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 MICROS

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.

(7/87) Printed in the USA

ORION UniLab II™ — Universal Development Laboratory

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)

PART NO.		PRICE
84101	UniLab II (32K emulation memory)	$ 4980.00
84102	UniLab II (64K emulation memory)	$ 5380.00
84103	UniLab II (128K emulation memory)	$ 5580.00

HIGH-SPEED UNILAB II

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

PART NO.		PRICE
84201	UniLab II (32K high-speed emulation memory)	$ 5380.00
84202	UniLab II (64K high-speed emulation memory)	$ 5780.00
84203	UniLab II (128K high-speed emulation memory)	$ 5980.00

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)

PART NO.		PRICE
82101	Universal Development Laboratory (32K emulation memory)	$ 2995.00
82102	Universal Development Laboratory (64K emulation memory)	$ 3390.00
82103	Universal Development Laboratory (128K emulation memory)	$ 3590.00

HIGH-SPEED UDL

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

PART NO.		PRICE
82201	Universal Development Laboratory (32K high speed emulation memory)	$ 3430.00
82202	Universal Development Laboratory (64K high speed emulation memory)	$ 3855.00
82203	Universal Development Laboratory (128K high speed emulation memory)	$ 4085.00

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 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.

ORION

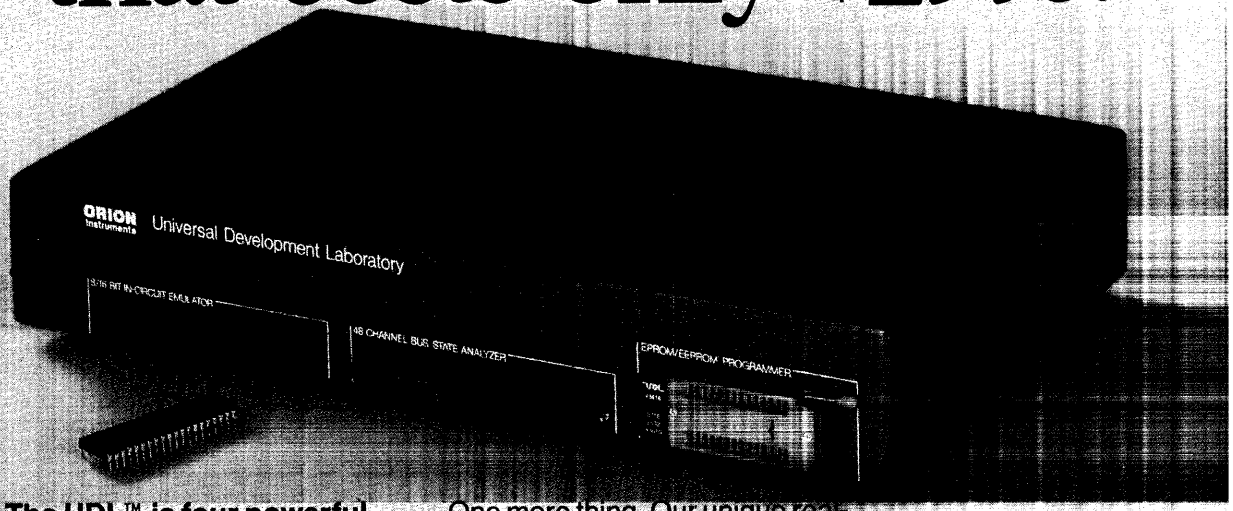
702 Marshall Street
Redwood City, California 94063
Telex: 530942
FAX: 415-361-8970

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

Why companies that can afford any I^2 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 I^2 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 I^2 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.

The UDL runs under PC-DOS® MS-DOS® CP-M® ISIS® or RT-II® Adapters available for Apple and VAX®

ORION Instruments

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

ORION

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 — **Lucky Balin**
32184 Oak Shore Drive
Westlake Village, CA 91361

Telephone: **818/706-8351**

- -

APS — Mort Katz
847 Triunfo Canyon Rd.
Westlake Village, CA 91361

Telephone: 805/496-7355

- -

APS — Mike Knapp
13130 Decant Drive
Poway, CA 92064

Telephone: 619/530-0200

ROCKY MOUNTAIN

Scientific Devices Inc. - SDI — **Brian Clark**
1006 Depot Hill Road — (also temporarily covering Utah)
Broomfield, CO 80020

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

- -

SDI — Jeff Gipe
3920 E. Indian School Rd.
Phoenix, AZ 85018

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

SDI
5215 Los Arboles NE
Albuquerque, NM 87110

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

Jim Vermillion

- -

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

Telephone: 714/840-3501

** Billing address only.

Dave Orahood

NORTHWEST

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

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

Craig Howland
Wayne Van Zandt

NO. CALIFORNIA

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

Telephone: **415/591-6535**

Jim Williams
Brad Rulien
Terri Tellez

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

- -

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

Telephone: 713/376-8666

Rick Morgenstern

- -

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

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

Terry Bissell

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 — **Dick Brown**
1728 Rue Mirador
Point Pleasant, NJ 08742

Telephone: **201/295-1357**

KEYSTONE STATES

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

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

CHESAPEAKE

Scientific Associates, Inc. - SAI — **Donald George**, Richard Keyes, Mark Holtzer, David Day, Donald Williams, Dennis McFadden
9512A Lee Highway
Fairfax, VA 22031

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

SOUTHEAST

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

Telephone: **813/785-2276**

- -

TCS — Dennis Harrington
1325 N. Atlantic Ave. #192
Cocoa Beach, FL 32931
Telephone: 305/783-3444

TCS
PO Box 11060
Ocala, FL 32673

Berni Preuss

Telephone: 904/347-1707

- -

TCS
PO Box 2786
Norcross, GA 30071-2786

Harold Donehoo

Telephone: 404/242-7215

- -

TCS
1019 A Old Monrovia RD
Suite 132
Huntsville, AL 35806

Cuz Lankford

Telephone: 205/883-6699

- -

TCS
3919 Quiling Road
Winston Salem, NC 27104

Bob Dinning

Telephone: 919/765-5553

NORTHEAST

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

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

Telephone: **617/658-3800**

7.28.87 REPLIST

UDL Specifications

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 × 8-bit or 16K × 16-bit standard. By cable, program option.
Expandable to 128K 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 debug 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 × 48 bit Trace memory

EPROM/EEPROM Programmer
Smart programming algorithm for high speed.
28-pin Textool 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 × 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 65,382 events or cycles before 4th step.
Before/After/At Pass count trigger enable.
Delay Counter: wait up to 65,382 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, memo 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 × 13″ wide × 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, 2516, 2532 EPROMs and 48016 EEPROMs, 2732A module, 2764/128 module, 27256/2764A/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-RS-232 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, 6301/3, 6500/02/C02, 6800/2/8, 6801/3, 6805, 6809E, 68000, 68008, 8048/35/39/40/49/50, 8051/31/32/52, 8085/80, 8086/186/286, 8088/188, Z-8, Z-80 and NSC-800.

(XUFQ PRICE LIST 06/01/87 08:57
)DB'S, EMULATION MODULES, MICROTARGETS, AND PERSONALITY PAKS

)C IPTION		PART ID	REV-RTN #	LIST PRICE
DB48	UDL, GROUP 48	77001	4203-907	395.000
)DB48	UNILAB, GROUP 48	77501	4203-907	395.000
)DB51	UDL, GROUP 51	77002	4203-907	395.000
)DB51	UNILAB, GROUP 51	77502	4203-907	395.000
)DB51P	UDL, GROUP 51	77003	4203-907	395.000
)DB51P	UNILAB, GROUP P	77503	4203-907	395.000
)DB611	UDL, GROUP 61	77004	4203-907	395.000
)DB611	UNILAB, GROUP 61	77504	4203-907	395.000
)DB63	UDL, GROUP 63	77005	4203-907	395.000
)DB63	UNILAB, GROUP 63	77505	4203-907	395.000
)DB65	UDL, GROUP 65	77006	4203-907	395.000
)DB65	UNILAB, GROUP 65	77506	4203-907	395.000
)DB65P	UDL, GROUP 11	77007	4203-907	395.000
)DB65P	UNILAB, GROUP 11	77507	4203-907	395.000
)DB68	UDL, GROUP 68	77008	4203-907	395.000
)DB68	UNILAB, GROUP 68	77508	4203-907	395.000
)DB681	UDL, GROUP 01	77009	4203-907	395.000
)DB681	UNILAB, GROUP 01	77509	4203-907	395.000
)DB682	UDL, GROUP 02	77010	4203-907	395.000
)DB682	UNILAB, GROUP 02	77510	4203-907	395.000
)DB685	UDL, GROUP 05	77011	4203-907	395.000
DB685	UNILAB, GROUP 05	77511	4203-907	395.000
DB688	UDL, GROUP 08	77012	4203-907	395.000
D[illegible]88	UNILAB, GROUP 08	77512	4203-907	395.000
D[illegible]9	UDL, GROUP 09	77013	4203-907	395.000
)DB689	UNILAB, GROUP 09	77513	4203-907	395.000
B68K	UDL, GROUP 68K	77014	4203-907	495.000
DB68K	UNILAB, GROUP 68K	77514	4203-907	495.000
DDB85	UDL, GROUP 85	77015	4203-907	395.000
DDB85	UNILAB, GROUP 85	77515	4203-907	395.000
DDB86	UDL, GROUP 86	77016	4203-907	495.000
DDB86	UNILAB, GROUP 86	77516	4203-907	495.000
DDB88	UDL, GROUP 88	77017	4203-907	395.000
DDB88	UNILAB, GROUP 88	77517	4203-907	395.000
DDB96	UDL, GROUP 96	77018	4203-907	495.000
DDB96	UNILAB, GROUP 96	77518	4203-907	495.000
DDBS8	UDL, GROUP S8	77019	4203-907	395.000
DDBS8	UNILAB, GROUP S8	77519	4203-907	395.000
DDBZ8	UDL, GROUP 08	77020	4203-907	395.000
DDBZ8	UNILAB, GROUP 08	77520	4203-907	395.000
DDBZ80	UDL, GROUP 80	77021	4203-907	395.000
DDBZ80	UNILAB, GROUP 80	77521	4203-907	395.000
DDBZ8K	UDL, GROUP Z8K	77022	4203-907	495.000
DDBZ8K	UNILAB, GROUP Z8K	77522	4203-907	495.000
DSM18	UDL, GROUP 18	77023	4203-907	200.000
DSM18	UNILAB, GROUP 18	77523	4203-907	200.000
EM6303R	EMULATION MODULE, 6303R	78014	4202-906	225.000
EM63P01	EMULATION MODULE	78003	4202-906	275.000
EM64180	EMULATION MODULE 64180	78012	4202-906	225.000
EM[illegible]02	EMULATION MODULE 6502	78010	4202-906	175.000
EM[illegible]5E2	EMULATION MODULE 6805E2	78015	4202-906	225.000
EM6805P	EMULATION MODULE	78007	4202-906	275.000
EM6809	EMULATION MODULE 6809	78001	4202-906	175.000

PPAK EM MT

XUFQ :
)DB'S,, EMULATION MODULES, MICROTARGETS, AND PERSONALITY PAKS

PRICE LIST

06/01/87 08:57

)ESCRIPTION	PART ID	REV-RTN #	LIST PRICE
EM EMULATION MODULE 68000	78013	4202-906	225.000
8015P EMULATION MODULE 8051P	78005	4202-906	375.000
M8031 EMULATION MODULE 8031	78008	4202-906	175.000
EM8048P EMULATION MODULE 8048P	78006	4202-906	275.000
EM8085 EMULATION MODULE 8085	78009	4202-906	175.000
EM8086MAX EMULATION MODULE	78016	4202-906	225.000
EM8086MIN EMULATION MODULE	78002	4202-906	225.000
EM8088MAX EMULATION MODULE 8088MAX	78017	4202-906	175.000
EM8088MIN EMULATION MODULE 8088MIN	78011	4202-906	175.000
EMZ8 EMULATION MODULE	78018	4202-906	175.000
EMZ80 EMULATION MODULE Z80	78004	4202-906	175.000
MT63P01 MICROTARGET	78507	4202-906	250.000
MT64180 MICROTARGET	78513	4202-906	200.000
MT6502 MICROTARGET	78508	4202-906	150.000
MT68K MICROTARGET	78510	4202-906	200.000
MT68P05 MICROTARGET	78509	4202-906	200.000
MT8031 MICROTARGET	78505	4202-906	150.000
MT8051P MICROTARGET	78502	4202-906	250.000
MT8085 MICROTARGET	78506	4202-906	175.000
MT8086MAX MICROTARGET	78504	4202-906	300.000
MT8086MIN MICROTARGET	78503	4202-906	250.000
MT8088MAX MICROTARGET	78512	4202-906	200.000
MT8088MIN MICROTARGETBOARD	78511	4202-906	200.000
MTZ8 MICROTARGET	78515	4202-906	175.000
MTZ80 MICROTARGET	78514	4202-906	100.000
PF 118 UDL	79023	4201-909	350.000
PK 8 UNILAB	79523	4201-909	350.000
PPAK63 UDL	79027	4201-909	600.000
AK63 UNILAB	79527	4201-909	600.000
PAK6303R UDL	79028	4201-909	550.000
PPAK6303R UNILAB	79528	4201-909	550.000
PPAK6305 UDL	79030	4201-909	550.000
PPAK6305 UNILAB	79530	4201-909	550.000
PPAK6305F UDL	79029	4201-909	600.000
PPAK6305F UNILAB	79529	4201-909	600.000
PPAK63P01 UDL	79005	4201-909	800.000
PPAK63P01 UNILAB	79505	4201-909	800.000
PPAK63P05 UDL	79031	4201-909	650.000
PPAK63P05 UNILAB	79531	4201-909	650.000
PPAK65 UDL	79006	4201-909	550.000
PPAK65 UNILAB	79506	4201-909	550.000
PPAK65/11 UDL	79035	4201-909	550.000
PPAK65/11 UNILAB	79535	4201-909	550.000
PPAK6502 UDL	79033	4201-909	600.000
PPAK6502 UNILAB	79533	4201-909	600.000
PPAK6510 UDL	79034	4201-909	500.000
PPAK6510 UNILAB	79534	4201-909	500.000
PPAK6511Q UDL	79007	4201-909	500.000
PPAK6511Q UNILAB	79507	4201-909	500.000
PPAK6512 UNILAB	79532	4201-909	500.000
PPAK6541Q UDL	79036	4201-909	500.000
PP 6541Q UNILAB	79536	4201-909	500.000
PK 5P UDL	79032	4201-909	500.000
PPAK64180 Unilab	74552	4201-906	700.000

EM, MT, PPAK

(XUFQ PRICE LIST 06/01/87 08:57
)DB'S, EMULATION MODULES, MICROTARGETS, AND PERSONALITY PAKS

DESCRIPTION	PART ID	REV-RTN #	LIST PRICE
PAK68 UDL	79008	4201-909	500.000
PAK68 UNILAB	79508	4201-909	500.000
PAK68008 UDL	79012	4201-909	550.000
PAK68008 UNILAB	79512	4201-909	550.000
PAK68010 UDL	79038	4201-909	650.000
PAK68010 UNILAB	79538	4201-909	650.000
PAK6805E2 UDL	79011	4201-909	550.000
PAK6805E2 UNILAB	79511	4201-909	550.000
PAK6805P UDL	79040	4201-909	750.000
PAK681 UDL	79009	4201-909	500.000
PAK681 UNILAB	79509	4201-909	500.000
PAK6811 UDL	79004	4201-909	550.000
PAK6811 UNILAB	79504	4201-909	550.000
PAK682 UDL	79010	4201-909	500.000
PAK682 UNILAB	79510	4201-909	500.000
PAK685 UDL	79039	4201-909	500.000
PAK685 UNILAB	79539	4201-909	500.000
PAK689 UDL	79013	4201-909	500.000
PAK689 UNILAB	79513	4201-909	500.000
PAK68K UDL	79014	4201-909	800.000
PAK68K UNILAB	79514	4201-909	800.000
PAK68P01 UDL	79037	4201-909	550.000
PAK68P01 UNILAB	79537	4201-909	550.000
P 8P05 UNILAB	79540	4201-909	750.000
P 00 UDL	79051	4201-909	500.000
PAK800 UNILAB	79551	4201-909	500.000
AK80186 UDL	79042	4201-909	650.000
PAK80186 UNILAB	79542	4201-909	650.000
PAK80188 UDL	79043	4201-909	550.000
PAK80286 UDL	79044	4201-909	650.000
PAK80286 UNILAB	79543	4201-909	550.000
PAK80286 UNILAB	79544	4201-909	650.000
PAK8031 UNILAB	79502	4201-909	600.000
PAK8048 UDL	79001	4201-909	500.000
PAK8048 UNILAB	79501	4201-909	500.000
PAK8048P UDL	79049	4201-909	550.000
PAK8048P UNILAB	79549	4201-909	550.000
PAK8051 UDL	79002	4201-909	600.000
PAK8051P UDL	79003	4201-909	850.000
PAK8051P UNILAB	79503	4201-909	850.000
PAK8080 UDL	79015	4201-909	500.000
PAK8080 UNILAB	79515	4201-909	500.000
PAK8085 UDL	79045	4201-909	650.000
PAK8085 UNILAB	79545	4201-909	650.000
PAK8086MAX UDL	79025	4201-909	850.000
PAK8086MAX UNILAB	79525	4201-909	850.000
PAK8086MIN UDL	79016	4201-909	800.000
PAK8086MIN UNILAB	79516	4201-909	800.000
PAK8088MAX UDL	79026	4201-909	650.000
P 8088MAX UNILAB	79526	4201-909	650.000
P 088MIN UDL	79017	4201-909	650.000
PAK8088MIN UNILAB	79517	4201-909	650.000
PAK8097 UDL	79046	4201-909	650.000

PPAK

:XUFQ PRICE LIST 06/01/87 08:57
)DB'S, EMULATION MODULES, MICROTARGETS, AND PERSONALITY PAKS

SCRIPTION	PART ID	REV-RTN #	LIST PRICE
PAK8097 UNILAB	79546	4201-909	650.000
PAK96 UDL	79018	4201-909	650.000
PAK96 UNILAB	79518	4201-909	650.000
PAKPLAIN UDL	79024	4201-909	200.000
PAKPLAIN UNILAB	79524	4201-909	200.000
PAKS8 UDL	79019	4201-909	500.000
PAKS8 UNILAB	79519	4201-909	500.000
PAKS8F UDL	79050	4201-909	550.000
PAKS8F UNILAB	79550	4201-909	550.000
PAKZ8 UDL	79048	4201-909	650.000
PAKZ8 UNILAB	79548	4201-909	650.000
PAKZ80 UDL	79021	4201-909	700.000
PAKZ80 UNILAB	79521	4201-909	700.000
PAKZ8001 UDL	79022	4201-909	650.000
PAKZ8001 UNILAB	79522	4201-909	650.000
PAKZ8002 UDL	79041	4201-909	600.000
PAKZ8002 UNILAB	79541	4201-909	600.000
PAKZ8F UDL	79020	4201-909	550.000
PAKZ8F UNILAB	79520	4201-909	550.000
PAKZ8P64 UDL	79047	4201-909	550.000
PAKZ8P64 UNILAB	79547	4201-909	550.000

180 RECORDS PRINTED

PPAK

XXUFQ PRICE LIST 06/01/87 09:07
U S

DESCRIPTION		PART ID	REV-RTN #	LIST PRICE
UDL128	UDL 128K STD SPEED	82103	4101-902	3590.000
UDL128	UDL 128K STD SPEED, AUSTRALIAN	82133	4101-902	3590.000
UDL128	UDL 128K STD SPEED, BRITISH	82113	4101-902	3740.000
UDL128	UDL 128K STD SPEED, GERMAN	82123	4101-902	3590.000
UDL128	UDL 128K STD SPEED, THORN	82143	4101-902	3590.000
UDL128HS	UDL 128K HIGH SPEED, AUSTRALIAN	82233	4101-902	4085.000
UDL128HS	UDL 128K HIGH SPEED, BRITISH	82213	4101-902	4235.000
UDL128HS	UDL 128K HIGH SPEED, GERMAN	82223	4101-902	4085.000
UDL128HS	UDL 128K HIGH SPEED, THORN	82243	4101-902	4085.000
UDL128HS	UDL 128K STD SPEED	82203	4101-902	4085.000
UDL32	UDL 32K STD SPEED	82101	4101-902	2995.000
UDL32	UDL 32K STD SPEED, AUSTRALIAN	82131	4101-902	2995.000
UDL32	UDL 32K STD SPEED, BRITISH	82111	4101-902	3145.000
UDL32	UDL 32K STD SPEED, GERMAN	82121	4101-902	2995.000
UDL32	UDL 32K STD SPEED, THORN	82141	4101-902	2995.000
UDL32HS	UDL 32K STD SPEED	82201	4101-902	3430.000
UDL32HS	UDL 32K HIGH SPEED, AUSTRALIAN	82231	4101-902	3430.000
UDL32HS	UDL 32K HIGH SPEED, BRITISH	82211	4101-902	3580.000
UDL32HS	UDL 32K HIGH SPEED, GERMAN	82221	4101-902	3430.000
UDL32HS	UDL 32K STD SPEED, THORN	82241	4101-902	3430.000
UDL64	UDL 64K STD SPEED	82102	4101-902	3390.000
UDL64	UDL 64K STD SPEED, AUSTRALIAN	82132	4101-902	3390.000
U 4	UDL 64K STD SPEED, BRITISH	82112	4101-902	3540.000
UDL64	UDL 64K STD SPEED, GERMAN	82122	4101-902	3390.000
DL64	UDL 64K STD SPEED, THORN	82142	4101-902	3390.000
DL64HS	UDL 64K HIGH SPEED	82202	4101-902	3855.000
UDL64HS	UDL 64K HIGH SPEED, AUSTRALIAN	82232	4101-902	3855.000
UDL64HS	UDL 64K HIGH SPEED, BRITISH	82212	4101-902	4005.000
UDL64HS	UDL 64K HIGH SPEED, GERMAN	82222	4101-902	3855.000
UDL64HS	UDL 64K HIGH SPEED, THORN	82242	4101-902	3855.000

30 RECORDS PRINTED

UDL

G:\GARY>TYPE PRCLST5S.RPT

06/03/87 17:11 ORION INSTRUMENTS PAGE 1

XY 1 PRICE LIST 06/01/87 09:10

LAB'S

DESCRIPTION	PART ID	REV-RTN #	LIST PRICE
UNI128 UNILAB II 128K STD SPEED,BRITISH	84113	4102-903	5730.000
UNI128 UNILABII 128K STD SPEED, AUSTRAL	84133	4102-903	5580.000
UNI128 UNILABII 128K STD SPEED, DOMESTI	84103	4102-903	5580.000
UNI128 UNILABII 128K STD SPEED, GERMAN	84123	4102-903	5580.000
UNI128 UNILABII 128K STD SPEED, THORN	84143	4102-903	5580.000
UNI128HS UNILAB II 128K HIGH SPEED,BRIT	84213	4102-903	6130.000
UNI128HS UNILABII 128K HIGH SPEED AUSTRL	84233	4102-903	5980.000
UNI128HS UNILABII 128K HIGH SPEED, THORN	84243	4102-903	5980.000
UNI128HS UNILABII 128K HIGH SPEED,DOMEST	84203	4102-903	5980.000
UNI128HS UNILABII 128K HIGH SPEED,GERMAN	84223	4102-903	5980.000
UNI32 UNILAB II 32K STD SPEED, BRITISH	84111	4102-903	5130.000
UNI32 UNILAB II 32K STD SPEED, DOMEST	84101	4102-903	4980.000
UNI32 UNILABII 32K STD SPEED, AUSTRAIL	84131	4102-903	4980.000
UNI32 UNILABII 32K STD SPEED, GERMAN	84121	4102-903	4980.000
UNI32 UNILABII 32K STD SPEED, THORN	84141	4102-903	4980.000
UNI32HS UNILABII 32K HIGH SPEED, AUSTRL	84231	4102-903	5380.000
UNI32HS UNILABII 32K HIGH SPEED, DOMEST	84201	4102-903	5380.000
UNI32HS UNILAB II 32K HIGH SPEED,BRITISH	84211	4102-903	5530.000
UNI32HS UNILABII 32K HIGH SPEED, GERMAN	84221	4102-903	5380.000
UNI32HS UNILABII 32K HIGH SPEED, THORN	84241	4102-903	5380.000
UNI64 UNILAB II 64K STD SPEED, BRITISH	84112	4102-903	5530.000
UNI64 UNILABII 64K STD SPEED, AUSTRAIL	84132	4102-903	5380.000
UNI64 UNILABII 64K STD SPEED, DOMESTIC	84102	4102-903	5380.000
UNI64 UNILABII 64K STD SPEED, GERMAN	84122	4102-903	5380.000
UNI64 UNILABII 64K STD SPEED, THORN	84142	4102-903	5380.000
UNI64HS UNILAB II 64K HIGH SPEED,BRITISH	84212	4102-903	5930.000
UNI64HS UNILABII 64K HIGH SPEED, AUSTRAL	84232	4102-903	5780.000
UNI64HS UNILABII 64K HIGH SPEED, DOMESTI	84202	4102-903	5780.000
UNI64HS UNILABII 64K HIGH SPEED, GERMAN	84222	4102-903	5780.000
UNI64HS UNILABII 64K HIGH SPEED, THORN	84242	4102-903	5780.000

30 RECORDS PRINTED

Unilabs

XXUFQ PRICE LIST 06/01/87 09:02
PERSONALITY MODULES

DESCRIPTION		PART ID	REV-RTN #	LIST PRICE
PM16	PERSONALITY MODULE 2716	73002	4202-906	50.000
PM32	PERSONALITY MODULE 2732A	73004	4202-906	50.000
PM3212	PERSONALITY MODULE 2732B, 12V	73014	4202-906	50.000
PM512	PERSONALITY MODULE 27512	73016	4202-906	50.000
PM5621A	PERS. MOD.,27256, 21V, REV A	73A10	4202-906	50.000
PM5621B	PERS. MOD. 27256, 21V, REV B	73010	4202-906	50.000
PM56A	PERSONALITY MODULE 27256, REV A	73A06	4202-906	50.000
PM56B	PERSONALITY MODULE 27256, REV B	73006	4202-906	50.000
PM64A	PERSONALITY MODULE 2764, REV A	73A08	4202-906	50.000
PM64B	PERSONALITY MODULE 2764, REV B	73008	4202-906	50.000

10 RECORDS PRINTED

Personality Modules

XXUFQ PRICE LIST 06/01/87 08:53
CABLES, UPGRADES

DESCRIPTION	PART ID	REV-RTN #	LIST PRICE
C1624 CABLE,EM, 16BIT/24PIN	74905	4202-906	95.000
C1628 CABLE, EMULAT, 16BIT/28PIN	74906	4202-906	95.000
C16D CABLE, EMULAT, 16BIT/DIRECT	74907	4202-906	85.000
C824 CABLE, ROM, 8BIT/24PIN	74900	4202-906	75.000
C828 CABLE, EMULAT, 8BIT/28PIN	74901	4202-906	75.000
C8D CABLE, EMULAT, 8BIT/DIRECT	74902	4202-906	85.000
CA CABLE, ANALY, GROUP A	74703	4202-906	85.000
CB CABLE, ANALY, GROUP B	74704	4202-906	85.000
CC CABLE, ANALY, GROUP C	74705	4202-906	85.000
CD CABLE, ANALY, GROUP D	74706	4202-906	85.000
CE CABLE, ANALY, GROUP E	74707	4202-906	85.000
CF CABLE, ANALY, GROUP F	74708	4202-906	85.000
CG CABLE, ANALY, GROUP G	74709	4202-906	85.000
CH CABLE, ANALY, GROUP H	74710	4202-906	85.000
CI CABLE, ANALY, GROUP I	74711	4202-906	85.000
CK CABLE, ANALY, GROUP K	74712	4202-906	85.000
CL CABLE, ANALY, GROUP L	74713	4202-906	85.000
CM CABLE, ANALY, GROUP M	74714	4202-906	150.000
CN CABLE, ANALY, GROUP N	74715	4202-906	85.000
CP CABLE, ANALY, GROUP P	74716	4202-906	85.000
CQ CABLE, ANALY, GROUP Q	74717	4202-906	85.000
CR CABLE, ANALY, GROUP R	74718	4202-906	85.000
CS CABLE, ANALY, GROUP S	74719	4202-906	85.000
CT CABLE, ANALY, GROUP T	74720	4202-906	85.000
F 64180 UNILAB	74552	4201-909	700.000
UNIUP UDL UPGRADE TO UNILAB	74322	4203-907	1985.000
UPGRADE 128K STD TO 128K HIGH SPEED	74318	4202-906	600.000
UPGRADE 32 STD TO 128 HIGH SPEED	74314	4202-906	1200.000
UPGRADE 32 STD TO 128 STD	74313	4202-906	800.000
UPGRADE 32 STD TO 32 HIGH SPEED	74310	4202-906	600.000
UPGRADE 32 STD TO 64 HIGH SPEED	74312	4202-906	1000.000
UPGRADE 32 STD TO 64 STD	74311	4202-906	600.000
UPGRADE 32K HIGH SPEED TO 128K H.S.	74320	4202-906	800.000
UPGRADE 32K HIGH SPEED TO 64K HIGH SPEED	74319	4202-906	600.000
UPGRADE 64 STD TO 64 HIGH SPEED	74315	4202-906	600.000
UPGRADE 64K HIGH SPEED TO 128K H.S.	74321	4202-906	400.000
UPGRADE 64K STD TO 128K HIGH SPEED	74317	4202-906	800.000
UPGRADE 64K STD TO 128K STD	74316	4202-906	400.000

38 RECORDS PRINTED

Cables - Upgrades

06/01/87

MISCELLANEOUS PART NUMBERS PAGE 1

PART DESCRIPTION		PART ID	STD COST	LIST PRICE
16 PIN DIP CLIP		21036	4202-906	$15.00
(note: usually shipped in units of 5)				
40PINCLIP	40 PIN DIP CLIP	21037	4202-906	55.00
48PINCLIP	48 PIN DIP CLIP	21038	4202-906	65.00
64P70MADPT	64PIN 70MIL MPU ADAPTER	74008	4202-906	100.00
64PINCLIP	64 PIN DIP CLIP	21046	4202-906	95.00
E220	220VAC/50HZ POWER SUPPLY	40003	4202-906	150.00
PB40SK	40 PIN TEST SOCKET	21063	4202-906	95.00
PB48ADAPT	ADAPTER 48PIN PIGGYBACK	21062	4202-906	96.00
PGADIP	ADAPTER 64PIN PGA-TO-DIP	74001	4202-906	275.00
PPA	PROG. PERFORMANCE ANALYZER	87001	4203-907	495.00
PROGO	UNILAB PROGRAMMERS GUIDE	87002	4301-901	35.00
QUIPADAPT	64 PIN QUIP ADAPTOR	74009	4202-906	60.00
UDLMAN	UDL MANUAL	81040	4301-901	25.00
UNI2MAN	UNILAB MANUAL	83040	4301-901	100.00
UPGRADE/B	UDL REV.B UPGRADE	74323	4202-906	99.00
XASM	CROSS ASSEMBLER	SEE ANNE	4202-906	

ORION
Instruments

HOW PC-BASED 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 microprocessor development system on the engineer's desk. With this power, project development time can be reduced significantly 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^2) 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^2 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^2 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 F0 TO FF DATA

8000 TO 8080 ADR will cause the analyzer to trigger if data greater than F0 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

```
cy# CONT      ADR DATA
 -5   B7     00A6 20FA      JR NZ,START-LOOP
 -3   B7     00A8 C9        RET
 -2   F7     17FE FF read
 -1   F7     17FF FF read
  0   B7     FFFF 1831      JR 32
```

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_{HEX}; RAM occupies locations 1800_{HEX} through $1FFF_{HEX}$.

LISTING 2

```
5 SR resetting

 cy# CONT          ADR DATA
  -5   B7         00A6 20FA      JR NZ,START-LOOP
  -3   B7         00A8 C9        RET
  -2   F7         17FE FF read
  -1   F7         17FF FF read
   0   B7         FFFF 1831      JR 32
resetting
 cy# CONT          ADR DATA
  -5   B7         00A6 20FA      JR NZ,START-LOOP
  -3   B7         00A8 C9        RET
  -2   F7         17FE FF read
  -1   F7         17FF FF read
   0   B7         FFFF 1831      JR 32
resetting
 cy# CONT          ADR DATA
  -5   B7         00A6 20FA      JR NZ,START-LOOP
  -3   B7         00A8 C9        RET
  -2   F7         17FE FF read
  -1   F7         17FF FF read
   0   B7         FFFF 1831      JR 32
```

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 $FFFF_{HEX}$; the RET instruction is trying to read the stack at $17FF_{HEX}$, where there is no RAM. The stack pointer was initialized to $18FF_{HEX}$, 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.

LISTING 3

```
cy# CONT     ADR DATA
F01  B7     00A8 C9        RET
F02  F7     18FA 90 read
F03  B7     00A8 C9        RET
F04  F7     18F8 90 read
F05  B7     00A8 C9        RET
F06  F7     18F6 90 read
F07  B7     00A8 C9        RET
F08  F7     18F4 90 read
F09  B7     00A8 C9        RET
F0A  F7     18F2 90 read
F0B  B7     00A8 C9        RET
F0C  F7     18F0 90 read
F0D  B7     00A8 C9        RET
F0E  F7     18EE 90 read
F0F  B7     00A8 C9        RET
F10  F7     18EC 90 read
```

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 $A8_{HEX}$ and the stack address accessed by that instruction. The trace (**Listing 3**) shows that each time the program reaches address $A8_{HEX}$, 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 $A8_{HEX}$, 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 $8B_{HEX}$ 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 $8B_{HEX}$ with a simple monitor command. After making that change,

LISTING 4

cy#	CONT		ADR	DATA	
-5	B7		00A3	20FD	JR NZ,START-LOOP
-3	B7		00A5	25	DEC H
-2	B7		00A6	20FA	JR NZ,START-LOOP
0	B7		00A8	C9	RET
1	F7		18FA	8F read	
2	F7		18FB	00 read	
3	B7		008F	C38800	JP SEND-OUT
6	B7	SEND-OUT	0088	D379	OUT (LEDS),A
8	5F		8079	80 out	
9	B7		008A	0F	RRCA
A	B7		008B	D5	PUSH DE
B	D7		18FB	40 write	
C	D7		18FA	20 write	
D	B7		008C	CDA000	CALL DELAY-SUB

LISTING 5

cy#	CONT		ADR	DATA	
-5	B7		00A3	20FD	JR NZ,START-LOOP
-3	B7		00A5	25	DEC H
-2	B7		00A6	20FA	JR NZ,START-LOOP
0	B7		00A8	C9	RET
1	F7		18FC	8F read	
2	F7		18FD	00 read	
3	B7		008F	C38800	JP SEND-OUT
6	B7	SEND-OUT	0088	D379	OUT (LEDS),A
8	5F		8079	80 out	
9	B7		008A	0F	RRCA
A	B7		008B	00	NOP
B	B7		008C	CDA000	CALL DELAY-SUB

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

```
DELAY-SUB   00A0   21FF40    LD HL,40FF
START-LOOP  00A3   2D        DEC L
            00A4   20FD      JR NZ,START-LOOP
            00A6   25        DEC H
            00A7   20FA      JR NZ,START-LOOP
            00A9   C9        RET
```

LISTING 7

```
AF=8051 (sZ-A-pnC) BC=   0 DE=  20 HL=FFFF IX=1234 IY=5678 SP=18FC PC=A0
DELAY-SUB  00A0  21FF40   LD HL,40FF                          (next step)
```

(a)

```
NMI
AF=8051 (sZ-A-pnC) BC=   0 DE=  20 HL=40FF IX=1234 IY=5678 SP=18FC PC=A3
START-LOOP 00A3  2D       DEC L                               (next step)
NMI
AF=80AB (Sz-a-pNC) BC=   0 DE=  20 HL=40FE IX=1234 IY=5678 SP=18FC PC=A4
           00A4  20FD     JR NZ,START-LOOP                    (next step)
NMI
AF=80AB (Sz-a-pNC) BC=   0 DE=  20 HL=40FE IX=1234 IY=5678 SP=18FC PC=A3
START-LOOP 00A3  2D       DEC L                               (next step)
NMI
AF=80AB (Sz-a-pNC) BC=   0 DE=  20 HL=40FD IX=1234 IY=5678 SP=18FC PC=A4
           00A4  20FD     JR NZ,START-LOOP                    (next step)
NMI
AF=80AB (Sz-a-pNC) BC=   0 DE=  20 HL=40FD IX=1234 IY=5678 SP=18FC PC=A3
START-LOOP 00A3  2D       DEC L                               (next step)
NMI
AF=80AB (Sz-a-pNC) BC=   0 DE=  20 HL=40FC IX=1234 IY=5678 SP=18FC PC=A4
           00A4  20FD     JR NZ,START-LOOP                    (next step)
```

(b)

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_{HEX} times, and an outer loop that runs 40_{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

```
cy# CONT            ADR DATA
 -4  B7 START-LOOP 00A3 2D       DEC L
 -3  B7            00A4 20FD     JR NZ,START-LOOP
 -1  B7 START-LOOP 00A3 2D       DEC L
  0  B7            00A4 20FD     JR NZ,START-LOOP
  2  B7            00A6 25       DEC H
  3  B7            00A7 20FA     JR NZ,START-LOOP
  5  B7 START-LOOP 00A3 2D       DEC L
  6  B7            00A4 20FD     JR NZ,START-LOOP
  8  B7 START-LOOP 00A3 2D       DEC L
  9  B7            00A4 20FD     JR NZ,START-LOOP
```

LISTING 9

```
cy# CONT            ADR DATA
 -5  F7            00A5 FD read
 -4  B7 START-LOOP 00A3 2D       DEC L
 -3  B7            00A4 20FD     JR NZ,START-LOOP
 -1  B7            00A6 25       DEC H
  0  B7            00A7 20FA     JR NZ,START-LOOP
  2  B7            00A9 C9       RET
  3  F7            18FC 90 read
  4  F7            18FD 00 read
  5  B7            0090 C38800   JP 88
  8  B7            0088 D379     OUT (LEDS),A
  A  5F            8079 80 out
  B  B7            008A 0F       RRCA
  C  B7            008B 14       INC D
  D  B7            008C 00       NOP
  E  B7            008D CDA000   CALL DELAY-SUB
 11  D7            18FD 00 write
 12  D7            18FC 90 write
 13  B7 DELAY-SUB  00A0 21FF40   LD HL,40FF
 16  B7 START-LOOP 00A3 2D       DEC L
```

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 $A0_{HEX}$; 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 $A0_{HEX}$ 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_{HEX} cycles were executed between addresses $A0_{HEX}$ and $A6_{HEX}$. 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 $3+(3\times FF_{HEX})=300_{HEX}$; 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 $40\times303_{HEX}$. 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_{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 single-stepping through programs, or pouring over program listings looking for errors, you are probably wasting a lot of time. The ideal debugging technique is to use analyzer triggers or breakpoints or both, as appropriate. A development system that includes an integrated analyzer-emulator renders both techniques available to you at all times. **EDN**

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 Logisticon 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 CommuniTree 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

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 algrorithm 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. Peatman describes the support for families of microprocessors.)

---

* Figures and Illustrations in preparation.

ORION

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. **ORION**

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)

(continued from page 1)

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>ULZ80  42F  LRUN
```

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

You also could have entered the comands separately:

```
C>ULZ80  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. **ORION**

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. **ORION**

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 Disassembler/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.

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

SYSTEM SOFTWARE	PREVIOUS RELEASE	LATEST RELEASE
UDL MSDOS	2.42	UDL2.5 (Nov. 25, 1986)
UniLab II	3.2	UNI3.3 (April 1987)
OptiLab	3.21	OPT3.3 (April 1987)

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. ORION

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 Emulaton 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 68POV07 6809E 6809 68000 80188 8031 80031 8032 8048/P 8050 8051P 87P50 80C51VS 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. ORION

ORION
Instruments

News Briefs:

- **NEW and HOT!** Full support for 68HC11 now available in stock! Save thousands over the competition!
- Z80 support now includes Alternate Register Display/Alter feature. See inside for update info.
- True single stepping, breakpoint on trigger condition, and auto-breakpoint features now implemented for 6805, 8048, 8051, Z8, and SUPER 8 processors.
- NEC latest processor, the 78310/12 is now supported by Orion.

ORION
Instruments

702 Marshall Street
Redwood City, CA 94063

Address Correction Requested

First Class
U.S. Postage
PAID
Redwood City, CA
Permit No. 266

First Class Mail

Newsletter for Users of Orion UniLab and UDL

ORION

TECHNICAL QUESTIONS & ANSWERS

June 12, 1987

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/CGA/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?

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 trigggr 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.

TECHNICAL QUESTIONS & ANSWERS

June 12, 1987

Hook-up or Connecting to the Target

- Connection	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 micro-controller 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.

TECHNICAL QUESTIONS & ANSWERS

June 12, 1987

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 gris 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.

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 trouble shooting 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.

TECHNICAL QUESTIONS & ANSWERS

June 12, 1987

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.

TECHNICAL QUESTIONS & ANSWERS

June 12, 1987

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 give 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 its 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!

THIS SHEET REPLACES CABLE WIRING INSTRUCTIONS OF THE INSTALLATION CHAPTER

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

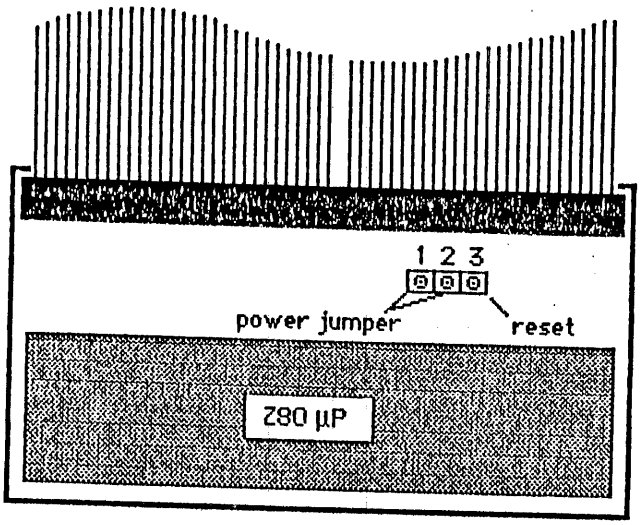
With the module connection, the UniLab still runs all the target code on your microprocessor. The module just moves your processor a centimeter or two away from the board.

2) Plug module into microprocessor socket

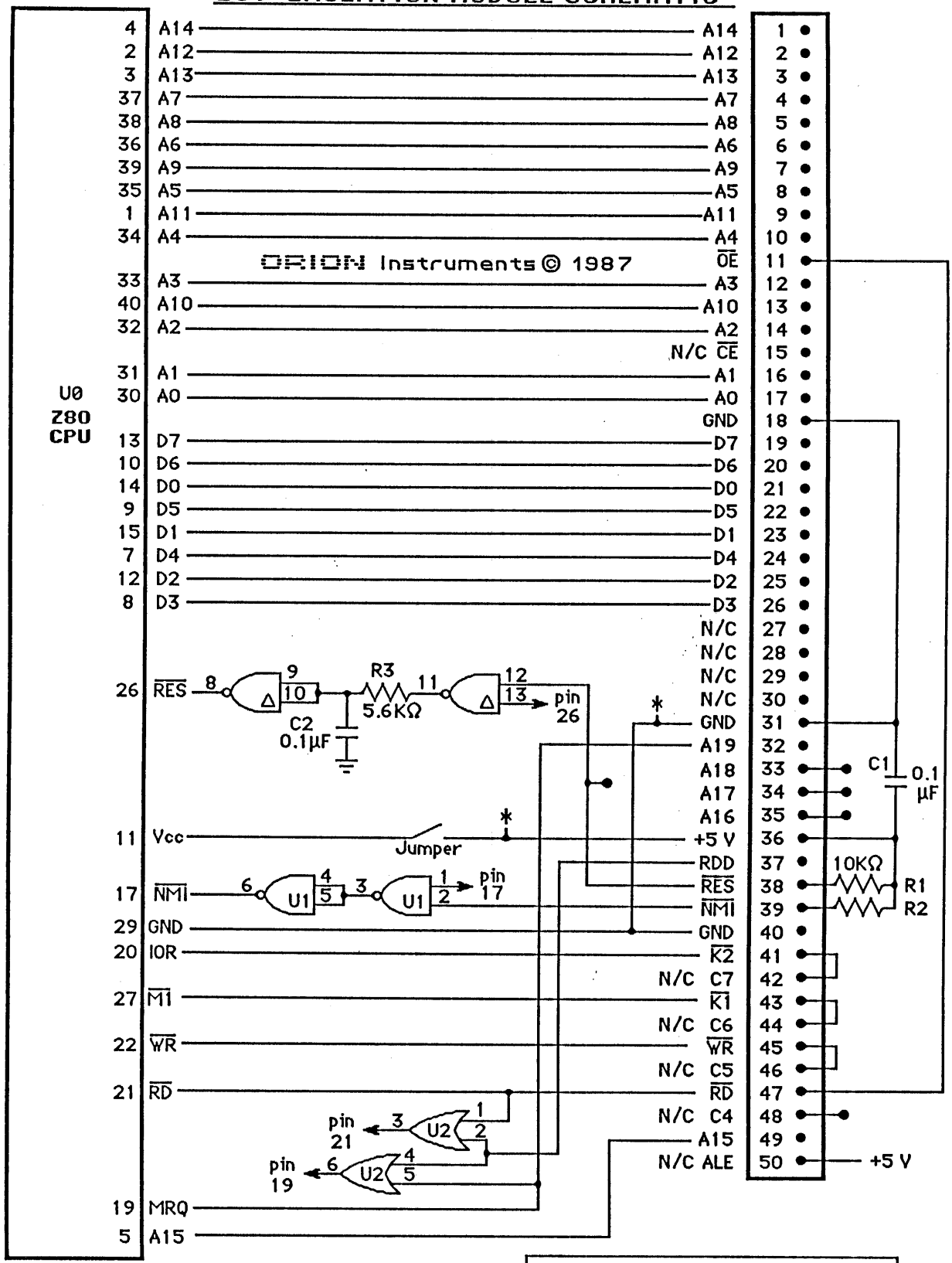
Put the module into the microprocessor socket on the target board. Double check that you have oriented the module correctly, so that pin one of the processor on the module lines up with pin one of the processor socket.

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.



Z80 EMULATION MODULE SCHEMATIC



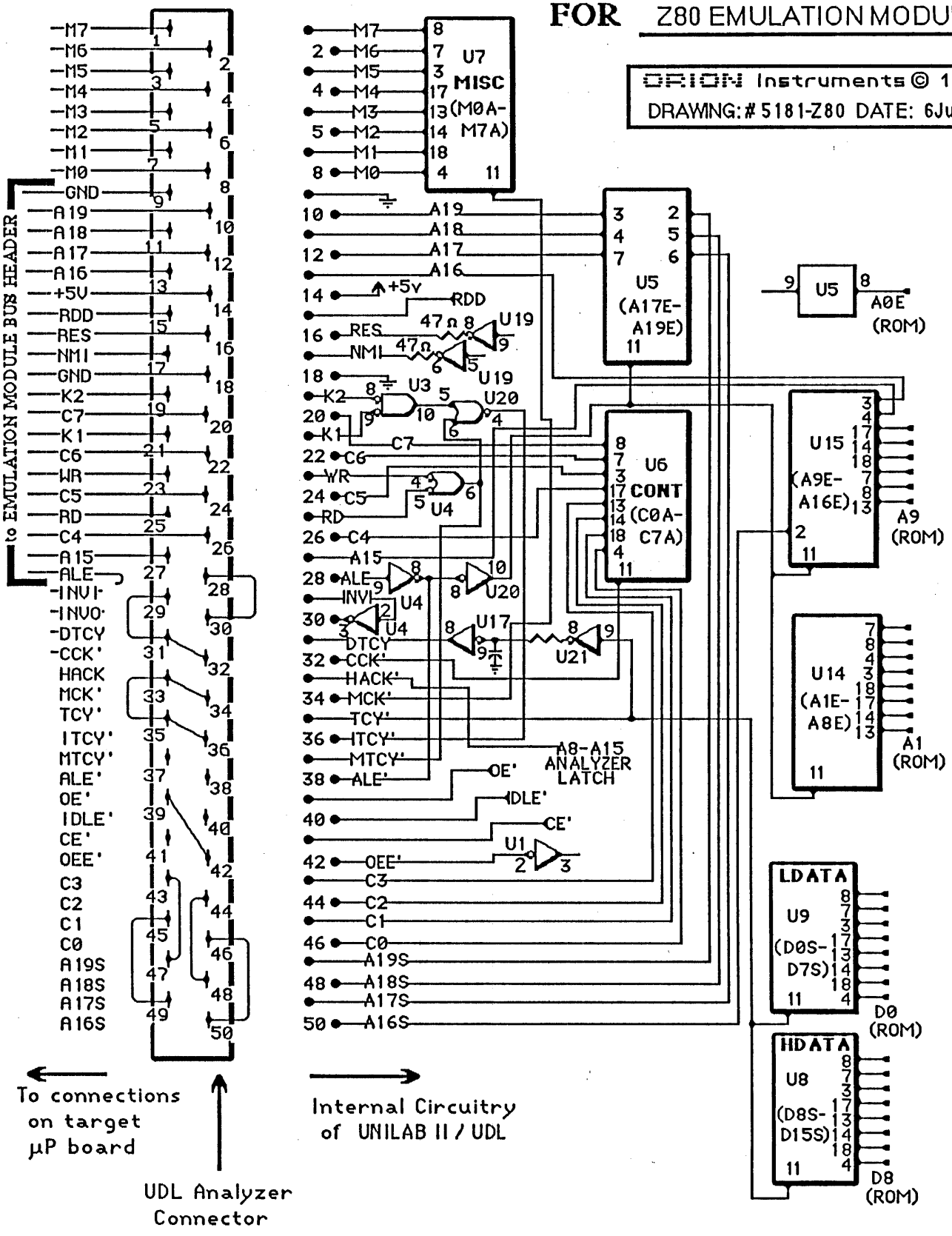
U1 HCT132 Schmitt Trigger NAND gate.
U2 ALS32 OR gate.
* Connected to logic IC's.

Title: EMZ80 Schematic		Rev: B	
DWG# 5126M	drawn by:PAB	appv'd:	ORION INSTRUMENTS, INC
DATE: 6July87			

EMZ80.02

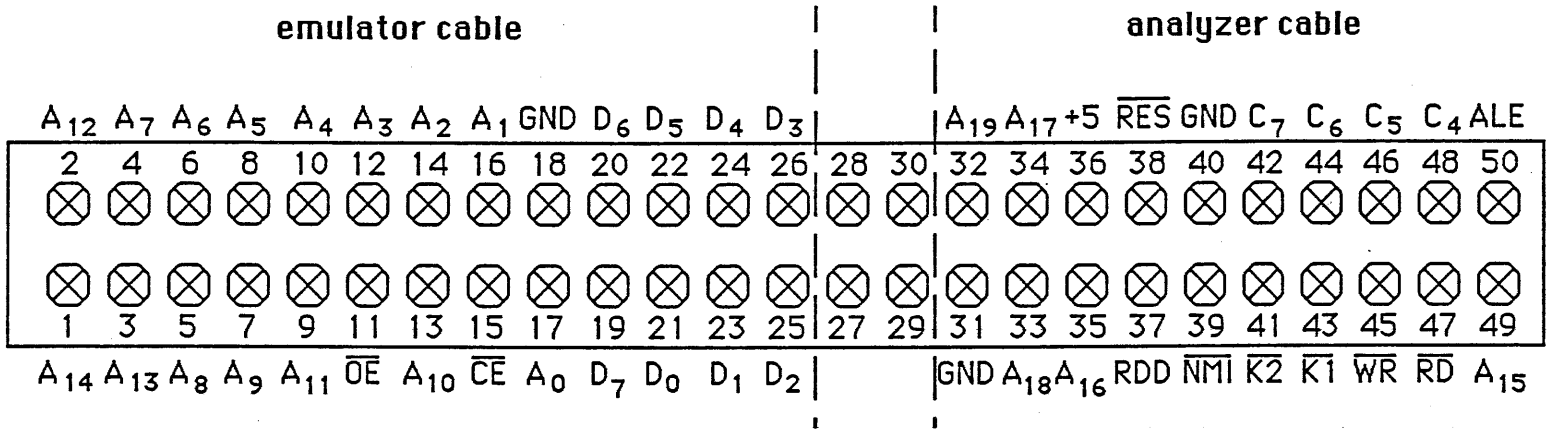
EM8-EC ANALYZER CABLE CONFIGURATION FOR Z80 EMULATION MODULE

ORION Instruments © 1987
DRAWING: # 5181-Z80 DATE: 6July87

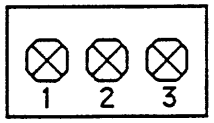


EMZ80.05 Rev.B

EMZ80 EMULATION MODULE HEADERS



50-PIN BUS HEADER

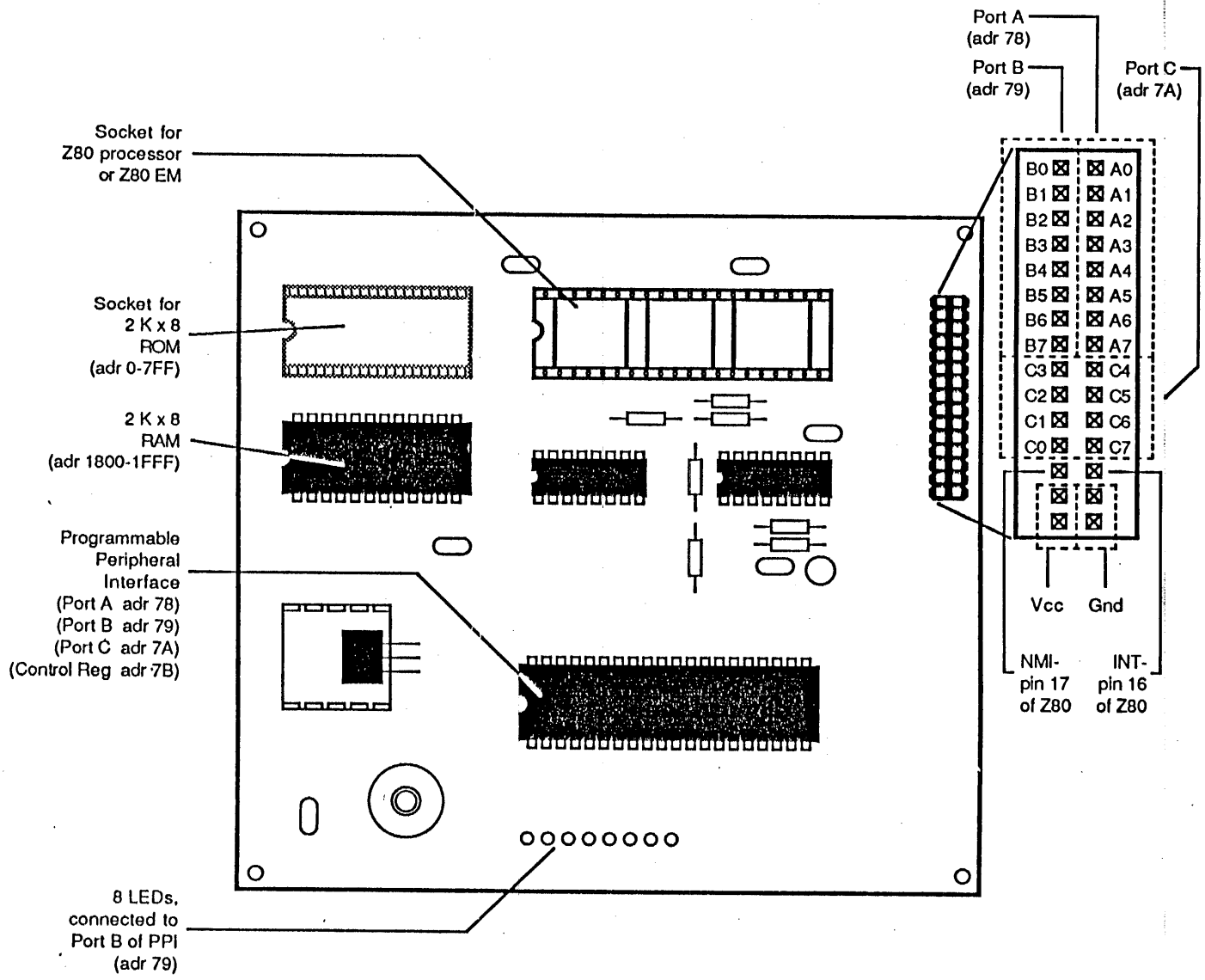


3-PIN JUMPER HEADER

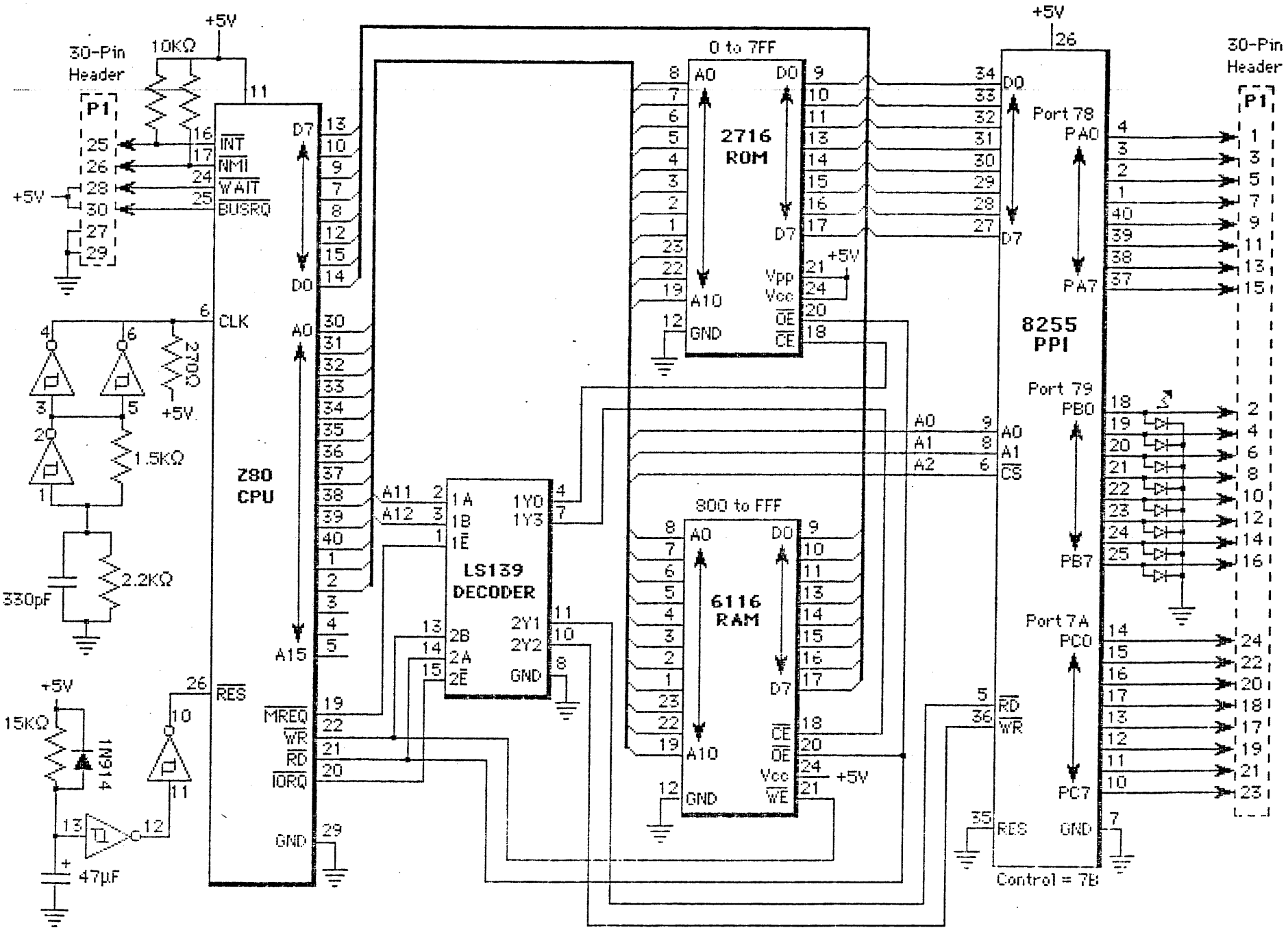
Title: EMZ80 Headers		Rev: B
DWG#: 5188	drawn by: PAB	appv'd:
Date: 6July87	ORION INSTRUMENTS, INC	

EMZ80.06

Z80 MicroTarget™



Consult the appendices of the Personality Pak documentation for target board schematics



REV D 9DEC86

Z80 TARGET SYSTEM

ORION Instruments © 1986

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 3. 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.

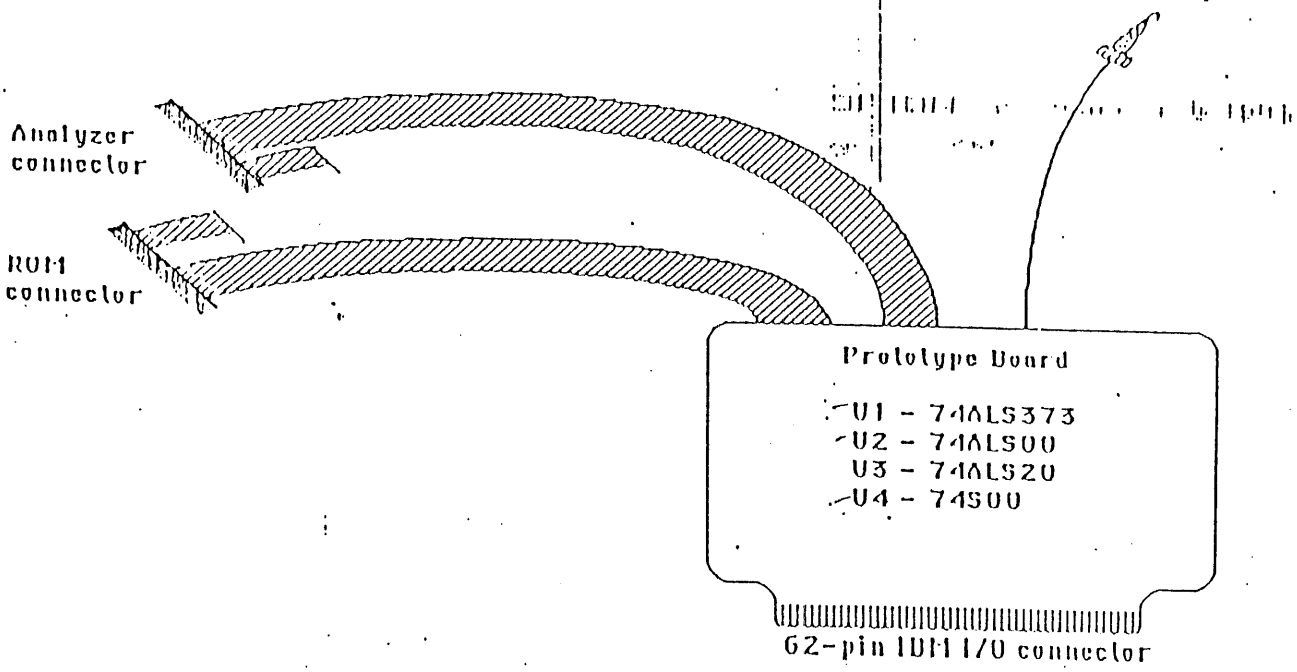
HOOKING UP TO THE TARGET IBM PC

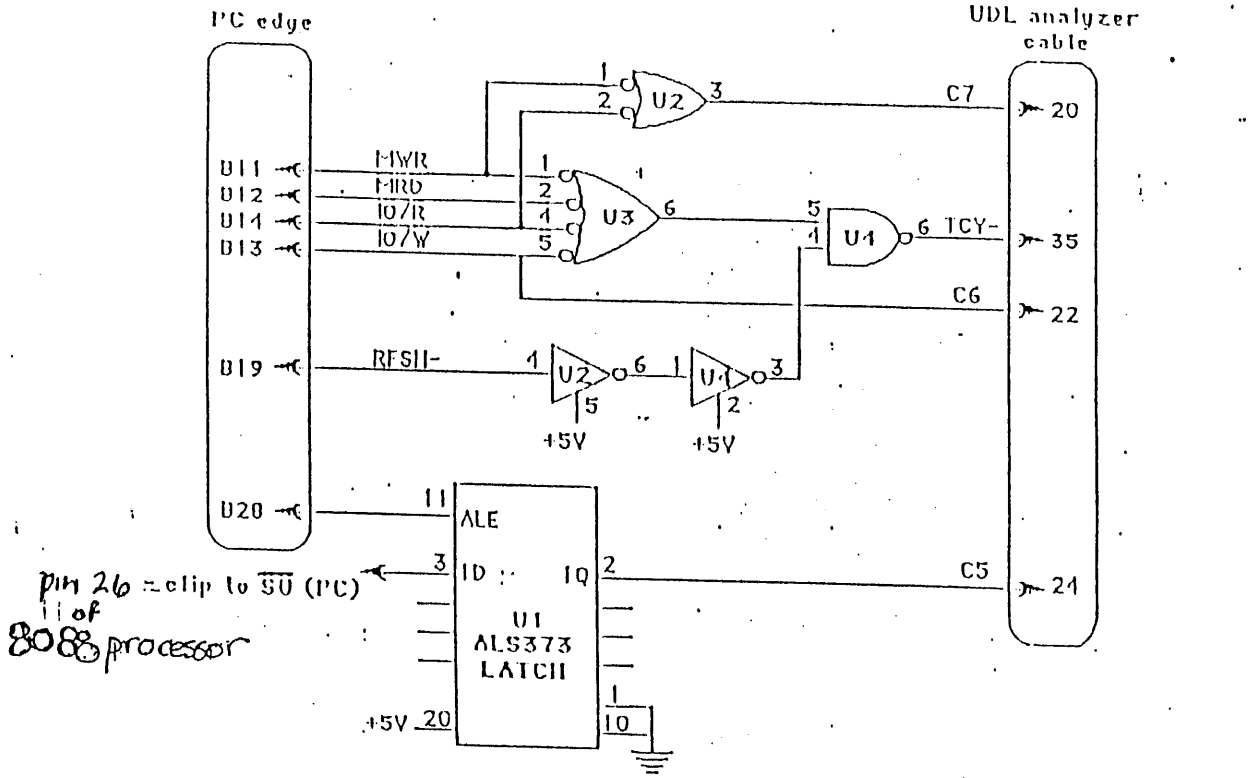
Once the filter circuit is made and the analyzer cable has been modified, both cables can be hooked up to the card which can then be inserted in the IBM PC. Cable-to-card connections are shown on pages 5 and 6.

The analyzer cable has connections going to the 62-pin edge connector, the filter circuit (denoted by 'CKT'), 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.

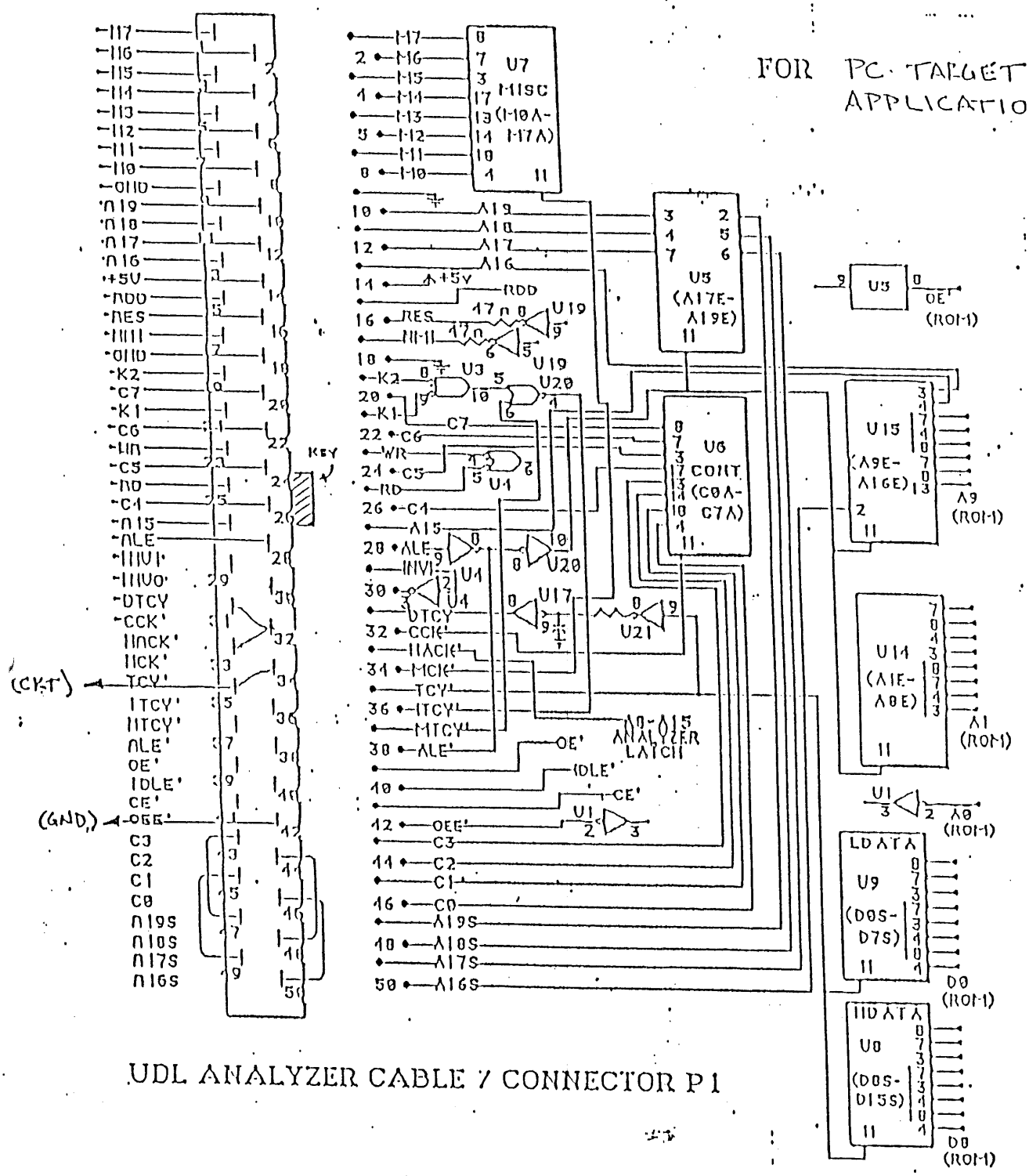
UDL-PC INTERFACE CIRCUIT





CABLE MODIFICATION

FOR PC TARGET APPLICATION



UDL ANALYZER CABLE 7 CONNECTOR P1

REV. 1
6NOV86

IBM-PC/UDL ANALYZER CABLE CONNECTIONS

62-pin edge connector	ANALYZER connector signals Pin #	Signal
-	1	M7
-	2	M6
-	3	M5
-	4	M4
-	5	M3
-	6	M2
-	7	M1
-	8	M0
B1	9	GND
A12	10	A19E
A13	11	A18E
A14	12	A17E
A15	13	A16E
-	14	+5V
-	15	RDD
PC RES-	16	RES- (to pin 11 of 8284 clock gen.)
8088 NMI	17	NMI- (to pin 17 of 8088 µP)
GND	18	GND
GND	19	K2-
(CKT)	20	C7
B12	21	K1-
(CKT)	22	C6
B11	23	WR-
(CKT)	24	C5
B13	25	RD-
A11	26	C4
A16	27	A15E
+5	28	ALE
-	29	INV1
-	30	INV0
-	31	DTCY
-	32	CCK'
-	33	HACK'
-	34	MCK'
(CKT)	35	TCY'
-	36	ITCY'
-	37	MTCY'
-	38	ALE'
-	39	OE'
-	40	IDLE'
-	41	CE'
-	42	OEE'
-	43	C3
-	44	C2
-	45	C1
-	46	C0
-	47	A19S
-	48	A18S
-	49	A17S
-	50	A16S

IBM-PC/UDL ROM CABLE CONNECTIONS

62-pin edge connector	ROM connector signals Pin #	Signal
A17	1	A14E
A19	2	A12E
A18	3	A13E
A24	4	A7E
A23	5	A8E
A25	6	A6E
A22	7	A9E
A26	8	A5E
A20	9	A11E
A27	10	A4E
B12	11	OE'
A28	12	A3E
A21	13	A10E
A29	14	A2E
-	15	CE'
A30	16	A1E
A31	17	A0E
B31	18	GND
A2	19	D7E
A3	20	D6E
A9	21	D0E
A4	22	D5E
A8	23	D1E
A5	24	D4E
A7	25	D2E
A6	26	D3E
-	27	D11E
-	28	D10E
-	29	D12E
-	30	D9E
-	31	D13E
-	32	D8E
-	33	D14E
-	34	D15E
-	35	D7A
-	36	D6A
-	37	D0A
-	38	D5A
-	39	D1A
-	40	D4A
-	41	D2A
-	42	D3A
-	43	D15A
-	44	D14A
-	45	D8A
-	46	D13A
-	47	D9A
-	48	D12A
-	49	D10A
-	50	D11A

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:

```
EMCLR
 Emulator Memory Enable Status:
No Memory Enabled ok

PCPATCH ok
SEG' 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

SEG' 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, FFF0. 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 E004 hex for this particular IBM PC clone in the examples that follow:

```
resetting
(from top of buffer)
 cyc CONT ADR DATA                          MONITR  MISC
   0  4F FFF0 EA04E000F0   JMP E004,F000:    11111111 00000000
   6  4F E004 FC           CLD               11111111 11111100
   7  4F E005 FA           CLI               11111111 11111010
   8  4F E006 B0Z0         MOV AL,0          11111111 00000000
   A  4F E008 E6A0         OUT A0,AL         11111111 10100000
   C  4F E00A E6D3         OUT D3,AL         11111111 11100110
   D   2   A0 00 outd                        11111111 10000011
   F  4F E00C BA0003       MOV DX,300        11111111 10111010
  10   2   D3 00 outd                        11111111 00000011
```

```
13  4F E00F EE          OUT DX,AL               11111111 11101110
14  4F E010 FEC0        INC AL                  11111111 11111110
15   2  300 00  outd                            11111111 11000000
17  4F E012 B200        MOV DL,D0               11111111 10111000
19  4F E014 EE          OUT DX,AL               11111111 11101110
1A  4F E015 E2FE        LOOP E015               11111111 11100010
1D   2  300 01  outd                            11111111 11100010
1E  4F E015 E2FE        LOOP E015               11111111 11100010
21  4F E015 E2FE        LOOP E015               11111111 11100010
24  4F E015 E2FE        LOOP E015               11111111 11100010
27  4F E015 E2FE        LOOP E015               11111111 11100010
2A  4F E015 E2FE        LOOP E015               11111111 11100010
Pg Dn or ^Y (trace resume)  Home (top) n IN (from step n) T (from n=-5 )
```

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:

```
DASM  ok
resetting
(from top of buffer)
 cyN CONT ADR DATA ADATA  MISC
   0  4F FFF0 EA 11111111 11101010
   1  4F FFF1 04 11111111 00000100
   2  4F FFF2 E0 11111111 11100000
   3  4F FFF3 00 11111111 00000000
   4  4F FFF4 F0 11111111 11110000
   5  4F FFF5 00 11111111 00000000
   6  4F E004 FC 11111111 11111100
   7  4F E005 FA 11111111 11111010
   8  4F E006 B0 11111111 10110000
   9  4F E007 00 11111111 00000000
   A  4F E008 E6 11111111 11100110
   B  4F E009 A0 11111111 10100000
   C  4F E00A E6 11111111 11100110
   D   2 000B 00 11111111 00000000
   E  4F E00D 03 11111111 10000011
   F  4F E00C BA 11111111 10111010
  10   2 0003 00 11111111 00000000
  11  4F E00D B0 11111111 10110000
  12  4F E00E 03 11111111 00000011
  13  4F E00F EE 11111111 11101110
  14  4F E010 FE 11111111 11111110
Pg Dn or ^Y (trace resume)  Home (top) n IN (from step n) T (from n=-5 )
```

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:

```
FFF0 I DM
FFF0  EA04E000F0    JMP E004,F000

E004 I5 DM
E004  FC            CLD
E005  FA            CLI
E006  B000          MOV AL,0
E008  E608          OUT 08,AL
E00A  E603          OUT 03,AL
E00C  BA0003        MOV DX,300
E00F  EE            OUT DX,AL
E010  FEC0          INC AL
E012  B200          MOV DL,00
E014  EE            OUT DX,AL
E015  E2FE          LOOP E015
E017  E2FE          LOOP E017
E019  B099          MOV AL,99
E01B  E663          OUT 63,AL
E01D  0CC8          MOV AX,C9
```

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 8088 based target system, the user program must install the required pointer at locations 00008C-00008F as described in the 8088/8086 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 FFC0 hex:

```
FFCD 15 DN
FFCD  C706060000F0 MOV [6],F000
FFD3  C706000001FF MOV [0],FFD1
FFD9  C706040000F0 MOV [4],F000
FFDF  C7062C0001FF MOV [2C],FFD1
FFE5  C706 0E0000F0 MOV [E],F000
FFEB  EA04E000F0   JMP E004,F000:
FFF0  C706040001FF MOV [4],FFD1
FFFG  EACDFF00F0   JMP FFCD,F000:
```

So now when a STARTUP is issued the trace will appear as follows:

```
STARTUP
resetting
(from top of buffer)
 cyN CONT ADR DATA                                  MDATA  MISC
   0  4F FFF0 C706040001FF MOV [4],FFD1             11111111 11101010
   7  C0    4 D1 write interrupt N1                 11111111 10110001
   8  C0    5 FF write interrupt N1                 11111111 11111111
   6  4F FFFG EACDFF00F0   JMP FFCD,F000:           11111111 00000000
   E  4F FFCD C706060000F0 MOV [6],F000            11111111 11000111
  15  C0    6 00 write interrupt N1                 11111111 00000000
  16  C0    7 F0 write interrupt N1                 11111111 11110000
  14  4F FFD3 C706000001FF MOV [0],FFD1             11111111 11000111
  1D  C0    8 D1 write interrupt N2                 11111111 10110001
  1E  C0    9 FF write interrupt N2                 11111111 11111111
  1C  4F FFD9 C706040000F0 MOV [4],F000            11111111 11000111
  25  C0    A 00 write interrupt N2                 11111111 00000000
  26  C0    B F0 write interrupt N2                 11111111 11110000
  24  4F FFDF C7062C0001FF MOV [2C],FFD1            11111111 11000111
  2D  C0    C D1 write interrupt N3                 11111111 10110001
  2E  C0    D FF write interrupt N3                 11111111 11111111
  2C  4F FFE5 C706 0E0000F0 MOV [E],F000            11111111 11101010
  35  C0    E 00 write interrupt N3                 11111111 00000000
  36  C0    F F0 write interrupt N3                 11111111 11110000
  34  4F FFEB EA04E000F0   JMP E004,F000:           11111111 11000111
  3C  4F E004 FC           CLD                     11111111 11111100
 Pg Dn or ^Y (trace resume)  Home (top) n IN (from step n) T (from n=-5 )
```

Now one can gain debug control as follows:

```
RESET E01D RD resetting

IP=E01D F=F002(---oditsz-a-p-c)  AX=6099 BX=0024 CX=0200 DX=0300
CS=F000 DS=0000 SS=0000 ES=0000  SP=0054 BP=FFF2 DI=15CD SI=E054
E01D  E663        OUT 63,AL                 (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
IP=E010 F=F002(---oditsz-a-p-c)  AX=6A39 BX=0024 CX=0000 DX=0300
CS=F000 DS=0000 SS=0000 ES=0000  SP=0054 BP=FFF2 DI=15CD SI=E054
E010  8CC0          MOV AX,CS                (next step)  ok
```

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

```
E044 RD
IP=E044 F=F002(---oditsz-a-p-c)  AX=1103 BX=0024 CX=7D00 DX=0300
CS=F000 DS=F000 SS=0030 ES=F000  SP=03FE BP=FFF2 DI=15CD SI=E054
E044  E2FE          LOOP E044                (next step)  ok

NMI
IP=E044 F=F102(---oditsz-a-p-c)  AX=1103 BX=0024 CX=7CFF DX=0300
CS=F000 DS=F000 SS=0030 ES=F000  SP=03FE BP=FFF2 DI=15CD SI=E054
E044  E2FE          LOOP E044                (next step)  ok

N
IP=E046 F=F002(---oditsz-a-p-c)  AX=1103 BX=0024 CX=0000 DX=0300
CS=F000 DS=F000 SS=0030 ES=F000  SP=03FE BP=FFF2 DI=15CD SI=E054
E046  32C0          XOR AL,AL                (next step)  ok

E050 RD
IP=E050 F=F046(---oditsz-a-p-c)  AX=0000 BX=0D70 CX=0000 DX=0300
CS=F000 DS=F000 SS=0030 ES=F000  SP=03FE BP=03E6 DI=04E6 SI=E060
E050  EB0E          JMP E060                 (next step)  ok

NMI
IP=E060 F=F146(---oditsz-a-P-c)  AX=0000 BX=0D70 CX=0000 DX=0300
CS=F000 DS=F000 SS=0030 ES=F000  SP=03FE BP=03E6 DI=04E6 SI=E060
E060  B054          MOV AL,54                (next 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 COM1 serial I/O port (3FD hex) for a UniLab connection that is not there:

```
NUM?
cyN CONT ADR DATA                                    NUMIA  MISC
-04  40 F505 FD          STD                         11111111 11111101
-03  40 F506 03EC        ADD BP,SP                   11111111 11101100
-01  40 F500 F6C001      TEST AL,1                   11111111 11110110
-00  E0  3FD 60   inp                                11111111 10111010
-99  40 F504 BAFD03      MOV DX,3FD                  11111111 00000011
-97  40 F507 EC          IN AL,DX                    11111111 11110110
-95  E0  3FD 60   inp                                11111111 11101100
-96  40 F500 F6C001      TEST AL,1                   11111111 10111010
-0F  40 F504 BAFD03      MOV DX,3FD                  11111111 00000011
-0C  40 F507 EC          IN AL,DX                    11111111 11110110
-0A  E0  3FD 60   inp                                11111111 11101100
-08  40 F500 F6C001      TEST AL,1                   11111111 10111010
-04  40 F504 BAFD03      MOV DX,3FD                  11111111 00000011
-01  40 F507 EC          IN AL,DX                    11111111 11110110
-7F  E0  3FD 60   inp                                11111111 11101100
-7D  40 F500 F6C001      TEST AL,1                   11111111 10111010
-79  40 F504 BAFD03      MOV DX,3FD                  11111111 00000011
-76  40 F507 EC          IN AL,DX                    11111111 11110110
-74  E0  3FD 60   inp                                11111111 11101100
-75  40 F500 F6C001      TEST AL,1                   11111111 10111010
-6E  40 F504 BAFD03      MOV DX,3FD                  11111111 00000011
 Pg Dn or ^Y (trace resume)  Home (top) n IN (from step n) T (from n=-5 ) ok
```

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 Danna, Applications Engineering

INSTRUCTIONS FOR 4617, 4617-1, 4617-3 IBM AT

1.0 BOARD DESCRIPTION

The 4617 plugboards for the IBM AT computer have a .1" x .1" pattern for unrestricted component placement. Power and ground buses surround the grid pattern on both the component and solder side of the board. The power and ground buses are connected to +5V 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 sizes 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.

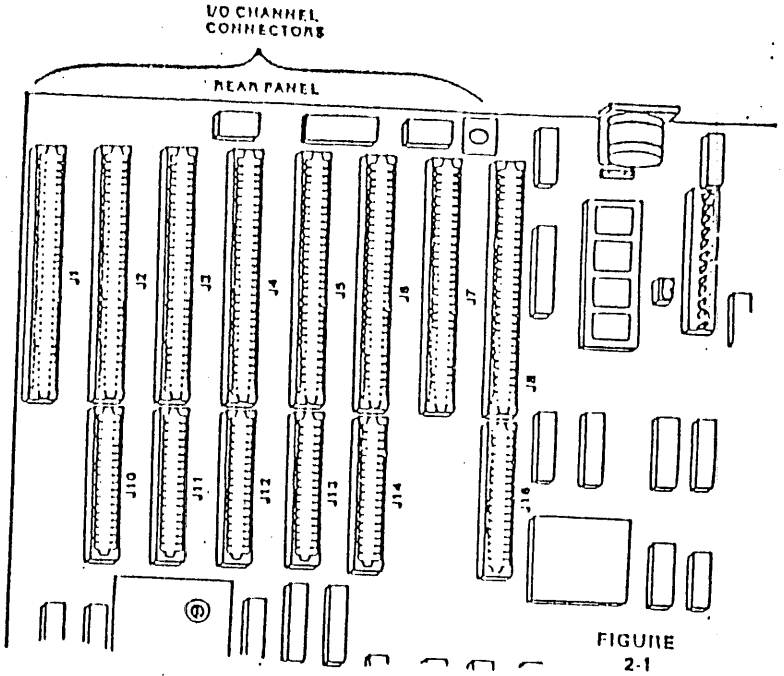


FIGURE 2-1

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 RDY
A11	AEN
A12	SA19
A13	SA18
A14	SA17
A15	SA16
A16	SA15
A17	SA14
A18	SA13
A19	SA12
A20	SA11
A21	SA10
A22	SA9
A23	SA8
A24	SA7
A25	SA6
A26	SA5
A27	SA4
A28	SA3
A29	SA2
A30	SA1
A31	SA0

I/O Pin	Signal Name
B1	GND
B2	RESET DRV
B3	+5 Vdc
B4	IRQ 9
B5	-5 Vdc
B6	DRQ2
B7	-12 Vdc
B8	OWS
B9	+12 Vdc
B10	GND
B11	-SMEMW
B12	-SMEMR
B13	-IOW
B14	-IOR
B15	-DACK3
B16	DRQ3
B17	-DACK1
B18	DRQ1
B19	-Refresh
B20	CLK
B21	IRQ7
B22	IRQ6
B23	IRQ5
B24	IRQ4
B25	IRQ3
B26	-DACK2
B27	T/C
B28	BALE
B29	+5 Vdc
B30	OSC
B31	GND

I/O Pin	Signal Name
C1	SBHE
C2	LA23
C3	LA22
C4	LA21
C5	LA20
C6	LA19
C7	LA18
C8	LA17
C9	-MEMR
C10	-MEMW
C11	SD08
C12	SD09
C13	SD10
C14	SD11
C15	SD12
C16	SD13
C17	SD14
C18	SD15

I/O Pin	Signal Name
D1	-MEM CS16
D2	-I/O CS16
D3	IRQ10
D4	IRQ11
D5	IRQ12
D6	IRQ15
D7	IRQ14
D8	-DACK0
D9	DRQ0
D10	-DACK5
D11	DRQ5
D12	-DACK6
D13	DRQ6
D14	-DACK7
D15	DRQ7
D16	+5 Vdc
D17	-MASTER
D18	GND

FIGURE 2-2

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. 3-1. The D-subminiature 9-pin AMP no. 745113-2; 25-pin AMP no. 745114-2; or 37-pin AMP no. 745116-2 or equivalent connector may be used if their dimensions match fig. 3-2 to fit the bracket and computer.

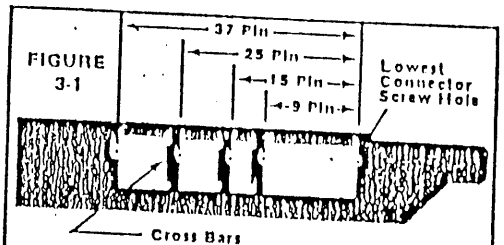


FIGURE 3-1

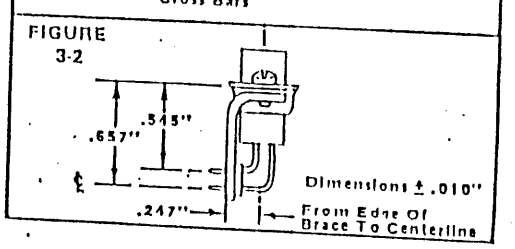


FIGURE 3-2

a registered trademark of International Business Machines

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 the Vendor 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 appreviations indicate the sources listed on the following pages. We would appreciate any user feedback so that we can keep this list current.

PROCESSOR	ASSEMBLER SUPPLIERS	COMPILER SUPPLIERS
1802/5	25 AV MI WE AA EN RE SD UW	
6301	25 AV CY EN LO RE UW	IT AR
6305	AV MT RE	
6502	25 AV LO MI EN RE SD	MX
6800/2/8	25 AA AV MI DM EN LO QU RE SD UW	
6801/3	25 AA AV DM EN LO MI RE SD UW	AR IT
6805	25 AV DM EN LO MI RE SD UW	IT
6809	25 AV DM EN LO MI RE SD UW	IT
68000	25 AV EN LO QU RE SD UN UW	MX IT MT LA UW
68HC11	AV CY LO RE SD UW	AR IT
NSC800	25 EN RE	MT
8048-50/41	25 AA AV CY LO MI RE SD UN	
8051/31	25 AA AV CY LO MI RE SD UN US UW	AR MC
8080	EN LO UN	
8085	25 AA AV CY EN LO MI RE SD UW	MT
8086/8	25 AV CY EN SD SP SW UN UW	MX MS MT LA
8096	25 AA AV CY LO UN	
Z-8	25 AV AA CY EN LO RE SD UW	
Z-80 64180	25 AV AA EN LO MT RE SD US UW	MX KY LA, AR
Z-8000	25 EN RE UN	

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

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 refered 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.

UW 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.

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

WW 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.

o: Orion
rom GOGET
Re: UniLab 3.31 Files
Date 24 May 87

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:

```
COMMAND   COM     23210   3-07-85   1:43p
AUTOEXEC  BAT       128   5-23-86   1:43p
CONFIG    SYS       128   5-06-86   8:03a

INSTALL   BAT      2176   4-09-86  11:28a

EDIT33    VIR      8533   5-19-87   8:56a
UTIL33    VIR      4450   5-19-87   8:57a

L1B331    VIR      8578   5-22-87  10:46a
L2B331    VIR      8873   5-22-87  10:46a
MEN331    VIR      6683   5-22-87  10:47a
UL331     VIR      3276   5-22-87  10:47a
OLASM331  VIR      6467   5-22-87  10:48a

ULZ80     EXE     46176   5-24-87   3:05p
ULZ80     OPR      2962   5-24-87   3:05p
ULZ80     MCR     14602   5-22-87   2:46p

UNI331    OVL     39744   5-22-87  10:48a
UNI331    OPR      1130   5-22-87  10:49a
UNI331    MCR      8633   5-22-87  10:48a

UNILAB    SCR     41984   5-22-87  10:58a

Z80       TBL      5155   3-16-87   9:09a

DEMO2     BIN       384   2-18-84  12:19a
DEMO3     BIN      2048   2-17-84  11:08a
DEMO4     BIN       676   3-12-86  11:56a

DEMO      SYM       128   3-06-86   8:29a
DEMO4     SYM       684   3-12-86   2:03p

DEMO4     C        2699   3-11-86  10:56a

TESTZ80   BIN        45   9-26-86  11:13a
DEMOZ80   TRC      1024   9-26-86  11:01a

DO        BAT        40   3-30-87   4:07p
FORMAT    COM      9398   3-07-85   1:43p
DIRSIZE   EXE      6972   9-15-86   9:02a
        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.

he 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.

DEMO.SYM - Symbols for sales DEMO2,3.BIN.
DEMO4.SYM - 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 <u>before</u> value 01 goes out to port.
type:	**N**	The program single-steps.
type:	**LP**	Program is released, runs until just after 80 goes out ot port 79.
repeat:	**LP**	Notice LED stepping along. This is it for a <u>basic</u> 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 <u>are</u> 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 <u>second</u> 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:	**F5**	Gets ASC command. Note equivalent command.
ress :	**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  EDB0      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  C38000    JP 80
```

```
; 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  CDA000    CALL A0
0090  C38800    JP 88
```

```
; 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   21FF40    LD HL,40FF        ; The delay value.
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   D379      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 initializaion/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 availiable 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 "1AFTER 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 occrence 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 M!". The fact that the BUG has been fixed is shown by re-typing 8 "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.

```
ok
resetting
(from top of buffer)
 cy# CONT ADR DATA                                    HDATA   MISC
   0  B7 0000 31FE18   LD SP,18FE                   11111111 11111111
   3  B7 0003 DD213412 LD IX,1234                   11111111 11111111
   7  B7 0007 FD217856 LD IY,5678                   11111111 11111111
   B  B7 000B 210019   LD HL,1900                   11111111 11111111
   E  B7 000E 34       INC (HL)                     11111111 11111111
   F  F7 1900 48 read                               11111111 11111111
  10  D7 1900 49 write                              11111111 11111111
  11  B7 000F 34       INC (HL)                     11111111 11111111
  12  F7 1900 49 read                               11111111 11111111
  13  D7 1900 4A write                              11111111 11111111
  14  B7 0010 14       INC D                        11111111 11111111
  15  B7 0011 14       INC D                        11111111 11111111
  16  B7 0012 14       INC D                        11111111 11111111
  17  B7 0013 14       INC D                        11111111 11111111
  18  B7 0014 210001   LD HL,100                    11111111 11111111
  1B  B7 0017 012000   LD BC,20                     11111111 11111111
  1E  B7 001A 110018   LD DE,1800                   11111111 11111111
  21  B7 001D EDB0     LDIR                         11111111 11111111
  23  F7 0100 16 read                               11111111 11111111
  24  D7 1800 16 write                              11111111 11111111
  25  B7 001D EDB0     LDIR                         11111111 11111111
  F5  or TR (trace resume)  TT (top) n TN (from step n)  T (from n=-5 )

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

AFTER A8 ADR S resetting

```
Not done till delay count = 0. Any key aborts. TD dumps trace buffer.
 cy# CONT ADR DATA                          HDATA    MISC
 F01   FF FFFF FF read                     11111111 11111111
 F02   B7 00A8 C9        RET               11111111 11111111
 F03   F7 18FA 90 read                     11111111 11111111
 F04   B7 00A8 C9        RET               11111111 11111111
 F05   F7 18F8 90 read                     11111111 11111111
 F06   B7 00A8 C9        RET               11111111 11111111
 F07   F7 18F6 90 read                     11111111 11111111
 F08   B7 00A8 C9        RET               11111111 11111111
 F09   F7 18F4 90 read                     11111111 11111111
 F0A   B7 00A8 C9        RET               11111111 11111111
 F0B   F7 18F2 90 read                     11111111 11111111
 F0C   B7 00A8 C9        RET               11111111 11111111
 F0D   F7 18F0 90 read                     11111111 11111111
 F0E   B7 00A8 C9        RET               11111111 11111111
 F0F   F7 18EE 90 read                     11111111 11111111
 F10   B7 00A8 C9        RET               11111111 11111111
 F11   F7 18EC 90 read                     11111111 11111111
 F12   B7 00A8 C9        RET               11111111 11111111
 F13   F7 18EA 90 read                     11111111 11111111
 F14   B7 00A8 C9        RET               11111111 11111111
 F15   F7 18E8 90 read                     11111111 11111111
  F5   or TR (trace resume)  TT (top) n TN (from step n)  T (from n=-5 )
```

A8 AS resetting

```
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 18FA 90 read                     11111111 11111111
   2   F7 18FB 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 D5       PUSH DE            11111111 11111111
   C   D7 18FB 01 write                    11111111 11111111
   D   D7 18FA 20 write                    11111111 11111111
   E   B7 008D CDA000   CALL A0            11111111 11111111
  11   D7 18F9 00 write                    11111111 11111111
  12   D7 18F8 90 write                    11111111 11111111
  13   B7 00A0 62       LD H,D             11111111 11111111
  14   B7 00A1 6B       LD L,E             11111111 11111111
  15   B7 00A2 2D       DEC L              11111111 11111111
  16   B7 00A3 20FD     JR NZ,A2           11111111 11111111
  F5   or TR (trace resume)  TT (top) n TN (from step n)  T (from n=-5 )
```

```
0 8C MI  ok
A8 AS resetting

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 )
```

```
ONLY NOT A0 TO B0 ADR AFTER 80 ADR S resetting

Not done till delay count = 0. Any key aborts. TD dumps trace buffer.
 F02  F7 0081 FF read                          11111111 11111111
 F02  B7 0082 3E80      LD A,80                11111111 11111111
 F04  B7 0084 D37B      OUT (7B),A             11111111 11111111
 F06  5F 807B 80 out                           11111111 11111111
 F07  B7 0086 3E01      LD A,01                11111111 11111111
 F09  B7 0088 D379      OUT (79),A             11111111 11111111
 F0B  5F 0179 01 out                           11111111 11111111
 F0C  B7 008A 0F        RRCA                   11111111 11111111
 F0D  B7 008B 14        INC D                  11111111 11111111
 F0E  B7 008C 00        NOP                    11111111 11111111
 F0F  B7 008D CDA000    CALL A0                11111111 11111111
 F12  D7 18FD 00 write                         11111111 11111111
 F13  D7 18FC 90 write                         11111111 11111111
 F14  F7 18FC 90 read                          11111111 11111111
 F15  F7 18FD 00 read                          11111111 11111111
 F16  B7 0090 C38800    JP 88                  11111111 11111111
 F19  B7 0088 D379      OUT (79),A             11111111 11111111
 F1B  5F 8079 80 out                           11111111 11111111
 F1C  B7 008A 0F        RRCA                   11111111 11111111
 F1D  B7 008B 14        INC D                  11111111 11111111
 F1E  B7 008C 00        NOP                    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
 F22  D7 18FD 00 write                         11111111 11111111
 F23  D7 18FC 90 write                         11111111 11111111
 F24  F7 18FC 90 read                          11111111 11111111
 F25  F7 18FD 00 read                          11111111 11111111
 F26  B7 0090 C38800    JP 88                  11111111 11111111
 F29  B7 0088 D379      OUT (79),A             11111111 11111111
 F2B  5F 4079 40 out                           11111111 11111111
 F2C  B7 008A 0F        RRCA                   11111111 11111111
 F2D  B7 008B 14        INC D                  11111111 11111111
 F2E  B7 008C 00        NOP                    11111111 11111111
 F2F  B7 008D CDA000    CALL A0                11111111 11111111
 F32  D7 18FD 00 write                         11111111 11111111
 F33  D7 18FC 90 write                         11111111 11111111
 F34  F7 18FC 90 read                          11111111 11111111
 F35  F7 18FD 00 read                          11111111 11111111
 F36  B7 0090 C38800    JP 88                  11111111 11111111
 F39  B7 0088 D379      OUT (79),A             11111111 11111111
 F3B  5F 2079 20 out                           11111111 11111111
 F3C  B7 008A 0F        RRCA                   11111111 11111111
 F3D  B7 008B 14        INC D                  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=
0080  16FF     LD D,FF          (next step)  ok
N
AF=1801 (sz-a-pnC) BC=    0 DE=FF20 HL=FFFF IX=1234 IY=5678 SP=18FE PC=
0082  3E80     LD A,80
  ok
N
AF=1801 (sz-a-pnC) BC=    0 DE=FF20 HL=FFFF IX=1234 IY=5678 SP=18FE PC=
0082  3E80     LD A,80          (next step)  ok
N
AF=8001 (sz-a-pnC) BC=    0 DE=FF20 HL=FFFF IX=1234 IY=5678 SP=18FE PC=
0084  D37B     OUT (7B),A       (next step)  ok

BYE
```

FINDING THE BUG IN DEMO2.BIN

After enabling emulation memory (0 to 7ff emenable) 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 A8 ADR S
- A8 AS
- 0 8C M!
- A8 AS
- ONLY NOT A0 TO B0 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.

- 1AFTER 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 80 ADR S

This command is optional because it is rather bulky. However, it is an interesting example of a combination of both filtering and qualifiers used in a trigger spec. This command tells the UniLab that you only want to see what happens after address 80 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 8C. 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.

THE 1800 MICROPROCESSOR FAMILY

8-bit microprocessor
Separate data and address lines
Multiplexed upper and lower address lines
64K memory addressing (16 address bits)
Operating Frequency : DC to 3.2, 5 MHz

Features: On-chip clock
On-chip DMA

DEVICE	DIRECT ADDRESSABLE EXTERNAL MEM. K-BYTES	ON-CHIP RAM BYTES	ON-CHIP ROM BYTES	MAX. CLOCK FREQ. MHz	INSTRUCTION TIME MIN./MAX. (μs)	INTER-RUPTS	TIMER/ COUNTER BITS	PRE-SCALER	BUS MUX/ NON
CDP1802A *	64	—	—	3.2	5.0/7.5	•	—	—	NON
CDP1802B *	64	—	—	5.0	3.2/4.8	•	—	—	NON
CDP1804A	64	64	2048	5.0	3.2/16.0	•	8	DIV. 32	NON
CDP1805A *	64	64	—	5.0	3.2/16.0	•	8	DIV. 32	NON
CDP1806A	64	—	—	5.0	3.2/16.0	•	8	DIV. 32	NON

* NOT SUPPORTED
* USE FOR 1804 DEVELOPMENT

UniLab interface considerations:

Analyzer Cable 6

Maximum Clock Speed:) 5 MHz clock input (Standard UniLab)
Processor clock to UniLab clock ratio: 8:1

input clock → UniLab processor

UniLab does not have debug capabilities for 1800 family.

Reserved Areas: None
Overlay area: None

UniLab clock:

RD-	MRD
WR-	MWR
K1	Not used
K2	Not used

UDL status lines:

C4	Not Used
C5	Not Used
C6	SC0
C7	SC1

Cycle types:

	CONT
fetch	3F
read	7F
write	5F

THE 6301/03 MICROPROCESSOR FAMILY

6301 Microprocessor

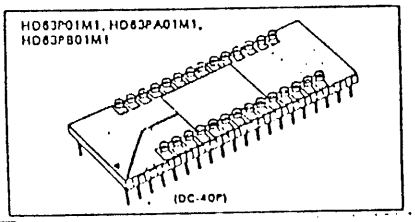
8-bit microprocessor
Multiplexed or Non-multiplexed modes
Internal ROM
Internal RAM
'Abundant' on-chip functions
Operating Frequency : 1, 1.5, 2 MHz

Type No.	Bus Timing
HD6301V1	1 MHz
HD63A01V1	1.5 MHz
HD63B01V1	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; EPROM 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.

Type No.	Bus Timing
HD6303R	1.0 MHz
HD63A03R	1.5 MHz
HD63B03R	2.0 MHz

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

(f = 0.5 ~ 1.0MHz; HD6303X
f = 0.5 ~ 1.5MHz; HD63A03X
f = 0.5 ~ 2.0MHz; HD63B03X)

(f = 0.1 to 1.0 MHz; HD6303Y
f = 0.1 to 1.5 MHz; HD63A03Y
f = 0.1 to 2.0 MHz; HD63B03Y)

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)

/ector Table

Address	Description
FFFE-FF	RES
FFEE-EF	TRAP (address or op-code error)
FFFC-FD	NMI
FFFA-FB	SWI (software int)
FFF8-F9	IRQ

SELECT ONE VECTOR FOR SOFTWARE BKPT.

UniLab interface considerations:

Analyzer Cable B for 6301, 6303R, 6303X and 6303Y
Analyzer Cable N for piggyback chip (63P01)

Reserved Areas: FFFA-FB or FFEE-EF Vector location for software breakpoint.
FFFC-FD Vector location for hardware breakpoint.
FFB9-BA Reserved location for interrupt routine (rel).

Overlay area: Starts at FF80, 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

ıb status lines:	6301/03R	63P01	6303X/03Y
C4	Not Used	Not Used	WR- (Port 7.1)
C5	Not Used	Not Used	RD- (Port 7.0)
C6	Not Used	Not Used	L1R (Port 7.3)
C7	SC2	Not Used	R/W (Port 7.2)

Cycle types:	6301/03R	63P01	6303X/03Y
fetch	FF	7F (int.op.)/ FF (ext.op.)	9F (first byte) DF
read	FF	7F/FF	DF
write	7F	7F/FF	6F

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

Microprocessors with Internal Two Phase Clock Generator		
Model	No. Pins	Addressable Memory
R6502	40	64K Bytes
R6503	28	4K Bytes
R6504	28	8K Bytes
R6505	28	4K Bytes
R6506	28	4K Bytes
R6507	28	8K Bytes
Microprocessors with External Two Phase Clock Input		
Model	No. Pins	Addressable Memory
R6512	40	64K Bytes
R6513	28	4K Bytes
R6514	28	8K Bytes
R6515	28	4K Bytes

1. All have IRQ interrupt except R6507.
2. NMI interrupt available only on R6502, 03, 12, 13.
3. SYNC line available only on R6502, 12.

65/11 and 65/41 'Backpack Emulator' (piggyback) chips

Multiplexed or Non-multiplexed modes
Operating frequency: 1, 2 MHz

The 65/11 is the PROM prototyping version of the 8-bit, masked-ROM 6500/11 processor.

Part Number	Memory Capacity	Compatible Memories	Temperature Range and Speed
R65/11EB	4K × 8	2732	0°C to 70°C 1MHz
R65/11EAB	4K × 8	2732A	0°C to 70°C 2 MHz

The 65/41 is the PROM prototyping version of the 8-bit, masked-ROM 6500/41 processor.

Part Number	Memory Capacity	Compatible Memories	Temperature Range and Speed
R65/41EB	4K × 8	2732	0°C to 70°C 1MHz
R65/41EAB	4K × 8	2732A	0°C to 70°C 2 MHz

6511Q and 6501Q 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

Part Number	Package Type	Frequency Option	Temp. Range
R6501Q	Plastic (QUIP)	1 MHz	0°C to 70°C
R6501AQ	Plastic (QUIP)	2 MHz	0°C to 70°C
R6511Q	Plastic (QUIP)	1 MHz	0°C to 70°C
R6511AQ	Plastic (QUIP)	2 MHz	0°C to 70°C

Software considerations:

Special commands

=ASAVE to change the PAGE 0 location used for saving the A register.
=DADR to change external RAM 'blind spot'.
=INTADR to point to user's IRQ handling routine, if any.
=RAM if your piggy back system has external RAM, use this one-way configuration word.
=64 for 65P package - for the 6510 processor in Commodor 64 computer.
=ALL shows all bus cycles.
=SYNC hides 'extra' bus cysles.

Vector Table

ADR	DESC.
FFFE-F	IRQ
FFFC-D	RES
FFFA-B	NMI

UniLab interface considerations:

Analyzer Cable B for 650X, 651X and 65X10
Analyzer Cable K for piggyback chips (65/11, 65/41)
Reserved Areas: FFFE-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:

RD-	Ø2
WR-	Not used
K1	Not used
K2	Not used

UniLab status lines:

C4	Not Used
C5	Not Used
C6	Sync (active high for op-code fetch)
C7	R/W-

(On piggyback chips, C7 connected to OE of piggyback socket; C6 not used)

Cycle types:

	650X w/SYNC	650X w/o SYNC	65/11, 65/41
fetch	FF (first byte)	FF	7F
	BF	FF	7F
read	BF	FF	7F
write	3F	7F	7F

EM65 Emulation Module available for 6502 and 65C02 microprocessors. Emulated target ROM must be removed from target system for proper operation.

THE 6800 MICROPROCESSOR FAMILY

The 6802 and 6808 are software compatible with the 6800.

The 6800 Microprocessor

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

Part	Speed
MC6800	1 MHz
MC68A00	1.5
MC68B00	2.0

6808 additional features

Internal clock oscillator and driver

Part	Speed
MC6808	1 MHz
MC68A08	1.5
MC68B08	2.0

6802 additional features

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

Part	Speed
MC6802	1 MHz
MC68A02	1.5
MC68B02	2.0

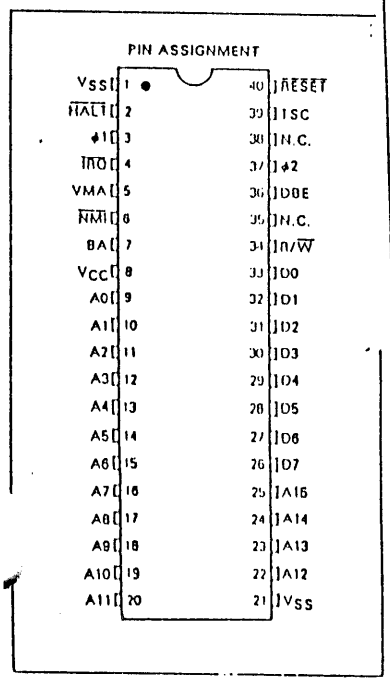
Software considerations:

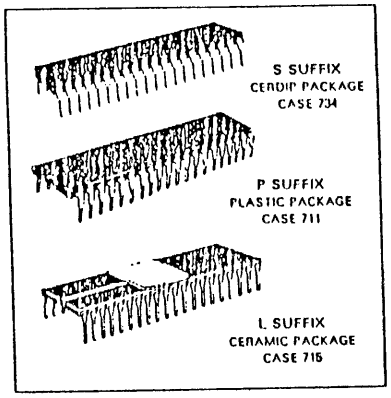
The 6802 package is necessary if there is internal RAM at addresses 0-128 (decimal). It can be used with 6800 and 6808, if there is external RAM at those addresses-- but we recommend against it.

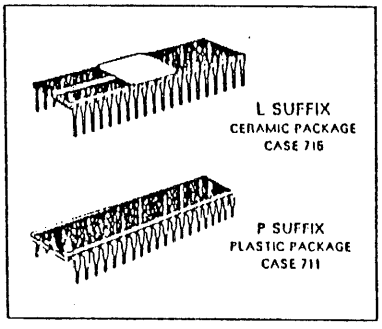
Special commands:

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

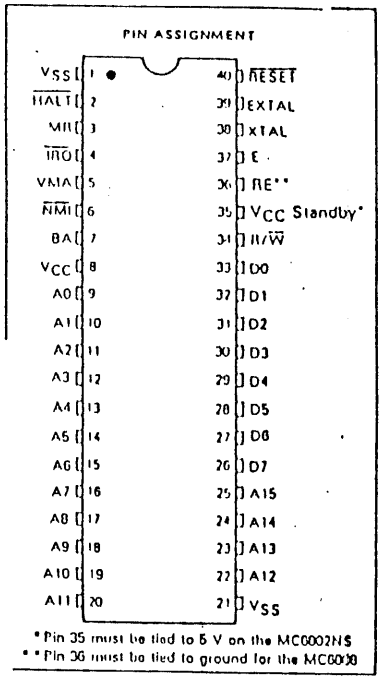
MC6800







MC6802
MC6808
MC6802NS



Vector Table

Address	Interrupt
FFFE-FF	RESET
FFFC-FD	NMI
FFFA-FB	SWI
FFF8-F9	IRQ

UniLab interface considerations

Analyzer cable B for 6800, 6802, 6808.

Reserved areas: FFFC-FD vector for hardware breakpoint.
FFFA-FB vector for software breakpoint.
FF89-BA for interrupt service routine. (rel)
6802 only: 50-53 in internal RAM. (rel)

Overlay area: FF8B-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 6808 have a 4:1 ratio between oscillator speed and processor clock speed.
With the 6800, the two clocks, Ø1 and Ø2, 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-	Ø2	(phase 2 clock)
	E	(clock output on 6802, 6808)
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 146805E2, 146805E3, 6305 and 6805 piggyback

HARDWARE FEATURES

146805
112 BYTES ON BOARD RAM
16 BIDIRECTIONAL I/O LINES
MULTIPLEXED ADDRESS DATA BUS
CAN ADDRESS UP 8K EXTERNAL MEMORY

146805E3

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 6 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.

PBACK Sets software package for piggyback mode
PBACK' Sets software for 146805E2
HD6305 Sets software for 6305
6805E3 Sets software for 6805E3
=ZP Sets ram location used in zero page by debug
Allcy/allcy' shows does not show all bus cycles
TCR/Tcr' Enables disables timer control register ($0009) at a breakpoint.

Vector Table

	146805E2	146805e3	68p05	6305
Reset	1FFE-F	FFFE-F	FFE-F	1FFE-F
SWI	1FFC-D	FFFC-D	FFC-D	1FFC-D

Unilab Interface Considerations

The 68p05 requires a special clock circuit which is part of the M-Cable.

Analyzer Cable: B for 146805e1/e3 and hd6305
M with clock circuit for 68p)%

Emulator Cable C824 or 28

Reserved Area: xFF89 - xFBA (x =1 for e2 and 6305, x = 0 for 68p05 and F for e3)

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

Maximum Clock Speed: MHZ

Unilab Clock:	146805e2/e3	68p05
	RD- as	To read on clockcircuit
	WR- N/C	N/C
	K1- N/C	N/C
	K2- N/C	N/C

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

NMI: NOT AVAIBLE 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.

6809

Differences from 6809 plain

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

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 "pluggy-back" 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

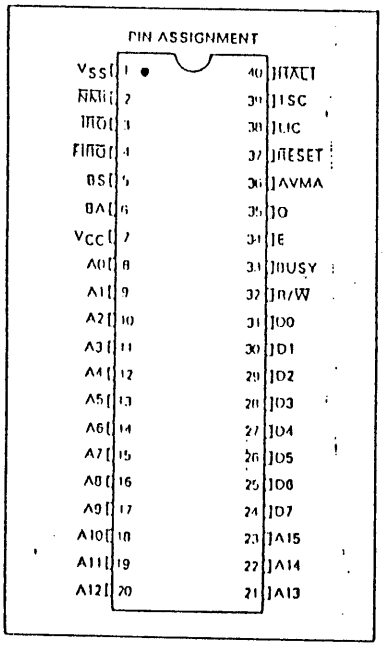
FFFA,B	SWI Vector (relocatable to FFF2,3 or FFF4,5)
FFFC,D	NMI Vector
FFB9-FFEF	Reserved area and overlay (relocatable

Motorola

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

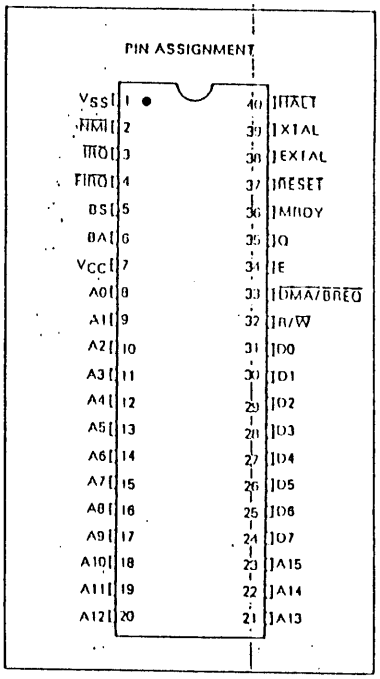
Note that FETCH is the same as READ, since AVMA only tells us that the next cycle will be the first byte of an instruction.

6809E



HARDWARE FEATURES
- External Clock Inputs, E and Q, Allow Synchronization
- TSC Input Controls Internal Bus Buffers
- LIC Indicates Opcode Fetch
- AVMA Allows Efficient Use of Common Resources in a Multiprocessor System
- BUSY Is a Status Line for Multiprocessing
- Fast Interrupt Request Input Stacks Only Condition Code Register and Program Counter
- Interrupt Acknowledge Output Allows Vectoring By Devices
- Sync Acknowledge Output Allows for Synchronization to External Event
- Single Bus Cycle RESET
- Single 5-Volt Supply Operation
- NMI Inhibited After RESET Until After First Load of Stack Pointer
- Early Address Valid Allows Use With Slower Memories
- Early Write Data for Dynamic Memories

6809



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

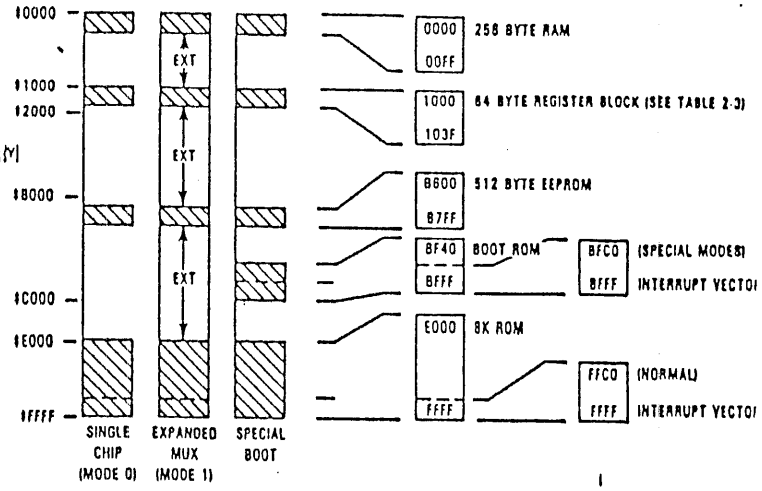
THE 68HC11 MICROCONTROLLER

HARDWARE FEATURES

model 1

8-bit Microcontroller
Multiplexed address and data
8K onboard ROM
512 bytes onboard 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

(Turn on disable mode)



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 16bit 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

=SFPAGE 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 disasembler.

SWI) Allows illegal opcode trap to be used instead of SWI for
SWI breakpoint.

Vector Taole

Table 3-1. Interrupt Vector Assignments

Vector Address	Interrupt Source	CC Register Mask	Local Mask
FFC0, C1	Reserved	—	—
•	•		
•	•		
FFD4, D5	Reserved	—	—
FFD6, D7	SCI Serial System	1 Bit	See Table 3-2
FFD8, D9	SPI Serial Transfer Complete	1 Bit	SPIE
FFDA, DB	Pulse Accumulator Input Edge	1 Bit	PAII
FFDC, DD	Pulse Accumulator Overflow	1 Bit	PAOVI
FFDE, DF	Timer Overflow	1 Bit	TOI
FFE0, E1	Timer Output Compare 5	1 Bit	OC5I
FFE2, E3	Timer Output Compare 4	1 Bit	OC4I
FFE4, E5	Timer Output Compare 3	1 Bit	OC3I
FFE6, E7	Timer Output Compare 2	1 Bit	OC2I
FFE8, E9	Timer Output Compare 1	1 Bit	OC1I
FFEA, EB	Timer Input Capture 3	1 Bit	OC3I
FFEC, ED	Timer Input Capture 2	1 Bit	OC2I
FFEE, EF	Timer Input Capture 1	1 Bit	OC1I
FFF0, F1	Real Time Interrupt	1 Bit	RTII
FFF2, F3	$\overline{IRQ}$ (External Pin or Parallel I/O)	1 Bit	See Table 3-3
FFF4, F5	$\overline{XIRQ}$ Pin (Pseudo Non-Maskable Interrupt) MK	X Bit	None
FFF6, F7	SWI BD	None	None
FFF8, F9	Illegal Opcode Trap	None	None
FFFA, FB	COP Failure (Reset)	None	NOCOP
FFFC, FD	COP Clock Monitor Fail (Reset)	None	CME
FFFE, FF	$\overline{RESET}$	None	None

Unilab Interface Considerations

The 68HC11 must be configured to run in the expande 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 internal stak also C8HD24/28

Reserved Area:. $FF71-$FF73

Overlay Area: $FF74-$FFBF

Maximum Clock Speed: MHZ

Unilab Clock:

RD-	E-clock
WR-	N/C
K1-	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 avaible for this processor at this time.

O R I O N I N S T R U M E N T S T E C H N I C A L N O T E S

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" ie.) 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 C8HD24 or C8HD28 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

68000 and 68008

The 68000 Family	Maximum Address	Maximum Data
68000 - 4,6,8,10,12.5,?	24-bit	16-bit
68008 - 8,10,?	20-bit	8-Bit
68010 -	24-bit	16-Bit
68020 -	32-bit	32-bit

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

80-83	Trap Vector (relocatable)
7C-7F	NMI Vector
7AC-7FF	Reserved area and overlay (relocatable)

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	Internal Program Memory	Internal Data Memory
8748H	EPROM	
8749H	EPROM	
8048AH	1K x8 ROM	64 x8
8049AH	2K x8 ROM	128 x8
8050AH	4K x8 ROM	256 x8
8035AHL	none	
8039AHL	none	
8040AHL	none	

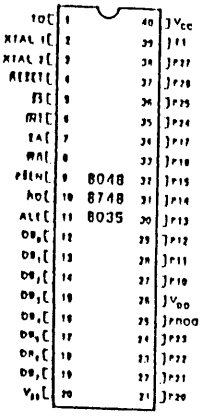
The major features are:

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

Oscilator 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



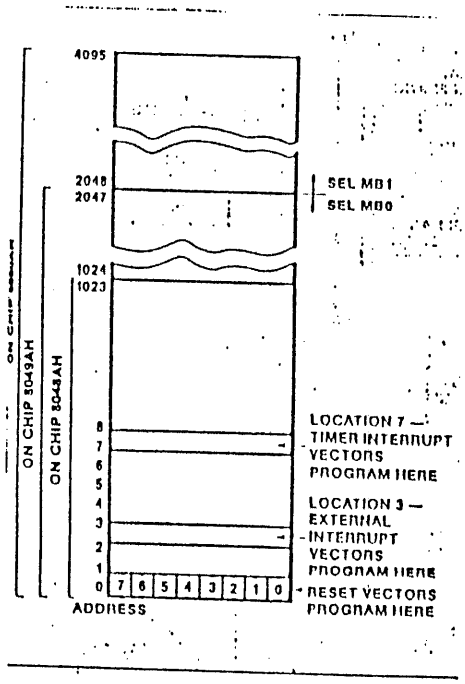


Figure 12-2. Program Memory Map

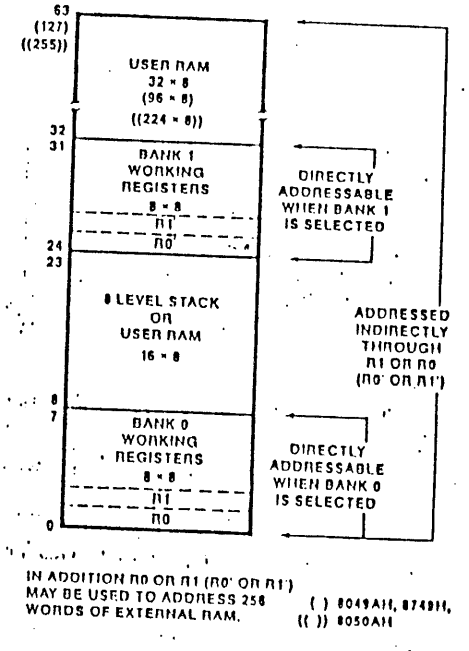


Figure 12-3. Data Memory Map

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.

type	vec address
reset	0H
external	3H
timer/ctr	7H

UniLab commands:

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

THE 8051 MICROPROCESSOR FAMILY

DOB-51: expanded mode (external rom)
Sales Ranking -- #2 with 13% of total sales

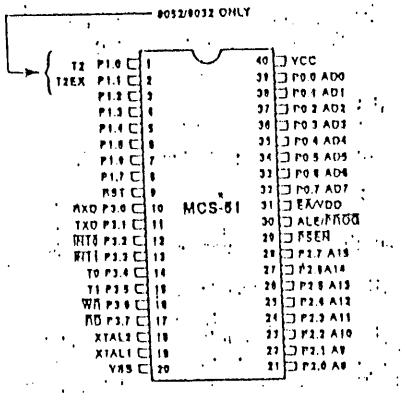
DOB-51P: OKI piggyback M80C51VS (interanl rom)
Sales Ranking -- #16 with less than 1%

The major features are:

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

Part	Technology	On-Chip Pgm Memory	On-Chip Data Mem
8051	HMOS	4K (ROM)	128
8031	HMOS	none	128
8751H	HMOS I	4K (EPROM)	128
80C51	CHMOS	4K (ROM)	128
80C31	CHMOS	none	128
8052	HMOS	8K (ROM)	256
8032	HMOS	none	256

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- PSN	RD- not used
	WR- WR6	WR- not used
	K1 RD7	K1 GRND
	K2 EA-	K2 ALE

Overlay Area: FFC0H -- FFF8H for expanded
FC0H -- FFF for piggyback

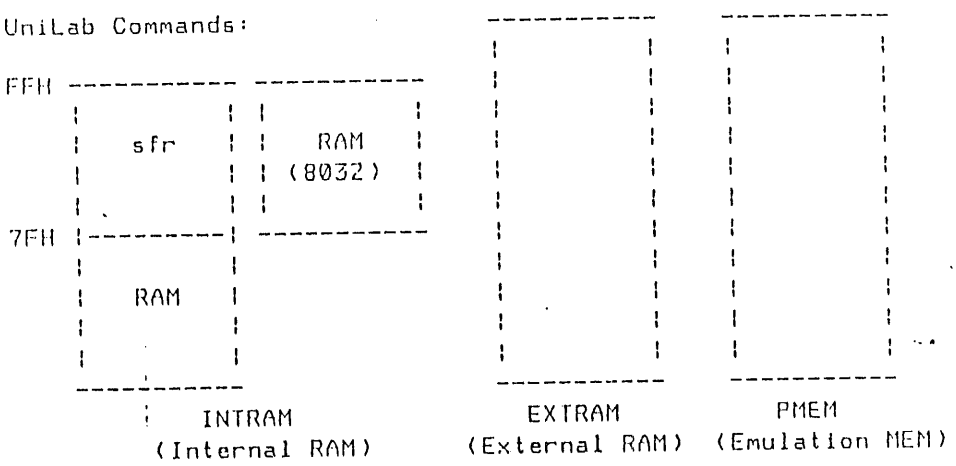
Breakpoint code: LCALL ovadr (12xxxx)

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

Interrupt: 2 levels of priority
No NMI

Interrupt Source	Vector Address	
IE0 (external request 0)	3H	^
TF0 (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	2BH	|

UniLab Commands:



8032

R? RI DR?
(DR? DR!) DR!
cannot use M here

=OVERLAY
n RNAME name assigns name to Rn

8085
(HMOS)

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 interupts (one is nonmaskable) and INTR line for a total of five interupts. The 8085 also supports serial I/O.

UniLab Interface

The 8085 uses analyzer cable "A"

Speed

		Standard UDL	High Speed UDL
8085AH	=3MHz	YES	YES
8085AH-2	=5MHz	YES	YES
8085AH-1	=6MHz		YES

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/OM pin 34

Cycle Types

FETCH	70 TO 7F
READ	50 TO 5F
WRITE	60 TO 6F
OUTPUT	E0 TO EF
INPUT	D0 TO DF
INTERRUPT	F0 TO FF
I/O	D0 TO EF

Special Commands

Gogo PATCH
=BP
=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

Part	Speed
8086	5 MHz
8086-2	8 MHz
8086-1	10 MHz

The 80186 Microprocessor
Incorporates the power of "15-20 chips" in one package. Higher performance, more instructions.

Part	Speed
80186	8 MHz
80186-6	6 MHz

The 80286 Microprocessor
24-bit address bus. 16 megabytes physical memory. 1 Megabyte "virtual" memory.

Part	Speed
80286-8	8 MHz
80286-6	6 MHz
80286-4	4 MHz

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:
8GMIN 186PATCH MAXMODE ATPATCH

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

MAX

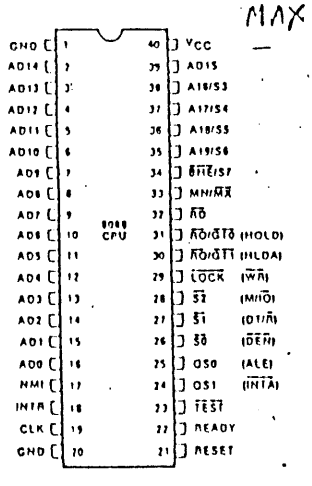


Figure 2. IAPX 86/10 Pin Configuration

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/$\overline{MX}$) 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/$\overline{MX}$ pin is strapped to GND, the 8086 treats pins 24 through 31 in maximum mode. An 8288 bus controller interprets status information coded into $\overline{S}_0, \overline{S}_1, \overline{S}_2$ to generate bus timing and control signals compatible with the MULTIBUS® architecture. When the MN/$\overline{MX}$ pin is strapped to V_{CC}, 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

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 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.
FFFB1-B2 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.

		80186	8086MIN	8086MAX	
UniLab clock:	RD-	RD	RD	11	These lines
	WR-	WR	WR	9	attach to
	K1-	DT/R	DTR	16	the bus
	K2-	DEN	INA	4	controller.

In addition, we use the Unilab input ALE, to determine when the address is valid: ALE ALE 5

		80186	8086MIN	8086MAX
Status lines:	C4	na	na	na
	C5	S0	na	S0
	C6	S1	na	S1
	C7	S2	MIO	S2

		8086max/80186	8086MIN	80286	AI
Cycle types:	fetch	80-9F	BF	B0-BF	60-7F
	read	A0-BF	80-BF	A0-AF	40-5F
	write	C0-DF	C0-FF	C0-DF	C0-DF
	output	40-5F	50-7F	50-5F	20-3F
	input	20-3F	00-4F	30-3F	E0-FF
	interrupt	00-1F		00-0F	00-1F

THE 8088 MICROPROCESSOR FAMILY

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

Part	Speed
8088	5 MHz
8088-2	8 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 88MAX PCPATCH

Special commands:

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

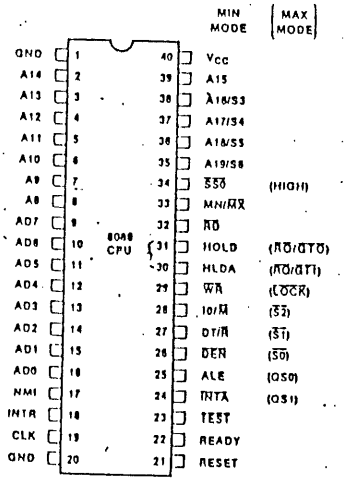


Figure 2. IAPX 88/10 Pin Configuration

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 (MN/$\overline{MX}$) 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 MN/$\overline{MX}$ pin is strapped to GND, the 8088 defines pins 24 through 31 and 34 in maximum mode. When the MN/$\overline{MX}$ pin is strapped to V_{CC}, 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:

00004-F vectors for single-step, for NMI and for software breakpoint.
FFFB1-B2 for interrupt service routine. (relocatable)

Overlay area: FFFB8-FFEF

Breakpoint code: INT 3 inserted at breakpoint.

Maximum oscillator speed:

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

		80189	8088MIN	8088MAX	
UniLab clock:	RD-	RD	RD	11	These lines
	WR-	WR	WR	9	attach to
	K1-	DT/R	DTR	16	the bus
	K2-	DEN	INA	4	controller.
UniLab input ALE		ALE	ALE	5	

		80188	8088MIN	8088MAX
Status lines:	C4	na	na	na
	C5	S0	(BHE)	S0
	C6	S1	na	S1
	C7	S2	MIO	S2

		8088max/80188	8088MIN	PC
Cycle types:	fetch	80-9F	1F	40-5F
	read	A0-BF	20-3F	60-6F
	write	C0-DF	40-5F	C0-CF
	output	40-5F	C0-DF	00-0F
	input	20-3F	A0-BF	E0-EF
	interrupt	00-1F	80-9F	

8096

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

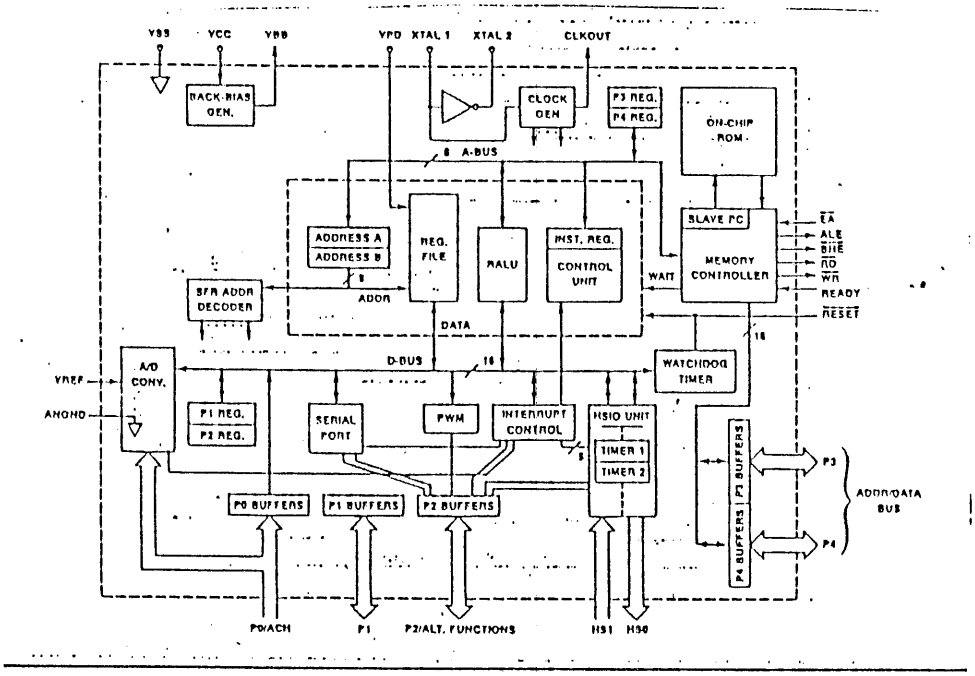


Figure 2-1. Block Diagram (For simplicity, lines connecting port registers to port buffers are not shown.)

Cycle Types: fetch F0 -- FF
read E0 -- EF
write C0 -- CF

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

To change overlay area : =OVERLAY

To change register used by Debugger : n =DBG

Dissasembler special features:
If no external RAM : n INTERNAL will signal dissasembler not to expect any memory cycles. (EXTERNAL)
ALL-INTERNAL
ALL-EXTERNAL

INTERRUPT STRUCTURE:

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

SPECIAL COMMANDS

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

Z80

General Information

The Z80 Microprocessor (40 pin dip package) was introdueced 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 interupts; Mode 0, Mode 1, and Mode 2. The UDL handles one byte interupts very well. At the present time we have problems with three (3) byte interupts.

The Z80 supports NMI.

UniLab Interface

The Z80 microprocessor uses analyzer cable "E".

Speed

	Standard UDL	High Speed UDL
Z80 = 2 MHz	YES	YES
Z80A = 4 MHz	YES	YES
Z80B = 6 MHz	NO	YES/MAYBE
Z80H = 8 MHz	NO	NO

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 *IORQ pin 20
A19 to *MEMRQ 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
(CMOS)

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, MMV, two serial ports and two 16-bit timers. The HD64180 supports the Z80 Mode 0, Mode 1, and Mode 2. It has many other interupts, i.e. from special functions on board plus two additional interupt lines. It also supports NMI.

UniLab Interface

The HD64180 uses analyzer cable "E"

User enters HD64180 patch word.

Speed

	Standard UDL	High Speed UDL
2 MHz	YES	YES
4 MHz	YES	YES
6 MHz	NO	NO

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	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

HD64180 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

NSC800
(CMOS)

General Information

The NSC-800 CMOS 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 interupt inputs it has. The Z80 has NMI and INT. The NSC-800 has 4 in addition to NMI. The interupts 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

	Standard UDL	High Speed UDL
2 MHz	YES	YES
4 MHz	YES	YES
6 MHz	NO	YES
8 MHz	NO	YES

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 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	E0 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 S8 MICROPROCESSOR FAMILY

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 A8-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

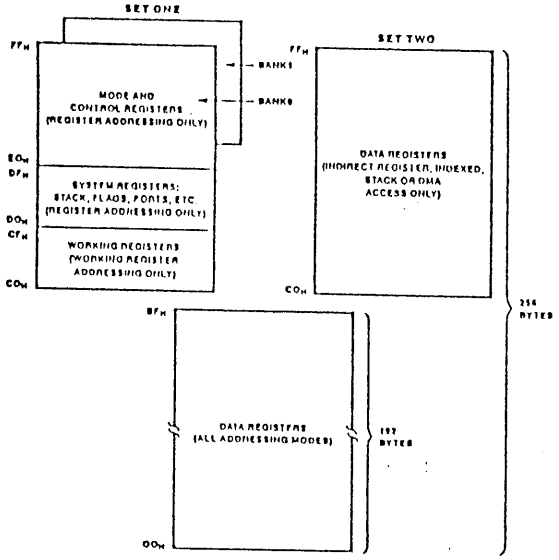
Reserved areas: 781 to 785 reserved (relocatable).
3 consecutive registers (default is register 60-63).

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

```
cy# CONT ADR DATA                    MDATA  MISC
 -2  FF 0020 E6F0FF   LD F0,#FF      11111111 11111111   < PORT 0 MODE REGISTER CONFIGURED.
  1  FF 0023 E6F128   LD F1,#28      11111111 11111111   < PORT 1 MODE REGISTER CONFIGURED.
  4  FF 0026 E6FE02   LD FE,#2       11111111 11111111   < EXT. MEMORY TIMING REG.
  7  FF 0029 E6D910   LD D9,#10      11111111 11111111   < STACK POINTER, LSB.
  A  FF 002C E6D800   LD D8,#0       11111111 11111111   < STACK POINTER, MSB.
  D  FF 002F E66007   LD 60,#7       11111111 11111111   < REGISTER POINTER FOR GAINING
 10  FF 0032 E66101   LD 61,#01      11111111 11111111     DEBUG CONTROL.
 13  FF 0035 C6C01234 LDW C0,#1234   11111111 11111111
 17  FF 0039 C6C25678 LDW C2,#5678   11111111 11111111
 1B  FF 003D C6C4ABCD LDW C4,#ABCD   11111111 11111111
 1F  FF 0041 C6C6EF12 LDW C6,#EF12   11111111 11111111
```

UniLab Commands:

n =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 =OVERLAY	Changes location of overlay area. Can only be placed at an even address. Reserved area is 5 bytes below this.
n =WREG	Set starting address for disassembler.
EXTRAM	Set debug operations to work in external data memory.
EXTROM	Set debug operations to work in program memory.
EXTSTACK	Set external stack mode.
INTSTACK	Set internal stack mode.
n DR!	Set the data register (C0-FF) to indicated value.
n DR?	Display current value of indicated data register.
n R!	Set register to indicated value.
n R?	Display current value of indicated data register.



. Super8 Registers

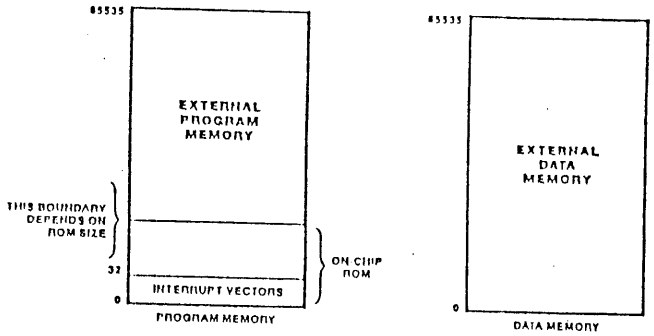


Figure 3-7. Program and Data Memory Address Spaces

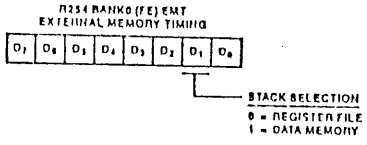


Figure 12-5. External Memory Timing

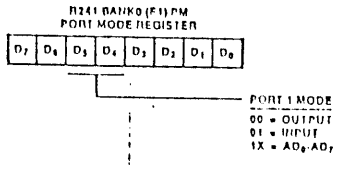


Figure 12-2. Configuring Port 1 for External Memory

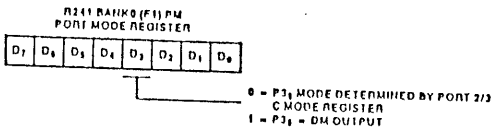


Figure 12-6. Data Memory

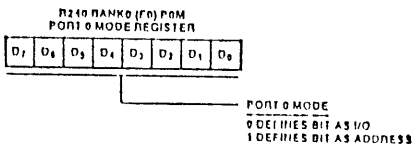


Figure 12-3. Configuring Port 0 for External Memory

THE Z8 MICROPROCESSOR FAMILY

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

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 29) 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:

	Z8681 int.stack	Z8681 ext.stack	Z8 piggyback
fetch	FF	FF	FF
read	FF	DF	FF
write	7F	5F	FF

UniLab Commands:

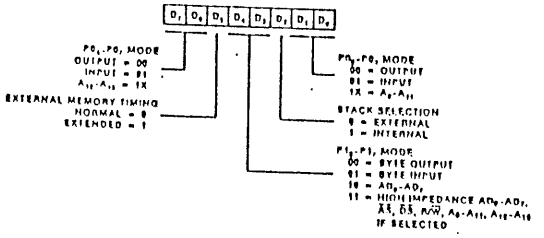
n =PTR	Changes reserved register pair, where 'n' indicates the lower numbered register in hex. Default is 60H. NOTE: Program must include instructions to load register pair with reserved area value. Breakpoints will not work properly unless they are placed after these initialization instructions.
n =OVERLAY	Changes location of overlay area. Can only be placed at an even address. Reserved area is 5 bytes below this.
EXTDATA	Set disassembler to expect stack operations in external RAM.
EXTRAM	Set debug operations to work in external data memory.
INTDATA	Set disassembler to expect stack operations in internal RAM.
PMEM	Set debug operations to work in external program memory.
PBACK	Set disassembler for Z-8 piggyback chip (Z8603 or Z8613).
PBACK'	Set disassembler for Z-8 expanded chip (Z8681 or Z8682)

```
cy# CONT ADR DATA                          MDATA  MISC
 -1  FF C00C E600C0  LD 0,#C0        11111111 11111111  ( MSB of address loaded.
  2  FF C00F E6F896  LD F8,#96       11111111 11111111  ( Port 0-1 Mode reg. config.
  5  FF C012 8DC015  JP ALW,C015     11111111 11111111  ( Three byte jump inst.
  8  FF C015 E66007  LD 60,#7        11111111 11111111  ( Register pointer for gaining
  B  FF C018 E661AF  LD 61,#AF       11111111 11111111     debug control.
  E  FF C01B E6FD10  LD FD,#10       11111111 11111111
 11  FF C01E E6FE00  LD FE,#0        11111111 11111111  ( Stack pointer, LSB.
 14  FF C021 E6FF40  LD FF,#40       11111111 11111111  ( Stack pointer, MSB.
 17  FF C024 E6FC00  LD FC,#0        11111111 11111111
 1A  FF C027 0C01    LD R0,#1        11111111 11111111
 1C  FF C029 1C23    LD R1,#23       11111111 11111111
```

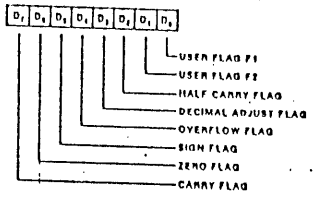
Control Registers

Registers (Continued)

R248 P01M
Port 0 and 1 Mode Register
($F8_H$; Write Only)



R252 FLAGS
Flag Register
(FC_H; Read/Write)



DEC		HEX	IDENTIFIERS
255	STACK POINTER (BITS 7-0)	FF	SPL
254	STACK POINTER (BITS 15-8)	FE	SPH
253	REGISTER POINTER	FD	RP
252	PROGRAM CONTROL FLAGS	FC	FLAGS
251	INTERRUPT MASK REGISTER	FB	IMR
250	INTERRUPT REQUEST REGISTER	FA	IRQ
249	INTERRUPT PRIORITY REGISTER	F9	IPR
248	PORTS 0-1 MODE	F8	P01M
247	PORT 3 MODE	F7	P3M
246	PORT 2 MODE	F6	P2M
245	T0 PRESCALER	F5	PRE0
244	TIMER/COUNTER 0	F4	T0
243	T1 PRESCALER	F3	PRE1
242	TIMER/COUNTER 1	F2	T1
241	TIMER MODE	F1	TMR
240	SERIAL I/O	F0	SIO
	NOT IMPLEMENTED		
127		7F	
	GENERAL-PURPOSE REGISTERS		
4		04	
3	PORT 3	03	P3
2	PORT 2	02	P2
1	PORT 1	01	P1
0	PORT 0	00	P0

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. Z8000 CPUs, Summary of Differences

	Z8001	Z8002	Z8003	Z8004
Addressing Spaces				
a. Segmented	Yes	No	Yes	No
b. Nonsegmented	Yes	Yes	Yes	Yes
Number of Output Address Bits	23	16	23	16
Virtual Memory Input Pin ($\overline{\text{AIIOMI}}$)	No	No	Yes	Yes
Separate External Interrupt Input Pin for Access Violation Signal from MMU	Yes	No	Yes	No
ISCI Instruction Enhancement	No	No	Yes	Yes
Package Size (Pins)	40	40	40	40

Table 2-1. Status Line Codes

St_3-St_0	Definition
0 0 0 0	Internal Operation
0 0 0 1	Memory Refresh
0 0 1 0	I/O Reference
0 0 1 1	Special I/O Reference
0 1 0 0	Segment Trap Acknowledge
0 1 0 1	Nonmaskable Interrupt Acknowledge
0 1 1 0	Nonvectored Interrupt Acknowledge
0 1 1 1	Vectored Interrupt Acknowledge
1 0 0 0	Data Memory Request
1 0 0 1	Stack Memory Request
1 0 1 0	Data Memory Request (CPU)
1 0 1 1	Stack Memory Request (CPU)
1 1 0 0	Instruction Space Access
1 1 0 1	Instruction Fetch, First Word
1 1 1 0	Transfer between CPU and CPU
1 1 1 1	Test and Set Data Access (Z8003 and Z8004 only)

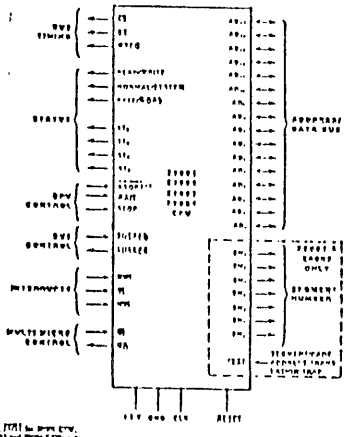


Figure 2-1. Z8000 Pin Functions

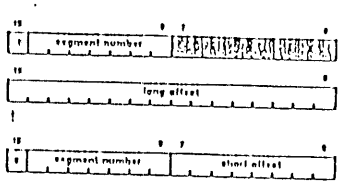


Figure 2-2. Segmented Memory Address Within Instruction

NOTE: Shaded area is reserved.

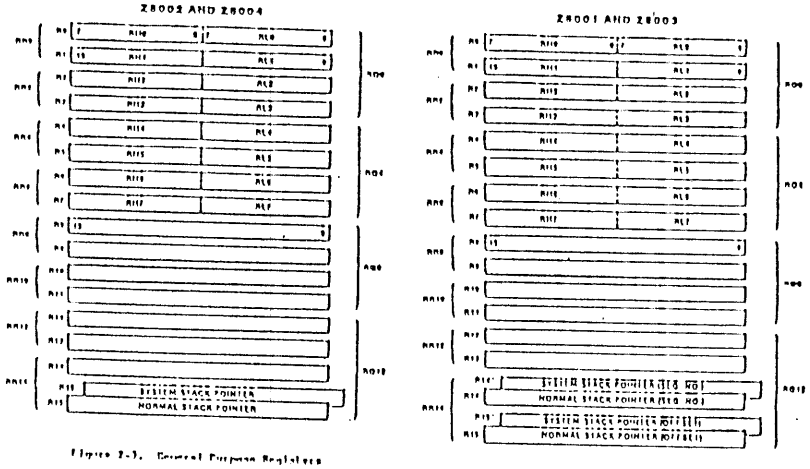


Figure 2-3. General Purpose Registers

Unilab Interface

Analyzer Cable C

UniLab Clock:

RD-	DS
WR-	not used
K1	not used
K2	not used

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

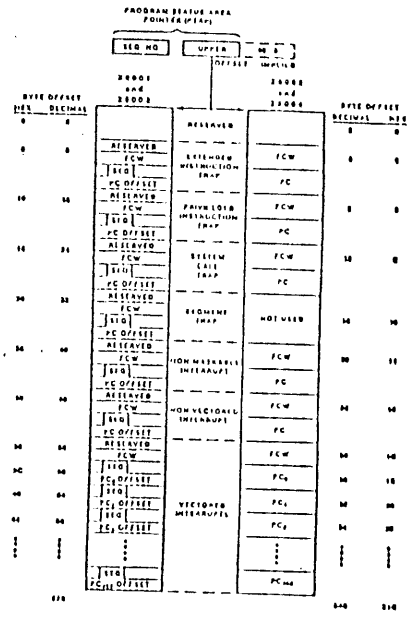
Break Point Opcode: EXTENO (F00)
SC xx (7Fxx) if co-processor is being used

Cycle Types:

read	C0 to CF
write	40 to 4F
fetch	E0 to FF
input	80 to 8F
output	00 to 0F
spec in	90 to 9F
spec out	10 to 1F
stack read	D0 to DF
stack write	50 to 5F

Interrupts:

TYPE	FCW adr	PC adr
reset	2	4
extend	PSA + 4	PSA + 6
sc	PSA + C	PSA + E
nmi	PSA + 14	PSA + 16



Program Status Area

UniLab Commands:

=RAM	set addres of RAM
=RAM.SEGMENT	set segment where RAM resides
=ROM.SEGMENT	set segment where ROM resides
SEGMENTED	turn segmented mode on
UNSEGMENTED	turn segmented mode off
=OVERLAY	

Feautres:

- 8-bit data bus
- 16-bit address bus
- CMOS
- 8K x8 Internal ROM
- 256 x8 Internal RAM
- 95 instruction type
- 451 total instructions

79312 Family:

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

Clock Speed: 12 Mhz (standard)

CHAPTER 1 PIN FUNCTIONS

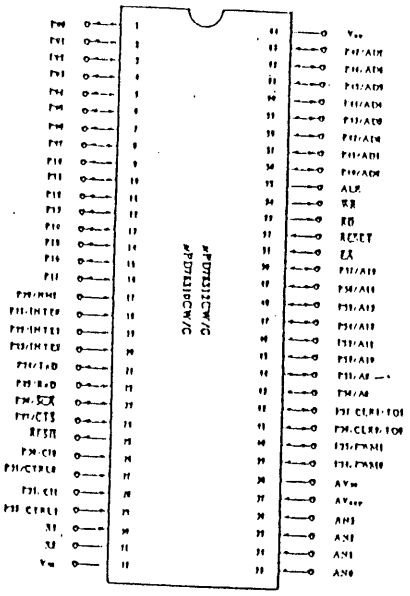
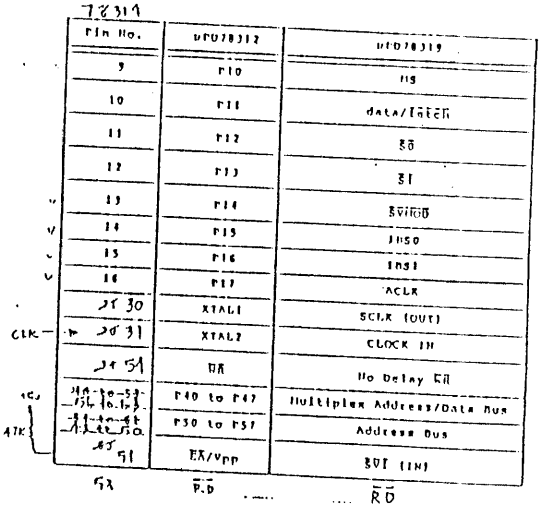


Figure 1.1 Pin Configurations SHRINK DIP and QUIP Packages



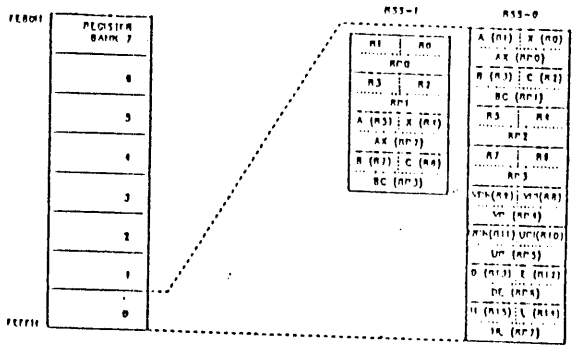


Figure 2.5 General-Purpose Register Designation and Storage

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: 80 -- 82
even address =OVERLAY
BPTRIGADR must be at odd address &
program uses one byte before BPTRIGADR

Break Point Code: BRK (5E)

Interrupts:

Table 4.1 Interrupt sources and Vector Addresses.

	DEFAULT PRIORITY		INTERRUPT SOURCE	MACRO SERVICE	VECTOR ADDRESS
	-	RESET	EXTERNAL RESET LINE	-	0000H
NON-MASKABLE INTERRUPTS	-	NMI	EXTERNAL NON-MASKABLE INT.	N	0002H
	-	WDT	WATCHDOG TIMER	N	000AH
MASKABLE INTERRUPTS	0	CNF00	UP/DOWN COUNTER	Y	001AH
	1	CNF01	UP/DOWN COUNTER	N	001CH
	2	CNF10	UP/DOWN COUNTER	Y	001EH
	3	CNF11	UP/DOWN COUNTER	N	0020H
	4	EXIF0	EXTERNAL INTERRUPT 0	Y	0004H
	5	EXIF1	EXTERNAL INTERRUPT 1	Y	0006H
	6	EXIF2	EXTERNAL INTERRUPT 2	Y	0008H
	7	TIMF0	TIMER FLAG 0	Y	000EH
	8	TIMF1	TIMER FLAG 1	Y	0010H
	9	TIMF2	TIMER FLAG 2	Y	0012H
	10	SEF	SERIAL PORT ERROR	N	0022H
	11	SRF	SERIAL PORT RECEIVE BUFFER	Y	0024H
	12	STF	SERIAL PORT TRANSMIT BUFFER	Y	0026H
	13	ADF	A/D CONVERTER DONE FLAG	Y	0028H
	14	TBF	TIMEBASE COUNTER FLAG	N	000CH
	-	BRK	BREAK INSTRUCTION	N	003EH

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

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

UniLab Commands:

SHOW-NAMES/HIDE-NAMES	show/hide sfr names
ALIGN/ALING	align cycles
adr SFR-SHOW	display name for sfr at adr
=OVERLAY	

TO: Bill White
FROM: Jim K.

Cross-assemblers and Cross-compilers

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 refered 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 variey 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 compatability

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 ROM 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 seqence 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:

```
cy#                ADR DATA
  0                0000 310019    LD SP,INIT.STACK
  3 START.LOOP     0003 3E12      LD A,12
  5                0005 015634    LD BC,INIT.BC
  8                0008 119A78    LD DE,INIT.DE
  B                000B 21DEBC    LD HL,INIT.HL
  E                000E C5        PUSH BC
  F                18FF 34 write
 10                18FE 56 write
 11                000F C1        POP BC
 12                18FE 56 read
 13                18FF 34 read
 14 FIRST.INCA     0010 3C        INC A
 15                0011 3C        INC A
  .                  .              .
  .                  .              .
  .                  .              .
 2A                0026 3C        INC A
 2B                0027 3C        INC A
 2C LAST.INCA      0028 3C        INC A
 2D END.LOOP       0029 C30300    JP START.LOOP
 30 START.LOOP     0003 3E12      LD A,12
 32                0005 015634    LD BC,INIT.BC
 35                0008 119A78    LD DE,INIT.DE
 38                000B 21DEBC    LD HL,INIT.HL
 3B                000E C5        PUSH BC
```

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
```

** EPROMS **

EPROM	VOLTAGE	PERSONALITY MODULE	READ/PROGRAM EPROM COMMAND	EPROM SIZE DEC.	HEX
2716 (1,a)	25	PM16	RPROM / P2716 OR PD2716	2K	7FF
2716B	12	PM16B	RPROM / P2716 OR PD2716		
27C16 (n)	25	PM16	RPROM / P2716 OR PD2716		
2732 (h)	21	PM32	R2732 / P2732A	4K	FFF
2732A (1,tx)	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.	PM64	RPROM / P2764 OR PD2764		
27C64 (1)	12.5	PM56	RPROM / P2764 OR PD2764		
27C64 (h,f)	21.	PM64	RPROM / P2764 OR PD2764		
27128 (1,h,t)	21.	PM64	RPROM / P27128	16K	3FFF
128A (1)	12.5	PM56	RPROM / P27128		
2128 (tx)	12.5	PM56	RPROM / P27128		
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 algorithum			
27512 (1)	12.5	PM512	(op) R27512 / P27512 * need 64K UDL in 8 bit mode * need 128K UDL in 16 bit mode	64K	FFFF

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

= optional

Orion also supports:

48016 (EEPROM)	PM16	P48016
2532	PM16	P2532
2564	can only emulate	

ORION
Instruments Orion Instruments, Inc. • 702 Marshall Street • Redwood City • CA • 94063 • USA • (415) 361-8883 • Telex 530942

BACKGROUNDER: 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 Paks^{tm} 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.S.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 MISSLES & 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
NORTHRUP 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 UNIVERISTY

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

Orion Instruments, Inc. • 702 Marshall Street • Redwood City • CA • 94063 • USA • (415) 361-8883 • Telex 530942

Pub. No. 15-01-10

(7/87) Printed in U.S.A.