Please do not upload this copyright pdf document to any other website. Breach of copyright may result in a criminal conviction.

This Acrobat document was generated by me, Colin Hinson, from a document held by me. I requested permission to publish this from Texas Instruments (twice) but received no reply. It is presented here (for free) and this pdf version of the document is my copyright in much the same way as a photograph would be. If you believe the document to be under other copyright, please contact me.

The document should have been downloaded from my website https://blunham.com/Radar, or any mirror site named on that site. If you downloaded it from elsewhere, please let me know (particularly if you were charged for it). You can contact me via my Genuki email page: https://www.genuki.org.uk/big/eng/YKS/various?recipient=colin

You may not copy the file for onward transmission of the data nor attempt to make monetary gain by the use of these files. If you want someone else to have a copy of the file, point them at the website. (https://blunham.com/Radar). **Please do not point them at the file itself as it may move or the site may be updated.**

It should be noted that most of the pages are identifiable as having been processed by me.

I put a lot of time into producing these files which is why you are met with this page when you open the file.

In order to generate this file, I need to scan the pages, split the double pages and remove any edge marks such as punch holes, clean up the pages, set the relevant pages to be all the same size and alignment. I then run Omnipage (OCR) to generate the searchable text and then generate the pdf file.

Hopefully after all that, I end up with a presentable file. If you find missing pages, pages in the wrong order, anything else wrong with the file or simply want to make a comment, please drop me a line (see above).

It is my hope that you find the file of use to you personally – I know that I would have liked to have found some of these files years ago – they would have saved me a lot of time !

Colin Hinson
In the village of Blunham, Bedfordshire.

TI-99/4A CONSOLE
AND
PERIPHERAL EXPANSION SYSTEM

TECHNICAL DATA

TABLE OF CONTENTS

Section A—TI-99/4A CONSOLE

Illustrations—Section A

Section B—INTERFACE CONSIDERATIONS

Illustrations—Section B

Section C—FILE MANAGEMENT SPECIFICATIONS

Illustrations—Section C

Section D—DEVICE SERVICE ROUTINE SPECIFICATIONS

SECTION A TI-99/4A CONSOLE

A.1 GENERAL DESCRIPTION

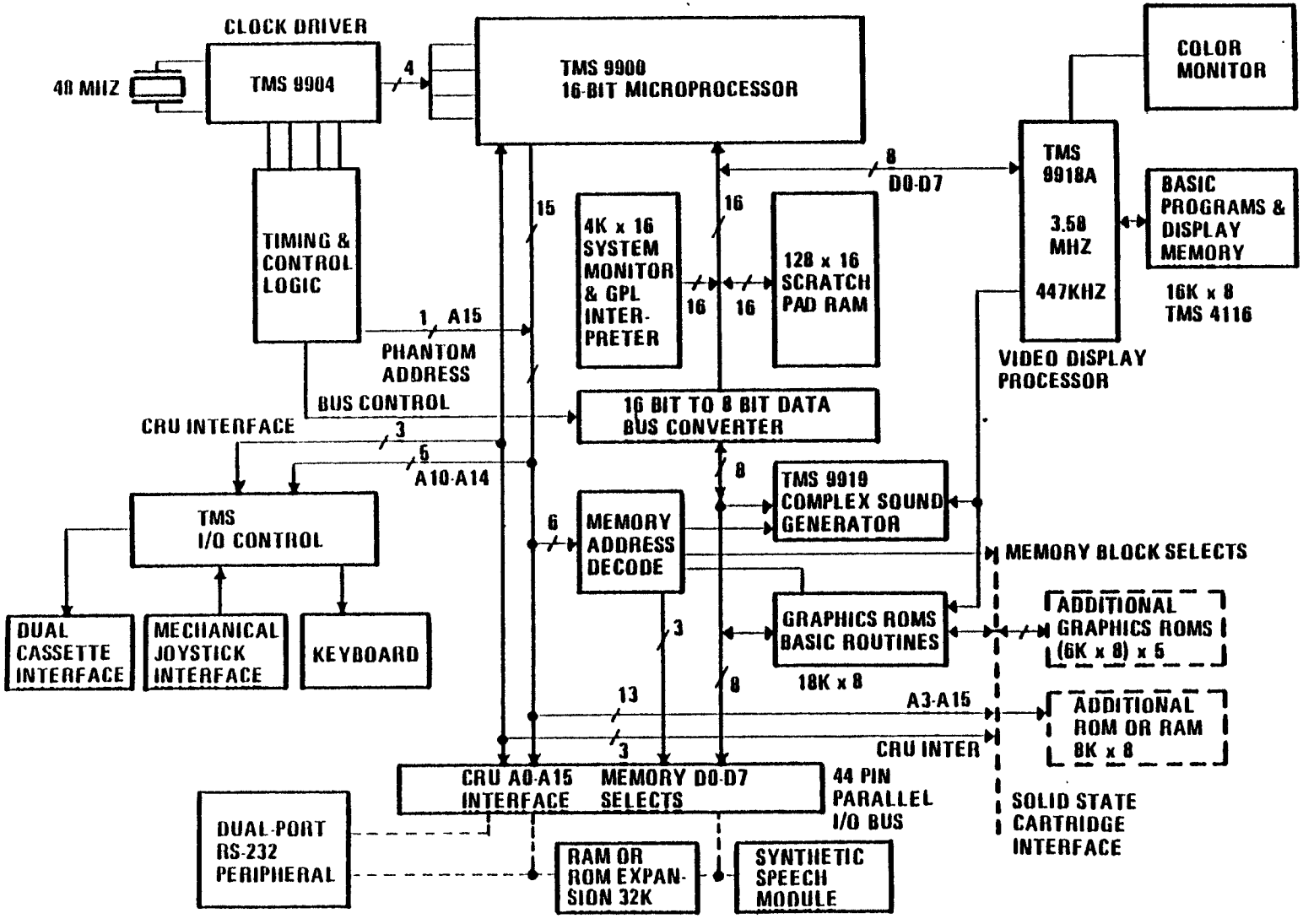
The purpose of this section is to provide neccesary information concerning the I/O bus (input/output signal connection) of the TI-99/4A console for persons interested in designing peripherals for the computer.

It is assumed that readers of this manual have a working knowledge of electronics and computers, especially in regard to the TMS 9900 microprocessor and its Communications Register Unit (CRU) I/O technique. Sources for this information are the TMS9900 Microprocessor Data Manual (MP001, Revision A) or the TMS9901 Programmable Systems Interface (MP003; July, 1978). These books may be obtained from TI semiconductor distributors or the TI Learning Center. A glossary is provided on page 57 of this manual.

The I/O bus provides an interface between the console and its peripherals. This bus utilizes both memory-mapped I/O and CRU I/O. The memory bus is frequently used for instruction fetch from read-only memory (ROM) in external peripherals and for data transfer to and from memory-mapped portions of these devices. The CRU bus is used for peripheral enable/disable, device control, and data transfer to/from CRU-mapped portions of these peripherals.

The TMS 9900 microprocessor accesses each peripheral to obtain instructions from the device service routine (DSR) ROM. Since each peripheral contains its own DSR, the TI-99/4A does not have to be designed to anticipate future peripheral requirements. The dual I/O bus capability, interrupt handling, and external DSRs provide flexibility at low cost.

FIGURE A. TI 99/4A SYSTEM BLOCK DIAGRAM



A.2 I/O PIN DESCRIPTION

SIGNATURE	PIN	I/O	DESCRIPTION
A0 (MSB)	31	Out	
A1	30	Out	ADDRESS BUS
A2	20	Out	
A3	10	Out	A0 through A15 comprise the address bus.
A4	7	Out	This bus provides the 16-bit memory
A5	5	Out	address vector to the external memory
A6	29	Out	system when MEMEN is active. Address
A7	17	Out	bit 15 is also used for CRU DATA OUT
A8	14	Out	on CRU output instructions.
A9	18	Out	
A10	6	Out	
A11	8	Out	
A12	11	Out	
A13	15	Out	
A14	16	Out	
A15/CRUOUT	19	Out	
D0 (MSB)	37	I/O	DATA BUS
D1	40	I/O	
D2	39	I/O	D0 through D7 comprise the bidirectional
D3	42	I/O	data bus. This bus transfers memory data
D4	35	I/O	to (when writing) and from (when reading)
D5	38	I/O	the external memory system when MEMEN is
D6	36	I/O	active.
D7	34	I/O	
			BUS CONTROL
MEMEN	32	Out	MEMory ENable. MEMEN indicates a memory access.
DBIN	9	Out	Data Bus IN. When active (high) the data buffers and 9900 are in the input mode.
WE	26	Out	Write Enable. WE indicates a memory write.
MBE	28	Out	Memory Block Enable. MBE indicates a memory access in memory block 4000-5FFF.
CRUCLK	22	Out	CRU CLocK. Input data line to the Home Computer
CRUIN	33	In	CRU data IN. Input data line to the Home Computer.

I/O PIN DESCRIPTION (CONTINUED)

SIGNATURE	PIN	I/O	DESCRIPTION
			MEMORY CONTROL
READY	12	In	READY (when $\overline{\text{MEMEN}}$ is active) indicates external memory is ready for a memory access.
IAQ	41	Out	Instruction AcQuisition indicates the CPU is acquiring an instruction during the current memory cycle.
			TIMING AND CONTROL
$\overline{\text{LOAD}}$	13	In	When active, $\overline{\text{LOAD}}$ causes the CPU to execute a nonmaskable interrupt; memory addresses FFFC and FFFE contain the new workspace and PC vectors, respectively.
$\overline{\text{RESET}}$	3	Out	When active, $\overline{\text{RESET}}$ causes the Home Computer and the peripherals to be reset. $\overline{\text{RESET}}$ will be held active for a minimum of five clock cycles.
$\overline{\text{EXT}}$ $\overline{\text{INT}}$	4	In	EXTernal INTerrupt. When active, $\overline{\text{EXT}}$ $\overline{\text{INT}}$ causes the CPU to execute an interrupt.
$\overline{\text{ø3}}$	24	Out	CPU Clock. Phase 3 of the CPU clock.
			POWER
GND	21,23 25,27		Ground reference.
			SPEECH MODULE SIGNALS
SBE	2	Out	Speech Block Enable. SBE indicates a memory access in the speech memory.
AUDIO IN	44	In	Input for the audio from the speech module
+5	1		Supply voltage (+5v Nom) for speech module (50ma Max)*
-5	43		Supply voltage (-5v Nom) for speech module (50ma Max)*

* NOTE: Pins 1 and 43 are not intended for use by consumer. Overload may cause permanent damage to console.

A.3 MEMORY ALLOCATION

The memory address space is broken into eight blocks of 8K bytes of memory. The third block (addresses 4000-5FFF) is predecoded and made available at the I/O port for the peripherals. The second, sixth, seventh, and eighth blocks (addresses 2000-3FFF and A000-FFFF) are in the Memory Expansion peripheral. For the speech module (addresses 9000-97FF), a predecoded line is available at the I/O port.

SYSTEM MEMORY MAP

HEX ADDRESS

0-1FFF	Console ROM space
2000-3FFF	Memory Expansion Peripheral
4000-5FFF	Peripheral Expansion (predecoded to I/O connector)
6000-7FFF	Cartridge ROM/RAM (predecoded to GROM connector)
8000-9FFF	Microprocessor ROM, VDP, GROM, SOUND and SPEECH select.
A000-BFFF	Memory Expansion peripheral
C000-DFFF	Memory Expansion peripheral
E000-FFFF	Memory Expansion peripheral

MEMORY-MAPPED DEVICES

ADDRESSES	A0	A1	A2	A3	A4	A5	A14	USE
8000	1	0	0	0	0	0	0	Internal RAM (8300-83FF)
8400	1	0	0	0	0	1	0	Sound
8800	1	0	0	0	1	0	0	VDP Read Data
8802	1	0	0	0	1	0	1	VDP Read Status
8C00	1	0	0	0	1	1	0	VDP Write Data
8C02	1	0	0	0	1	1	1	VDP Write Address
9000	1	0	0	1	0	0	0	Speech Read
9400	1	0	0	1	0	1	0	Speech Write
9800	1	0	0	1	1	0	0	GROM Read Data
9802	1	0	0	1	1	0	1	GROM Read Address
9C00	1	0	0	1	1	1	0	GROM Write Data
9C02	1	0	0	1	1	1	1	GROM Write Address

NOTE: Memory-mapped devices at addresses >8000 through >9FFF are only partially decoded. Thus, the devices will respond not only at the base addresses listed above, but also at other addresses within the 1K block. For more information, see Section 8.1 on page 25.

A.4 CRU ALLOCATION

Of the available 4K of CRU bits, the first 1K (addresses 0000-07FE) are used internally in the console. The second 1K (addresses 0800-0FFE) are reserved for future use. The last 1.9K (addresses 1000-1FFE) are reserved for the peripherals to be plugged in the I/O port. A block of 128 CRU bits is assigned to each peripheral as listed below.

CRU ASSIGNMENTS

CRU ADDRESSES	A3	A4	A5	A6	A7	USE
0000-0FFE	0	X	X	X	X	Internal Use
1000-10FE	1	0	0	0	0	Unassigned
1100-11FE	1	0	0	0	1	Disk Controller Card
1200-12FE	1	0	0	1	0	Modems
1300-13FE	1	0	0	1	1	RS 232 (primary)
1400-14FE	1	0	1	0	0	Unassigned
1500-15FE	1	0	1	0	1	RS 232 (secondary)
1600-16FE	1	0	1	1	0	Unassigned
1700-17FE	1	0	1	1	1	HEX-BUS™
1800-18FE	1	1	0	0	0	Thermal Printer
1900-1FFE	1	1	0	0	0	EPROM Programmer
1A00-1AFE	1	1	0	1	0	Unassigned
1B00-1BFE	1	1	0	1	1	Unassigned
1C00-1CFE	1	1	1	0	0	Video Controller Card
1D00-1DFE	1	1	1	0	1	IEEE 488 Controller Card
1E00-1EFE	1	1	1	1	0	Unassigned
1F00-1FFE	1	1	1	1	1	P-Code Card

A.5 INTERRUPT HANDLING

The interrupt available on the I/O port is one of the maskable interrupts of the TMS 9901 Programmable Systems Interface.

9900 INTERRUPTS

INTERRUPT LEVEL	VECTOR LOC. (MEMORY ADDR. IN HEX)	CPU PIN	DEVICE ASSIGNMENT
(Highest Priority)	0000-WSP 0002-PC	RESET	RESET
0	FFFC-WSP FFFE-PC	LOAD	LOAD
1	0004-WSP 0006-PC	--	EXT DEV (9901)

Interrupting is done only on Level 1. The additional interrupts available are implemented on 9901. Interrupt Level 1 is decoded by software to be either (1) VDP vertical sync., (2) 9901 internal timer, or (3) I/O bus generated.

9901 BIT ORGANIZATION

ADDRESS	CRU BIT	9901	PIN	FUNCTION
0000	0	Control		Control
0002	1	$\overline{\text{INT1}}$	17	External Interrupt
0004	2	$\overline{\text{INT2}}$	18	Video Display Processor Vertical Sync Interrupt
0006	3	$\overline{\text{INT3}}$	9	9901 Internal Timer Interrupt, keyboard "=" line, joystick "FIRE"
0008	4	$\overline{\text{INT4}}$	8	Keyboard "Space" line, joystick "Left"
000A	5	$\overline{\text{INT5}}$	7	Keyboard "ENTER" line, joystick "Right"
000C	6	$\overline{\text{INT6}}$	6	Keyboard "0" line, joystick "Down"
000E	7	$\overline{\text{INT7}}$ (P15)	34	Keyboard "FCTN" line, joystick "Up"
0010	8	$\overline{\text{INT8}}$ (P14)	33	Keyboard "SHIFT" line
0012	9	$\overline{\text{INT9}}$ (P13)	32	Keyboard "CTRL" line
0014	10	$\overline{\text{INT10}}$ (P12)	31	Keyboard "Z" line
0016	11	$\overline{\text{INT11}}$ (P11)	30	Not Used As Interrupt
0018	12	$\overline{\text{INT12}}$ (P10)	29	Reserved, High Level
001A-1E	13-15	$\overline{\text{INT13}}$-$\overline{\text{INT15}}$	28,27 and 23	Not Used As Interrupt

9901 I/O MAPPING

ADDRESS	CRU BIT	9901	PIN	FUNCTION
0020	16	P0	38	Reserved
0022	17	P1	37	Reserved
0024	18	P2	26	Bit 2 of Keyboard Select
0026	19	P3	22	Bit 1 of Keyboard Select
0028	20	P4	21	Bit 0 (MSB) of Keyboard Select
002A	21	P5	20	Keyboard (ALPHA LOCK)
002C	22	P6	19	Cassette Control 1 (motor control)
002E	23	P7 ($\overline{\text{INT15}}$)	23	Cassette Control 2 (motor control)
0030	24	P8 ($\overline{\text{INT14}}$)	27	Audio Gate
0032	25	P9 ($\overline{\text{INT13}}$)	28	Mag Tape Out
0034	26	P10 ($\overline{\text{INT12}}$)	29	Reserved
0036	27	P11 ($\overline{\text{INT11}}$)	30	Mag Tape Input
0038-003E	28-31	P12-P15	31-34	Not Used IN I/O Mapping

A.6 ELECTRICAL CHARACTERISTICS

DRIVE CAPABILITY OF I/O SIGNALS

SIGNAL NAME	DRIVER
$\overline{\text{Ø3}}$	74LS244
$\overline{\text{CRUCLK}}$	74LS244
$\overline{\text{WE}}$	74LS244
A0	74LS244
A1	74LS244
DBIN	74LS04
$\overline{\text{MBE}}$	74LS244
$\overline{\text{MEMEN}}$	74LS32
A3-A14	74LS367
D0-D7	74LS245
A15/CRUOUT	74LS244
SBE	74LS03
HOLD	74LS32
$\overline{\text{RESET}}$	74LS04

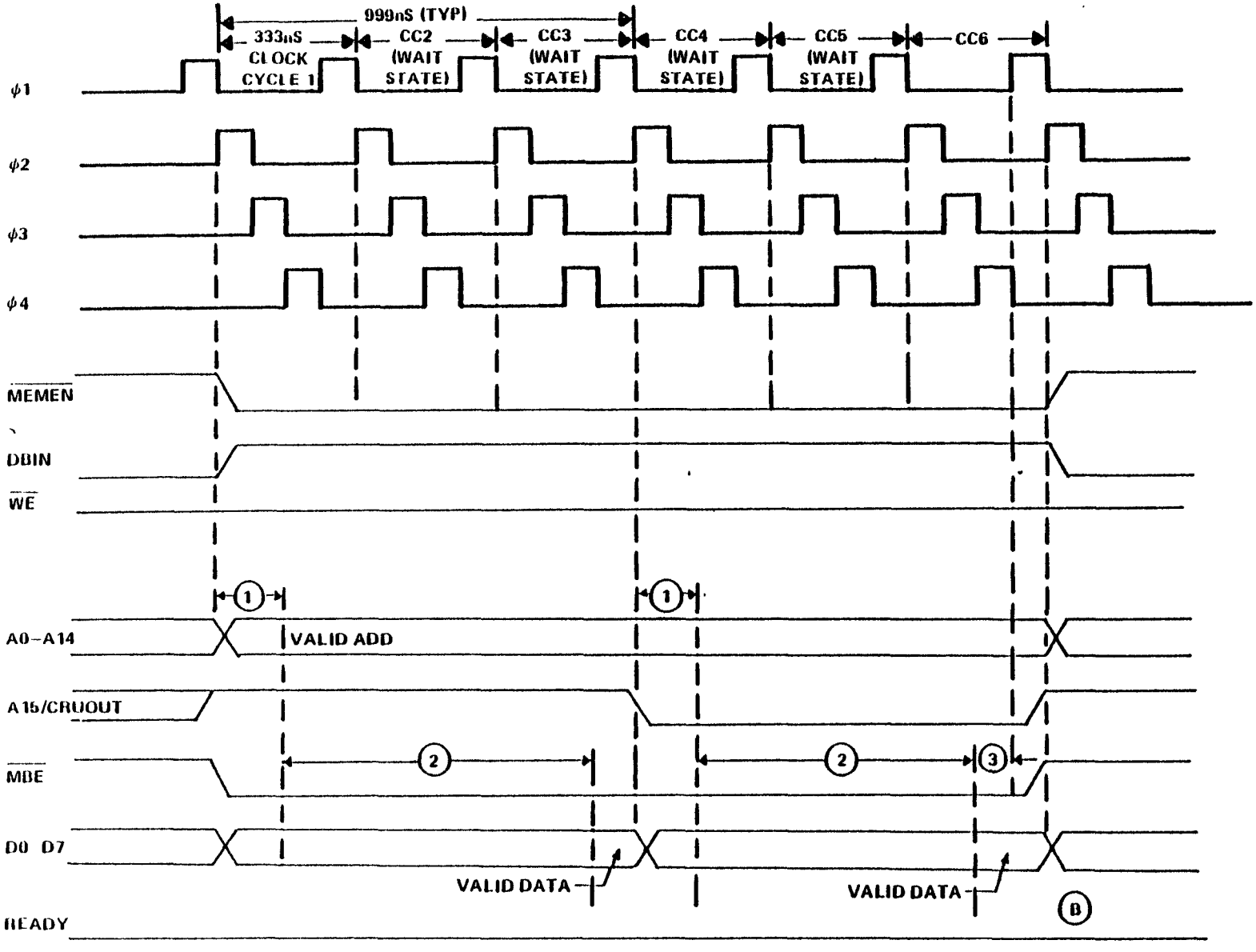
I/O READ

A CPU Read cycle for the external device consists of two 8-bit read cycles (Figure B). The two bytes read are assembled as a 16-bit word before they are presented to the 9900. Shown in Figure B are two 8-bit read cycles with one wait state inserted in each to work with slow memories.

At the beginning of clock cycle 1, $\overline{\text{MEMEN}}$ goes low true and DBIN goes high true. At the same time that $\overline{\text{MEMEN}}$ goes true, the address bus goes active. $\overline{\text{WE}}$ stays high false during the entire cycle.

In order to eliminate noise and glitches (associated with crosstalk and simultaneous switching), a minimum of 100 nanoseconds should be allowed for the address lines to settle. $\overline{\text{MBE}}$ (predecoded from A0, A1, and A2) goes true during the leading edge of Ø2 of clock cycle 1. Data read from the peripherals must be valid 750 nanoseconds after the start of clock cycle 1.

The CPU will look at the full 16-bit data bus during the leading edge of Ø1 of clock cycle 6. Under worst-case conditions, data must be valid 100 nanoseconds before that time.



① SETTLING TIME = 100nS (MIN)
② ACCESS TIME FOR DSR ROM + DATA Ts + DATA BUFF DELAY (PERIPHERAL + MAINFRAME) = 650nS (MAX)
③ SETUP TIME FOR 9900 = 60nS (MIN)

FIGURE B. I/O READ TIMING

I/O WRITE

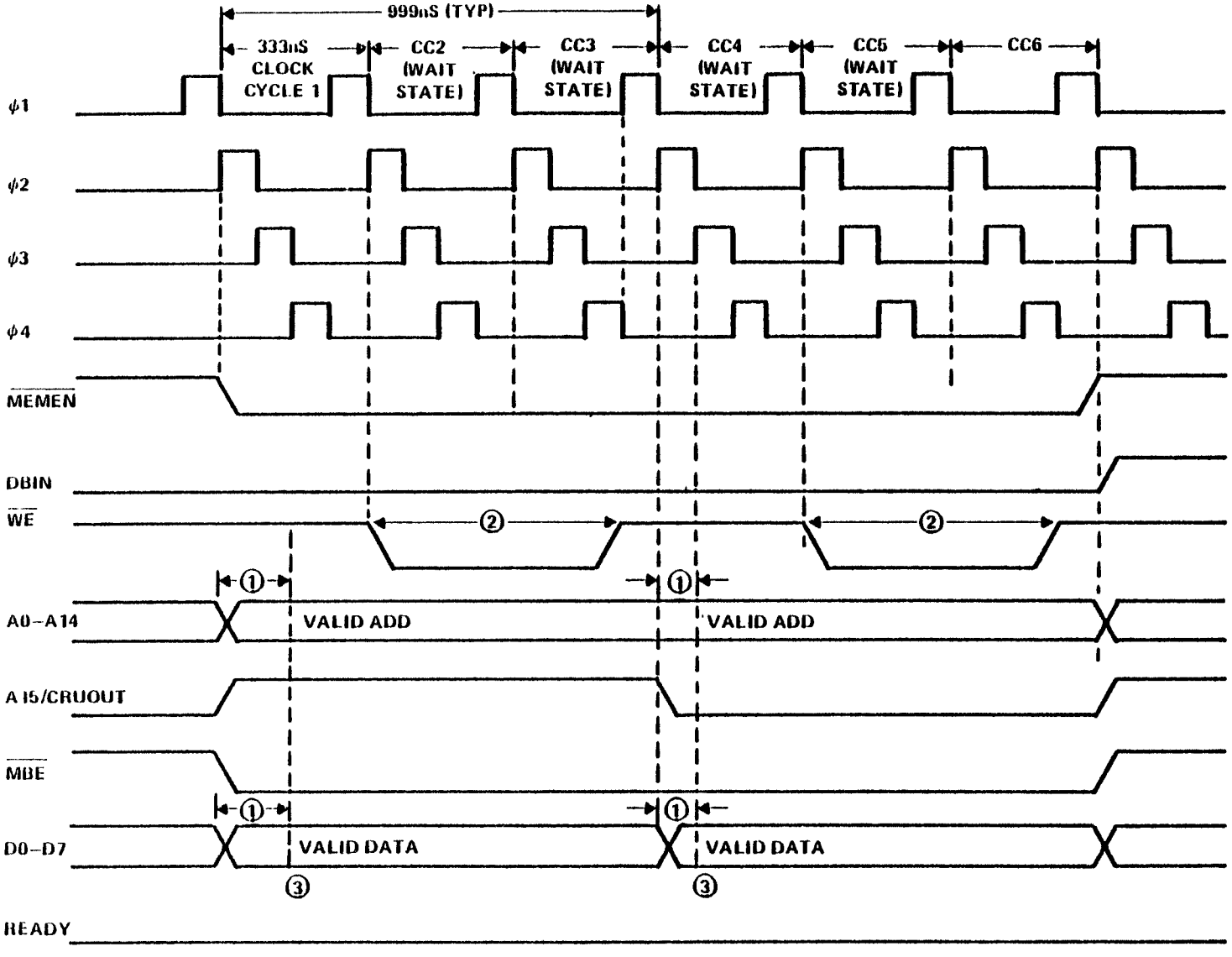
Figure C shows a 16-bit I/O write cycle. As described earlier, it is composed of two 8-bit writes. A write cycle will always be preceded by an ALU cycle.

$\overline{\text{MEMEN}}$ and DBIN go true (low) at the start of the cycle. A time of 100 nanoseconds (min) is allowed for the address lines to settle. $\overline{\text{WE}}$ goes true (low) on the leading edge of ø2 during the wait states and stays true for 666 nanoseconds (TYP).

During a Read or a Write the odd byte (LSBY) is accessed first, then the even byte (MSBY). A15/CRUOUT changes its state 1 microsecond (TYP) after the cycle is initiated. The second 8-bit write cycle is identical to the first 8-bit write. $\overline{\text{MBE}}$ stays true (low) during the entire (1.8 microseconds) cycle.

I/O BUS LOADING

SIGNAL	TOTAL SWITCHING LOAD (pF)	MAXIMUM PERIPHERAL LOAD (pF)
$\overline{\text{D0-D7}}$	210	90
A0-A2	100	90
A3-A14	100	90
A15/CRUOUT	110	100
$\overline{\text{ø3}}$	110	100
$\overline{\text{RESET}}$	100	90
READY	80	70
CRUIN	125	90
$\overline{\text{CRUCLK}}$	100	90
$\overline{\text{MBE}}$	100	90
$\overline{\text{WE}}$	100	90
$\overline{\text{SBE}}$	35	25
DBIN	100	90
$\overline{\text{MEMEN}}$	100	90
HOLD	80	70



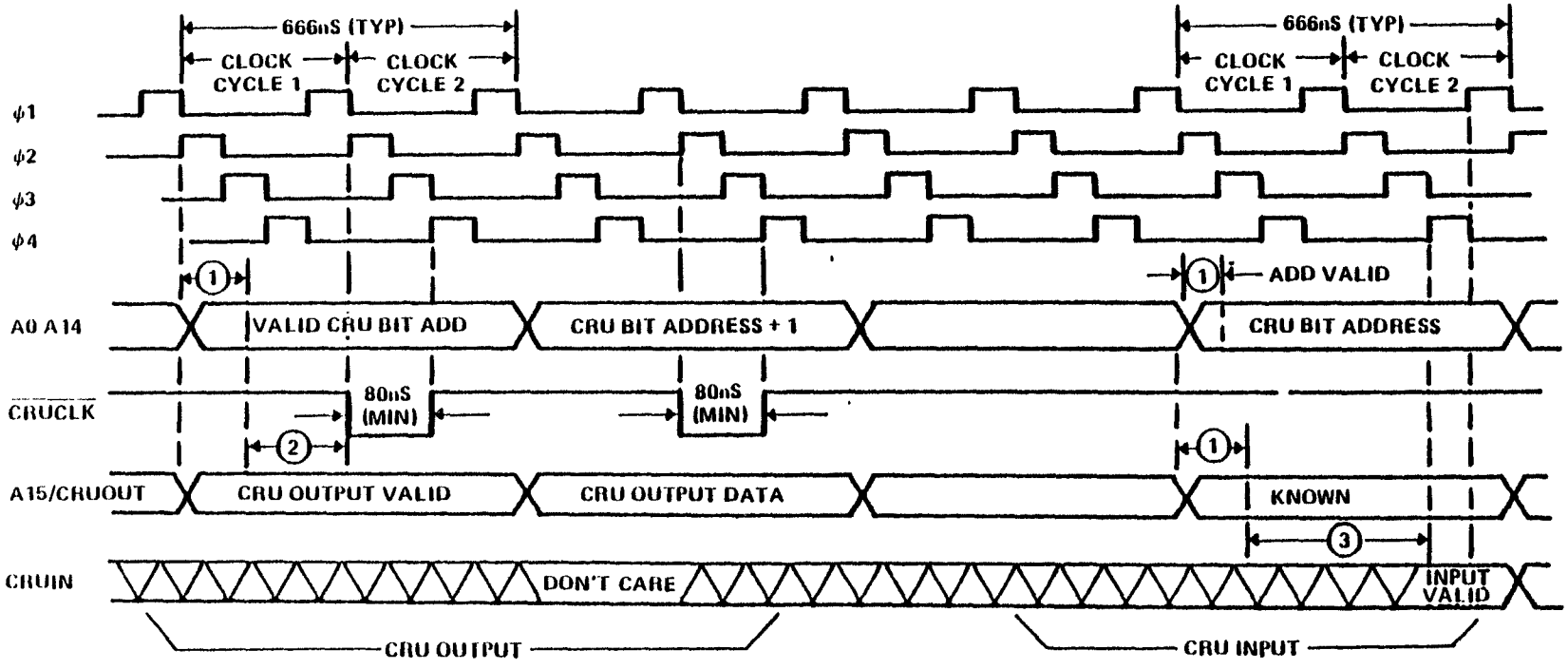
① SETTLING TIME = 100nS (MIN)
② $\overline{WE}$ PULSE WIDTH = 578nS (TYP)
③ VALID DATA

FIGURE C. I/O WRITE TIMING

CRU TIMING

CRU interface timing is shown in Figure D. The CRUOUT cycle is composed of two clock cycles. When placed on the address bus A0 through A14, the CRU bit address is allowed to settle for 100 nanoseconds (min). CRUCLK is a 63 nanoseconds (max) low true signal which occurs on the trailing edge of ø1 of clock cycle 2. CRUOUT data is valid at the start of clock cycle 1 and is latched by the CRUCLK in the respective peripheral.

CRUIN also consists of two clock cycles of 666 nanoseconds (TYP). Again 100 nanoseconds is allowed for the address bus to settle. The CPU samples the CRUIN line on the leading edge of ø1 of clock cycle 2. Data must be valid 40 nanoseconds (min) before that. This implies an access time of less than 400 nanoseconds for CRUIN.



(1) SETTLING TIME = 100nS (MIN)
(2) ADD VALID TO $\overline{\text{CRUCLK}}$ = 233nS (TYP)
(3) ADD VALID TO VALID CRUIN = 400nS (MAX)

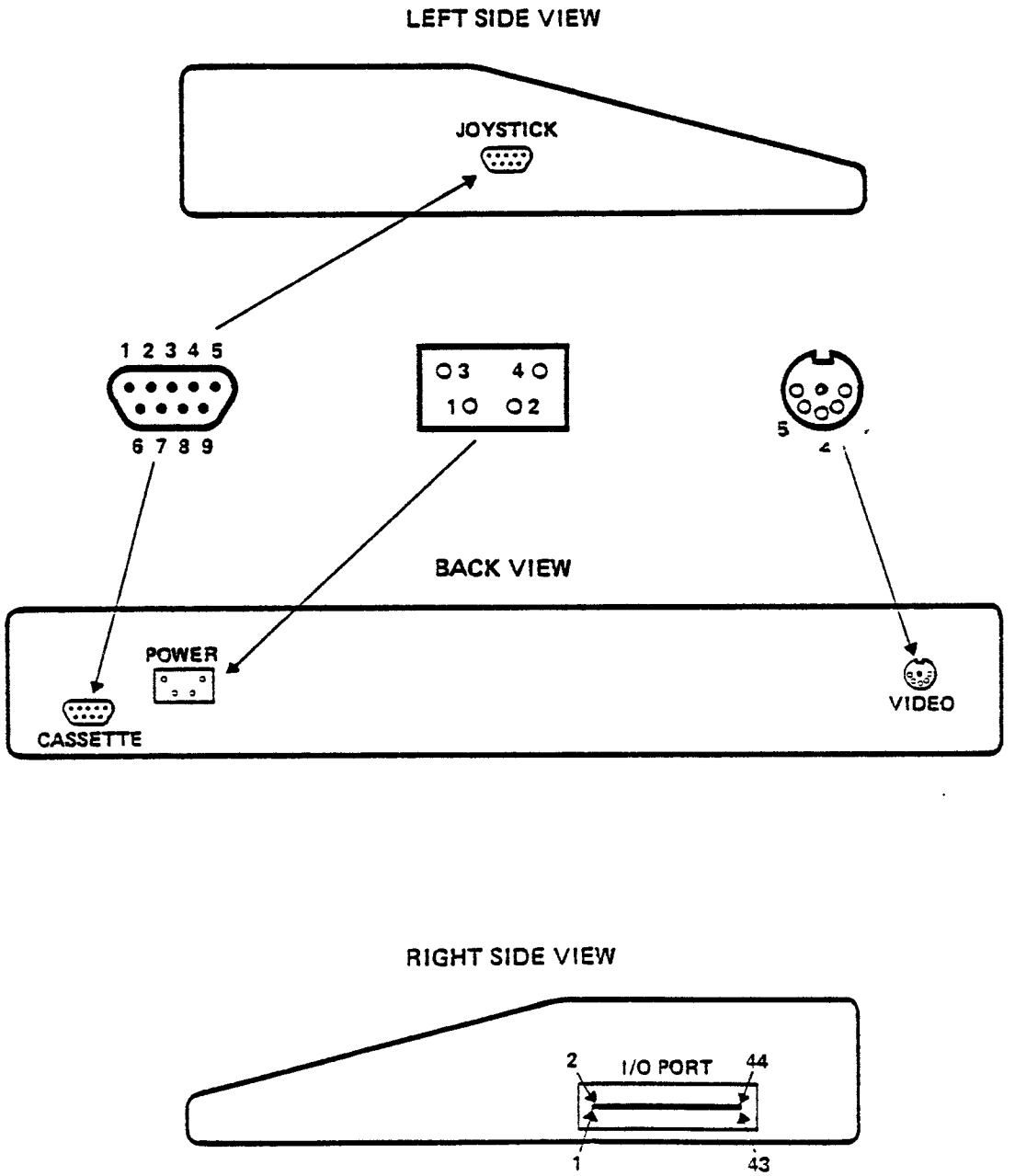


FIGURE E. CONNECTOR PIN IDENTIFICATION DIAGRAM

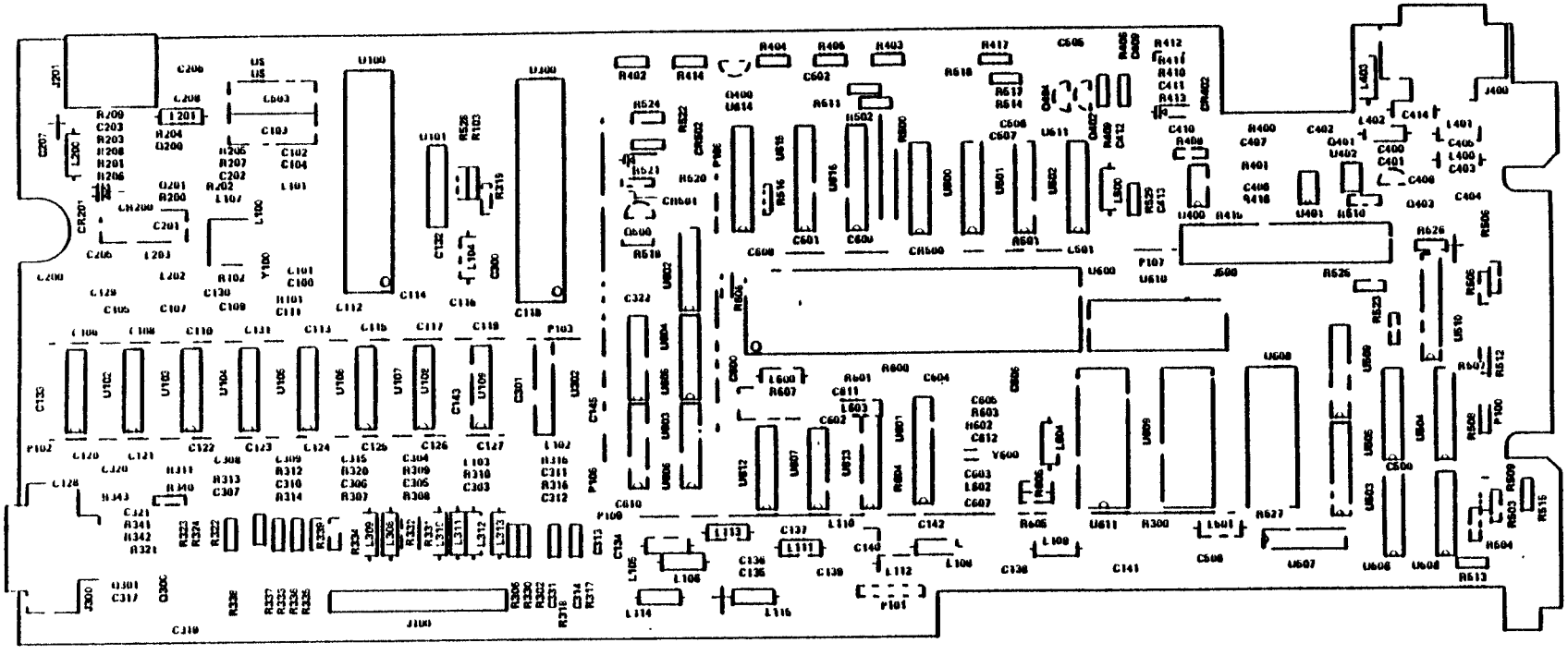
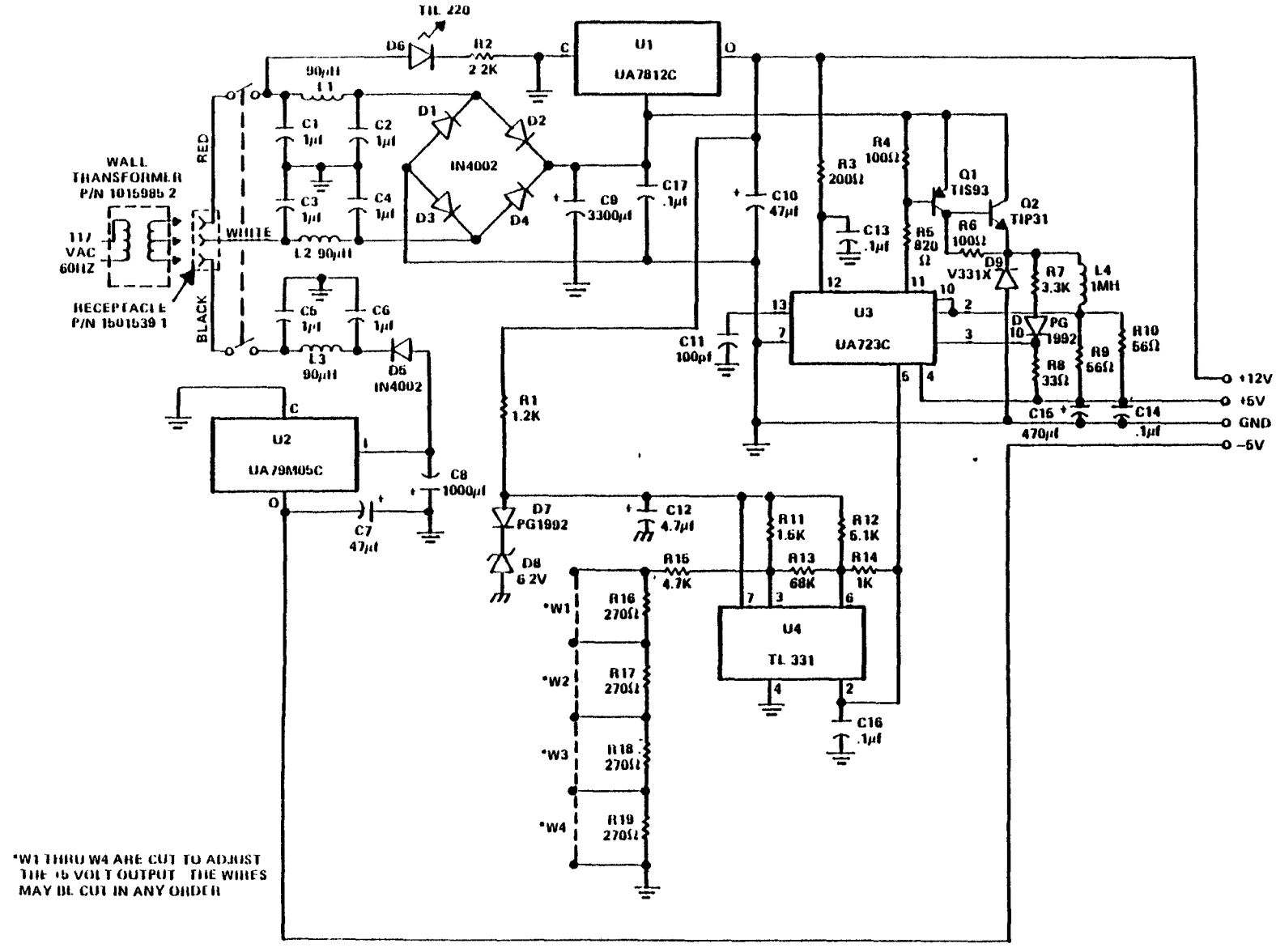


FIGURE F. TI-99/4A LOGIC BOARD COMPONENT LOCATION DIAGRAM



*W1 THRU W4 ARE CUT TO ADJUST THE +5 VOLT OUTPUT THE WIRES MAY BE CUT IN ANY ORDER

FIGURE G. TI-99/4A CONSOLE POWER SUPPLY

SECTION B—INTERFACE CONSIDERATIONS

B.1—CPU REQUIREMENTS

B.1.1 General Comments

The purpose of this section is to outline considerations necessary or advisable for hardware used for interfacing peripherals to the TI-99/4A.

Required TI-99/4A timing is shown on pages 12-17 in Section A of this manual, and Section D includes details concerning the software requirements for peripheral interfacing.

B.1.2 TI-99/4A Memory Space Organization

The memory map of the TI-99/4A is as follows. The allocated spaces are shown in parentheses, and the actual memory space used is shown where applicable. This is notable of the memory-mapped devices (MMD) in the >8000 to >9FFF space. All of these devices reside in 1K blocks, and block decoding is used to select them. Even though they are shown to respond at the base address, they will respond ANYWHERE in the 1K block assigned to them. An example is the 256 byte SRAM which is decoded to respond at a base of >8000. This SRAM repeats at bases of >8100, >8200, and >8300, and software is written to conform to the >8300 base. Note that there is a given amount of space which cannot be used because of the basic space definition and the decoding used on existing products.

8.1.2.1 READ Before WRITE Considerations

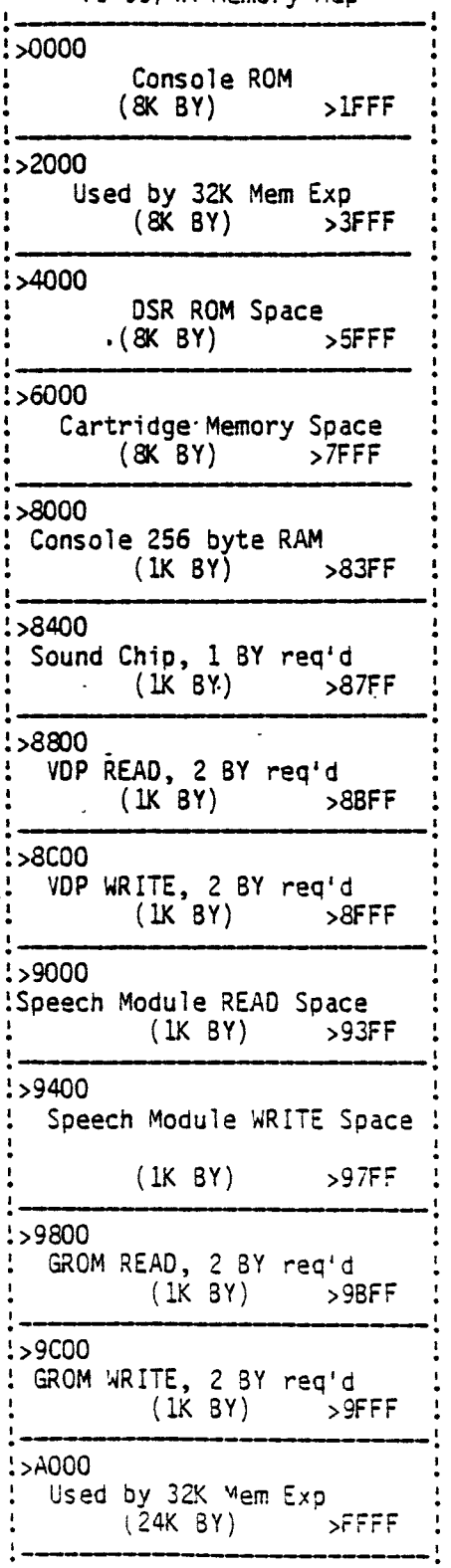
There are different READ and WRITE addresses for most of the memory-mapped devices (MMDs). This is because the TMS 9900 does a READ operation at the destination address prior to writing to it. Many of the MMDs have internal address registers that autoincrement after either a READ or a WRITE operation. This autoincrement characteristic of the MMD may not produce the desired results if it is not taken into consideration when designs or modifications are made.

The READ before WRITE exists because the 9900 is a word-oriented machine from a memory access standpoint. The several byte-oriented instructions are carried out by the machine in a word execution format, and the other byte in the word must not be altered. The machine itself must save the unaltered byte, concatenate the new byte with it, and return the word to memory. The internal logic of the TMS 9900 is designed this way because it was to the designers distinct advantage to do this same READ before WRITE on both byte and word moves.

8.1.2.2 Cartridge Memory Space

The cartridge memory space at >6000 must be treated as if it is logically connected to the cartridge port. The GROM chips are rather poor line drivers at best, and their life on the I/O data bus is limited. Even though the GROM data bus is currently connected directly to the I/O data bus, there is no guarantee that updated versions of the TI-99/4A will have the same connection. The new generation TI console will probably isolate the GROM data bus from the I/O data bus. DON'T TRY TO ACCESS THE CARTRIDGE MEMORY SPACE FROM THE I/O PORT!

TI-99/4A Memory Map



8.1.3 CRU (Communications Register Unit) Space Definition

As with the memory space decoding, the CRU space devices are not fully decoded. Thus, most devices will respond just as well to other addresses in their block as they will to the block base address. By and large, the CRU space is allocated to peripherals with only a very small amount devoted to internal machine control (keyboard scan, cassette control, etc.).

8.1.3.1 Disallowed Instructions.

The control signal "CRUCLK" is also not fully decoded. Because the most-significant three address bus bits are not included in the decode, certain instructions of the TMS 9900 must be disallowed. These are: LREX, CKOF, CKON, RSET, and IDLE..
DO NOT USE THESE INSTRUCTIONS. THE LOGIC BLOCKS FED BY CRUCLK WILL INTREPRET THAT CLOCK SIGNAL AS A CRU OPERATION.

8.1.3.2 CRU OUTPUT Definition.

There is a single restriction on the definition of CRU OUTPUT bits for peripheral use: the base address bit. This bit must be used to "page in" DSR ROM's when SET, and should, where possible, be RESET by the Power-Up RESET line in the I/O Port. Thus, the DSR ROM will be "paged out" by a RESET operation. For more information, see the paragraph on DSR ROM CONSIDERATIONS.

8.1.3.3 CRU INPUT Definition.

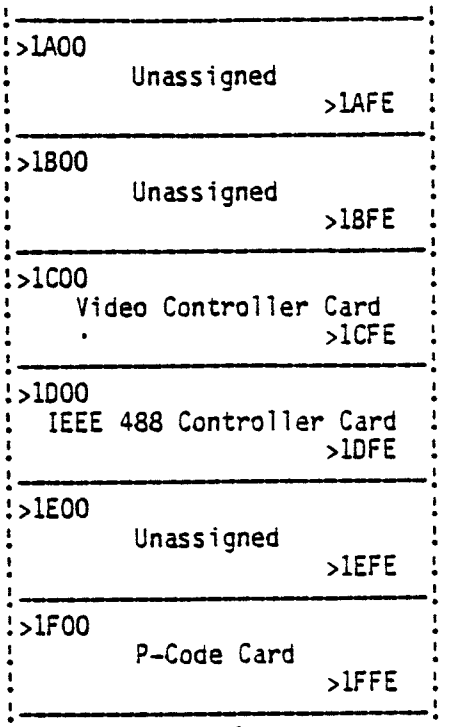
There is no definition for CRU Input lines, and one is free to define them within that peripheral space as desired.

8.1.3.4 Spare CRU Bit Use.

Spare CRU bits may be put to good use as test bits. If spare INPUT and OUTPUT bits are available, the ones used should be chosen so that spare INPUT's may be connected to spare OUTPUT's (each pair displaced by the same amount from the base) to form software FLAG bits. Otherwise, spare INPUTS bits should be tied to VCC or GROUND in some manner that will be verified during the power-up routine execution for that peripneral. Spare OUTPUT bits may be left unconnected or terminated on an etch pad to be connected to automatic test equipment. This is particularly useful when combined with signature test hardware and test software.

CRU MAP

>0000	
CPU TMS 9901 space 32 lines required	>03FE
>0400	
Unassigned	>0FFE
>1000	
For test equipment use on production line (128 Lines)	>10FE
>1100	
Disk Controller (128 Lines)	>11FE
>1200	
Modem	>12FE
>1300	
Primary RS232	>13FE
>1400	
Unassigned	>14FE
>1500	
Secondary RS232	>15FE
>1600	
Unassigned	>16FE
>1700	
HEX-BUS™	>17FE
>1800	
Thermal Printer	>18FE
>1900	
EPROM Programmer	>19FE


>1A00
Unassigned
>1AFE
>1B00
Unassigned
>1BFE
>1C00
Video Controller Card
>1CFE
>1D00
IEEE 488 Controller Card
>1DFE
>1E00
Unassigned
>1EFE
>1F00
P-Code Card
>1FFE

8.1.4 TI-99/4A Peripheral Polling

The TI-99/4A polls the I/O Port for the presence of peripherals on either a software restart or a hardware RESET. This polling starts in the CRU space at >1000, and continues in increments of >100 until >1F00 has been polled. Power-up routines will be executed for each peripheral as its DSR ROM is polled (if the DSR header indicates that a power-up sequence is required).

8.1.5 Indicator LED

An indicator LED shall be provided, and shall be driven by a unique CRU OUTPUT bit. If all CRU OUTPUT bits have been defined, it shall be acceptable to connect the LED drive to the DSR ROM page bit. This LED provides visual feedback to the user that the peripheral card is present and active. All cards presently marketed by TI must provide this indicator. The LED is amber or yellow.

8.1.6 DSR ROM Considerations

The DSR ROMs for all peripherals must be based at >4000 and may be contiguous through >5FFF. These ROMs must not respond unless the respective CRU DSR ROM page bit is set ON (set to a high level with either a SBO instruction or "1" data in a LDCR instruction). It is the responsibility of the CPU to insure the proper control of all of the DSR ROM page bits.

8.1.6.1 Extended DSR ROM Techniques.

ROM Space in excess of 8K bytes may be obtained by bringing sequential 8K blocks into the DSR ROM page. Secondary blocks (pages) may be placed anywhere in available TI-99/4A Memory Space. The secondary page bit may be obtained from the assigned CRU space to that peripheral (as was done on the P-Code PCB) or by writing to a ROM address and then decoding that condition to strobe a flip-flop. The latter method was used on the Extended Basic cartridge to page the upper 4K half of that space. Either a data bus or an address bus bit may be used for data to the FF. Both schemes provided a nonpaged 4K base ROM, and paged the upper 4K half of the space.

B.2—EXPANSION UNIT REQUIREMENTS

B.2.1 System Bus Requirements

There are only two design considerations that must be understood to interface with the system bus. One controls a Data Bus driver on the TI-99/4A end of the connecting cable, and the other is used during printed circuit board (PCB) burn-in.

B.2.1.1 Remote Data Bus Driver.

As has been previously noted, there are additional requirements for interfacing through the Peripheral Expansion Unit (PEU). Regardless of the processor driving the PEU, a signal must be provided to indicate to the interface that a memory cycle is being requested for a PCB in the PEU. This signal is termed "RDBENA*", and is used to enable Data Bus drivers on the TI-99/4A end of the cable. It must function as an open-collector signal, but a tri-state gate may be used instead if the input is grounded and the tri-state controller is connected in parallel with that gate for the Data Bus Buffer.

B.2.1.2 Burn-In Consideration.

A single high true line shall be provided to enable the PCB to respond to both memory and CRU accesses. This line is held at a high level in the PEU, but is driven to allow a parallel connection of 16 PCBs during burn-in. When low, this signal shall disable the PCB from driving any of the system memory and CRU bus lines. The interrupt sensing has been excepted from this in the past.

B.2.2 Power Allocation Assumptions

The following is a guide for maximum load current a PCB should present to the PEU.

- 250 ma on the +15v unregulated bus.
- 500 ma on the +8v unregulated bus.
- 30 ma on the -15v unregulated bus.

8.2.3 System Bus Pin Definition

.100" PIN TO PIN SPACING, ATTLEBORO CONNECTOR

PIN NO.	MNEMONIC	FUNCTION
1		+5v 3-T Regulator supply voltage
2		+5v 3-T Regulator supply voltage
3	GND	Logic Ground
4	READY A	System READY
5	GND	Logic Ground
6	RESET*	Active LOW system-driven RESET
7	GND	Logic ground
8	SCLK	System clock
9	LCP*	Second generation CPU indicator. 0=Second generation CPU; 1=TI-99/4
10	AUDIO	Input audio
11	RBDENA*	Active LOW remote data bus driver Enable control line
12	PCBEN	Active HIGH PCB enable for burn-in
13	HOLD*	Active LOW CPU HOLD request Second generation CPU only
14	IAQHA	IAQ HOLDA logical OR Second generation CPU only
15	SENILA*	Interrupt Level A Sense Enable
16	SENILB*	Interrupt Level B Sense Enable Second generation CPU only
17	INTA*	Active LOW Interrupt Level A
18	LOAD*	Low for TI-99/4 Memory Expansion, High for linear memory space. Second generation CPU only. TMS9900 LOAD* input
19	D7	System DATA Bus, LSB
20	GND	Logic Ground
21	D5	System DATA Bus
22	D6	System DATA Bus
23	D3	System DATA Bus
24	D4	System DATA Bus
25	D1	System DATA Bus
26	D2	System DATA Bus
27	GND	Logic Ground
28	D0	System DATA Bus, MSB
29	A14.A	Address Bit
30	A15/CRUOUT.A	Address Bit, LSB
31	A12.A	Address Bit
32	A13.A	Address Bit
33	A10.A	Address Bit

System Bus Pin Definition (Continued)

PIN NO.	MNEMONIC	FUNCTION
34	A11.A	Address Bit
35	A08.A	Address Bit
36	A09.A	Address Bit
37	A06.A	Address Bit
38	A07.A	Address Bit
39	A04.A	Address Bit
40	A05.A	Address Bit
41	A02.A	Address Bit
42	A03.A	Address Bit
43	A00.A	Address Bit
44	A01.A	Address Bit
45	AMB.A	Address Bit, normally HIGH
46	AMA.A	Address Bit, normally HIGH
47	GND	Logic Ground
48	AMC.A	Address Bit, MSB normally HIGH
49	GND	Logic Ground
50	CLKOUT*	Active LOW CPU CLOCK
51	CRUCLK.A*	Active LOW CRU Output Clock
52	DBIN.A	Data Bus Dir'tn, HIGH is CPU READ
53	GND	Logic Ground
54	WE.A*	LOW true CPU Write Enable
55	CRUIN	HIGH true CRU Input data
56	MEMEN.A*	Active LOW memory request
57		-12v 3-T Regulator supply voltage
58		-12v 3-T Regulator supply voltage
59		+12v 3-T Regulator supply voltage
60		+12v 3-T Regulator supply voltage

B.3—DESIGN AND DEVELOPMENT REQUIREMENTS

This listing covers both hardware and PCB design rules.

- Each PCB must be completely disabled by an active low disable (active high enable). The disable must disable both memory and CRU functions on the PCB and is used for burn-in purposes.

- The +12v, -5v, and -12v power etches shall be double spaced from other etches where possible. This minimizes a TTL to power line short or a power etch to power etch short.

- Signals such as CRUIN that are taken very far from the system I/O connector shall be guarded with GROUND etch. Where practical, these signals should be buffered with tri-state buffers.

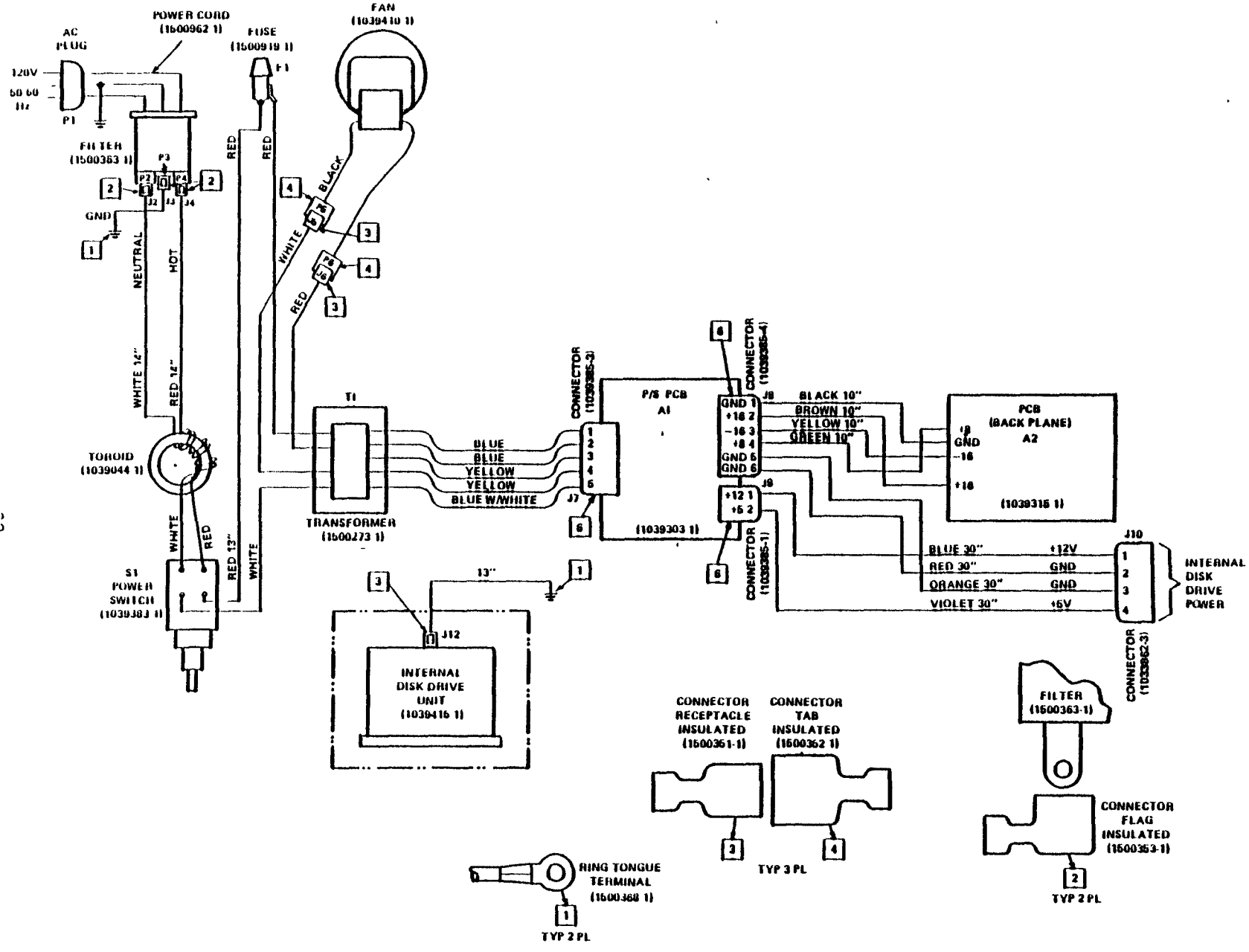
- Buffers to drive the bus must be physically close to the I/O Bus. The prototyping PCBs have most of this already in place, but signals such as READY and CRUIN do not have them because they are not always used. A connection to the system bus constitutes a stub connection, which generally obeys transmission line theory. These stubs should be kept as short as is practical.

- All 19 system address bits shall be included in the memory space decode for that peripheral. Assume AMC, AMB, and AMA to be in the high state.

- Address lines A00 through A07 shall be included in the CRU Space decode even though the TI-99/4A assumes A00 thru A02 to be zero.

- There are two levels on which to interrupt, but the TI-99/4A supports only one (INTA*). THIS IS THE ONE YOU MUST USE. Interrupt level status bits are defined by the Personal Computer PCC at Texas Instruments, and for the moment are not sensed by the TI-99/4A. If they were to be sensed, the TI-99/4A would cause a line to go low (SENILA*), which tells the PCB logic to gate its status bit to the system data bus.

- As a design aid, the Personal Computer Group has available a prototyping printed circuit board (PCB). This PCB will accomodate about 36 16-pin DIP sockets, and has the necessary system bus buffers in place with etch connections to the system I/O pins. The PCB designer need only to hook the buffered signal to his logic design. Schematics of this PCB are available (TI part number 1039334) and standard TI-99/4A timing diagrams should be used for the PEU PCB Design



PERIPHERAL EXPANSION UNIT SCHEMATIC DIAGRAM

Section C—FILE MANAGEMENT SPECIFICATIONS

C.1 Introduction

This document contains a complete specification for the file management system of the TI-99/4A Home Computer.

The text in this document has been completely revised to reflect changes that have been made to facilitate new options in the TI-99/4A Home Computer.

The TI-99/4A Home Computer will support all file features described in this document. Some minor additions have been made concerning the status (C.3.2.10) and the error code section (C.3.4) to reflect changes for the TI-99/4A Disk Peripheral. These changes do not affect any DSR that currently exists.

Section C.4.3 has been provided to give information to the GPL applications programmer on how to access DSRs and for the peripheral programmer on how the information is passed on a DSR.

C.2 I/O Handling

The approach used in the TI-99/4A Home Computer File Management System has been that all devices should look the same to an application program. Therefore when peripherals are added to the computer, it should not affect the BASIC interpreter. Only the peripheral driver (Device Service Routine, or DSR) will have to be added to the software.

All peripherals for the TI-99/4A Home Computer, with the exception of the keyboard/display, are considered equivalent by an application program, within physical limitations (reading data from a thermal printer is clearly impossible). These physical limitations are determined in the device service routine and returned to the application program as an error condition.

The TI-99/4A Home Computer File Management System supports both random and sequential access files. Both file types use the same supervisor call mechanism in order to insure a high degree of device independence. Hardware devices (such as a line printer) are accessed as sequential files except that no file name is appended to the device name.

The following sections describe the file structures available to the BASIC interpreter, how to create and delete files, and how to perform file I/O.

Two different kind of file organizations are supported:

- o Sequential files
- o Relative record files (random access)

C.2.1 File Organization and Use

The following paragraphs discuss the file organization and use for each of the two file types.

C.2.1.1 Sequential Files

Sequential files are both used for disk-based files and for I/O to other devices. They consist of fixed or variable record length files whose records are always accessed serially, like the output to the thermal printer.

C.2.1.2 Relative Record Files

Relative record files are also called random access files because, unlike a sequential file, records may be accessed in an arbitrary order. Therefore relative record files can only be supported on random access media such as floppy disks.

The records within a relative record file are addressed by a unique record number. To access record X, the value X has to be placed in the appropriate field of the I/O Peripheral Access Block. The range of record numbers is from zero to one less than the number of records in the file. Records in a relative record files are a fixed length, specified at file creation.

Records in a relative record file can also be accessed in a sequential way by specifying only the first record in the sequence. The supervisor then automatically updates the record number each time after a record has been read.

C.2.2 File Management Overview

C.2.2.1 Terminology

A file consists of a collection of data groupings called logical records. This division of the file into logical records does not necessarily correspond to the physical division of data on the medium (like a sector on a disk). Thus there are two types of records:

- Logical records—The data grouping of a file as seen by an application program.
- Physical records—The buffers physically transferred between memory and medium.

Relative file I/O from a program is done on a logical record with a fixed length. This enables the system to compute the actual location of any logical record relative to the beginning of file.

Sequential files allow both fixed and variable length records.

When a file is created, the logical record size must be specified. For relative record files this size must be exact. For sequential files the specification indicates an upper limit for the size of a variable record, or the exact length of a fixed-length record. In case a zero is specified for either filetype, the DSR must select a default for the record size.

The physical record size for any medium is specified within the DSR and is implementation dependent.

C.2.2.2 File-Type Attribute

The file-type attribute specifies the format in which the data in the file is represented. The two file types are:

- DISPLAY—Displayable or printable character strings. Each data record corresponds to one print line.
- INTERNAL—Data in INTERNAL machine format.

The file-type attribute is internal to the application program. It is merely stored and passed on by the DSR as a distinction between two data types, without affecting the actual data stored.

C.2.2.3 Mode of Operation

A file is opened for a specific mode of operation, specified in the OPEN I/O call. The four modes of operation are:

- INPUT - The contents of the file may be read, but may not be altered.
- OUTPUT - The file is being created. Its contents may be written but not read.
- UPDATE - The contents of the file may be both written and read. Note that this mode of operation is only be supported by random access devices.
- APPEND - New data may be added at the end of the file, but the contents of the file may not be read

Each DSR decides whether or not a specific mode for an I/O operation can be accepted by the corresponding device.

C.2.2.4 Temporary Files

In the subsets of TI standard BASIC used for the TI-99/4A Home Computer, the file-life attribute is not implemented. Therefore there is no need for the File Management System to support temporary files, and all files are permanent by definition.

C.3 Implementation

As mentioned in section C.2, the DSRs should present a uniform interface between the File Management System and the peripherals. This section will give details about implementing this interface.

C.3.1 Peripheral Access Block Definition

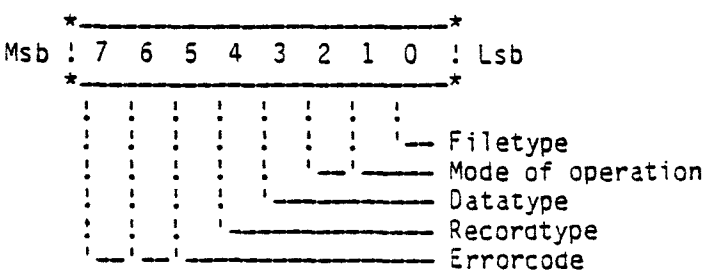
All DSRs are accessed through a Peripheral Access Block (PAB). The definition for the PAB is the same for every peripheral. The only difference between peripherals, as seen by any application program, is that some peripherals will not support every option provided for in the PAB.

All PAB's are physically located in VDP RAM. They are created before the OPEN call and are not to be released until the I/O has been closed for that device or file.

Figure C.3.1.1 shows the layout of a PAB. The PAB has a variable length, depending upon the length of the file descriptor.

The meaning of the bytes and bits within the PAB is explained below.

Byte	Bit	Meaning
0	-	I/O opcode— Contains opcode for the current I/O call. A description of the valid opcodes are given in section C.3.2.
1	-	Flagbyte/status— All the information the system needs about file-type, mode of operation, and data-type is stored in this byte. The meaning of the bits within this flagbyte is:



Byte	Bit	Meaning
1	0	Filetype--Indicates file-type. 0 = Sequential file 1 = Relative record file
	1,2	Mode of operation--Indicates operation mode for which file has been opened. 00 = UPDATE 01 = OUTPUT 10 = INPUT 11 = APPEND
	3	Datatype--Indicates type of data stored in the file. 0 = DISPLAY type data 1 = INTERNAL type data
	4	Recordtype--Indicates type of record used. 0 = Fixed length records 1 = Variable length records
	5-7	Error code--These three bits indicate, in combination with the I/O opcode, the error type that has occurred (0 = no error).
2,3	-	Data buffer address--Address of the data buffer the data has to be written to or read from.
4	-	Logical record length--Indicates the logical record length for fixed length records, or the maximum length for a variable length record (see flagbyte).
5	-	Character count--Number of characters to be transferred for a WRITE opcode, or the number of bytes actually read for a READ opcode (not equivalent to INPUT and OUTPUT mode).
6,7	-	Record number--Only required if the file opened is of the relative record type. Indicates the record number the current I/O operation is to be performed upon (this limits the range of record numbers 0-32767). The highest bit will be ignored by the DSR.

Byte	Bit	Meaning
8	-	Screen offset--Offset of the screen characters in respect to their normal ASCII value.
9	-	Name length--Length of the file descriptor following the PAB.
10+	-	File descriptor--Device name and, if required, the filename and options. The length of this descriptor is given in 9.

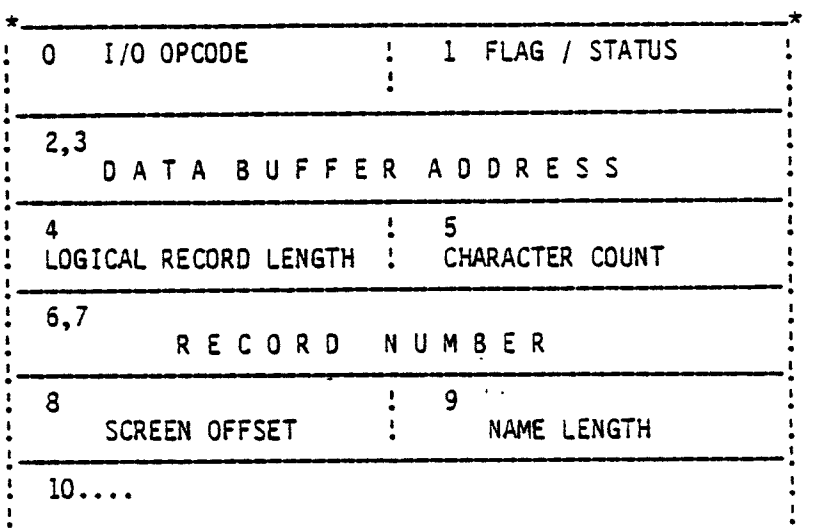


Figure C.3.1.1 PAB Layout

C.3.2 I/O Opcodes

This section describes the valid opcodes that can be used in a PAB. These valid opcodes are shown in Figure C.3.2.1.

The following section will describe the general actions invoked by an I/O call with each of the I/O opcodes. Each I/O call returns any error codes in the FLAG/STATUS byte of the PAB.

Opcode	Meaning
00	OPEN
01	CLOSE
02	READ
03	WRITE
04	RESTORE/REWIND
05	LOAD
06	SAVE
07	DELETE
08	SCRATCH RECORD
09	STATUS

Figure C.3.2.1 I/O Opcodes

C.3.2.1 OPEN

The OPEN operation should be performed before any data transfer operation except LOAD or SAVE. The file remains open until a CLOSE operation is performed. The mode of operation for which the file has to be opened should be indicated in the flagbyte of the PAB. In case this mode is UPDATE, APPEND, or INPUT, the record length will be returned in byte four. Any given non-zero record length will be checked against this stored length. For OUTPUT the record length can be specified, or a default can be used by specifying record length zero.

For any device, an OPEN operation must be performed before any other I/O operation. The DSR need only check the record length and I/O mode on an OPEN. Changing I/O modes after an OPEN may cause unpredictable results.

C.3.2.2 CLOSE

The CLOSE operation informs the DSR that the current I/O sequence to that DSR has been completed.

After the CLOSE operation, the space allocated for the PAB may be used for other purposes. As long as a PAB is connected to an active device, the contents of that PAB must be preserved.

If file or device is opened for OUTPUT or APPEND mode, an EOF (end of file) record is written to the device or file before disconnecting the PAB.

C.3.2.3 READ

The READ operation reads a record from the selected device and copies the bytes in the specified buffer. The buffer address is specified in PAB entry 2 and 3; the buffer size is specified in PAB entry 4. The actual number of characters stored is returned in PAB entry 5 (CHARACTER COUNT). If the length of the input record exceeds the buffer size, the remaining characters will be discarded.

C.3.2.4 WRITE

The WRITE operation writes a record from the buffer specified in the PAB to the specified device. The number of bytes to be written is specified in byte five of the PAB.

C.3.2.5 RESTORE/REWIND

The RESTORE/REWIND operation repositions the file READ/WRITE pointer either to the beginning of the file, or, in the case of a relative record file, to the record specified in bytes six and seven of the PAB.

A RESTORE can only be used if the file is opened for INPUT or UPDATE mode. For relative record files, a RESTORE could be simulated in any I/O mode by specifying the record at which the file has to be positioned in bytes six and seven of the PAB. The next I/O operation then automatically uses the indicated record.

C.3.2.6 LOAD

The LOAD operation loads an entire memory image from an external device or file into VDP RAM. All the control information the application program needs should be concatenated to the program image. No intermediary buffers are used. The entire memory image is dumped starting at the specified location.

The LOAD operation is a stand-alone operation, i.e. the LOAD operation is used without a previous OPEN operation.

For the LOAD operation, the PAB needs to contain the following information:

Bytes 2,3—Start address of the memory dump area

Bytes 6,7—Number of bytes available.

Aside from the I/O opcode and the file descriptor, no other PAB entry is required for a LOAD operation.

C.3.2.7 SAVE

SAVE is the complementary operation for LOAD. It is used for writing memory images to a device or file. All necessary control information should be linked to the memory image, so that the information plus program image use on contiguous memory area. Again, only a small part of the PAB is used . Aside from the usual information (I/O opcode and file descriptor), the PAB contains:

Bytes 2,3—Start address of the memory area

Bytes 6,7—Number of bytes to be saved.

C.3.2.8 DELETE

The DELETE operation deletes the specified file from the specified device. This operation also CLOSEs the I/O sequence.

C.3.2.9 SCRATCH RECORD

The SCRATCH RECORD operation scratches the specified record from the specified relative record file. The record to be scratched is specified in bytes six and seven of the PAB. This operation will cause an error for sequential files and devices. This operation is not currently supported by any device.

C.3.2.10 STATUS

This information can be asked for at any time, although some information is only meaningful if a file has been opened for access.

To indicate the current status of the file, byte eight (SCREEN OFFSET) is used. Upon the DSR call byte eight should contain the usual screen characters base address. The DSR can use only this byte and is guaranteed not to destroy any other entry in the PAB.

The meanings of the bits within byte eight after return from the DSR are:

0 Logical end of file. If this bit is set, the file is at the end of its previously created contents. This is usually the case if the file has been opened for APPEND mode. Depending upon the mode of operation for which the file has been opened, data can still be written to the file (APPEND, OUTPUT or UPDATE mode). However, a "read" operation will cause an ATTEMPT TO READ PAST EOF error to occur.

1 Physical end of file. If set, no more data can be written, since the physical limits of the device have been reached. Generally this means an end of medium has been detected on the device.

2 Record type. If set, the record type is VARIABLE length. If cleared the record type is FIXED length.

3 Filetype. If set, the file is a program file. If cleared, the file is a data file.

4 Data type. If set, the data type is binary (INTERNAL). If cleared the data type is ASCII (DISPLAY) or file is program file.

5 Reserved for the future use. Fixed to zero in the current peripherals.

6 PROTECT flag. If set, the file is protected against modifications. If cleared, the file is not protected.

7 File requested does not exist. A device will not set this bit, which means that on a device, any file exists.

Bits two--seven are valid if the file has not previously been opened. Bits zero and one can only be used for files that are currently opened for access. A file that is not currently open for access should indicate a zero in these two bits.

C.3.3 Error Codes

The File Management System supports a number of error codes. They are:

Error Code	Meaning
0	Bad device name; the device indicated is not in the system.
1	Device is write protected.
2	Bad open attribute such as incorrect file type, incorrect record length, incorrect I/O mode, or no records in a relative record file.
3	Illegal operation; i.e. an operation not supported on the peripheral or a conflict with the OPEN attributes.
4	Out of table or buffer space on the device.
5	Attempt to read past the end of file. When this error occurs, the file is closed. Also given for non-extant records in a relative record file.
6	Device error. Covers all hard device errors such as parity and bad medium errors.
7	File error such as program/data file mismatch, non-existing file opened in INPUT mode, etc.

C.4 DSR Operations

This section describes how a variety of DSRs should react on the different I/O calls. It also discusses detailed software operations descriptions such as available registers and memory.

C.4.1 DSR Actions and Reactions

In the Home Computer File Management System, several assumptions are made about the way in which DSRs should react on conditions such as errors, special I/O modes, defaults, etc. This section is intended to explain the reactions of a DSR on these conditions.

C.4.1.1 Error Conditions

C.4.1.1.1. Non-existing DSRs

If a non-existing DSR is called by an application program, the File Management System will automatically return with the COND bit set. In this case, no DSR has actually been called, so the error code will show no errors.

The DSR search mechanism of the File Management System takes care of searching for the requested DSR. It tries to match the file descriptor to the DSR entries in the system. The matching algorithm matches the end of the descriptor, or the first period, whichever comes first. This enables the applications program to add special information for the DSR in the file descriptor, such as filename, BAUD-rate, print-width, character set, etc.

C.4.1.1.2 DSR-detected Errors

DSR-detected errors (see section C.3.4) should be indicated in the flag byte of the PAB. It is the application program's responsibility to clear this flag byte before every I/O call, and check it after the I/O call. This type of error is NOT indicated with the COND bit.

The DSR may provide additional information about the error type in the I/O opcode byte, although it is good practice not to destroy the least significant four bits of this byte, since they specify the I/O call.

At no time should the DSR use bits zero—four of the flagbyte for error indication, since these bits might contain vital system information about the file device.

C.4.1.2 Special I/O Modes

To enable the application program to use special device-dependent functions, the File Management System DSR search algorithm only uses a well-defined part of the file descriptor for its search (see section C.4.1.1.1). The remainder of the descriptor may be used to indicate special device-related functions such as BAUD rate, print width, etc. It is advisable for a DSR to ignore descriptor parts it doesn't recognize, so that the same application program might be used for different devices. In the latter case it would handle specific device-dependent functions only if the device used was capable of performing them. If the descriptor dealt with a function not applicable to a particular device, the DSR would not recognize the descriptor and would ignore it.

An example of a special I/O mode descriptor could be

RS232.BAUDRATE=1200.DATABITS=7.CHECKPARITY.PARITY=ODD

C.4.1.3 Default Handling

Sometimes, especially if a file is opened for UPDATE, INPUT, or APPEND, it is useful to provide a default value for the record length. In the above cases, the application program will usually use the value specified on file creation. Therefore, if the application program does not provide a value for the record length, the DSR should provide this value for it. In case the application program does provide a record length, the DSR should check this value against the value given on record creation. An error should be indicated in case of incompatibilities between stored and provided record length. If logical record length is given as zero on OPEN, the DSR must provide a default record length.

C.4.2 Memory Requirements

Because of the limited amount of register memory (256 bytes of RAM), the register usage for DSRs has to be restricted to the following registers/memory to avoid interference with application programs

- Registers R0—R10 of the calling workspace.
- Memory locations >0A through >0F are available if the DSR is called in a non interrupt-driven mode, i.e. through a standard DSR entry.
- A standard scratch area of 36 bytes, at locations >4A through >6D, has been assigned for DSR usage.

The base address for CPU memory in the TI-99/4A Home Computer is >8300. To allow for future changes in this base address, it will be derived from the given value of the workspace pointer. This means the loss of one of the workspace registers for this purpose.

C.4.3 GPL Interface to DSRs

The GPL interpreter interfaces to DSRs through the monitor. The GPL program that wants to access a DSR has to use the following GPL CALL sequence:

```
CALL >10
DATA 8
```

This will cause the monitor to start searching for a DSR with the same name as the string pointed to by CPU location >56. This search routine will stop comparing names at the end of the given string or at the first imbedded period, wnichever comes first.

On the DSR side, CPU location >56 is left pointing at the first character behind the DSR name, i.e. a period or an end of string. CPU location >54 contains the DSR name length (one word). To get the start address of the PAB in VDP RAM, the following formula has to be computed:

```
CPU(>56) - CPU(>54) - >0A
```

The result will point at the I/O OPCODE entry in the PAB.

It is up to the DSR to check for any switches in the name. For this purpose the length of the PAB name string has been given in the PAB. Comparing this length against the length given in CPU location >54 will show if the user specified more than just the DSR name.

C.5 Linkage to BASIC

This section describes the way the BASIC versions of the Professional Computer and the Home Computer are linked to the File Management System.

This section also describes how to access PAB's from GPL subroutines that are callable from BASIC and assume a PAB link structure has been set up by BASIC.

C.5.1 BASIC PAB modifications

Aside from the control information contained within the PAB, as already discussed in section C.3, BASIC adds four more bytes to the top of the PAB for specific BASIC-related control information. The new PAB structure within BASIC is drawn in Figure C.5.1.

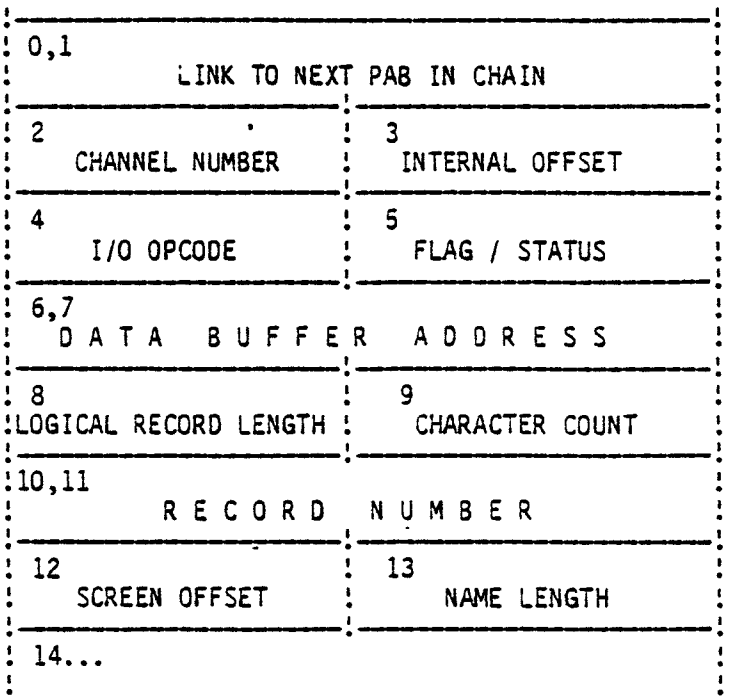


FIGURE C.5.1 Modified PAB Layout

The additional four bytes contain additional control information BASIC needs for its internal PAB linkage structure. The PAB's within the BASIC control structure form a simple linked list, which means each PAB has a pointer to the next PAB in the system. The last PAB in the list has a zero ("0") link, indicating that it is at the end of the list.

Bytes 0 and 1 in the BASIC PABs contain the link mentioned above. Byte 2 contains the actual BASIC channel or file number (1-255); byte 3 contains the offset of the current data pointer within the data block.

The offset indicated in byte 3 of the BASIC PAB indicates the position of the current data pointer within the data buffer given in bytes 6 and 7. If byte 3 equals zero, the current data buffer is "blank"; i.e. if in "read" mode, a new buffer has to be read in before any further processing, but in "write" mode the entire buffer is still available for data storage.

If byte 3 is non-zero, it contains an "offset" within the data buffer. Added to the start address of the data buffer, it will give the actual address of the first data byte to be read or written. This is only the case if we have pending PRINT operations (the most recent PRINT ended on ";" or ",") or pending INPUT operations (the most recent INPUT ended on a ","). In all other cases, byte 3 will be zero.

C.5.2 BASIC PAB Linkage

As mentioned already, BASIC utilizes a simple linked structure for the management of its PAB's. Each PAB contains a link to the next PAB in the chain. In order to access the chain, we need to have a link to the first PAB in that chain. This link is given in CPU location >3C for GPL programs, which is equal to location >833C in 9900 assembly language.

A graphical representation of the link structure could be:

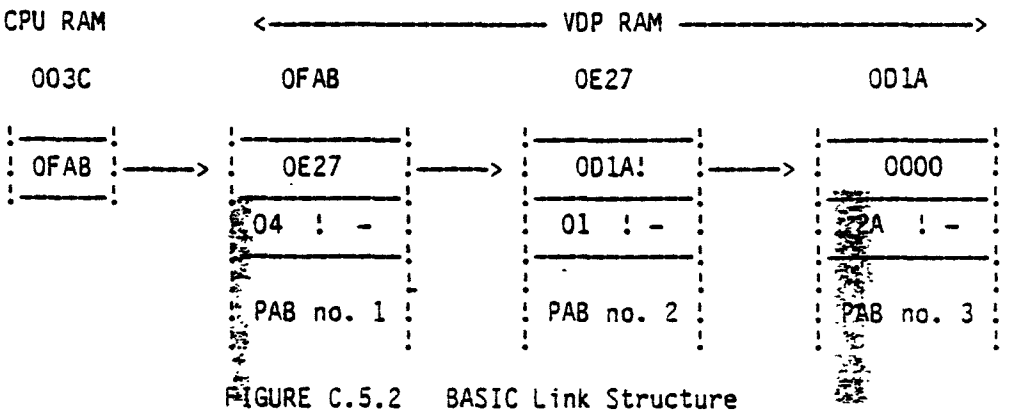


FIGURE C.5.2 BASIC Link Structure

Please note that all PAB's are located in VDP memory, whereas the initial link to them is located in CPU memory. Also note that although usually the PAB's will be allocated in lexical order, depending upon the program, they can be allocated in any arbitrary order.

Section D—DSR SPECIFICATIONS

D.1 Introduction

This document describes the hardware and software interfaces between the console and peripherals of the TI-99/4A Home Computer family. The purpose of this document is to provide a reference for third parties who want to design new peripherals for the TI-99/4A family.

D.1.1 General Interface

Each peripheral should incluce a nonvolatile Device Service Routine (DSR) software package to drive the peripheral. The DSR communicates with the console software through certain memory locations. The console software sets up information in these memory locations and passes it to the user-selected peripheral. From there on, the selected peripheral's DSR should have the capability to interpret the information set up by the console software, physically drive the peripheral, and pass the peripheral's data or status to the console software if neccessary.

D.2 I/O Bus—The purpose of this section is to describe the pin assignments of the I/O Bus, funtions of I/O signals, sources of Output Signals, and destinations of Input Signals.

D.2.1 I/O Bus Pin Assignments and Descriptions

Pin No.	Notation	I/O	Description
1	+5V	O	5V Power Supply
2	SBE	O	Low when MPU read from >90XX or write to >94XX memory
3	$\overline{\text{RESET}}$	O	Master Reset, low active
4	$\overline{\text{EXTINT}}$	I	External Interrupt, low active
5	A5	O	Address Bit 5
6	A10	O	Address Bit 10
7	A4	O	Address Bit 4
8	A11	O	Address Bit 11
9	DBIN	O	Derived from MPU's DBIN pin, same parity
10	A3	O	Address Bit 3
11	A12	O	Address Bit 12
12	READY	I	If device or memory is ready after being addressed by MPU in memory R/W cycle, device or memory should cause this input to go high
13	$\overline{\text{LOAD}}$	I	To 9900's LOAD pin
14	A8	O	Address Bit 8
15	A13	O	Address Bit 13
16	A14	O	Address Bit 14
17	A7	O	Address Bit 7
18	A9	O	Address Bit 9
19	A15/CRUOUT	O	CRU Output/Address Bit 15, LSB
20	A2	O	Address Bit 2
21	GND	O	Signal Ground
22	$\overline{\text{CRUCLK}}$	O	Inversion of MPU's CRUCLK pin
23	GND	O	Signal Ground
24	$\overline{\phi 3}$	O	Inversion of Phase 3 Clock
25	GND	O	Signal Ground
26	$\overline{\text{WE}}$	O	Derived from MPU's $\overline{\text{WE}}$ pin, same parity
27	GND	O	Signal Ground
28	$\overline{\text{MBE}}$	O	Low when MPU addressing >4000—>5FFF Memory
29	A6	O	Address Bit 6
30	A1	O	Address Bit 1
31	A0	O	Address Bit 0, MSB
32	$\overline{\text{MEMEN}}$	O	Derived from MPU's $\overline{\text{MEMEN}}$ pin, same parity
33	CRUIN	I	CRU Input to MPU
34	D7	I/O	Data Bus Bit 7, LSB
35	D4	I/O	Data Bus Bit 4
36	D6	I/O	Data Bus Bit 6
37	D0	I/O	Data Bus Bit 0, MSB
38	D5	I/O	Data Bus Bit 5
39	D2	I/O	Data Bus Bit 2
40	D1	I/O	Data Bus Bit 1
41	IAQ	O	MPU's IAQ pin
42	D3	I/O	Data Bus Bit 3
43	-5V	O	-5V Power Supply
44	AUDIO IN	I	To Sound Generator Controller's AUDIO IN pin

D.3 Hardware Structure of DSR

D.3.1 DSR ROM

Normally, a DSR is written in the 9900 Assembly Language and is housed in a ROM, which is itself part of the TI-99/4A family's peripheral. All DSRs must begin at address >4000 and should not exceed >5FFF. All of the eight data pins of the DSR ROM must be buffered before being connected to the data bus of the I/O Bus. This buffer is enabled and disabled by a preassigned CRU output bit which is controlled by the console software, so that not more than one DSR is accessed at any time.

D.3.2 CRU Mapping

The CRU I/O is used by the system to access the peripherals, if the speed of data transfer is not too crucial. The decoding format for the CRU addressing is indicated as:

A0 A1 A2	A3 A4 A5	A6 A7	A8 A9 A10 A11 A12 A13 A14	A15/CRUOUT
0 0 0	User ID	Device	CRU I/O Bit Decode	0

The CRU address space, ranging from >0 to >1FFE with A0, A1, A3, and A15 unused, is broken into eight blocks of 512 bits each. The User ID, indicated through the Address Decoding of A3, A4, and A5, represents each of the 512 Bit Block. They are assigned as follows:

MPU Address	A3 A4 A5	Assignment
0000-03FE	0 0 0	Console Use
1000-13FE	1 0 0	TI Peripheral Space No. 1
1400-17FE	1 0 1	TI Peripheral Space No. 2
1800-1BFE	1 1 0	TI Peripheral Space No. 3
1C00-1FFE	1 1 1	TI Peripheral Space No. 4

The standard device decoding through A6 and A7 gives a total of four device blocks within each User ID Block. The CRU I/O Bit decoding through A8-A14 allows 128 addressable bits each of input and output. In the 128 CRU Output bits, the first bit, with A8-A14 all 0's, is reserved for enabling the Data Buffer of the DSR ROM in each peripheral as mentioned in the last subsection. Setting this bit to logic one enables the DSR ROM, while setting it to logic zero disables the DSR ROM.

D.4 Software Structure of DSR

A DSR must follow a specific format in order to communicate with the console software properly. The purpose of this predefined format is to let the console software have the least overhead and the DSR have the maximum flexibility in device servicing.

A DSR, in general, contains the following elements:

1. Symbol Definition Block
2. Header and Linkage Block
3. Power-up Routine (optional)
4. Main Device Service Routine
5. Interrupt Routine (optional)

Each of the above elements will be discussed in detail in the following sections.

D.4.1 Symbol Definition Block

The Symbol Definition Block serves two purposes. First, it equates frequently used data or addresses with symbols for ease of recognition. Secondly, it specifies the CPU RAM location in a certain way so that each DSR can be used around future and existing CPU family members such as the TI-99/4 and the TI-99/4A with the least modification. The first purpose is common in every program, but the second one needs more explanation.

Each model in this family has a different memory structure within the console. Therefore, care must be taken in handling memory addressing in a DSR so that the DSR can support future Home Computer models. The console software always enters the DSR through the instruction:

```
        BL    *R9
```

where R9 contains the DSR entry address. Upon entry of the DSR, the workspace pointer contains the beginning address of the Register File, which the console software uses. If all the CPU RAM's which the DSR may access are predefined with respect to this workspace pointer, the DSR does not have to know the memory map of each console. The DSR can address each CPU RAM through index addressing via the workspace pointer. For example, if you want to move the contents of a CPU RAM (having a displacement [DISP] with respect to the workspace pointer) to R0, do the following:

```
ENTRY   STWP  R4            entry of DSR
        .
        .
        .
        MOV   at DISP (R4), R0
        .
        .
```

Some of the frequently used CPU RAM's symbols and locations are listed below.

```
PAD       EQU       ->E0          start of CPU RAM in TI-99/4A console
FAC       EQU       PAD+4A        start of 36 bytes available to DSR
ROLB      EQU       PAD+E1        lower byte of R0
R1LB      EQU       PAD+E3        lower byte of R1
OPCODE    EQU       FAC+0         beginning of PAB, I/O operation code
FLGSTS    EQU       FAC+1         PAB--Flag/Status
BUFADR    EQU       FAC+2         PAB--Data Buffer Address
LRECLN    EQU       FAC+4         PAB--Logical Record Length
CHRCNT    EQU       FAC+5         PAB--Character Count
RECNUM    EQU       FAC+6         PAB--Record Number
SCNOFF    EQU       FAC+8         PAB--Screen Offset
OPTLEN    EQU       FAC+9         PAB--Option Length
DEVLEN    EQU       FAC+10        PAB--Device Length
PABVDP    EQU       FAC+12        PAB--Pointer to PAB in VDP RAM
VWA       EQU       >8C02         address for VDP Write-Address operation
GRD       EQU       >9800         address for GROM Read-Data operation
```

D.4.2 Header and Linkage Block

The DSR must contain a certain header starting at >4000 so that the linkage to the console software can be established properly. This header contains the following information:

1. validation flag (>AA).
2. name(s) of device(s) being serviced by this DSR
   (the device name should be 7 characters or less).
3. entry point(s) of the device(s) being serviced by this DSR.
4. entry point of Power-up Routine, if necessary.
5. entry point of Interrupt Routine, if necessary.

D.4.2.1 A Sample Program for Header and Linkage Block

This section gives an example of the Header and Linkage Block in a typical DSR. The syntax of DX10 Assembler is obeyed in this program. DX10 Assembler Language is similar to, but not identical with, the TI Editor/Assembler. Information on the language can be found in the 9900 Family Design Book.

Sample Program Showing DSR Header Format

```
          RORG >4000                    start of DSR
          BYTE >AA                      telling console this is a valid DSR
          BYTE 1                        version number, always 1
          DATA 0                        not used in DSR calls; leave it 0
          DATA PWRLNK                   Power-up Routines' Link.  PWRLNK is
                                        replaced by 0 if power-up set is not
                                        necessary
          DATA 0                        not used in DSR calls; leave it 0
          DATA DSRLNK                   DSR link; can't be 0 here
          DATA 0                        not used in DSR calls; leave it 0
          DATA INTLNK                   Interrupt Routine's Link.  INTLNK is
                                        replaced by 0 if interrupt is not used
          DATA 0                        not used in DSR calls; leave it 0

*         *** This Power-up Routine can be omitted if power-up
*         *** initialization is not necessary for this peripheral
*
PWRLNK    DATA 0                        linkage, set to 0
          DATA PWRUP                    entry point of power-up routine
          BYTE 0                        name length; set to 0
          EVEN
          .
          .
PWRUP     .                             entry of power-up routine
          .
          .
          B     *R11                    return to console software through R11,
                                        if it is not destroyed
          .
          .
*         *** This Interrupt Routine can be omitted if Interrupt
*         *** Request is never issued by this peripheral
*
INTLNK    DATA 0                        linkage; set to 0
          DATA INTDSR                   entry point of the Interrupt Routine
          BYTE 0                        name length; set to 0
          EVEN
          .
          .
```

Sample Program (continued)

```
INTDSR    .                          entry of interrupt routine:
          .                          IF   interrupt request from this
                                          peripheral = true
                                     THEN begin interrupt service;
                                          reset interrupt request;
                                          GOTO INTEND
                                          END
                                     ELSE GOTO INTEND

INTEND B     *R11                    return to console software through R11,
                                     *    if it is not destroyed
          .
*
*         *** Main Device Service Routine.  Assuming three devices,
*         *** with names DEVICE, DEVICE/1, DEVICE/2, are
*         *** supported in this peripheral.
*
DSRLNK    DATA DSRLK2                linkage to next device field
          DATA ENTRY1                entry point of 1st device
          BYTE 6                     name length of 1st device
          TEXT 'DEVICE'              name of 1st device
          EVEN
DSRLK2    DATA DSRLK3                linkage to next device field
          DATA ENTRY2                entry point of 2nd device
          BYTE 8                     name length of 2nd device
          TEXT 'DEVICE/1'            name of 2nd device
          EVEN
DSRLK3    DATA 0                     linkage to next device field (none)
          DATA ENTRY3                entry point of 3rd device
          BYTE 8                     neme length of 3rd device
          TEXT 'DEVIC/2'             name of 3rd device
          EVEN
          .
          .
ENTRY1    .                          entry of 1st device servicing
          .
          .
ENTRY2    .                          entry of 2nd device servicing
          .
          .
ENTRY3    .                          entry of 3rd device servicing
          .
          .
EXIT      INCT R11                   return to console software througn
          B    *R11                  *  R11, if it is not destroyed
```

D.4.3 Power-Up Routine

For some peripherals, it is necessary to initialize the hardware at power-up time. It is suggested that a power-up routine be included to do the initialization through software for these peripherals. Power-up Link should be set in the Header Field as described above.

Power-up routines are executed whenever the system is reset by either hardware or software. The console software searches all peripheral DSR for power-up routine addresses and executes them if they are found. Each power-up routine can use R0-R10. Upon entry, R12 is set to the beginning address of the CRU space for that peripheral DSR (note that this address is used to enable the DSR ROM). R11 contains the return address. R13 and R15 contain the memory-mapped addresses of GROM Read Data and VDP Write Address, respectively. All VDP and GROM operations can be indexed from these two registers. The power-up routine may use VDP RAM from 0 to the location pointed by CPU RAM >70, offset from workspace pointer - >E0). It can use all CPU RAM except location >55, <6D, and >C0 through >DF offset from workspace pointer - >E0.

All power-up routines must return with B *R11.

D.4.4 Interrupt Routine

If the peripheral issues Interrupt Request to the 9900 CPU, the DSR should also include an interrupt routine and Interrupt Link in the Header as described above. Every interrupt that is not recognized as console interrupt (VDP or 9901 Timer) causes the console software to execute every interrupt routine it can find. The interrupt routine must check to see whether the Interrupt Request is raised by this peripheral. If it is not, exit by B *R11.

The interrupt routine may use R1-R8, and R10. The contents of R11-R15 are the same as those of the Power-up Routine Section. The interrupt routine and Main DSR can split the allocation of CPU RAM from >4A to >6D offset from workspace pointer - >E0).

All interrupt routines end with B *R11. Interrupt Request raised by the peripheral must be cleared before exit.

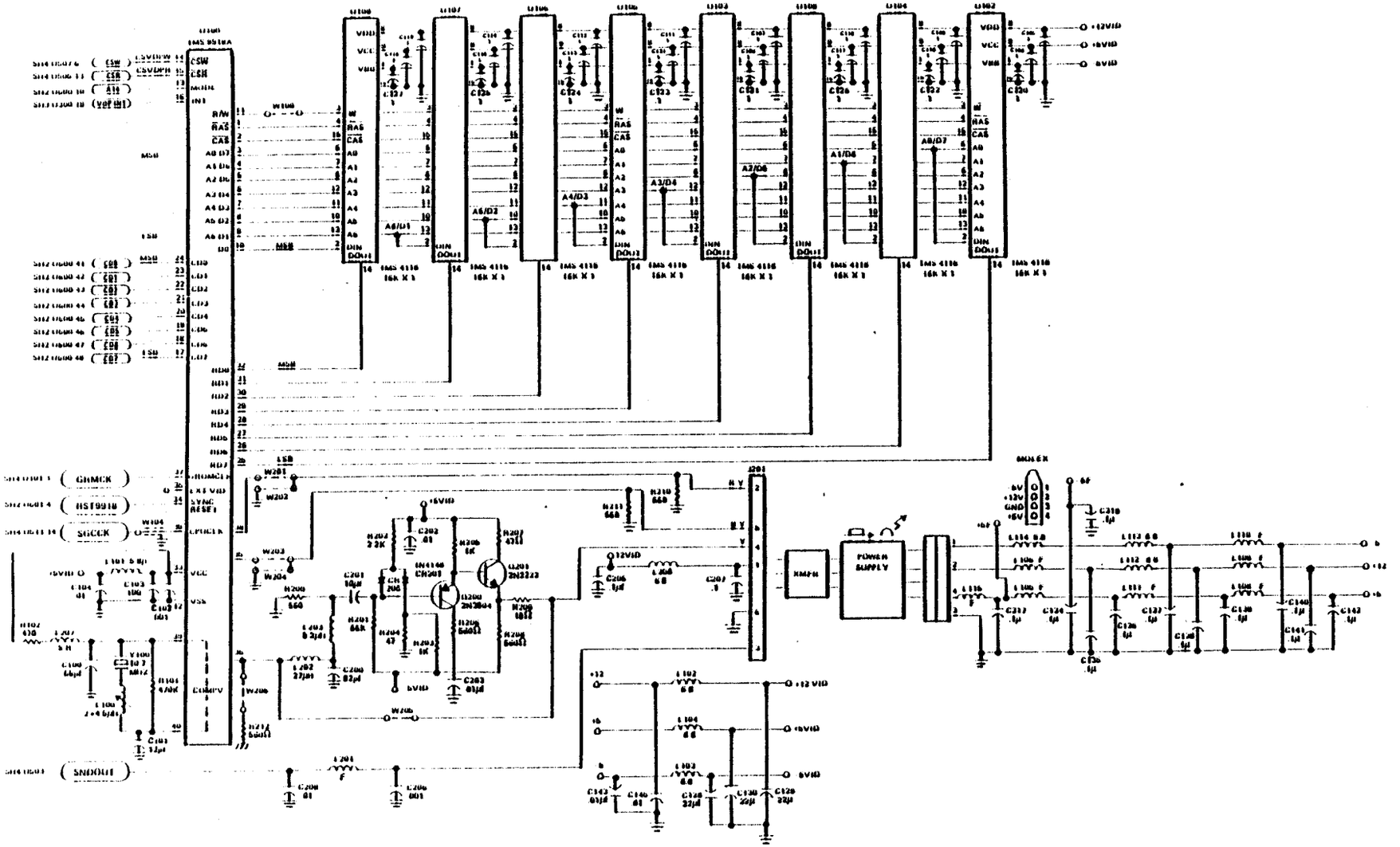
To exit DSR, do

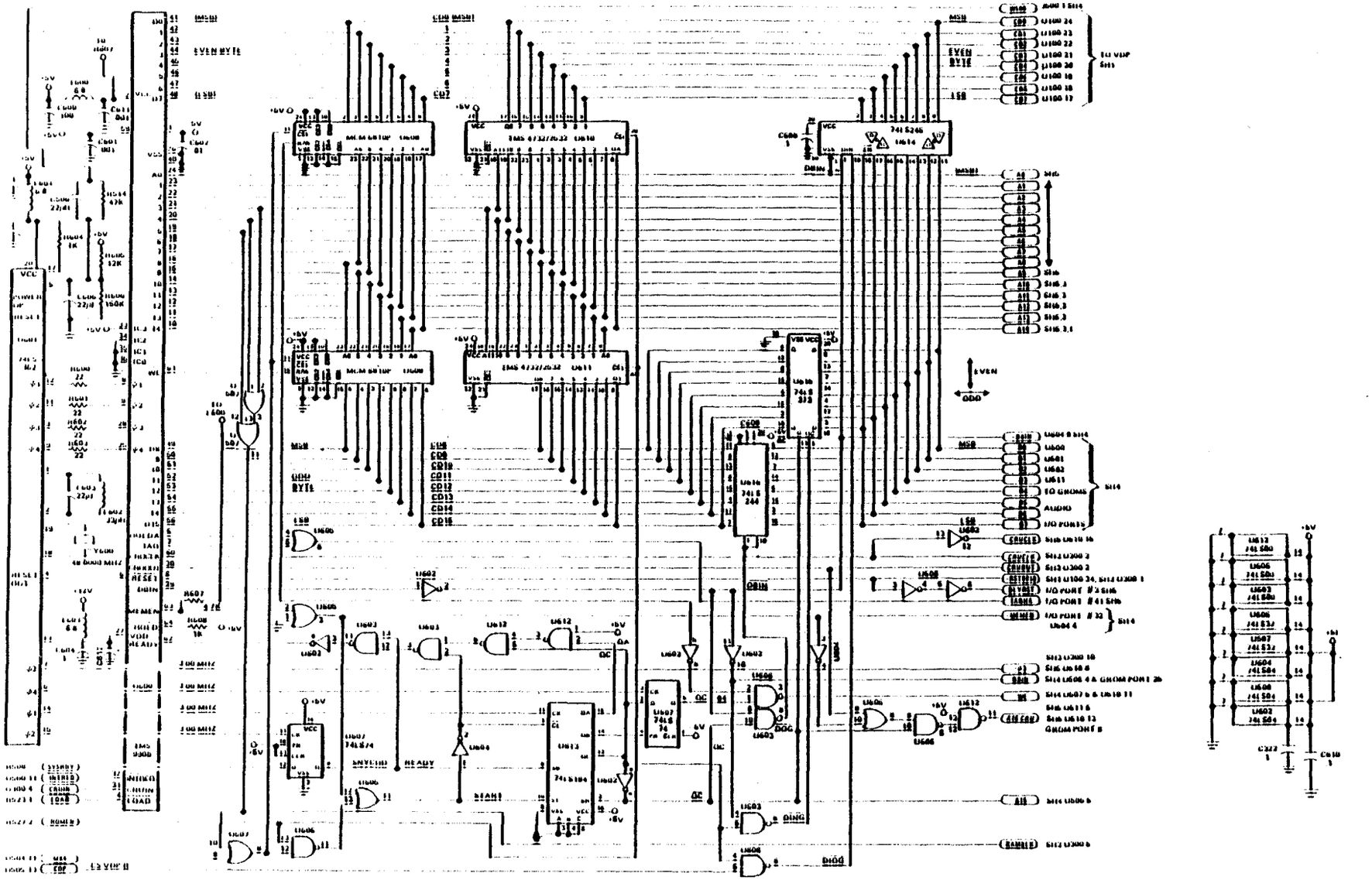
```
INCT   R11
B      *R11
```

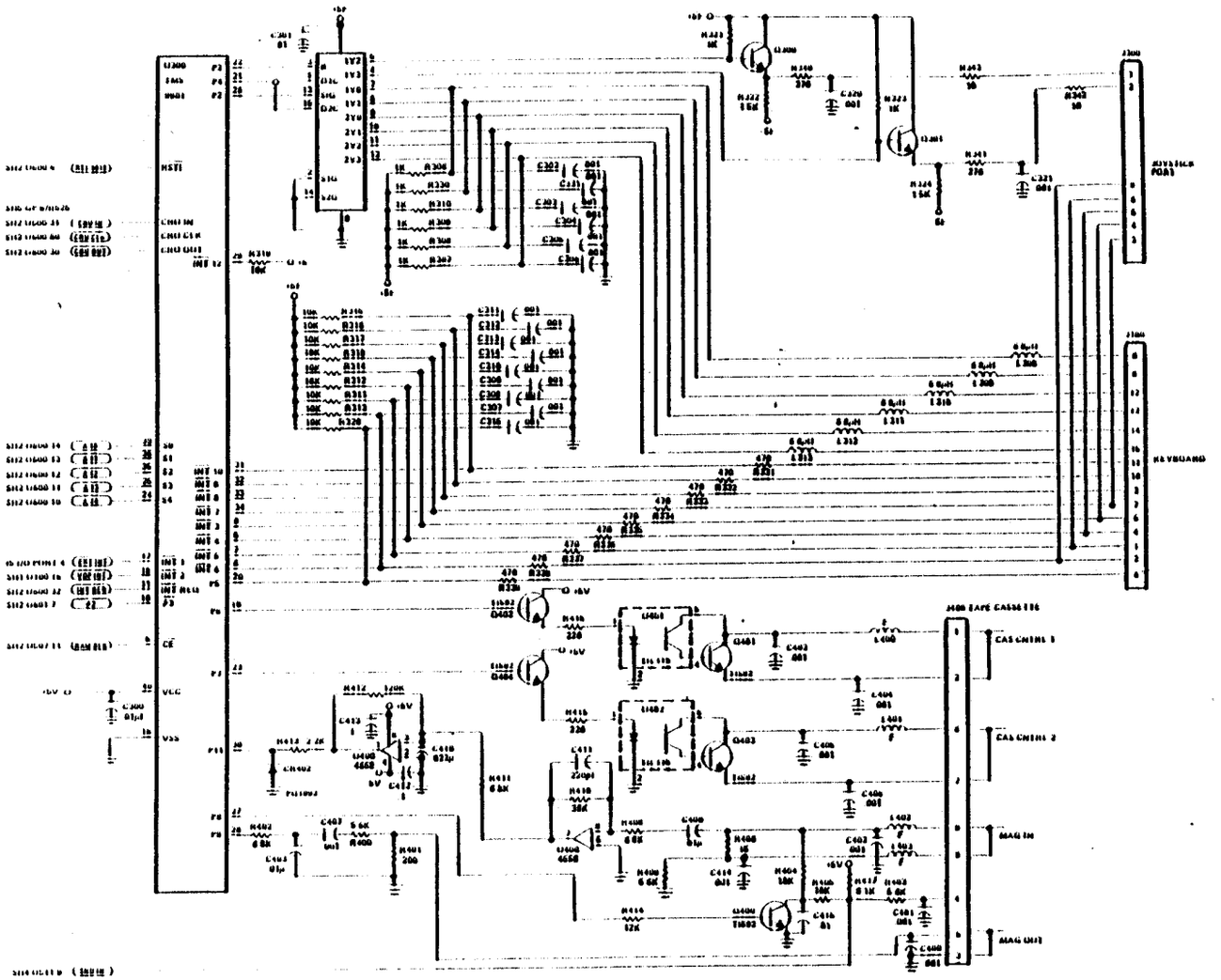
GLOSSARY

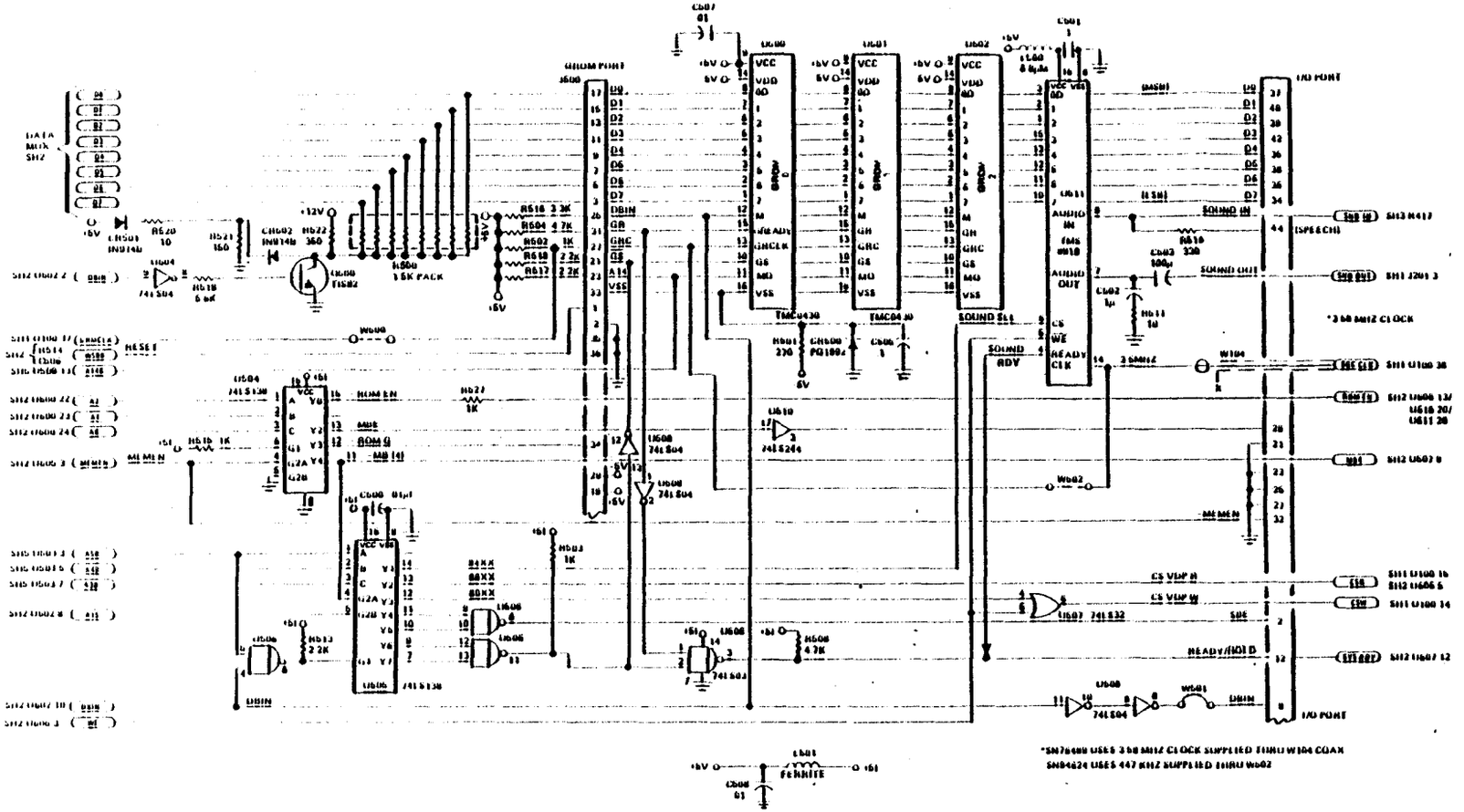
ALU	Arithmetic Logic Unit.
bit	Smallest unit of memory; BInary digiT.
byte	8 bits of memory.
cartridge memory	ROM space found within a plug-in cartridge.
CPU	Central Processing Unit.
CRU	Communication Register Unit (I/O techique for TMS 9900 Microprocessor).
DRAM	Dynamic Random Access Memory.
DSR	Device Service Routine (TMS 9900 machine language for Device Interface).
GPL	Graphics Programming Language.
GROM	Graphics Read-Only Memory (TMC 0430). This memory device is a 6144 byte read-only memory with on-board 13-bit program counter. The program counter can be written or read through an 8-bit interface and will automatically increment.
I/O	Input/Output.
LSB	Least Significant Bit.
LSBY	Least Significant Byte.
MMD	Memory-Mapped Device.
MSB	Most Significant Bit.
MSBY	Most Significant Byte.
MPU	Microprocessor Unit.
PAB	Peripheral Access Block.
PC	Program Counter.
PCB	Printed Circuit Board.
PEU	Peripheral Expansion Unit.

RAM	Random Access Memory.
ROM	Read-Only Memory.
SRAM	Static Random Access Memory.
VDP	Video Display Processor (TMS 9918A).
WORD	16 bits/2 bytes of memory.


POWER SUPPLY
RMPH
SNDOUT
GRMCK
RST9918
SGCCK

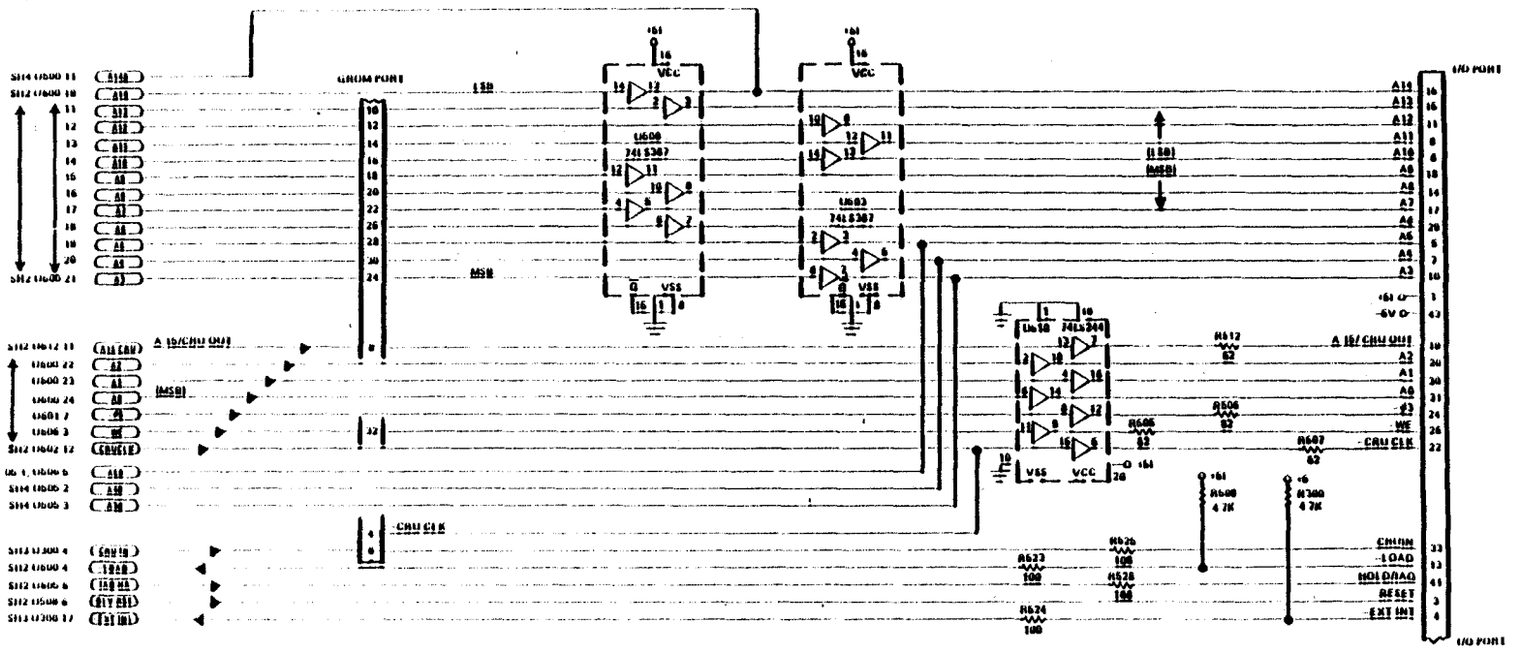


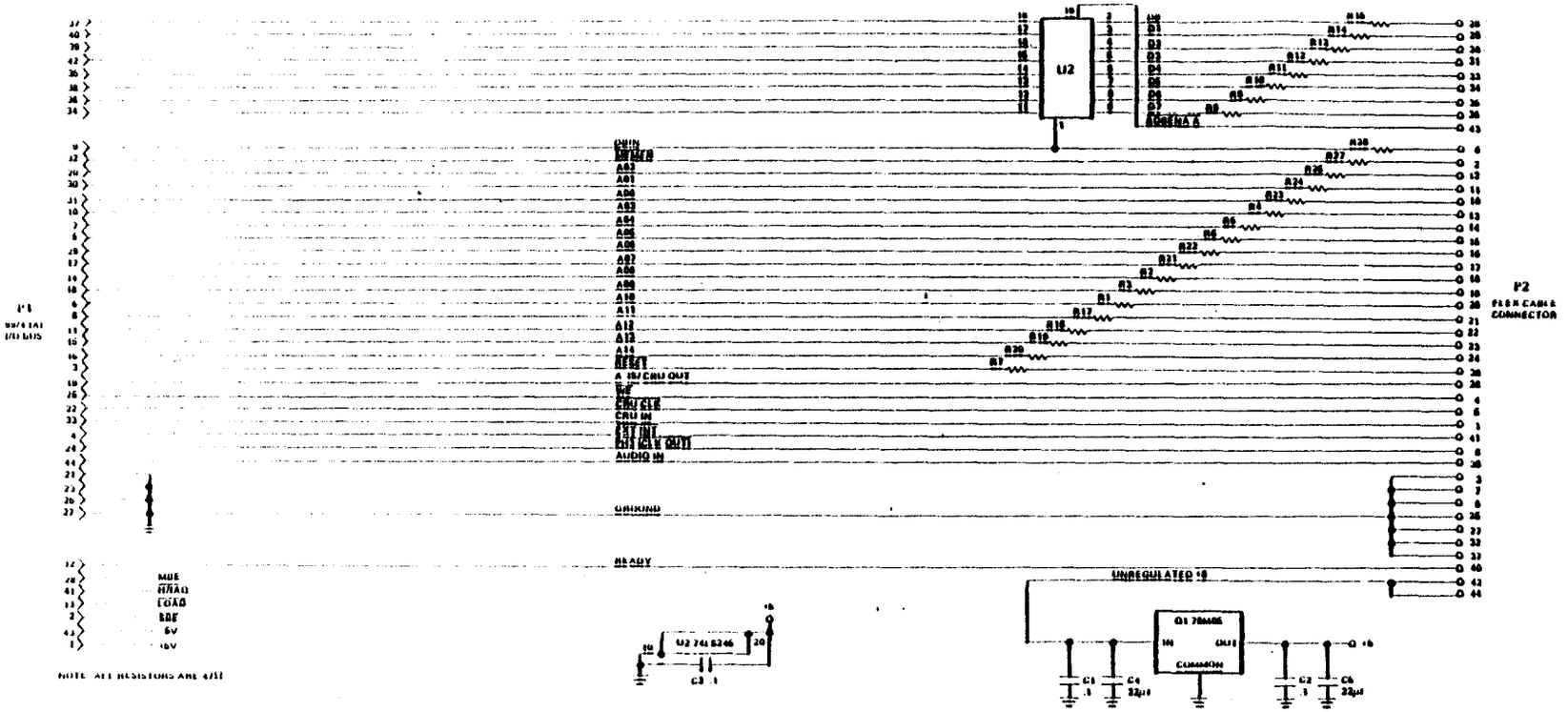




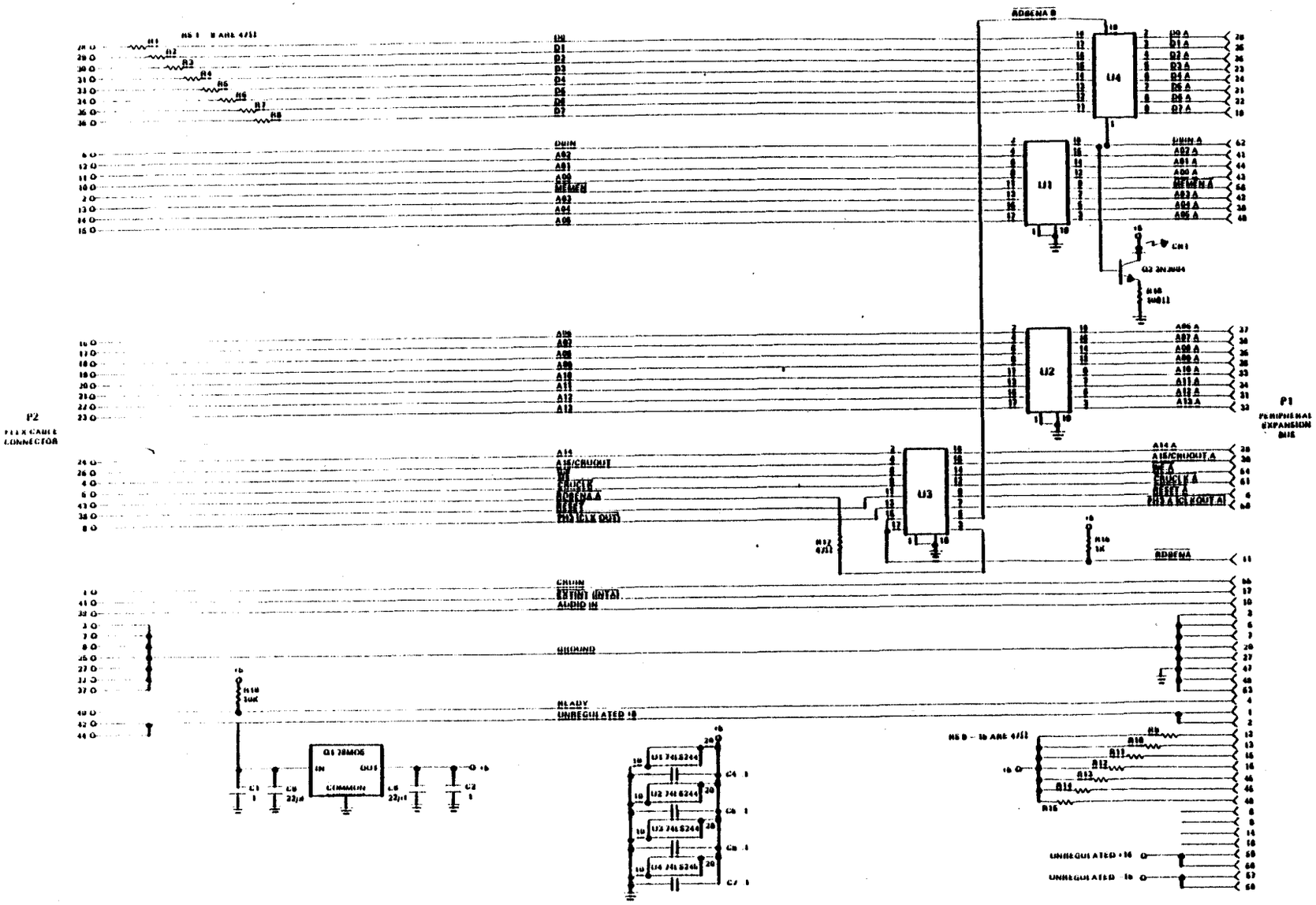
*SN76489 USES 3.58 MHZ CLOCK SUPPLIED THRU W104 COAX
SN94624 USES 447 KHZ SUPPLIED THRU W602

TI 99/4A SCHEMATIC DIAGRAM

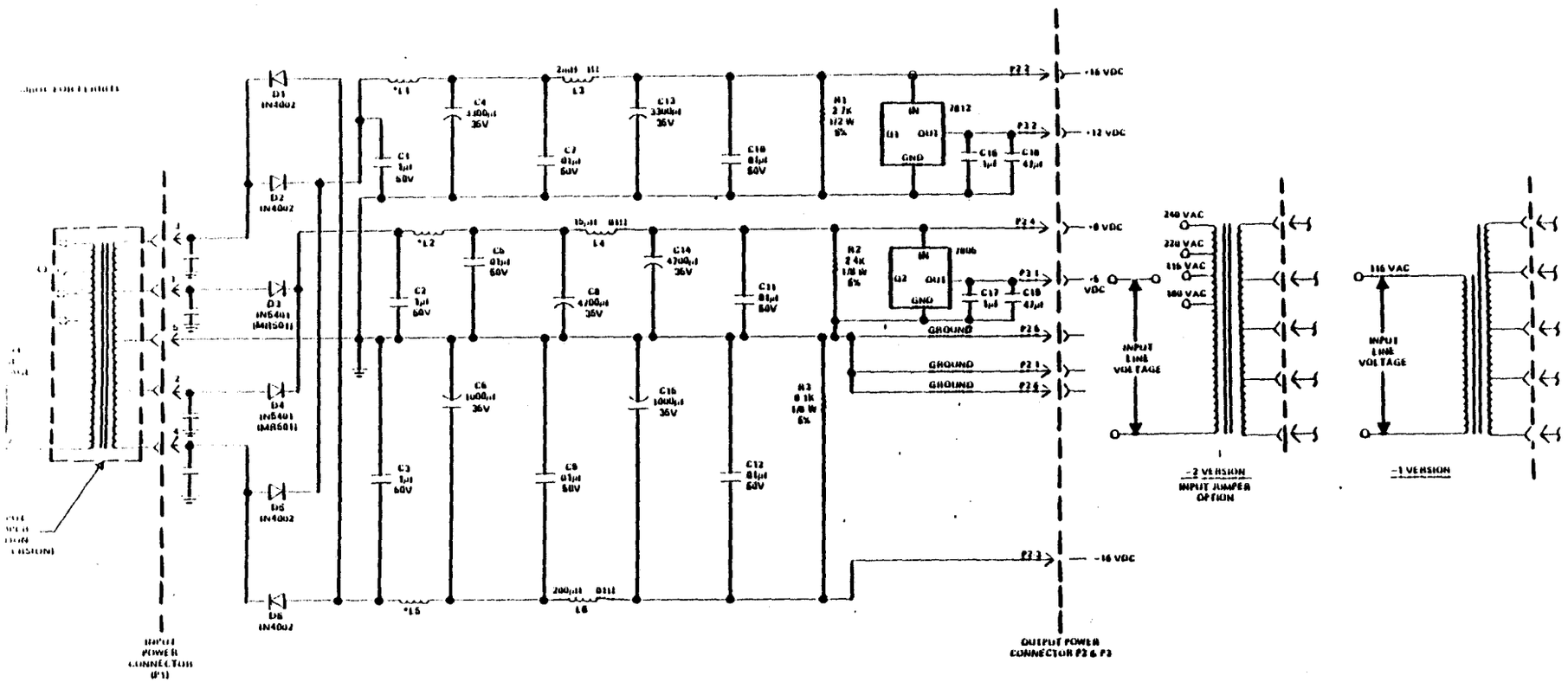




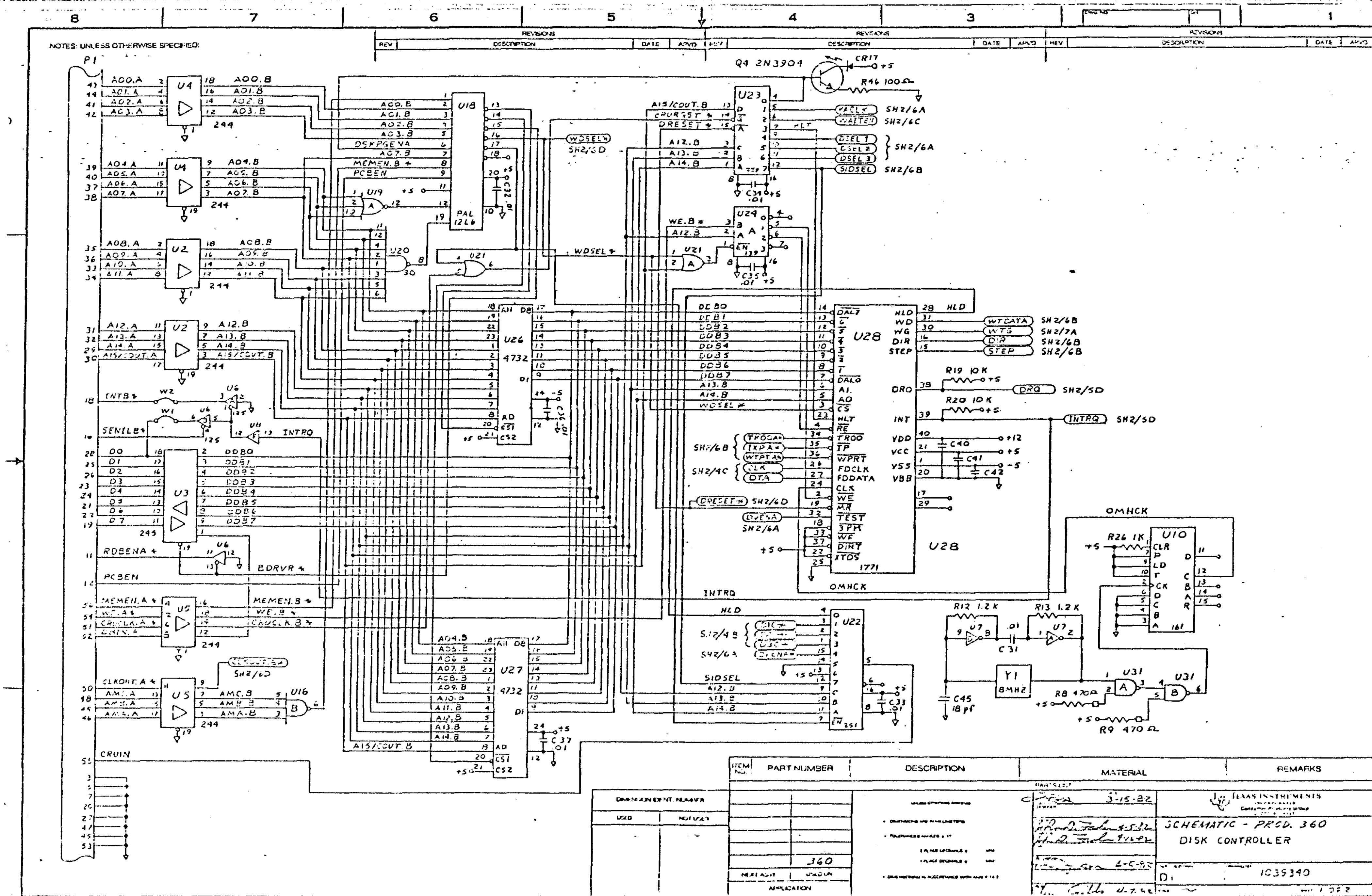
FLEX CABLE SCHEMATIC DIAGRAM
(CONSOLE END OF CABLE)



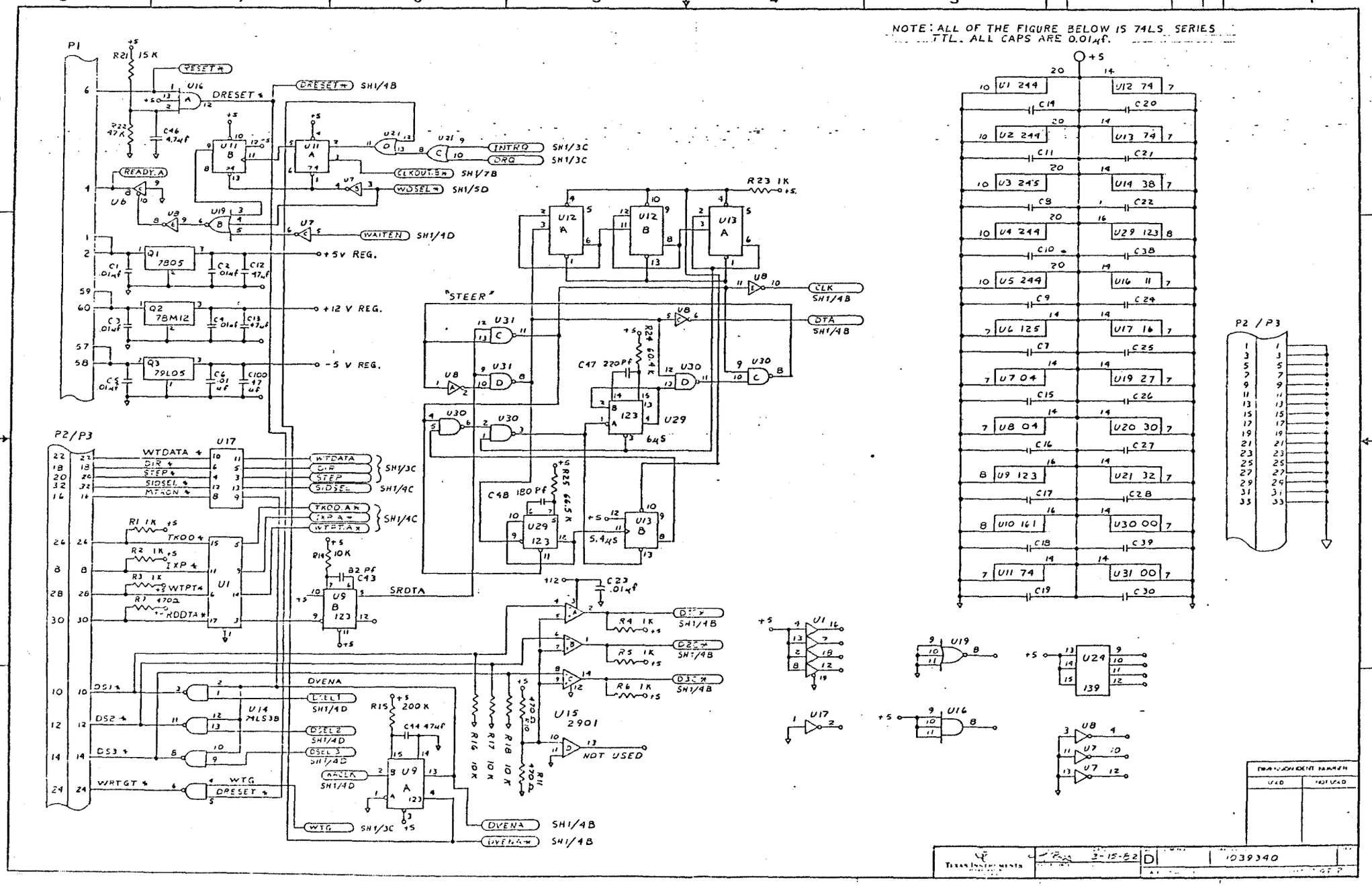
FLEX CABLE SCHEMATIC DIAGRAM
(PERIPHERAL EXPANSION UNIT END OF CABLE)

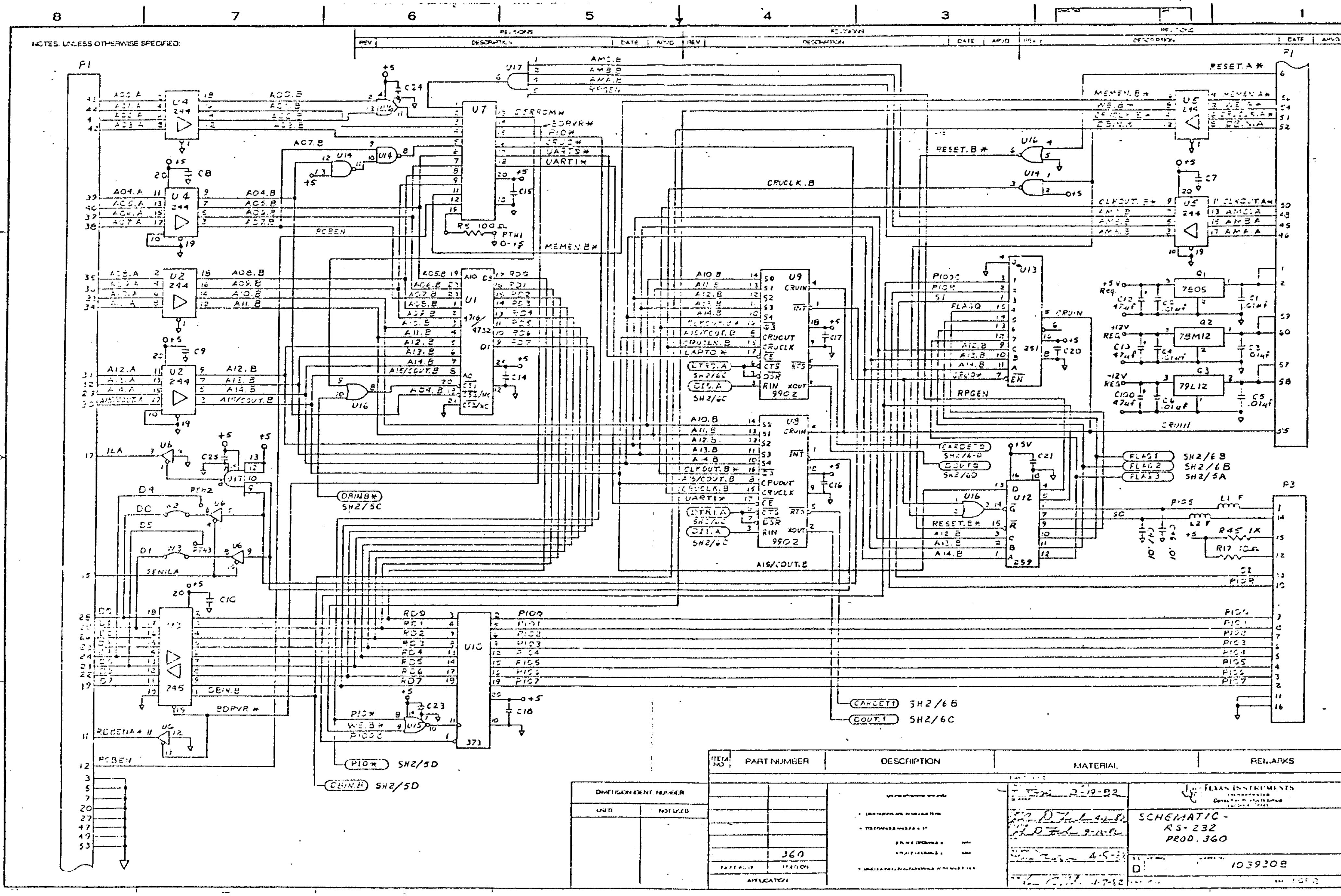


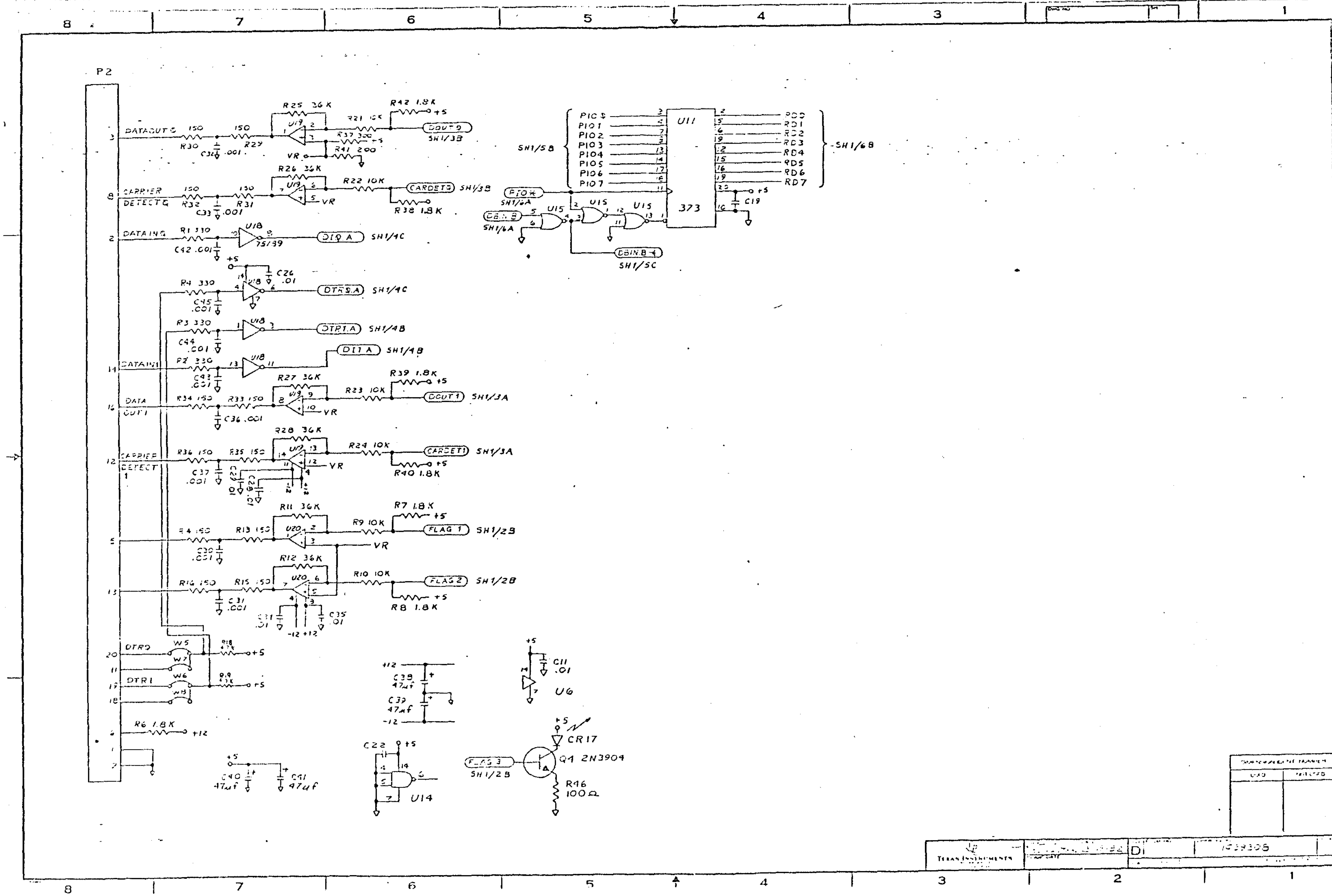
PERIPHERAL EXPANSION UNIT POWER SUPPLY



NOTES: UNLESS OTHERWISE SPECIFIED:
REVISIONS
P1
U4 244
U2 244
U3 245
U5 244
U16
U18 PAL 12L6
U19
U20
U21
U23
U24
U26 4732
U27 4732
U28 1771
U22 251
U10 161
U7
U31
Y1 8MHZ
Q4 2N3904
CR17
R46 100Ω
R19 10K
R20 10K
R26 1K
R12 1.2K
R13 1.2K
R8 470Ω
R9 470Ω
C45 18pF
INTRQ
OMHCK
CRUIN
WDSEL
PART NUMBER
DESCRIPTION
MATERIAL
REMARKS
TEXAS INSTRUMENTS
SCHEMATIC - PRSD. 360
DISK CONTROLLER
3-15-82
360
1C35340


NOTE: ALL OF THE FIGURE BELOW IS 74LS SERIES TTL. ALL CAPS ARE 0.01μf.
P1
P2/P3
P2 / P3
RESET*
DRESET*
READY.A
WAITEN
WDSEL*
INTRQ
DRQ
CLKOUT.5*
"STEER"
CLK
DTA
WTDATA*
DIR*
STEP*
SIDSEL*
MTRON*
TKOO A*
IXP A*
WTPT A*
TKOO*
IXP*
WTPT*
RDDTA*
SRDTA
DS1*
DS2*
DS3*
WRTGT*
WTG
DVENA
DVENA*
NOT USED
+5V REG.
+12 V REG.
-5 V REG.
SH1/3C
SH1/4C
SH1/4B
SH1/4D
SH1/5D
SH1/7B


NOTES, UNLESS OTHERWISE SPECIFIED:
PART NUMBER
DESCRIPTION
MATERIAL
REMARKS
TEXAS INSTRUMENTS
SCHEMATIC -
RS-232
PROD. 360
D
1039308
360
APPLICATION
USED

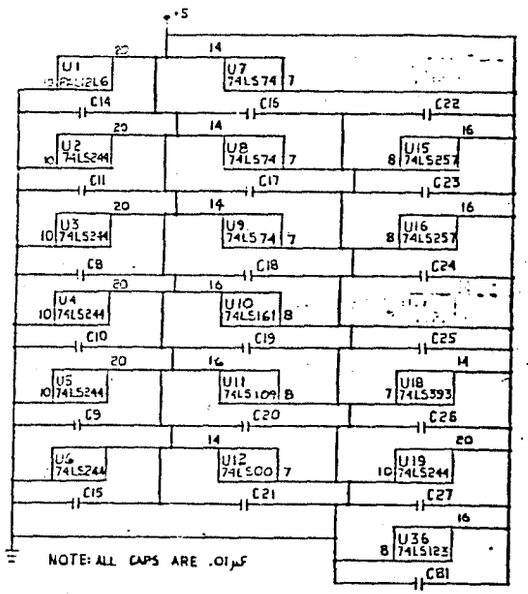

P2
DATAOUT 0
CARRIER DETECT 0
DATA IN 0
DATA IN 1
DATA OUT 1
CARRIER DETECT 1
DTR 0
DTR 1
U11
373
U15
U18
U19
U20
U6
U14
Q1 2N3904
CR17
R46 100Ω
DOUT 0
CARDET 0
DI 0 A
DTR 0 A
DTR 1 A
DI 1 A
DOUT 1
CARDET 1
FLAG 1
FLAG 2
FLAG 3
PIO 0
PIO 1
PIO 2
PIO 3
PIO 4
PIO 5
PIO 6
PIO 7
RD 0
RD 1
RD 2
RD 3
RD 4
RD 5
RD 6
RD 7
SH1/5B
SH1/6B
SH1/3B
SH1/4C
SH1/4B
SH1/3A
SH1/2B
SH1/6A
SH1/5C
TEXAS INSTRUMENTS

NOTES, UNLESS OTHERWISE SPECIFIED

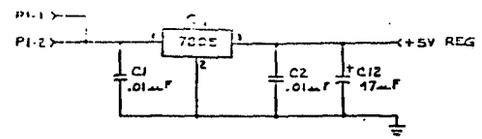
REVISIONS							
REV	DESCRIPTION	DATE	APVD	REV	DESCRIPTION	DATE	APVD

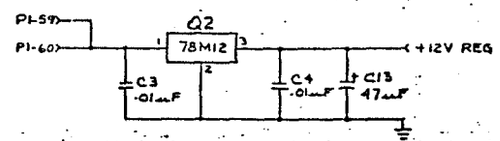
REVISIONS			
REV	DESCRIPTION	DATE	APVD
A	NO ECN - M.L Medlock 3-22-82 REDRAW SCHEMATIC.		
	FORMAL RELEASE		
B	ECN 102936 Tom Chase 7-6-82 CHANGED U1 TO A PAL12L6	6/30/82	PR

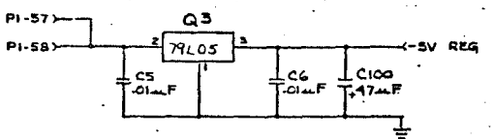
DRAWING 1039330 SH 1

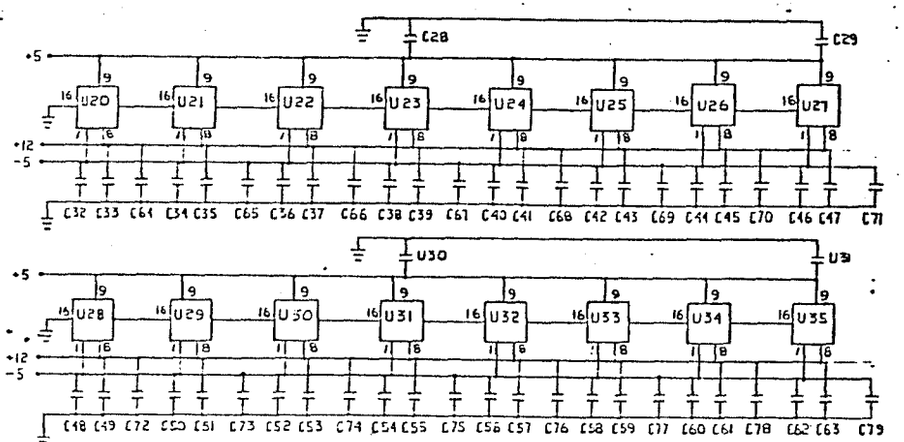


TTL POWER









NOTE: C28-63 ARE .1μF
C64-79 ARE 10μF

TMS 4116 POWER

ITEM NO	PART NUMBER	DESCRIPTION	MATERIAL	REMARKS

DIMENSION IDENT. NUMBER	
USED	NOT USED

360

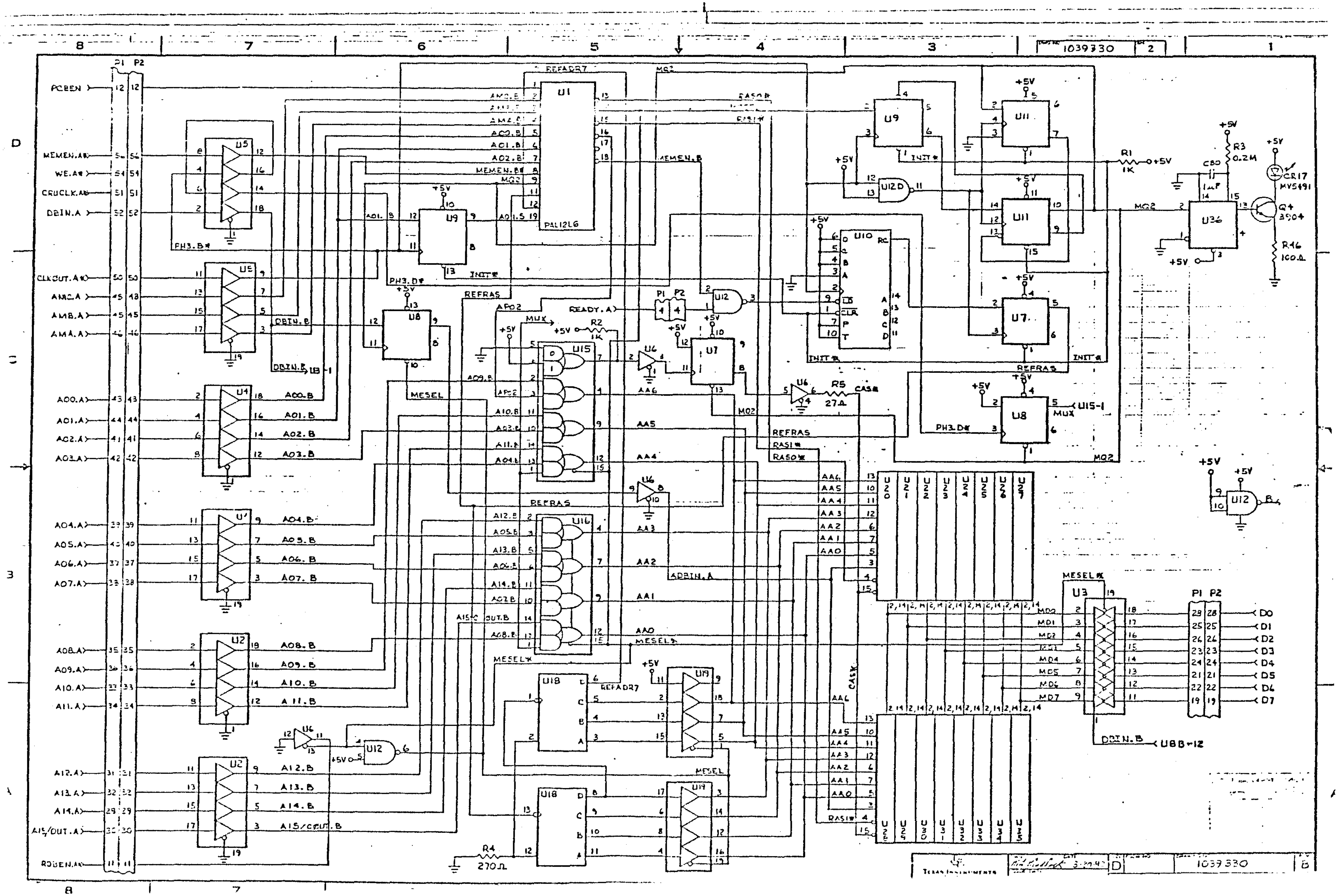
M.L Medlock 3-22-82

TEXAS INSTRUMENTS

SCHEMATIC PCB
PROJ. 360
MEMORY EXPAN. - JOHNNY BOX

SIZE B 1039330

1 OF 2



1039330
2
PCBEN
MEMEN.A
WE.A
CRUCLK.A
DBIN.A
CLKOUT.A
AMC.A
AMB.A
AMA.A
A00.A
A01.A
A02.A
A03.A
A04.A
A05.A
A06.A
A07.A
A08.A
A09.A
A10.A
A11.A
A12.A
A13.A
A14.A
A15/OUT.A
RDBEN.A
P1 P2
U1
PAL12L6
REFADR7
MO2
RASO
RAS1
MEMEN.B
PH3.B
PH3.D
INIT
REFRAS
AP02
MUX
READY.A
R1 1K
R2 1K
R3 0.2M
R4 270Ω
R5 27Ω
R46 100Ω
C80 1µF
CR17 MV5491
Q4 3904
U36
MESEL
CASE
AA0
AA1
AA2
AA3
AA4
AA5
AA6
ADBIN.A
DBIN.B
U8B-12
MD0
MD1
MD2
MD3
MD4
MD5
MD6
MD7
D0
D1
D2
D3
D4
D5
D6
D7
+5V
TEXAS INSTRUMENTS
1039330
D
B