used by emulator
170 cycle x 48 bit Trace memory

EPROM/EEPROM Programmer
Smart programming algorithm for high speed.
28-pin Testol zero insertion force socket handles 24 and 28 pin devices.
Programs single supply EPROMs and EEPROMs.
One personality module handles 2716, TMS2516, 2532, 48016.
Personality modules for 2732A, 2764/128, 2764A/128A/256 also included.
Optional module available for 27512.

Signal Inputs
TTL logic levels. (74ALS inputs)
1 ma maximum loading includes emulator & analyzer.

Analyzer Trigger
4-step sequential trigger
RAM truth tables allow search for any function of 8-bits at each 8-bit group, for each step.
8 truth tables per step x 4 steps = 32 256-bit tables
16-bit inside/outside range detection on address lines
4-bit segment enable gives 20-bit address capability.
Pass Counter: wait up to 65,362 events or cycles before 4th step.
Before/After/AT Pass count trigger enable.
Delay Counter: wait up to 65,362 events or cycles to stop trace.
Filter feature. Records only cycles which satisfy trigger criterion plus 0-3 cycles after each qualified cycle.
Oscilloscope sync output. (Sync on trigger.)
Interrupt output: Interrupt target on trigger (if enabled)
LED indicates searching for trigger. Stand-alone operation possible while waiting for trigger.

Software Features
Command driven with single context for all 4 instruments.
Extensive macro capability
On-line Help screens.
Menu driven shell generates command lines.
User definable function keys.
Calculator, ASCII table, IC pinout library, menu pad, terminal emulation, and DOS access.

Signal Outputs
TTL logic levels (74LS244 outputs)
100 ohms forward terminating resistors on Emulator data lines.
Reset output: open collector, 7406 thru 47 ohms.
Interrupt output: open collector, 7406, low true
9 Stimulus outputs (at EPROM socket) (8255 NMOS outputs)

Physical
Size: 2.1" hi x 13" wide x 7.8" deep.
Weight: 4 lbs. (11 lbs. shipping weight)
Fits easily in a slim-line brief case.

Power
100 KHz switching supply
110v ± 10% 50/60 Hz 15 Watts (standard)
220v ± 10% 50/60 Hz 15 Watts (optional)

Accessories Included
UDL User's Manual
Personality Module for 2716, 2758, 2756, 2532 EPROMs and 48016 EEPROMs, 2732A module, 2764/128 module, 2726/2764A/27128A personality module.
1-ROM Emulator cable 8-bit, 24-pin version unless otherwise specified.
1-Analyzer cable pre-configured for your target processor
1-Jumper wiring tool (3M)
1-40-pin IC clip
1-16-pin IC clip
1-8232 cable, 10-ft. retractile (coiled cord)
1-Input stimulus cable
2-Component clip adaptor probes
1-System Control Program Diskette

Disassembler/Debugger Software
Includes symbolic single-step, target memory and register display and change, program start/stop/branch, input/output. Available for the following processors:
1802/4/5/6, 6800/3, 6800/02/00, 6800/2/8, 6801/3, 6802, 6805, 6809, 68010, 68008, 8080/5/35/39/40/49/50, 8051/31/32/52, 8085/80, 8086/8/286, 8088/188, Z-8, Z-80 and NSC-800.
<table>
<thead>
<tr>
<th>OPTION</th>
<th>PART ID</th>
<th>REV-RTN #</th>
<th>LIST PRICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB48</td>
<td>UDL, GROUP 48</td>
<td>77001</td>
<td>4203-907</td>
</tr>
<tr>
<td>DB51</td>
<td>UDL, GROUP 51</td>
<td>77002</td>
<td>4203-907</td>
</tr>
<tr>
<td>DB51</td>
<td>UNILAB, GROUP 51</td>
<td>77003</td>
<td>4203-907</td>
</tr>
<tr>
<td>DB51P</td>
<td>UDL, GROUP 51</td>
<td>77004</td>
<td>4203-907</td>
</tr>
<tr>
<td>DB51P</td>
<td>UNILAB, GROUP P</td>
<td>77005</td>
<td>4203-907</td>
</tr>
<tr>
<td>DB611</td>
<td>UDL, GROUP 61</td>
<td>77006</td>
<td>4203-907</td>
</tr>
<tr>
<td>DB63</td>
<td>UDL, GROUP 63</td>
<td>77007</td>
<td>4203-907</td>
</tr>
<tr>
<td>DB63</td>
<td>UNILAB, GROUP 63</td>
<td>77008</td>
<td>4203-907</td>
</tr>
<tr>
<td>DB65</td>
<td>UDL, GROUP 65</td>
<td>77009</td>
<td>4203-907</td>
</tr>
<tr>
<td>DB65</td>
<td>UNILAB, GROUP 65</td>
<td>77010</td>
<td>4203-907</td>
</tr>
<tr>
<td>DB65P</td>
<td>UDL, GROUP 11</td>
<td>77011</td>
<td>4203-907</td>
</tr>
<tr>
<td>DB65P</td>
<td>UNILAB, GROUP 11</td>
<td>77012</td>
<td>4203-907</td>
</tr>
<tr>
<td>DB68</td>
<td>UDL, GROUP 68</td>
<td>77013</td>
<td>4203-907</td>
</tr>
<tr>
<td>DB68</td>
<td>UNILAB, GROUP 68</td>
<td>77014</td>
<td>4203-907</td>
</tr>
<tr>
<td>DB681</td>
<td>UDL, GROUP 01</td>
<td>77015</td>
<td>4203-907</td>
</tr>
<tr>
<td>DB681</td>
<td>UNILAB, GROUP 01</td>
<td>77016</td>
<td>4203-907</td>
</tr>
<tr>
<td>DB682</td>
<td>UDL, GROUP 02</td>
<td>77017</td>
<td>4203-907</td>
</tr>
<tr>
<td>DB682</td>
<td>UNILAB, GROUP 02</td>
<td>77018</td>
<td>4203-907</td>
</tr>
<tr>
<td>DB695</td>
<td>UDL, GROUP 05</td>
<td>77019</td>
<td>4203-907</td>
</tr>
<tr>
<td>DB695</td>
<td>UNILAB, GROUP 05</td>
<td>77020</td>
<td>4203-907</td>
</tr>
<tr>
<td>DB698</td>
<td>UDL, GROUP 08</td>
<td>77021</td>
<td>4203-907</td>
</tr>
<tr>
<td>DB698</td>
<td>UNILAB, GROUP 08</td>
<td>77022</td>
<td>4203-907</td>
</tr>
<tr>
<td>DB689</td>
<td>UDL, GROUP 09</td>
<td>77023</td>
<td>4203-907</td>
</tr>
<tr>
<td>DB689</td>
<td>UNILAB, GROUP 09</td>
<td>77024</td>
<td>4203-907</td>
</tr>
<tr>
<td>DB68K</td>
<td>UDL, GROUP 68K</td>
<td>77025</td>
<td>4203-907</td>
</tr>
<tr>
<td>DB68K</td>
<td>UNILAB, GROUP 68K</td>
<td>77026</td>
<td>4203-907</td>
</tr>
<tr>
<td>DB695</td>
<td>UDL, GROUP 85</td>
<td>77027</td>
<td>4203-907</td>
</tr>
<tr>
<td>DB695</td>
<td>UNILAB, GROUP 85</td>
<td>77028</td>
<td>4203-907</td>
</tr>
<tr>
<td>DB696</td>
<td>UDL, GROUP 86</td>
<td>77029</td>
<td>4203-907</td>
</tr>
<tr>
<td>DB696</td>
<td>UNILAB, GROUP 86</td>
<td>77030</td>
<td>4203-907</td>
</tr>
<tr>
<td>DB698</td>
<td>UDL, GROUP 88</td>
<td>77031</td>
<td>4203-907</td>
</tr>
<tr>
<td>DB698</td>
<td>UNILAB, GROUP 88</td>
<td>77032</td>
<td>4203-907</td>
</tr>
<tr>
<td>DB696</td>
<td>UDL, GROUP 96</td>
<td>77033</td>
<td>4203-907</td>
</tr>
<tr>
<td>DB696</td>
<td>UNILAB, GROUP 96</td>
<td>77034</td>
<td>4203-907</td>
</tr>
<tr>
<td>DB696</td>
<td>UDL, GROUP 98</td>
<td>77035</td>
<td>4203-907</td>
</tr>
<tr>
<td>DB696</td>
<td>UNILAB, GROUP 98</td>
<td>77036</td>
<td>4203-907</td>
</tr>
<tr>
<td>DB698</td>
<td>UDL, GROUP 08</td>
<td>77037</td>
<td>4203-907</td>
</tr>
<tr>
<td>DB698</td>
<td>UNILAB, GROUP 08</td>
<td>77038</td>
<td>4203-907</td>
</tr>
<tr>
<td>DB698</td>
<td>UDL, GROUP 08</td>
<td>77039</td>
<td>4203-907</td>
</tr>
<tr>
<td>DB698</td>
<td>UNILAB, GROUP 08</td>
<td>77040</td>
<td>4203-907</td>
</tr>
<tr>
<td>DB698</td>
<td>UDL, GROUP 08</td>
<td>77041</td>
<td>4203-907</td>
</tr>
<tr>
<td>DB698</td>
<td>UNILAB, GROUP 08</td>
<td>77042</td>
<td>4203-907</td>
</tr>
<tr>
<td>DB698</td>
<td>UDL, GROUP 08</td>
<td>77043</td>
<td>4203-907</td>
</tr>
<tr>
<td>DB698</td>
<td>UNILAB, GROUP 08</td>
<td>77044</td>
<td>4203-907</td>
</tr>
<tr>
<td>DS18</td>
<td>UDL, GROUP 18</td>
<td>77045</td>
<td>4203-907</td>
</tr>
<tr>
<td>DS18</td>
<td>UNILAB, GROUP 18</td>
<td>77046</td>
<td>4203-907</td>
</tr>
<tr>
<td>EM6303R</td>
<td>EMULATION MODULE, 6303R</td>
<td>78001</td>
<td>4202-906</td>
</tr>
<tr>
<td>EM63P01</td>
<td>EMULATION MODULE</td>
<td>78002</td>
<td>4202-906</td>
</tr>
<tr>
<td>EM64180</td>
<td>EMULATION MODULE 64180</td>
<td>78003</td>
<td>4202-906</td>
</tr>
<tr>
<td>EM64180</td>
<td>EMULATION MODULE 64180</td>
<td>78004</td>
<td>4202-906</td>
</tr>
<tr>
<td>EM6502</td>
<td>EMULATION MODULE 6502</td>
<td>78005</td>
<td>4202-906</td>
</tr>
<tr>
<td>EM5E2</td>
<td>EMULATION MODULE 6502E2</td>
<td>78006</td>
<td>4202-906</td>
</tr>
<tr>
<td>EM6805P</td>
<td>EMULATION MODULE</td>
<td>78007</td>
<td>4202-906</td>
</tr>
<tr>
<td>EM6809</td>
<td>EMULATION MODULE 6809</td>
<td>78008</td>
<td>4202-906</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>PART ID</td>
<td>REV-RTN #</td>
<td>LIST PRICE</td>
</tr>
<tr>
<td>--------------------------------------------------</td>
<td>---------</td>
<td>-----------</td>
<td>------------</td>
</tr>
<tr>
<td>80416P EMULATION MODULE 60000</td>
<td>78013</td>
<td>4202-906</td>
<td>225.000</td>
</tr>
<tr>
<td>8051P EMULATION MODULE 8051P</td>
<td>78005</td>
<td>4202-906</td>
<td>375.000</td>
</tr>
<tr>
<td>8031 EMULATION MODULE 8031</td>
<td>78008</td>
<td>4202-906</td>
<td>175.000</td>
</tr>
<tr>
<td>8048P EMULATION MODULE 8048P</td>
<td>78006</td>
<td>4202-906</td>
<td>275.000</td>
</tr>
<tr>
<td>8085 EMULATION MODULE 8085</td>
<td>78007</td>
<td>4202-906</td>
<td>175.000</td>
</tr>
<tr>
<td>8086EMAX EMULATION MODULE</td>
<td>78016</td>
<td>4202-906</td>
<td>225.000</td>
</tr>
<tr>
<td>8086MIN EMULATION MODULE</td>
<td>78002</td>
<td>4202-906</td>
<td>225.000</td>
</tr>
<tr>
<td>8088MAX EMULATION MODULE 8088MAX</td>
<td>78017</td>
<td>4202-906</td>
<td>175.000</td>
</tr>
<tr>
<td>8088MIN EMULATION MODULE 8088MIN</td>
<td>78011</td>
<td>4202-906</td>
<td>175.000</td>
</tr>
<tr>
<td>2Z8 EMULATION MODULE</td>
<td>78018</td>
<td>4202-906</td>
<td>175.000</td>
</tr>
<tr>
<td>2Z80 EMULATION MODULE Z80</td>
<td>78004</td>
<td>4202-906</td>
<td>175.000</td>
</tr>
<tr>
<td>1T63P01 MICROTARGET</td>
<td>78507</td>
<td>4202-906</td>
<td>250.000</td>
</tr>
<tr>
<td>1T64180 MICROTARGET</td>
<td>78513</td>
<td>4202-906</td>
<td>200.000</td>
</tr>
<tr>
<td>1T6502 MICROTARGET</td>
<td>78508</td>
<td>4202-906</td>
<td>150.000</td>
</tr>
<tr>
<td>1T68K MICROTARGET</td>
<td>78510</td>
<td>4202-906</td>
<td>200.000</td>
</tr>
<tr>
<td>1T68P05 MICROTARGET</td>
<td>78509</td>
<td>4202-906</td>
<td>200.000</td>
</tr>
<tr>
<td>1T8031 MICROTARGET</td>
<td>78505</td>
<td>4202-906</td>
<td>250.000</td>
</tr>
<tr>
<td>1T8051P MICROTARGET</td>
<td>78502</td>
<td>4202-906</td>
<td>150.000</td>
</tr>
<tr>
<td>1T8085 MICROTARGET</td>
<td>78506</td>
<td>4202-906</td>
<td>175.000</td>
</tr>
<tr>
<td>1T8086MAX MICROTARGET</td>
<td>78504</td>
<td>4202-906</td>
<td>250.000</td>
</tr>
<tr>
<td>1T8086MIN MICROTARGET</td>
<td>78503</td>
<td>4202-906</td>
<td>250.000</td>
</tr>
<tr>
<td>1T8088MAX MICROTARGET</td>
<td>78512</td>
<td>4202-906</td>
<td>200.000</td>
</tr>
<tr>
<td>1T8088MIN MICROTARGETBOARD</td>
<td>78511</td>
<td>4202-906</td>
<td>200.000</td>
</tr>
<tr>
<td>1T28 MICROTARGET</td>
<td>78515</td>
<td>4202-906</td>
<td>175.000</td>
</tr>
<tr>
<td>1T280 MICROTARGET</td>
<td>78514</td>
<td>4202-906</td>
<td>100.000</td>
</tr>
<tr>
<td>F &quot; 18 UDL</td>
<td>79023</td>
<td>4201-909</td>
<td>350.000</td>
</tr>
<tr>
<td>AK9 UNILAB</td>
<td>79523</td>
<td>4201-909</td>
<td>350.000</td>
</tr>
<tr>
<td>PAK63 UNILAB</td>
<td>79027</td>
<td>4201-909</td>
<td>600.000</td>
</tr>
<tr>
<td>PAK63 UNILAB</td>
<td>79527</td>
<td>4201-909</td>
<td>600.000</td>
</tr>
<tr>
<td>PAK6303R UDL</td>
<td>79028</td>
<td>4201-909</td>
<td>550.000</td>
</tr>
<tr>
<td>PAK6303R UNILAB</td>
<td>79528</td>
<td>4201-909</td>
<td>550.000</td>
</tr>
<tr>
<td>PAK6305 UDL</td>
<td>79030</td>
<td>4201-909</td>
<td>550.000</td>
</tr>
<tr>
<td>PAK6305 UNILAB</td>
<td>79530</td>
<td>4201-909</td>
<td>550.000</td>
</tr>
<tr>
<td>PAK6305F UDL</td>
<td>79029</td>
<td>4201-909</td>
<td>600.000</td>
</tr>
<tr>
<td>PAK6305F UNILAB</td>
<td>79529</td>
<td>4201-909</td>
<td>600.000</td>
</tr>
<tr>
<td>PAK63P01 UDL</td>
<td>79005</td>
<td>4201-909</td>
<td>800.000</td>
</tr>
<tr>
<td>PAK63P01 UNILAB</td>
<td>79505</td>
<td>4201-909</td>
<td>800.000</td>
</tr>
<tr>
<td>PAK63P05 UDL</td>
<td>79031</td>
<td>4201-909</td>
<td>650.000</td>
</tr>
<tr>
<td>PAK63P05 UNILAB</td>
<td>79531</td>
<td>4201-909</td>
<td>650.000</td>
</tr>
<tr>
<td>PAK65 UDL</td>
<td>79006</td>
<td>4201-909</td>
<td>550.000</td>
</tr>
<tr>
<td>PAK65 UNILAB</td>
<td>79506</td>
<td>4201-909</td>
<td>550.000</td>
</tr>
<tr>
<td>PAK67/11 UDL</td>
<td>79035</td>
<td>4201-909</td>
<td>550.000</td>
</tr>
<tr>
<td>PAK67/11 UNILAB</td>
<td>79535</td>
<td>4201-909</td>
<td>550.000</td>
</tr>
<tr>
<td>PAK6502 UDL</td>
<td>79033</td>
<td>4201-909</td>
<td>600.000</td>
</tr>
<tr>
<td>PAK6502 UNILAB</td>
<td>79533</td>
<td>4201-909</td>
<td>600.000</td>
</tr>
<tr>
<td>PAK6510 UDL</td>
<td>79034</td>
<td>4201-909</td>
<td>500.000</td>
</tr>
<tr>
<td>PAK6510 UNILAB</td>
<td>79534</td>
<td>4201-909</td>
<td>500.000</td>
</tr>
<tr>
<td>PAK6511Q UDL</td>
<td>79007</td>
<td>4201-909</td>
<td>500.000</td>
</tr>
<tr>
<td>PAK6511Q UNILAB</td>
<td>79507</td>
<td>4201-909</td>
<td>500.000</td>
</tr>
<tr>
<td>PAK6512 UNILAB</td>
<td>79532</td>
<td>4201-909</td>
<td>500.000</td>
</tr>
<tr>
<td>PAK6512 UNILAB</td>
<td>79036</td>
<td>4201-909</td>
<td>500.000</td>
</tr>
<tr>
<td>PAK6541Q UDL</td>
<td>79036</td>
<td>4201-909</td>
<td>500.000</td>
</tr>
<tr>
<td>PAK6541Q UNILAB</td>
<td>79536</td>
<td>4201-909</td>
<td>500.000</td>
</tr>
<tr>
<td>PAK64180 UNILAB</td>
<td>74552</td>
<td>4201-906</td>
<td>700.000</td>
</tr>
<tr>
<td>PART ID</td>
<td>REV-RTN #</td>
<td>LIST PRICE</td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>-----------</td>
<td>------------</td>
<td></td>
</tr>
<tr>
<td>PAK68 UDL</td>
<td>79008</td>
<td>4201-909</td>
<td>500.000</td>
</tr>
<tr>
<td>PAK68 UNILAB</td>
<td>79508</td>
<td>4201-909</td>
<td>500.000</td>
</tr>
<tr>
<td>PAK80008 UDL</td>
<td>79012</td>
<td>4201-909</td>
<td>550.000</td>
</tr>
<tr>
<td>PAK80008 UNILAB</td>
<td>79512</td>
<td>4201-909</td>
<td>550.000</td>
</tr>
<tr>
<td>PAK8010 UDL</td>
<td>79038</td>
<td>4201-909</td>
<td>650.000</td>
</tr>
<tr>
<td>PAK8010 UNILAB</td>
<td>79538</td>
<td>4201-909</td>
<td>650.000</td>
</tr>
<tr>
<td>PAK8055 UDL</td>
<td>79011</td>
<td>4201-909</td>
<td>550.000</td>
</tr>
<tr>
<td>PAK8055 UNILAB</td>
<td>79511</td>
<td>4201-909</td>
<td>550.000</td>
</tr>
<tr>
<td>PAK8055 P UDL</td>
<td>79040</td>
<td>4201-909</td>
<td>750.000</td>
</tr>
<tr>
<td>PAK811 UDL</td>
<td>79009</td>
<td>4201-909</td>
<td>500.000</td>
</tr>
<tr>
<td>PAK811 UNILAB</td>
<td>79509</td>
<td>4201-909</td>
<td>500.000</td>
</tr>
<tr>
<td>PAK811 UDL</td>
<td>79004</td>
<td>4201-909</td>
<td>550.000</td>
</tr>
<tr>
<td>PAK811 UNILAB</td>
<td>79504</td>
<td>4201-909</td>
<td>550.000</td>
</tr>
<tr>
<td>PAK821 UDL</td>
<td>79010</td>
<td>4201-909</td>
<td>500.000</td>
</tr>
<tr>
<td>PAK821 UNILAB</td>
<td>79510</td>
<td>4201-909</td>
<td>500.000</td>
</tr>
<tr>
<td>PAK851 UDL</td>
<td>79039</td>
<td>4201-909</td>
<td>500.000</td>
</tr>
<tr>
<td>PAK851 UNILAB</td>
<td>79539</td>
<td>4201-909</td>
<td>500.000</td>
</tr>
<tr>
<td>PAK899 UDL</td>
<td>79013</td>
<td>4201-909</td>
<td>500.000</td>
</tr>
<tr>
<td>PAK899 UNILAB</td>
<td>79513</td>
<td>4201-909</td>
<td>500.000</td>
</tr>
<tr>
<td>PAK89 UDL</td>
<td>79014</td>
<td>4201-909</td>
<td>800.000</td>
</tr>
<tr>
<td>PAK89 UNILAB</td>
<td>79514</td>
<td>4201-909</td>
<td>800.000</td>
</tr>
<tr>
<td>PAK89 UDL</td>
<td>79037</td>
<td>4201-909</td>
<td>550.000</td>
</tr>
<tr>
<td>PAK89 UNILAB</td>
<td>79537</td>
<td>4201-909</td>
<td>550.000</td>
</tr>
<tr>
<td>PAK805 UNILAB</td>
<td>79540</td>
<td>4201-909</td>
<td>750.000</td>
</tr>
<tr>
<td>PAK800 UDL</td>
<td>79051</td>
<td>4201-909</td>
<td>500.000</td>
</tr>
<tr>
<td>PAK800 UNILAB</td>
<td>79551</td>
<td>4201-909</td>
<td>500.000</td>
</tr>
<tr>
<td>PAK80186 UDL</td>
<td>79042</td>
<td>4201-909</td>
<td>650.000</td>
</tr>
<tr>
<td>PAK80186 UNILAB</td>
<td>79542</td>
<td>4201-909</td>
<td>650.000</td>
</tr>
<tr>
<td>PAK80188 UDL</td>
<td>79043</td>
<td>4201-909</td>
<td>550.000</td>
</tr>
<tr>
<td>PAK80266 UDL</td>
<td>79044</td>
<td>4201-909</td>
<td>650.000</td>
</tr>
<tr>
<td>PAK80266 UNILAB</td>
<td>79543</td>
<td>4201-909</td>
<td>550.000</td>
</tr>
<tr>
<td>PAK80266 UNILAB</td>
<td>79544</td>
<td>4201-909</td>
<td>650.000</td>
</tr>
<tr>
<td>PAK8031 UNILAB</td>
<td>79502</td>
<td>4201-909</td>
<td>600.000</td>
</tr>
<tr>
<td>PAK8046 UDL</td>
<td>79001</td>
<td>4201-909</td>
<td>500.000</td>
</tr>
<tr>
<td>PAK8048 UNILAB</td>
<td>79501</td>
<td>4201-909</td>
<td>500.000</td>
</tr>
<tr>
<td>PAK8048 P UDL</td>
<td>79049</td>
<td>4201-909</td>
<td>550.000</td>
</tr>
<tr>
<td>PAK8048 P UNILAB</td>
<td>79549</td>
<td>4201-909</td>
<td>550.000</td>
</tr>
<tr>
<td>PAK8051 UDL</td>
<td>79002</td>
<td>4201-909</td>
<td>600.000</td>
</tr>
<tr>
<td>PAK8051 P UDL</td>
<td>79003</td>
<td>4201-909</td>
<td>850.000</td>
</tr>
<tr>
<td>PAK8051 P UNILAB</td>
<td>79503</td>
<td>4201-909</td>
<td>850.000</td>
</tr>
<tr>
<td>PAK8060 UDL</td>
<td>79015</td>
<td>4201-909</td>
<td>500.000</td>
</tr>
<tr>
<td>PAK8060 UNILAB</td>
<td>79515</td>
<td>4201-909</td>
<td>500.000</td>
</tr>
<tr>
<td>PAK8085 UDL</td>
<td>79045</td>
<td>4201-909</td>
<td>650.000</td>
</tr>
<tr>
<td>PAK8107 UNILAB</td>
<td>79545</td>
<td>4201-909</td>
<td>650.000</td>
</tr>
<tr>
<td>PAK8107 UNILAB</td>
<td>79545</td>
<td>4201-909</td>
<td>650.000</td>
</tr>
<tr>
<td>PAK8186MAX UDL</td>
<td>79025</td>
<td>4201-909</td>
<td>650.000</td>
</tr>
<tr>
<td>PAK8186MAX UNILAB</td>
<td>79525</td>
<td>4201-909</td>
<td>650.000</td>
</tr>
<tr>
<td>PAK8186MIN UDL</td>
<td>79016</td>
<td>4201-909</td>
<td>800.000</td>
</tr>
<tr>
<td>PAK8186MIN UNILAB</td>
<td>79516</td>
<td>4201-909</td>
<td>800.000</td>
</tr>
<tr>
<td>PAK8188MAX UDL</td>
<td>79026</td>
<td>4201-909</td>
<td>650.000</td>
</tr>
<tr>
<td>PAK8188MAX UNILAB</td>
<td>79526</td>
<td>4201-909</td>
<td>650.000</td>
</tr>
<tr>
<td>PAK8188MIN UDL</td>
<td>79017</td>
<td>4201-909</td>
<td>650.000</td>
</tr>
<tr>
<td>PPAK8188MIN UNILAB</td>
<td>79517</td>
<td>4201-909</td>
<td>650.000</td>
</tr>
<tr>
<td>PAK8097 UDL</td>
<td>79046</td>
<td>4201-909</td>
<td>650.000</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>PART ID</td>
<td>REV-RTN #</td>
<td>LIST PRICE</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------</td>
<td>-----------</td>
<td>------------</td>
</tr>
<tr>
<td>PAK8097 UNILAB</td>
<td>79546</td>
<td>4201-909</td>
<td>650.000</td>
</tr>
<tr>
<td>PAK96 UDL</td>
<td>79018</td>
<td>4201-909</td>
<td>650.000</td>
</tr>
<tr>
<td>PAK96 UNILAB</td>
<td>79518</td>
<td>4201-909</td>
<td>650.000</td>
</tr>
<tr>
<td>PAKPLAIN UDL</td>
<td>79824</td>
<td>4201-909</td>
<td>200.000</td>
</tr>
<tr>
<td>PAKPLAIN UNILAB</td>
<td>79524</td>
<td>4201-909</td>
<td>200.000</td>
</tr>
<tr>
<td>PAKSB UDL</td>
<td>79019</td>
<td>4201-909</td>
<td>500.000</td>
</tr>
<tr>
<td>PAKSB UNILAB</td>
<td>79519</td>
<td>4201-909</td>
<td>500.000</td>
</tr>
<tr>
<td>PAKSBP UDL</td>
<td>79050</td>
<td>4201-909</td>
<td>550.000</td>
</tr>
<tr>
<td>PAKSBP UNILAB</td>
<td>79550</td>
<td>4201-909</td>
<td>550.000</td>
</tr>
<tr>
<td>PAKZ8 UDL</td>
<td>79048</td>
<td>4201-909</td>
<td>650.000</td>
</tr>
<tr>
<td>PAKZ8 UNILAB</td>
<td>79548</td>
<td>4201-909</td>
<td>650.000</td>
</tr>
<tr>
<td>PAKZBO UDL</td>
<td>79021</td>
<td>4201-909</td>
<td>700.000</td>
</tr>
<tr>
<td>PAKZBO UNILAB</td>
<td>79521</td>
<td>4201-909</td>
<td>700.000</td>
</tr>
<tr>
<td>PAKZBO01 UDL</td>
<td>79022</td>
<td>4201-909</td>
<td>650.000</td>
</tr>
<tr>
<td>PAKZBO01 UNILAB</td>
<td>79522</td>
<td>4201-909</td>
<td>650.000</td>
</tr>
<tr>
<td>PAKZBO02 UDL</td>
<td>79041</td>
<td>4201-909</td>
<td>600.000</td>
</tr>
<tr>
<td>PAKZBO02 UNILAB</td>
<td>79541</td>
<td>4201-909</td>
<td>600.000</td>
</tr>
<tr>
<td>PAKZBF UDL</td>
<td>79020</td>
<td>4201-909</td>
<td>550.000</td>
</tr>
<tr>
<td>PAKZBF UNILAB</td>
<td>79520</td>
<td>4201-909</td>
<td>550.000</td>
</tr>
<tr>
<td>PAKZBP64 UDL</td>
<td>79047</td>
<td>4201-909</td>
<td>550.000</td>
</tr>
<tr>
<td>PAKZBP64 UNILAB</td>
<td>79547</td>
<td>4201-909</td>
<td>550.000</td>
</tr>
</tbody>
</table>

180 RECORDS PRINTED
<table>
<thead>
<tr>
<th>DESCRIPTION</th>
<th>PART ID</th>
<th>REV-RTN #</th>
<th>LIST PRICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>UDL128  UDL 128K STD SPEED</td>
<td>82103</td>
<td>4101-902</td>
<td>3590.000</td>
</tr>
<tr>
<td>UDL128  UDL 128K STD SPEED, AUSTRALIAN</td>
<td>82133</td>
<td>4101-902</td>
<td>3590.000</td>
</tr>
<tr>
<td>UDL128  UDL 128K STD SPEED, BRITISH</td>
<td>82113</td>
<td>4101-902</td>
<td>3740.000</td>
</tr>
<tr>
<td>UDL128  UDL 128K STD SPEED, GERMAN</td>
<td>82123</td>
<td>4101-902</td>
<td>3590.000</td>
</tr>
<tr>
<td>UDL128  UDL 128K STD SPEED, THORN</td>
<td>82143</td>
<td>4101-902</td>
<td>3590.000</td>
</tr>
<tr>
<td>UDL128HS UDL 128K HIGH SPEED, AUSTRALIAN</td>
<td>82233</td>
<td>4101-902</td>
<td>4085.000</td>
</tr>
<tr>
<td>UDL128HS UDL 128K HIGH SPEED, BRITISH</td>
<td>82213</td>
<td>4101-902</td>
<td>4235.000</td>
</tr>
<tr>
<td>UDL128HS UDL 128K HIGH SPEED, GERMAN</td>
<td>82223</td>
<td>4101-902</td>
<td>4085.000</td>
</tr>
<tr>
<td>UDL128HS UDL 128K HIGH SPEED, THORN</td>
<td>82243</td>
<td>4101-902</td>
<td>4085.000</td>
</tr>
<tr>
<td>UDL128HS UDL 128K STD SPEED</td>
<td>82203</td>
<td>4101-902</td>
<td>4085.000</td>
</tr>
<tr>
<td>UDL32   UDL 32K STD SPEED</td>
<td>82101</td>
<td>4101-902</td>
<td>2995.000</td>
</tr>
<tr>
<td>UDL32   UDL 32K STD SPEED, AUSTRALIAN</td>
<td>82131</td>
<td>4101-902</td>
<td>2995.000</td>
</tr>
<tr>
<td>UDL32   UDL 32K STD SPEED, BRITISH</td>
<td>82111</td>
<td>4101-902</td>
<td>3145.000</td>
</tr>
<tr>
<td>UDL32   UDL 32K STD SPEED, GERMAN</td>
<td>82121</td>
<td>4101-902</td>
<td>2995.000</td>
</tr>
<tr>
<td>UDL32   UDL 32K STD SPEED, THORN</td>
<td>82141</td>
<td>4101-902</td>
<td>2995.000</td>
</tr>
<tr>
<td>UDL32HS UDL 32K STD SPEED</td>
<td>82201</td>
<td>4101-902</td>
<td>3430.000</td>
</tr>
<tr>
<td>UDL32HS UDL 32K HIGH SPEED, AUSTRALIAN</td>
<td>82231</td>
<td>4101-902</td>
<td>3430.000</td>
</tr>
<tr>
<td>UDL32HS UDL 32K HIGH SPEED, BRITISH</td>
<td>82211</td>
<td>4101-902</td>
<td>3580.000</td>
</tr>
<tr>
<td>UDL32HS UDL 32K HIGH SPEED, GERMAN</td>
<td>82221</td>
<td>4101-902</td>
<td>3430.000</td>
</tr>
<tr>
<td>UDL32HS UDL 32K STD SPEED, THORN</td>
<td>82241</td>
<td>4101-902</td>
<td>3430.000</td>
</tr>
<tr>
<td>UDL64   UDL 64K STD SPEED</td>
<td>82102</td>
<td>4101-902</td>
<td>3390.000</td>
</tr>
<tr>
<td>UDL64   UDL 64K STD SPEED, AUSTRALIAN</td>
<td>82132</td>
<td>4101-902</td>
<td>3390.000</td>
</tr>
<tr>
<td>UDL64   UDL 64K STD SPEED, BRITISH</td>
<td>82112</td>
<td>4101-902</td>
<td>3540.000</td>
</tr>
<tr>
<td>UDL64   UDL 64K STD SPEED, GERMAN</td>
<td>82122</td>
<td>4101-902</td>
<td>3390.000</td>
</tr>
<tr>
<td>UDL64   UDL 64K STD SPEED, THORN</td>
<td>82142</td>
<td>4101-902</td>
<td>3390.000</td>
</tr>
<tr>
<td>UDL64HS UDL 64K HIGH SPEED</td>
<td>82202</td>
<td>4101-902</td>
<td>3855.000</td>
</tr>
<tr>
<td>UDL64HS UDL 64K HIGH SPEED, AUSTRALIAN</td>
<td>82232</td>
<td>4101-902</td>
<td>3855.000</td>
</tr>
<tr>
<td>UDL64HS UDL 64K HIGH SPEED, BRITISH</td>
<td>82212</td>
<td>4101-902</td>
<td>4005.000</td>
</tr>
<tr>
<td>UDL64HS UDL 64K HIGH SPEED, GERMAN</td>
<td>82222</td>
<td>4101-902</td>
<td>3855.000</td>
</tr>
<tr>
<td>UDL64HS UDL 64K HIGH SPEED, THORN</td>
<td>82242</td>
<td>4101-902</td>
<td>3855.000</td>
</tr>
</tbody>
</table>

30 RECORDS PRINTED
<table>
<thead>
<tr>
<th>DESCRIPTION</th>
<th>PART ID</th>
<th>REV-RTN #</th>
<th>LIST PRICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>UN12B UNILAB II 128K STD SPEED, BRITISH</td>
<td>84113</td>
<td>4102-903</td>
<td>5750.000</td>
</tr>
<tr>
<td>UN12B UNILABII 128K STD SPEED, AUSTRAL</td>
<td>84133</td>
<td>4102-903</td>
<td>5580.000</td>
</tr>
<tr>
<td>UN12B UNILABII 128K STD SPEED, DOMESTI</td>
<td>84103</td>
<td>4102-903</td>
<td>5580.000</td>
</tr>
<tr>
<td>UN12B UNILABII 128K STD SPEED, GERMAN</td>
<td>84123</td>
<td>4102-903</td>
<td>5580.000</td>
</tr>
<tr>
<td>UN12B UNILABII 128K STD SPEED, THORN</td>
<td>84143</td>
<td>4102-903</td>
<td>5580.000</td>
</tr>
<tr>
<td>UN12BHS UNILAB II 128K HIGH SPEED, BRIT</td>
<td>84213</td>
<td>4102-903</td>
<td>6130.000</td>
</tr>
<tr>
<td>UN12BHS UNILABII 128K HIGH SPEED AUSTRAL</td>
<td>84233</td>
<td>4102-903</td>
<td>5980.000</td>
</tr>
<tr>
<td>UN12BHS UNILABII 128K HIGH SPEED, THORN</td>
<td>84243</td>
<td>4102-903</td>
<td>5980.000</td>
</tr>
<tr>
<td>UN12BHS UNILABII 128K HIGH SPEED, DOMEST</td>
<td>84203</td>
<td>4102-903</td>
<td>5980.000</td>
</tr>
<tr>
<td>UN12BHS UNILABII 128K HIGH SPEED, GERMAN</td>
<td>84223</td>
<td>4102-903</td>
<td>5980.000</td>
</tr>
<tr>
<td>UN132 UNILAB II 32K STD SPEED, BRITISH</td>
<td>84111</td>
<td>4102-903</td>
<td>5130.000</td>
</tr>
<tr>
<td>UN132 UNILAB II 32K STD SPEED, DOMEST</td>
<td>84101</td>
<td>4102-903</td>
<td>4980.000</td>
</tr>
<tr>
<td>UN132 UNILABII 32K STD SPEED, AUSTRAL</td>
<td>84131</td>
<td>4102-903</td>
<td>4980.000</td>
</tr>
<tr>
<td>UN132 UNILABII 32K STD SPEED, GERMAN</td>
<td>84121</td>
<td>4102-903</td>
<td>4980.000</td>
</tr>
<tr>
<td>UN132 UNILABII 32K STD SPEED, THORN</td>
<td>84141</td>
<td>4102-903</td>
<td>4980.000</td>
</tr>
<tr>
<td>UN132HS UNILABII 32K HIGH SPEED, AUSTRAL</td>
<td>84231</td>
<td>4102-903</td>
<td>5380.000</td>
</tr>
<tr>
<td>UN132HS UNILABII 32K HIGH SPEED, DOMEST</td>
<td>84201</td>
<td>4102-903</td>
<td>5380.000</td>
</tr>
<tr>
<td>UN132HS UNILABII 32K HIGH SPEED, BRITISH</td>
<td>84211</td>
<td>4102-903</td>
<td>5550.000</td>
</tr>
<tr>
<td>UN132HS UNILABII 32K HIGH SPEED, GERMAN</td>
<td>84221</td>
<td>4102-903</td>
<td>5380.000</td>
</tr>
<tr>
<td>UN132HS UNILABII 32K HIGH SPEED, THORN</td>
<td>84241</td>
<td>4102-903</td>
<td>5380.000</td>
</tr>
<tr>
<td>UN164 UNILAB II 64K STD SPEED, BRITISH</td>
<td>84112</td>
<td>4102-903</td>
<td>5530.000</td>
</tr>
<tr>
<td>UN164 UNILAB II 64K STD SPEED, AUSTRAL</td>
<td>84132</td>
<td>4102-903</td>
<td>5380.000</td>
</tr>
<tr>
<td>UN164 UNILABII 64K STD SPEED, DOMESTIC</td>
<td>84102</td>
<td>4102-903</td>
<td>5380.000</td>
</tr>
<tr>
<td>UN164 UNILABII 64K STD SPEED, GERMAN</td>
<td>84122</td>
<td>4102-903</td>
<td>5380.000</td>
</tr>
<tr>
<td>UN164 UNILABII 64K STD SPEED, THORN</td>
<td>84142</td>
<td>4102-903</td>
<td>5380.000</td>
</tr>
<tr>
<td>UN164HS UNILAB II 64K HIGH SPEED, BRITISH</td>
<td>84212</td>
<td>4102-903</td>
<td>5930.000</td>
</tr>
<tr>
<td>UN164HS UNILABII 64K HIGH SPEED, AUSTRAL</td>
<td>84232</td>
<td>4102-903</td>
<td>5780.000</td>
</tr>
<tr>
<td>UN164HS UNILABII 64K HIGH SPEED, DOMESTI</td>
<td>84202</td>
<td>4102-903</td>
<td>5780.000</td>
</tr>
<tr>
<td>UN164HS UNILABII 64K HIGH SPEED, GERMAN</td>
<td>84222</td>
<td>4102-903</td>
<td>5780.000</td>
</tr>
<tr>
<td>UN164HS UNILABII 64K HIGH SPEED, THORN</td>
<td>84242</td>
<td>4102-903</td>
<td>5780.000</td>
</tr>
</tbody>
</table>

30 RECORDS PRINTED

Unilabs
<table>
<thead>
<tr>
<th>DESCRIPTION</th>
<th>PART ID</th>
<th>REV-RTN #</th>
<th>LIST PRICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>PERSONALITY MODULE 2716</td>
<td>73002</td>
<td>4202-906</td>
<td>50.000</td>
</tr>
<tr>
<td>PERSONALITY MODULE 2732A</td>
<td>73004</td>
<td>4202-906</td>
<td>50.000</td>
</tr>
<tr>
<td>PERSONALITY MODULE 2732B, 12V</td>
<td>73014</td>
<td>4202-906</td>
<td>50.000</td>
</tr>
<tr>
<td>PERSONALITY MODULE 27512</td>
<td>73016</td>
<td>4202-906</td>
<td>50.000</td>
</tr>
<tr>
<td>PERS. MOD., 27256, 21V, REV A</td>
<td>73A10</td>
<td>4202-906</td>
<td>50.000</td>
</tr>
<tr>
<td>PERS. MOD. 27256, 21V, REV B</td>
<td>73010</td>
<td>4202-906</td>
<td>50.000</td>
</tr>
<tr>
<td>PERSONALITY MODULE 27256, REV A</td>
<td>73A06</td>
<td>4202-906</td>
<td>50.000</td>
</tr>
<tr>
<td>PERSONALITY MODULE 27256, REV B</td>
<td>73006</td>
<td>4202-906</td>
<td>50.000</td>
</tr>
<tr>
<td>PERSONALITY MODULE 2764, REV A</td>
<td>73A08</td>
<td>4202-906</td>
<td>50.000</td>
</tr>
<tr>
<td>PERSONALITY MODULE 2764, REV B</td>
<td>73008</td>
<td>4202-906</td>
<td>50.000</td>
</tr>
</tbody>
</table>

10 RECORDS PRINTED
<table>
<thead>
<tr>
<th>DESCRIPTION</th>
<th>PART ID</th>
<th>REV-RTN #</th>
<th>LIST PRICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1624 CABLE, EM, 16BIT/24PIN</td>
<td>74905</td>
<td>4202-906</td>
<td>95.000</td>
</tr>
<tr>
<td>C1628 CABLE, EMULAT, 16BIT/28PIN</td>
<td>74906</td>
<td>4202-906</td>
<td>95.000</td>
</tr>
<tr>
<td>C16D CABLE, EMULAT, 16BIT/DIRECT</td>
<td>74907</td>
<td>4202-906</td>
<td>85.000</td>
</tr>
<tr>
<td>C024 CABLE, ROM, 8BIT/24PIN</td>
<td>74900</td>
<td>4202-906</td>
<td>75.000</td>
</tr>
<tr>
<td>C028 CABLE, EMULAT, 8BIT/28PIN</td>
<td>74901</td>
<td>4202-906</td>
<td>75.000</td>
</tr>
<tr>
<td>C0D CABLE, EMULAT, 8BIT/DIRECT</td>
<td>74902</td>
<td>4202-906</td>
<td>85.000</td>
</tr>
<tr>
<td>CA CABLE, ANALY, GROUP A</td>
<td>74703</td>
<td>4202-906</td>
<td>85.000</td>
</tr>
<tr>
<td>CB CABLE, ANALY, GROUP B</td>
<td>74704</td>
<td>4202-906</td>
<td>85.000</td>
</tr>
<tr>
<td>CC CABLE, ANALY, GROUP C</td>
<td>74705</td>
<td>4202-906</td>
<td>85.000</td>
</tr>
<tr>
<td>CD CABLE, ANALY, GROUP D</td>
<td>74706</td>
<td>4202-906</td>
<td>85.000</td>
</tr>
<tr>
<td>CE CABLE, ANALY, GROUP E</td>
<td>74707</td>
<td>4202-906</td>
<td>85.000</td>
</tr>
<tr>
<td>CF CABLE, ANALY, GROUP F</td>
<td>74708</td>
<td>4202-906</td>
<td>85.000</td>
</tr>
<tr>
<td>CG CABLE, ANALY, GROUP G</td>
<td>74709</td>
<td>4202-906</td>
<td>85.000</td>
</tr>
<tr>
<td>CH CABLE, ANALY, GROUP H</td>
<td>74710</td>
<td>4202-906</td>
<td>85.000</td>
</tr>
<tr>
<td>CI CABLE, ANALY, GROUP I</td>
<td>74711</td>
<td>4202-906</td>
<td>85.000</td>
</tr>
<tr>
<td>CK CABLE, ANALY, GROUP K</td>
<td>74712</td>
<td>4202-906</td>
<td>85.000</td>
</tr>
<tr>
<td>CL CABLE, ANALY, GROUP L</td>
<td>74713</td>
<td>4202-906</td>
<td>85.000</td>
</tr>
<tr>
<td>CM CABLE, ANALY, GROUP M</td>
<td>74714</td>
<td>4202-906</td>
<td>150.000</td>
</tr>
<tr>
<td>CN CABLE, ANALY, GROUP N</td>
<td>74715</td>
<td>4202-906</td>
<td>85.000</td>
</tr>
<tr>
<td>CP CABLE, ANALY, GROUP P</td>
<td>74716</td>
<td>4202-906</td>
<td>85.000</td>
</tr>
<tr>
<td>CQ CABLE, ANALY, GROUP Q</td>
<td>74717</td>
<td>4202-906</td>
<td>85.000</td>
</tr>
<tr>
<td>CR CABLE, ANALY, GROUP R</td>
<td>74718</td>
<td>4202-906</td>
<td>85.000</td>
</tr>
<tr>
<td>CS CABLE, ANALY, GROUP S</td>
<td>74719</td>
<td>4202-906</td>
<td>85.000</td>
</tr>
<tr>
<td>CT CABLE, ANALY, GROUP T</td>
<td>74720</td>
<td>4202-906</td>
<td>85.000</td>
</tr>
<tr>
<td>64180 UNILAB</td>
<td>74552</td>
<td>4201-909</td>
<td>700.000</td>
</tr>
<tr>
<td>UNIUPT UDL UPGRADE TO UNILAB</td>
<td>74322</td>
<td>4203-907</td>
<td>1985.000</td>
</tr>
<tr>
<td>UPGRADE 128K STD TO 128K HIGH SPEED</td>
<td>74318</td>
<td>4202-906</td>
<td>600.000</td>
</tr>
<tr>
<td>UPGRADE 32 STD TO 128 HIGH SPEED</td>
<td>74314</td>
<td>4202-906</td>
<td>1200.000</td>
</tr>
<tr>
<td>UPGRADE 32 STD TO 128 STD</td>
<td>74313</td>
<td>4202-906</td>
<td>800.000</td>
</tr>
<tr>
<td>UPGRADE 32 STD TO 64 HIGH SPEED</td>
<td>74310</td>
<td>4202-906</td>
<td>600.000</td>
</tr>
<tr>
<td>UPGRADE 32 STD TO 64 STD</td>
<td>74312</td>
<td>4202-906</td>
<td>1000.000</td>
</tr>
<tr>
<td>UPGRADE 32K HIGH SPEED TO 128K H.S.</td>
<td>74311</td>
<td>4202-906</td>
<td>600.000</td>
</tr>
<tr>
<td>UPGRADE 32K HIGH SPEED TO 64K HIGH SPEED</td>
<td>74319</td>
<td>4202-906</td>
<td>800.000</td>
</tr>
<tr>
<td>UPGRADE 64 STD TO 64 HIGH SPEED</td>
<td>74315</td>
<td>4202-906</td>
<td>600.000</td>
</tr>
<tr>
<td>UPGRADE 64K HIGH SPEED TO 128K H.S.</td>
<td>74317</td>
<td>4202-906</td>
<td>400.000</td>
</tr>
<tr>
<td>UPGRADE 64K STD TO 128K HIGH SPEED</td>
<td>74316</td>
<td>4202-906</td>
<td>800.000</td>
</tr>
<tr>
<td>UPGRADE 64K STD TO 128K STD</td>
<td>74316</td>
<td>4202-906</td>
<td>400.000</td>
</tr>
</tbody>
</table>

38 RECORDS PRINTED
<table>
<thead>
<tr>
<th>PART DESCRIPTION</th>
<th>PART ID</th>
<th>STD COST</th>
<th>LIST PRICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>16 PIN DIP CLIP</td>
<td>21036</td>
<td>4202-906</td>
<td>$15.00</td>
</tr>
<tr>
<td>(note: usually shipped in units of 5)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40PINCLIP 40 PIN DIP CLIP</td>
<td>21037</td>
<td>4202-906</td>
<td>55.00</td>
</tr>
<tr>
<td>48PINCLIP 48 PIN DIP CLIP</td>
<td>21038</td>
<td>4202-906</td>
<td>65.00</td>
</tr>
<tr>
<td>64P70MADPT 64PIN 70MIL MPU ADAPTR</td>
<td>74008</td>
<td>4202-906</td>
<td>100.00</td>
</tr>
<tr>
<td>64PINCLIP 64 PIN DIP CLIP</td>
<td>21046</td>
<td>4202-906</td>
<td>95.00</td>
</tr>
<tr>
<td>E220 220VAC/50HZ POWER SUPPLY</td>
<td>40003</td>
<td>4202-906</td>
<td>150.00</td>
</tr>
<tr>
<td>PB40SK 40 PIN TEST SOCKET</td>
<td>21063</td>
<td>4202-906</td>
<td>95.00</td>
</tr>
<tr>
<td>PB48ADAPT ADAPTER 48PIN PIGGYBACK</td>
<td>21062</td>
<td>4202-906</td>
<td>96.00</td>
</tr>
<tr>
<td>FGADIP ADAPTER 64PIN F5A-TD-DIP</td>
<td>74001</td>
<td>4202-906</td>
<td>275.00</td>
</tr>
<tr>
<td>PFA PROG. PERFORMANCE ANALYZER</td>
<td>87001</td>
<td>4203-907</td>
<td>495.00</td>
</tr>
<tr>
<td>PROGO UNILAB PROGRAMMERS GUIDE</td>
<td>87002</td>
<td>4301-901</td>
<td>35.00</td>
</tr>
<tr>
<td>QUIPADAPT 64 PIN QUIP ADAPTOR</td>
<td>74009</td>
<td>4202-906</td>
<td>60.00</td>
</tr>
<tr>
<td>UDLMAN UDL MANUAL</td>
<td>81040</td>
<td>4301-901</td>
<td>25.00</td>
</tr>
<tr>
<td>UN12MAN UNILAB MANUAL</td>
<td>83040</td>
<td>4301-901</td>
<td>100.00</td>
</tr>
<tr>
<td>UPGRADE/B UDL REV.B UPGRADE</td>
<td>74323</td>
<td>4202-906</td>
<td>99.00</td>
</tr>
<tr>
<td>XASM CROSS ASSEMBLER</td>
<td>SEE ANNE</td>
<td>4202-906</td>
<td></td>
</tr>
</tbody>
</table>
HOW PCBASED ENGINEERING INSTRUMENTATION SPEEDS
MICROPROCESSOR DEVELOPMENT TIME

by Thomas R Blakeslee, Orion Instruments Inc.
Redwood City, CA

Personal computers have become so cost effective that it is
now practical to put a complete, PC-based, integrated micro-
processor development system on the engineer's desk. With
this power, project development time can be reduced signifi-
cantly compared to traditional methodologies.

With the increased use of microprocessors in development
projects, debugging time is becoming more and more of a major
issue. While standard debugging techniques on conventional
development systems can get the job done, weeks or months are
often wasted tracking down subtle bugs.

With the Integrated Instrumentation (I) approach, a bus
state analyzer is integrated into the development system so that
the repertoire of debugging techniques is greatly expanded. In
addition to the normal in-circuit-emulator techniques of setting
breakpoints and single-stepping, you can now record real-time
traces of program operation without stopping the processor.
Complex triggers can be set up based on data, cycle types, and
address ranges to capture precisely the circumstances which led
to the undesired condition.

Orion's Universal Development Laboratory, the UDL, and now
the new generation UniLab II model, provides all the hardware
necessary to convert a personal computer into a complete I
development system. This system includes real-time in-circuit
emulation, a 48-channel bus state analyzer, an EPROM programmer, and a stimulus generator. All four instruments are controlled by an integrated control program running on the personal computer via an RS-232 port. Since only command and control data is passed over the RS232 link, system response is instantaneous from the user's perspective.

FINDING NASTY BUGS

If we analyze where the time is spent in real development projects, we usually find that the initial design and debugging is not usually the real problem. Generally schedules start slipping when an inordinate amount of time is spent on one or more bugs which are really nasty in that they occur at mysterious times which are impossible to define as simple breakpoints. This is where the built-in bus state analyzer in an I development system can save the day.

A bus state analyzer is essentially a logic analyzer which is optimized for recording a trace of microprocessor bus cycles in real-time. It differs from the trace buffer found on many emulators in that it can record bus cycles on the fly, without stopping the target processor at a breakpoint. It also has powerful triggering logic capable of triggering on much more than just address breakpoints. Since the trigger specification determines which portion of program operation you will view, good trigger logic should make it possible to trigger on the symptom of the bug rather than just at a specific address.

The analyzer in the UniLab II provides four complete sets of truth-tables so that triggers and qualifiers can be defined which
look for any logical function at each of the 6 input bytes. It is thus possible to trigger on specific cycle types, being inside or outside of specific address ranges, specific data combinations, and configurations of input and output signals.

All of the normal emulator debugging functions such as single-stepping, stopping at a breakpoint, and displaying and changing target registers and memory are also available from the same integrated control program, so you can use the analyzer or the emulator as required. Program and symbol table loading, PROM programming, and stimulus generator commands are also integrated into the same control program.

In a conventional development system, emulator breakpoints are the main tool used in finding program bugs. The reason that nasty bugs take so long to find on such systems is that the bug is often in an entirely unexpected part of the program, so setting breakpoints at the expected places turns out to be a waste of time. With a built-in analyzer, you can generally find the bug quickly by triggering the analyzer on the symptom.

For example, if out-of-range data is being written to a particular memory location, you can define an analyzer trigger to look for the symptom, then look at the trace to see what part of the program wrote the bad data. Generally the trace itself shows you what the bug was, but if you want to single-step in the defective part of the program, you have now identified where to set the breakpoint and what to look for.

Generally trigger specifications are entered in commands which resemble plain english. For example, WRITE FO TO FF DATA
8000 TO 8080 ADR will cause the analyzer to trigger if data greater than FO is written to any address between 8000 and 8080. Note that since this trigger is a description of the symptom itself, it will find the problem even if it comes from unexpected parts of the program or even defective hardware.

ABOUT THE AUTHOR

Thomas R. Blakeslee is the founder and V.P., Research and Development at Orion Instruments, and a leading expert on digital systems design. His textbook titled Digital Design with Standard MSI and LSI (Wiley Interscience) has been adopted as a text by over 75 colleges and universities. He is also author of The Right Brain published by Doubleday. He graduated from Caltech in 1962 and before Orion was a founder and Engineering Vice-President at Logisticon, Inc. in Sunnyvale, California.
Test hardware helps pinpoint software bugs

Setting breakpoints through the use of software debugging tools can solve some of your problems, but other bugs are more difficult to isolate and fix. Your chances of doing so improve considerably when you add an in-circuit emulator and a bus-state analyzer to your debugging arsenal.

Thomas R Blakeslee and John S James, Orion Instruments

Hardware-assisted software debugging of programs can help you track down quickly the hard-to-find or subtle bugs that might otherwise delay a project for weeks or even months. Typical examples include interrupt-related and other timing-critical bugs that you can’t reproduce; errors that appear only while the system is running at full speed; and system crashes that occur very infrequently (often only once in days or months) and that wipe out all evidence of what went wrong.

Using traditional, software-only debugging techniques, you may spend weeks looking for the bug, and even then you may never find it at all. Inspection of the source-code listings may not tell you anything useful, and trying to check progress of the program is a slow business because you have to set breakpoints by trial-and-error methods. Besides, breakpoints have a number of limitations.

First, if the program never reaches a breakpoint, but instead ends up in a loop or crashes, there may be no indication of how the program reached the point at which it looped or crashed, and what operations caused that event. Second, you can only trap instruction execution points; you have no way of trapping other bug symptoms, such as the writing of garbage to the memory. Third, breakpoints don’t give you any record of the machine activity leading up to the breakpoint (or the crash). Finally, breakpoints stop all processing, so the information they give you is inadequate to identify bugs related to real-time interactions.

Hardware extends breakpoint capability

This is not to say that breakpoints are useless; on the contrary, they are quite adequate for finding some types of bugs. When breakpoints are appropriate, you can make them more effective by using an in-circuit emulator, which provides all of the basic breakpoint functions of a software debugger (stopping at a breakpoint, single-stepping, and displaying and changing registers and memory).

The emulator has several advantages over the software debugger. First, the emulator can work with ROM-resident programs. Second, it retains control at all times. Consequently, it can prevent the program under test from overwriting or destroying either the
Using traditional, software-only debugging techniques, you may spend weeks looking for a bug, and even then you may never find it at all.

### LISTING 1

<table>
<thead>
<tr>
<th>cy#</th>
<th>CONT</th>
<th>ADR</th>
<th>DATA</th>
<th>JR NZ, START-LOOP RET</th>
</tr>
</thead>
<tbody>
<tr>
<td>-5</td>
<td>B7</td>
<td>00A6</td>
<td>20FA</td>
<td></td>
</tr>
<tr>
<td>-3</td>
<td>B7</td>
<td>00A8</td>
<td>C9</td>
<td></td>
</tr>
<tr>
<td>-2</td>
<td>F7</td>
<td>17FE</td>
<td>FF read</td>
<td></td>
</tr>
<tr>
<td>-1</td>
<td>F7</td>
<td>17FF</td>
<td>FF read</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>B7</td>
<td>FFFF</td>
<td>1831</td>
<td>JR 32</td>
</tr>
</tbody>
</table>

debugger or the operating system. Also, you can stop the program under test at any time to examine registers and memory—a facility that very few debuggers can provide.

When the bug is so complex or subtle that you have no idea where to set breakpoints, a hardware bus-state analyzer can be even more helpful. The analyzer continuously records bus cycles while the program runs at full speed. It saves all bus information, up to the limit of the available trace-buffer memory, which is usually hundreds of cycles.

**Bus-state analyzer tests all cycle types**

Meanwhile, the analyzer tests all cycles, looking for any requested combinations or sequences of values. For example, if the program is writing bad data into an array or any other part of memory, you could have the analyzer test for a write operation that specifies a range or set of data values written into a range of addresses that correspond to the array. When the conditions you specified occur, the analyzer will trigger and stop collecting new bus cycles. Then you can examine the trace, using such tools as a symbolic disassembler, to see not only what instructions wrote the bad data, but also what happened before the instructions were executed.

The trigger specification can involve more than a single cycle. For example, you could specify two or more independent tests that must occur on sequential bus cycles. More commonly, a delay count delays the trigger for a given number of cycles after the other conditions are met; this count allows you to see what happens after the triggering event, as well as before it. Other analyzer capabilities include filtering (selecting only certain cycles) and counting the exact number of cycles between two points of a program.

Analyzers watch the program run at full speed, with complete transparency. What’s more, the methodology of triggering on the symptom of the bug avoids the guesswork of where to place breakpoints, which must in any case stop the program in order to be useful. An integrated analyzer-emulator can provide the best of both worlds: It can trigger in response to the symptoms of a problem and use a nonmaskable interrupt (NMI) to simulate a breakpoint at that time. Also, integrated analyzer-emulators provide most or all of the following features:

- The analyzer can cause a scope or other external equipment to trigger as a result of any of the logical tests mentioned previously.
- You can progressively append new tests to the trigger specification, and you can display the cumulative specification at any time.
- You can save the current state of the work as a disk file, and later continue from where the session left off. The file can include macros specifying sequential software tests for automated testing of hardware.
- An analyzer can run unattended for weeks if necessary, perhaps at a customer’s site. When a problem arises, the analyzer will save the trace for local examination, or for transmission by telephone.
- An analyzer with automatic reset can restart the system after a crash. Adding new tests to the trigger specification or inspecting a different area of the trace after each crash will add new data to help you identify the problem.

**See the analyzer in action**

Two examples show how to use an integrated analyzer-emulator to find the cause of a crash and how to check that a loop is operating correctly. The first example is the debugging of a Z80 program that has a tendency to work properly for about a minute and then crash. The program code occupies locations 0 through 1000<sub>HEX</sub>; RAM occupies locations 1800<sub>HEX</sub> through 1FFF<sub>HEX</sub>.  

EDN September 4, 1986
### LISTING 2

<table>
<thead>
<tr>
<th>cy# CONT</th>
<th>ADR</th>
<th>DATA</th>
</tr>
</thead>
<tbody>
<tr>
<td>-5 B7</td>
<td>00A6</td>
<td>20FA JR NZ,START-LOOP</td>
</tr>
<tr>
<td>-3 B7</td>
<td>00A8</td>
<td>C9  RET</td>
</tr>
<tr>
<td>-2 F7</td>
<td>17FF</td>
<td>FF read</td>
</tr>
<tr>
<td>-1 F7</td>
<td>17FF</td>
<td>FF read</td>
</tr>
<tr>
<td>0 B7</td>
<td>FFFF</td>
<td>1831 JR 32</td>
</tr>
</tbody>
</table>

### LISTING 3

<table>
<thead>
<tr>
<th>cy# CONT</th>
<th>ADR</th>
<th>DATA</th>
</tr>
</thead>
<tbody>
<tr>
<td>00A8 C9</td>
<td>B7</td>
<td>00A8 C9 RET</td>
</tr>
<tr>
<td>18FA 90 read</td>
<td>F7</td>
<td>18FA 90 read</td>
</tr>
<tr>
<td>00A8 C9</td>
<td>B7</td>
<td>00A8 C9 RET</td>
</tr>
<tr>
<td>18F8 90 read</td>
<td>F7</td>
<td>18F8 90 read</td>
</tr>
<tr>
<td>00A8 C9</td>
<td>B7</td>
<td>00A8 C9 RET</td>
</tr>
<tr>
<td>18F6 90 read</td>
<td>F7</td>
<td>18F6 90 read</td>
</tr>
<tr>
<td>00A8 C9</td>
<td>B7</td>
<td>00A8 C9 RET</td>
</tr>
<tr>
<td>18F4 90 read</td>
<td>F7</td>
<td>18F4 90 read</td>
</tr>
<tr>
<td>00A8 C9</td>
<td>B7</td>
<td>00A8 C9 RET</td>
</tr>
<tr>
<td>18F2 90 read</td>
<td>F7</td>
<td>18F2 90 read</td>
</tr>
<tr>
<td>00A8 C9</td>
<td>B7</td>
<td>00A8 C9 RET</td>
</tr>
<tr>
<td>18F0 90 read</td>
<td>F7</td>
<td>18F0 90 read</td>
</tr>
<tr>
<td>00A8 C9</td>
<td>B7</td>
<td>00A8 C9 RET</td>
</tr>
<tr>
<td>18EE 90 read</td>
<td>F7</td>
<td>18EE 90 read</td>
</tr>
<tr>
<td>00A8 C9</td>
<td>B7</td>
<td>00A8 C9 RET</td>
</tr>
<tr>
<td>18EC 90 read</td>
<td>F7</td>
<td>18EC 90 read</td>
</tr>
</tbody>
</table>

The first step is to set up the analyzer to look for instruction-fetch cycles at addresses outside the code area. When the program crashes, the screen displays the instruction trace (ie, the contents of the trace buffer) shown in Listing 1. Sure enough, the program is attempting to execute an instruction at FFFFHex; the RET instruction is trying to read the stack at 17FFHex, where there is no RAM. The stack pointer was initialized to 18FFHex, so it appears the bug is a stack overflow.

The second step is to use the analyzer's start-and-repeat feature to see whether the error is always the same. The command 5 SR starts the analyzer and target program, displays five lines of the triggered display, then resets the target system and restarts both the target program and the analyzer. This sequence continues indefinitely until you stop it by pressing a key. In the case of an intermittent crash, you could send the screen output to a printer or to a disk file, in order to gather, over a period of hours or days, data on all conditions leading up to a crash. The resulting trace (Listing 2) confirms that the crash does occur in the same way every time.

EDN September 4, 1986
In the case of an intermittent crash, you can send the screen output to a printer or a disk file in order to gather data on conditions leading up to a crash.

The third step is to use the analyzer’s filter feature to obtain a better understanding of the stack-overflow problem. Filtering means that the analyzer examines the inputs in real time and keeps only certain bus cycles in the trace buffer. You could also include the cycles after each of the selected ones. In this case, you’d use these features to look at the RET instruction at address A8H and the stack address accessed by that instruction. The trace (Listing 3) shows that each time the program reaches address A8H, the stack pointer address decreases, indicating that at some point the program is pushing a 16-bit value onto the stack but never popping it off again. You know this event is occurring sometime after the program reaches address A8H, so you can ask for a trace that triggers on that address and shows three cycles before the trigger and 10 cycles after it.

The resulting trace (Listing 4) shows that there’s a Push command at address 8BH that could be the cause of the problem; reference to the program listing shows that, in fact, the Push instruction doesn’t belong there. You can test this idea immediately by storing a zero (a no-op instruction) to memory location 8BH with a simple monitor command. After making that change,

**LISTING 4**

<table>
<thead>
<tr>
<th>cy#</th>
<th>CONT</th>
<th>ADR</th>
<th>DATA</th>
<th>ADDR</th>
<th>DATA</th>
</tr>
</thead>
<tbody>
<tr>
<td>-5</td>
<td>B7</td>
<td>00A3</td>
<td>20FD</td>
<td>JR NZ,START-LOOP</td>
<td></td>
</tr>
<tr>
<td>-3</td>
<td>B7</td>
<td>00A5</td>
<td>25</td>
<td>DEC H</td>
<td></td>
</tr>
<tr>
<td>-2</td>
<td>B7</td>
<td>00A6</td>
<td>20FA</td>
<td>JR NZ,START-LOOP</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>B7</td>
<td>00A8</td>
<td>C9</td>
<td>RET</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>F7</td>
<td>18FA</td>
<td>8F read</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>F7</td>
<td>18FB</td>
<td>00 read</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>B7</td>
<td>008F</td>
<td>C38800</td>
<td>JP SEND-OUT</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>B7</td>
<td>SEND-OUT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>5F</td>
<td>0088</td>
<td>379</td>
<td>OUT (LEDS),A</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>B7</td>
<td>8079</td>
<td>80 out</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>B7</td>
<td>008A</td>
<td>0F</td>
<td>RRCA</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>D7</td>
<td>008B</td>
<td>D5</td>
<td>PUSH DE</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>D7</td>
<td>18FB</td>
<td>40 write</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>B7</td>
<td>18FA</td>
<td>20 write</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>B7</td>
<td>008C</td>
<td>CDA000</td>
<td>CALL DELAY-SUB</td>
<td></td>
</tr>
</tbody>
</table>

**LISTING 5**

<table>
<thead>
<tr>
<th>cy#</th>
<th>CONT</th>
<th>ADR</th>
<th>DATA</th>
<th>ADDR</th>
<th>DATA</th>
</tr>
</thead>
<tbody>
<tr>
<td>-5</td>
<td>B7</td>
<td>00A3</td>
<td>20FD</td>
<td>JR NZ,START-LOOP</td>
<td></td>
</tr>
<tr>
<td>-3</td>
<td>B7</td>
<td>00A5</td>
<td>25</td>
<td>DEC H</td>
<td></td>
</tr>
<tr>
<td>-2</td>
<td>B7</td>
<td>00A6</td>
<td>20FA</td>
<td>JR NZ,START-LOOP</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>B7</td>
<td>00A8</td>
<td>C9</td>
<td>RET</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>F7</td>
<td>18FC</td>
<td>8F read</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>F7</td>
<td>18FD</td>
<td>00 read</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>B7</td>
<td>008F</td>
<td>C38800</td>
<td>JP SEND-OUT</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>B7</td>
<td>SEND-OUT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>5F</td>
<td>0088</td>
<td>379</td>
<td>OUT (LEDS),A</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>B7</td>
<td>8079</td>
<td>80 out</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>B7</td>
<td>008A</td>
<td>0F</td>
<td>RRCA</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>B7</td>
<td>008B</td>
<td>00</td>
<td>NOP</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>B7</td>
<td>008C</td>
<td>CDA000</td>
<td>CALL DELAY-SUB</td>
<td></td>
</tr>
</tbody>
</table>
the program seems to run correctly (Listing 5), indicating that the intrusive Push instruction was in fact the reason for the crash.

You now have a small hierarchy of choices: You can immediately burn an EPROM that contains the modified program or save the patched program to a disk file directly from the emulator memory, or you can enter the DOS command to return to PC-DOS so that you can edit and reassemble the program.

If you prefer to fix several bugs before correcting the source code, you can invoke a logging feature that will automatically send just your patches (not the rest of the screen output) to the printer. You can safely continue the session after making each patch and then later correct the source code from the fixes recorded in the printed log.

A second example shows how you use the cycle-counting feature of an analyzer-emulator to check a

### Listing 6

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>DELAY-SUB</td>
<td>00A0 21FF40</td>
</tr>
<tr>
<td>START-LOOP</td>
<td>00A3 2D</td>
</tr>
<tr>
<td></td>
<td>00A4 20FD</td>
</tr>
<tr>
<td></td>
<td>00A6 25</td>
</tr>
<tr>
<td></td>
<td>00A7 20FA</td>
</tr>
<tr>
<td></td>
<td>00A9 C9</td>
</tr>
</tbody>
</table>

### Listing 7

(a) 

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AF=8051 (SZ-pNC) BC=</td>
<td>O DE= 20 HL=FFFF IX=1234 IY=5678 SP=18FC PC=A0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(next step)</td>
</tr>
<tr>
<td>DELAY-SUB</td>
<td>00A0 21FF40</td>
<td>LD HL,40FF</td>
</tr>
</tbody>
</table>

(b) 

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NMI</td>
<td>AF=8051 (SZ-pNC) BC=</td>
<td>O DE= 20 HL=40FF IX=1234 IY=5678 SP=18FC PC=A3</td>
</tr>
<tr>
<td></td>
<td>START-LOOP 00A3 2D</td>
<td>DEC L</td>
</tr>
<tr>
<td>NMI</td>
<td>AF=80AB (SZ-pNC) BC=</td>
<td>O DE= 20 HL=40FE IX=1234 IY=5678 SP=18FC PC=A4</td>
</tr>
<tr>
<td></td>
<td>00A4 20FD</td>
<td>JR NZ,START-LOOP</td>
</tr>
<tr>
<td>NMI</td>
<td>AF=80AB (SZ-pNC) BC=</td>
<td>O DE= 20 HL=40FE IX=1234 IY=5678 SP=18FC PC=A3</td>
</tr>
<tr>
<td></td>
<td>START-LOOP 00A3 2D</td>
<td>DEC L</td>
</tr>
<tr>
<td>NMI</td>
<td>AF=80AB (SZ-pNC) BC=</td>
<td>O DE= 20 HL=40FD IX=1234 IY=5678 SP=18FC PC=A4</td>
</tr>
<tr>
<td></td>
<td>00A4 20FD</td>
<td>JR NZ,START-LOOP</td>
</tr>
<tr>
<td>NMI</td>
<td>AF=80AB (SZ-pNC) BC=</td>
<td>O DE= 20 HL=40FD IX=1234 IY=5678 SP=18FC PC=A3</td>
</tr>
<tr>
<td></td>
<td>START-LOOP 00A3 2D</td>
<td>DEC L</td>
</tr>
<tr>
<td>NMI</td>
<td>AF=80AB (SZ-pNC) BC=</td>
<td>O DE= 20 HL=40FC IX=1234 IY=5678 SP=18FC PC=A4</td>
</tr>
<tr>
<td></td>
<td>00A4 20FD</td>
<td>JR NZ,START-LOOP</td>
</tr>
</tbody>
</table>

EDN September 4, 1986
You can safely continue a session after making each patch and then later correct the source code from the fixes recorded in the printed log.

delay routine very quickly. Listing 6 shows the listing of a typical routine consisting of an inner loop that runs FF\textsubscript{HEX} times, and an outer loop that runs 40\textsubscript{HEX} times. This listing was disassembled directly from RAM memory in the emulator, because when you're testing a program, the emulator must keep full control of execution. The emulator can't exercise that control if the program resides in ROM, so the emulator always copies the program from ROM into its own RAM for test execution.

If you're using only software debugging methods, you'll have to single-step through the loop, examining the registers at each step to ensure that the correct number of iterations are executed. To single-step through the loop, you must first have the software debugger obtain "breakpoint control" by executing a

### Listing 8

<table>
<thead>
<tr>
<th>Cy#</th>
<th>Cont</th>
<th>ADR</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>-4</td>
<td>B7</td>
<td>00A3</td>
<td>2D</td>
</tr>
<tr>
<td>-3</td>
<td>B7</td>
<td>00A4</td>
<td>20FD</td>
</tr>
<tr>
<td>-1</td>
<td>B7</td>
<td>00A3</td>
<td>2D</td>
</tr>
<tr>
<td>0</td>
<td>B7</td>
<td>00A4</td>
<td>20FD</td>
</tr>
<tr>
<td>2</td>
<td>B7</td>
<td>00A6</td>
<td>25</td>
</tr>
<tr>
<td>3</td>
<td>B7</td>
<td>00A7</td>
<td>20FA</td>
</tr>
<tr>
<td>5</td>
<td>B7</td>
<td>00A3</td>
<td>2D</td>
</tr>
<tr>
<td>6</td>
<td>B7</td>
<td>00A4</td>
<td>20FD</td>
</tr>
<tr>
<td>8</td>
<td>B7</td>
<td>00A3</td>
<td>2D</td>
</tr>
<tr>
<td>9</td>
<td>B7</td>
<td>00A4</td>
<td>20FD</td>
</tr>
</tbody>
</table>

### Listing 9

<table>
<thead>
<tr>
<th>Cy#</th>
<th>Cont</th>
<th>ADR</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>-5</td>
<td>F7</td>
<td>00A5</td>
<td>FD read</td>
</tr>
<tr>
<td>-4</td>
<td>B7</td>
<td>00A3</td>
<td>2D</td>
</tr>
<tr>
<td>-3</td>
<td>B7</td>
<td>00A4</td>
<td>20FD</td>
</tr>
<tr>
<td>-1</td>
<td>B7</td>
<td>00A6</td>
<td>25</td>
</tr>
<tr>
<td>0</td>
<td>B7</td>
<td>00A7</td>
<td>20FA</td>
</tr>
<tr>
<td>2</td>
<td>B7</td>
<td>00A9</td>
<td>C9</td>
</tr>
<tr>
<td>3</td>
<td>F7</td>
<td>18FC</td>
<td>90 read</td>
</tr>
<tr>
<td>4</td>
<td>F7</td>
<td>18FD</td>
<td>00 read</td>
</tr>
<tr>
<td>5</td>
<td>B7</td>
<td>0090</td>
<td>C38800</td>
</tr>
<tr>
<td>8</td>
<td>B7</td>
<td>0088</td>
<td>D379</td>
</tr>
<tr>
<td>A</td>
<td>5F</td>
<td>8079</td>
<td>80 out</td>
</tr>
<tr>
<td>B</td>
<td>B7</td>
<td>008A</td>
<td>0F</td>
</tr>
<tr>
<td>C</td>
<td>B7</td>
<td>008B</td>
<td>14</td>
</tr>
<tr>
<td>D</td>
<td>B7</td>
<td>008C</td>
<td>00</td>
</tr>
<tr>
<td>E</td>
<td>B7</td>
<td>008D</td>
<td>CDA000</td>
</tr>
<tr>
<td>11</td>
<td>D7</td>
<td>18FD</td>
<td>00 write</td>
</tr>
<tr>
<td>12</td>
<td>D7</td>
<td>18FC</td>
<td>90 write</td>
</tr>
<tr>
<td>13</td>
<td>B7</td>
<td>00A0</td>
<td>21FF40</td>
</tr>
<tr>
<td>16</td>
<td>B7</td>
<td>00A3</td>
<td>2D</td>
</tr>
</tbody>
</table>

EDN September 4, 1986
A quick and elegant way to make sure that a loop works properly is to use an analyzer command to count the number of cycles between two points in the program.

breakpoint at address A0\text{HEX}; you'll see the display shown in Listing 7a. As you can see from the contents of the program counter (PC), the instruction at address A0\text{HEX} has not yet been executed.

Once you've established debug control, you can step through the program by pressing a function key that generates an NMI, thereby allowing the program to execute just one instruction. You can continue stepping through the program in this manner for as long as you like (Listing 7b), but you will soon find that going through tens of thousands of steps quickly gets boring.

Analyzer counts program cycles

A much quicker and more elegant way to make sure that the loop works properly is to use an analyzer command to count the number of cycles between two points in the program. If, for example, you enter the command "A0 A6 CYCLES?", the response will be "= 300", indicating that 300\text{HEX} cycles were executed between addresses A0\text{HEX} and A6\text{HEX}. The LD HL instruction takes three cycles, and the two instructions that compose the inner loop take a total of three cycles. The total for the inner loop should be 3+(3\times\text{FF\text{HEX}})=300\text{HEX}; the display thus indicates that the inner loop works as desired.

To check the operation of the entire delay subroutine, just enter "A0 A9 CYCLES?"; the response is "= C0C0", which is 40\times300\text{HEX}. This response confirms that the outer loop (which includes the inner loop plus three cycles) was executed 40 times. This cycle-counting technique is excellent for making gross checks on a new program to see what parts of it are or are not functioning as intended.

To examine loop operation more closely, you can set the analyzer trigger to show only the interesting parts of the trace. For example, to look at the end of an inner loop, you can set the trigger to wait until address A3, the decrement instruction in the inner loop, has occurred FF\text{HEX} times. Listing 8 shows the resulting trace. The trigger cycle (cy#0) is the next instruction after the FFth pass round the inner loop. You'll see that on completion of this pass, the outer loop correctly decrements the H register (cy#2), after which the inner loop resumes. Similarly, you could trace the exit from the subroutine by triggering on the fortieth iteration of the outer loop. The resulting trace (Listing 9) shows the correct completion of the fortieth iteration and the return to the section of code that called the delay subroutine.

In general, if you find yourself spending long hours

Authors' biographies

Thomas R Blakeslee is the founder of Orion Instruments (Redwood City, CA) and an expert on digital systems design. His textbook Digital Design with Standard MSI and LSI (Wiley Interscience) has been adopted as a text by more than 75 colleges and universities. He is also author of The Right Brain, published by Doubleday. He is a 1962 Caltech graduate and was a founder and engineering vice president of Logisys Inc in Sunnyvale, CA. He enjoys music, skiing, tennis, and windsurfing.

John S James is an independent consultant who has worked with Orion Instruments on various projects. He is a specialist in the Forth programming language and has written technical and tutorial articles on its use. He is also a partner in the CommunTree Group, for which he co-designed a computer-conferencing bulletin board. Other interests include technical writing in the area of medical research.

Article Interest Quotient (Circle One)
High 476 Medium 477 Low 478

EDN September 4, 1986
Excerpt from the forthcoming McGraw-Hill textbook on microprocessors

by

DR. JOHN PEATMAN

GEORGIA INSTITUTE OF TECHNOLOGY
EMULATOR/LOGIC ANALYZER DEVICES

The broad acceptance of the IBM PC as a standard around which products can be built has made relatively low-cost emulation and logic analysis possible. As we discussed in the last section, the logic analyzer in a full-featured development system gives the designer an outstanding view of what the microcontroller is doing. The ability to capture and display data can also be achieved in a unit which uses an IBM PC for its display, for command entry, and for formatting captured data.

A particularly intriguing unit is Orion Instruments' Universal Development Laboratory (UniLab II), shown in Fig. 7-17.* The designers of this unit have developed a universal hardware interface for a large number of microprocessors. It includes a plug for a ROM socket, which picks up the data bus and most of the address bus lines. Then it includes a DIP clip to probe the remaining address and read/write control lines on the microprocessor itself. In this way, the UniLab is party to all bus activity. Furthermore, by having access to microprocessor through the ROM address space, the UniLab presents a rather non-invasive addition to the target system. That is, the target system runs with the microprocessor in its own socket, being clocked by its own clock.

The UniLab monitors all bus transactions. It exerts control by the program instructions which it presents to the microprocessor in the ROM address space. In a sense, it exerts control like the BUFFALO monitor discussed in conjunction with the circuit of Fig. 7-7.* However, with more sophisticated hardware control, the monitor instructions are switched into and out of the ROM address space. Consequently, the monitor does not consume any of the target ROM or RAM space.
Code is written for the target system using the target system's ROM and RAM addresses, not some intermediate addresses (as had to be done for the circuit of Fig. 7-7*).

(The next section, which describes various target systems, has been omitted for brevity.)

Because of the universality of the hardware of the Orion UniLab, it is less expensive than other emulator/logic analyzer units. Its customization comes through the software which is downloaded from the IBM PC to the UniLab for doing monitor-type jobs. The software which runs on the IBM PC must also be customized for disassembling the instructions which the CPU has executed and for customizing the display of CPU registers.

The logic analyzer built into the UniLab is 48-bits wide and can collect 170 memory transactions. Since 48 input lines are more than are needed to monitor just the address bus and data bus structure, the extra lines can be used to probe other parts of the circuit (e.g., the of the microcontroller chip). An example display of a trace is shown in Fig. 7-21*.

One key to the potential power of a logic analyzer is its ability to set up and use complex triggering conditions. As the microcontroller executes hundreds of thousands of instructions per second, we want to be able to trigger on just that condition which warrants our attention. The UniLab unit permits us to trigger after four successive conditions have been met on the 48 lines. It permits triggering on any access within a specified address range, or on any access outside a specified address range. It permits triggering on the ANDing of simultaneously occurring conditions and the ORing of alternative conditions. An arbitrary delay can be interposed after triggering occurs and before data is collected. It can be told to
collect only those events which meet the trigger condition. Alternatively, it can collect this trigger condition plus the one, two, three, or four cycles which follow the trigger condition.

As an example of the power of a good logic state analyzer, consider the case in which an algorithm works fine when run by itself. However, when this algorithm is run in conjunction with the complete instrument or device software needed by an application, one of its variables becomes corrupted. The UniLab permits us to trigger only on writes to the variable and to collect not only the data written to that address but also the memory transaction which takes place during the following CPU cycle. In this way, we can see where the following instruction is fetched from and determine the offending part of the program.

(At this point, Dr. Pestman describes the support for families of microprocessors.)

*Figures and Illustrations in preparation.*
Orion Express Rolls On

This Newsletter is designed to provide Orion customers with important and useful information to help you get the most from your Orion purchase. It contains product facts from our engineering and marketing staff as well as contributions from our customers.

We invite you to share your experiences, questions, and "discoveries" with other Orion users through this Newsletter. Please mail your material to the attention of Editor, Orion Express. Naturally, we'll give you credit for any contributions.

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

UniLab Macros for the Power User

The UniLab command set can be easily extended by creating custom commands specific to your needs. The basic techniques are to use combinations of commands to form Macros, and to use PC-DOS's "command tail" feature to pass these macros to the UniLab from the keyboard or from a DOS batch file.

For example, if you switch back and forth often between the UniLab and a cross-assembler to make progressive changes in your code, you may have found yourself entering commands like this each time you re-enter the UniLab system:

```
0 7FF BINLOAD TARGET.BIN (load new binary file from cross assembler)
SYMFILE TARGET.SYM (load new symbol table from cross assembler)
STARTUP (Start the target system up from reset)
42F RB (get a breakpoint at address 042F, reset still on from STARTUP)
NMI (single step)
NMI (single step again)
```

There are a couple of ways you can automate much of this procedure.

---

Automate your procedures

If you wanted to automate only the file loading part, you could make a macro called LOAD-FILES like this:

```
: LOAD-FILES 0 7FF BINLOAD TARGET.BIN SYMFILE TARGET.SYM
```

Now, every time you want to load in the binary file and symbol file with these names you just type LOAD-FILES.

Suppose you change the name of the binary file for different versions. You can construct another macro a little differently to make it prompt you for the file name when you enter LOAD-FILES1:

```
: LOAD-FILES1 0 7FF [COMPILE] BINLOAD SYMFILE TARGET.SYM
```

Using the command [COMPILE] immediately before BINLOAD causes BINLOAD to ask for the file name when the macro is executed, rather than when the macro is defined (compiled). You could also have preceded SYMFILE with [COMPILE] and then a macro called LOAD-FILES2 could prompt you for both file names.

More time savers

The second part of your procedures could also be automated by a macro command. For purposes of this demonstration, let's define a macro and name it RUN.

```
: RUN STARTUP 42F RB NMI NMI
```

Now, you can just type LOAD-FILES RUN to do everything.

One point: If you wanted to set the breakpoint at another address, you would have to retype in all of the commands that are in the RUN macro separately. If RUN were defined without the 42F address, then you could supply the address as a parameter to the macro when it was executed, like this:

```
: RUN+ STARTUP RB NMI NMI
```

This allows you to execute 42F RUN+ or 174 RUN+, or any address you want.

Putting these two together produces a macro which can save you a lot of keystrokes:

```
: LRUN LOAD-FILES RUN+
```

(continued on page 2)
Create your own custom UniLab

Now, if you save your customized UniLab system to disk with SAVE-SYS, your new commands will be permanently added to the regular command set of your system. You can get rid of them by typing FORGET LOAD-FILES. (Using FORGET will delete every new macro made since LOAD-FILES was defined.)

There is another technique which can take the place of some of these commands, or can be used to extend them even further.

If you had defined these commands and saved the system, you could enter the system and execute the commands immediately by entering from DOS:

```
C>UL280 42F LRUN
```

Now this "command tail" will be executed by the UniLab after it boots up.

You also could have entered the commands separately:

```
C>UL280 0 7FF BINLOAD TARGET.BIN SYMFILE TARGET.SYM 482 RUN+
```

(Note: Type as one line on screen or in batch file.)

Using the command tail lets you make very useful batch files. You may already have a batch file to use with your cross- assembler. By adding an additional line like the one above, you can quickly get from DOS into your editor, then to your cross-assembler to assemble a new file, and finally to the UniLab, executing commands automatically. A real time saver.

Now, if you end the command tail with BYE, then you would automatically exit from the UniLab, and be back in DOS, or the next line of your executing batch file.

The power of DOS and the UniLab macro capability gives you great flexibility in setting up your own development environment. So have fun and let us know what especially useful macros you create. See the article below for advanced programming information.

---

New Programmer's Guide

A comprehensive new UniLab Programmer's Guide is available for advanced users of the UniLab and OptiLab systems. This 8½ by 11-inch, 45-page document enables programmers familiar with the Forth programming language to write advanced macros utilizing the UniLab's operating system. Differences from the PADS Forth system are noted, and special words are explained which are not covered in the UniLab manual. Material covered:

- File and Editor Commands
- Accessing the UniLab trace buffer directly
- UniLab String Package
- Changes to the PADS nucleus

Examples of automated test routines are included, as is a source listing of the UniLab Forth nucleus. The manual, affectionately given Part Number "PROGO", is priced at U.S. $45.00, including airmail postage, and is available from stock.

---

Ask Doctor D

Each month our product gurus select interesting questions asked by users for inclusion in this column. Maybe some of Doctor D's answers can help you too.

Q. I understand the UDL and UniLab need four overlay bytes in ROM space. I have code at your overlay area. Is there anything I can do?

A. All Disassemble/Debug (DDB) packages now have an =OVERLAY command to relocate the area that our DEBUG program needs. If you type HO or press Ctrl-F3, the help screen for DEBUG will show the current location for the overlay area. One common mistake in changing the overlay area, is moving up too high. The amount of space needed varies from processor to processor, but a good rule of the thumb is to keep the low byte of the overlay area the same or less than the low byte of the default overlay address.

For instance, if the overlay area is currently set to FFB2, then don't try to change it to FFC4. This might make the overlay area cross a page boundary, or try to use non-existent emulation memory. In some processors it could even overwrite the reset and interrupt vector area. It is better to move it to FEB2 or FF20, allowing the overlay area to remain on a single page of memory.

Send your questions to Doctor D. We'll publish the ones of most general interest, but if you'll include your phone number, we'll give you a personal call back. Write today! Right now!
Do You Have the Latest and Greatest?

Here's a listing of the latest revisions of popular Orion software. Check to see which revision you have by typing the command: "DDB" while in the Disassemble/Debug (DDB) package. If you don't have the latest, read below to see how you can get updated.

<table>
<thead>
<tr>
<th>PROCESSOR</th>
<th>LATEST REVISION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1802</td>
<td>October 23, 1986</td>
</tr>
<tr>
<td>6301</td>
<td>May 4, 1987</td>
</tr>
<tr>
<td>65P</td>
<td>November 12, 1987</td>
</tr>
<tr>
<td>6502</td>
<td>October 13, 1986</td>
</tr>
<tr>
<td>6800</td>
<td>December 18, 1986</td>
</tr>
<tr>
<td>6801</td>
<td>May 4, 1987</td>
</tr>
<tr>
<td>6802</td>
<td>December 18, 1986</td>
</tr>
<tr>
<td>6805</td>
<td>January 13, 1987</td>
</tr>
<tr>
<td>6809E</td>
<td>November 5, 1986</td>
</tr>
<tr>
<td>68000</td>
<td>January 14, 1987</td>
</tr>
<tr>
<td>68008</td>
<td>October 15, 1986</td>
</tr>
<tr>
<td>68HC11</td>
<td>May 4, 1987</td>
</tr>
<tr>
<td>8048</td>
<td>November 11, 1986</td>
</tr>
<tr>
<td>8051</td>
<td>January 29, 1987</td>
</tr>
<tr>
<td>8051P</td>
<td>January 29, 1987</td>
</tr>
<tr>
<td>8085</td>
<td>May 6, 1987</td>
</tr>
<tr>
<td>8086</td>
<td>February 5, 1987</td>
</tr>
<tr>
<td>8088</td>
<td>December 18, 1986</td>
</tr>
<tr>
<td>8096</td>
<td>October 15, 1986</td>
</tr>
<tr>
<td>SUPER 8</td>
<td>May 5, 1987</td>
</tr>
<tr>
<td>Z-80</td>
<td>May 4, 1987</td>
</tr>
<tr>
<td>Z-8000</td>
<td>October 21, 1986</td>
</tr>
<tr>
<td>78312</td>
<td>May 4, 1987</td>
</tr>
</tbody>
</table>

How to Obtain Latest Releases:

Of course, if you haven't purchased a particular package from Orion, call your local Orion Sales Representative or our Sales Hotline to place your order. If all you need is an update, each is available for U.S. $50.00 including airmail postage. Subscribers to Orion's Support Services Option receive updates free of charge. The Support Services Option costs just $500/year and gives you free updates plus unlimited free Applications Engineering telephone support, and free factory repair of your unit in case of failure. Ask for full details.

New Convenience and Power for Your System

Emulate with a MicroTarget™

Many customers have wanted to test their software before they actually have their own target hardware up and running. Orion's new MicroTargets allow you to do just that. The MicroTargets are completely functional circuit boards built around the most popular microprocessor types. They include RAM, Parallel I/O, and in many cases timer chips.

Even after your own target hardware is up, the MicroTargets can provide a ready means of verifying that your Orion system is functioning properly. MicroTargets can be purchased separately, and include the microprocessor itself and a schematic of the circuit. You may find that the MicroTarget is all the hardware you need for some smaller projects.

Available MicroTargets are:

63PO1  64180  6502  68PO5  68000  8031  8051P  8085  8086MIN  8086MAX  8088MIN  8088MAX  Z8  Z80.

Prices range from $150 to $300.

Emulation Modules Speed Hookup

As you probably know, Orion's technique for performing emulation functions uses an actual microprocessor in your target circuit. This technique, which we call In-Place Emulation, eliminates the timing and signal errors often experienced with conventional emulators due to long cables between processor and target. The result is that Orion allows your circuit to run at full speed while you observe the performance in real-time.

Orion's new line of In-Place Emulation Modules™ make connecting your Orion system to your target circuit easy. Just remove your microprocessor from its socket and plug-in the Emulation Module! The cabling is all done for you, and all necessary connections are automatically made. Of course, if you have a soldered-in microprocessor or extremely limited physical space where the Emulation Module won't fit, you can still connect to your circuit with ROM cables and jumpers — another advantage of using the versatile Orion approach.

Emulation Modules are available for these processors:

6303R  63PO1  64180  6502  65C02  6805E2  14-6805E2  68PO5WO  68PV07  6809E  6809  68000  80188  8031  8031  8032  8048/P  8050  8051P  87P50  80C51S  8085  8086  (MIN/MAX/C)  8088 (MIN/MAX/C)  Z8  Z80.

Prices range from $190 to $390.

Call the Orion Sales Hotline, 800-245-8500 for more information on these useful new products.
Supported by Orion.

- The latest processor, the 78310/12, is now available.
- Each processor has 8 processors.
- New features include step-by-step/breakpoints on single conditions.
- New and updated support for 68HC11 available.

Newsletter for Users of Orion UniLab and UDL
Host Requirements for the UniLab

Q: What are the Host requirements?

- Compatibility
  IBM computer or 100% True Compatible

- DOS Version
  Version 2.1 or greater

- Disk Drive
  Minimum of one (1) disk drive, preferably two (2) and ultimately a hard disk drive

- Host Interface
  Via the RS232 High Speed data link into the RS232 C port (switch selectable @ either 19,200 or 9600 baud)

- Interface Connection
  Requires a serial port configured for data communications equipment connection (modem)

- Copy Protection
  UniLab software is not copy protected - it can be copied to your hard disk for faster operation

- RAM Requirements
  Minimum of 320K RAM for the UniLab - if you are also running the Program Performance Analyzer (PPA) you will require 512K RAM

- Monitor/Graphics
  Hercules compatible monitor card or EGA/C GA/RGB Text Mode. Color, black and white modes.
TECHNICAL QUESTIONS & ANSWERS
June 12, 1987

**Common questions on UniLab Host Requirement:**

**Q:** Can the VAX be used as the host?

**A:** Yes. The UniLab is geared to work in a PC environment enabling it to use the power and flexibility of the PC however, CompuMech Corp., an OEM partner of Orion Instruments, has reconfigured the UniLab to work in a VAX environment.

**Q:** Can a host computer that runs MS DOS but is not an IBM or compatible run with the UniLab?

**A:** No. The UniLab requires an IBM PC/XT/AT or true compatible. The UDL however, will run with a variety of machines that run DOS and are not IBM or compatibles.

**Q:** Can we use a CPM based computer as the host?

**A:** The UniLab requires an IBM PC/XT/AT or true compatible. The UDL however, supports a variety of CPM formats depending on the exact type of host computer.

**Q:** Can the UniLab work with a dumb terminal as a host?

**A:** No. The UniLab is designed to make use of the power and flexibility provided by personal computers. The power of the PC combined with the UniLab Software makes the UniLab a true engineering work station.

**Q:** Will other RAM resident utilities, i.e. SideKick, etc. conflict with the UniLab Software?

**A:** Due to the speed of our communications we require full attention of the host. The UniLab, in most cases, does not run with background tasks or desk accessory programs.
TECHNICAL QUESTIONS & ANSWERS

June 12, 1987

EPROMS and EEPROM Programmer

Q: EPROM Programming?

- Programming Variety
  Equipped to program most of the standard EPROMS on the market (from a 2716 thru a 27256) - consult the spec sheet for listing

- Uploading from EPROM
  UniLab's programmer can be used for both uploading & downloading an EPROM directly from emulation memory

- Types of EPROMS
  UniLab will program & read both NMOS & CMOS EPROMS.

- Speed of Programming
  Using a smart programming algorithm, the UniLab programs almost as fast as Data I/O.

Common questions on the EPROM & EEPROM Programming

Q: Will the UniLab program the 27512?

A: The UniLab will program the 27512 with the optional PM512 module. You will also need at least 64K of emulation memory in an 8 bit application or the 128K model in a 16 bit application.

Q: Will the UniLab program PAL's and Array Logics?

A: No. However, the UniLab can program almost any EPROM.

Q: Will the UniLab program a 2708 EPROM?

A: No. The 2708 is an older style 1K EPROM that needs an additional 12V supply which is not provided by the UniLab.

Q: Is the UniLab available without the EPROM programmer?
TECHNICAL QUESTIONS & ANSWERS

June 12, 1987

A: The EPROM programmer is a built-in feature of the UniLab which cannot be removed.

Q: Can the UniLab program a microcontroller or single chip micro with internal EPROM?

A: No. However, a variety of manufacturers are now providing adapters which allow a PROM programmer to read the EPROM inside a single-chip micro as a standard EPROM.

The Stimulus Generator

Q: Stimulus Generator?

- Description
  The Stimulus Generator is part of the Bus-State Analyzer Cable which plugs into the EPROM programming socket bringing the signal out to .025 receptacles.

- Operation
  The Stimulus Generator sends a known bit pattern to the target and uses the UniLab to read the processor I/O port.

- Features
  Generated from the EPROM programmer circuit, it allows the capability to set/reset as a group, individually or to produce a repeating pattern of 8 signals from the keyboard.

- Uses
  In system checkout, the capability to build a switch panel to allow system inputs to be changed easily. The UniLab provides eight latched output bits - controlled from your keyboard - used for system or I/O checkout.
TECHNICAL QUESTIONS & ANSWERS

June 12, 1987

Common questions on the Stimulus Generator

Q: Will the Stimulus Generator provide signature analysis?

A: Not the traditional signature analysis, although you can use the generator to change the inputs to the target system in sequence and then compare the resulting traces.

Triggering for the Analyzer Function

Q: What type of inputs can the analyzer trigger on?

A: In addition to the address, data and control lines, the UniLab has an extra eight miscellaneous inputs which can be hooked up to external signals anywhere on your board for extra triggering capability. You can also delay the trigger to begin a trace after the event has occurred, i.e. you can start the trace after the event has occurred 50 times. There is also a filtering capability that allows you to exclude/include a range of addresses, i.e. as in excluding loop cycles which otherwise could completely fill the trace buffer.

Q: Can the UniLab triggr on a write to the internal RAM?

A: Yes. The UniLab has the capability of triggering on a particular register value by patching the target program so that the register value appears on the bus. The bus-state analyzer can then see it and get a trace or breakpoint.

Q: Can the UniLab cross-trigger between the analyzer and emulator?

A: Yes. Using the RI SI command, the same triggering power of the analyzer can trigger a breakpoint from the emulator.
Hook-up or Connecting to the Target

For the most popular processors, In-Place Emulation Modules (tm) are available. This allows you to simply remove your CPU from your board & plug the emulation module in its place.

For less popular processors, our traditional hook-up consists of clipping a dual in-line package over the CPU & connecting the cable to that. A separate emulation cable, with a ROM plug, is plugged into the ROM socket.

Common Questions on Target Hook-up and Connection

Q: What are the advantages of connecting to my target using the emulation modules?

A: Functionally, there are no differences between using the emulation modules or the standard hook-up arrangement. The main advantages come from the ease of use and the durability. In addition, the emulation modules will allow you to run with your ROMS still in your target board. This is especially useful in using the analyzer in a completely transparent, passive analysis mode.

Q: How would I hook-up to my single chip micro with internal ROM or on-board ROM?

A: Most processors with internal ROM also have a piggy-back version. Using the piggy-back version for development is faster because you don't have to burn a new microcontroller for every change you make in your code. Some processors also have ROMless versions with off-board ROM. With a ROMless application, Orion's emulation modules would plug directly into the CPU socket. We also make several piggy back emulation modules.
Q: My board has four ROMs. Do I need four ROM plugs? How do we hook-up two or more ROMs with just one ROM plug?

A: Orion's use of the ROM is an easy way for us to pick up the required address and data lines. You simply need to remove your ROMs from the board and plug the emulation cable into one of them. The remaining control and status signals are picked up directly at the CPU. However, Orion's In-place Emulation Modules eliminate the need for us to pick up signals at the EPROM socket.

Q: How would I hook up to my processor if it is a PGA (pin grid array) or PLCC package?

A: One way would be to set up a bus header since the analyzer cable only needs to connect to a few signals, usually less than a dozen pins. The second way would be to use an adaptor which converts a PGA to a QUAD in-line package. These can be obtained from a company called Emulation Technology. For specifics, call Orion's Application Engineers.

Q: How does the MicroProcessor stop at a break point if it is simply emulating the ROM?

A: The microprocessor is not actually stopped. Instead, the target processor is put into an "idle loop" during which a proprietary subroutine enables us to put the contents of the registers in the trace memory where the analyzer can pick them up. Once displayed, you have the capability of changing the contents of any register as well as patching any RAM or emulated ROM locations or change ports.

Q: What happens in the "idle loop" and how is it accomplished?

A: The "idle loop" is accomplished by continuously feeding the processor a "jump minus two" instruction, which makes it continuously jump to itself.
TECHNICAL QUESTIONS & ANSWERS

June 12, 1987

Q: How many break points can be set?
A: The UniLab can set up to nine software breakpoints or use the analyzer's triggering logic to set a hardware breakpoint. In general, breakpoints are set "on demand" and in this sense an unlimited number of breakpoints can be set.

AVAILABLE SUPPORT FOR THE UNILAB

Q: What kind of technical help or assistance do I receive with the system?
A: The UniLab comes with a detailed step-by-step tutorial, a handy reference card and an extensive reference manual providing answers to virtually any questions which might arise. In addition are existing on-line help facilities, menu displays and a complete on-line glossary of UniLab commands and features.

Q: Why doesn't Orion provide schematics with systems?
A: Orion considers the schematics to be proprietary information and they are not available for customer access. Our technicians are fully trained in troubleshooting systems problems. Therefore, Orion feels that repairs should be done at the factory or under advise of our application engineers.

Q: Are memory or speed upgrades available for the UniLab at a later date?
A: UniLab is both memory and speed upgradeable. When this service is needed please call a sales engineer to supply you with the details.

Q: Is the UniLab hardware field upgradeable?
A: No. The UniLab needs to be returned to Orion's factory for all hardware upgrades. When the upgrade is complete Orion's technicians run it through a complete set of diagnostics and reinstate the 90 day hardware warranty.
General Functionality

Q: Is the UniLab a real time in-circuit emulator?

A: Yes. The UniLab provides both a real time emulator as well as a true in-circuit emulation.

Q: Do you simply emulate the ROM? Are you just a ROM emulator?

A: No. We plug the target processor into your target circuit with an emulation module. The ROM is emulated in the same fashion as other development systems and emulators, however due to Orion's innovative approach to interfacing with each target CPU, UniLab can work with almost any processor.

Q: What is the UniLab?

A: The UniLab is a microprocessor analysis system which is actually four instruments in one. It combines a real time in-circuit emulator and an advanced 48-channel bus state analyzer together with a stimulus generator and a built-in EPROM programmer. All these functions are tied together in a small, compact unit which is designed to interface with an IBM PC via the RS232 High Speed data link. It is an extremely powerful and versatile tool for debugging microprocessor-based designs.

Q: How does the UniLab work?

A: The UniLab actually uses your microprocessor hardware itself to provide the emulation for the software. Rather than have the CPU inside the emulator, Orion's approach leaves the target CPU on the target board and the UniLab controls it externally.

Q: Is the UniLab a logic analyzer?

A: No. UniLab is an in-circuit emulator combined with a 48 channel bus state analyzer.
TECHNICAL QUESTIONS & ANSWERS

June 12, 1987

Q: How is a bus state analyzer different from a logic analyzer? What is a bus state analyzer?

A: A bus state analyzer is essentially an analyzer which has been optimized for capturing trace cycles of the bus. It will only run synchronously with the target's clock.

Q: Can you look at and change the target registers?

A: Yes. Once you have stopped at a break point, the target microprocessor is put into an "idle loop" and all the register contents will be displayed. At this point you can change register contents. You can also examine and change the contents of target RAM or emulated ROM locations as well as I/O ports.

Q: How much emulation memory is available to the user?

A: The UniLab comes in three memory sizes, 32K, 64K and 128K. It is a common misconception that we use part of this emulation memory for running the UniLab software. Actually, our software is run from the host computer and the emulation memory is left strictly for the users program. If the target program is larger than the UniLab's emulation memory, you can divide the program into smaller sections or you can add memory to the UniLab.

Q: Can the processor be started up at a specific address?

A: Yes. Once you have established debug control, the UniLab has a GO TO command which will allow the user to adjust the program counter (when at a breakpoint) thus modifying program flow.

Q: Does the UniLab support high level language debugging?

A: The UniLab is an assembly language debugger. However, the UniLab can read symbol tables generated by the high level language compilers or assemblers. With certain compilers, the C Language Source Code can be interspersed in the UniLab's display of the assembly code.
Q: Does the UniLab support symbolic debugging and can it read assembler generated symbol tables?
A: Yes. The Unilab can read symbol tables generated by cross assemblers. It totally supports symbolic debugging on an assembly language level.

Q: Can we support bond-out technology?
A: With our method of emulation, there is no need of bond-out technology.

Q: How deep is the trace buffer?
A: Our 1K trace buffer enables the user to view approximately 170 bus cycles.

Q: How can you justify only 170 cycles? I would need at least a 2K buffer.
A: The UniLab has extensive filtering capabilities as well as four-step sequential triggering. The end result is that we sort through the cycles that are not of interest showing you only the desired cycles. In fact, you'll probably find that the UniLab, by virtue of its high speed RAM truth tables has the most sensitive and powerful trigger on the market.

Q: Will the UniLab do timing analysis?
A: No, not in the same way that a logic analyzer would. However, it does have a Program Performance Analyzer (tm) or PPA which will give you insight into the actual workings of your program by giving you both address domain and time domain analysis for histogram displays of program execution. In other words, it will show you where your program is spending all its time, and how much time it's spending on unknown sectors of the code.

Q: Does Orion support PLM or any other high level languages?
A: As long as the compiler will generate an Intel hex file, the processor machine code can be downloaded to our emulation memory and debug can continue at the assembly language level.
TECHNICAL QUESTIONS & ANSWERS

June 12, 1987

Q: Can you make program patches using mnemonics?
A: UniLab's software comes standard with the single line-by-line assembler which allows you to make immediate changes to your program by simply entering the address and the assembly language mnemonics. This is a great time saver. So, you would not have to exit the UniLab environment and look up the codes in order to enter a patch into your code.

Q: Can you split the Emulation ROM in the target system?
A: Yes, in 2K increments. In other words, with 2K resolution.

Q: Does the UniLab support 32 bit processors, i.e. the 68020 or 8386 or TMS 32020, etc.?
A: As it is currently set up, the UniLab is designed to support virtually any eight or 16 bit microprocessor. At the current time we do not intend to undertake any hardware design modifications which would allow us to work with 32 bit processors. However, I can put your name on the wish list in case we receive enough requests.

Q: Can the UniLab determine between a prefetch and an executed instruction?
A: Yes. The disassembler will re-order the display of cycles in the trace buffer so that the memory cycles associated with a given instruction appear immediately after that instruction. It hides from view the display of program cycles that were never really executed.

Q: Do you download Intel hex code directly into the UniLab?
A: Yes, our software provides the communication protocol to download the hex file from the PC to UniLab's emulation memory.

Q: Can you do a memory map, part in target and part in memory?
A: Yes, in 2K blocks.
TECHNICAL QUESTIONS & ANSWERS

June 12, 1987

Q: Can we use a PC or AT as a target?
A: Yes, the UniLab can work with a PC or AT as a target. However, it does entail some special considerations.

Q: Do you provide "real time" emulation and trace capabilities?
A: Yes. UniLab does provide "real time" emulation and trace capabilities. In fact, one of the unique things about the UniLab is it does not actually require that the microprocessor be stopped in order to provide a trace. In essence, what this gives you is a real time debugger as opposed to a static debugger.

Q: When you say 32K of emulation memory, what is that equivalent to?
A: The 32K refers to how many thousand bytes of object code are in your program. It does not concern RAM, (i.e. data space) as RAM is left on your target and not put into UniLab's emulation memory.

Q: How can you look at a stack pointer?
A: Whenever you are stopped at a breakpoint the contents of the registers, including the stack pointer, will be displayed.

Q: Can the UniLab read Motorola S records?
A: No. The UniLab cannot read S records. However, almost any assembler or cross assembler that generates S records will also generate an Intel hex file. The UniLab can read both binary and hex files as well as reading the program directly from a PROM. Also, several utility packages on the market will convert S records to one of the above formats compatible with the UniLab, i.e. Avocet.

Q: On your price list you say expanded mode next to some processors. What does that mean?
A: Expanded mode refers to a mode of CPU operation where any and all internal on-chip ROM has been disabled and only ROM external to the CPU is used.
TECHNICAL QUESTIONS & ANSWERS

June 12, 1987

Q: Can you support the 80286 in protected mode?
A: No. We do not support the protected mode on the 80286.

Q: What do I need to change from one processor to the next?
A: With the UniLab, all it requires is a diskette and cable change. The cost is minimal compared with most other development systems.

Q: How do I down load my program into the UniLab?
A: There are two ways. One is to simply load your program from a disc file. This will download along the RS232 from the Host computer to the UniLab's emulation memory. The other way is to load your program directly from a PROM by placing it in the UniLab's programming socket.

Q: What sort of CPU do you have inside the UniLab?
A: The UniLab utilizes a Z80 microprocessor.

Q: Can you emulate RAM?
A: As long as the program flow goes to RAM, yes we can follow the program into RAM, and once we have established debug control, we can set breakpoints and single step thru RAM.
Note: RAM must be available on customer target (only ROM program map runs out of Emulation memory).

Q: How many microprocessors does the UniLab support?
A: UniLab currently has ready support for over 150 different types of microprocessors.

Q: Does the UniLab come with a cross assembler? Is the DDB a cross assembler?
A: The UniLab does come with a single line-by-line assembler, we do not have cross assembler support. We have a list of third party vendors who provide the Cross Assembler support which runs on the PC. UniLab DDB software is a Dis - assembler - Debugger, not a cross assembler.
TECHNICAL QUESTIONS & ANSWERS

June 12, 1987

Q: Which cross assemblers or cross compilers is the UniLab compatible with?

A: Virtually any cross assembler or cross compiler which will generate an Intel hex file or binary object file. So, it's compatible with virtually any cross assembler or cross compiler on the market.

Q: What is the Program Performance Analyzer?

A: The Program Performance Analyzer is an extremely useful feature that will actually show you where your program is spending its time, and how much time it's spending there. The advantage to this is that you can actually get inside into the real workings of your program and then make modifications that will ultimately make the program run faster, more efficiently, use less memory, and ultimately give the end user of your product greater functionality. It does so by doing both time domain and address domain analysis which will give you a real time histogram display of your program execution.

If you have any further questions, contact our Applications Engineering Team!
Connecting the UniLab to your Z80 target board with an In-Place (tm) Emulation Module

Introduction

The In-Place (tm) Emulation Module is the new way to connect the UniLab to your target board: you simply remove the microprocessor from your board and put the module in its place. Please read all instructions before attempting to install your new module, as incorrect procedures may cause damage to the UniLab.

Power Supply

The module can even supply power to your target, if your board takes 5 Vdc power and draws less than 1 amp.

Power is normally provided to the Z80 from the target system, but, the UniLab can also provide power to the Z80 and the target system by placing a jumper at the appropriate pins on the 3-pin jumper header on the emulation module.

WARNING: If your target board requires more than 1 amp of power, or does not take TTL voltage, then you will need a separate power supply. When you use your own power supply you must keep the +5V jumper disconnected—otherwise damage to the UniLab may result.

3-pin Jumper Header (see diagram on next page)

The EMZ80 Emulation Module comes with a Jumper Header that is configured according to your target system set-up. Please read over the next section to be sure that the EMZ80 is hooked up correctly for you.

Power Jumper (pins 1, 2)

When connected, the UniLab supplies power to the processor and target board. Disconnect if you are supplying power to the target board.

Reset Pin (pin 3)

The Z80 Emulation Module generates the appropriate reset pulse to the processor. But, if you have peripheral chips requiring to be reset, the Z80 Emulation Module provides a Reset Pin where you can place a micro jumper and attach to the master reset of the target system.
The Three Step Connection Process:

1) Remove microprocessor from board

With the module connection, the UniLab still runs all the target code on your microprocessor. The module just moves your processor a centimeter or two away from the board.

2) Plug module into microprocessor socket

Put the module into the microprocessor socket on the target board. Double check that you have oriented the module correctly, so that pin one of the processor on the module lines up with pin one of the processor socket.

3) Plug cables into UniLab sockets

Plug the 50-pin connector labeled "Emulator" into the left socket on the UniLab, plug the "Analyzer" connector into the right socket. Both connectors must be plugged in with the plastic "key" on the upper surface, and the red edge of the cable to the left.
U1 HCT132 Schmitt Trigger NAND gate.
U2 ALS32 OR gate.
* Connected to logic IC's.
EMZ80 EMULATION MODULE HEADERS

emulator cable

A_{12} A_{7} A_{6} A_{5} A_{4} A_{3} A_{2} A_{1} GND D_{6} D_{5} D_{4} D_{3}

<table>
<thead>
<tr>
<th>2</th>
<th>4</th>
<th>6</th>
<th>8</th>
<th>10</th>
<th>12</th>
<th>14</th>
<th>16</th>
<th>18</th>
<th>20</th>
<th>22</th>
<th>24</th>
<th>26</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1</th>
<th>3</th>
<th>5</th>
<th>7</th>
<th>9</th>
<th>11</th>
<th>13</th>
<th>15</th>
<th>17</th>
<th>19</th>
<th>21</th>
<th>23</th>
<th>25</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A_{14} A_{13} A_{8} A_{9} A_{11}</th>
<th>OE</th>
<th>A_{10} CE</th>
<th>A_{0} D_{7} D_{0} D_{1} D_{2}</th>
</tr>
</thead>
<tbody>
<tr>
<td>28</td>
<td>30</td>
<td>32</td>
<td>34</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A_{19} A_{17} +5</th>
<th>RES</th>
<th>GND C_{7} C_{6} C_{5} C_{4} ALE</th>
</tr>
</thead>
</table>

analyzer cable

50-PIN BUS HEADER

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

Vcc, +5V RES

μP

3-PIN JUMPER HEADER

Title:EMZ80 Headers
Rev: B

Dwg#: 5188  Drawn by: PAB  Appv'd:
Date: 6July87  Orion Instruments, Inc
Consult the appendices of the Personality Pak documentation for target board schematics
UNION INSTRUMENTS

--- ENGINEERING APPLICATION NOTES ---

RE: IBM PC TARGET SYSTEM AND THE UNILAB II

The Unilab can be used in applications that require the use of an IBM PC as the target system. All Unilab features are available to the user, including breakpoints, single-stepping, etc.

HARDWARE CONFIGURATION

The primary consideration in such an application is that a circuit must be constructed to filter out refresh cycles due to the dynamic RAM; otherwise one will get a trace full of refresh cycles instead of processor operations, and debugger control will be impossible. This filter circuit can be built on a standard IBM PC prototype card and is illustrated on page 2. Bus headers should also be installed on the prototype card so the necessary analyzer and emulator cable connections can be made, as shown on pages 4 and 5.

The analyzer cable will also need to be modified in order to properly clock in the bus cycles, as shown on page 5. The existing blue wires that need to be changed should be removed completely, and new wire should be installed. Refer to Appendix D in the Unilab reference manual for more information on custom cables.

HOOKING UP TO THE TARGET IBM PC

Once the filter circuit is made and the analyzer cable has been modified, both cables can be hooked up to the card which can then be inserted in the IBM PC. Cable-to-card connections are shown on pages 5 and 6.

The analyzer cable has connections going to the 62-pin edge connector, the filter circuit (denoted by 'CKI'), and to the PC clock generator and processor. The NMI line must be input through an inverter (or a spare NAND gate on the filter circuit) before being attached to the NMI pin on the 8088.

Note that the lower address and data lines on the emulator cable can be picked up by plugging the ROM emulator plug into a vacant ROM socket on your target board. If you plan to emulate the PC BIOS, that ROM can be replaced directly; however, before trying to emulate it is recommended that the analyzer be used to trace the target activity.
CABLE MODIFICATION

FOR PC. TARGET APPLICATION

UDL ANALYZER CABLE / CONNECTOR P1

REV. 1
6 NOV 86
### IBM-PC/UDL ANALYZER CABLE CONNECTIONS

<table>
<thead>
<tr>
<th>62-pin edge connector</th>
<th>ANALYZER connector signals</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pin #</strong></td>
<td><strong>Signal</strong></td>
</tr>
<tr>
<td>1</td>
<td>M7</td>
</tr>
<tr>
<td>2</td>
<td>M6</td>
</tr>
<tr>
<td>3</td>
<td>M5</td>
</tr>
<tr>
<td>4</td>
<td>M4</td>
</tr>
<tr>
<td>5</td>
<td>M3</td>
</tr>
<tr>
<td>6</td>
<td>M2</td>
</tr>
<tr>
<td>7</td>
<td>M1</td>
</tr>
<tr>
<td>8</td>
<td>M0</td>
</tr>
<tr>
<td>B1</td>
<td>GND</td>
</tr>
<tr>
<td>A12</td>
<td>A10E</td>
</tr>
<tr>
<td>A13</td>
<td>A10E</td>
</tr>
<tr>
<td>A14</td>
<td>A17E</td>
</tr>
<tr>
<td>A15</td>
<td>A16E</td>
</tr>
<tr>
<td>14</td>
<td>+5V</td>
</tr>
<tr>
<td>15</td>
<td>FDDD</td>
</tr>
<tr>
<td>PC RES-16</td>
<td>RES-16</td>
</tr>
<tr>
<td>0000 NMI</td>
<td>NMI-16</td>
</tr>
<tr>
<td>GND</td>
<td>GND</td>
</tr>
<tr>
<td>GND</td>
<td>GND</td>
</tr>
<tr>
<td>(CK1)</td>
<td>C7</td>
</tr>
<tr>
<td>B12</td>
<td>K1</td>
</tr>
<tr>
<td>(CK1)</td>
<td>C6</td>
</tr>
<tr>
<td>B11</td>
<td>WR</td>
</tr>
<tr>
<td>(CK1)</td>
<td>C5</td>
</tr>
<tr>
<td>B13</td>
<td>RD</td>
</tr>
<tr>
<td>A11</td>
<td>C4</td>
</tr>
<tr>
<td>A10</td>
<td>A15E</td>
</tr>
<tr>
<td>+5</td>
<td>ALE</td>
</tr>
<tr>
<td>29</td>
<td>INVI</td>
</tr>
<tr>
<td>30</td>
<td>INV0</td>
</tr>
<tr>
<td>31</td>
<td>D1CY</td>
</tr>
<tr>
<td>32</td>
<td>CCK</td>
</tr>
<tr>
<td>33</td>
<td>HACK</td>
</tr>
<tr>
<td>34</td>
<td>MCK</td>
</tr>
<tr>
<td>(CK1)</td>
<td>TCY</td>
</tr>
<tr>
<td>36</td>
<td>ITCY</td>
</tr>
<tr>
<td>37</td>
<td>MIFC</td>
</tr>
<tr>
<td>38</td>
<td>ALE'</td>
</tr>
<tr>
<td>39</td>
<td>OE'</td>
</tr>
<tr>
<td>40</td>
<td>IDLE'</td>
</tr>
<tr>
<td>41</td>
<td>CE'</td>
</tr>
<tr>
<td>42</td>
<td>OE'</td>
</tr>
<tr>
<td>43</td>
<td>C3</td>
</tr>
<tr>
<td>44</td>
<td>C2</td>
</tr>
<tr>
<td>45</td>
<td>C1</td>
</tr>
<tr>
<td>46</td>
<td>C0</td>
</tr>
<tr>
<td>47</td>
<td>A18S</td>
</tr>
<tr>
<td>48</td>
<td>A10S</td>
</tr>
<tr>
<td>49</td>
<td>A17S</td>
</tr>
<tr>
<td>50</td>
<td>A16S</td>
</tr>
<tr>
<td>62-pin edge connector</td>
<td>ROM connector signals</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>A17</td>
<td>A14E</td>
</tr>
<tr>
<td>A19</td>
<td>A12E</td>
</tr>
<tr>
<td>A18</td>
<td>A13E</td>
</tr>
<tr>
<td>A24</td>
<td>A7E</td>
</tr>
<tr>
<td>A23</td>
<td>A6E</td>
</tr>
<tr>
<td>A25</td>
<td>A6E</td>
</tr>
<tr>
<td>A22</td>
<td>A9E</td>
</tr>
<tr>
<td>A26</td>
<td>A5E</td>
</tr>
<tr>
<td>A20</td>
<td>A11E</td>
</tr>
<tr>
<td>A27</td>
<td>A4E</td>
</tr>
<tr>
<td>B12</td>
<td>OE</td>
</tr>
<tr>
<td>A28</td>
<td>A3E</td>
</tr>
<tr>
<td>A21</td>
<td>A10E</td>
</tr>
<tr>
<td>A29</td>
<td>A2E</td>
</tr>
<tr>
<td>A30</td>
<td>CE</td>
</tr>
<tr>
<td>A31</td>
<td>A1E</td>
</tr>
<tr>
<td>B31</td>
<td>GND</td>
</tr>
<tr>
<td>A2</td>
<td>D7E</td>
</tr>
<tr>
<td>A3</td>
<td>D6E</td>
</tr>
<tr>
<td>A9</td>
<td>D6E</td>
</tr>
<tr>
<td>A4</td>
<td>D5E</td>
</tr>
<tr>
<td>A0</td>
<td>D1E</td>
</tr>
<tr>
<td>A5</td>
<td>D4E</td>
</tr>
<tr>
<td>A7</td>
<td>D2E</td>
</tr>
<tr>
<td>A6</td>
<td>D3E</td>
</tr>
<tr>
<td>A1</td>
<td>D11E</td>
</tr>
<tr>
<td>A1</td>
<td>D10E</td>
</tr>
<tr>
<td>A1</td>
<td>D12E</td>
</tr>
<tr>
<td>A1</td>
<td>D9E</td>
</tr>
<tr>
<td>A1</td>
<td>D13E</td>
</tr>
<tr>
<td>A1</td>
<td>D8E</td>
</tr>
<tr>
<td>A1</td>
<td>D14E</td>
</tr>
<tr>
<td>A1</td>
<td>D15E</td>
</tr>
<tr>
<td>A1</td>
<td>D7A</td>
</tr>
<tr>
<td>A1</td>
<td>D6A</td>
</tr>
<tr>
<td>A1</td>
<td>D5A</td>
</tr>
<tr>
<td>A1</td>
<td>D1A</td>
</tr>
<tr>
<td>A1</td>
<td>D4A</td>
</tr>
<tr>
<td>A1</td>
<td>D2A</td>
</tr>
<tr>
<td>A1</td>
<td>D3A</td>
</tr>
<tr>
<td>A1</td>
<td>D15A</td>
</tr>
<tr>
<td>A1</td>
<td>D14A</td>
</tr>
<tr>
<td>A1</td>
<td>D8A</td>
</tr>
<tr>
<td>A1</td>
<td>D13A</td>
</tr>
<tr>
<td>A1</td>
<td>D9A</td>
</tr>
<tr>
<td>A1</td>
<td>D12A</td>
</tr>
<tr>
<td>A1</td>
<td>D10A</td>
</tr>
<tr>
<td>A1</td>
<td>D11A</td>
</tr>
</tbody>
</table>

These connections can also be made by plugging the UDL ROM plug into a vacant PC ROM socket (or the ROM bios socket if emulating).
GETTING A TRACE

Before tracing the PC target program, one must configure the UniLab software for an IBM PC target as follows:

SETUP

Emulator Memory Enable Status: No Memory Enabled ok
PCPATCH ok
SEG1 ok

EMCLR clears UniLab emulation memory, allowing the analyzer to trace the target memory execution.

PCPATCH enables the disassembler/debugger to recognize the different cycle types. They are identified by the high 4 bits of the CONT column of the analyzer display:

- FETCH: 40 to 5F CONT
- READ: 60 to 6F CONT
- WRITE: C0 to CF CONT
- OUTPUT: 00 to 0F CONT
- INPUT: E0 to EF CONT

SEG1 disables the UniLab's segmented mode, causing the trace to show physical memory locations (as opposed to a SEG+offset address format).

When a startup command is issued the UniLab will issue a reset to the IBM PC and the first cycles will be traced. This should include the 0000 reset address, FFFF. The disassembler may occasionally be out of synch at the beginning of the trace, but should be fine once the jump is made to the main body of the BIOS program, which is at 6004 hex for this particular IBM PC clone in the examples that follow:

| (from top of buffer) | | (H circuits)
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4F 10 0000</td>
<td>JMP E004,FO00</td>
<td>1111111111111111</td>
<td></td>
</tr>
<tr>
<td>0F 6004 00</td>
<td>CLI</td>
<td>1111111111111111</td>
<td></td>
</tr>
<tr>
<td>7F 6005 FF</td>
<td>CLI</td>
<td>1111111111111111</td>
<td></td>
</tr>
<tr>
<td>0E 6006 00</td>
<td>MOV AL,0</td>
<td>1111111111111111</td>
<td></td>
</tr>
<tr>
<td>0A 6007 00</td>
<td>OUT AL,AL</td>
<td>1111111111111111</td>
<td></td>
</tr>
<tr>
<td>0C 6008 00</td>
<td>OUT 03,AL</td>
<td>1111111111111111</td>
<td></td>
</tr>
<tr>
<td>0D 00 00 00</td>
<td>MOV DX,JO0</td>
<td>1111111111111111</td>
<td></td>
</tr>
<tr>
<td>0F 0D 00 00</td>
<td>MOV DX,300</td>
<td>1111111111111111</td>
<td></td>
</tr>
<tr>
<td>10 01 00 00</td>
<td>MOV DX,JO0</td>
<td>1111111111111111</td>
<td></td>
</tr>
</tbody>
</table>
In this mode, all of the triggering capabilities of the UniLab can be used.

If your display does not appear correct, turn off the disassembler to get a cycle by cycle display of what's happening. This may help you in determining bad connections on any of the UniLab inputs:
USING THE EMULATOR

To use the UniLab's emulation capabilities you will have to first read in the PC's BIOS ROM. First, turn the PC off and remove the ROM. The UniLab's emulator plug can then be inserted in this socket:

Next, read the ROM into the UniLab as described in section 2 of chapter 6 of the reference manual. Remember to enable the proper memory segment that you wish to emulate. A disassembly from memory should show you the same code that was executed when the analyzer traced the PC activity in the above example:

```
1140 DB
FFCD CAME00000 : JMP E000,10000:
E000 IS ON
E004 RC
E005 CL
E006 000001
E008 000000
E009 000000
E00A 000000
E00B 000000
E00C 000000
E00D 000000
E00E 000000
E00F 000000
```

Issuing a startup should result in the same trace that was obtained when the analyzer did a trace. In emulation mode, any of the triggering capabilities are available, and emulation memory can be altered.

USING THE DEBUGGER

To gain debug control on an 8080 based target system, the user program must install the required pointer at locations 8000-800F as described in the 8080/8086 target notes. There are also optional pointers that must be installed for the NMI and Single-Step features. To do this one must find an unused portion of emulation memory (or create a 'dummy' memory segment) to insert the code. In our PC-clone there was enough space to do this just before the reset vector, starting at FFCD hex.
So now when a STARTUP is issued the trace will appear as follows:

STARTUP

resetting
(from top of buffer)

(cyg cont now data)

MAIN MISC:

0 4F F0 3F CNC005000F HX (1), FF01
7 0D 40 write interrupt 11
0 0D 3F write interrupt 11
6 4F F0 CNC008000F HX (1), FF01
E 4F F0 CNC000000F HX (1), FF01
3 00 60 write interrupt 11
16 00 70 write interrupt 11
14 4F F0 CNC005000F HX (1), FF01
10 0D 00 write interrupt 12
1C 0D 70 write interrupt 12
1C 4F F0 CNC008000F HX (1), FF01
23 0D A0 write interrupt 12
26 0D 90 write interrupt 12
24 4F F0 CNC005000F HX (1), FF01
20 0D 80 write interrupt 13
2E 0D 80 write interrupt 13
2C 4F F0 CNC005000F HX (1), FF01
35 0D E0 write interrupt 13
36 0D F0 write interrupt 13
34 4F F0 CNC008000F HX (1), FF01
3C 4F 0004 FC

PG On or "Y" (Trace resume) Home (top n IN (from step n) 1 (from n=5)

Now one can gain debug control as follows:

RESET LOAD AD resetting

IV=601B F=f002----ad1sz-a-p-cl AX=6039 DX=0021 CX=0000 DX=0100
CS=4000 DS=0000 SS=0000 ES=0000 SP=054A DI=F7F2 DI=15C0 SI=E05A
LOAD EGGJ OUT 63, NL (next step) ok
At the above breakpoint you can see a display of all target register contents and a disassembly of the next instruction to be executed. Note that when a breakpoint is obtained in this manner the PC will reboot since a target reset command was issued.

The Unilab command ‘N’ will allow the user to single-step through the program code, and ‘NMI’ will allow the user to follow branches:

N
0010 f=f021 odL3x.x-p-c / AX=0028 DX=0024 CX=0000 DX=0300
CS=0030 DS=0000 ES=0000 SP=0031 DI=F0F72 SI=0054
LOAD 0600 MOV AX,ES

(next step) ok

One can jump ahead in the program and set another breakpoint without resetting the system as follows:

E044 MV
0054 f=f021 odL3x.x-p-c / AX=1103 DX=0024 CX=7000 DX=0300
CS=0000 DS=0000 ES=0000 SP=025E DI=F0F72 SI=0051
E044 ECFE LOUP E044

(next step) ok

NMI
0054 f=f021 odL3x.x-p-c / AX=1103 DX=0024 CX=707F DX=0300
CS=0000 DS=0000 ES=0000 SP=025E DI=F0F72 SI=0054
E044 ECFE LOUP E044

(nex step) ok

K
0056 f=f021 odL3x.x-p-c / AX=1103 DX=0024 CX=0800 DX=0300
CS=0000 DS=0000 ES=0000 SP=025E DI=F0F72 SI=0051
E046 3200 JMP NL/ML

(nex step) ok

E049 MV
0057 f=f165 odL3x.x-p-c / AX=0000 DX=0070 CX=0000 DX=0300
CS=0000 DS=0000 ES=0000 SP=025E DI=F0F72 SI=0056
E050 E000 JMP E000

(nex step) ok

NMI
0060 f=f165 odL3x.x-p-c / AX=0000 DX=0070 CX=0000 DX=0300
CS=0000 DS=0000 ES=0000 SP=025E DI=F0F72 SI=0056
E060 0054 MOV AL,51

(nex step) ok

One can quickly get by the initialization routine by doing a STARTUP and then a NOW? command after the PC has rebooted. In the example below, the PC has been rebooted by the Unilab, and then Unilab software has been loaded into the target PC! A NOW? command shows the target PC in an infinite loop that is looking
at the CUNL serial I/O port (3FD hex) for a UniLab connection that is not there:

```
40 F305 FB  MOV BX, SP
40 F306 RSC
40 F302 BUCW9P
-50 80 3F 60 inp
-50 40 F504 DIWDJ
-77 40 F507 EC IN AL, DX
-75 80 3F 60 inp
```

Once a starting point is established for an area of interest, the powerful triggering capabilities of the UniLab can be utilized to allow the user to look specifically at what he is interested in. In this way the user can move through his program and observe his program flow as the PC 8088 executes it.

Paul Varma, Applications Engineering
INSTRUCTIONS FOR 4617, 4617-1, 4617-3 IBM AT

1. BOARD DESCRIPTION

The 4617 plugboards for the IBM AT computer have a .1" x .1" pattern for unshielded component placement. Power and ground lines surround the grid pattern on both the component and solder side of the board. The power and ground lines are connected to 5 V and ground pins of the edge connectors.

Universal D-subminiature and .1" x .1" patterns are provided on the rear board edge.

A layout sheet is provided to aid in I.C. and component placement.

A universal bracket is provided for four sites of I/O connectors or to secure the board into the computer chassis.

2.0 CONNECTOR LOCATION AND NUMBERING

IBM motherboard connector locations are shown in fig. 2-1. Motherboard pin numbers and signal names are shown in fig. 2-2.

I/O Pin | Signal Name
--- | ---
A1 | I/O CH CK
A2 | SD7
A3 | SD6
A4 | SD5
A5 | SD4
A6 | SD3
A7 | SD2
A8 | SD1
A9 | SD0
A10 | I/O CH RUY
A11 | \( \_ \_ \_ \_ \)EN
A12 | \( \_ \_ \_ \_ \)E
A13 | \( \_ \_ \_ \_ \)T
A14 | \( \_ \_ \_ \_ \)R
A15 | \( \_ \_ \_ \_ \)S
A16 | \( \_ \_ \_ \_ \)A
A17 | \_\_\_\_IOW
A18 | \_\_\_\_SMEMR
A19 | \_\_\_\_SMEMW
A21 | \_\_\_\_SMEMW
A22 | \_\_\_\_SMEMR
A23 | \_\_\_\_SMEMR
A24 | \_\_\_\_SMEMW

I/O Pin | Signal Name
--- | ---
B1 | GND
B2 | RESET DRV
B3 | 15 Vdc
B4 | IN
B5 | 5 Vdc
B6 | DIRQ2
B7 | -12 Vdc
B8 | OWS
B9 | -12 Vdc
B10 | GND
B11 | -SMEMW
B12 | -SMEMR
B13 | -IOW
B14 | -IOR
B15 | -DACK3
B16 | -DIRQ3
B17 | -DACK1
B18 | -DIRQ1
B19 | Refresh
B20 | CLK
B21 | IRQ7
B22 | IRQ6
B23 | IRQ5
B24 | IRQ4
B25 | IRQ3
B26 | DACK2
B27 | F/C
B28 | BALE
B29 | +5 Vdc
B30 | OSC
B31 | GND

3.0 UNIVERSAL BRACKET

A universal bracket is provided for mounting one of 4 I/O connectors; it can also be used to secure the board without an I/O connector. See fig. 2-1. The D-subminiature 9-pin AMP no. 745112-2; 15-pin AMP no. 745113-2; 25-pin AMP no. 745114-2; or 37-pin AMP no. 75115-2 or equivalent connectors may be used if their dimensions match fig. 2-2 to fit the bracket and computer.

VECTOR ELECTRONIC COMPANY, 12460 Gladstone Avenue, Sylmar, CA., 91342, phone (818) 365-9661
Appendix B: Sources of Cross Assemblers and C Compilers

The UniLab software is designed to work with any assembler or compiler. The only thing the UniLab needs is the object code in either binary format or INTEL hex format.

Even this hurdle can be overcome with one of the various conversion programs on the market. For example, Avocet has a product which converts Motorola S-records into binary format. See theVendor listing for Avocet below.

As a service to our users we have compiled the following list of inexpensive cross assemblers and compilers. The two character abbreviations indicate the sources listed on the following pages. We would appreciate any user feedback so that we can keep this list current.

<table>
<thead>
<tr>
<th>PROCESSOR</th>
<th>ASSEMBLER SUPPLIERS</th>
<th>COMPILER SUPPLIERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1802/5</td>
<td>25 AV MI WE AA EN RE SD UW</td>
<td>IT AR</td>
</tr>
<tr>
<td>6301</td>
<td>25 AV CY EN LO RE UW</td>
<td></td>
</tr>
<tr>
<td>6305</td>
<td>AV MT RE</td>
<td></td>
</tr>
<tr>
<td>6502</td>
<td>25 AV LO MI EN RE SD</td>
<td>MX</td>
</tr>
<tr>
<td>6800/2/8</td>
<td>25 AA AV MI DM EN LO QU RE SD UW</td>
<td></td>
</tr>
<tr>
<td>6801/3</td>
<td>25 AA AV DM EN LO MI RE SD UW</td>
<td>AR IT</td>
</tr>
<tr>
<td>6809</td>
<td>25 AV DM EN LO MI RE SD UW</td>
<td>IT</td>
</tr>
<tr>
<td>68000</td>
<td>25 AV EN LO QU RE SD UN UW</td>
<td>MX IT MT LA UW</td>
</tr>
<tr>
<td>68HC11</td>
<td>AV CY LO RE SD UW</td>
<td>AR IT</td>
</tr>
<tr>
<td>NSC800</td>
<td>25 EN RE</td>
<td>MT</td>
</tr>
<tr>
<td>8048-50/41</td>
<td>25 AA AV CY LO MI RE SD UN</td>
<td></td>
</tr>
<tr>
<td>8051/31</td>
<td>25 AA AV CY LO MI RE SD UN US UW</td>
<td>AR MC</td>
</tr>
<tr>
<td>8080</td>
<td>EN LO UN</td>
<td></td>
</tr>
<tr>
<td>8085</td>
<td>25 AA AV CY EN LO MI RE SD UW</td>
<td>MT</td>
</tr>
<tr>
<td>8086/8</td>
<td>25 AV CY EN SD SP SW UN UW</td>
<td>MX MS MT LA</td>
</tr>
<tr>
<td>8096</td>
<td>25 AA AV CY LO UN</td>
<td></td>
</tr>
<tr>
<td>Z-8</td>
<td>25 AV AA CY EN LO RE SD UW</td>
<td>MX KY LA, AR</td>
</tr>
<tr>
<td>Z-80 64180</td>
<td>25 AV AA EN LO MT RE SD US UW</td>
<td></td>
</tr>
<tr>
<td>Z-80000</td>
<td>25 EN RE UN</td>
<td></td>
</tr>
</tbody>
</table>

Vendor List begins on next page.

2500 AD Software (303) 369-5001 or 1-800-843-8144
Avocet Systems (302) 734-0151 or 800-448-8500

March 25, 1987
IT  Introl, (414) 276-2937.
    C cross compiler for 6801, 6301, 6805, 6809, 68HC11,
    68000, 68020. $1950.

KY  KYSO, (503) 389-3452.
    C cross compiler for Z80.

LA  Lattice, (312) 858-7950.
    C cross compiler for 68000, 8088, Z80. $500.

LO  Logical Systems
    6184 Teall Station
    Syracuse, NY 13217  (315) 457-9416
    Cross-assemblers for a variety of processors.

MC  MicroComputer Control, (609) 466-1751.
    C cross compiler for 8051. $1495.

MI  Midwest Micro-DelTek, Inc.
    5930 Brooklyn Blvd.
    Brooklyn Center, MN 55429, (612) 560-6530.
    Limited macros, cross reference, conditionals, 1K of
    object/minute. $300.

MS  MicroSoft
    10700 Northup Way
    Bellevue, WA 98004. C compiler for 8086, 8088. $495.
    As of release 4.0 of their software, MicroSoft does not make
    ROMable code directly. You can purchase utilities which are
    supposed to make the output of the MicroSoft compiler into
    ROMable code.

NOTE:
Microtec Research is not the same as "Microtek." The
MicroTek symbol table format referred to in the SYMTYPE menu is
compatible with the "MicroTek/New Micro" products.

MT  Microtec Research
    Box 60337
    Santa Clara, CA 94088, (408) 733-2919.
    C cross compiler for 68000, 68008, 68010, 68020. $1750.
    C cross compiler for 8085, Z80, 64180, 8088, 8086,
    80188. $1550.

NM  MicroTek/New Micro
    Supports a wide variety of processors. Call for latest
    product availability.  (213) 538-5369.

March 25, 1987            Page B-3            -- Appendix B --
UniWare
Software Development Systems
3110 Woodcreek Dr.
Downers Grove, IL 60515, (312) 971-8170.
  8 and 16-bit cross-assemblers, $295.
  C cross-compiler for 68000, $595.

Westico
25 Vanzant St.
Norwalk, CT 06885, (203) 853-6880. $225 Macro, $225 Linker.

Western Wares
Box C,
Norwood, CO 81423, (303) 327-4898. $395.

All of the Intel Series III MDS software can be run on the IBM PC with the UDI package from Real-Time Computer Science Corp., P.O. Box 3000-886, Camarillo, CA 93011, (805) 482-0333 ($500), or the ACCESS package from Genesis Microsystems, 196 Castro St., Mountain View, CA 94041, (415) 964-9001.
A number of people have asked for an explanation of the files that are on the UniLab disk. Here is a directory of the UniLab Sales Demo:

<table>
<thead>
<tr>
<th>File</th>
<th>Type</th>
<th>Size</th>
<th>Date</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMMAND</td>
<td>COM</td>
<td>23210</td>
<td>3-07-85</td>
<td>1:43p</td>
</tr>
<tr>
<td>AUTOEXEC</td>
<td>BAT</td>
<td>128</td>
<td>5-23-86</td>
<td>1:43p</td>
</tr>
<tr>
<td>CONFIG</td>
<td>SYS</td>
<td>128</td>
<td>5-06-86</td>
<td>8:03a</td>
</tr>
<tr>
<td>INSTALL</td>
<td>BAT</td>
<td>2176</td>
<td>4-09-86</td>
<td>11:28a</td>
</tr>
<tr>
<td>EDIT33</td>
<td>VIR</td>
<td>8533</td>
<td>5-19-87</td>
<td>8:56a</td>
</tr>
<tr>
<td>UTIL33</td>
<td>VIR</td>
<td>4450</td>
<td>5-19-87</td>
<td>8:57a</td>
</tr>
<tr>
<td>L1B331</td>
<td>VIR</td>
<td>8578</td>
<td>5-22-87</td>
<td>10:46a</td>
</tr>
<tr>
<td>L2B331</td>
<td>VIR</td>
<td>8873</td>
<td>5-22-87</td>
<td>10:46a</td>
</tr>
<tr>
<td>NEN331</td>
<td>VIR</td>
<td>6643</td>
<td>5-22-87</td>
<td>10:47a</td>
</tr>
<tr>
<td>UL331</td>
<td>VIR</td>
<td>3276</td>
<td>5-22-87</td>
<td>10:47a</td>
</tr>
<tr>
<td>OLSASM331</td>
<td>VIR</td>
<td>6467</td>
<td>5-22-87</td>
<td>10:48a</td>
</tr>
<tr>
<td>ULZ80</td>
<td>EXE</td>
<td>46176</td>
<td>5-24-87</td>
<td>3:05p</td>
</tr>
<tr>
<td>ULZ80</td>
<td>OPR</td>
<td>2962</td>
<td>5-24-87</td>
<td>3:05p</td>
</tr>
<tr>
<td>ULZ80</td>
<td>HCR</td>
<td>14602</td>
<td>5-22-87</td>
<td>2:46p</td>
</tr>
<tr>
<td>UNI331</td>
<td>OVL</td>
<td>39744</td>
<td>5-22-87</td>
<td>10:48a</td>
</tr>
<tr>
<td>UNI331</td>
<td>OPR</td>
<td>1130</td>
<td>5-22-87</td>
<td>10:49a</td>
</tr>
<tr>
<td>UNI331</td>
<td>HCR</td>
<td>8633</td>
<td>5-22-87</td>
<td>10:48a</td>
</tr>
<tr>
<td>UNILAB</td>
<td>SCR</td>
<td>41984</td>
<td>5-22-87</td>
<td>10:58a</td>
</tr>
<tr>
<td>Z80</td>
<td>TBL</td>
<td>5155</td>
<td>3-16-87</td>
<td>9:09a</td>
</tr>
<tr>
<td>DEMO2</td>
<td>BIN</td>
<td>384</td>
<td>2-18-84</td>
<td>12:19a</td>
</tr>
<tr>
<td>DEMO3</td>
<td>BIN</td>
<td>2048</td>
<td>2-17-84</td>
<td>11:08a</td>
</tr>
<tr>
<td>DEMO4</td>
<td>BIN</td>
<td>676</td>
<td>3-12-86</td>
<td>11:56a</td>
</tr>
<tr>
<td>DEMO4</td>
<td>SYM</td>
<td>128</td>
<td>3-06-86</td>
<td>8:29a</td>
</tr>
<tr>
<td>DEMO4</td>
<td>SYM</td>
<td>684</td>
<td>3-12-86</td>
<td>2:03p</td>
</tr>
<tr>
<td>DEMO4</td>
<td>C</td>
<td>2699</td>
<td>3-11-86</td>
<td>10:56a</td>
</tr>
<tr>
<td>TESTZ80</td>
<td>BIN</td>
<td>45</td>
<td>9-26-86</td>
<td>11:13a</td>
</tr>
<tr>
<td>DEMO280</td>
<td>TRC</td>
<td>1024</td>
<td>9-26-86</td>
<td>11:01a</td>
</tr>
<tr>
<td>DO</td>
<td>BAT</td>
<td>40</td>
<td>3-30-87</td>
<td>4:07p</td>
</tr>
<tr>
<td>FORMAT</td>
<td>COM</td>
<td>9398</td>
<td>3-07-85</td>
<td>1:43p</td>
</tr>
<tr>
<td>DIRSIZE</td>
<td>EXE</td>
<td>6972</td>
<td>9-15-86</td>
<td>9:02a</td>
</tr>
</tbody>
</table>

30 File(s)  49152 bytes free
There are 30 files here on a double-density double-sided diskette. Note that there are only 49,251 bytes free. This is not enough room to also put the PPA files on. If PPA demos are given, you will either need to take some of these files off this disk to make room, or demo it on a hard disk (preferred, anyway). (With the OptiLab being dropped, there will have to be some adjustments in the PPA demo.)

Some of these files are required, some are optional. The UniLab system requires the next three files to be on every disk when the system boots up:

**COMMAND.COM**
This is DOS's file, required to boot up any diskette. We do not deliver this with UniLab diskettes for customers, since we haven't licensed it from IBM or Microsoft. It is put on the sales disk for convenience only, and should not be distributed outside of Orion.

**AUTOEXEC.BAT**
This file sets up the two "environment strings" with the DOS SET command. This is absolutely essential for the UniLab to operate. This tells the UniLab system where to look for its own files and the glossary.

The user can type these two SET commands even after the system boots up to establish these environments. DOS executes the commands in any file named AUTOEXEC.BAT when it first boots up. If you have other things that are initialized by an AUTOEXEC.BAT file (clock, calender, etc.), then the two lines in this AUTOEXEC.BAT file should be added to your own AUTOEXEC.BAT file. You can see the environment setting commands in this file by typing TYPE AUTOEXEC.BAT.

Note that these environments are entirely separate from the DOS PATH command. You can still have a PATH set up which does not need these environment strings, but UniLab must have these established before it is called.

These "environments" are a feature of UniLab. The user can be in his own directory doing cross-assembly or whatever, and type something like:

```
/UNI/ULZ80
```

to invoke the UniLab. The current directory will not change, and when the DOS DIR command is executed from UniLab, the files in the current directory will be shown. This also allows the user to keep multiple DDB's in separate directories while sharing a common glossary.
The following files are not essential to running UniLab:

- DEMO2.BIN - Sales demo with bug.
- DEMO3.BIN - Sales demo with bug fixed.
- DEMO4.BIN - Binary object code produced by DEMO4.C.

- DEMO4.SYM - Symbols for sales DEMO2,3.BIN.
- DEMO4.SYN - Symbols and map file for DEMO4.BIN

- DEMO4.C - High level source for DEMO4.BIN

- TESTZ80.BIN - LTARG in a binary file. Use 0 7FF BINLOAD.
- DEMOZ80.TRC - Trace of LTARG. Use TSHOW.

- DO.BAT - Master clone command. Will make a copy of this diskette.

- FORMAT.COM - DOS command to format a diskette.
- DIRSIZE.EXE - This checks the number of files on the diskette, and is called by the DO.BAT file to make sure that the copy was done correctly.
OPTI / UNILAB DEMO SEQUENCE

Starting from command mode:

press: F10 Shows you main menu. Explain that the menus are learning tools, show them the various options within the main menu. Then explain that we will demonstrate in the command mode.

press: F10 Here we are in command mode.

press: F2 The screen splits.

press: END The cursor jumps to upper window.

type: 0 200 BINLOAD DEMO 3. BIN
The binary file is loaded into emulation ROM.

press: F9 STARTUP is executed. Trace fills upper 1/2 of the screen. Bring their attention to the flashing LEDs on the target.

type: RES- The target stops. The processor is held in RESET state.

type: 80 DN Disassembly appears in upper right window.

press: END Cursor jumps to lower window.

type: 88 AS New trace appears. We see the value 01 going out to port 79. Program continues to execute.

type: RESET 88 RB Program is stopped just before value 01 goes out to port.

type: N The program single-steps.

type: LP Program is released, runs until just after 80 goes out to port 79.

repeat: LP Notice LED stepping along. This is it for a basic demo. We can continue from here, and show some visually attractive features. See next page for remainder of demo.
OPTI /UNI DEMO CONTINUED

type: 40 MODIFY The bottom 1/2 of screen is wiped clean, then a dump of memory appears.

type: <hexadecimal numbers> Notice the memory locations are altered.

press: CTRL ↑ At the same time. Now the cursor is in the ASCII display area.

type: <an ASCII string> The memory locations are altered — display changes in both HEX display and ASCII display area.

press: ESC So that your changes are not saved.

type: 90 MODIFY Because now we are actually going to change the machine code.

alter the two locations, 91 & 92, so that they now contain 00 & 18.

press: END Your changes are saved.

press: F9 The program starts up. Behaves very different now.

type: 80 DN Provides a disassembly showing what you changed.

type: 90 ASM Invokes the line by line assembler at address 90.

at the 90 prompt type: JP 88 Press a second carriage return to get out of assembler.

press: F9 Starts the program up.

press: F10 Complete demo with the menu system.

press: F8 Gets ToolKit submenu.


press: any key to continue — return to the ToolKit menu.

press: F10 Now you are back in main menu.
ABOUT DEMO3.BIN

This assembly language program lights up each LED on the target board in sequence. These LEDs are connected to port 79. (Sequence is 1, 80, 40, 20, 10, 8, 4, 2, 1, etc.)

You can alter one instruction so that the program now lights up only the two LEDs at each end, first one then the other. (Sequence is 1, 80, 1, 80, etc.)

Program sections

; The first section of the program initializes some registers and ; then loads a section of program memory into RAM.
0000 31FE18 LD SP,18FE ; Initialize Stack Pointer
0003 DD213412 LD IX,1234 ; Init IX to randomn value
0007 FD217856 LD IY,5678 ; Init IY to randomn value
000B 210019 LD HL,1900 ; These six instructions simply
000E 3600 LD (HL),0 ; demonstrate how the bus
0010 34 INC (HL) ; state analyzer captures write
0011 34 INC (HL) ; and read information from the
0012 14 INC D ; bus.
0013 14 INC D

; Now set up registers for the move of a block of memory from ; ROM to RAM. We will move twenty bytes from ROM starting at ; address 100 up to RAM starting at address 1800.
0014 210001 LD HL,100 ; pointer to source
0017 012000 LD BC,20 ; # of bytes to move
001A 110018 LD DE,1800 ; pointer to destination
001D ED80 LDIR ; perform the move

; The next three instructions are, as far as I can tell, ; pointless
001F 21FFFF LD HL,FFFF
0022 3E18 LD A,18
0024 77 LD (HL),A

; This is the jump up to the output routine and main loop. We ; jump around the reserved and overlay areas.
0025 C30000 JP 80

-- About DEMO3 --

Page 1

April 17, 1987
; Up here at address 80 we first (pointlessly) load a value into
; register D.
0080 16FF           LD D,FF

; Then we load a value into A which is put out to port 7B.
; This is the control register for the port chip.
0082 3E80           LD A,80
0084 D37B           OUT (7B),A

; Next we put a one into register A.
0086 3E01           LD A,1

; Now, put the value in register A out to port 79. The LED on
; the end lights up. During normal operation of the program,
; this instruction will be jumped to after each call to the
; delay loop (the program calls A0, then jumps to 88 after the
; return from the routine at A0).
0088 D379           OUT (79),A

; Next, rotate the value in register A. Since only one bit in
; the register is turned on, this will cause the bit to chase
; around the LEDs in a boring sequence. For fun, you can set a
; breakpoint at this place, put a different value into A
; (say, 7F), and then let that get rotated around. The command
; sequence to do this is:  RESET 88 RB
;                       7F01 =AF
;                       RZ
008A 0F              RRCA

; The next two instructions are pointless.
008B .14            INC D
008C 00              NOP

; Next we call up to the subroutine that causes a delay between
; writes to the port. Without this delay, all of the LEDs would
; appear to be continually, faintly lit. When the program
; returns from the delay subroutine, we jump back to the output
; instruction at address 88.
; You can alter that jump instruction, and instead jump up to
; address 1800. At 1800 is the code that was moved from ROM to
; RAM by the LDIR instruction at address 1D. If you make the
; change (JP 1800 instead of JP 88), then the LEDs will flash
; in a distinctively different pattern.
008D CDAA000          CALL A0
0090 C38800           JP 88

-- About DE03 --

Page 2

April 17, 1987
; This is the delay loop at address A0. First it loads the
; delay value into the HL register, then goes through an inner
; loop that decrements the L register. Once the L register
; is down to zero, the H register is decremented once and then
; the program jumps back to the inner loop, decrementing the L
; register 100 (hex) times. Eventually the H register counts
down to zero, and the program falls down to the RET (return)
; instruction.
00A0 21F40  LD HL,40FF
00A3 2D    DEC L       ; Decrement L.
00A4 20FD  JR NZ,A3    ; Jump back if L is not yet zero.
00A6 25    DEC H       ; Decrement H.
00A7 20FA  JR NZ,A3    ; Jump back if H not yet zero.
00A9 C9    RET       ; Return to the place called from.

; This is the code up at address 1800. It gets put here by the
; LDIR at address 1D.
; This code puts the value one out to port 79, then rotates the
; one in register A (so it is now an 80). After that, it calls
; the delay loop at A0 (previous page) and then jumps to the
; output command at address 88.
; This code will never be executed unless you alter the code
; at address 90, so that it jumps to 1800 rather than to 88.
; If you make that change, the code here will put 01 out to
; port 79, then the code at address 88 will put an 80 out to
; port 79.
1800 16FF  LD D,FF
1802 3E80  LD A,80
1804 D37B  OUT (7B),A
1806 3E01  LD A,1
1808 D37F  OUT (79),A
180A 0F    RRCA
180B 14    INC D
180C 00    NOP
180D CDA000 CALL A0
1810 C38800 JP 88
So, the normal flow of the program is:

**ADDRESS**       **DESCRIPTION**
**REGISTER INITIALIZATION**
0-13            Initialize registers, demonstrate the increment of a RAM location.

**BLOCK MOVE FROM ROM TO RAM**
14-1E:    Prepare registers for the block move instruction (LDIR), and then perform it.

**JUMP TO THE MAIN LOOP**
1F-24       Unnecessary processing.
25-28       Jump up to the port initialization/main loop.

**PORT INITIALIZATION**
80-85       Initialize the port chip.
86-87       Initialize the A register.

**MAIN LOOP: OUTPUT TO PORT 79**
88-89       Put the value in the A register out to port 79.
8A          Rotate right (circular) the value in the A register.
8B-8C       Unnecessary processing.
8D-8F       Call the delay routine at A0.

**DELAY LOOP SUB-ROUTINE**
A0-A8       Load 40FF into the HL register, and then decrement that register down to zero.
A9          Return to the address right after the one this subroutine was called from.

**MAIN LOOP: JUMP BACK TO OUTPUT COMMAND**
90-92       The delay routine at A0 returns to address 90.
The instruction at address 90 jumps to the output command at address 88.
The demo board is small Z80 controller. It consists of a Z80 u-p together with 2K of RAM, 2K of ROM power supply regulation and a PPI device. The system is used to demonstrate the operation of the UDL when used with the demo2.bin program that has been written for it. The small board is also used to demonstrate the ease with the UDL is inter-connected to a target system. The demo board also serves as a very adequate Z80 development system and can be sold for $99.00.

The demo program can be loaded in the emulated ROM of the target system from the system ROM chip as provided or from the binary object file called demo2.bin. The system 2K RAM is located in the address range 1800 to 1FFF, while the system ROM is located at address range 0 to 7FF.

The demo2 program is loaded and the analyser is used to trace the first cycles. This is usually done from the menu mode and the first cycles are shown without the augmentation of the symbols which are available after of the loading of the symbol tables. The first few program steps are then explained and gone through as a vehicle to show the layout of the trace display. The beginning of the block move section of the programs structure is pointed up and the second page of the trace display is shown by pressing the number 1 (as shown on screen). The highly boring nature of this display is noted and the subject of trace depth and filtering functions is bought up. A brief review of the command vs menu operation provides the lead to "esc", "esc" and access the command mode. On the way to the command mode the HELP screen is encountered and explained. Once in command mode pressing F9 provides a reset and the re-display of the trace as shown previously, the difference is now you are in command mode and have a command cursor at the bottom of the screen. Pressing F5 takes you to the second page exactly as you had it before. The topic of filtering is explored and demonstrated by typing "ONLY WRITE S". The system resets and refills the trace buffer only with the write cycles. The benefits of this capability is explored and you proceed on by returning to the initial display. Ask for questions etc, etc.

Now the time to point out the fact that the program has crashed. A brief discussion of the structure of the program and that since it has crashed the possibility that it has gone outside the expected program range may exist. You now ask the UDL to trigger on the crash condition by typing "NORMB FETCH NOT 0 TO 100 ADR S" explain the structure of the command and the triggering features and relate those features to the associated benefits. Then you press "return" The system will reset as well as start to run as evidenced by the LEDs. Point out the display on screen as well not triggered light on the UDL. Make a point to show the extinguishing of the trigger light on the UDL as the target system crashes and the screen displays the events leading up to the event. Note the RET and the 17FF address. A brief word about the fact that RAM is between 1800 and 1FFF is in order. A quick story about catching this type of event and the
babysitting capability can be inserted here. The reset and repeat operation can now be demonstrated, by typing 7 SR.

After the system has repeated the same trace showing that the "crash condition" is the same each time we can now try to uncover the cause of the offending event, namely the attempt to read from a non-existing RAM address. The use of the command "lAFTER A8 ADR S" shows that the stack is growing to the point where it overflows. The cause of this stack growth is identified by using the command "A8 AS" this shows the PUSH at memory location 8C the hex code for which is D5. Explain that the occurrence of the push without a POP will cause what stack overflow that is occurring.

A brief discussion of memory patching may now be inserted and then demonstrated by changing the D5 to a 00. The 00 is the hex code for a no-op. This operation is performed by using the command "0 8C M1". The fact that the BUG has been fixed is shown by re-typing A8 AS and noting that the push has been replaced by a no op, and that the program no longer crashes. Now "S" restarts and F5 fills the trace with boring delay loop cycles. You may now explain how these boring cycles may be eliminated as block. This done by using "ONLY NOT A0 TO B0 ADR AFTER 80 ADR S" This shows program operation without delay loop. The basic demonstration of the analyser is for the most part complete.

Proceed on to the debugger. Discuss running to a breakpoint, running to multiple breakpoints etc. "RESET 80 RB" runs to step 80 but does not execute. The internal registers are displayed and may be modified etc.
<table>
<thead>
<tr>
<th>cy#</th>
<th>CONT</th>
<th>ADR</th>
<th>DATA</th>
<th>HDATA</th>
<th>MISC</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>B7</td>
<td>0000</td>
<td>31FE18</td>
<td>LD SP,18FE</td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>3</td>
<td>B7</td>
<td>0003</td>
<td>DD213412</td>
<td>LD IX,1234</td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>7</td>
<td>B7</td>
<td>0007</td>
<td>PDD217856</td>
<td>LD IIY,5678</td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>B</td>
<td>B7</td>
<td>000B</td>
<td>210019</td>
<td>LD HL,1900</td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>E</td>
<td>B7</td>
<td>000E</td>
<td>34</td>
<td>INC (HL)</td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>F</td>
<td>F7</td>
<td>1900</td>
<td>48 read</td>
<td></td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>10</td>
<td>D7</td>
<td>1900</td>
<td>49 write</td>
<td></td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>11</td>
<td>B7</td>
<td>000F</td>
<td>34</td>
<td>INC (HL)</td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>12</td>
<td>F7</td>
<td>1900</td>
<td>49 read</td>
<td></td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>13</td>
<td>D7</td>
<td>1900</td>
<td>4A writer</td>
<td></td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>14</td>
<td>B7</td>
<td>0010</td>
<td>14</td>
<td>INC D</td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>15</td>
<td>B7</td>
<td>0011</td>
<td>14</td>
<td>INC D</td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>16</td>
<td>B7</td>
<td>0012</td>
<td>14</td>
<td>INC D</td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>17</td>
<td>B7</td>
<td>0013</td>
<td>14</td>
<td>INC D</td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>18</td>
<td>B7</td>
<td>0014</td>
<td>210001</td>
<td>LD HL,100</td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>1B</td>
<td>B7</td>
<td>0017</td>
<td>012000</td>
<td>LD BC,20</td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>1E</td>
<td>B7</td>
<td>001A</td>
<td>110018</td>
<td>LD DE,1800</td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>21</td>
<td>B7</td>
<td>001D</td>
<td>EDB0</td>
<td>LDIR</td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>23</td>
<td>F7</td>
<td>0100</td>
<td>16 read</td>
<td></td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>24</td>
<td>D7</td>
<td>1800</td>
<td>16 write</td>
<td></td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>25</td>
<td>B7</td>
<td>001D</td>
<td>EDB0</td>
<td>LDIR</td>
<td>11111111 11111111</td>
</tr>
</tbody>
</table>

F5 or TR (trace resume)  TT (top) n TN (from step n)  T (from n=-5 )

<table>
<thead>
<tr>
<th>cy#</th>
<th>CONT</th>
<th>ADR</th>
<th>DATA</th>
<th>HDATA</th>
<th>MISC</th>
</tr>
</thead>
<tbody>
<tr>
<td>27</td>
<td>F7</td>
<td>0101</td>
<td>FF read</td>
<td></td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>28</td>
<td>D7</td>
<td>1801</td>
<td>FF write</td>
<td></td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>29</td>
<td>B7</td>
<td>001D</td>
<td>EDB0</td>
<td>LDIR</td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>2B</td>
<td>F7</td>
<td>0102</td>
<td>3E read</td>
<td></td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>2C</td>
<td>D7</td>
<td>1802</td>
<td>3E write</td>
<td></td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>2D</td>
<td>B7</td>
<td>001D</td>
<td>EDB0</td>
<td>LDIR</td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>2F</td>
<td>F7</td>
<td>0103</td>
<td>80 read</td>
<td></td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>30</td>
<td>D7</td>
<td>1803</td>
<td>80 write</td>
<td></td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>31</td>
<td>B7</td>
<td>001D</td>
<td>EDB0</td>
<td>LDIR</td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>33</td>
<td>F7</td>
<td>0104</td>
<td>D3 read</td>
<td></td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>34</td>
<td>D7</td>
<td>1804</td>
<td>D3 write</td>
<td></td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>35</td>
<td>B7</td>
<td>001D</td>
<td>EDB0</td>
<td>LDIR</td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>37</td>
<td>F7</td>
<td>0105</td>
<td>7B read</td>
<td></td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>38</td>
<td>D7</td>
<td>1805</td>
<td>7B write</td>
<td></td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>39</td>
<td>B7</td>
<td>001D</td>
<td>EDB0</td>
<td>LDIR</td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>3B</td>
<td>F7</td>
<td>0106</td>
<td>3E read</td>
<td></td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>3C</td>
<td>D7</td>
<td>1806</td>
<td>3E write</td>
<td></td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>3D</td>
<td>B7</td>
<td>001D</td>
<td>EDB0</td>
<td>LDIR</td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>3F</td>
<td>F7</td>
<td>0107</td>
<td>01 read</td>
<td></td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>40</td>
<td>D7</td>
<td>1807</td>
<td>01 write</td>
<td></td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>41</td>
<td>B7</td>
<td>001D</td>
<td>EDB0</td>
<td>LDIR</td>
<td>11111111 11111111</td>
</tr>
</tbody>
</table>

F5 or TR (trace resume)  TT (top) n TN (from step n)  T (from n=-5 )
ONLY WRITE S resetting

Not done till delay count = 0. Any key aborts. TD dumps trace buffer.
TRIGGER Wait Status: DELAY Count = 0 PASS Count = NONE (from top of buff.

cy# CONT ADR DATA HDATA MISC
F01 FF FFCFC FF read 11111111 11111111
F02 D7 1900 4B write 11111111 11111111
F03 D7 1900 4C write 11111111 11111111
F04 D7 1800 16 write 11111111 11111111
F05 D7 1801 FF write 11111111 11111111
F06 D7 1802 3E write 11111111 11111111
F07 D7 1803 80 write 11111111 11111111
F08 D7 1804 D3 write 11111111 11111111
F09 D7 1805 7B write 11111111 11111111
F0A D7 1806 3E write 11111111 11111111
F0B D7 1807 01 write 11111111 11111111
F0C D7 1808 D3 write 11111111 11111111
F0D D7 1809 79 write 11111111 11111111
F0E D7 180A 0F write 11111111 11111111
F0F D7 180B 14 write 11111111 11111111
F10 D7 180C 62 write 11111111 11111111
F11 D7 180D 6B write 11111111 11111111
F12 D7 180E 2D write 11111111 11111111
F13 D7 180F 20 write 11111111 11111111
F14 D7 1810 FD write 11111111 11111111
F15 D7 1811 25 write 11111111 11111111
F5 or TR (trace resume) TT (top) n TN (from step n) T (from n=-5)

NORMB FETC NOT 0 TO 100 ADR S resetting

Not done till delay count = 0. Any key aborts. TD dumps trace buffer.
cy# CONT ADR DATA HDATA MISC
-5 B7 00A6 20FA JR NZ,A2 11111111 11111111
-3 B7 00A8 C9 RET 11111111 11111111
-2 F7 17FE FF read 11111111 11111111
-1 F7 17FF FF read 11111111 11111111
 0 B7 FFFF 1831 JR 32 11111111 11111111
 2 B7 0032 57 LD D,A 11111111 11111111
 3 B7 0033 F3 DI 11111111 11111111
 4 B7 0034 00 MOP 11111111 11111111
End of Trace Buffer TT (top) n TN (from step n) T (from n=-5)

7 SR resetting

cy# CONT ADR DATA HDATA MISC
-5 B7 00A6 20FA JR NZ,A2 11111111 11111111
-3 B7 00A8 C9 RET 11111111 11111111
-2 F7 17FE FF read 11111111 11111111
-1 F7 17FF FF read 11111111 11111111
 0 B7 FFFF 1831 JR 32 11111111 11111111
 2 B7 0032 57 LD D,A 11111111 11111111
 3 B7 0033 F3 DI 11111111 11111111

4
A8 ADR S resetting

<table>
<thead>
<tr>
<th>Cy#</th>
<th>CONT</th>
<th>ADR</th>
<th>DATA</th>
<th>HDATA</th>
<th>MISC</th>
</tr>
</thead>
<tbody>
<tr>
<td>F01</td>
<td>FF</td>
<td>FFFF</td>
<td>FF</td>
<td>read</td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>F02</td>
<td>B7</td>
<td>00A8</td>
<td>C9</td>
<td>RET</td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>F03</td>
<td>F7</td>
<td>18FA</td>
<td>90</td>
<td>read</td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>F04</td>
<td>B7</td>
<td>00A8</td>
<td>C9</td>
<td>RET</td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>F05</td>
<td>F7</td>
<td>18F8</td>
<td>90</td>
<td>read</td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>F06</td>
<td>B7</td>
<td>00A8</td>
<td>C9</td>
<td>RET</td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>F07</td>
<td>F7</td>
<td>18F6</td>
<td>90</td>
<td>read</td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>F08</td>
<td>B7</td>
<td>00A8</td>
<td>C9</td>
<td>RET</td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>F09</td>
<td>F7</td>
<td>18F4</td>
<td>90</td>
<td>read</td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>F0A</td>
<td>B7</td>
<td>00A8</td>
<td>C9</td>
<td>RET</td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>F0B</td>
<td>F7</td>
<td>18F2</td>
<td>90</td>
<td>read</td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>F0C</td>
<td>B7</td>
<td>00A8</td>
<td>C9</td>
<td>RET</td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>F0D</td>
<td>F7</td>
<td>18F0</td>
<td>90</td>
<td>read</td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>F0E</td>
<td>B7</td>
<td>00A8</td>
<td>C9</td>
<td>RET</td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>F0F</td>
<td>F7</td>
<td>18EE</td>
<td>90</td>
<td>read</td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>F10</td>
<td>B7</td>
<td>00A8</td>
<td>C9</td>
<td>RET</td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>F11</td>
<td>F7</td>
<td>18EC</td>
<td>90</td>
<td>read</td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>F12</td>
<td>B7</td>
<td>00A8</td>
<td>C9</td>
<td>RET</td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>F13</td>
<td>F7</td>
<td>18EA</td>
<td>90</td>
<td>read</td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>F14</td>
<td>B7</td>
<td>00A8</td>
<td>C9</td>
<td>RET</td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>F15</td>
<td>F7</td>
<td>18E8</td>
<td>90</td>
<td>read</td>
<td>11111111 11111111</td>
</tr>
</tbody>
</table>

F5 or TR (trace resume) TT (top) n TN (from step n) T (from n=-5)

A8 AS resetting

<table>
<thead>
<tr>
<th>Cy#</th>
<th>CONT</th>
<th>ADR</th>
<th>DATA</th>
<th>HDATA</th>
<th>MISC</th>
</tr>
</thead>
<tbody>
<tr>
<td>-5</td>
<td>B7</td>
<td>00A3</td>
<td>20FD</td>
<td>JR NZ,A2</td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>-3</td>
<td>B7</td>
<td>00A5</td>
<td>25</td>
<td>DEC H</td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>-2</td>
<td>B7</td>
<td>00A6</td>
<td>20FA</td>
<td>JR NZ,A2</td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>0</td>
<td>B7</td>
<td>00A8</td>
<td>C9</td>
<td>RET</td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>1</td>
<td>F7</td>
<td>18FA</td>
<td>90</td>
<td>read</td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>2</td>
<td>F7</td>
<td>18FB</td>
<td>00</td>
<td>read</td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>3</td>
<td>B7</td>
<td>0090</td>
<td>C38800</td>
<td>JP 88</td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>6</td>
<td>B7</td>
<td>00B8</td>
<td>D379</td>
<td>OUT (79),A</td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>8</td>
<td>5F</td>
<td>8079</td>
<td>80</td>
<td>out</td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>9</td>
<td>B7</td>
<td>008A</td>
<td>0F</td>
<td>RRCA</td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>A</td>
<td>B7</td>
<td>008B</td>
<td>14</td>
<td>INC D</td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>B</td>
<td>B7</td>
<td>008C</td>
<td>D5</td>
<td>PUSH DE</td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>C</td>
<td>D7</td>
<td>18FB</td>
<td>01</td>
<td>write</td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>D</td>
<td>D7</td>
<td>18FA</td>
<td>20</td>
<td>write</td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>E</td>
<td>B7</td>
<td>008D</td>
<td>CDA000</td>
<td>CALL A0</td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>11</td>
<td>D7</td>
<td>18F9</td>
<td>00</td>
<td>write</td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>12</td>
<td>D7</td>
<td>18F8</td>
<td>90</td>
<td>write</td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>13</td>
<td>B7</td>
<td>00A0</td>
<td>62</td>
<td>LD H,D</td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>14</td>
<td>B7</td>
<td>00A1</td>
<td>6B</td>
<td>LD L,E</td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>15</td>
<td>B7</td>
<td>00A2</td>
<td>2D</td>
<td>DEC L</td>
<td>11111111 11111111</td>
</tr>
<tr>
<td>16</td>
<td>B7</td>
<td>00A3</td>
<td>20FD</td>
<td>JR NZ,A2</td>
<td>11111111 11111111</td>
</tr>
</tbody>
</table>

F5 or TR (trace resume) TT (top) n TN (from step n) T (from n=-5)
Not done till delay count = 0. Any key aborts. TD dumps trace buffer.

```
  cy#  CONT  ADR  DATA       HDATA       MISC
-5  B7   00A3  20FD      JR NZ,A2      11111111  11111111
-3  B7   00A5  25       DEC H       11111111  11111111
-2  B7   00A6  20FA      JR NZ,A2      11111111  11111111
  0  B7   00A8  C9       RET         11111111  11111111
 1  F7   18FC  90      read         11111111  11111111
 2  F7   18FD  00      read         11111111  11111111
 3  B7   0090  C38800   JP 88       11111111  11111111
 6  B7   0088  D379      OUT (79),A  11111111  11111111
 8  5F   8079  80      out         11111111  11111111
 9  B7   008A  0F       RRCA       11111111  11111111
 A  B7   008B  14       INC D       11111111  11111111
 B  B7   008C  00       NOP        11111111  11111111
 C  B7   008D  CDA000   CALL A0    11111111  11111111
 F  D7   18FD  00      write      11111111  11111111
10  D7   18FC  90      write      11111111  11111111
11  B7   00A0  62      LD H,D      11111111  11111111
12  B7   00A1  6B      LD L,E      11111111  11111111
13  B7   00A2  2D      DEC L       11111111  11111111
14  B7   00A3  20FD     JR NZ,A2    11111111  11111111
16  B7   00A2  2D      DEC L       11111111  11111111
17  B7   00A3  20FD     JR NZ,A2    11111111  11111111
```

F5 or TR (trace resume)  TT (top) n TN (from step n)  T (from n=-5 )

```
  cy#  CONT  ADR  DATA       HDATA       MISC
 19  B7   00A2  2D      DEC L       11111111  11111111
 1A  B7   00A3  20FD     JR NZ,A2    11111111  11111111
 1C  B7   00A2  2D      DEC L       11111111  11111111
 1D  B7   00A3  20FD     JR NZ,A2    11111111  11111111
 1F  B7   00A2  2D      DEC L       11111111  11111111
 20  B7   00A3  20FD     JR NZ,A2    11111111  11111111
 22  B7   00A2  2D      DEC L       11111111  11111111
 23  B7   00A3  20FD     JR NZ,A2    11111111  11111111
 25  B7   00A2  2D      DEC L       11111111  11111111
 26  B7   00A3  20FD     JR NZ,A2    11111111  11111111
 28  B7   00A2  2D      DEC L       11111111  11111111
 29  B7   00A3  20FD     JR NZ,A2    11111111  11111111
 2B  B7   00A2  2D      DEC L       11111111  11111111
 2C  B7   00A3  20FD     JR NZ,A2    11111111  11111111
 2E  B7   00A2  2D      DEC L       11111111  11111111
 2F  B7   00A3  20FD     JR NZ,A2    11111111  11111111
 31  B7   00A2  2D      DEC L       11111111  11111111
 32  B7   00A3  20FD     JR NZ,A2    11111111  11111111
 34  B7   00A2  2D      DEC L       11111111  11111111
 35  B7   00A3  20FD     JR NZ,A2    11111111  11111111
 37  B7   00A2  2D      DEC L       11111111  11111111
```

F5 or TR (trace resume)  TT (top) n TN (from step n)  T (from n=-5 )
Not done till delay count = 0. Any key aborts. TD dumps trace buffer.

F02 B7 0082 3E80 LD A,80 11111111 11111111 11111111 11111111
F04 B7 0084 D37B OUT (7B),A 11111111 11111111 11111111 11111111
F06 5F 807B 80 out 11111111 11111111 11111111 11111111
F07 B7 0086 3E01 LD A,01 11111111 11111111 11111111 11111111
F09 B7 0088 D379 OUT (79),A 11111111 11111111 11111111 11111111
F0B 5F 0179 01 out 11111111 11111111 11111111 11111111
F0C B7 008A 0F RRCA 11111111 11111111 11111111 11111111
F0D B7 008B 14 INC D 11111111 11111111 11111111 11111111
F0E B7 008C 00 NOP 11111111 11111111 11111111 11111111
F0F B7 008D CDA000 CALL A0 11111111 11111111 11111111 11111111
F12 D7 18FD 00 write 11111111 11111111 11111111 11111111
F13 D7 18FC 90 write 11111111 11111111 11111111 11111111
F14 F7 18FC 90 read 11111111 11111111 11111111 11111111
F15 F7 18FD 00 read 11111111 11111111 11111111 11111111
F16 B7 0090 C38800 JP 88 11111111 11111111 11111111 11111111
F19 B7 0088 D379 OUT (79),A 11111111 11111111 11111111 11111111
F1B 5F 8079 80 out 11111111 11111111 11111111 11111111
F1C B7 008A 0F RRCA 11111111 11111111 11111111 11111111
F1D B7 008B 14 INC D 11111111 11111111 11111111 11111111
F1E B7 008C 00 NOP 11111111 11111111 11111111 11111111

F5 or TR (trace resume) TT (top) n TN (from step n) T (from n=-5 )

cy# CONT ADR DATA HDATA MISC
F1F B7 008D CDA000 CALL A0 11111111 11111111 11111111 11111111
F22 D7 18FD 00 write 11111111 11111111 11111111 11111111
F23 D7 18FC 90 write 11111111 11111111 11111111 11111111
F24 F7 18FC 90 read 11111111 11111111 11111111 11111111
F25 F7 18FD 00 read 11111111 11111111 11111111 11111111
F26 B7 0090 C38800 JP 88 11111111 11111111 11111111 11111111
F29 B7 0088 D379 OUT (79),A 11111111 11111111 11111111 11111111
F2B 5F 4079 40 out 11111111 11111111 11111111 11111111
F2C B7 008A 0F RRCA 11111111 11111111 11111111 11111111
F2D B7 008B 14 INC D 11111111 11111111 11111111 11111111
F2E B7 008C 00 NOP 11111111 11111111 11111111 11111111
F2F B7 008D CDA000 CALL A0 11111111 11111111 11111111 11111111
F32 D7 18FD 00 write 11111111 11111111 11111111 11111111
F33 D7 18FC 90 write 11111111 11111111 11111111 11111111
F34 F7 18FC 90 read 11111111 11111111 11111111 11111111
F35 F7 18FD 00 read 11111111 11111111 11111111 11111111
F36 B7 0090 C38800 JP 88 11111111 11111111 11111111 11111111
F39 B7 0088 D379 OUT (79),A 11111111 11111111 11111111 11111111
F3B 5F 2079 20 out 11111111 11111111 11111111 11111111
F3C B7 008A 0F RRCA 11111111 11111111 11111111 11111111
F3D B7 008B 14 INC D 11111111 11111111 11111111 11111111

F5 or TR (trace resume) TT (top) n TN (from step n) T (from n=-5 )
RESET 80 RB resetting

AF=1801 (sz-a-pnC) BC= 0 DE=1820 HL=FFFF IX=1234 IY=5678 SP=18FE PC= (next step) ok
0080 16FF LD D,FF
N
AF=1801 (sz-a-pnC) BC= 0 DE=FF20 HL=FFFF IX=1234 IY=5678 SP=18FE PC= ok
0082 3E80 LD A,80

AF=1801 (sz-a-pnC) BC= 0 DE=FF20 HL=FFFF IX=1234 IY=5678 SP=18FE PC= (next step) ok
0082 3E80 LD A,80
N
AF=8001 (sz-a-pnC) BC= 0 DE=FF20 HL=FFFF IX=1234 IY=5678 SP=18FE PC= (next step) ok
0084 D37B OUT (7B),A

BYE
FINDING THE BUG IN DEMO2.BIN

After enabling emulation memory (0 to 7ff enable) and loading the demo program (0 to 7ff binload demo2.bin) start it up by using F9 in the command mode. The screen should fill up with a trace of the program's initial execution and the LED's should ripple. In a short while (half-minute or so) the program should crash as is evidenced by the fact that the LED's have ceased rippling. We've got a bug.

FINDING THE BUG: THE COMMANDS (ALWAYS FOLLOWED BY RETURN)

- NORMB FETCH NOT 0 TO 100 ADR S
- 7 SR
- 1AFTER AB ADR S
- AB AS
- 0 BC M!
- AB AS
- ONLY NOT AO TO BO ADR AFTER 80 ADR S

EXPLANATION OF EACH COMMAND IN ORDER OF APPEARANCE:

- NORMB FETCH NOT 0 TO 100 ADR S
NORMB tells the Unilab to clear out any current trigger specs and put the trigger event at the bottom of the buffer. In other words, the majority of the trace buffer will be filled with events before the triggering condition with only five bus cycles after it. The trigger is always labeled cycle 0. FETCH says that you only want to trigger on a fetch and not on some other operation.
NOT 0 TO 100 ADR says that you want the Unilab to trigger when the CPU is operating (in this case, fetching) outside of the address range of 0 TO 100.
S simply means reset and start the processor.

SUMMARY: This command tells the Unilab to freeze the trace buffer (trigger) when the processor tries to fetch from outside the address range of 0 to 100 and put that event at the bottom of the buffer. At cycle 0 in the trace (the trigger event) you should see that the processor tried to fetch from address FFFF, clearly outside its normal range of around 0 to 100.
It tried to fetch from here because garbage was read from RAM.
on the two preceding cycles.

- 7 SR
7 SR tells the UniLab to start the processor again with the same trigger specification and show just 7 cycles. After it does this it will automatically restart and do it again until you press any key. This is just a quick test to see if it just happened to crash this way only this time.

- I AFTER A8 ADR S
This command tells the UniLab that you want the trace buffer to be filled only with what happens at address A8 and 1 (one) cycle after it during multiple passes through the program. This is an example of the UniLab's filtering capabilities. We are using this to see what's happening after A8 because the crash occurs right after A8 when garbage somehow put and eventually read from RAM.

- A8 AS
This is an abbreviated command which tells the UniLab to put the trigger at the top of the buffer and start it at address A8. This could also be entered by typing NORMT A8 S. We use this command to look further down the buffer after A8 to see what it reveals. We find that at address 8C there is a push instruction. Scrolling down through the buffer reveals that this PUSH is never popped. We thus get a stack overflow which is putting garbage in RAM and causing the program to crash.

- 0 8C M!
This command tells the UniLab to store a 0 at address 8C. M! is a command which simply stores a byte of data at the specified address. In this case, we're storing 0 (a NOP instruction in Z80 code) at address 8C in emulation memory in order to nullify the offending PUSH instruction. This is the point where we are actually patching or fixing the program. We could have also used the on-line assembler by entering: 8C ASM (to invoke the on-line assembler at 8C). 0 (entering the NOP instruction at address 8C). Pressing return twice.

- A8 AS
Explained above. Note, however, that at address 8C we now have 0 (Z80 NOP instruction) and not the offending PUSH instruction.

- ONLY NOT A0 TO B0 ADR AFTER B0 ADR S
This command is optional because it is rather bulky. However, it is an interesting example of a combination of both filtering and qualifications used in a trigger spec. This command tells the UniLab that you only want to see what happens after address B0 that is not within the range of A0 to B0. In this way we get to see address 8C three times on a single screen and are able to verify that indeed 0 is in
location BC. A user may do something like this when he or
she already knows that the code in a certain range — in this
case A0 to B0 — is functional and therefore does not want it
filling up valuable space in the trace buffer.
8-bit microprocessor
Separate data and address lines
Multiplexed upper and lower address lines
64K memory addressing
Operating Frequency: DC to 3.2, 5 MHz

Features:
- On-chip clock
- On-chip DMA

<table>
<thead>
<tr>
<th>DEVICE</th>
<th>DIRECT ADDRESSABLE EXTERNAL MEM. K-BYTES</th>
<th>ON-CHIP RAM BYTES</th>
<th>ON-CHIP ROM BYTES</th>
<th>MAX. CLOCK FREQ. (MHz)</th>
<th>INSTRUCTION TIME MIN./MAX. (μs)</th>
<th>INTERRUPTS</th>
<th>TIMER/COUNTER BITS</th>
<th>PRESCALER</th>
<th>BUS MUX/ NON</th>
</tr>
</thead>
<tbody>
<tr>
<td>CDP1802A</td>
<td>64</td>
<td>-</td>
<td>-</td>
<td>3.2</td>
<td>5.0/7.5</td>
<td>•</td>
<td>-</td>
<td>-</td>
<td>NON</td>
</tr>
<tr>
<td>CDP1802B</td>
<td>64</td>
<td>-</td>
<td>-</td>
<td>5.0</td>
<td>3.2/4.8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>NON</td>
</tr>
<tr>
<td>CDP1804A</td>
<td>64</td>
<td>64</td>
<td>2048</td>
<td>5.0</td>
<td>3.2/16.0</td>
<td>•</td>
<td>8</td>
<td>DIV 32</td>
<td>NON</td>
</tr>
<tr>
<td>CDP1806A</td>
<td>64</td>
<td>64</td>
<td>-</td>
<td>5.0</td>
<td>3.2/16.0</td>
<td>•</td>
<td>8</td>
<td>DIV 32</td>
<td>NON</td>
</tr>
<tr>
<td>CDP1808A</td>
<td>64</td>
<td>-</td>
<td>-</td>
<td>5.0</td>
<td>3.2/16.0</td>
<td>•</td>
<td>8</td>
<td>DIV 32</td>
<td>NON</td>
</tr>
</tbody>
</table>

.ILab interface considerations:

- Analyzer Cable 6
- Maximum Clock Speed: 5 MHz clock input (Standard UniLab)
- Processor clock to UniLab clock ratio: 5:

UniLab does not have debug capabilities for 1800 family.

Reserved Areas: None
Overlay area: None

UniLab clock:

RD- MAD
WR- MiWR
K1 Not used
K2 Not used

UDL status lines:
C4 Not Used
C5 Not Used
C6 SCI
C7 SCI

Cycle types:

- fetch 3F
- read 7F
- write 5F
THE_6301/03_MICROPROCESSOR_FAMILY

6301 Microprocessor

0-bit microprocessor
Multiplexed or Non-multiplexed modes
Internal ROM
Internal RAM
'Abundant' on-chip functions
Operating Frequency: 1, 1.5, 2 MHz

7 different modes of operation. UniLab supports those modes that have internal ROM disabled (Modes 1, 2 and 4)

63P01 (piggyback) microprocessor

Supports 6301 Microprocessor
ROMless version; EEPROM socket on top of chip. UniLab supports modes (1-7).

6303R Microprocessor

Supports 6301 Microprocessor
ROMless version
3 different modes of operation; UniLab supports all modes.

6303X and 6303Y Microprocessors

CPU compatible with 6301 Microprocessor
64-pin package
ROMless chip
More RAM
More I/O Ports
More abundant on-chip features

6301X and 6301Y Microprocessors

CPU compatible with 6301 Microprocessor
Internal ROM

6301X has internal ROM; must be used in Mode 1 (internal ROM disabled)
6301Y has internal ROM; must be used in Mode 1 (internal ROM disabled)
Vector Table

<table>
<thead>
<tr>
<th>Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FF00-FF</td>
<td>RES</td>
</tr>
<tr>
<td>FF00-EF</td>
<td>TRAP (address or op-code error)</td>
</tr>
<tr>
<td>FF0C-FF</td>
<td>NMI</td>
</tr>
<tr>
<td>FFFA-FB</td>
<td>SWI (software int)</td>
</tr>
<tr>
<td>FFF8-F9</td>
<td>IRQ</td>
</tr>
</tbody>
</table>

Reserved Areas: FF00-FB or FF01-EF Vector location for software breakpoint.
FF0C-FF Vector location for hardware breakpoint.
FF09-BA Reserved location for interrupt routine (rel).

Overlay area: Starts at FF0B, 10-20 bytes deep; debug operations will not work properly in this area.

Maximum Clock Speed: 10 MHz clock input (Standard UniLab)
Processor clock to UniLab clock ratio: 4:1

UniLab clock:
- RD- E clock
- WR- Not used
- K1 Not used
- K2 Not used

16 status lines:
- 6301/03R: C4 Not Used, C5 Not Used, C6 Not Used, C7 SC2
- 63P01: Not Used, Not Used, Not Used, Not Used
- 6303X/03Y: WR- (Port 7.1), RD- (Port 7.0), LIH (Port 7.3), A/W (Port 7.2)

Cycle types:
- 6301/03R: fetch FF, read FF, write 7F
- 63P01: fetch 7F (int.op.), read 7F/FF, write 7F/FF
- 6303X/03Y: fetch 9F (first byte), read 7F/FF, write 7F/FF

EM63 Emulation Module is proposed for the 6303R processor. Release date to be announced.

Software considerations:

Special commands
- TRAP change breakpoint opcode from SWI (3F) to 00. The 'illegal' opcode gets trapped, vectoring to the installed pointer.
THE 6500 MICROPROCESSOR FAMILY

650X and 651X Microprocessors

8-bit microprocessor
Separate data and address lines
Operating Frequency: 1.2, 3 MHz

Features:
- Interrupt request
- Non-maskable Interrupt
- 64K addressing
- Bus compatible with 6800

6511 and 6514 'Backpack Emulator' (piggyback) chips

Multiplexed or Non-multiplexed modes
Operating frequency: 1.2 MHz

- The 6511 is the PROM prototyping version of the 8-bit, masked-ROM 6500/11 processor.

- The 6514 is the PROM prototyping version of the 8-bit, masked-ROM 6500/41 processor.

6510 and 65010 processors

Multiplexed or Non-multiplexed mode
Operating frequency: 1.2 MHz
Features:
- 192 byte static RAM
- 4 I/O ports
- Two 16-bit programmable timers
- Serial port
- Ten interrupts

Software considerations:

Special commands

=SAVE to change the PAGE 0 location used for saving the A' register.
=DA8R to change external RAM 'blinds' spot.
=INTABR to point to user's IRQ handling routine, if any.
=FRAM if your piggy back system has external RAM, use this one-way configuration word.
<4 for GSP package - for the 6510 processor in Commodore 64 computer. shows all bus cycles.
AL hides 'extra' bus cycles.
Vector Table

<table>
<thead>
<tr>
<th>ADDR</th>
<th>DESC</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFEE-F</td>
<td>IRQ</td>
</tr>
<tr>
<td>FFEC-D</td>
<td>RES</td>
</tr>
<tr>
<td>FFFA-b</td>
<td>NMI</td>
</tr>
</tbody>
</table>

Unilab interface considerations:

Analyzer Cable B for 650X, 651X and 65X10
Analyzer Cable K for piggyback chips (65/11, 65/41)
Reserved Areas: FFEE-F Vector location for software breakpoint.
FFFA-B Vector location for hardware breakpoint.
FFAC-FFBC Reserved area for interrupt routine (rel).
Page 0 location to save the A register (user specified-rel)

FE00-FEFF are used in such a way that you cannot look at external RAM at these addresses. Can be reassigned by user.

Overlay area: Starts at FFBC, 10-20 bytes deep; debug operations will not work properly in this area.

Maximum Clock Speed: 2.5 MHz clock input (Standard Unilab)
3.4 MHz clock input (High Speed Unilab)

Unilab clock:

<table>
<thead>
<tr>
<th>RD</th>
<th>WR</th>
<th>K1</th>
<th>K2</th>
</tr>
</thead>
<tbody>
<tr>
<td>02</td>
<td>Not used</td>
<td>Not used</td>
<td>Not used</td>
</tr>
</tbody>
</table>

Unilab status lines:

<table>
<thead>
<tr>
<th>C4</th>
<th>C5</th>
<th>C6</th>
<th>C7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not Used</td>
<td>Not Used</td>
<td>Sync (active high for op-code fetch)</td>
<td>R/W-</td>
</tr>
<tr>
<td>(On piggyback chips, C7 connected to DE of piggyback socket; C6 not used)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Cycle types:

<table>
<thead>
<tr>
<th>650X w/SYNC</th>
<th>650X w/o SYNC</th>
<th>65/11, 65/41</th>
</tr>
</thead>
<tbody>
<tr>
<td>fetch</td>
<td>FF (first byte)</td>
<td>FF</td>
</tr>
<tr>
<td>read</td>
<td>FF</td>
<td>FF</td>
</tr>
<tr>
<td>write</td>
<td>7F</td>
<td>7F</td>
</tr>
</tbody>
</table>

EM65 Emulation Module available for 6502 and 65002 microprocessors.
Emulated target ROM must be removed from target system for proper operation.
THE 6800 MICROPROCESSOR FAMILY

The 6802 and 6800 are software compatible with the 6800.

The 6800 Microprocessor

8-bit microprocessor
Operating frequency: 1, 1.5, 2 MHz

6800 additional features

- Internal clock oscillator and driver

<table>
<thead>
<tr>
<th>Part</th>
<th>Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC6800</td>
<td>1 MHz</td>
</tr>
<tr>
<td>MC6808</td>
<td>1.5</td>
</tr>
<tr>
<td>MC6800B</td>
<td>2.0</td>
</tr>
</tbody>
</table>

6802 additional features

- Internal clock oscillator and driver
- 128 bytes of internal RAM
- First 32 bytes RAM retainable
  (6802NS lacks this last feature.)

<table>
<thead>
<tr>
<th>Part</th>
<th>Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC6802</td>
<td>1 MHz</td>
</tr>
<tr>
<td>MC6802B</td>
<td>1.5</td>
</tr>
<tr>
<td>MC6802B</td>
<td>2.0</td>
</tr>
</tbody>
</table>

Software considerations:

The 6802 package is necessary if there is internal RAM at addresses 0-128 (decimal). It can be used with 6800 and 6800B, if there is external RAM at those addresses--but we recommend against it.

Special commands:

- 6802 only: =ZP to change the starting address of the four bytes of RAM that are reserved by debug.

---

**MC6800**

**MC6802**

**MC6808**

**MC6802NS**

---

*Pin 35 must be tied to 5 V on the MC6802NS
*Pin 35 must be tied to ground for the MC6800**
### Vector Table

<table>
<thead>
<tr>
<th>Address</th>
<th>Interrupt</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFFE-FF</td>
<td>RESET</td>
</tr>
<tr>
<td>FFFC-FD</td>
<td>NMI</td>
</tr>
<tr>
<td>FFFA-FB</td>
<td>SWI</td>
</tr>
<tr>
<td>FFFB-F9</td>
<td>IRQ</td>
</tr>
</tbody>
</table>

### UniLab interface considerations

Analyzer cable B for 6800, 6802, 6809.

**Reserved areas:**
- FFFC-FD vector for hardware breakpoint.
- FFFA-FB vector for software breakpoint.
- FFB9-BA for interrupt service routine. (rel)
  - G082 only: 50-S3 in internal RAM. (rel)

**Overlay area:** FFBB-FFEF

**Breakpoint code:** 3F(SWI) inserted at breakpoint.

**Maximum oscillator speed:**
- Standard UniLab handles all family members.
- 10.0 MHz oscillator (2.5 MHz clock) with standard UniLab.
- High-speed not necessary.

**Note:** The 6802 and 6809 have a 4:1 ratio between oscillator speed and processor clock speed. With the 6800, the two clocks, C1 and C2, must be provided by outside circuitry. For all members of this family there is a 1:1 ratio between processor and UniLab clocks.

**UniLab clock:**
- RD- C2 (phase 2 clock)
- E (clock output on 6802, 6809)
- WR- VMA (valid memory address)
- K1 not used
- K2 not used

**Status lines:**
- C4 not used
- C5 not used
- C6 not used
- C7 R/W

**Cycle types:**
- fetch FF
- read FF
- write 7F

**Note:** Debug notes for 6802 incorrectly show C6 connected to the normally low BA output. BA goes high when the processor is in a WAIT state.
THE 6805 MICROPROCESSOR FAMILY

The Unilab support is designed for the 1468005SE, 1468005E3, 6305 and 5005 piggyback

HARDWARE FEATURES

1468005
112 bytes on board RAM
16 bidirectional I/O lines
Multiplexed address/data bus
Can address up 8K external memory

1468005E3

Same as above except 64K external memory addressing

68p05

Hitachi piggyback part for development

CPU registers
1 8 bit accumulator
1 8 bit index register X
1 12 bit Stack Pointer Register with bit 8 set to 1
1 12 bit Program Pointer
1 5 bit Condition Code Register

SOFTWARE CONSIDERATIONS

Since this package supports many versions of the 6805 there are a number of patch words to configure it.

PEACK  Sets software package for piggyback mode
PEACK'  Sets software for 1468005SE
MH6305  Sets software for 6305
G805E3  Sets software for 6805E3
=ZP  Sets ram location used in zero page by debug
Alley/alicy' shows does not show all bus cycles
TCR/Tcr' Enables/disables timer control register ($0029) at a breakpoint.
Vector Table

<table>
<thead>
<tr>
<th></th>
<th>146805E2</th>
<th>146905e3</th>
<th>68p05</th>
<th>6305</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reset</td>
<td>1FFE-F</td>
<td>FFPE-F</td>
<td>FFE-F</td>
<td>1FFE-F</td>
</tr>
<tr>
<td>SWI</td>
<td>1FFC-D</td>
<td>FFPC-D</td>
<td>FF-C</td>
<td>1FFC-D</td>
</tr>
</tbody>
</table>

Unilac Interface Considerations

The 60p05 requires a special clock circuit which is part of the M-Cable.

Analyzer Cable: B for 146805e1/e3 and h6305
  M with clock circuit for 60p0%  

Emulator Cable C624 or 20

Reserved Area: xFF89 - xFBA (x =1 for e2 and 6305, x = 0 for 60p05 and F for e3)

Overlay Area: xFBA-xFE3 (x same as above)

Maximum Clock Speed: MHz

Unilac Clock:

<table>
<thead>
<tr>
<th></th>
<th>146805e2/e3</th>
<th></th>
<th>68p05</th>
</tr>
</thead>
<tbody>
<tr>
<td>RD-</td>
<td>cs</td>
<td></td>
<td>To read on clock circuit</td>
</tr>
<tr>
<td>WR-</td>
<td>N/C</td>
<td></td>
<td>N/C</td>
</tr>
<tr>
<td>K1-</td>
<td>N/C</td>
<td></td>
<td>N/C</td>
</tr>
<tr>
<td>K2-</td>
<td>N/C</td>
<td></td>
<td>N/C</td>
</tr>
</tbody>
</table>

Unilac Status Lines:

<table>
<thead>
<tr>
<th>C4</th>
<th>N/C</th>
<th></th>
<th>N/C</th>
</tr>
</thead>
<tbody>
<tr>
<td>C5</td>
<td>N/C</td>
<td></td>
<td>N/C</td>
</tr>
<tr>
<td>C6</td>
<td>LI</td>
<td></td>
<td>N/C</td>
</tr>
<tr>
<td>C7</td>
<td>R/W-</td>
<td></td>
<td>N/C</td>
</tr>
</tbody>
</table>

NMI: NOT AVAILABLE ON THIS PROCESSOR.

Cycle Types:

- Fetch F0 to FF CONT
- Read B0 to BF CONT
- Write 30 to 3F CONT

No emulation module available as yet.
### Differences from 6809 plain

<table>
<thead>
<tr>
<th>PIN#</th>
<th>6809E</th>
<th>6809</th>
</tr>
</thead>
<tbody>
<tr>
<td>33</td>
<td>BUSY - for multiprocessing</td>
<td>DMA - Direct Memory Access</td>
</tr>
<tr>
<td>36</td>
<td>AVMA - processor will use bus on next cycle</td>
<td>MRDY - stretch E, Q to extend data access time</td>
</tr>
<tr>
<td>38</td>
<td>LIC - Last instruction cycle</td>
<td>EXTAL - for simple TTL or crystal clock hookup</td>
</tr>
<tr>
<td>39</td>
<td>TSC - three state control</td>
<td>XTAL - ground for simple clock</td>
</tr>
</tbody>
</table>

**Speed**

- 1 MHz part: MC6809E
- 1.5 Mhz: MC68A09E
- 2 Mhz: MC68B09E

We can handle all of the above with a standard speed UniLab. Bus cycle time is crystal speed divided by 4. We can easily handle a 8 Mhz crystal driving the 6809E with a standard speed UDL.

**Problems with supporting the 6809**

We do have a "plug-in-circuit" emulation board in the works for those who are not using either the MRDY or DMA lines in a 6809. We use a 6809E in the circuit, and have a clock circuit built in. The usual reason people choose the 6809 over the 6809E is that the clock oscillator is simpler. We support the 6809E because of the LIC and AVMA pins which help our disassembler identify what is going on. The number of instructions, and the number of extra cycles on the bus, makes the job of writing a disassembler that will stay in sync without looking at these pins almost impossible.

**Special Commands**

- SWI
- SWI2 To change the breakpoint opcode
- SWI3

**Reserved Vectors and Memory Areas**

- FFFA,B: SWI Vector (relocatable to FFF2,3 or FFF4,5)
- FFFC,D: NMI Vector
- FFB9-FFEF: Reserved area and overlay (relocatable)
**Breakpoint Opcode**
- 3F for SWI
- 10 3F for SWI2
- 11 3F for SWI3

**UniLab Interface**
- Analyzer Cable B.

**UniLab Clock**
- RD E clock input

**Status Lines**
- C4 AVMA (advanced VMA)
- C5 LIC (last instruction cycle)
- C6 R/W
- C7 BS (bus status)

**Cycle Types**
- READ 10 to 3F
- WRITE 40 to 7F
- INTERRUPT 80 to FF

Note that FETCH is the same as READ, since AVMA only tells us that the next cycle will be the first byte of an instruction.

**HARDWARE FEATURES**
- External Clock Inputs, E and O, Allow Synchronization
- ISC Input Controls Internal Bus Buffers
- LIC Indicating Operates Fetch
- AVMA Allows Efficient Use of Common Resources in a Multiprocessor System
- BUSY is a Status Line for Multitasking
- Fast Interrupts Request Input Stacks Only Condition Code Register and Program Counter
- Interrupt Acknowledge Output Allows Vectoring by Devices
- Sync Acknowledge Output Allows Synchronization to External Event
- Single Bus Cycle RESET
- Single 5-Volt Supply Operation
- NMI Inhibited After Reset Until After First Read of Stack Pointer
- Early Address Valid Allows Use With Slower Memories
- Early Write Data for Dynamic Memories

**HARDWARE FEATURES**
- On-Chip Oscillator (Crystal Frequency = 4 MHz)
- NMI Inputs Allows DMA Operation on Memory Refresh
- Fast Interrupt Request Input Stacks Only Condition Code Register and Program Counter
- NMI Input Extends Data Access Times for Use With Slow Memory
- Interrupt Acknowledge Output Allows Vectoring by Devices
- Sync Acknowledge Output Allows Synchronization to External Event
- Single Bus Cycle RESET
- Single 5-Volt Supply Operation
- NMI Inhibited After Reset Until After First Read of Stack Pointer
- Early Address Valid Allows Use With Slower Memories
- Early Write Data for Dynamic Memories
THE 68HC11 MICROCONTROLLER

HARDWARE FEATURES

- 8-bit Microcontroller
- Multilingexed address and data
- 8K on-board ROM
- 512 bytes on-board RAM
- 256 bytes of onboard standby RAM
  (Mapable to any 4k Boundary)
- 8-bit Pulse accumulator
- Serial Communications Interface
- Serial Peripheral Interface
- Eight channel, 8 bit A to D
- Real time interrupt
- Computer Operating Properly (COP) Watchdog timer

SOFTWARE FEATURES

- Enhanced 6800/01 instruction set
- 16x16 integer and fraction divide
- Bit manipulation
- WAIT mode
- STOP mode

CPU Registers
- 2 8 bit Accumulators A and B or one 16 bit double accumulator D
- 2 16 bit Index Registers X and Y
- 1 16 bit Stackpointer Register
- 1 16 bit Program Pointer
- 1 8 bit Condition Code Register
- 64 Special function Registers

SOFTWARE CONSIDERATIONS

In order to ensure proper operation of debug the COP should be disabled.

Special Commands

=SRPAGE Used to tell the disassembler where sixty-four special function registers are mapped. The default is 1 mapping them from $1000 to $103f.

SHOW-REGS Displays special function registers

SHOW-NAMES / HIDE-NAMES Show - do not show Special function register names with disassembler.}

/ SWIL) Allows illegal opcode trap to be used instead of SWI for breakpoint.
Table 3-1. Interrupt Vector Assignments

<table>
<thead>
<tr>
<th>Vector Address</th>
<th>Interrupt Source</th>
<th>CC Register Mask</th>
<th>Local Mask</th>
</tr>
</thead>
<tbody>
<tr>
<td>FF00, C1</td>
<td>Reserved</td>
<td>1 Bit</td>
<td>See Table 3-2</td>
</tr>
<tr>
<td>FF01, C2</td>
<td>Reserved</td>
<td>1 Bit</td>
<td></td>
</tr>
<tr>
<td>FF10, D9</td>
<td>Serial Transfer Complete</td>
<td>1 Bit</td>
<td>SPIE</td>
</tr>
<tr>
<td>FF11, D8</td>
<td>Pulse Accumulator Input Edge</td>
<td>1 Bit</td>
<td>PA11</td>
</tr>
<tr>
<td>FF12, D7</td>
<td>Pulse Accumulator Overflow</td>
<td>1 Bit</td>
<td>PAOVI</td>
</tr>
<tr>
<td>FF13, D6</td>
<td>Timer Overflow</td>
<td>1 Bit</td>
<td>TOI</td>
</tr>
<tr>
<td>FF20, E1</td>
<td>Timer Output Compare 5</td>
<td>1 Bit</td>
<td>OCI9</td>
</tr>
<tr>
<td>FF21, E2</td>
<td>Timer Output Compare 4</td>
<td>1 Bit</td>
<td>OC4I</td>
</tr>
<tr>
<td>FF22, E3</td>
<td>Timer Output Compare 3</td>
<td>1 Bit</td>
<td>OC3I</td>
</tr>
<tr>
<td>FF23, E4</td>
<td>Timer Output Compare 2</td>
<td>1 Bit</td>
<td>OC2I</td>
</tr>
<tr>
<td>FF30, E9</td>
<td>Timer Input Capture 3</td>
<td>1 Bit</td>
<td>OCI1</td>
</tr>
<tr>
<td>FF31, E8</td>
<td>Timer Input Capture 2</td>
<td>1 Bit</td>
<td>OC3I</td>
</tr>
<tr>
<td>FF32, E7</td>
<td>Timer Input Capture 1</td>
<td>1 Bit</td>
<td>OC2I</td>
</tr>
<tr>
<td>FF40, F1</td>
<td>Real Time Interrupt</td>
<td>1 Bit</td>
<td>RTI1</td>
</tr>
<tr>
<td>FF41, F2</td>
<td>IRQ (External Pin or Parallel I/O)</td>
<td>1 Bit</td>
<td>See Table 3-3</td>
</tr>
<tr>
<td>FF42, F3</td>
<td>XIRQ Pin (Pseudo Non-Maskable Interrupt)</td>
<td>1 Bit</td>
<td>None</td>
</tr>
<tr>
<td>FF43, F4</td>
<td>SWI, PIP</td>
<td>X Bit</td>
<td>None</td>
</tr>
<tr>
<td>FF50, F8</td>
<td>Illegal Opcode Trap</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>FFFA, D8</td>
<td>COP Failure (Reset)</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>FFFA, F8</td>
<td>COP Clock Monitor Fall (Reset)</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>FFFA, FF</td>
<td>RESET</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>

Unilab Interface Considerations

The 68HC11 must be configured to run in the expand multiplexed mode (mode 1) by having MODB and MODA pulled high in reset.

Analyzer Cable B
Emulator Cable if Data Lines are Buffered C8HD24/28

Reserved Area: $FF71-$FF73

Overlay Area: $FF74-$FFBF

Maximum Clock Speed: 16 MHZ

Unilab Clock:
- RD - 3-clock
- WR - N/C
- X1 - N/C
- K2 - N/C

Unilab Status Lines:
- C4 - N/C
- C5 - N/C
- C6 - MODA/LIR
- C7 - R/W

NMI: Connects to XIRQ-

Cycle Types:
- Fetch BF
- Read FF
- Write 7F

No Emulation Module available for this processor at this time.
ORION INSTRUMENTS TECHNICAL NOTES

This note applies to the Motorola 68HC11 chip.

I.  If you are using the Motorola EVB Board there are a few things which have to be done in order to run on this target board.

These items are as follows:

1.) The analyzer ground wire needs to be connected to pin 8 of U5.

2.) The ground wire associated with the Miscellaneous lines to pin 7 of U6.

3.) The emulator cable needs to be shielded.

4.) A 10k ohm resistor needs to be mounted for NMI to work. This resistor is connected between pin 26 and pin 18 of connector P1 on the underside of the board. The UNILAB NMI wire is then connected to pin 18 of P1.

5.) The UNILAB reset- wire is connected to R14 at the end closest to the power connector.

6.) The 68HC11 does not bring out to the bus the reads from internal memory therefore the data will look the same as the low order address unless you have external "shadow RAM" (i.e.) external ram mapped in the address space as internal RAM.

II. The last consideration with this processor is that if you use an internal stack and have the data lines buffered then you need to use a CB4HD24 or CB4HD28 Emulator cable so the analyzer data inputs can be connected on the processor side of buffers, otherwise the debug will not function consistently.

Jack Neithardt
Applications Engineer
The 68000 Family

<table>
<thead>
<tr>
<th></th>
<th>Maximum Address</th>
<th>Maximum Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>68000</td>
<td>4, 6, 8, 10, 12.5, ?</td>
<td>24-bit</td>
</tr>
<tr>
<td>68008</td>
<td>8, 10, ?</td>
<td>20-bit</td>
</tr>
<tr>
<td>68010</td>
<td></td>
<td>24-bit</td>
</tr>
<tr>
<td>68020</td>
<td></td>
<td>32-bit</td>
</tr>
</tbody>
</table>

Bus cycle time is clock speed divided by 8. The actual timing requirements for the UDL/UniLab are more related to the time it takes to access ROM data. We can easily handle a 10 Mhz 68000 with a standard speed UDL.

Problems with supporting the 68010, 68020 have to do with the internal instruction cache. The 68010 has the capability of 3 words, the 68020 handles 128 words. These caches can be disabled via hardware, so if the customer is willing to run in this mode during development, we can support him.

Comparison with other similar processors

68000 overlaps fetching of each opcode for two step "pipeline"
Z8000 only does prefetch under certain circumstances
8086 uses extensive prefetch of up to 6 bytes

8086 and Z800 have methods of simple and "expanded" (max) modes.
68000 is always in "max" mode

8000 can address directly up to 16M bytes, the 68020 up to 4000M bytes
8086 handles 64K directly, but uses up to 1M with segment registers
Z8001 handles up to 48M bytes using internal segment registers and external memory management.

68000 operates in Supervisor or User mode,
Z8000 operates in System or Normal mode.
8086 has no modes
80286 has Protected and Unprotected mode.

68000 has 17 32-bit registers, 8 designated as data, 9 as address,
and all registers can function as index registers.
8086 has 4 16-bit registers and 3 separate 16-bit index registers

68000 has separate data and address lines
Z8000 and 8086 have multiplexed data and address lines
(smaller packages)
Special Commands

FROMMV  All used to extend addresses beyond 16 bit entry
TOMV
TOGO

TRAP  Change trap vector from 0 to F

INTADR  Set lower memory boundary used in disassembler
         logic. Defaults to $0400. Use
         value 'INTADR'

We are always in the supervisor mode when DEBUG control is
established.

Reserved Vectors and Memory Areas

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>80-83</td>
<td>Trap Vector (relocatable)</td>
</tr>
<tr>
<td>7C-7F</td>
<td>NMI Vector</td>
</tr>
<tr>
<td>7AC-7FF</td>
<td>Reserved area and overlay (relocatable)</td>
</tr>
</tbody>
</table>

Breakpoint Opcode  4E 4n  where n is trap 

UniLab Interface  Analyzer Cable P, 16 bit rom cable for 68000,
8 bit cable for 68008. Chip comes in a variety of packaging, from
64-pin dip to 68 pin quad pack, leadless chip carrier, and pin
grid array. True NMI requires toggling 3 pins. May require extra
circuitry.

Maximum oscillator speed:

Standard handles all family members. The timing of the 68000
is quite complex. The simplest statement to make is that the
clock is 8 times the bus cycle time. The limiting factor has
more to do with memory access times. We have not experienced any
difficulty running at 12 Mhz with a standard unit.

UniLab Clock

K2  AS (address strobe)
K1  DTACK (normally grounded
    for simple systems)
RD  Not used, recommend strapping high
WR  Not used, recommend strapping high

Status Lines

C4  FC0
C5  FC1
C6  FC2
C7  R/W

Cycle Types

User write  1x
Supervisor write  5x
User read  9x
User fetch  Ax
Supervisor read  Dx
Supervisor fetch  Ex
Interrupt  Fx
The 8048 Microprocessor Family

Part | Program Memory | Data Memory
---|---|---
8748II | EPROM | Internal
8749II | EPROM | Internal
8048AH | 1K x8 ROM | 64 x8
8049AH | 2K x8 ROM | 128 x8
8050AH | 4K x8 ROM | 256 x8
803SAHL | none | none
803BAHL | none | none
8040AHL | none | none

The major features are:

8-Bit CPU
1K x8 ROM Program Memory
64 x8 RAM Data Memory
27 I/O Lines
8-Bit Timer/Event Counter

Oscillator speed: 1 to 11 MHz.
5:1 ratio between oscillator speed and processor clock speed.

Locations in data memory is accessed indirectly using R0 & R1

PIN CONFIGURATION

Figure 12-2. Program Memory Map

Figure 12-3. Data Memory Map
UniLab Interface considerations

Analyzer Cable E

UniLab clock: RD- PSN
WR- WR-
K1 RD-
K2 not used

Cycle types: read A0 to BF
write C0 to DF
fetch E0 to FF

Overlay area: 5 bytes at the top of 2K bank

Breakpoint code: CALL 7FA (F4FA)

Interrupts: does not have NMI.

<table>
<thead>
<tr>
<th>type</th>
<th>vec address</th>
</tr>
</thead>
<tbody>
<tr>
<td>reset</td>
<td>0H</td>
</tr>
<tr>
<td>external</td>
<td>3H</td>
</tr>
<tr>
<td>timer/ctr</td>
<td>7H</td>
</tr>
</tbody>
</table>

UniLab commands:

=OVERLAY
n RNAME name assigns name to Rn
ENI, DISI enable/disable interrupt
ENTCTRL, DISTCTRL enable/disable counter
INTRAM, EXTRAM use of internal/external data memory area
PIGGYBACK patch for NS87P50.
PIGGYBACK' for expanded chip.
THE 8051 MICROPROCESSOR FAMILY

8051:
expanded mode (external rom)
Sales Ranking -- #2 with 13% of total sales

8051P:
OKI piggyback 80C51UV (internal rom)
Sales Ranking -- #16 with less than 1%

The major features are:

- 8-bit CPU
- On-Chip oscillator and clock circuitry
- 32 I/O lines
- 64K address space for external data memory
- Two 16-bit timer/counters
  (Three on 8032/8051)
- A five-source interrupt structure
  (Six sources on 8032/8052) with two priority levels
- Full duplex serial port
- Boolean processor

<table>
<thead>
<tr>
<th>Part</th>
<th>Technology</th>
<th>On-Chip Program Memory</th>
<th>On-Chip Data Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>8051</td>
<td>NMOS</td>
<td>4K (ROM)</td>
<td>128</td>
</tr>
<tr>
<td>8031</td>
<td>NMOS</td>
<td>none</td>
<td>128</td>
</tr>
<tr>
<td>8751H</td>
<td>NMOS I</td>
<td>4K (EPROM)</td>
<td>128</td>
</tr>
<tr>
<td>80C51</td>
<td>CHMOS</td>
<td>4K (ROM)</td>
<td>128</td>
</tr>
<tr>
<td>80C31</td>
<td>CHMOS</td>
<td>none</td>
<td>128</td>
</tr>
<tr>
<td>8052</td>
<td>NMOS</td>
<td>8K (ROM)</td>
<td>256</td>
</tr>
<tr>
<td>8032</td>
<td>NMOS</td>
<td>none</td>
<td>256</td>
</tr>
</tbody>
</table>

Oscillator speed: 3.5 to 12 MHz.
6:1 ratio between oscillator speed and processor clock speed.
UniLab interface considerations:

Analyzer Cable E

UniLab clock: 8031 Expanded 8051 Piggyback
    RD- P5N            RO- not used
    WR- WR6            WR- not used
    K1 RD7
    K2 EA-

Overlay Area: FFC0H -- FFFBH for expanded
               FC0H -- FFF for piggyback

Breakpoint code: LCALL ovadr (12xxxx)

Cycle types: fetch 70 -- 7F
             read 30 -- 3F
             write 50 -- 5F

Interrupt: 2 levels of priority
           No NMI

Interrupt Source  Vector Address
IEO (external request 0)  3H
TFO (internal timer/ctr 0) BH priority
IE1 (external request 1)  13H within
TF1 (internal timer/ctr 1) 1BH level
RI+TI (internal serial port) 23H
TF2+EXF2

UniLab Commands:

7FH
    RAM
    (0032)

FFH
    sfr

--------------------
                   
INTRAM (Internal RAM)
--------------------
                   
EXTRAM (External RAM)
--------------------
                   
PMEM (Emulation MEM)
--------------------

R7? RI DR7
(DR7 DR1) DR1
cannot use M here

=OVERLAY
n RNAME name assigns name to Rn
General Information

The 8085 microprocessor (40 pin dip package) is an 8-bit CPU with addressing capability of 64K. The 8085 has 8 data/address multiplexed lines for lower address A0-A7 and data D0-D7. The upper 8-bit address lines are not multiplexed. It has four vectored interrupts (one is nonmaskable) and INTR line for a total of five interrupts. The 8085 also supports serial I/O.

UniLab Interface

The 8085 uses analyzer cable "A"

Speed

<table>
<thead>
<tr>
<th></th>
<th>Standard UDL</th>
<th>High Speed UDL</th>
</tr>
</thead>
<tbody>
<tr>
<td>8085AH</td>
<td>3MHz</td>
<td>YES</td>
</tr>
<tr>
<td>8085AH-2</td>
<td>5MHz</td>
<td>YES</td>
</tr>
<tr>
<td>.085AH-1</td>
<td>6MHz</td>
<td>YES</td>
</tr>
</tbody>
</table>

Reserved Vectors and Memory Areas

User can choose any of these RST vectors.

RST = BP
0   C7
8   CF
10  D7
20  DF
28  EF
30  F7
38  FF

You can change the overlay area by placing a 3 byte JP instruction at one of the above restart vectors and then entering XXXX, where XXXX is the address placed at the above restart vectors.

Breakpoint Opcode

CF is the default

Use xx = BP to change opcode
Unilab Clock

RD- WR- Read, Write pins 32,31

Status Lines

K1 to INTA pin 11
K2 to RESET OUT pin 3
C4 to S1 pin 33
C5 to S0 pin 29
C7 to I/OH pin 34

Cycle Types

FETCH 70 TO 7F
READ 50 TO 5F
WRITE 60 TO 6F
OUTPUT E0 TO EF
INPUT DO TO DF
INTERRUPT F0 TO FF
I/O DO TO EF

Special Commands

GO GO PATCH
=P
=OVERLAY
INP Used to read I/O port
OUT Used to write I/O port

NMI Not supported.
THE 8086 MICROPROCESSOR FAMILY

The 8086 Microprocessor

- 16-bit data bus
- 20-bit address bus

<table>
<thead>
<tr>
<th>Part</th>
<th>Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>8086</td>
<td>5 MHz</td>
</tr>
<tr>
<td>8086-2</td>
<td>8 MHz</td>
</tr>
<tr>
<td>8086-1</td>
<td>10 MHz</td>
</tr>
</tbody>
</table>

The 80186 Microprocessor

- Incorporates the power of "15-20 chips" in one package.
- Higher performance, more instructions.

<table>
<thead>
<tr>
<th>Part</th>
<th>Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>80186</td>
<td>8 MHz</td>
</tr>
<tr>
<td>80186-6</td>
<td>6 MHz</td>
</tr>
</tbody>
</table>

The 80286 Microprocessor

- 24-bit address bus, 16 megabytes physical memory.
- 1 Megabyte "virtual" memory.

<table>
<thead>
<tr>
<th>Part</th>
<th>Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>80286-8</td>
<td>8 MHz</td>
</tr>
<tr>
<td>80286-6</td>
<td>6 MHz</td>
</tr>
<tr>
<td>80286-4</td>
<td>4 MHz</td>
</tr>
</tbody>
</table>

Software considerations:

- The same package supports the 8086, the 80186, the 80286 and the AT. Support for the 80286 and the AT is not good.

Patch commands:

- 86MIN 168PATCH MAXMODE 811PATCH

Special commands:

- All the words dealing with segments. See the one page writeup on segmentation.

---

**MINIMUM AND MAXIMUM MODES**

The requirements for supporting minimum and maximum IAPX 86/10 systems are sufficiently different that they cannot be done efficiently with 40 uniquely defined pins. Consequently, the 8086 is equipped with a strap pin (MN/AX) which defines the system configuration. The definition of a certain subset of the pins changes dependent on the condition of the strap pin. When MN/AX pin is strapped to GND, the 8086 treats pins 24 through 31 in maximum mode. An 8288 bus controller interprets status information coded into S0, S1, S2 to generate bus timing and control signals compatible with the MULTIBUS® architecture. When the MN/AX pin is strapped to VCC, the 8086 generates bus control signals itself on pins 24 through 31, as shown in parentheses in Figure 2. Examples of minimum mode and maximum mode systems are shown in Figure 4.
Vector Table

<table>
<thead>
<tr>
<th>Address</th>
<th>Interrupt</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFFF0-F</td>
<td>RESET</td>
</tr>
<tr>
<td>00000-3FF</td>
<td>A 256 element vector table. Each vector is 4 bytes long; a two byte segment followed by a two byte address. We use interrupts 1, 2 and 3.</td>
</tr>
</tbody>
</table>

UniLab interface considerations

Analyzer cable A for 8086 Min mode and for 80186.

Analyzer cable L for 8086 Max mode.

Reserved areas:

- 00004-F vectors for single-step, for NMI and for software breakpoint.
- FFFB-02 for interrupt service routine. (relocatable)

Overlay area: FFFBB-FFEF

Breakpoint code: INT 3 inserted at breakpoint.

Maximum oscillator speed:
- Standard UniLab handles all family members.
- High-speed not necessary.

<table>
<thead>
<tr>
<th>UniLab clock</th>
<th>80186</th>
<th>8086MIN</th>
<th>8086MAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>RD-</td>
<td>RD</td>
<td>RO</td>
<td>11</td>
</tr>
<tr>
<td>WR-</td>
<td>WR</td>
<td>WR</td>
<td>9</td>
</tr>
<tr>
<td>K1-</td>
<td>DT/R</td>
<td>DTR</td>
<td>16</td>
</tr>
<tr>
<td>K2-</td>
<td>DEN</td>
<td>INA</td>
<td>4</td>
</tr>
</tbody>
</table>

In addition, we use the UniLab input ALE, to determine when the address is valid:

<table>
<thead>
<tr>
<th>Status lines</th>
<th>80186</th>
<th>8086MIN</th>
<th>8086MAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>C4</td>
<td>na</td>
<td>na</td>
<td>na</td>
</tr>
<tr>
<td>C5</td>
<td>S0</td>
<td>na</td>
<td>S0</td>
</tr>
<tr>
<td>C6</td>
<td>S1</td>
<td>na</td>
<td>S1</td>
</tr>
<tr>
<td>C7</td>
<td>S2</td>
<td>MIO</td>
<td>S2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cycle types</th>
<th>8086max/80186</th>
<th>8086MIN</th>
<th>8086AF</th>
<th>AT</th>
</tr>
</thead>
<tbody>
<tr>
<td>fetch</td>
<td>80-9F</td>
<td>BF</td>
<td>80-BF</td>
<td>60-7F</td>
</tr>
<tr>
<td>read</td>
<td>80-BF</td>
<td>80-BF</td>
<td>A0-AF</td>
<td>40-5F</td>
</tr>
<tr>
<td>write</td>
<td>C0-DF</td>
<td>C0-FF</td>
<td>C0-DF</td>
<td>C0-DF</td>
</tr>
<tr>
<td>output</td>
<td>40-5F</td>
<td>50-7F</td>
<td>50-5F</td>
<td>20-3F</td>
</tr>
<tr>
<td>input</td>
<td>20-3F</td>
<td>00-4F</td>
<td>30-3F</td>
<td>E0-FF</td>
</tr>
<tr>
<td>interrupt</td>
<td>00-1F</td>
<td>00-0F</td>
<td>00-FF</td>
<td>00-1F</td>
</tr>
</tbody>
</table>
THE 8088 MICROPROCESSOR FAMILY

The 8088 Microprocessor
8-bit data bus
20-bit address bus

Part | Speed
---|---
8088 | 5 MHz
8086-2 | 6 MHz

The 80188 Microprocessor
Incorporates the power of "15-20 chips" in one package. Higher performance, more instructions.

Part | Speed
---|---
80188 | 8 MHz
80188-6 | 6 MHz

Software considerations:

The same package supports the 8088, the 80188 and the PC. Support for the PC as target is not good.

Patch commands:
88MIN 188PATCH 88MX PCPATCH

Special commands:
All the words dealing with segments. See the one page writeup on segmentation.

Figure 2. IAPX 88/10 Pin Configuration

Minimum and Maximum Modes
The requirements for supporting minimum and maximum 8088 systems are sufficiently different that they cannot be done efficiently with 40 uniquely defined pins. Consequently, the 8088 is equipped with a strap pin (MIN/MAX) which defines the system configuration. The definition of a certain subset of the pins changes, dependent on the condition of the strap pin. When the MIN/MAX pin is strapped to GND, the 8088 defines pins 24 through 31 and 34 in maximum mode. When the MIN/MAX pin is strapped to VCC, the 8088 generates bus control signals itself on pins 24 through 31 and 34.
Vector Table

**Address** | **Interrupt**
--- | ---
FFFF0-F | RESET
00000-3FF | A 256 element vector table. Each vector is 4 bytes long: a two byte segment followed by a two byte address. We use Interrupts 1, 2 and 3.

UniLab interface considerations

Analyzer cable A for 8088 Min mode and for 80188.
Analyzer cable L for 8088 Max mode.

Reserved areas:
00000-3F vectors for single-step, for NMI and for software breakpoint.
FFFFB1-B2 for interrupt service routine. (relocatable)

Overlay area: FFFB0-FFFF

Breakpoint code: INT 3 inserted at breakpoint.

Maximum oscillator speed:
Standard UniLab handles all family members.
High-speed not necessary.

<table>
<thead>
<tr>
<th>UniLab clock:</th>
<th>80188</th>
<th>8088MIN</th>
<th>8088MAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>RO-</td>
<td>RD</td>
<td>RD</td>
<td>11 These lines attach to the bus controller.</td>
</tr>
<tr>
<td>WR-</td>
<td>WR</td>
<td>WR</td>
<td>9</td>
</tr>
<tr>
<td>K1-</td>
<td>DTR</td>
<td>DTR</td>
<td>16</td>
</tr>
<tr>
<td>K2-</td>
<td>INA</td>
<td>INA</td>
<td>4</td>
</tr>
</tbody>
</table>

| UniLab input ALE | ALE | ALE | 5 |

<table>
<thead>
<tr>
<th>Status lines:</th>
<th>80188</th>
<th>8088MIN</th>
<th>8088MAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>C4</td>
<td>na</td>
<td>na</td>
<td>na</td>
</tr>
<tr>
<td>C5</td>
<td>S0</td>
<td>(BHE)</td>
<td>S0</td>
</tr>
<tr>
<td>C6</td>
<td>S1</td>
<td>na</td>
<td>S1</td>
</tr>
<tr>
<td>C7</td>
<td>S2</td>
<td>MIO</td>
<td>S2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cycle types:</th>
<th>80Max/80188</th>
<th>8088MIN</th>
<th>PC</th>
</tr>
</thead>
<tbody>
<tr>
<td>fetch</td>
<td>80-9F</td>
<td>1F</td>
<td>40-5F</td>
</tr>
<tr>
<td>read</td>
<td>A0-6F</td>
<td>20-3F</td>
<td>60-6F</td>
</tr>
<tr>
<td>write</td>
<td>C0-DF</td>
<td>40-5F</td>
<td>C0-CF</td>
</tr>
<tr>
<td>output</td>
<td>40-5F</td>
<td>C0-DF</td>
<td>00-0F</td>
</tr>
<tr>
<td>input</td>
<td>20-3F</td>
<td>A0-BF</td>
<td>E0-EF</td>
</tr>
<tr>
<td>interrupt</td>
<td>00-1F</td>
<td>80-9F</td>
<td></td>
</tr>
</tbody>
</table>
General Information

The 8096 can be separated into several sections for the purpose of describing its operation. There is a CPU, a programmable High Speed I/O Unit, an analog to digital converter, a serial port, and a Pulse With Modulated (PWM) output for digital to analog conversion. In addition to these functional units, there are some sections which support overall operation of the chip such as the clock generator.

DDB-96: Sales ranking 21 with 1%

Oscillator frequency: 6 MHz and 12 MHz

Unilab interface: Analyzer cable R

Unilab clock: WR, WR', RD, RD', ALE, ALE'

Overlay Area: 2016H -- 201CH

Soft. Vector: 2010H

---

![Block Diagram](image)

*Figure 2-1. Block Diagram (For simplicity, lines connecting port registers to port buffers are not shown.)*
Cycle Types: fetch F0 -- FF
         read E0 -- EF
         write C0 -- CF

Debugger special reg. : 00 -- 2000 as emulated ROM (for NMI)
Register 50H

To change overlay area : =OVERLAY

To change register used by Debugger : n=DBG

Disassembler special features:
If no external RAM : n INTERNAL will signal
disassembler not to expect any memory
cycles, (EXTERNAL)
ALL-INTERNAL
ALL-EXTERNAL

INTERRUPT STRUCTURE:

<table>
<thead>
<tr>
<th>SOURCE</th>
<th>VECTOR LOCATION</th>
<th>PRIORITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software</td>
<td>2010H</td>
<td>N/A</td>
</tr>
<tr>
<td>Extinct</td>
<td>200EH</td>
<td>7 (highest)</td>
</tr>
<tr>
<td>Serial Port</td>
<td>2000H</td>
<td>6</td>
</tr>
<tr>
<td>Software Tim.</td>
<td>200AH</td>
<td>5</td>
</tr>
<tr>
<td>HSI.0</td>
<td>2008H</td>
<td>4</td>
</tr>
<tr>
<td>High Speed Outputs</td>
<td>2006H</td>
<td>3</td>
</tr>
<tr>
<td>HSI Data Available</td>
<td>2004H</td>
<td>2</td>
</tr>
<tr>
<td>A/D Conversion Complete</td>
<td>2002H</td>
<td>1</td>
</tr>
<tr>
<td>Timer Overflow</td>
<td>2000H</td>
<td>0 (lowest)</td>
</tr>
</tbody>
</table>

SPECIAL COMMANDS

=AX  (sets working reg.)  IS-AX
=BX  IS-BX
=BX  IS-CX
=DX  IS-DX
**General Information**

The Z80 Microprocessor (40 pin dip package) was introduced by Zilog. The Z80 is an 8-bit processor with 16 address lines, 8 data lines and can address 64K of memory. The Z80 supports I/O mapped peripherals with special control lines. You can have up to 512 individual ports using this scheme. The Z80 can also have its I/O functions memory mapped. When the Z80 is operated at 6 MHz or 8 MHz, the memory access timing is critical and can cause the designer fits as high speed memories are needed. The bus cycle time necessary for the faster chips seems to have been stolen out of the read/write times.

The Z80 supports three (3) modes of interrupts; Mode 0, Mode 1, and Mode 2. The UDL handles one byte interrupts very well. At the present time we have problems with three (3) byte interrupts.

The Z80 supports NMI.

**UniLab Interface**

The Z80 microprocessor uses analyzer cable "E".

**Speed**

<table>
<thead>
<tr>
<th></th>
<th>Standard UDL</th>
<th>High Speed UDL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z80 = 2 MHz</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>Z80A = 4 MHz</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>Z80B = 6 MHz</td>
<td>NO</td>
<td>YES/MAYBE</td>
</tr>
<tr>
<td>Z80H = 8 MHz</td>
<td>NO</td>
<td>NO</td>
</tr>
</tbody>
</table>

**Reserved Vectors and Memory Areas**

User can choose any of these RST vectors.

- RST = BP
- 0 = C7
- 8 = CF
- 10 = D7
- 20 = DF
- 28 = EF
- 30 = F7
- 38 = FF

You can change the overlay area by placing a 3 byte JP instruction at one of the above restart vectors and then entering XXXX, where XXXX is the address placed at the above restart vectors.
**Breakpoint Opcode**

FF is default opcode

Use xx=BP to change opcode

**UniLab Clock**

RD- WR- Read, Write pins 21,22

**Status Lines**

K1 to *M1 pin 27
K2 to *I0RQ pin 20
A19 to *HEMRQ pin 19

**Cycle Types**

FETCH A0 TO BF
READ E0 TO FF
WRITE C0 TO DF
OUTPUT 40 TO 5F
INPUT 60 TO 7F
I/O 40 TO 7F
INTERUPT 20 TO 3F

**Special Commands**

=BP
=OVERLAY
INP Used to read I/O port
OUT Used to write I/O port
TNB Shows lines leading up to current breakpoint
HD64180
(CHOS)

General Information

The HD64180 is a 64 pin shrink dip package by Hitachi. It is instruction compatible with the Z80 and has many added instructions. The 64180 is an 8-bit microprocessor with 18 address lines and can access 512K bytes of memory. The processor has many added features such as DMA, MMU, two serial ports and two 16-bit timers. The HD64180 supports the Z80 Mode 0, Mode 1, and Mode 2. It has many other interrupts, i.e. from special functions on board plus two additional interrupt lines. It also supports NMI.

UniLab Interface

The HD64180 uses analyzer cable "E"

User enters HD64180 patch word.

Speed


<table>
<thead>
<tr>
<th></th>
<th>Standard UDL</th>
<th>High Speed UDL</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 MHz</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>4 MHz</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>6 MHz</td>
<td>NO</td>
<td>NO</td>
</tr>
</tbody>
</table>

Reserved Vectors and Memory Areas

User can choose any of these RST vectors.

RST = BP
0  C7
8  CF
10 D7
20 DF
28 EF
30 F7
38 FF

You can change the overlay area by placing a 3 byte JP instruction at one of the above restart vectors and then entering XXXX, where XXXX is the address placed at the above restart vectors.
Breakpoint Opcode

FF is default opcode
Use xx=BP to change opcode

UniLab Clock
RD- WR- Read, Write pins 63, 62

Status Lines
K1 to LIR
K2 to IOE
A19 to ME

Cycle Types
FETCH AO TO BF
READ BO TO PF
WRITE CO TO DF
OUTPUT 40 TO 5F
INPUT 60 TO 7F
I/O 40 TO 7F
INTERUPT 20 TO 3F

Special Commands

HD64180 patch word

=BP
=OVERLAY
INF Used to read I/O port
OUT Used to write I/O port
TNB Shows lines leading up to current breakpoint
General Information

The NSC-800 CHOS microprocessor (40 pin dip package) is National Semiconductor's answer to the Z80. The NSC-800 is fully compatible with the Z80 instruction set. It has 8 data lines multiplexed with the lower eight address lines. The addressing range for the NSC-800 is 64K memory (16 address lines total). The NSC-800 has the same I/O addressing capability as the Z80; both memory mapped and I/O mapped. The NSC-800 differs in the number of interrupt inputs it has. The Z80 has NMI and INT. The NSC-800 has 4 in addition to NMI. The interrupts supported by the NSC-800 include those of the Z80; Mode 0, Mode 1, Mode 2 and in addition include INTRA, INTRB, INTRC. These correspond to RSTA (3C), RSTB (34) and RSTC (2C).

UniLab Interface

The NSC-800 uses analyzer cable "Q".

User types NSC-800 patch word.

Speed

<table>
<thead>
<tr>
<th>Standard UDL</th>
<th>High Speed UDL</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 MHz</td>
<td>YES</td>
</tr>
<tr>
<td>4 MHz</td>
<td>YES</td>
</tr>
<tr>
<td>6 MHz</td>
<td>NO</td>
</tr>
<tr>
<td>8 MHz</td>
<td>NO</td>
</tr>
</tbody>
</table>

Reserved Vectors and Memory Areas

User can choose any of these RST vectors.

RST = BP
0 = C7
8 = CF
10 = D7
20 = DF
28 = EF
30 = F7
38 = FF

You can change the overlay area by placing a 3 byte JP instruction at one of the above restart vectors and then entering XXXX, where XXXX is the address placed at the above restart vectors.
Breakpoint Opcode
FF is default opcode
Use xx=EP to change opcode

UniLab Clock
RD- WR- Read, Write pins 32,31

Status Lines
K1 to INA pin 26
K2 to RSO pin 37
C4 to S1 pin 27
A19 to IOM pin 34

Cycle Types
FETCH 70 TO 7F
READ 50 TO 5F
WRITE 60 TO 6F
OUTPUT ED TO EF
INPUT D0 TO DF
I/O D0 TO EF

Special Commands
NSC-800 Patch Word
=BP
=OVERLAY
INP Used to read I/O port
OUT Used to write I/O port
TNB Shows lines leading up to current breakpoint
The Super8 family consists of basic microcomputers, protopack emulators, and ROMless microcomputers. The various family members differ in the amount of on-chip ROM and the physical packaging. Super8 is a 48-pin device.

Major features:

- Full-duplex UART.
- On-chip baud-rate generator.
- Two 16-bit programmable counter/timers.
- DMA controller.
- Two register pointers that allow use of 'fast' instructions to access register groups within 600ns.
- Additional instructions that support threaded-code languages, such as FORTH.

Hardware considerations:

Input clock speed: 1-12 MHz

8:1 Input clock to bus cycle ratio

(Std. UniLab is fine)

Reset address is 0020H.

Port 1 used for AD0-AD7, Port 0 used for A0-A15.

Stack can be internal or external. Configured in Port Mode register (F1H): P35 (pin 39) active if external stack.

UniLab interface considerations:

S8 uses Analyzer Cable D.

Reserved areas: 701 to 705 reserved (relocatable).

3 consecutive registers (default is register 60-63).

Overlay area: Immediately above reserved area (default is 786).
UniLab Commands:

\[ n = \text{PTR} \] Changes reserved registers, where 'n' indicates the lowest numbered register in hex. Default is 60H. NOTE: Program must include instructions to load register pair 60H with reserved area value. Breakpoints will not work properly unless they are placed after these initialization instructions.

\[ n = \text{OVERLAY} \] Changes location of overlay area. Can only be placed at an even address. Reserved area is 5 bytes below this.

\[ n = \text{WREG} \] Set starting address for disassembler.

\[ \text{EXTRAM} \] Set debug operations to work in external data memory.

\[ \text{EXTROM} \] Set debug operations to work in program memory.

\[ \text{INTSTACK} \] Set internal stack mode.

\[ \text{n DR!} \] Set the data register (C0-FF) to indicated value.

\[ \text{n DR?} \] Display current value of indicated data register.

\[ \text{n R!} \] Set register to indicated value.

\[ \text{n R?} \] Display current value of indicated data register.

---

Figure 3-7. Program and Data Memory Address Spaces

Figure 12-5. External Memory Timing

Figure 12-2. Configuring Port 1 for External Memory

Figure 12-6. Data Memory

Figure 12-3. Configuring Port 0 for External Memory
THE Z8 MICROPROCESSOR FAMILY

<table>
<thead>
<tr>
<th>Part</th>
<th>ROM Capacity</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z8601</td>
<td>2K</td>
<td>Masked ROM part, used in high volume production.</td>
</tr>
<tr>
<td>Z8603</td>
<td>0</td>
<td>Piggyback part used for development purposes.</td>
</tr>
<tr>
<td>Z8611</td>
<td>4K</td>
<td>Masked ROM part, used in high volume production.</td>
</tr>
<tr>
<td>Z8612</td>
<td>0</td>
<td>64-pin ROMless part, used in development systems.</td>
</tr>
<tr>
<td>Z8613</td>
<td>0</td>
<td>Piggyback part, used for development purposes.</td>
</tr>
<tr>
<td>Z8681/82</td>
<td>0</td>
<td>LOW COST ROMless production part with reduced I/O.</td>
</tr>
</tbody>
</table>

Major features:

- Two counter/timers
- Six vectored interrupts
- UART for serial I/O
- Power-down option

Hardware considerations:

Input clock speed: 12 MHz maximum.
8:1 Input clock to bus cycle ratio
(Std. UniLab is fine)

Reset address is determined by user as XX0C (see application notes).

Stack can be internal or external. Configured in Port 0-1 and Port 3 register; C5 connected to P34 (pin 23) if external stack.

UniLab interface considerations:

Z8681 uses Analyzer Cable D
Piggyback part uses Analyzer Cable E

Reserved areas: 7AF to 7B3 reserved (relocatable).
One register pair (default is register pair 60).

Overlay area: Immediately above reserved area (default is 7B4).

Cycle Types:

<table>
<thead>
<tr>
<th>Z8681 fetch</th>
<th>Z8681 int. stack</th>
<th>Z8681 ext. stack</th>
<th>Z8 piggyback</th>
</tr>
</thead>
<tbody>
<tr>
<td>read</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
</tr>
<tr>
<td>write</td>
<td>7F</td>
<td>DF</td>
<td>FF</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SF</td>
<td>FF</td>
</tr>
</tbody>
</table>
Features:
System/Normal operating mode
System/Normal mode stack
110 instruction types
414 total instructions

Additional Features:
Extended Processing Unit
Memory Management Unit

Maximum Clock Frequency: 7.59 MHz (standard)
10.1 MHz (high speed)

Table 1.1. 28000 CPU, Summary of Differences

<table>
<thead>
<tr>
<th></th>
<th>28000</th>
<th>28002</th>
<th>28003</th>
<th>28004</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addressing Mode</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>On Separated</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Address Bits</td>
<td>23</td>
<td>16</td>
<td>23</td>
<td>16</td>
</tr>
<tr>
<td>Virtual Memory</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Segmentation</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Error Protection</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Package Size (pins)</td>
<td>48</td>
<td>40</td>
<td>45</td>
<td>40</td>
</tr>
</tbody>
</table>

Table 2.1. Status Line Codes

<table>
<thead>
<tr>
<th>Bit-String</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>Internal Operation</td>
</tr>
<tr>
<td>0001</td>
<td>Memory Refresh</td>
</tr>
<tr>
<td>0010</td>
<td>I/O Reference</td>
</tr>
<tr>
<td>0011</td>
<td>Special I/O Reference</td>
</tr>
<tr>
<td>0100</td>
<td>Segment Trim Acknowledge</td>
</tr>
<tr>
<td>0101</td>
<td>Nonmaskable Interrupt Acknowledge</td>
</tr>
<tr>
<td>0110</td>
<td>Maskable Interrupt Acknowledge</td>
</tr>
<tr>
<td>0111</td>
<td>Vectored Interrupt Acknowledge</td>
</tr>
<tr>
<td>1000</td>
<td>Data Memory Request</td>
</tr>
<tr>
<td>1001</td>
<td>Stack Memory Request</td>
</tr>
<tr>
<td>1010</td>
<td>Data Memory Request (CPU)</td>
</tr>
<tr>
<td>1011</td>
<td>Stack Memory Request (CPU)</td>
</tr>
<tr>
<td>1100</td>
<td>Instruction Space Access</td>
</tr>
<tr>
<td>1101</td>
<td>Instruction Fetch, First Word</td>
</tr>
<tr>
<td>1110</td>
<td>Transfer between CPUS and CPU</td>
</tr>
<tr>
<td>1111</td>
<td>Test and Set Data Access (28000 and 28004 only)</td>
</tr>
</tbody>
</table>

Figure 5.1. Expanded Memory Address Within Instruction
NOTE: Shaded area is reserved.

Figure 5.2. General Purpose Registers

Figure 5.3. Memory Instructions

Figure 5.4. Memory Instructions

Figure 5.5. Memory Instructions

Figure 5.6. Memory Instructions

Figure 5.7. Memory Instructions
Unilab Interface

Analyzer Cable C

Unilab Clock:  
- RD = 05
- WR = not used
- K1 = not used
- K2 = not used

Overlay Area:  
700 -- 703
Placed at an even address
This address must be put in the interrupt vector by the user.

Break Point Opcode:  
EXTEND (F00)
SC xx (7Fx) if co-processor is being used

Cycle Types:  
- read = C0 to CF
- write = 40 to 4F
- fetch = 60 to FF
- input = 80 to 8F
- output = 00 to 0F
- spec in = 90 to 9F
- spec out = 10 to 1F
- stack read = 00 to DF
- stack write = 50 to 5F

Interrupts:

<table>
<thead>
<tr>
<th>TYPE</th>
<th>FCW addr</th>
<th>PC addr</th>
</tr>
</thead>
<tbody>
<tr>
<td>reset</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>extend</td>
<td>PSA + 4</td>
<td>PSA + 6</td>
</tr>
<tr>
<td>sc</td>
<td>PSA + C</td>
<td>PSA + E</td>
</tr>
<tr>
<td>nmi</td>
<td>PSA + 14</td>
<td>PSA + 16</td>
</tr>
</tbody>
</table>

Unilab Commands:

=RAM
=RAM.SEGMENT
=ROM.SEGMENT
SEGMENTED
UNSEGMENTED
=OVERLAY

set address of RAM
set segment where RAM resides
set segment where ROM resides
turn segmented mode on
turn segmented mode off
Features:
- 9-bit data bus
- 16-bit address bus
- CMOS
- 8K x 9 Internal ROM
- 256 x 9 Internal RAM
- 95 instruction type
- 451 total instructions

78312 Family:
- 78310 ROM-less version
- 78319 has different pinout
  has extra control signals for:
  - data fetch
  - internal RAM read
  - internal RAM write

Clock Speed: 12 MHz (standard)

Figure 1.1 Pin Configurations: SHRINK DIP and DIP Packages

Figure 2.3 General-Purpose Register Designation and Storage
UniLab Interface

Analyzer Cable A

UniLab Clock:  
- RD- internal read, external read & instruction fetch
- WR- internal write & external write
- K1 not used
- K2 not used

Cycle Types:  
- read F0 to FF
- write D0 to DF
- fetch B0 to BF

Overlay Area:  
- B0 -- B2
  
even address = OVERLAY
  
BPTRIGAD must be at odd address &
  
program uses one byte before BPTRIGAD

Break Point Code:  
- BRK (SE)

### Table 4.1: Interrupt sources and Vector Addresses

<table>
<thead>
<tr>
<th>DEFAULT</th>
<th>INTERRUPT SOURCE</th>
<th>MICRO</th>
<th>VECTOR ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESET</td>
<td>EXTERNAL ON-CHIP INTERRUPT</td>
<td>H</td>
<td>0D0H</td>
</tr>
<tr>
<td>NON-Maskable Interrupts</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 CPU@0</td>
<td>EXTERNAL CPU@0 INTERRUPT</td>
<td>H</td>
<td>0D0H</td>
</tr>
<tr>
<td>1 CPU@1</td>
<td>EXTERNAL CPU@1 INTERRUPT</td>
<td>Y</td>
<td>0D1H</td>
</tr>
<tr>
<td>2 CPU@2</td>
<td>EXTERNAL CPU@2 INTERRUPT</td>
<td>Y</td>
<td>0D2H</td>
</tr>
<tr>
<td>3 CPU@3</td>
<td>EXTERNAL CPU@3 INTERRUPT</td>
<td>Y</td>
<td>0D3H</td>
</tr>
<tr>
<td>4 CPU@4</td>
<td>EXTERNAL CPU@4 INTERRUPT</td>
<td>Y</td>
<td>0D4H</td>
</tr>
<tr>
<td>5 CPU@5</td>
<td>EXTERNAL CPU@5 INTERRUPT</td>
<td>Y</td>
<td>0D5H</td>
</tr>
<tr>
<td>6 CPU@6</td>
<td>EXTERNAL CPU@6 INTERRUPT</td>
<td>Y</td>
<td>0D6H</td>
</tr>
<tr>
<td>7 CPU@7</td>
<td>EXTERNAL CPU@7 INTERRUPT</td>
<td>Y</td>
<td>0D7H</td>
</tr>
<tr>
<td>8 CPU@8</td>
<td>EXTERNAL CPU@8 INTERRUPT</td>
<td>Y</td>
<td>0D8H</td>
</tr>
<tr>
<td>9 CPU@9</td>
<td>EXTERNAL CPU@9 INTERRUPT</td>
<td>Y</td>
<td>0D9H</td>
</tr>
<tr>
<td>10 CPU@10</td>
<td>EXTERNAL CPU@10 INTERRUPT</td>
<td>Y</td>
<td>0DAH</td>
</tr>
<tr>
<td>11 CPU@11</td>
<td>EXTERNAL CPU@11 INTERRUPT</td>
<td>Y</td>
<td>0DBH</td>
</tr>
<tr>
<td>12 CPU@12</td>
<td>EXTERNAL CPU@12 INTERRUPT</td>
<td>Y</td>
<td>0DCH</td>
</tr>
<tr>
<td>13 CPU@13</td>
<td>EXTERNAL CPU@13 INTERRUPT</td>
<td>Y</td>
<td>0DDH</td>
</tr>
<tr>
<td>14 CPU@14</td>
<td>EXTERNAL CPU@14 INTERRUPT</td>
<td>Y</td>
<td>0DEH</td>
</tr>
<tr>
<td>15 CPU@15</td>
<td>EXTERNAL CPU@15 INTERRUPT</td>
<td>Y</td>
<td>0DFH</td>
</tr>
</tbody>
</table>

Address 0 through 7F will not be disassembled from trace.

- 0 -- 3F interrupt vector
- 40 -- 7F call table area

UniLab Commands:

- SION-NAMES/HIDE-NAMES show/hide sfr names
- ALIGN/ALING align cycles
- addr SFR-SHOW display name for sfr at addr
- OVERLAY
Introduction

Terminology

Cross-compilers and cross-assemblers take a source code file and produce a relocatable object module. The object modules are then joined together by a linker, which produces executable code, a symbol table and (usually) a load map.

Often people do not draw a distinction between the linker and the assembler/compiler.

The executable code is the machine language instructions understood by the microprocessor.

The symbol table tells what values are held by the labels and variables used in the source code. This information is needed to understand what source code generated what executable code.

The load map shows where each relocatable object module was placed. The load map produced by Microsoft compilers also shows the absolute address of the executable code produced by each line of source code.

Possible confusion in terminology

The executable code is also referred to as the object code or the machine code. This is not the same as the object module.

In the PC world, two types of files run under DOS: .COM files, and .EXE files. These files are both "executable code" for the PC.

Our customers

Unilab customers use cross-compilers and cross-assemblers to produce the programs that control their microprocessor-based target systems. Orion does not make assemblers or compilers, so our customers purchase them from any of a large number of vendors.

Once a customer has produced executable code, he/she needs to transfer information to the Unilab system. The information produced by a linker can come in a wide variety of formats. The Unilab system can accept some of those formats, but not all.
Symbol file formats

Why?

Once a customer has loaded in a symbol file, they can use the symbolic names in place of hexadecimal numbers. In addition, the symbols will be substituted for addresses and values in the trace, breakpoint and disassembly displays.

Supported

File formats and compatibility

Executable file formats

Why?

Our customers need to load their program into emulation memory if they want to perform DEBUG operations. The trigger and trace capabilities do work on a program running from a ROM chip, but in general it is easier to put a program into emulation PROM than it is to put it into an EPROM.

Supported

The UniLab system can load in two different types of object code files: binary and Intel extended hex.

The simplest format for an executable file is binary. The file contains nothing but a sequence of hexadecimal values. The customer has to know where to start loading the file into memory. Load binary files with <start addr> <end addr> BINLOAD <file>.

An Intel hex file contains a series of records which specify the location to which each byte of object code must be loaded. With this format you can easily place information here and there throughout memory. Load Intel hex files with HEXLOAD <file>.

Other

There are two other schemes for encoding information that are similar to the Intel hex format:

- Tektronix hex format,
- Motorola S records.

In addition, there is a "mixed format" Intel file, which has symbolic information and object code mixed together. We will support that format in the near future.
Symbol example

The trace printout below shows a disassembled trace with symbol translation.

First eight symbol names were entered by hand:

1900 IS Init.Stack
3 IS Start_LOOP
29 IS End LOOP
10 IS First.IncA
3456 IS Init.BC
789A IS INIT.DE
BCDE IS INIT.HL
28 IS LAST.INCA

And then F9 was pressed, to get a trace of the startup:

<table>
<thead>
<tr>
<th>cy#</th>
<th>ADR</th>
<th>DATA</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0000</td>
<td>310019</td>
<td>LD SP,INIT_STACK</td>
</tr>
<tr>
<td>3</td>
<td>0003</td>
<td>3E12</td>
<td>LD A,12</td>
</tr>
<tr>
<td>5</td>
<td>0005</td>
<td>015634</td>
<td>LD BC,INIT.BC</td>
</tr>
<tr>
<td>8</td>
<td>0008</td>
<td>119A78</td>
<td>LD DE,INIT.DE</td>
</tr>
<tr>
<td>B</td>
<td>000B</td>
<td>21DEBC</td>
<td>LD HL,INIT.HL</td>
</tr>
<tr>
<td>E</td>
<td>000E</td>
<td>C5</td>
<td>PUSH BC</td>
</tr>
<tr>
<td>F</td>
<td>18FF</td>
<td>34</td>
<td>write</td>
</tr>
<tr>
<td>10</td>
<td>18FE</td>
<td>56</td>
<td>write</td>
</tr>
<tr>
<td>11</td>
<td>000F</td>
<td>C1</td>
<td>POP BC</td>
</tr>
<tr>
<td>12</td>
<td>18FE</td>
<td>56</td>
<td>read</td>
</tr>
<tr>
<td>13</td>
<td>18FF</td>
<td>34</td>
<td>read</td>
</tr>
<tr>
<td>14</td>
<td>0010</td>
<td>3C</td>
<td>INC A</td>
</tr>
<tr>
<td>15</td>
<td>0011</td>
<td>3C</td>
<td>INC A</td>
</tr>
<tr>
<td>2A</td>
<td>0026</td>
<td>3C</td>
<td>INC A</td>
</tr>
<tr>
<td>2B</td>
<td>0027</td>
<td>3C</td>
<td>INC A</td>
</tr>
<tr>
<td>2C</td>
<td>0028</td>
<td>3C</td>
<td>INC A</td>
</tr>
<tr>
<td>2D</td>
<td>0029</td>
<td>C30300</td>
<td>JP START_LOOP</td>
</tr>
<tr>
<td>30</td>
<td>0003</td>
<td>3E12</td>
<td>LD A,12</td>
</tr>
<tr>
<td>32</td>
<td>0005</td>
<td>015634</td>
<td>LD BC,INIT.BC</td>
</tr>
<tr>
<td>35</td>
<td>0008</td>
<td>119A78</td>
<td>LD DE,INIT.DE</td>
</tr>
<tr>
<td>38</td>
<td>000B</td>
<td>21DEBC</td>
<td>LD HL,INIT.HL</td>
</tr>
<tr>
<td>3B</td>
<td>000E</td>
<td>C5</td>
<td>PUSH BC</td>
</tr>
</tbody>
</table>

March 25, 1987
-- Interpret the Trace --

After these symbols have been loaded in, you can set a
trigger or a breakpoint using the symbolic name:

LAST.INCA AS

The last example, below, shows breakpoint displays with
these same symbols defined:

RESET

END_LOOP RB resetting

AF=2B28 (sz-a-pnc) BC=3456 DE=789A HL=BCDE IX=FFFF IY=FDFF SP=1900
END_LOOP 0029 C30300 JP START_LOOP (next step) ok

SSTEP NMI

AF=2B28 (sz-a-pnc) BC=3456 DE=789A HL=BCDE IX=FFFF IY=FDFF SP=1900
START_LOOP 0003 3E12 LD A, 12 (next step) ok

N

AF=1228 (sz-a-pnc) BC=3456 DE=789A HL=BCDE IX=FFFF IY=FDFF SP=1900
0005 015634 LD BC,INIT.BC (next step) ok

-- In Detail --

6-26
** EPROMS **

<table>
<thead>
<tr>
<th>EPROM</th>
<th>VOLTAGE</th>
<th>PERSONALITY MODULE</th>
<th>READ/PROGRAM EPROM COMMAND</th>
<th>EPROM SIZE DEC.</th>
<th>HEX</th>
</tr>
</thead>
<tbody>
<tr>
<td>2716 (1,a)</td>
<td>25</td>
<td>PM16</td>
<td>RPROM / P2716 OR PD2716</td>
<td>2K</td>
<td>7FF</td>
</tr>
<tr>
<td>2716B</td>
<td>12</td>
<td>PM16B</td>
<td>RPROM / P2716 OR PD2716</td>
<td></td>
<td></td>
</tr>
<tr>
<td>27C16 (n)</td>
<td>25</td>
<td>PM16</td>
<td>RPROM / P2716 OR PD2716</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| 2732 (h)   | 21      | PM32                | R2732 / P2732A              | 4K              | FFF  |
| 2732A (1,t) | 21      | PM32                | R2732 / P2732A              |                 |      |
| 2732B   | 12      | PM32B               | R2732 / P2732A              |                 |      |
| 27C32   | 21      | PM32                | R2732 / P2732A              |                 |      |
| 27C32 (n) | 25      | PM32                | R2732 / P27C32 (break pin 8 of PM module) |       |      |

| 2764A (1) | 12.5    | PM56                | RPROM / P2764 OR PD2764     | 8K              | 1FFF |
| 2764 (1,h) | 21      | PM56                | RPROM / P2764 OR PD2764     |                 |      |
| 2764C4 (1) | 12.5    | PM56                | RPROM / P2764 OR PD2764     |                 |      |
| 2764C6 (h,f) | 21    | PM56                | RPROM / P2764 OR PD2764     |                 |      |

| 27256 (1,h,a) | 12.5 | PM56                | R27256 / P27256              | 32K             | 7FFF |
| 27256 (t)   | 21     | PM56-21 (op)        | R27256 / P27256              |                 |      |
| 27C256 (n)  | 12.5   | PM56                | R27256 / P27256              |                 |      |
| 27C256 (f)  | 21     | will not work /different programming algorithm |                 |      |      |

| 27512 (1)   | 12.5   | PM512 (op)          | R27512 / P27512             | 64K             | FFFF |

* need 64K UDL in 8 bit mode
* need 128K UDL in 16 bit mode

a = AMD
f = Fujitsu
h = Hitachi
i = Intel
n = National
tx = Texas Instruments
t = Toshiba
= optional

Orion also supports:

- 48016 (EEPROM) PM16 P48016
- 2532 PM16 P2532
- 2564 can only emulate
BACKGROUND: ORION INSTRUMENTS, INC., Redwood City, CA

Microprocessor Development Systems and Engineering Instrumentation

Orion Instruments, Inc. was founded in 1980 by Thomas Blakeslee, an electronics engineer himself in search of tools to help solve design development problems. As he was unable to find affordable products of adequate performance, Mr. Blakeslee decided to design his own. Using readily available hardware components in innovative ways, and integrating a capable software environment to control the hardware, Mr. Blakeslee created products of remarkable price-performance and utility.

The company's first major product, the Universal Development Laboratory, or UDL, was aimed at developers of microprocessor based designs. The UDL offered performance previously available only at a much higher price and was adaptable to handle virtually any eight or 16 bit microprocessor with only inexpensive accessories. Mr. Blakeslee's philosophies today remain a cornerstone of Orion's mission statement: "Provide value-rich integrated instrumentation for the engineering professional."

Mr. Blakeslee's accomplishments as the holder of 10 high-tech patents, and as the noted author of a digital logic design textbook used at CalTech, MIT, Stanford, and more than
75 other universities, identify him as an innovator and achiever. He was also founder and V.P. Engineering at Logisticon, Inc., which pioneered the idea of computer automated warehousing equipment.

During its first few years, Orion was largely a family business financed on retained earnings. Products were sold successfully by mail order and telephone to prospects who responded to magazine ads. Although he had built a solid business, Mr. Blakeslee decided that his company's ultimate success would hinge on his being able to assemble a core senior management team geared to rapid growth in today's market. He also wanted to free his own time to concentrate more on product development, and so decided to recruit a new President to manage the company's day-to-day operations. Mr. Blakeslee retains his position as Chairman of Orion's Board, but his daily responsibilities are now as V.P. Research and Development, directing the company's aggressive product development plans.

Orion's President, David E. Kahn, brought with him an impressive and distinguished background. In 1983 he founded and headed up MicroGuild, Inc., a successful software development company of international scope. Earlier, he was Vice President of Marketing for National Nuclear Corporation, where he was responsible for the marketing of walk-through contamination monitors now in use at most U.S. and overseas nuclear power plants. Mr. Kahn has an MBA degree from Stanford
University, an MS degree in Engineering from Carnegie-Mellon University, and a BS degree from Rensselaer Polytechnic Institute.

In March, 1987, Orion recruited Bill White to join the company as V.P. Sales. Mr. White had been V.P. Sales and Marketing at Arium Corporation, a manufacturer of logic analyzers, and V.P. Sales and Marketing at Kikusui, which manufacturers oscilloscopes, before joining Orion. Mr. White was responsible for securing the first large instrumentation order ever placed by the U.S. Government with a Japanese supplier. He enjoys an excellent reputation as a senior sales professional in the electronic instrumentation field.

Mr. White has helped round out the management team necessary for Orion to become a major force in the exploding engineering tools and instrumentation market.

The firm's principal product today is a line of PC-based microprocessor development systems which support more than 150 different microprocessors. Orion Personality Pakstm include all the hardware and software needed for fast and easy connection with the specific target processor.

Orion's UDL, and the newer UniLab II, and OptiLab development systems combine into an integrated working environment, all the tools needed to produce error-free and efficient microprocessor code. This approach saves the user hours of time compared to conventional development systems.

Orion's development systems provide a full selection
of symbolic debug commands so the user can quickly display and change all microprocessor registers, memory and ports, plus set hardware and software breakpoints on demand. Single stepping is also supported. Up to 128K bytes of emulation memory handles nearly any target program.

Both software and hardware problems can usually be tracked down in a hurry with the included 48-channel bus-state analyzer whose powerful triggering language lets the user find bugs by simply specifying their symptoms. This technique goes far beyond the limited "1 0 X" triggering found in most development systems. Further efficiency comes from Orion's useful filter feature which captures only those cycles the user is interested in, eliminating needless sorting through long trace records. The user can focus on either causes or effects of problems by collecting either "pre-trigger" or "post-trigger" data. Program loading from hex or binary disk files, and a line-by-line assembler which permits code changes to be instantly patched in and tested save even more time.

A built-in Stimulus Generator is included to allow system inputs to be specified right from the same instrument. A convenient built-in EPROM Programmer rounds out the instrument feature set.

A recently announced Software Program Performance Analyzer helps optimize code by revealing exactly where a program is spending its time. Full graphical and tabular displays simplify results analysis.
Believing in the relationship between customer satisfaction and company success, Orion has put in place an experienced team of professional Applications Engineers who can be contacted by telephone. After the warranty period, customers can purchase an on-going Support Services Option which includes Extended Warranty coverage and software updates, as well as unlimited telephone Applications Engineering support. A single Support Services Option covers the main system unit and all the specific Orion microprocessor support packages the user may have purchased.

Orion's line provides an ideal platform for product extensions which will bring further automation to the hardware and software development process for microprocessor-based designs. Orion is part of the expanding brigade of PC-based Computer-Aided-Engineering (CAE) tool manufacturers, at the same time forging a clear path helping to define the future of Computer-Integrated Instrumentation. There are thousands of Orion installations worldwide. With new products now well along in the development cycle, Orion intends to further expand the industry's concept of engineering instrumentation and development workstations.

Orion Instruments is privately held, profitable, and enjoys a major line of bank credit. The firm is located in Northern California, just north of famed Silicon Valley, and just south of San Francisco. World-renowned Stanford University is one of Orion's neighbors. This location boasts a
fast-paced climate which is steeped in technology and has fostered so many successful high-tech companies.

**FOR FURTHER INFORMATION CONTACT:**

Orion Instruments, Inc.

702 Marshall Street

Redwood City, CA 94063, USA

Tel: (415) 361-8883

Updated 4.87
CUSTOMER LIST

Here is a partial list of the more than 2000 customers who have chosen the Orion Universal Development Laboratory to support more than 150 microprocessor types. Find out more about how the value-rich UniLab II can benefit your development projects, too!

ADAPTEC, INC.
AEROSPACE CORP.
ALLIED COMMUNICATIONS
AMPEX CORP.
AMPHENOL
ARCO OIL & GAS CO.
A.T. RESEARCH
AT&T
BELL NORTHERN RESEARCH
CERMETEK
CIPHER DATA PRODUCTS
COMBUSTION ENGINEERING
COMMONWEALTH EDISON CO.
COMPUTER MEMORIES
DATAPRODUCTS CORP.
E.I. DUPONT NEMOURS & CO.
EASTMAN KODAK
EATON CORP.
ELECTRONIC DATA SYSTEMS INC.
EMERSON ELECTRIC
EVEREX SYSTEMS
FAIRCHILD WESTON CORP.
FEDERAL EXPRESS
FMC CORP.
FORD AEROSPACE CORP.
FORTUNE SYSTEMS
FOXBORO COMPANY
GENERAL ELECTRIC CO.
GENERAL MOTORS RESEARCH LABS
GENRAD
GEORGIA TECH UNIVERSITY
GOODYEAR
GOULD
GRANGER ASSOCIATES
GTE COMMUNICATIONS
HARRIS DIGITAL TELEPHONE
HEWLETT PACKARD
HONEYWELL INFORMATION SYSTEMS
HUGHES AIRCRAFT COMPANY
IBM CORPORATION
INTEL
ITT
JET PROPULSION LAB
JOHN DEERE HARVESTER
KELTRON CORP.
LAFAYETTE COLLEGE
LAWRENCE LIVERMORE LABS
LOCKHEED ELECTRONICS CORP.
LOCKHEED MISSILES & SPACE CORP.
LOS ALAMOS NATIONAL LABS.
LOUISIANA TECHNICAL UNIVERSITY
MAGNAVOX ELECTRONICS
MARTIN MARIETTA
Mitre CORPORATION
MITSUBISHI PRO AUDIO
MOBIL OIL
MOLECULAR DEVICES
MOTOROLA
NASA
NATIONAL SEMICONDUCTOR
NCR CORP.
NEC
NEW YORK STOCK EXCHANGE
NORTHERN BELL
NORTHROP CORP.
OREGON INSTITUTE OF TECHNOLOGY
PACIFIC BELL
PANAMETRICS
PELOUZE SCALE CO.
PITNEY BOWES
PRATT & WHITNEY
PRINCETON UNIVERSITY
PROCTOR & GAMBLE
PURDUE UNIVERSITY
QUANTEL
RACAL VADIC
RAYCHEM
RAYTHEON
ROBERTSHAW CONTROLS CO.
ROLM CORPORATION
SAINT LAWRENCE COLLEGE
SAMSUNG
SAN FRANCISCO STATE UNIVERSITY
SCIENTIFIC ATLANTA
SIEMENS CORP.
SONY CORP. OF AMERICA
SPECTRA PHYSICS CORPORATION
SPERRY CORPORATION
SRI INTERNATIONAL
SUFFOLK UNIVERSITY
3M CORPORATION
TRW
UNIVERSITY OF CALIFORNIA
UNIVERSITY OF MISSOURI
UNIVERSITY OF OREGON
UNIVERSITY OF RHODE ISLAND
UNIVERSITY OF VIRGINIA
U.S. GEOLOGICAL SURVEY
US INSTRUMENTS RENTAL
VARIAN
VIRGINIA POLYTECHNIC INSTITUTE
WAVETEK
WESTERN DIGITAL CORPORATION
WESTINGHOUSE CORP.
WOODS HOLE OCEANOGRAPHIC INST.
WYSE TECHNOLOGY
XEROX CORPORATION
ZENITH