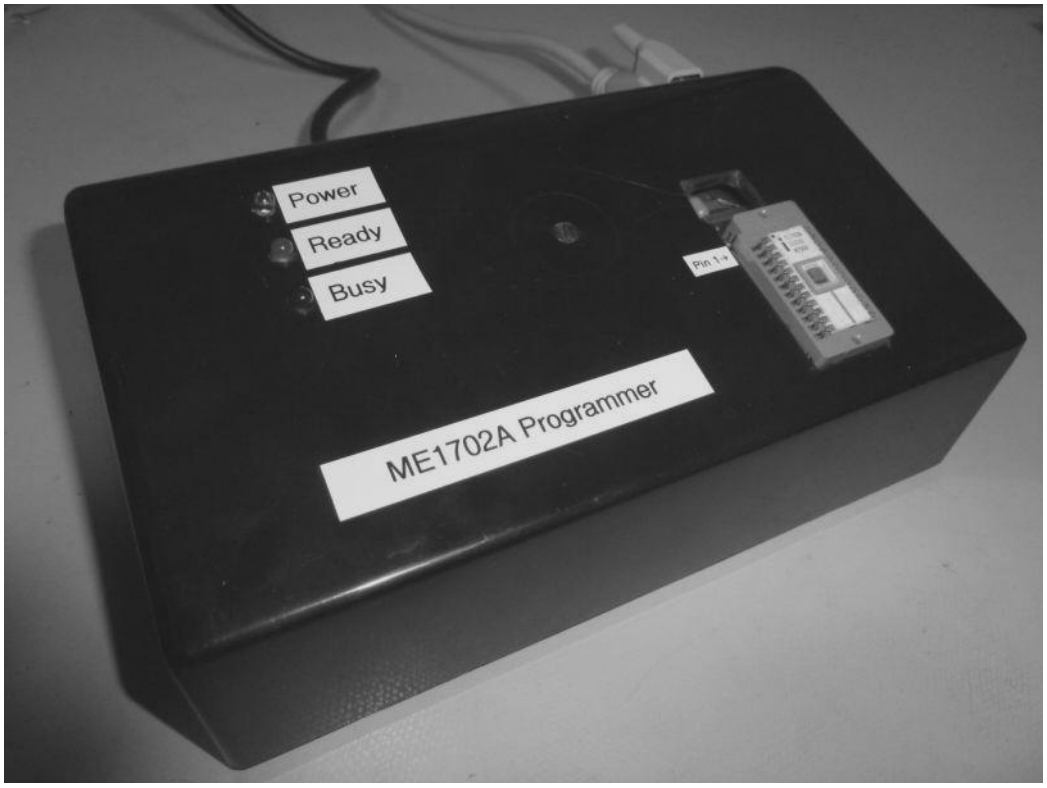
ME1702/A

1702 and 1702A EPROM Programmer

By Martin Eberhard



Intentionally Blank

ME1702/A

Martin Eberhard's 1702 and 1702A EPROM Programmer

Rev C through F PC Boards, Rev 2.01 & 2.02 Firmware

Welcome

Congratulations on purchasing the newest programmer for the oldest EPROMS. This programmer was designed at least thirty years after the 1702 and 1702A EPROMS that it programs became obsolete. The computing power of the ME1702/A is about ten times that of any computer designed to use these antique EPROMs, and its internal EEROM storage space is equivalent to 56 1702s.

The ME1702/A is a full-function programmer designed to program and read 1702A and 1702 (plain) PMOS EPROMs. The ME1702/A connects to a computer with an RS-232C serial port, requiring a terminal program that can send and receive text files, and supports XOFF/XON handshaking. You can transfer files to and from the ME1702/A in either Intel Hex or Motorola S-record format. You can also edit the EPROM file while it is in the ME1702/A's buffer.

Rev 2.00 firmware added a significant new feature: you can upgrade new firmware via the serial port - see section 8 for details. Rev 2.01 adds automatic page-address detection for file downloads. (See PA command.) Rev 2.02 firmware fixed a bug when the user chose to continue programming after a voltage was reported to be out of bounds.

A few words of caution:

Be respectful of the voltages within the ME1702/A. Obviously, the incoming 120V or 240V are dangerous. There are 60V on the EPROM's pins during programming, so **touching the EPROM during programming can be dangerous**. High voltage is present on the EPROM pins whenever the red 'Busy' light is lit.

The ME1702/A Programmer is designed to run on either 100VAC-130VAC or 200VAC-260VAC, user-selectable via the externally-accessible fuse drawer. 50 Hz and 60 Hz are both acceptable. **Please set the line voltage correctly before plugging the unit into the wall!** See Section 5.1.

The ME1702/A Programmer comes as a kit that must be assembled and adjusted before use. Basic assembly skills and tools are required to assemble the PC board assembly, and to build the wiring harnesses. Some rudimentary woodworking or plastics skills and tools are required to construct the enclosure. I've tried to make this manual thorough - I recommend reading it through before starting to assemble the ME1702/A. Have fun assembling!

This manual describes rev C through F PC boards. Rev D mainly improves some clearances inside the cabinet. Rev E corrects a few silkscreen errors, and improves clearances for the PIC ICP connector. Rev F replaced the trimpots (because the previous ones became obsolete), and made some silkscreen and copper labeling improvements.

-Martin Eberhard
8 October 2021

ME1702/A Revision History

PCB	Firmware	Date	Change Notes
A		28 Dec 2011	Created. The voltage regulators had inadequate precision, and drifted with temperature. Layout problem with fuse.
	1.00	19 Dec 2011	Original code, never tested
	1.01	8 Jan 2012	Added interrupt-driven UART routines. Never tested.
B		25 Jan 2012	Improved voltage regulator design. Improved clearance from V4 to the large transformer in the chassis. This version has a mistake in the pinout of V4, and requires rework.
	1.02	27 Jan 2012	Improved Smart-Programming algorithm. Better ^C testing. First version debugged with (reworked rev A) hardware.
	1.03	12 Feb 2012	Support plain 'S9' as an S-Record EOF. This version (and earlier versions) has a bug in the ER command, which will cause an ER to read more bytes into the buffer than requested by the user.
	1.04	15 Feb 2012	Fixed the ER bug, added code to measure and bounds-check power supply voltages during read and program operations. Added code to checksum the buffer. Added ES command.
	1.05	20 Feb 2012	Detect a backwards EPROM. Check for non-blank EPROM in the requested programming range, with option to continue
	1.06	19 Mar 2012	Support 1702 (plain). (**Must change R4 to 100 ohms**, and adjust power supply voltages a little higher.) Correct a few manual mistakes. Add DS command and a few help screens.
	1.07	20 Mar 2012	Test for ^C after each byte during programming. Improve downloading files that span several EPROMs: mismatched Page Address is not an error - it just doesn't load into the buffer. Also report the number of records that did load into the buffer.
C		29 May 2012	Corrected V4 layout mistake and a few silkscreen errors. Changed trimpots to ones that can also be adjusted from the top, and located them at the hole for the ZIF socket lever. Added ground testpoint there for testing. Tidied up board and beefed up ground while I was at it. Renamed device ME1702/A (was ME1702A) to reflect 1702 and 1702A support. Changed both fuses from 1A to 3/4A.
D		25 Sep 2012	Increased clearances to transformers by moving the top board edge down, moving the two trim pots down, and moving the transistors on the left side of the board down. Increased clearance between ground point at the lower left corner and the mounting screw. Drive RS232 handshake outputs high. Some general layout cleanup as well.
	2.00	21 Jan 2013	Rev 2.00 firmware supports firmware downloading via the serial port, in conjunction with ME Loader Kernel 1.00. Changed R4 to 220 ohms, and corrected pinout instructions for Transformer T1. (Thanks Corey!) Changed values for data output resistors R45, R46, R75, R76, R81-R84 to 10K
E		4 Feb 2014	Correct some silkscreen errors (including R45, R46, R75, R76, R81-R84), improve clearance for PIC ICP connector.
		11 Feb 2015	Minor typo corrections
	2.01	17 Aug 2015	Add automatic page-address detection mode. Improve some messages by adding 'range' when operating on less than 256 bytes.
		23 Feb 2017	Page 19: don't perform step 8 with an EPROM installed.
		10 Aug 2017	Change RS232 connections from DTE to DCE
	2.02	4 Oct 2020	Mention firmware version 2.02. (No other manual changes.)
F	2.03	21 Feb 2021	Add support for Rev F PC board, and for Textool Socket Adapter. Remove special instructions for rev B boards. Correct a few mistakes.
	2.04	8 Oct 2021	Change R19 to 13K in the assembly instructions and BOM (matching the schematic)
	2.05	27 July 2025	Correct wiring harness instructions

Contents

Section 1. ME1702/A Programmer Assembly

Assembly requires basic electronics skills, a decent soldering iron and solder, needle-nosed pliers, diagonal cutters, wire strippers, and a couple of screwdrivers. Construction of the enclosure requires a drill, and router table (or similar tool) to cut irregular holes in plastic. A good connector-pin crimping tool is not absolutely required, but helps.

Take your time to install all components in their correct locations, with the correct orientation. Install all components flush to the PC board, and with good, clean soldering. Inspect your work when you are done.

The silkscreen on the PC board is verbose, mainly to help you assemble it correctly and to aide in debugging. But the silkscreen may not be perfect. When in doubt, refer to these assembly instructions.

Be careful with diode type and orientation: the diode’s stripe must align with the stripe on the silkscreen. The silkscreen has an abbreviation of the diode number, to aide in putting the correct diode in each location.

Pay attention to the orientation of the three electrolytic capacitors. Reversing these capacitors can cause some excitement.

There is logic to the order of assembly - the smaller components first, the larger ones later. Also, unusual components are installed first, so that bulk components will not be installed in the wrong places. For example, there are four TO-92 devices that are NOT MPSA42 transistors. These devices are installed first, to minimize the chance of the wrong device being installed.

Most unlabeled 1/4W resistors are 1K-ohm, 5% resistors. The rest of the 1/4W resistors are labeled to help you get the right resistors in the right locations.

All unlabeled 1/2W resistors (all of the 1/2W resistors) are 6.8K-ohms.

When installing IC sockets and connectors, check their orientation, and make sure they seat completely against the PC board with no pins bent under. I suggest soldering them in place with just one or two pins, then re-heating these solder connections while pressing the component to the board, to get them nice and tight. Solder the rest of the pins once the component is flush to the PC board.

Prior to Rev F boards, Four components (the ZIF socket the 3 LEDs) are installed on the solder-side of the board, so that they will protrude through the Enclosure when assembled. Rev F addas two trimpots to the revers-mounted components. These are installed last.

Wire colors are of course optional, but I recommend using these colors for standardization.

This manual has check boxes next to every step, so you can check off each step when it is complete. Some of the check boxes are at the left margin; others are the left column of tables.

1.1 Printed Circuit Board Assembly

Step 1. Install the following 1/4 W, 1% resistors.

√	Qty	Locations	Value	Digikey Part Number
	1	R3	12 ohms 1%	S12CACT-ND
	3	R48,R78,R79	10 K-ohms 1%	10.0KXBK-ND
	3	R47,R77,R80	154 K-ohms 1%	154KXBK-ND
	2	R1 R38	324 ohms 1%	324XBK-ND
	2	R2 R40	976 ohms 1%	976XBK-ND

Step 2. Install the following 1/4 W, 5% resistors. Note that R4 is mislabeled on the Rev C through E PC boards, but should be 220-ohm 1/4 W.

√	Qty	Locations	Value	Digikey Part Number
	1	R19	13 K-ohms 5%	13KQBK-ND
	1	R4	220 ohms 5%	220QBK-ND
	2	R55,R59	330 ohms 5%	330QBK-ND
	13	R37,R45,R46,R50,R54,R57, R61,R75,R76,R81-R84 *	10 K-ohms 5%	10KQBK-ND
	10	R12,R13,R21,R23,R25, R27,R29,R31,R33,R35	100 ohms 5%	100QBK-ND

Step 3. Install 1K, 5% resistors in the remaining 1/4 W resistor locations:

√	Qty	Locations	Value	Digikey Part Number
	34	R14,R15,R18*,R39,R41-R44,R49, R51-R53,R56,R58,R60, R62-R67, R71-R74,R85-R93	1 K-ohm 5%	CF14JT1K00CT-ND

* R18 is mislabeled ‘10K’ on rev D PC boards. Install a 1K resistor, despite the label.

Step 4. Install the following diodes. **Be very careful** about orientation and also **be sure** to put the correct diode in each location. The diodes are often difficult to read - use magnifying glass if you are not sure. Bend the leads such that the last 2 or 3 digits of the diode number will be visable when the diodes are soldered in place.

√	Qty	Locations	Value	Digikey Part Number
	6	D4-D9	1N4004	641-1311-1-ND
	2	D2,D3	1N4148	1N4148TACT-ND
	1	D1	1N5817	1N5817-TPCT-ND
	1	Z1	NZX36B,133	568-5902-1-ND
	1	Z2	1N4744A	1N4744A-ND
	1	Z3	1N4758A	1N4758ADICT-ND

Step 5. Install 6.8 K-ohm, 1/2 W resistors in the following 20 locations.

√	Qty	Locations	Value	Digikey Part Number
	20	R5-R11,R16,R17,R22,R24,R26, R28,R30,R32,R34,R36, R68-R70	6.8 K-ohm, 1/2W, 5%	6.8KH-ND

Step 6. Install 2 DIP sockets:

√	Qty	Locations	Value	Digikey Part Number
	1	U3	16-pin DIP	A100206-ND
	1	U2	40-pin DIP	3M5471-ND

Step 7. Install the following 13 capacitors:

√	Qty	Locations	Value	Digikey Part Number
	7	C1-C5,C7,C8	0.1 uF, 100V	399-4330-ND
	6	C10,C12-C16	1 uF, 25V	445-2857-ND

Step 8. Install 2 10K, 2W resistors in the following locations:

√	Qty	Locations	Value	Digikey Part Number
	2	R94,R95	10 K-ohm, 2W, 5%	OY103KE-ND

Step 9. Install the 7909 (TO-220 package) voltage regulator in location V1. Bend its leads such that it lies flat against the board, and then solder it in place. No screw is necessary.

√	Qty	Locations	Value	Digikey Part Number
	1	V1	7909	NJM7909FA-ND

Step 10. Install the following four TO-92 devices. Be very sure you put the right component in each location. Double-check their orientation. When installed, these components should have about 1/8 inch of lead between the PC board and their plastic bodies.

√	Qty	Locations	Value	Digikey Part Number
	1	Q1	ZTX451	ZTX451-ND
	2	V2,V3	LM317L	LM317LZ-ND
	1	Q12	2N3906	2N3906FS-ND

Step 11. Install 33 MPSA42 transistors in the following locations. Double-check their orientation. Note especially that the components on the right side of U1 face the opposite was as those on the left. When installed, these components should have about 1/8 inch of lead between the PC board and their plastic bodies.

√	Qty	Locations	Value	Digikey Part Number
	33	Q2-Q11,Q13-Q33,Q35,Q36	MPSA42	MPSA42FS-ND

Step 12. Install two fuse-clips at FS1. Look at the clips carefully - they must be installed with the correct orientation. They have a feature bent into them that prevents the fuse from sliding out the end. The clips should be installed such that this feature is toward the outside, away from where the fuse goes. Make sure the fuse clips are installed flush to the board. Note that the board has been laid out to accept two different types of clips - you can use either type.

√	Qty	Locations	Value	Digikey Part Number
	2	FS1	Fuse clip	F4186-ND
	0	FS1	Alternate fuse clip	BK-6005-ND

Step 13. Install the following electrolytic capacitors. Be sure to install them with the correct orientation. The negative sign on each capacitor should be farthest from the + sign on the PC board.

√	Qty	Locations	Value	Digikey Part Number
	2	C6,C9	330 uF 35V	732-8742-1-ND
	1	C11	330 uF 180V	338-1603-ND

Step 14. Install the following TO-220 voltage regulators. Each should first be attached to a heat sink loosely with a screw and a nut, with the nut against the TO-220 device. (You can put a tiny bit of heat sink grease on the mating surface of the TO-220 device if you want, though it is not necessary.) Then insert the assembly into the PC board and solder in place, including the heat sink’s pin. Finally, tighten the screw and nut thoroughly.

√	Qty	Locations	Value	Digikey Part Number
	2	V4,V5	Vertical TO-220 heat sink	HS368-ND
	2	V4, V5	Pan-head screw, 4-40,3/8"	H781-ND
	2	V4, V5	Nut, 4-40	H216-ND
	1	V4	LM317	LM317TFS-ND
	1	V5	7805	MC7805CTGOS-ND

Step 15. Install the large power transistor at location Q34. First, clip the transistor into its heat sink, sliding the spring-clip all the way onto the transistor and heat sink. Then insert the assembly into the PC board so that the transistor is all the way down to the board and the spring-clip’s ends are through the PC board holes. Position the heat sink so that it is a tad off of the surface of the PC board, so that it can not short out traces beneath it.

√	Qty	Locations	Value	Digikey Part Number
	1	Q34	TO-247 heat sink	WA-T247-101E-ND
	1	Q34	2SC4468	2SC4468-ND

Step 16. Install two 0.1″ headers in the following locations. Be sure to seat them all the way against the PC board. Install J3 such that the flat alignment tang is away from the nearby mounting screw-hole, as indicated by the silkscreen marking. (You can install an additional ground pin at the GND point near U2. These GND pins are only for attaching meter and scope probes during diagnostics and adjustments.)

√	Qty	Locations	Value	Digikey Part Number
	1	J3	4-pin 0,1″ header	A19431-ND
	1	J2	6-pin 0,1″ header	A31116-ND

Step 17. Install a 6-pin 0.156″ header in location J1. Be sure to seat it all the way against the PC board. J1's alignment tang should be closest to the PC board edge, as indicated by the silkscreen marking.

√	Qty	Locations	Value	Digikey Part Number
	1	J1	6-pin 0,156″ header	WM4624-ND

Step 18. **REVERSE MOUNTED on Rev F boards ONLY:** Install 2 trim-pots in locastions VR1 and VR2, and set them to the center of their ranges. (Note that the type of trim-pot depends on the PC board revision.) For rev F boards **ONLY**, these trim-pots should be mounted on the solder side of the board. For the rev C boards, try to push the trim-pots as far from the board edge as you can - they will be a little close to the big transformer when the chassis is assembled.

√	Qty	Loc.	Value	Digikey Part Number	
				Rev C-E PCB	Rev F PCB
	2	VR1,VR2	2K trim-pot	K4A23-ND	3362P-202LF-ND

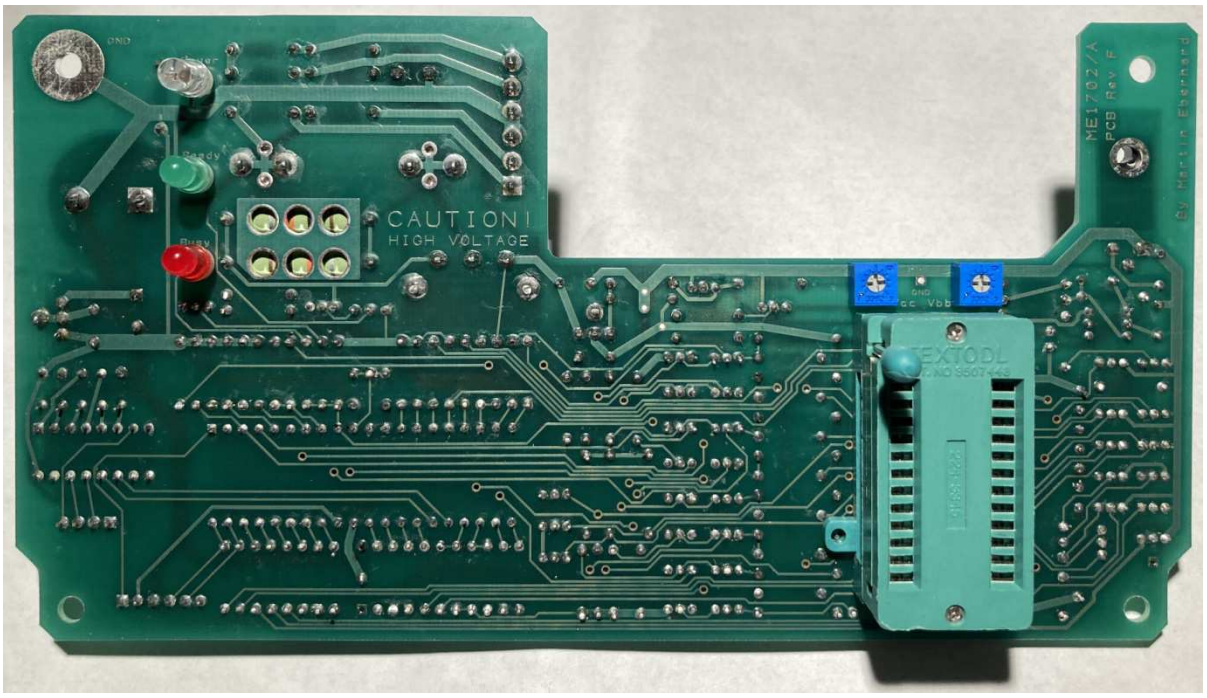
Step 19. **REVERSE-MOUNTED COMPONENT**: Optionally, attach the ZIF socket to the Socket Adapter. Install the Textool ZIF socket (or socket with extender) **on the solder-side of the PC board.** Install the socket with its handle toward the cut-out side of the PC board - the handle should be closest to the square pin-1 pad. It is **very important** to open the socket (handle perpendicular to the PC board) before you solder it in place. Failure to open the socket before soldering will cause the socket to open incorrecty during use.

√	Qty	Locations	Value	Digikey Part Number
	(1)	U1 Solder Side	Textool Socket Adapter	3M 224-1275-39-0602J
	1	U1 **Solder Side**	Textool 24-pin ZIF	3M2402-ND

Step 20. **REVERSE-MOUNTED COMPONENTS**: Install the following 3 LEDs **on the solder-side of the PC board.** (Optionally, slide a stand-off onto each LED before inserting the LED in the PC board.) Pay attention to the LED orientation - the shorter lead on the LED (closest to the flat side of the LED's plastic body) goes in the square hole in the PC board. Push the LED and stand-off snugly against the PC board when you solder them in place. Be careful not to melt the plastic casing on the nearby electrolytic capacitors with your soldering iron.

√	Qty	Locations	Value	Digikey Part Number
	3	LED1-LED3	optional LED standoff	8311K-ND

	1	LED1 **Solder Side**	Blue T1 LED	C503B-BCS-CV0Z0461-ND
	1	LED2 **Solder Side**	Green T1 LED	754-1265-ND
	1	LED3 **Solder Side**	Red T1 LED	754-1266-ND



Six Reverse-Mounted Components Installed in rev F board

Step 21. Insert a 3/4-amp fast-blow fuse in the fuse clips at FS1. (This fuse helps protect the circuitry against damage from backwards or wrong components in the ZIF socket. Further protection from this mistake is done by firmware.)

√	Qty	Locations	Value	Digikey Part Number
	1	FS-1	3/4-amp fast fuse	283-2631-ND

Step 22. ☐ Inspect your work! Check for shorts, inadequate solder, component orientation, etc. This is a high-voltage circuit, and construction mistakes will probably damage components.

Note that the two ICs are not yet installed on the PCBA. This will be done after some power supply checkout.

1.2 Enclosure Fabrication

√	Qty	Hammond Part No.	Description	Allied Part No.
	1	1595EBK	1955-Series Instrument Console	806-4180

Hammond intended the 1955-Series Instrument Console to be used with its plastic box on the bottom and the aluminum panel as the top. For the ME1702/A, the box is upside down, with the aluminum panel as the bottom.

The plastic box needs several holes drilled into it, and three larger non-round holes cut into it. The simple holes can be drilled with a hand-drill or

a drill press. The larger non-round holes can be cut using a Dremel tool or a router table with a small router bit, using a file to tighten the corners where needed. To avoid scratching the plastic with the router table, cover the relevant faces of the plastic box with masking tape. When drilling the plastic, use a slow drill speed to minimize melting. You can remove any melted plastic cleanly with a sharp knife, provided that you covered the surface of the plastic with masking tape before drilling.

Step 1. Make copies of the four templates in the **Enclosure Templates** section of this manual. Cut these templates out neatly along their perimeters. ☐ Cut out the inside of the outlines for the power-entry and DA-9 connector holes in both rear templates, and also cut out the inside of the ZIF socket hole in the Top Inside template. Punch out the four mounting post holes in the Top Inside template with a normal 1/4" hole punch, or cut them out with an Exacto knife.

Step 2. Tape the right side template to the outside-left of the box, aligning the edge of the template with the open edge of the box. Center it as ☐ best you can on between the rounded edges of the box. Drill two 3/16″ holes at the designated places, through the template and the box.

Step 3. Tape the inside-rear template to the inside of the rear of the box, aligning the edge of the template with the open edge of the box. ☐ Center it as best you can between the rounded edges of the box, behind the support posts inside the box. Use a router table with a 1/8″ straight bit to cut holes for the power entry and the DA-9 connector, using the template as a guide. Tidy up with a file as needed.

Step 4. Tape the outside-rear template to the rear of the box, being aligning the holes in the template with the holes that you cut in the previous ☐ step. Drill two 3/16″ and four 5/32″ holes at the designated places, through the template and the box.

Step 5. Tape the top template to the inside of the top of the box, positioning it over the four mounting posts in the box. Use a router table with a ☐ 1/8″ straight bit to cut the correct hole for the ZIF socket, using the template as a guide. Use a 13/64″ bit to drill the three LED holes. Tidy up with a file and a sharp knife as needed.

Step 6. Stick four rubber feet diagonally on the corners of the metal cover plate, such that they don’t cover the screw holes. Stick them on the surface that does not have the white plastic coating.

√	Qty	Component	Digikey Part Number
	4	0.3″ high adhesive foot	SJ5523-0-ND

1.3 Chassis Wiring and Assembly

Step 1. Construct the RS-232 Cable Subassembly

Use three 9-inch pieces of AWG 22 stranded wire (or 9 inches of 3-strand ribbon cable) to build the RS-232 Cable Subassembly, using the following components.

√	Qty	Manufacturer/ Part No.	Description	Digikey Part No.
	1	Molex/22-01-3047	4-position 0.1″ connector housing	WM2002-ND
	3	Molex/08-50-0113	AWG22-AWG26 contact	WM1114-ND

	1	3M/8R09-N001	9-pin female DSUB (DA-9) conn.	3M10608-ND

If you do not use a real crimping tool for the Molex pins, then solder them after you crimp them. If you used individual wires instead of ribbon cable, twist the three wires lightly and tie them together into a neat bundle with small wire ties or waxed dental floss. The pinout for this harness is as follows:

√	DA-9 Pin	Molex pin	Signal
	5	1	Ground
	3	2	Data In (into the ME1702/A)
	2	3	Data Out (out of the ME1702/A)
	6	4	Optional DSR (always true, out of the ME1702/A)*

* Molex pin 4 not connected on C boards.

Step 2. Construct the Multifunction Inlet Subassembly

The multifunction inlet comprises four subcomponents:

1. The power cord inlet, which takes a standard IEC power cord
2. The power switch
3. The main fuses, hidden inside a fuse drawer
4. The fuse drawer reverses to configure the power supply for either 120V or 240V operation

√	Qty	Manufacturer/ Part No.	Description	Supplier/ Part Number
	1	Delta Electronics/ 10C3	Multifunction Inlet	Digikey/ 1144-1003-ND
	1	TE Connectivity/ 61793-1	#6 Ring terminal	Digikey/ A29900CT-ND

Connect wires to the inlet with free ends that will be connected later. Leave about 8″ of wire on these free ends. The green wire (that goes to the ring terminal) should be about 3″ long.

Be careful - the plastic on the switch is wasily damaged by overheating the terminals while soldering. (You might use fast-on connectors for the switch connections instead of soldering.)

See the Chassis Wiring Schematic in Section 9.3, and double-check with an ohm meter.

√	Inlet Pin	Also to	Wire Color	Length
	A	1	White	8″
	B	4	Black	8″
	2		Blue	8″
	3		Yellow	8″
	G		Green	3″ to ring terminal

Step 3. Configure the Multifunction Inlet

√	Qty	Description	Digikey Part Number
	2	3/4A 1-1/4″ fast fuse, 20mm x 5mm	FSC-750MA

Pry the fuse drawer from the Multifunction Inlet with a screwdriver. Install two 3/4A fuses in the plastic clips. Insert the drawer back in the inlet, such that the triangular pointer points to the 110V-120V arrow. (For 240 volts, install the drawer the other way.)

Step 4. Assemble the Components into the Chassis

√	Qty	Manufacturer/ Part No.	Description	Supplier/ Part Number
	1		Inlet Subassembly	From Step 2 above
	2		3/8" 4-40 flat-head screw	Allied/9501
	2		3/8" 4-40 pan-head screw	Allied/9301
	4		4-40 nut	Digikey/H216-ND
	4		3/8" 6-32 pan-head screw	Allied/9308
	2		#6 flat washer	Digikey/H778-ND
	4		#6 lock washer	Digikey/H240-ND
	4		6-32 nut	Digikey/H220-ND
	4		3/8" 4-40 self-tapping screw	Allied/9575
	3		#4 flat washer	Digikey/5205820-3-ND
	1	Hammond/ 186E120	0.5A 120V Center-Tapped Transformer, 120V/240V	Digikey/ HM4702-ND
	1	Hammond/ 186B24	0.25A 24V Center-Tapped Transformer, 120V/240V	Digikey/ HM2129-ND

☐ **Step 4A:** Insert the Inlet Subassembly through its hole in the rear of the box (from the outside) with the switch closest to the edge of the box. Screw it down with two 1/2″ 4-40 flat-head screws and two 4-40 nuts.

☐ **Step 4B:** Install the PC Board Assembly in the chassis, guiding its LEDs and ZIF socket into their holes in the chassis. Screw it in place with four 4-40 self-tapping screws and three #4 flat washers, pushing the board down toward the front (shallow end) of the chassis as you tighten it. Install the ground lug on the Inlet Assembly instead of a flat washer in the corner closest to the inlet.

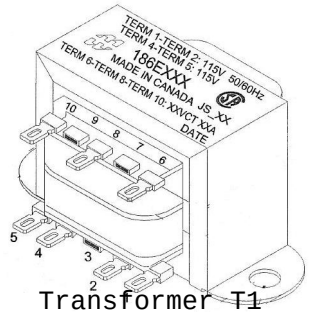
☐ **Step 4C:** Insert the RS232 Cable Subassembly through the rear of the box (from the outside), and screw it down with two 1/2″ 4-40 pan-head screws and two 4-40 nuts. Plug the cable into J3 on the PCBA.

☐ **Step 4D:** Screw T1 (the larger transformer, 0.5A 120V) in place in the rear panel of the box, with its contacts toward the box opening. (**For rev C boards:** if the transformer touches VR1 and VR2 on the PCBA, insulate the transformer with a piece of electrical tape.) Use two 6-32 screws, two #6 flat washers, two #6 lock washers, and two 6-32 nuts. The flat washers go between the transformer and the box, to clear the plastic ribs inside the box. If any of the ribs are in the way of the washers, trim them off with a sharp knife.

☐ **Step 4E:** Screw T2 (the 0.25A, 24V transformer) in place on the right side of the chassis (Left side, when the unit is upside down), with its contacts toward the box opening. Use two 6-32 screws, two #6 lock washers, and two 6-32 nuts. If any of the ribs are in the way of the transformer body (preventing the mounting tabs from touching the sidewall), trim them off with a sharp knife. Use a long pair of needle-nose pliers to get the washers and nuts in place.

Step 5. Build the Line-Voltage Wiring Harness

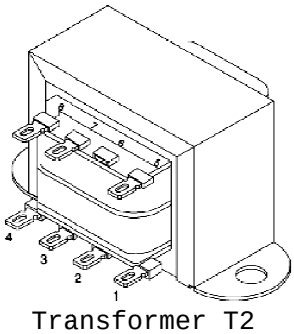
Wire the Line-Voltage harness as shown below. (The transformer pins are numbered as shown on the right.) Keep solder connections tight and low so that they won't short against the bottom panel when it is installed.



Transformer T1

Cut the wires to length, keeping the harness back, close to the rear of the box. Don't nick the wire when you strip the ends.

Route the wires neatly toward the back of the box, and tie them together with small wire ties or waxed dental floss.



Transformer T2

√	From Inlet Subassembly	To T1 pin	To T2 pin	Wire Color
	White wire	1	1	White
	Yellow wire	2	2	Yellow
	Blue wire	4	3	Blue
	Black wire	5	4	Black

Step 6. Build the Isolated Wiring Harness

Use the following components, and some AWG 18 wire to create the isolated wiring harness.

√	Qty	AMP Part No.	Description	Digikey Part No.
	1	09-50-3061	6-pos. 0.156” conn. housing	WM2104-ND
	6	08-52-0072	AWG18-AWG24 contact	WM2302-ND

The AMP connector will plug into J1 on the PCBA. Make your wires long enough to reach comfortably, but not overly-long. Look at the PC board to determine which end is pin 1 - it is the farthest from the back panel. Double-check the wiring against the labels on the PC board.

If you do not use a real crimping tool for the contacts, solder them after you crimp them.

Route the wires neatly and plug the connector into J1 on the PCBA. Tie them together into a neat bundle using small wire ties or waxed dental floss.

√	From J1 pin	To T1 pin	To T2 pin	Wire Color
	1	10		White
	3	8		Blue
	2	6		White
	5		8	Yellow
	4		7	Blue
	6		5	Yellow

Inside of completed chassis (Rev F board shown)

Step 7. Inspect your work!

- ☐ Check for correct wiring, stray wire strands, loose crimps, etc. If your solder joints on the transformers are too high and might touch the bottom panel, you can insulate them with a few dabs of silicone glue.

Section 2. Checkout and Adjustment

Basic checkout requires a volt meter and either a computer terminal (such as the most excellent Wyse WY-30) or a PC with a serial port and a terminal emulation program. These tests are sequential - if you find a defect, do not move on until the defect has been corrected!

2.1 Line-Voltage Wiring Harness Checkout

☐ **Step 1.** Make sure the power switch (on the Multifunction Inlet) is off. Unplug J1 from the PCBA if it is plugged in.

☐ **Step 2.** Install a line cord in the inlet and plug it into the wall. Measure the AC voltage between pins N and L of the Multifunction Inlet. If they are not 117VAC (or so), then your power cord is installed incorrectly or defective.

☐ **Step 3.** Measure the AC voltage between A and B (on the switch). If this is NOT 0V, then the switch is on, or you have a wiring mistake. Now turn the switch on. You should measure 117V (or so) across A and B.

☐ **Step 4.** Measure between pins 1 and 5 of T1, and between 1 and 4 of T2, again looking for 117VAC. Check your wiring if you do not see this.

☐ **Step 5.** Measure between pins 1 and 2 of each transformer. If you do not see 117VAC, then you either have the fuse drawer installed backwards, or you have a wiring mistake. (Note that you must move the fuse in the fuse drawer to the other position if you turn the drawer over.) Measure also between pins 3 and 4 of each transformer. Again, you should see 117VAC.

Step 6. Measure the following voltages between the pins of the 6-pin connector that plugs into PCBA J1. The voltages are approximate, and will read high when there is no load. Correct defects before moving on.

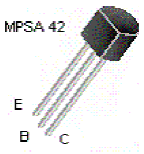
√	From	To	AC Voltage
	1	2	120 VAC
	1	3	60 VAC
	2	3	60 VAC
	5	6	24 VAC
	4	5	12 VAC
	4	6	12 VAC
	3	4	Close to 0V

2.2 Basic PCBA Checkout

☐ **Step 7.** Turn the power off, and plug the harness connector into J1. Hook the ground lead of your voltmeter to the GND pin near the corner of the PCBA opposite from J1. Turn the power on. The blue LED should light.

☐ **Step 8.** Measure DC voltage at U2 pins 11 and 32. Both should be +5V +/-0.25V. If not, power off and debug the logic supply first.

☐ **Step 9.** Measure the voltage at the emitter of Q4 and Q10. (Q4 pinout is shown to the right.) The voltage should be -9V +/-0.45V. If not, power down and debug the -9V supply.



☐ **Step 10.** Measure the voltage at both ends of the fuse FS1. You should see around +92V at both ends. If this voltage is only at one end, the fuse has blown, probably because of an assembly problem on the board. Power down and debug as needed.

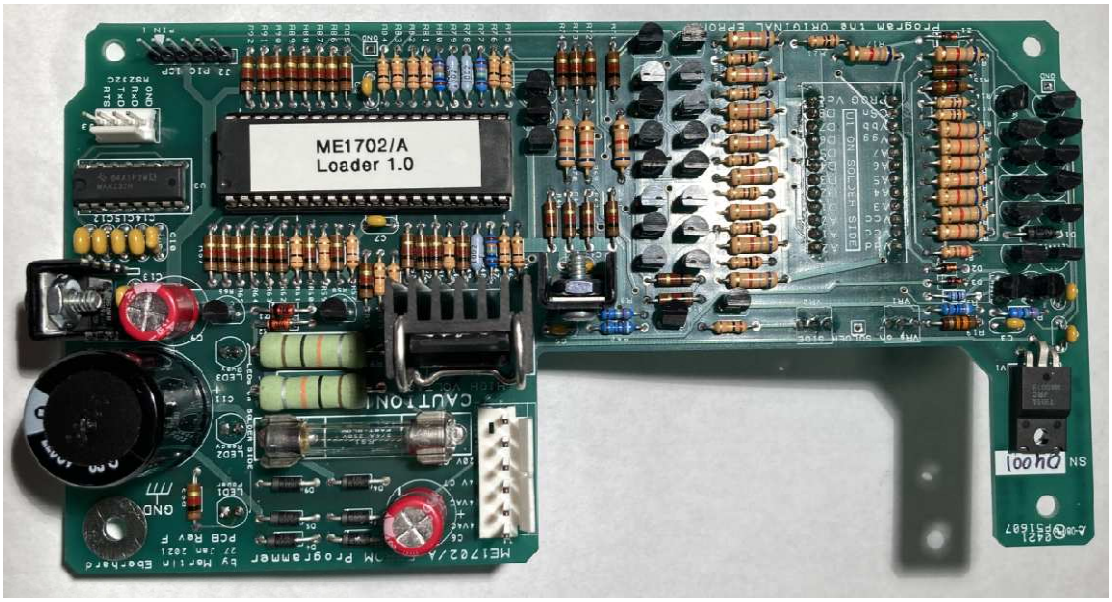
☐ **Step 11.** Measure the voltage at all pins of the ZIF socket, U1. They should all be close to 0V (less than 1.1V).

Step 12. Power down and unplug the ME1702/A Programmer. Install two ICs in the following locations, paying attention to orientation. Be careful not to bend any leads as you insert the ICs.

√	Qty	Locations	Value	Digikey Part Number
	1	U3	MAX232 RS-232 transceiver	296-1402-5-ND
	1	U2	PIC16F1517 microcontroller	PIC16F1517-I/P-ND

U2 is a PIC microcontroller with internal flash memory. You must use a PIC that has been pre-programmed with the ME Loader Kernel 1.0, or program it in place yourself, using a PC, a Microchip PICkit-3 programming device, and my program file. (J2 is the PICkit-3 compatible in-circuit programming connector for this purpose.) The PIC must also be loaded with the ME1702/A Programming Firmware, which can be loaded via the serial port - see section 8.

DIP IC leads usually come with their leads bent slightly outward to facilitate machine-insertion. Straighten the leads before inserting them into their sockets. (Lay one side of the IC against your workbench and press the IC against the table, straightening all the leads on one side at once. Repeat for the other side.)



Completed Rev F PC Board, Component Side

2.3 Microcontroller Bring-Up

Step 1. Plug a terminal (or a PC with a terminal program) into the ME1702/A Programmer's serial port connector, making sure to connect the transmit signal (TxD, pin 3) from the ME1702/A to receive signal of the terminal and the receive signal (RxD, pin 2) from the ME1702/A to the transmit signal of the terminal. For a normal PC, you will need a symmetrical "null modem" DA-9 to DA-9, as shown in Section 6.

Step 2. Set up the terminal (or terminal program) this way:

Baud Rate	9600
Stop Bits	1
Parity	None
Handshake	XON/XOFF

Step 3. Plug the ME1702/A Programmer into the wall and turn it on. On the terminal screen, you should see a sign-on banner and a prompt like this:

```
*====================================*
*             ME1702/A               *
*====================================*
*    1702A & 1702 EPROM Programmer   *
*         By Martin Eberhard         *
*            Version 2.02            *
*====================================*

Type ? for command list

>
```

If you do not see this banner, check the following:

√	Check
	Is the terminal setup right? - baud rate, etc. as above
	If you are using a PC (maybe with an RS-232C - to - USB dongle), check that this is all working correctly. You can roughly test it with a loop-back from pin 2 to pin 3.
	TxD, RxD and GND wiring from the ME1702/A Programmer to the terminal. Are TxD and RxD reversed?
	RS232 Wiring harness - correct pins? Good connections?
	Are IC2 and IC3 inserted correctly?
	Is IC1 in fact programmed? (With the right code?)

Step 4. Two LEDs should now be lit. Debug if not.

√	LED	State	Meaning
	LED1 Power	On	Correct
		Off	LED1 Orientation?
	LED2 Ready	On	Correct
		Off	LED2 Orientation?
	LED3 Busy	On	PC board short?
		Off	Correct

Step 5. Type '?' to see a full help screen. You will try out all of the commands on this screen in the following sections.

2.4 tm 1Microcontroller-Assisted Checkout and Adjustment

NOTE: The following steps involve dialog with the ME1702/A's monitor. The monitor's prompt is '>'. You should type what is in **bold**, and the monitor will respond as indicated. If you turn off the power between steps, you will need to repeat the dialog up to the point where you are working, when you power back on.

☐ All voltages are referenced to ground. Reconnect the voltmeter ground lead to the PCBA GND pin. If the terminal and/or ME1702/A Programmer are off, then turn them back on.

Step 1. Test Master Power-Off

☐ Measure the voltage at the right side (farthest from the heat sink) of R95. This should be close to 0V. If not, debug the circuit that includes Q35 and R36.

Step 2. Test Master Power and Low-voltage Control

Monitor dialog:

```
>TM 1
Master Power on
>
```

Now, only one LED should be lit:

√	LED	State	Meaning
	LED1 Power	On	Correct
		Off	LED1 Orientation?
	LED2 Ready	On	Mistyped command? PC board short?
		Off	Correct
	LED3 Busy	On	PC board short?
		Off	Correct

Measure the voltage at the right side (farthest from the heat sink) of R95.

√	Measurement	Meaning	Debug
	14.5V to 18V	Correct operation	--
▨	>18V	Yikes! Power off!	Q26, R58 circuit
▨	0V to 14.5V	Problem	• Correct orientation for Z2 • Correct component in Z2 • Q35, R36 circuit

Step 3. Test Low-voltage Unregulated Output

☐ Measure the voltage at the right side (striped end) of D2. This should be close to the same voltage measured in the previous step. If not, debug V3, V4, and Q34.

Step 4. Test Low-voltage Regulated Outputs

Measure the following voltages, and debug as needed:

√	Measure	Measurement	Debug if incorrect
	U1 Vcc	5.0V +/- 0.25V	V4, Q22 circuits. Check values: R38, R40
	U1 Vbb	5.0V +/- 0.25V	V3, V2, Q18 circuits. Check values: R1, R2, R3
	U1 Vdd	5.0V +/- 0.25V	Q1, C10, Q12, D1 circuits
	U1 Vgg	5.0V +/- 0.25V	Q4, Q11, Q12 circuits
	U1 PROG	5.0V +/- 0.25V	Q6 circuit
	U1 CSn	5.0V +/- 0.25V	Q5 Circuit

Step 5. Test Chip Enable Signal

Monitor dialog:

```
>TC 1
  -CE Signal On
>
```

☐ U1 CSn should now be less than 0.5V. If not, debug Q5 circuit.

Step 6. Test PROG Signal

Monitor dialog:

```
>TP 1
-PROG Signal On
>
```

☐ U1 PROG should now be less than 0.5V. If not, debug Q6 circuit.

Step 6. Test -9 Volt Control

Monitor dialog:

```
>T9 1
   Vdd & Vgg off
Vcc 48 volts off
    -9V Power on
>
```

Measure the following voltages, and debug as needed:

√	Measure	Measurement	Debug if incorrect
	U1 Vdd	-9,0V +/- 0.45V	Q1, C10, Q12, D1 circuits
	U1 Vgg	-9,0V +/- 0.45V	Q4, Q11, Q12 circuits

Step 7. Test the Address Drivers

Monitor dialog:

```
>WA 0
>
```

☐ Measure each address pin on U1. They should all be less than 0.5V.

Monitor dialog:

```
>WA FF
>
```

☐ Measure each address pin on U1. They should all be near 5V.

☐ Use the WA command to write various address values, checking to see

if what you wrote is reflected on the address pins of U1. Debug the address driver transistors as needed.

Step 8. Test the Data Drivers

(Do not perform this step with an EPROM in the socket.)

Monitor dialog:

```
>WD 0
>
```

☐ Measure each data pin on U1. They should all be less than 0.8V.

Monitor dialog:

```
>WD FF
>
```

☐ Measure each data pin on U1. They should all be close to 5V.

☐ Use the **WD** command to write various data values, checking to see if what you wrote is reflected on the data pins of U1. Debug the data driver transistors as needed.

Step 9. Test the Data Receivers

Monitor dialog:

```
>WD 00
>RD
Data Read: FF
```

☐ If you get any value besides FF, double check that you typed WD 00, and then debug the data input transistor circuits as needed.

☐ Clip a test jumper to one of the GND pins on the PC board. Use the other end of the jumper to ground one of the data pins on U1. Use the **RD** command to see that this bit now reads as 0. Repeat for each of the data bits, and debug the data input transistors as needed.

Step 10. Test High-Voltage Power Supply Control

Monitor dialog:

```
>WD FF
>WA FF
>TP 0
-PROG Signal off
>TC 0
  -CE Signal off
>TV 1
   -9V Power off
Vcc 48 volts on
>
```

Now, two LED should be lit:

√	LED	State	Meaning
	LED1 Power	On	Correct
		Off	LED1 Orientation?
	LED2 Ready	On	Mistyped command? PC board short?
		Off	Correct
	LED3 Busy	On	correct
		Off	LED3 orientation? PC board short?

Measure the voltage at the right side (farthest from the heat sink) of R95.

√	Measurement	Meaning	Debug
	68V to 75V	Correct operation	--
	>75V	Yikes! Power off!	Correct component in Z3?
	4.5V to 68V	Problem	• Correct orientation for Z3 • Correct component in Z3 • Q35, Q36 circuits
	0V to 4.5V	Problem	Q35, Q36 circuits

Step 11. Test High-Voltage Power Supply Unregulated Output

☐ Measure the voltage at the right side (striped end) of D2. This should be close to the same voltage measured in the previous step. If not, debug V3, V4, and Q34.

Step 12. Test and Adjust High-Voltage Power Supply Outputs

For Rev C and later boards, use a plastic adjustment tool, or insulate your screwdriver shaft, as the trim-pot will be at high voltage.

Measure the following voltages, adjust and debug as needed. (For rev C and later boards, these adjustments can be made from the top, using an insulated screwdriver through the chassis hole for the ZIF socket.)

√	Measure	Measurement	Meaning
	U1 Vcc (pins 12, 22, 23)	42V to 52V	Adjust VR2 for 48.0V +/- 0.1V. If you can't get the right voltage, debug V4 circuit
		>52V	Debug V4 circuit. Check value of R37
		5.25V to 42V	
		0V to 5.25V	Q22 circuit
	U1 Vbb (pin 15)	54V to 64V	Adjust VR1 for 59.9V +/- 0.1V. If you can't get the right voltage, debug V2 and V3 circuits
		>64V	Debug V2 and V3 circuits. Check value of R19
		5.25V to 54V	
		0V to 5.25V	Debug Q13 circuit
	U1 Vdd (pin 24)	47.8V to 48.1V	Correct operation
		0V to 47.79V	Debug Q1 circuit
		<0V	Debug Q10, Q12 circuits
	U1 Vgg (pin 16)	47.8V to 48.1V	Correct operation
		8.5V to 47.79V	Debug Q11 circuit
		<8.5V	Debug Q4, Q12 circuits. Check orientation and value of Z1
	U1 PROG (pin 13)	47.8V to 48.1V	Correct Operation
	U1 CSn (pin 14)	47.8V to 48.1V	Correct Operation
	U1 A0-A7	47.8V to 48.1V	Correct operation
	U1 D1-D8	47.8V to 48.1V	Correct operation

Step 13. Test the Pulsed Power Signals, Vdd and Vgg

Monitor Dialog

```
>TD 1
   -9V Power off
   Vdd & Vgg on
>
```

Measure the following voltages, and debug as needed:

√	Measure	Measurement	Meaning
	U1 Vdd	0V to 0.7V	Correct operation
		0.7V to 5V	Incorrect transistor in Q1? (Should be ZTX451, not MPSA42)
		>5V	Debug Q1, circuit. Check orientation of D1
		<0V	Debug Q10, Q12 circuits
	U1 Vgg	9.5V to 13V	Correct operation
		>13V	Check 48V adjustment in step 12. Debug Q11 circuit. Check orientation and value of Z1
		<9.5V	

Step 14. Test High-Voltage Logic Signals

☐ Use the **WA**, **WD**, **TP**, and **TC** commands (as above) to test the address, data, PROG and CSn signals. Logic high should read 47.9V +/- 0.2V. Logic low should read between 0V and 0.7V.

Step 15. Reset When Done

☐ Monitor Dialog

```
>RE

*====================================*
*             ME1702/A               *
*====================================*
*    1702A & 1702 EPROM Programmer   *
*         By Martin Eberhard         *
*            Version 2.02            *
*====================================*

Type ? for command list

>
```

Step 16. Test with an EPROM

☐ Install a 1702A EPROM in the ZIF socket, and repeat steps 1 through 14, but skipping step 8. This EPROM may get programmed while you test, so plan to erase it when you are done. Minimize the time you spend on step 13, so that you do not leave Vdd active for an extended time. This may damage the EPROM. I suggest connecting the voltmeter to Vdd, then typing ‘TD 1’ and looking at the voltage, then typing ‘TD 0’. Move the voltmeter lead to Vgg and repeat.

This concludes the checkout. You can turn on and off various signals further if you like, measuring the results on the pins of U1. When you are done, you can type **RE** to reset the ME1702/A Programmer, which will power-off all pins.

Section 3. Functional Testing

Power-off the ME1702/A Programmer, and screw on the bottom panel. Turn it right side up. Connect it to a computer with a terminal program that can send and receive files. Set up the terminal program for 9600 baud, 1 stop bit, no parity. This program expects a display screen that is at least 24 rows of 80 columns, so adjust the display of your terminal appropriately.

Power-on the ME1702/A, and see that your terminal program can talk to it.

Note: The ME1702/A Programmer uses a ‘Page Address’ when uploading and downloading files. The Page Address is set by the user (with the **PA** command), and defines the high address byte in the hex files. During uploads, this page address is inserted in the hex records. During downloads, the record data is only loaded into the buffer if the high address byte in the hex record matches the user-set Page Address. (See **PA** command.)

During downloads, the hex records are checked for valid record types, correct checksum, legitimate hexadecimal characters, correct record count (for Motorola S5 records). Any errors during download will generate a brief error message and bump the error count.

The record count, loaded record count (records with matching Page Address), and error count are displayed, and then reset whenever an end-of-file record is encountered.

Note that no command is required to start downloading to the ME1702/A Programmer. The ME1702/A simply detects a valid Intel Hex (any line that starts with ‘:’) or Motorola record (any line that starts with ‘S’). (Interestingly, you could mix and match S-records and Intel Hex records in the same download...)

All numbers (typed by you and printed by the ME1702/A) are hexadecimal. For hex file transfers, the allowable characters are {0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F}. For other operations, you can also use lower-case {a,b,c,d,e,f}.

3.1 Basic Buffer Operations and File Transfer

Step 1. Display the Default Buffer Data

Monitor dialog:

```
>DB
00: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
10: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
20: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
30: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
40: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
50: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
60: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
70: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
80: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
90: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Buffer checksum: 00
>
```

Step 2. Fill the Buffer with a Constant

Monitor dialog:

```
>FB 55
Buffer filled with 55
>DB
00: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
10: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
20: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
30: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
40: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
50: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
60: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
70: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
80: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
90: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
A0: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
B0: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
C0: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
D0: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
E0: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
F0: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
Buffer checksum: 00
> FB AA
Buffer filled with AA
>DB
00: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
10: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
20: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
30: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
40: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
50: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
60: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
70: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
80: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
90: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
A0: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
B0: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
C0: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
D0: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
E0: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
F0: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
Buffer checksum: 00
>
```

Step 3. Edit the Buffer

Monitor dialog:

```
>MB 10
10: AA 01 AA 02 AA 03 AA 04 AA 05 AA 06 AA 07 AA 08
18: AA 09 AA <control-C>

>DB
00: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
10: 01 02 03 04 05 06 07 08 09 AA AA AA AA AA AA AA
20: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
30: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
40: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
50: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
60: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
70: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
80: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
90: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
```

```
A0: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
B0: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
C0: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
D0: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
E0: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
F0: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
Buffer checksum: 33
>
```

Step 4. Upload Buffer Contents to your Computer as an Intel Hex File

You will need to use your terminal program to capture the file in your computer. I suggest calling the file INTEST.TXT.

For this demonstration, I am arbitrarily setting the page address to 0x68 - you will see the result in the file.

Monitor dialog:

```
>PA 68
Page address: 68
>UI          {start file capture before hitting Return}
:10680000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAE8
:1068100001020304050607080 9AAAAAAAAAAAAAA5
:10682000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAC8
:10683000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAB8
:10684000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA8
:10685000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA98
:10686000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA88
:10687000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA78
:10688000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA68
:10689000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA58
:1068A000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA48
:1068B000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA38
:1068C000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA28
:1068D000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA18
:1068E000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA08
:1068F000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAF8
:00000001FF
>
```

{Stop capturing and close the file on your computer}
Use a text editor to examine the file INTEST.TXT, and make sure it transferred correctly.

Step 5. Upload Buffer Contents to your Computer as a Motorola S-record File

You will need to use your terminal program to capture the file in your computer. I suggest calling the file STEST.TXT.

For this demonstration, I am arbitrarily setting the page address to 0x31 - you will see the result in the file.

Monitor dialog:

```
>PA 31
Page address: 31
>US          {start file capture before hitting Return}

S1133100AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA1B
S1133110010203040506070809AAAAAAAAAAAAAAD8
S1133120AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAFB
S1133130AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAEB
S1133140AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAADB
S1133150AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAACB
S1133160AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABB
S1133170AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAB
```

```
S1133180AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA9B
S1133190AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA8B
S11331A0AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA7B
S11331B0AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA6B
S11331C0AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA5B
S11331D0AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA4B
S11331E0AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA3B
S11331F0AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA2B
S9030000FC
>
```

{Stop capturing and close the file on your computer}

Use a text editor to examine the file STEST.TXT, and make sure it transferred correctly.

Step 6. Test downloading files to the ME1702/A Programmer, using the two files we just created. First we will fill the buffer with something different, to be sure. (If you are paranoid, power-cycle the ME1702/A.) Note that we set the page-address to match the page address in the hex file - otherwise nothing will get loaded into the buffer.

Monitor Dialog:

```
>FB 99
Buffer filled with 99
>DB
00: 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99
10: 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99
20: 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99
30: 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99
40: 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99
50: 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99
60: 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99
70: 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99
80: 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99
90: 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99
A0: 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99
B0: 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99
C0: 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99
D0: 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99
E0: 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99
F0: 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99 99
Buffer checksum: 00
>PA 68
Page address: 68
```

{Now, start sending the file INTEST.TXT to the ME1702/A}

```
>:10680000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAE8
:1068100001020304050607080 9AAAAAAAAAAAAAA5
:10682000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAC8
:10683000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAB8
:10684000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA8
:10685000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA98
:10686000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA88
:10687000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA78
:10688000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA68
:10689000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA58
:1068A000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA48
:1068B000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA38
:1068C000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA28
:1068D000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA18
:1068E000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA08
:1068F000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAF8
```

```
:00000001FF
Records: 11, Records loaded in Page 68: 10, Bad Records: 00
>DB
00: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
10: 01 02 03 04 05 06 07 08 09 AA AA AA AA AA AA AA
20: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
30: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
40: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
50: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
60: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
70: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
80: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
90: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
A0: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
B0: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
C0: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
D0: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
E0: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
F0: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
Buffer checksum: 33
>
```

You can test with the file STEST.TXT the same way. Remember that its page address is 31.

Step 7. Test backwards EPROM detection - to the firmware, a backwards EPROM looks a lot like no EPROM is installed. Perform this operation with no EPROM installed.

Monitor Dialog:

```
>EP
Please be sure the EPROM is inserted correctly, with pin 1
closest to the socket handle. Ready to program (Y/N)? Y
All bytes of this EPROM are FF. Is it inserted backwards? Is it really a 1702A?
Abort
>
```

3.2 EPROM Reading and Programming

Here, you need a blank 1702A EPROM - preferably several of them. Known-good EPROMs would be nice. You will also want an EPROM eraser, as you will be filling them with junk.

Step 1. Low-voltage Operations

Power-on the ME1702/A Programmer, and then install a blank 1702A EPROM in the ZIF socket, with its pin 1 closest to the ZIF socket handle.
Monitor Dialog:

```
>EB
EPROM is blank
>

{or...}
Error Address: XX  EPROM: ZZ
{perhaps several errors}
Fail
>
```

Whether or not the EPROM is blank, you can read it back and see what it contains:

Monitor Dialog:

```
>ER
Success
```

```
>DB
00: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
10: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
20: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
30: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
40: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
50: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
60: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
70: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
80: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
90: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Buffer checksum: 00
>
```

(Obviously, if the EPROM was not blank, it would not read as all 0's and the checksum would be different.) You can now compare the buffer to the EPROM. Then, you can change the buffer data to force a failure.

Monitor Dialog:

```
>EC
EPROM matches buffer
>MB 85
85: 00 77 00 <control-C>
>EC
Error Address: 85  Buffer: 77  EPROM: 00
Fail
>
```

Step 2. High-Voltage Operations

First, create some interesting data.

Monitor Dialog:

```
>FB 55
Buffer filled with 55
>MB
00: 55 1 55 2 55 4 55 8 55 10 55 20 55 40 55 80
08: 55 AA 55 <control-C>
>MB 9A
9A: 55 12 55 34 55 56 55 78 55 9A 55 BC
B0: 55 DE 55 F0 55 <control-C>
>DB
00: 01 02 04 08 10 20 40 80 AA 55 55 55 55 55 55 55
10: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
20: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
30: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
40: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
50: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
60: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
70: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
80: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
90: 55 55 55 55 55 55 55 55 55 55 12 34 56 78 9A BC
A0: DE F0 55 55 55 55 55 55 55 55 55 55 55 55 55 55
B0: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
C0: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
D0: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
```

```
E0: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
F0: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
Buffer checksum: 3C
>
```

Now, program an EPROM. The default algorithm is the P+4P smart algorithm, which will program the EPROM until it matches the buffer, then over-program it 4 times the number of times it took to match the buffer. Each dot printed during programming represents one pass through the range of EPROM being programmed. Many 1702’s will match the buffer on the first try, requiring only 4 more passes to complete the programming In the following example, it took two passes before the EPROM matched the buffer.

Raise the handle of the ZIF socket, put a blank 1702A in the socket, with pin 1 up, closest to the handle, and then push the ZIF socket handle down to lock the EPROM in place.

Monitor Dialog:

```
>EB
EPROM is blank
>EP
1702A selected
PROG pulse = 2.5 mS, 19% duty cycle
P+4P Smart programming algorithm enabled
Please be sure the EPROM is inserted correctly, with pin 1
closest to the socket handle. Ready to program (Y/N)? Y
P+4P Smart Programming..
P=02. 4P over-programming........
Verifying
EPROM matches buffer
>
```

If you get any error messages, try again with another EPROM, to determine if the problem is with the EPROM or the ME1702/A Programmer.

Clear the buffer, and then calculate the EPROM’s checksum. It should be the same as it was in the buffer:

Monitor Dialog:

```
>FB 0
Buffer filled with 00
>ES
EPROM checksum: 3C
>
```

Read the EPROM back into the buffer and have a look. If all goes well, it will go like this:

Monitor Dialog:

```
>ER
Success
>DB
00: 01 02 04 08 10 20 40 80 AA 55 55 55 55 55 55 55
10: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
20: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
30: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
40: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
50: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
60: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
70: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
80: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
```

```
90: 55 55 55 55 55 55 55 55 55 55 12 34 56 78 9A BC
A0: DE F0 55 55 55 55 55 55 55 55 55 55 55 55 55 55
B0: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
C0: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
D0: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
E0: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
F0: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
Buffer checksum: 3C
>
```

Congratulations, your ME1702/A EPROM Programmer appears to function correctly. If you are still reading this, try typing ‘?C’ for an Easter egg in the code.

Section 4. Programming Algorithms

You can choose between two programming algorithms, using the **PC** command. The exact timing of these algorithms depends on which EPROM type has been selected with the ET command: The PROG pulse width is 2.5 mSec with a 19% duty cycle for 1702As, and is 10 mSec with a 1.9% duty cycle for 1702s. See the timing diagrams in section 7's Programming Timing subsections for details.

Some 1702A data sheets recommend the P+4P Smart Programming Algorithm, others recommend the simple algorithm, with the cycle-count typically set to 32 decimal (20 hex). With limited testing, the smart algorithm seems to work perfectly on all 1702A's, regardless of the manufacturer's recommendation.

The Intel 1702 data sheet recommends a simple algorithm with 5 cycles of 20 mSec pulses, (or 10 cycles of 10 mSec, or 20 cycles of 5 mSec, etc.) Since the ME1702/A generates 10 mSec PROG pulses when in 1702 mode, 10 cycles meet the Intel spec.

With either algorithm, you can program the entire EPROM (the default), or just a portion of the EPROM, by specifying the starting address and the byte count in the **EP** command. Note that most 1702A data sheets seem to imply that you should only program the entire EPROM at once, starting at address 00 and proceeding in order to address FF. Programming a portion of an EPROM is not specified, though it appears to work fine.

4.1 Simple Programming Algorithm

Setting the number programming passes to anything greater than 00 will select the simple programming algorithm.

Programming Phase: The simple programming algorithm first prints "Programming" on the console. Then the buffer contents are written to the EPROM repeatedly, as many times as you specified. After each pass through the EPROM range, a period is printed on the console.

Verifying Phase: Once programming is complete, "Verifying" is printed on the console. The EPROM is verified by comparing it to the buffer, reporting errors to the console.

4.2 'P+4P' Smart Programming Algorithm

Setting the number of programming cycles to 00 (**PC 00**) selects the P+4P Smart Programming Algorithm.

Programming Phase: This algorithm first prints "P+4P Smart Programming" on the console. Then it writes one complete pass through the EPROM and then compares it to the buffer. This is repeated (keeping count in P) until the EPROM matches the buffer. After each pass through the EPROM range, a period is printed on the console.

During the programming phase, the EPROM is deemed to match the buffer if all bits that are '1' in the buffer are also '1' in the EPROM. If any additional bits are '1' in the EPROM, no amount of additional programming cycles will make them become '0' - these errors will be caught during the verify stage.

If the EPROM does not match the buffer after 256 passes during the P phase, then an error is reported to the console, and programming is aborted.

Over-Programming Phase: When the EPROM does match the buffer, the value for P is printed on the console, followed by "4P over-programming". The EPROM is then programmed another 4 times P passes. After each pass through the EPROM

range, a period is printed on the console.

Verifying Phase: Once programming is complete, “Verifying” is printed on the console. The EPROM is verified by comparing it to the buffer, reporting errors to the console.

Note: Many EPROMS will program successfully on the first programming pass, so that the EPROM will see a total of 1 + 4 = 5 passes.

4.3 Programming Time

Nominally, each programming pass through an entire 1702A will take about 3.3 seconds. However, any byte that is to be programmed as 00 will be skipped, since this is the erased state of a 1702A data byte. Skipped bytes take much less time.

Nominally, each programming pass through an entire 1702 will take about **2.2 minutes** - meaning that the default 10-pass programming procedure will take around 22 minutes. For the 1702, any byte that is to be programmed as FF will be skipped (reducing the programming time), since this is the erased state of a 1702 data byte.

Note that you can abort programming by typing control-C.

Section 5. ME1702A Commands

5.1 EPROM Commands

The EPROM commands generally deal with the 256-byte EPROM, and the 256-byte buffer in the ME1702A. For Compare, Program, and Read commands, the EPROM address is always the same as the buffer address.

For these EPROM commands, <beg> is the beginning address, both for the EPROM and for the buffer. <cnt> is the byte count for the command. <cnt>=00 is interpreted as <cnt>=100 hex. If you don't enter values for <beg> and <cnt>, they both default to 00. If you enter only one value, it is assumed to be <beg>, and <cnt> defaults to 00 (which means 100h). The EPROM and buffer addresses will wrap around to 00 once they pass FF.

>**EB <beg> <cnt> Blank-Check EPROM**

EB starts at address <beg> and checks <cnt> bytes of the EPROM to see if they are blank (data=00 for 1702As, data=FF for 1702s). Any non-blank bytes are reported to the console. If all bytes in the specified range are blank, this command responds with "EPROM is blank".

>**EC <beg> <cnt> Compare EPROM to Buffer**

EC compares <cnt> bytes of the buffer to the EPROM, starting at address <beg>. Any differences are reported to the console. If all bytes are the same, this command responds with "EPROM matches buffer".

>**EP <beg> <cnt> Program EPROM from Buffer**

EP programs <cnt> bytes of the EPROM with data from the buffer, the EPROM and buffer addresses both starting at <beg>. The programmed range is verified when programming is complete, and any errors are reported to the console. If the programmed range matches the buffer when done, then this command will respond with "Success".

Before programming, the relevant states of the ME1702A (EPROM type, programming algorithm, programming cycles) are displayed, and you are asked if you want to proceed.

Typing control-C during programming will abort cleanly, leaving the EPROM in a reasonable state.

See the **Programming Algorithms** section for additional details.

>**ER <beg> <cnt> Read EPROM into Buffer**

ER reads <cnt> bytes of data from the EPROM into the buffer, both starting at address <beg>. This command always responds with "Success".

>ES <beg> <cnt> Compute EPROM Checksum

ES reads and adds together <cnt> bytes of data from the EPROM, starting at address <beg>. Only the low byte of the sum is kept. This command responds with "EPROM checksum: XX".

>ET <0/1> Set EPROM Type

ET 0 sets up the ME1702A to program 1702As, setting the PROG pulse width to 2.5 mS and the duty cycle to 19%. ET 0 also selects the P+4P Smart Programming algorithm.

ET 1 sets up the ME1702A to program 1702s, setting the PROG pulse width to 10 mS and the duty cycle to 1.9%. ET 1 also sets the Programming Cycles to 10.

If you just type ET, then the 1702A is selected.

You can change the programming algorithm and programming cycles (with the PC command) after you set the EPROM type. However, the only way to change the PROG pulse width and duty cycle is with the ET command.

>PC <cnt> Set Programming Pass Count to <cnt>

PC sets the number of programming passes through the EPROM to <cnt>. PC 00 selects the P+4P Smart Programming Algorithm. If you don't type a value for <cnt>, then 00 (P+4P Smart Programming Algorithm) is assumed.

?E Help with EPROM Commands

?E prints a help screen for these EPROM commands.

5.2 File Transfer Commands

>**PA Automatic Page Address Mode** (default)

PA (with no parameter) selects automatic Page Address mode. When downloading a file to the ME1702/A, the Page Address will be taken from the high address byte of the first received record. When uploading a file from the ME1702/A, the Page Address will be 00.

>**PA <pa> Set Page Address**

PA <pa> sets the Page Address for uploads and downloads to <pa>.

The Page Address is the (fixed) the upper byte of the 2-byte addresses used in Intel Hex records and Motorola S-records. See the **Page Addresses** subsection under **ME1702A Programmer Usage** for further details about the Page Address, and how it is used during uploading and downloading.

>**UI Upload Buffer as Intel Hex**

UI prints the entire buffer contents on the console as Intel Hex files. The high address byte for each record is the Page Address. All records are 16 bytes long, and the transfer ends with an Intel Hex end-of-file (type 01) record. To upload into a file, start the file capture after you type UI, but before you type <return>.

>US Upload Buffer as Motorola S-Records

US prints the entire buffer contents on the console as Motorola S-record files. The high address byte for each record is the Page Address. All records are 16 bytes long, and the transfer ends with a Motorola S-record end-of-file (S9) record. To upload into a file, start the file capture after you type US, but before you type <return>.

>: Begin Intel hex record

If the record type is 00 (data record) and the high byte of the address matches the Page Address, then the data in this record is put in the buffer memory starting at the address specified by the low address byte in the record.

If automatic Page Address mode is selected and this is the first record since reset or since the last end-of-file record was received, then the Page Address will be set to the high address byte of this record. For all other records, if the high byte of the address does not match the Page Address, then the record is simply not loaded into the buffer.

If the record checksum does not match the checksum computed by the ME1702A, then “? Csm” is printed and the error count is incremented.

Invalid hex characters (including lowercase a-f) print “? Hex” and cause the error count to be incremented.

Record types other than 00 (data) and 01 (end-of-file) print “? Rec” and cause the error count incremented.

Byte count > 00 for a type 01 record (end-of-file) print “? Rec” and cause the error count to be incremented.

The prompt is not displayed after receipt of an Intel Hex record, except as below.

If the record type is either 00 (data) or 01 (end-of-file), and the record byte count is 00, then the record count, the count of records loaded into the buffer, and error count are displayed and then cleared, and the prompt is displayed.

>S Begin Motorola S-record

If the record type is S1 (data record) and the high byte of the address matches the Page Address, then the data in this record is put in the buffer memory starting at the address specified by the low address byte in the record.

If automatic Page Address mode is selected and this is the first record since reset or since the last end-of-file record was received, then the Page Address will be set to the high address byte of this record. For all other records, if the high byte of the address does not match the Page Address, then the record is simply not loaded into the buffer.

An S5 (record count) record will compare the ME1702A’s record count to the record count in the record. If they do not match, then “? Cnt” is printed, and the error count is incremented.

If the record checksum does not match the checksum computed by the ME1702A, then “? Csm” is printed and the error count is incremented.

Invalid hex characters (including lowercase a-f), will print “? Hex” and cause the error count to be incremented.

Any record type besides S1 (data), S5 (record count), and S9 (end-of-file) will print “? Rec” and cause the error count incremented.

An end-of-file record may be a full S9 record (with byte count, address, and checksum fields), or it may be just ‘S9’. (This abbreviated S9 record is sometimes found in old S-record files.)

If the byte count for a type S5 (record count) or S9 (end-of-file) record is not 00, then “? Rec” is printed, and the error count is incremented.

The prompt is not displayed after receipt of a Motorola S-record, except as below.

If the record type is either S1 (data) or S9 (end-of-file), and the record byte count is 00, then the record count, the count of records loaded into the buffer, and error count are displayed and then cleared, and the prompt is displayed.

?F Help with File Transfer Commands

?F prints a help screen for these file transfer commands.

5.3 Buffer Commands

>DB <beg> <cnt> Display Buffer Contents

DB displays the specified range of buffer contents. Data are displayed 16 hex bytes to a line, except the first line, if its least-significant digit is not 0. Each line is preceded by a 2-digit hex address. If you don't enter values for <beg> and <cnt> then the entire buffer contents will be displayed. The checksum of the specified region of the buffer is printed last.

>FB <val> Fill Buffer with Value

FB fills the entire buffer with <val>. If you don't enter a value for <val>, then the buffer will be filled with 00.

This command responds with "Buffer filled with <val>".

>MB <addr> Modify Buffer

MB allows you to modify the buffer contents starting at address <addr>. The address and its contents are first printed on the console. If you type <return>, the contents will remain unchanged. If you type a hexadecimal number and then <return>, the value you type will replace the buffer contents at that address.

After you type <return>, the contents of the next address in the buffer are displayed, allowing you to modify that address. This continues until you type <control-C>.

Every address that ends with 0 or 8 will start a new line, displaying first the address, then the data.

?B Help with Buffer Commands

?B prints a help screen for these buffer commands.

5.4 Diagnostic Commands

These commands are intended only for diagnosing the ME1702A, especially during initial bring-up and when you want to adjust the programming voltages. They allow you to control various signals to the EPROM socket directly, so that you can measure and adjust voltages, and test functionality.

For these commands, if you don't enter a value (0 or 1), then 0 is assumed.

>TM <0/1> Test Master Power

TM 1 turns on power to the EPROM socket, TM 0 turns it off. If programming voltage is off (TV command), then Vcc will be +5 volts when you type TM 1, and 0 volts when you type TM 0. Typing TM 0 also performs T9 0.

>TC <0/1> Test Chip Select Signal

TC 1 turns the EPROM chip select on, TC 0 turns it off. since Chip Select is an active-low signal, on means at 0 volts. Off means at Vcc, which depends on TM and TV.

>**TD <0/1> Test Vdd & Vgg**

TD 1 turns Vdd and Vgg on, TD0 turns them off. These are active low, so off means at Vcc, which depends on TM and TV. When on, Vdd will be at 0 volts, and Vgg will be at 0V or about 36V below Vcc, whichever is higher. In other words, if Master Power is on and Programming Voltage is off, then Vgg will be at 0V when on. However, if you turn Programming voltage on, then Vcc = 48V, so Vgg will be around 12 volts. Typing TD 1 will also perform T9 0.

>**TP <0/1> Test PROG Signal**

TP 1 turns the EPROM PROG signal on, TP 0 turns it off. since PROG is an active-low signal, on means at 0 volts. Off means at Vcc, which depends on TM and TV.

>**TV <0/1> Test Programming Voltage**

TV 1 turns on the high-voltage programming supply, and puts Vcc to 48 volts. Typing TV 1 will also perform T9 0

>**T9 <0/1> Test -9V**

T9 1 turns on the -9V supply to Vdd and Vgg. Typing T9 1 will also perform TV 0 and TD 0

>**WA <val> Write Address**

WA writes <val> to the address pins of the EPROM socket. Every ‘0’ bit will drive the corresponding pin to 0 volts. Every 1 bit will drive the corresponding pin to Vcc.

If you don’t enter a value for <val>, then 00 is written.

>**WD <val> Write Data**

WD writes <val> to the data pins of the EPROM socket. Every ‘0’ bit will drive the corresponding pin to 0 volts. Every 1 bit will drive the corresponding pin to Vcc.

If you don’t enter a value for <val>, then 00 is written.

>**RD Read Data**

RD reads the EPROM data pins and displays the results on the screen. Note that the data write drivers interact with the read circuitry. If you want to read the data pins correctly, you need to type WD 0 first.

Note also that the pins are pulled up to Vcc. If Master Power is off, then there is no pull-up, and you will not read anything meaningful. Therefore, you should type TM 1 typing RD.

You can use a jumper wire to ground, to test each pin - ground the pin and read the result.

>**RV Read Voltages**

RV reads the pre-regulated voltage, Vcc, and Vbb. Each of these pins has a divide-by-16 resistor network. The 160bit value printed is the direct result of the A/D converter. Feel free to interpret these numbers as you please...

?D Help with Diagnostics

?D prints a help screen for these diagnostic commands.

5.5 Other Commands

>? Help

Typing a question mark prints a help screen that briefly explains all the commands.

>TE <0/1> Set Terminal Echo

TE 0 turns terminal echo off; TE 1 turns it on.

>DS Display all Settings

Displays the following states: Loader Kernel revision level, Page Address, EPROM Type, Programming Cycles/Smart Programming mode, and Echo State.

>RE Reset ME1702A Programmer

RE is just like power-cycling the ME1702A.

?L View Firmware Loader Notes

?L displays notes on loading new firmware into the ME1702/A via the serial port. Firmware loading is discussed in a later section.

^S Pause Serial Port Output

Control-S (XOFF) tells the ME1702/A to stop sending data. Any subsequent character (including XON, which is control-Q) will re-enable the ME1702/A output, and that character will be discarded by the ME1702/A.

Section 6. ME1702/A Programmer Usage

The previous sections walked you through the basic ME1702/A Programmer operation. Here are some specifications, and the procedures for common EPROM operations.

6.1 120V and 240V Operation

The ME1702/A Programmer will operate on either 100VAC-130VAC, or 200VAC-260VAC, at 50Hz or 60 Hz. Select your input voltage using the fuse drawer right next to the power cord inlet. Use a small screwdriver to pry the fuse drawer from the inlet.

For 100VAC-130VAC operation, install the 3/4A fuse in the side that has the 110-120V arrow pointing to it. Insert the drawer back in the inlet with the triangular pointer near the power cord inlet points to the 110V-120V arrow.

For 200VAC-260VAC operation, install the 3/4A fuse in the side that has the 220-240V arrow pointing to it. Insert the drawer back in the inlet with the triangular pointer near the power cord inlet points to the 220V-240V arrow.

To avoid damage to the ME1702/A, **please be sure the fuse drawer is oriented correctly for your line voltage**!

6.2 Connector Pinout

The DA-9 connector is a compatible with standard PC serial port. Rev C boards do not drive any of the handshaking signals. Rev D and later boards drive DSR high=true. All versions do not respond to any incoming handshaking signals. The connected pins are as follows:

Female DA-9 Pin	Signal
3	Data In (in to the ME1702/A)
2	Data Out (out of the ME1702/A)
5	Ground
6	DSR (Driven high=true on Rev D & later boards)

For a normal PC connection, you will need a standard “null modem” male DA-9 to male DA-9 cable like this.

Male DA-9 Pin	Signal	Male DA-9 Pin
1	Daa Carrier Detect (N/C the ME1702A)*	1
2	Data Out (out of the ME1702/A)	2
3	Data In (in to the ME1702/A)	3
4	Data Terminal Ready (N/C in the ME1702A)*	4
5	Ground	5
6	Data Set Ready (driven high by the ME1702/A)*	6
7	Request to Send (N/C in the ME1702A)*	7
8	Clear to Send (N/C in the ME1702A)*	8
9	Ring Indicater (N/C in the ME1702A)*	9

* These connections are optional. Rev C boards have no connections to any handshake pins. Rev D and later boards will drive handshake outputs high, and ignore handshake inputs.

6.3 LEDs

Three LEDs tell you the state of the programmer.

The top (blue) LED is ‘Power’, and indicates that the power switch is on. It is connected across the Microcontroller’s 5-volt supply.

The middle (green) LED is ‘Ready’. It is connected to the control signal that enables power to the EPROM. When lit, this LED indicates that no power is applied to the EPROM. It is safe to insert or remove an EPROM **only** when this LED is lit.

The bottom (red) LED is ‘Busy’. It is connected to the control signal for the high-voltage (programming) power supply. When lit, the EPROM has high voltage, and should not be touched.

6.4 Power Supply Voltage Checking

The ME1702A tests some of its internal voltages when it performs EPROM read and programming operations, and also when it ends these operations. The A/D converter and related scaling hardware have significant tolerances, so the measurements are somewhat rough. Measurements are made mainly to detect gross errors that may damage the ME1702/A or an EPROM. The software tolerances are set wide enough that you should never see a voltage error message with a correctly-functioning ME1702/A. Any voltage error message should cause you to debug the hardware.

Because of these sloppy tolerances, you cannot assume that the ME1702/A is properly adjusted just because it does not give you any ‘voltage out of bounds’ messages. You should follow the adjustment procedure (earlier in this manual) to set the voltages correctly.

If the pre-regulator voltage is ever too high, the ME1702/A will print a panic message and abort whatever it was doing. You should power off immediately, to prevent further damage. The pre-regulator produces around 15 volts when set up for EPROM reading, and around 70 volts when set up for EPROM programming. The measurement allows for some variation in these voltages - panic occurs when these voltages are significantly high.

During reading, the ME1702/A checks Vcc and Vbb. If either is too far above or below 5 volts, an error message will be printed, and you will be given the opportunity either to continue with the operation or to abort.

Also during reading, the pre-regulator voltage is checked for a reasonable voltage. If it is too high, the ME1702/A will panic (and you should shut it off). If it is too low, you will be given the option to continue.

During programming, the ME1702/A checks Vcc to see if it is roughly 48 volts. Vbb is similarly checked, to see if it is roughly 60 volts. If either is too high, the ME1702/A will panic, preventing programming. If either is too low, an error message will be printed, and you will be given the option to continue.

When any EPROM operation completes, the three voltages are checked to see if they are near 0 volts. If not, you will get an error message. You should debug the ME1702/A if this happens - otherwise you will have a hot socket, and will risk damaging EPROMS when you install or remove them.

6.5 Page Addresses

Simple Intel Hex files and Motorola S-record files have 2-byte addresses (represented as 4 hex digits). A 1702A EPROM contains 256 bytes of data, which is defined here as a page. Thus, the address in a hex file can be thought of as having 2 components: the high byte (first 2 hex digits) is the Page Address, and the low byte (second 2 hex digits) represents the address within the EPROM.

The EPROM data may be a computer program intended to run at address 0000, or at some other address. If it is intended to run at some other address besides 0000, then the high byte of the 2-byte addresses in the hex file that you will send to the EPROM will not be 00 - these bytes will be the high byte of the intended target address.

You can tell the ME1702/A what the address high byte is, using the **PA** (Page Address) command. If you specify automatic Page Address mode (by typing the **PA** command with no value), then the Page Address for downloads will be set to the high address byte from the first received record of the file. Automatic Page Address mode is the default after reset.

When you download a hex file to the ME1702/A, the high address bytes are compared to this address. Any records with a different Page Address will not be loaded into the buffer. When the download is complete, a count of the number of records that were loaded into the buffer will be printed.

Note that if you have a hex file that spans several EPROMs, you can program each EPROM sequentially by first setting the Page Address for one of the EPROMs, then clearing the buffer, and then sending the whole hex file, and then programming the EPROM. Repeat this procedure for each EPROM - setting the Page Address appropriately. Each time, only the records that match the Page Address will get loaded into the buffer - the others will be ignored.

Similarly, if you are reading an EPROM that is intended to run at an address besides 0000, you can generate the correct hex file by setting the Page Address before uploading the buffer. The Page Address that you specify with the **PA** command will be the high address byte in every record uploaded. If you have selected automatic Page Address mode (by typing the **PA** command with no value), then the Page Address for uploads will be 00.

6.6 Selecting the EPROM Type

The default EPROM type is the much more common 1702A. You can change the EPROM type to 1702 by typing ET 1. (You can change it back to 1702A by typing ET 0.) Changing the EPROM type also sets the programming algorithm to the default for each type. For 1702As, the Smart Algorithm is the default. For 1702s, the simple algorithm with 10 cycles is the default.

6.7 Selecting a Programming Algorithm

See the more extensive **Programming Algorithms** section for details about the two supported programming algorithms.

The default programming algorithm is the “P+4P” Smart Programming Algorithm, which observes how many programming passes it takes to get a successful read-back, and then programs it four times that many more passes. To select the P+4P Smart Programming Algorithm, set the Programming Count to 0:

Monitor Dialog:

```
>PC 0
P+4P Smart programming algorithm enabled
>
```

Otherwise, you can simply tell the ME1702/A Programmer how many passes to program the EPROM. 32 decimal (20 hex) is a common 1702A programming specification:
Monitor Dialog:

```
>PC 20
Programming Cycles: 20
>
```

6.8 Programming an EPROM from a File

Here is how you program an EPROM from a file, assuming the Smart algorithm is used, and assuming (for the sake of demonstration) that the EPROM took 2 programming passes to read back successfully.
Monitor Dialog:

```
{Power-on}
{insert blank EPROM in the socket}
>EB
EPROM is blank
>ER
Success {This fills the buffer with either FF or 00, depending on EPROM type}
>PA XX {XX is the address high byte for the EPROM}
Page Address: XX
>   {send S-Record or Intel Hex file to the ME1702/A}
>EP
Please be sure the EPROM is inserted correctly, with pin 1
closest to the socket handle. Ready to program (Y/N)? Y
P+4P Smart Programming..
P=02. 4P over-programming........
Verifying
EPROM matches buffer
>
```

6.9 Reading an EPROM into a File

You can read an EPROM, and save the data in a standard format (either Intel Hex or Motorola S-Record) on your computer.

Monitor Dialog:

```
{Power-on}
{insert programmed EPROM in the socket}
>PA XX {XX is the address high byte for the file}
Page Address: XX
>ER
Success
>UI    {start file capture on your computer before hitting return}
{Intel Hex file follows}
>
```

If you want the file in Motorola S-Record format, use **US** instead of **UI**.

6.10 Copying an EPROM

You can read an EPROM, and then write it into any number of blank EPROMs.

Monitor Dialog:

```
{Power-on}
{insert source EPROM in the socket}
>ER
Success
{insert blank EPROM in the socket}
>EB
EPROM is blank
```

```
>EP
Please be sure the EPROM is inserted correctly, with pin 1
closest to the socket handle. Ready to program (Y/N)? Y
P+4P Smart Programming..
P=02. 4P over-programming........
Verifying
EPROM matches buffer
{insert another blank EPROM in the socket}
>EB
EPROM is blank
>EP
Please be sure the EPROM is inserted correctly, with pin 1
closest to the socket handle. Ready to program (Y/N)? Y
P+4P Smart Programming..
P=02. 4P over-programming........
Verifying
EPROM matches buffer
>
```

Section 7. ME1702/A Theory of Operation

7.1 Architecture

The following is a very brief description of the ME1702/A Programmer's circuit operation. For further detail, see the 1702A and 1702 specifications.

The ME1702/A Programmer comprises several power supplies, a microcontroller with embedded EEROM, RAM and serial port, and a ZIF socket with a read/write interface for the 1702A EPROM.

7.2 Logic Supply and -9V Supply

Transformer T2 produces a center-tapped 24VAC, which is rectified by D4,D5, D7, and D8, and smoothed by C6 and C9 to produce unregulated +/- 20V. V5 regulates the +20V to produce 5VDC for the microcontroller. V1 regulates the -20v to produce -9VDC for the 1702A EPROM during read operations.

7.3 Microcontroller-Controlled High-Voltage Supply

The 1702A EPROM programming voltages are specified as Vbb = +12V, Vgg = -36V, Vdd = -48V, and logic signals at 0V and -48V to the EPROM's inputs. These voltages are referenced to the EPROMs Vcc, defined as 0V. When reading the EPROM, voltages are referenced to ground, defined as 5V below the voltage of the Vcc pin, since the 1702A does not have a ground pin.

To make interfacing simpler, the programming voltages are translated up by 48 volts, referenced to Vdd when Vdd is on. The voltages translate as follows:

Pin name	Specified Voltage			Translated Voltage		
	Read	Ready[1]	Program	Read	Ready[1]	Program
Vbb	5V ±5%	12V ±10%	12V ±10%	5V ±5%	60V ±0.2V	60V ±0.2V
Vcc	5V ±5%	0V	0V	5V ±5%	48V ±0.1V	48V ±0.1V
Vdd	-9V ±5%	0V	-47±1V	-9V ±5%	48V ±0.1V	0.3V
Vgg	-9V ±5%	0V	-37.5V ±2.5V	-9V ±5%	48V ±0.1V	12V ±2.5V
PROG	5V ±5%	0V	-47±1V	5V ±5%	48V ±0.1V	0.2V
CSn	0V	-47±1V	-47±1V	0V	0V	0.2V
A<7:0> 0/1[2]	0V/5V	0V/-47V	-47V/0V	0V/5V	48V/0V	0V/48V
D<8:1> 0/1[2]	output	0V	0V/-47V	output	48V	48V/0V

1. Ready for programming
2. logic 0/logic 1 voltages

Transformer T1 produces a center-tapped, isolated 120VAC, which is rectified to around 90VDC by D6 and D9, and C11. Fuse F1 will blow if too much current is drawn from this source.

Vcc is generated by an LM317 programmable voltage regulator (V4), with precision resistors R38 and R40 setting the 5V level, and R37 and VR2 setting the adjustable 48V level. The microcontroller controls Q22 to select between these two voltages.

Q34, Z2, and Z3 serve as a programmable pre-regulator, controlled by Q35 and Q36. The microcontroller can turn the EPROM's positive supplies completely off by turning Q35 on. It can switch between roughly 15V and roughly 71V by controlling Q36. (This preregulation is necessary because the maximum voltage across the LM317 is 40V, so the input voltage of the LM317 must be reduced when its output is to be 5V.) Note that R94 and R94 will get kind of warm

when the power supply is off. This is normal.

V2, V3, and surrounding components serve as a programmable voltage regulator for Vbb, capable of providing either 5V or 60V, controlled by Q13. The 59 volt output is adjusted with VR1. V3 and R3 serve as a 100 mA current limit for Vbb, as required by the 1702A spec. V2 and V3 are also fed by the Q34 pre-regulator.

Vdd is at 0V when Q1 is on, and is at -9V when EN_MINUS9 (from the microcontroller) is high. If neither is active then R5 pulls Vdd up to Vcc. The microcontroller will never turn Q1 and EN_MINUS9 at the same time.

Q1 is a different transistor than the other small NPN transistors, because Vdd sources more current (up to 300 mA) while programming. The particular transistor was chosen because it can handle 1 amp of collector current, and has a low collector-emitter saturation voltage.

When Q11 is on, Z1 will regulate Vgg to about 36 volts below Vcc, or about 11V. When EN_Minus9 is high, then Vgg is at -9V. If neither is active then R11 pulls Vgg up to Vcc. The microcontroller will never turn Q11 and EN_MINUS9 at the same time.

7.4 EPROM Read/Write Interface

Every 1702A logic signal is driven by a resistor-transistor circuit that translates TTL signals from the Microcontroller to the necessary voltages for reading and programming. These circuits use a pull-up resistor to Vcc (which may either be 5V or 48V) and a transistor that can drive the pin to 0V.

The driver circuit for each of the EPROM data pins also includes a resistor-transistor circuit that allows the EPROM to be read by the microcontroller, without ever exposing the microcontroller to the high programming voltages. Note that in order to read from the EPROM, the write-driver transistors must be turned on by the microcontroller.

Because the address and data drivers must operate at 48 volts, these circuits are pretty slow when running at 5 volts (e.g. when reading). The microcontroller waits more than 20 uS after writing an address before reading the EPROM data, to insure a correct read.

7.5 Microcontroller

All ME1702/A Programmer operations are controlled by a 40-pin PIC16F1517 microcontroller, running at 16 MHz. This PIC includes a UART that is connected to the serial port connector through a MAX232 RS232C transceiver.

The PIC has 512 bytes of internal RAM, which is used for the EPROM buffer, as well as for transmit and receive queues, variable storage, and general purpose registers.

The PIC also has 8K of 14-bit program memory EEPROM, which holds the ME1702/A Programmer’s firmware. This EEPROM can be reprogrammed in place via the 6-pin PIC ICP connector and a Microchip PICkit III programming device.

The PIC has a multi-channel A/D converter that is connected (via 3 resistor divider circuits) to Vpre, Vcc, and Vbb, so that the PIC can measure these voltages and perhaps could warn the user if voltages are out of spec.

7.6 Voltage Measurement

Three A/D converters are implemented in the Microcontroller. They each connect to the power supply via a divide-by-16.4 resistor divider. One measures the output of the pre-regulator. The second one measures Vcc, and the third one measures Vbb. The A/D converters use the PIC's 5-volt Vcc as a reference.

The PIC's Vcc is supplied by a 7805 voltage regulator, which has a tolerance of about +/-2.5%. The resistors in the divider are +/-1%. When you combine these errors, the voltage measurements have an error of about +/-4.5%. This is not good enough for detecting whether or not the power supply is properly adjusted, but it is good enough to detect many potential failure modes, including a failed pre-regulator control, and a failed power supply switch circuit.

The firmware uses these A/D converters for this purpose. Because the measurements are rough anyway, the firmware only looks at the 8 most significant bits of the A/D converter's 10-bit result.

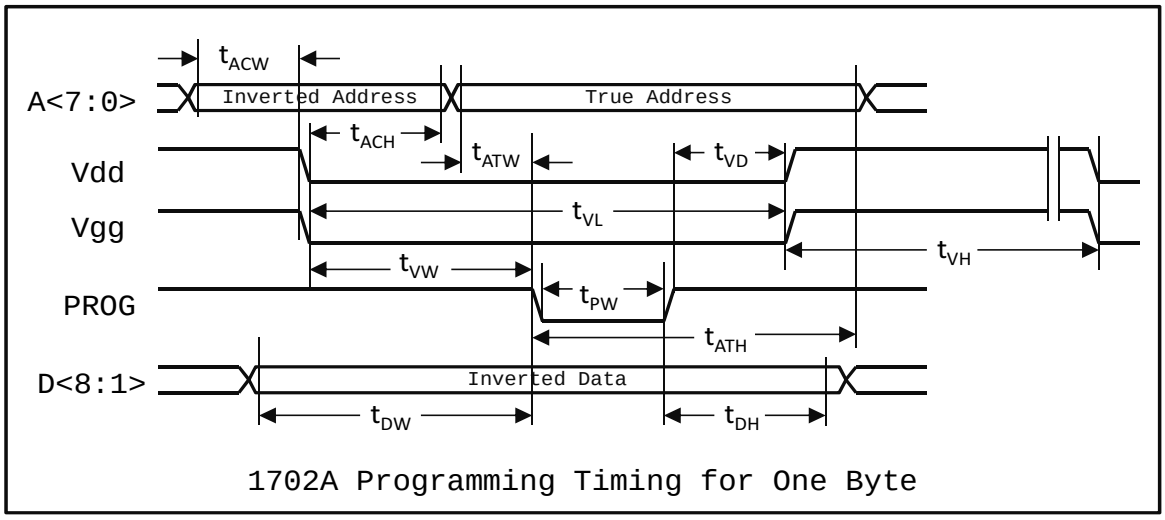
Hopefully, this logic will prevent a small failure in the ME1702/A from cascading into a larger failure, and also help prevent damaging a (scarce) EPROM.

You can work out the math for the A/D values. Here is a table of some interesting A/D values, and what they mean. (These are the A/D result limits used by the firmware.) The Minimum and Maximum Voltage columns are the extremes when all the tolerances are worst-case in either direction. The highlighted values are the limiting cases.

A/D high result	Minimum Voltage	Nominal Voltage	Maximum Voltage	Note
00	>0V	0V - 0.24V	>0V	Acceptable 'off' voltage range
01	<0.61V	0.32V - 0.56V	<0.67V	
0E	>4.4V	4.48V - 4.72V	**>4.6V**	Acceptable 5V (low voltage) Vcc
11	**<5.52V**	5.45V - 5.69V	<6.03V	
28	>12.2V	12.3 - 12.5V	**>13.4V**	Acceptable low-voltage pre-regulator output
38	**<17.5V**	17.95V -18.2V	<19.1V	
8E	>43.5V	45.5V - 45.8V	**>47.5V**	Acceptable 48V (high-voltage) Vcc
9C	**<48.1V**	50.0V - 50.3V	<52.5V	
AF	>53.6V	56.1V - 56.3V	**>58.5V**	Acceptable 60V (high voltage) Vbb
C3	**<60.1V**	62.5V - 62.8V	<65.5V	
D0	>63.7V	66.7V - 66.9V	**>69.6V**	Acceptable high-voltage Pre-regulator output
F0	**<77.3V**	76.9V - 77.2V	<80.7V	

7.7 1702A Programming Timing

Below is the timing specification for programming the 1702A EPROM, compared to the timing of the ME1702/A Programmer's 1702A mode. Note that Microchip's spec for the PIC's internal clock frequency has an accuracy of +/-10%. Note also that inaccuracy in the PIC's clock frequency will not affect the pulsed power supply duty cycle, D_V.

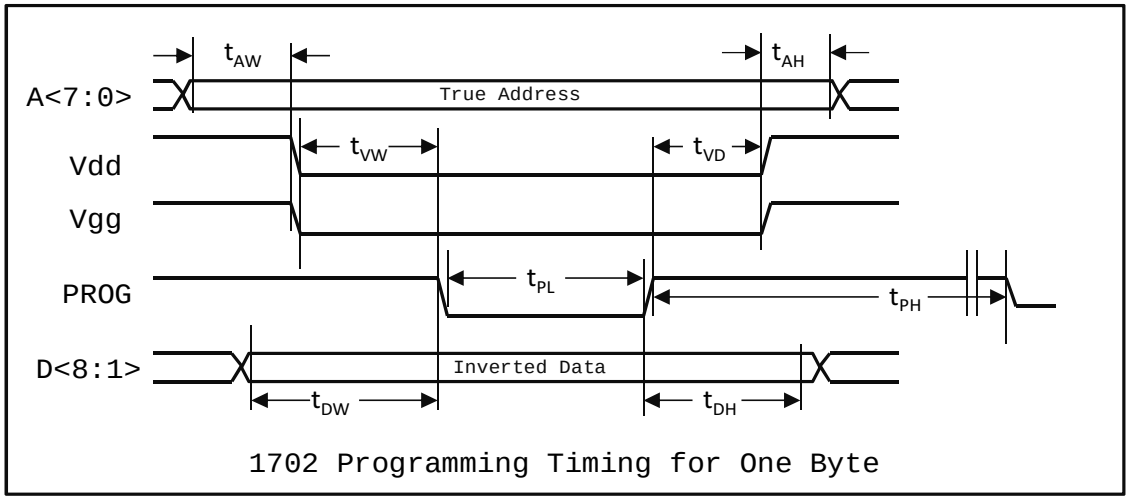


1702A Programming Timing for One Byte

Symbol	Parameter	1702A Spec		ME1702/A (±10%)
		Min	Max	
T_{PW}	PROG Pulse Width		3 mSec	2.5 mSec
T_{DW}	Data Setup to PROG↓	25 uSec		500 uSec
T_{DH}	Data Hold from PROG↑	10 uSec		10 mSec
T_{VW}	Vdd, Vgg Setup to PROG↓	100 uSec		130 uSec
T_{VD}	Vdd, Vgg Hold from PROG↑	10 uSec	100 uSec	50 uSec
T_{ACW}	Address Complement Setup to Vdd and Vgg↓	25 uSec		500 uSec
T_{ACH}	Address Complement Hold from Vdd and Vgg ↓	25 uSec		30 uSec
T_{ATW}	Address True Setup to PROG↓	10 uSec		100 uSec
T_{ATH}	Address True Hold from PROG↑	10 uSec		10 mSec
T_{VL}	Vdd and Vgg low			2.65 mSec
T_{VH}	Vdd and Vgg high			11 mSec
D_V	Vdd and Vgg Duty Cycle $= T_{VL} / (T_{VL} + T_{VH})$		20%	19.0% ±0.1%

7.8 1702 Programming Timing

Below is Intel's timing specification for programming the 1702 EPROM, compared to the timing of the ME1702/A Programmer's 1702 mode. Note that Microchip's spec for the PIC's internal clock frequency has an accuracy of +/-10%. Note also that inaccuracy in the PIC's clock frequency will not affect the programming pulse duty cycle, D_P.



1702 Programming Timing for One Byte

Symbol	Parameter	1702 Spec		ME1702/A (±10%)
		Min	Max	
T_{PW}	PROG Pulse Width		20 mSec	9.96 mSec
T_{DW}	Data Setup to PROG↓	1 uSec		500 uSec
T_{DH}	Data Hold from PROG↑	1 uSec		>> 1 mSec
T_{VW}	Vdd, Vgg Setup to PROG↓			10 uSec
T_{VD}	Vdd, Vgg Hold from PROG↑	1 uSec		10 uSec
T_{AW}	Address Setup to Vdd,Vgg↓	0 uSec		501 uSec
T_{AH}	Address Hold from Vdd,Vgg↑			10 mSec
T_{PL}	PROG low			10 mSec
T_{PH}	PROG high			508 mSec
D_P	PROG Duty Cycle $= T_{PL} / (T_{PL} + T_{PH})$		2%	1.88% ±0.01%

7.9 Backwards EPROM Detection

Attempting to program an EPROM that is installed backwards can damage the ME1702/A. In particular, the fuse will blow, and one or two of the transistors might be damaged. Prior to programming, the ME1702/A firmware attempts to detect an EPROM that is installed backwards, to try and avoid damage.

A backwards 1702A reads as all bytes=FF. Reading a backwards 1702A does not harm the ME1702/A (and does not seem to harm the EPROM either). Note that unprogrammed bytes in a 1702A read as 00 - unlike the plain 1702 and unlike newer EPROM types.

Prior to programming a 1702A, the ME1702/A will check for an EPROM that is completely filled with FFs, and will abort with a backwards-EPROM error message if this is the case. This prevents damage to the ME1702/A from trying to program a backwards EPROM, as well as damaging a 1702 with the wrong algorithm.

Unfortunately, this technique for detecting a backwards EPROM cannot detect a backwards 1702 because unprogrammed bytes read as FF. The user must therefore be extra careful when working with 1702 EPROMs.

Section 8. Downloading Firmware via the Serial Port

You can load new ME1702/A firmware via the serial port. Most likely, you will load a firmware update that I have emailed to you. However, if you have the programming skills and I am not providing some firmware feature that you need, then you can create your own firmware (probably by modifying mine), and download that to your ME1702/A.

The ME1702/A firmware is divided into two components: the ME Loader Kernel (the Loader, which cannot be downloaded via the serial port) and the ME1702/A Programming Firmware (the Programming Firmware). The primary function of the Loader is to load new Programming Firmware via the serial port, eliminating the need to use a PIC programming device, such as the PICkit 3.

This section describes how to load new Programming Firmware. It also provides some details about how it all works, so that you could modify the firmware - assuming that you are versed in PIC assembly language.

8.1 Firmware Download Instructions

Connect your ME1702/A to the serial port of your computer (or a serial port dongle on the USB port of your computer) and start a terminal program, such as Hyperterm. Set up the terminal program this way: 9600 baud, 8 data bits, no parity, XOFF/XON handshaking enabled.

To enter the Loader, type capital L a few times immediately after you turn on the ME1702/A, or immediately after issuing a Reset command. You will see the Loader message, instead of the ME1702/A sign-on banner:

```
ME Loader 1.0
```

When you see this message, the ME1702/A is ready to receive a Programming Firmware file in Intel Hex format. The Loader expects to see an Intel Hex file exactly like that produced by Microchip’s MPASM assembler.

The PIC microcontroller has insufficient RAM to load and validate the entire Programming Firmware before writing it to FLASH. Instead, the Loader writes to FLASH whenever it gets a complete 32-word ‘row’ of data (where a word is 2 bytes in the hex file). This means that a failed Programming Firmware load will probably corrupt the Programming Firmware in FLASH memory. (However, the Loader itself is hardware-protected against being overwritten.)

Before you send the hex file, make sure your terminal program has XOFF/XON handshaking enabled. Handshaking is required so that the Loader can pause the file transmission from time to time, to write the received data into FLASH. Otherwise, data will be lost, and the Programming Firmware will be corrupted.

Once your terminal program is set up correctly, send the hex file (typically, me1702a.hex) to the ME1702/A. It will take a few minutes to download, and you should see the hex file as it downloads. If there are any errors, they will be flagged with a brief error message:

Message	Meaning
?Csm	Checksum error in Intel Hex record
?Hex	Illegal (non-hex) character in a hex record (Only ‘0’-‘9’ and ‘A’-‘F’ are allowed)
?Ver	Flash read-back verify error

When the firmware load is complete, the Loader will print the total number of Intel Hex records loaded, as well as the number of errors detected. If the error count is anything but 0000, the firmware load failed, and the loaded firmware is most likely corrupted.

If you got your new firmware file from me, then I will have included a comment that tells you how many records should have been loaded, which will be printed in your terminal program's window at the end of the hex load. If the reported number of loaded records does not match the number that I provided, then the firmware load was probably not successful, and the loaded firmware is most likely corrupted.

If the load is successful, you can jump to the new Programming Firmware by typing the ESC key several times, or you can power-cycle the ME1702/A.

If the load fails, you can try again immediately after the failed load, when Loader message is printed. Or, you can type capital L immediately after powering on the ME1702/A to invoke the Loader to try loading again.

8.2 Intel Hex File Format for Firmware Downloads

This section specifies the format of Intel Hex files that are accepted by the Loader, as well as the error messages that are printed by the Loader. This Intel hex format is exactly the format produced by Microchip's MPASM assembler. Note that this specification is slightly different than Intel Hex files accepted by the Programming Firmware.

An Intel Hex record is defined as follows:

:NNAAaaTTDDDDDD...DDDDCC

- A colon marks the beginning of an Intel Hex record. All characters are ignored until a colon has been received. This means that comment lines in the Intel Hex file (that contain no colons) will be ignored. This also means that any record where the initial colon has been corrupted will be ignored without being caught as an error.
- NN defines the number of Data bytes in the record.
- AAaa is the address field of the record. AA is the most significant address byte; aa is the least significant address byte.
- TT is the record type.
- DD is a data byte. Data bytes belong in memory at sequential addresses, starting at AAaa. The record should have NN data bytes.
- CC is the checksum of the record. The low byte of the sum of NN, AA, aa, TT, all the DDs, and CC should be 00.
- A carriage return (CR), a line feed (LF), or both, is optional.

Three Intel Hex record types are accepted; all other records are ignored.

1. **Type 00 records** are data records. Data records are written to FLASH only if a Type 04 record has already been received, with extended address = 0000. If no Type 04 record has been received yet, or if the last Type 04 record set the extended address to something other than 0000, then the data record will be ignored. This means (for example) that an Intel Hex file cannot write to the Config registers of the PIC.
2. **A Type 01 record** (with 0 bytes of data) is an End-Of-File record, and is required at the end of the file to force a write to FLASH of the last RAM buffer full of data. NOTE: a data record (Type 00) with 0 bytes of data is NOT treated as an End-of-file record.
3. **Type 04 records** set the extended address for the subsequent records. The "address" field of the Type 04 record (bytes 2 and 3) is ignored. The first 2 bytes of the "data" field (bytes 5 and 6) set the extended address. MPASM sets the extended address to 0000 for FLASH data, and 0001 for the PIC Config registers.

The PIC's FLASH memory is organized in 32-word ‘rows’, where a word is 14 bits wide. Microchip’s MPASM assembler splits each word into two sequential bytes in the hex file, with the low 8 bits in the first byte and the high 6 bits in the second byte. This means that the address in each Intel Hex record is the FLASH address times 2.

The Loader assumes that each Intel Hex record fits completely within one 32-word (64-byte) FLASH row. However, one FLASH row may be (and probably will be) made up of several hex records. (In other words, no record is allowed to have data that is split between two 32-word FLASH rows.) Hex files generated by MPASM meet this requirement.

If the hex file has data for only part of a FLASH row, then only the data supplied in the Hex file will be changed - the other bytes in that FLASH row will remain unchanged, because the Loader first reads the old FLASH data from each row into its RAM buffer, filling in the missing data for that row.

After each row is written, the Loader verifies the write by reading it back and comparing it to the row data in its RAM buffer.

The Loader checks the checksum for all records, including ignored records.

The following conditions will generate an error message, and increment error count that is reported after the end of the file:

1. **Checksum Error** (The checksum of the record was incorrect.)
2. **Bad Hex Error** (something besides ‘0’-‘9’ or ‘A’-‘F’ when expecting hex)
3. **Verify Error** (FLASH read-back did not match the RAM buffer data)

The Loader actually writes to FLASH when a new hex record addresses FLASH memory in a different FLASH row than the previous record, or when a type 01 record is received. This means that FLASH write-verification occurs after the address and record-type fields of the next hex data record have been received, and before its data bytes are received. And this (in turn) means that if a verify fails, then the verify error message will be printed in the middle of the next hex record, and will refer to the previous record(s).

Erasing and writing one FLASH row takes significantly longer than a character time at 9600 baud, and the FLASH write cannot be interrupted. For this reason, your terminal program must respect XOFF/XON handshaking: the Loader will issue an XOFF prior to programming a FLASH row, and then will wait for your terminal program to respond to the XOFF, accepting up to about 127 more characters after sending the XOFF. When the Loader receives no characters for 3 mS (3 character times at 9600 baud) after sending an XOFF, it assumes that your terminal program has paused transmission, and will program the FLASH row. The Loader will send an XON when the FLASH row programming is complete. The Loader will not receive any characters from its serial port while it is busy programming FLASH - such characters will be lost.

The hex file must end with a type 01 (End-Of-File) record. Receipt of a type 01 record causes the following actions:

1. The total number of records that were actually loaded into the FLASH is printed. (Type 01 and type 04 records, as well as any type 00 records that were not written to FLASH do not count.) This can be checked manually, to make sure no records were dropped, and the load was in fact successful.
2. The total number of errors detected is printed. Anything other than 0000 means that the load was unsuccessful, and the loaded code should not be trusted.

3. Control returns to the Loader, reprinting the Loader's brief sign-on message. (To run the newly loaded code, hit the ESC key 3 times in a row)

8.3 The ME Loader Kernel

This section gives some details of the ME Loader Kernel, interesting only of you intend to create or modify the Programming Firmware.

If you are considering modifying the Programming Firmware, refer to Microchip document number DS41452B, “PIC16(L)F1516/7/8/9 Data Sheet.” Also, get a copy of Microchip’s MPLAB IDE, which includes their MPASM assembler. (At the time of this document, the data sheet and MPLAB IDE are available for free at www.microchip.com.)

The ME Loader Kernel resides in PIC FLASH memory below address 0x0200, an area that is hardware-protected (via the PIC CONFIG registers) against writes by firmware. The ME Loader Kernel executes first after reset, and transfers control to the Programming Firmware only after giving the user an opportunity to invoke the Loader (by typing some ‘L’s), instead of executing the Programming Firmware. Thus, if a firmware load goes badly and the Programming Firmware becomes corrupted, the Loader will still be there, and will allow you to re-load the Programming Firmware.

The Loader manages the UART transmit and receive interrupts, and also the transmit and receive queues, whose memory locations are given below. The best way to access the queues is through the Loader’s call vectors. However, you can also directly access the queues by using their pointers, as follows.

The pointers are the low byte of the address of the next queue slot. The high byte for these queue pointers are fixed, as defined below. The two UART queues are circular. If incrementing a queue pointer causes it to point beyond the last queue address, then it should be reset to the first queue address. Note that global interrupts should be masked while manipulating the queue pointers.

For the transmit queue, new data should be entered where TQ_IPTR points, before the the pointer is advanced. However, if TQ_IPTR is equal to TQ_OPTR, and the TQ_EMPTY bit is cleared, then the queue is full, and firmware should wait for this state to end before enqueuing a character. The TQ_EMPTY bit should always be cleared upon enqueuing a character. Unless the XOFF_STATE bit is set, the transmit interrupt should be enabled, once enqueuing is completed. Note that global interrupts should be masked around this manipulation.

For the receive queue, the next available character is where RQ_OPTR points, and the pointer should be advanced after reading the available character. However, if RQ_OPTR is equal to RQ_IPTR and the RQ_FULL bit is cleared, then the receive queue is empty.

The receive interrupt code translates any CR-LF sequence, any LF-CR sequence, and any stand-alone LF, into a simple CR. The transmit interrupt translates every CR into a CR-LF sequence. These translations make the ME1702/A agnostic about how lines are terminated: CR-LF (e.g. CP/M, MS-DOS, Microsoft Windows), LF-CR (e.g. Acorn BBC), plain LF (e.g. Unix, Mac OS-X, BeOS), or plain CR (e.g. Apple][through Mac OS-9, TRS-80, and Commodore) are all acceptable.

The ME Loader Kernel provides several functions to the Programming Firmware, accessed via fixed-location call-vectors. The Programmer Firmware can call these functions, and they will return when completed. The following table defines all of the functions provided by the ME Loader Kernel. The subsequent Common Memory table defines the registers, R0 and R1. All of the other registers referenced in this table are defined in the PIC Data Sheet.

Address	Call Vector	Function
0x0002	K_KERN_REV	**Get Loader Kernel Revision Level** On Return: W bits [7:4] = major revision level W bits [3:0] = minor revision level
0x0003	K_PROG_ID	**Get Programmer ID** On Return: W=00 for ME1702/A, W=01 for ME5204
0x0005	K_CONIN	**Get one character from Receive Queue** On Return: New character is in W & R0, Z is cleared If Rx queue is empty: W = R0 = 00, Z is set Trashes FSR0
0x0006	K_CONOUT	**Print one character** (via Tx queue) On Call: W = character to send On Return: R0 = sent character, unlees it was CR If character was CR then R0 = LF Trashes W, FSR0, BSR
0x0007	K_PRINTF	**Print null-terminated string** (via Tx queue) On Call: BSR must be set to 0x03 String address = PMADRH, PMADRL String data b packed via MPASM ‘da’ directive String is terminated with 14-bit word = 0x0000 On Return: Trashes W,R0,FSR0,BSR,PMDATL,PMDATH,PMADRL,PMADRH
0x0008	K_PRINTHEX1	**Print binary value as 1 ASCII hex digit** (via Tx queue) On Call: W bits 3:0 = 4-bit binary value to send W bits 7:4 don’t matter On Return: Trashes W, R0, FSR0, BSR
0x0009	K_PRINTHEX2	**Print binary value as 2 ASCII hex digits** (via Tx queue) On Call: W = 8-bit binary value On Return: Trashes W, R0, R1, FSR0, BSR
0x000A	K_HEX2BIN	**Convert ASCII hex value to binary, and combine result with previous nibble** On Call: R0 = ASCII hex value {“0”-“9” or “A”-“F”} R1 bits 3:0 = 4-bit previous binary value R1 bits 7:4 don’t matter On Return: C set if R0 <> “0”-“9” or “A”-“F” R0 = new binary nibble (trash if C is set) W = R1 = R1 << 4 + R0 (both trash if C set)
0x000B	K_STALL25M	**Stall for W * 25 mSec** (Clears W, R0 & R1)
0x000C	K_STALL250U	**Stall for W * 250 uSec** (Clears W, R0)
0x000D	K_STALL1U	**Stall for W + 1 uSec** (Clears W)
0x0200	LOADED_CODE	Execution address for Programming Firmware

The PIC's **Common RAM** locations are assigned as follows:

Address	Name	Function		
0x70	R0	General purpose register		
0x71	R1	General purpose register		
0x72-0x7A	available	Available for use by Programming Firmware		
0x7B	RQ_IPTR	Receive queue in-pointer (Note 1)		
0x7C	RQ_OPTR	Receive queue out-pointer. This is the address low byte of the next character available in the queue. The high address byte is high(RX_QUEUE).		
0x7D	TQ_IPTR	Transmit queue in-pointer This is the address low byte of the next empty slot in the queue. The high address byte is high(TX_QUEUE).		
0x7E	TQ_OPTR	Transmit queue out-pointer (Note 1)		
0x7F	INT_FLAGS	Flags used by the Loader Kernel:		
		Bit	**Name**	**Meaning when Set**
		0	XOFF_STATE	Transmitter is stopped due to received XOFF (Note 1)
		1	TQ_EMPTY	The transmit queue is empty (Note 2)
		2	PREV_CR	The previous chr was CR (Note 1)
		3	PREV_LF	The previous chr was LF (Note 1)
		4	RQ_FULL	The receive queue is full (Note 2)
		5-7	reserved	For future use by the Loader

Notes:

1. This variable is reserved for use by the interrupt service routines, and should not be written by the Programming Firmware
2. This bit is set by the interrupt service routine and cleared by the Programming Firmware.

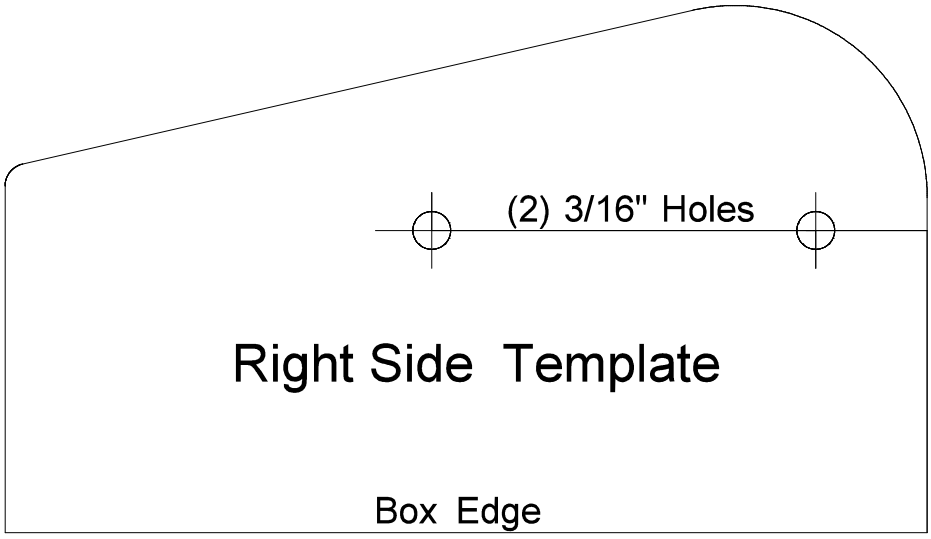
The PIC's **Linear RAM** is assigned as follows:

Address	Name	Function
0x2000	RAM_BUF	Available for 256-byte EPROM image buffer.
0x2100	RX_QUEUE	UART receive circular queue (128 bytes)
0x2180	TX_QUEUE	UART transmit circular queue (16 bytes)
0x2190-0x21EF	Available	Available for use by Programming Firmware

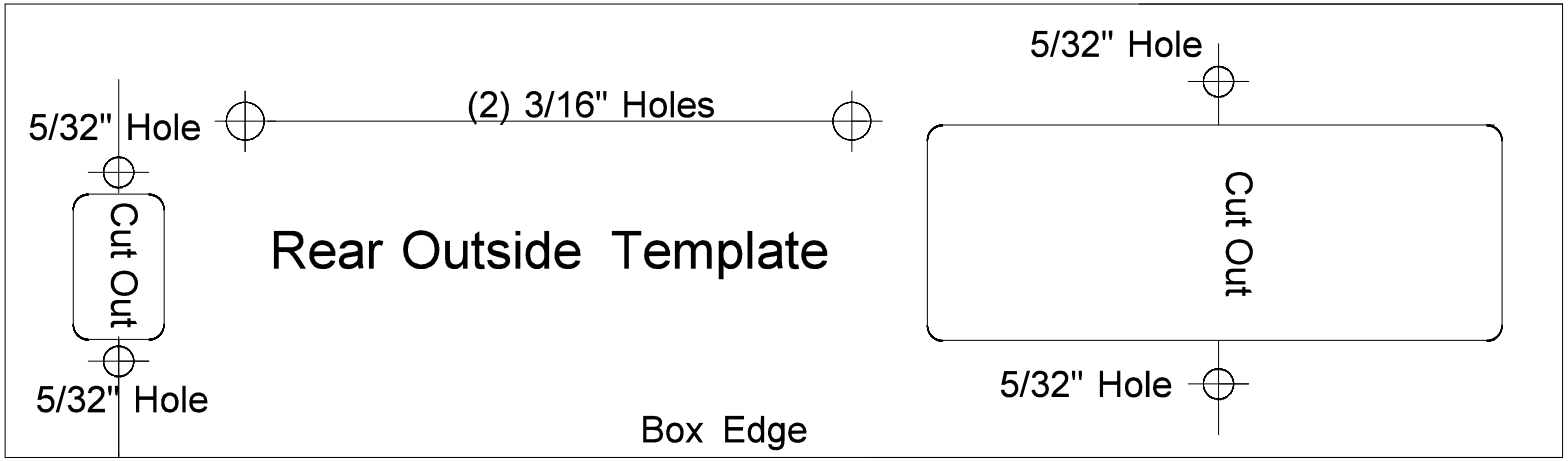
Section 9. Drawings

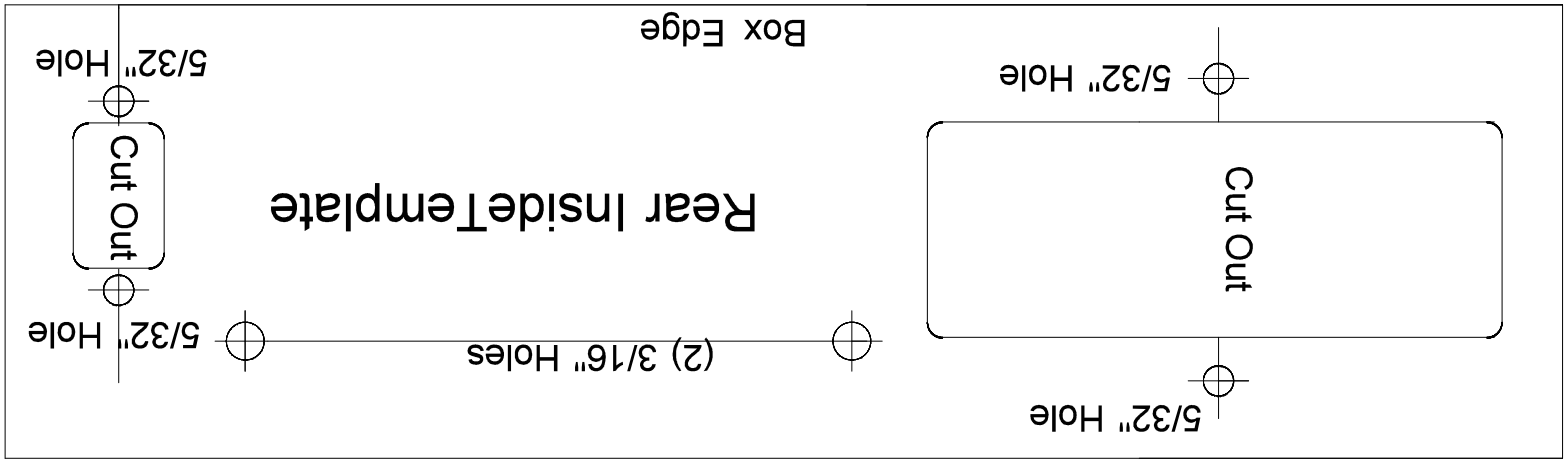
9.1 Enclosure Templates

The following drawings should be photocopied, and then used as templates for drilling and cutting the Enclosure’s plastic box. See **Section 1.2 Enclosure Fabrication**.

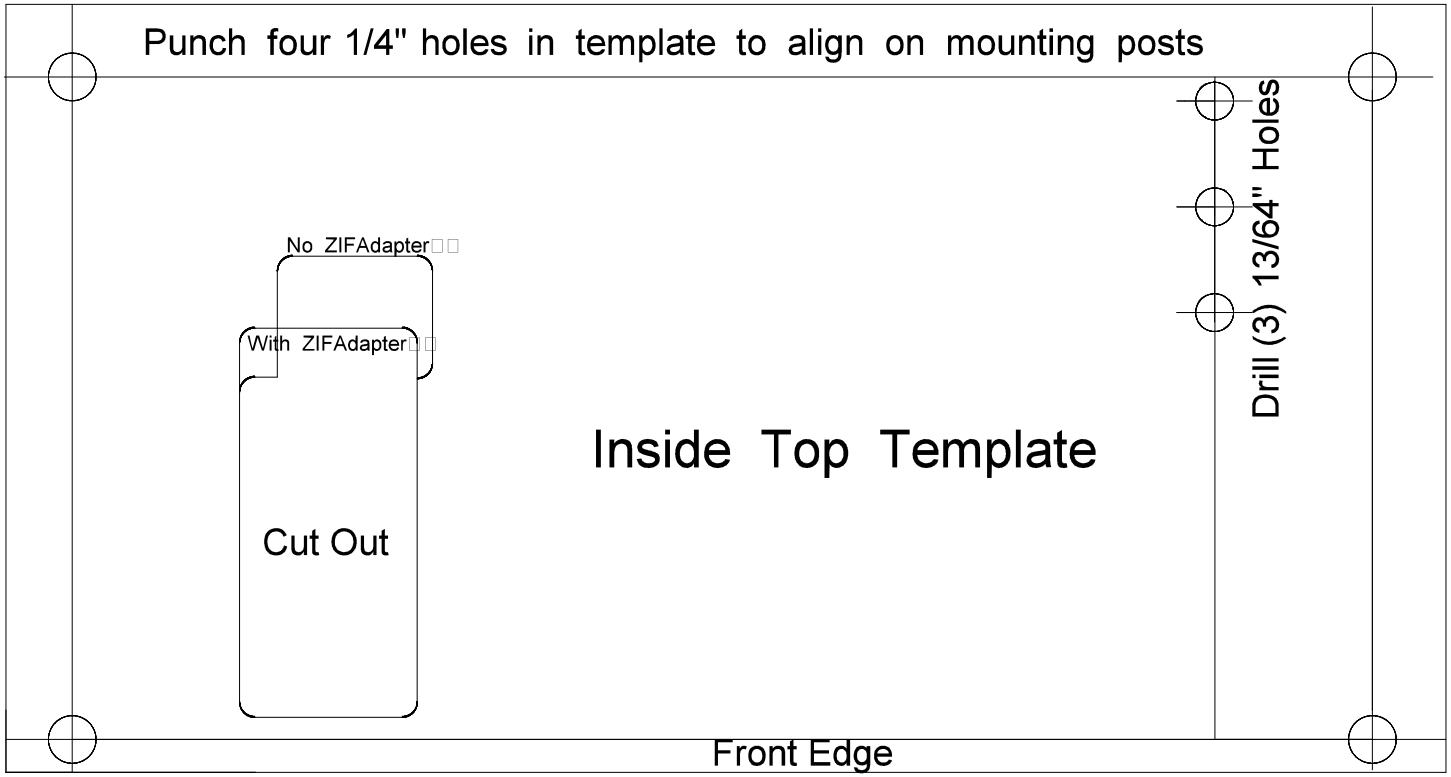

(2) 3/16" Holes
Right Side Template
Box Edge

Intentionally Blank


5/32" Hole
(2) 3/16" Holes
5/32" Hole
Cut Out
Rear Outside Template
Cut Out
5/32" Hole
5/32" Hole
Box Edge


Box Edge
5/32" Hole
Cut Out
Rear InsideTemplate
Cut Out
5/32" Hole
(2) 3/16" Holes
5/32" Hole
5/32" Hole

Intentionally Blank


Punch four 1/4" holes in template to align on mounting posts
Drill (3) 13/64" Holes
No ZIFAdapter
With ZIFAdapter
Cut Out
Inside Top Template
Front Edge

Intentionally Blank

9.2 Bill of Materials

The following three pages are the complete Bill of Materials for the ME1702/A. The first table is the printed circuit board (everything except the printed circuit board assembly), and the second table is the chassis.

Note that variable resistors VR1 and VR2 require different components depending on the PC board revision. No other components change between revisions.

The Digikey and Allied Electronics part numbers are more or less current at the time this was written.

Note that a few components have alternate components listed, with quantity 0. You can substitute the alternate component with no significant effect.

If you choose to substitute any of the other components, be sure that you choose suitable substitutions:

- 1% resistors are used when they control output voltages - so do not substitute with lower-tolerance components.
- When substituting capacitors, be careful about operating voltages. The ME1702/A has higher voltages than typical digital circuits, and using a capacitor that is not rated for a high enough voltage will cause an early failure.
- When substituting transistors, you need to look at Vce max, hfe, Ic max, Vce at saturation, and pinout. Some of the transistors (Q34 and Q35) need to tolerate at least 100V Vce. Most others must tolerate at least 50V Vce. Some (e.g. Q1) must pass 500 mA Ic with relatively low (less than 0.5V) voltage drop.
- Zener diode voltage, current ratings, and zener voltage tolerances must be no worse than those for the specified zener diodes.
- The other diodes were chosen for their forward voltage drop (e.g. D1), or reverse breakdown voltage (e.g. D4, D5, D7 and D8). Substitutions must be at least as good as those for the specified components.
- If you replace either transformer with one that does not support both 120V and 240V inputs, then you should disable the appropriate circuit on the Multifunction Inlet Subassembly.
- T2 has a higher output voltage than needed - you could replace this with an 18V center-tapped transformer. This transformer should be rated for at least 50 mA output.
- Similarly, T1 could be replaced with a 110-volt center-tapped transformer, capable of at least 400 mA output.

PC Board Bill of Materials

(Not including the bare PC board and the microcontroller)

Qty	Component	Value	locations	Digikey Part
1	Resistor, 1/4W, 1%	12 1%	R3	S12CACT-ND
3	Resistor, 1/4W, 1%	10K 1%	R48,R78,R79	10.0KXBK-ND
3	Resistor, 1/4W, 1%	R 154K 1%	R47,R77,R80	154KXBK-ND
2	Resistor, 1/4W, 1%	R 324 1%	R1 R38	324XBK-ND
2	Resistor, 1/4W, 1%	R 976 1%	R2 R40	976XBK-ND
1	Resistor, 1/4W	R 13K	R19	13KQBK-ND
1	Resistor, 1/4W	220	R4	220QBK-ND
2	Resistor, 1/4W	330	R55,R59	330QBK-ND
13	Resistor, 1/4W	10K	R37,R45,R46R50,R54,R57,R61, R75,R76,R81-R84	10KQBK-ND
10	Resistor, 1/4W	100	R12,R13,R21,R23,R25,R27,R29, R31,R33,R35	100QBK-ND
34	Resistor, 1/4W	1K	R14,R15,R18,R39,R41-R44,R49, R51-R53, R56,R58,R60, R62-R67, R71-R74,R85-R93	1.00KXBK-ND
6	Diode	1N4004	D4-D9	641-1311-1-ND
2	Diode	1N4148	D2,D3	1N4148TACT-ND
1	Diode, Schottky	1N5817	D1	1N5817-TPCT-ND
1	Diode, Zener, 1/2W, 36V, 2%	NZX36B,133	Z1	568-5902-1-ND
1	Diode, Zener, 1W, 15V	1N4744A	Z2	1N4744A-ND
1	Diode, Zener, 1W, 56V	1N4758	Z3	1N4758ADICT-ND
20	Resistor, 1/2W	6.8K	R5-R11,R16,R17,R22,R24,R26, R28,R30,R32, R34,R36,R68-R70	6.8KH-ND
2	Trim Pot, 2K **(Rev F board)**	2K	VR1, VR2	3362P-202LF-ND
0	Trim Pot, 2K **(Rev C - E boards)**	2K	VR1, VR2	K4A23-ND
1	Socket, 16-pin, low profile		U3	A100206-ND
1	Socket, 40-pin, low profile		U2	3M5471-ND
7	Capacitor, chip, 100V	0.1 uF	C1-C5,C7,C8	399-4330-ND
6	Capacitor, chip, 25V	1 uF	C10,C12-C16	445-2857-ND
2	Resistor, 2W	10K	R94,R95	OY103KE-ND
1	Voltage Regulator,TO-220, -9V	7909	V1	NJM7909FA-ND
2	Voltage Regulator, TO92, adj.	LM317L	V2,V3	LM317LZ-ND
1	Transistor, PNP	2N3906	Q12	2N3906FS-ND
1	Transistor, NPN	ZTX451	Q1	ZTX451-ND
33	Transistor, NPN	MPSA42	Q2-Q11,Q13-Q33,Q35,Q36	MPSA42FS-ND
2	Fuse Clip		FS1 (both ends)	F4186-ND
0	Fuse Clip		FS1 (both ends) (Alternate)	BK-6005-ND
2	Capacitor, electrolytic	330 uF 35V	C6,C9	732-8742-1-ND
1	Capacitor, electrolytic	330 uF 180V	C11	338-1603-ND

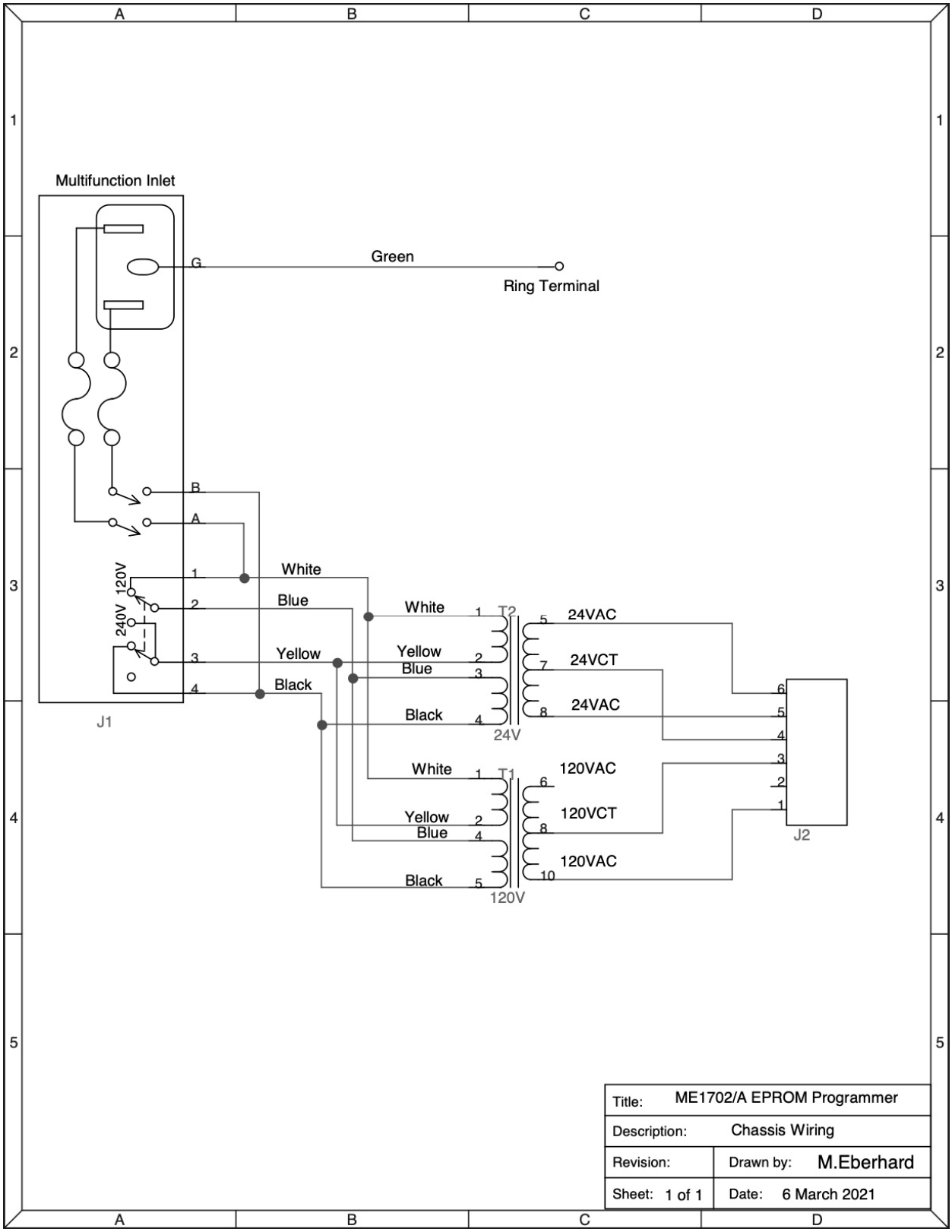
Qty	Component	Value	locations	Digikey Part
2	Heatsink, TO-220, vertical	Heatsink	V4,V5	HS368-ND
2	Screw, 4-40,3/8", pan head	Screw	V4, V5	H781-ND
2	Nut, 4-40	Nut	V4, V5	H216-ND
1	Voltage Regulator, TO220, adj.	LM317	V4	LM317TFS-ND
1	Voltage Regulator, TO-220, 5V	7805	V5	MC7805CTGOS-ND
1	Heatsink, TO-247, vertical	Heatsink	Q34	WA-T247-101E-ND
1	Transistor, NPN, power	2SC4468	Q34	2SC4468-ND
0	Transistor, NPN, power	FJAF4310	Q34 (Alternate)	FJAF4310YTU-ND
0	Transistor, NPN, power	2STC4468	Q34 (Alternate)	1026-2STC4468-CHP
1	Header, 0.1", 4 pin		J3	A19431-ND
1	Header, 0.1", 6 pin		J2	A31116-ND
1	Header, 0.156", 6 pin		J1	WM4624-ND
1	Socket, 24 pin, ZIF		U1	3M2402-ND
3	Spacer, T1-3/4 LED, 0.12"	LED spacer	LED1-LED3	8311K-ND
1	LED, blue		LED1	C503B-BCS-CV0Z0461-ND
1	LED, green		LED2	754-1265-ND
1	LED, red		LED3	754-1266-ND
1	IC, RS232 tranceiver	MAX232	U3	296-1402-5-ND
2	Fuse, 1-1/4", 3/4A, fast, 20mm x 5mm	3/4A fuse	FS1	FSC-750MA

Chassis Bill of Materials

Note: The cabinet can be ordered from Allied Electronics.

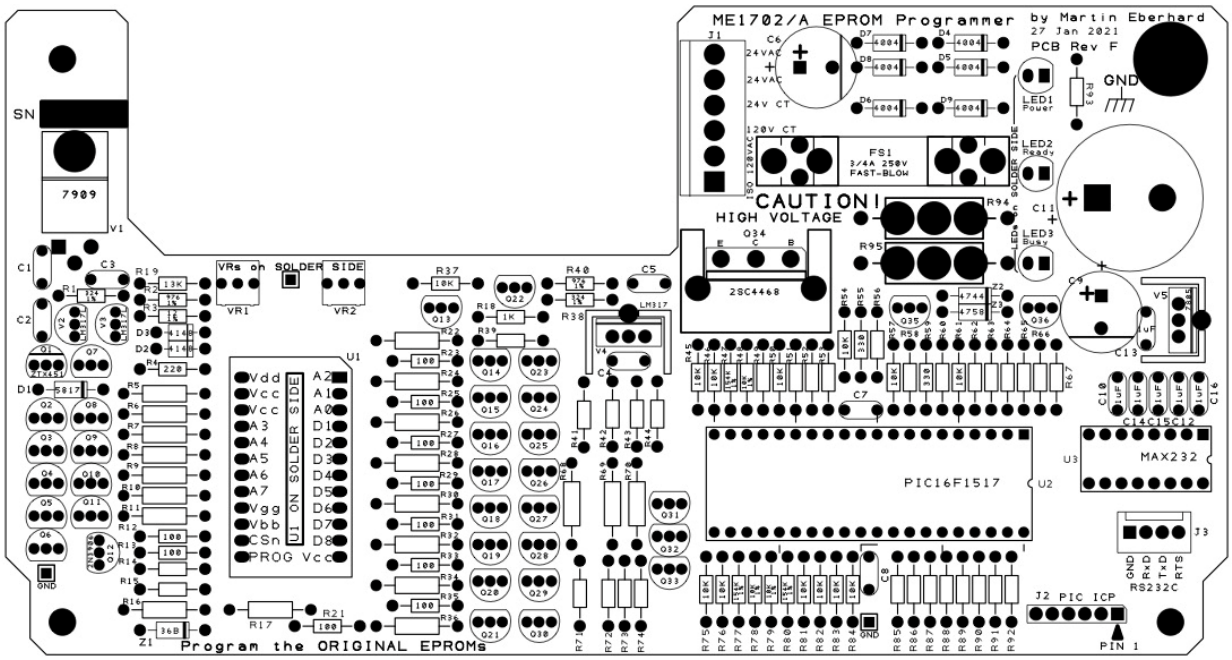
Qty	Part	Manufacturer	Mfg part.	Digikey Part
1	0.25A 24V center-tapped Transformer, 120V/240V input	Hammond	186B24	HM2129-ND
1	0.5A 120V center-tapped Transformer, 120V/240V input	Hammond	186E120	HM4702-ND
1	slope-top cabinet	Hammond	1595EB	Allied: 70166925
1	Multifunction Inlet (Power Entry Module)	Delta Electronics	10C3	1144-1003-ND
1	Connector housing, 6-position .156	AMP	09-50-3061	WM2104-ND
6	Connector Contact	AMP	08-52-0072	WM2302-ND
1	Connector housing, 4 position, .10	Molex	22-01-3047	WM2002-ND
3	Connector Contact	Molex	08-50-0113	WM1114-ND
1	Connector, 9-position, female, D-sub	3M	8R09-N001	3M10608-ND
1	Ring Terminal, #6, star	TE Connectivity	F1759-ND	A29900CT-ND
2	Fuse, 1-1/4", 3/4A, fast			283-2631-ND
4	Screw, 6-32, 3/8", Pan Head			
2	Flat washer, #6			
4	Lock Washer, #6			
4	Nut, 6-32			
2	Screw, 4-40, 3/8", flat head			
2	Screw, 40-40, 3/8", pan head			
4	Nut, 4-40			
4	3/8" self-tapping 4-40 screws			
4	Flat washer, #4			5205820-3-ND
4	0.3" high adhesive rubber foot	3M	SJ-5523	SJ5523-0-ND
1	IEC Line Cord			993-1039-ND

9.3 Chassis Wiring Diagram

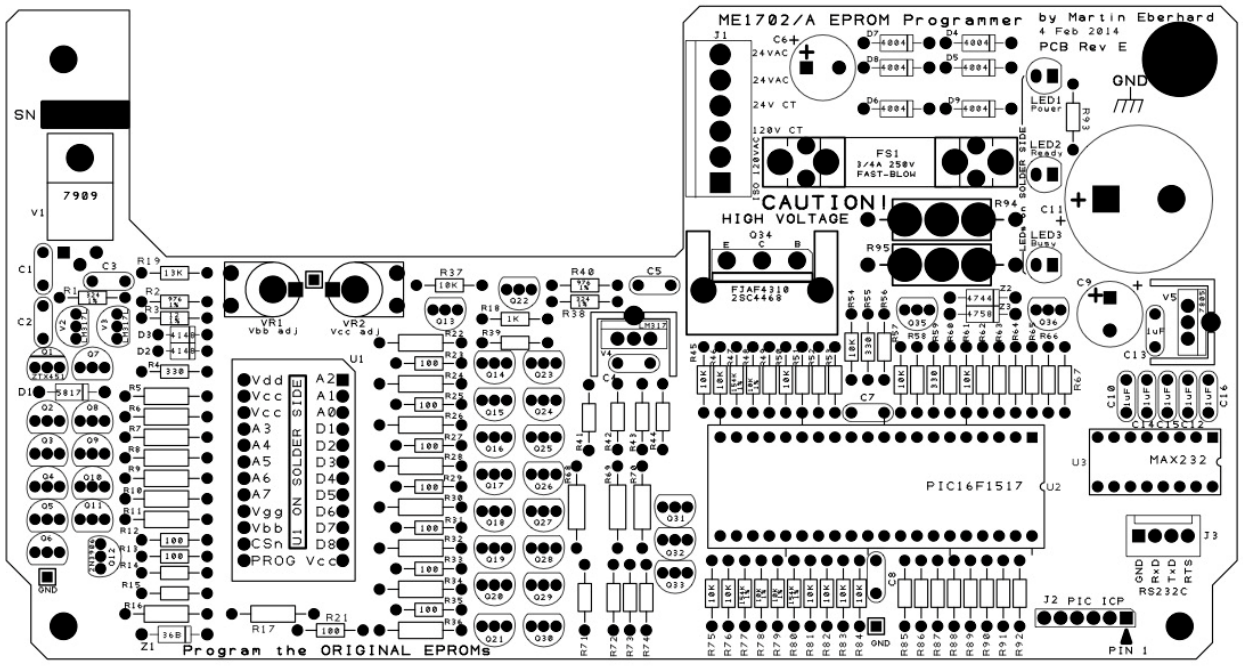


9.4 PCBA Component Placement

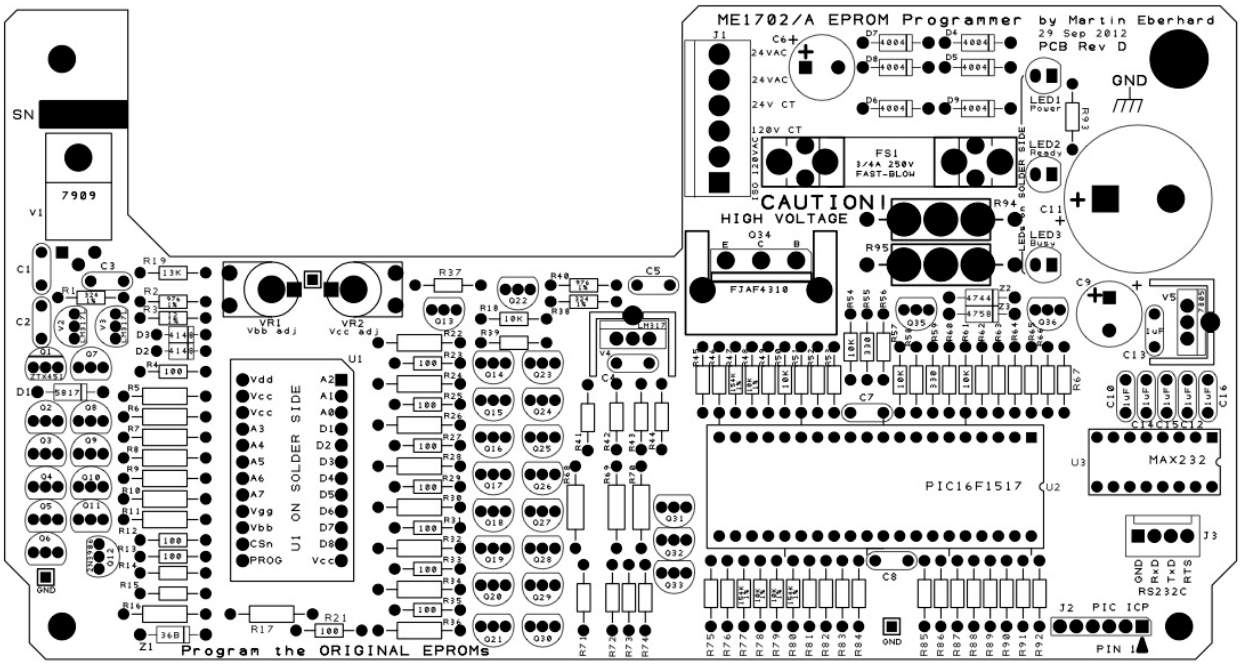
The following pages show the printed circuit board outline and silkscreen layer for revs Bthrough F PC boards. These can help you find components on the printed circuit board assembly.



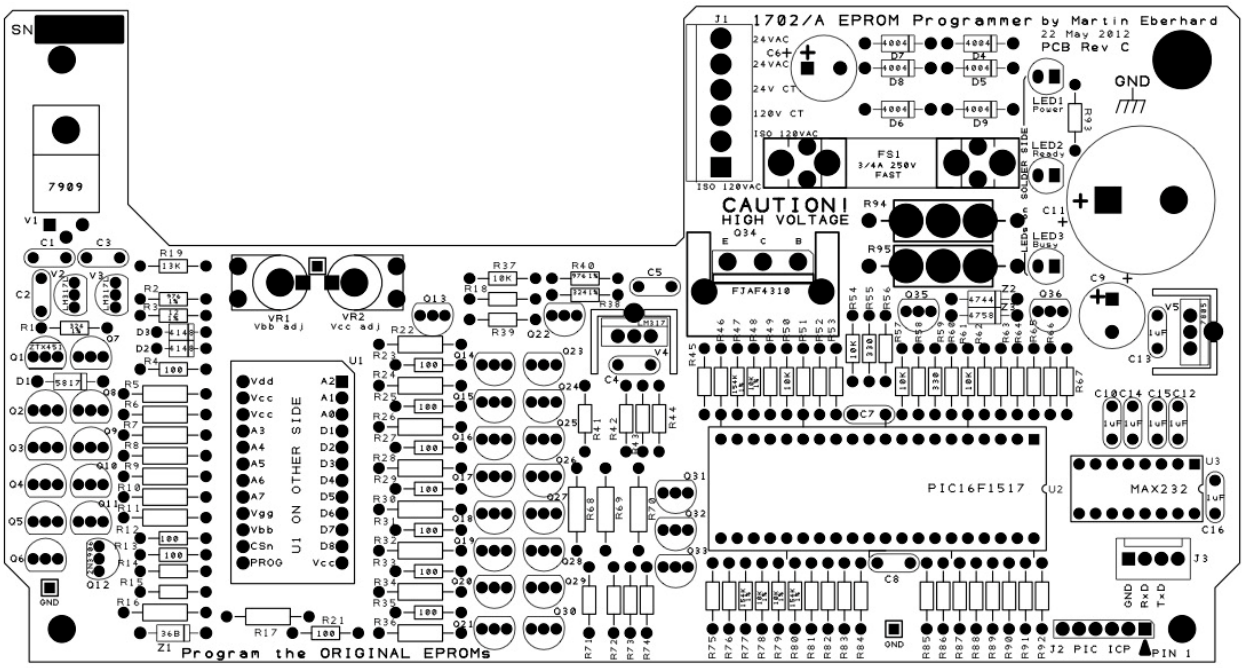
ME1702/A Rev F Component Placement



ME1702/A Rev E Component Placement



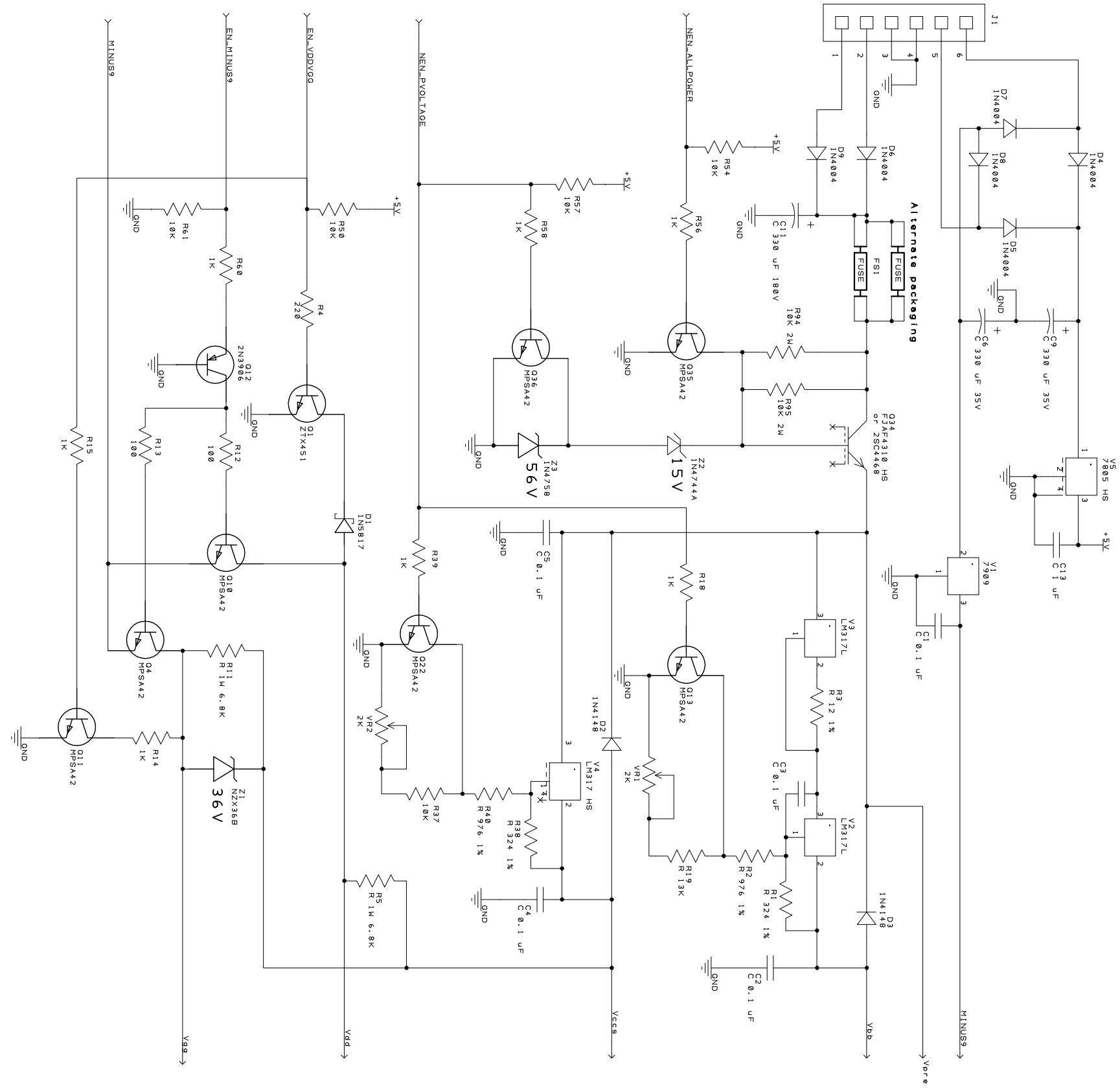
ME1702/A Rev D Component Placement



ME1702/A Rev C Component Placement

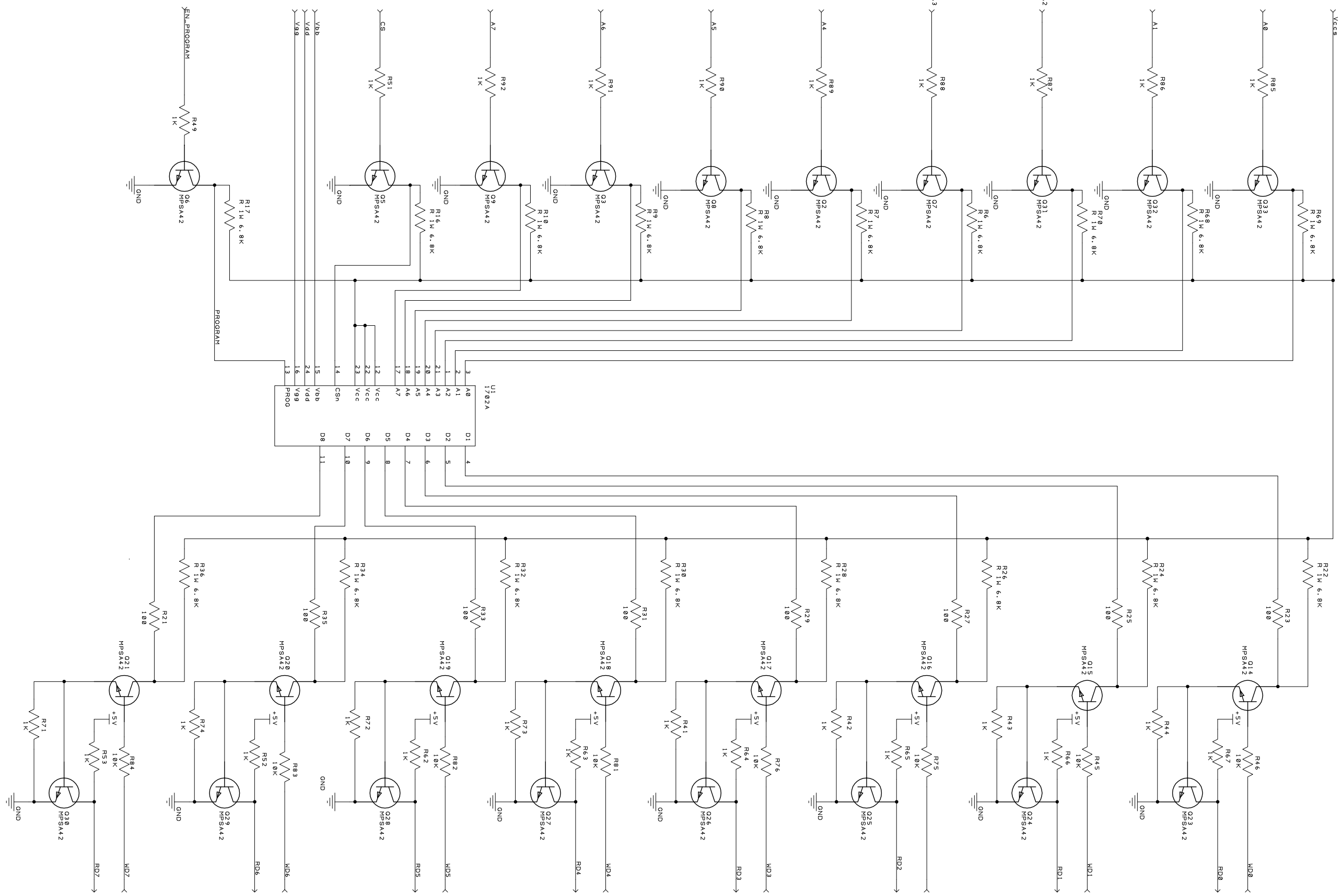
9.5 PCBA Schematics

The following three pages are the schematics for the ME1702/A Programmer’s printed circuit board assembly. The Rev D and E and F PC board schematics are the same as the Rev C schematic, except that a previously-unused gate in the RS-232 driver chip (U3) is used to provide an always-true handshake signal on J3 pin 4. (This is useful when connecting to a terminal that requires a true signal on its handshake inputs.)



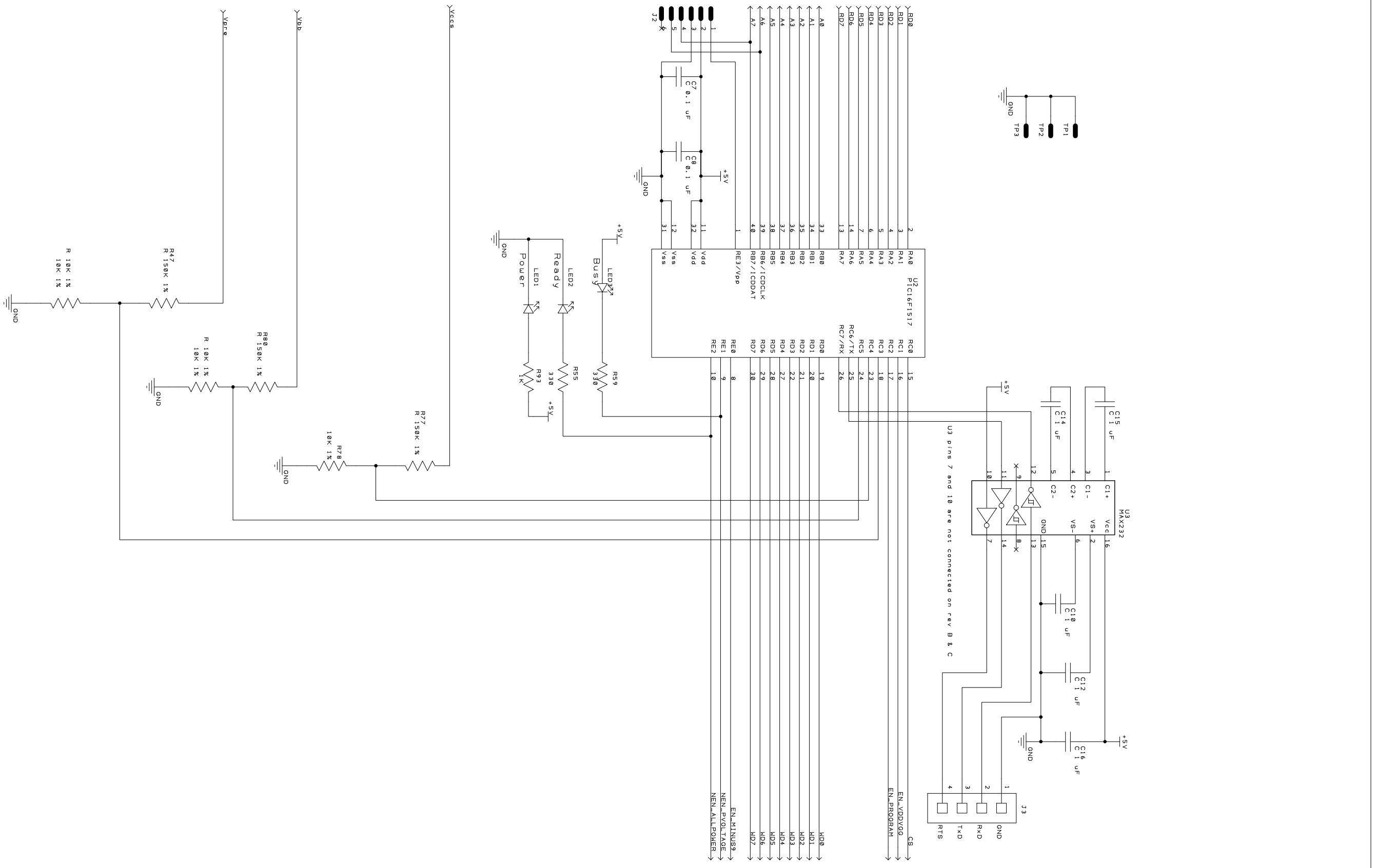
Project: 1702/A EPROM Programmer PCBA	Engineer: Martin Eberhard	
Sheet Title: Programmable Power Supply	Filename:	
Revision: B through F	Date: 27 January 2021	Sheet: 1 of 3

Martin Eberhard



Project: 1702/A EPROM Programmer PCBA	Engineer: Martin Eberhard	
Sheet Title: EPROM Interface	Filename:	
Revision: B through F	Date: 27 January 2021	Sheet: 2 of 3

Martin Eberhard



Project: 1702/A EPROM Programmer PCBA	Engineer: Martin Eberhard	
Sheet Title: Microcontroller	Filename:	
Revision: B through F	Date: 27 January 2021	Sheet: 3 of 3

Martin Eberhard

9.6 1702A EPROM Specification

The following pages are Nation Semiconductor’s MM1702A MOS EPROM specification. It is typical of specifications from the other manufacturers of the 1702A.

National Semiconductor

MOS EPROMs

MM1702A 2048-Bit (256 x 8) UV Erasable PROM

General Description

The MM1702A is a 256 word by 8-bit electrically programmable ROM ideally suited for uses where fast turn-around and pattern experimentation are important. The MM1702A undergoes complete programming and functional testing on each bit position prior to shipment, thus insuring 100% programmability.

The MM1702AQ is packaged in a 24-pin dual-in-line package with a transparent lid. The transparent lid allows the user to expose the chip to ultraviolet light to erase the bit pattern. A new pattern can then be written into the device. The MM1702AD is packaged in a 24-pin dual-in-line package with a metal lid and is not erasable.

The circuitry of the MM1702A is entirely static; no clocks are required.

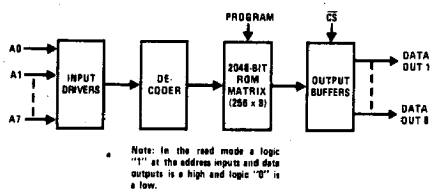
A pin-for-pin metal mask programmed ROM, the MM1302 is ideal for large volume production runs of systems initially using the MM1702A.

The MM1702A is fabricated with silicon gate technology. This low threshold technology allows the design and production of higher performance MOS circuits and provides a higher functional density on a monolithic chip than conventional MOS technologies.

Features

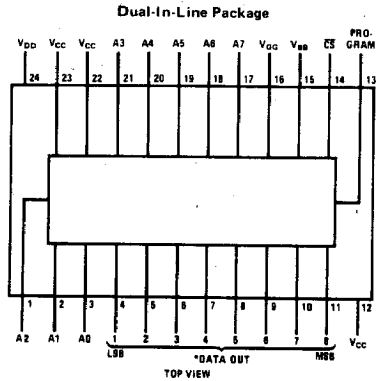
- Fast programming—30 seconds for all 2048 bits
- All 2048 bits guaranteed programmable—100% factory tested
- Fully decoded, 256 x 8 organization
- Static MOS—no clocks required
- Inputs and outputs DTL and TTL compatible
- TRI-STATE® output—OR-tie capability
- Simple memory expansion—chip select input lead
- Direct replacement for the Intel 1702A

Block and Connection Diagrams



Note: In the read mode a logic "1" at the address inputs and data outputs is a high and logic "0" is a low.

Dual-In-Line Package



*This pin is the data input lead during programming.

Order Number MM1702AQ
See NS Package J24CQ

Pin Names

A0—A7	Address Inputs
$\overline{CS}$	Chip Select Input
$D_{OUT\,1} - D_{OUT\,8}$	Data Outputs

Pin Connections*

MODE/PIN	12 (V_{CC})	13 (PROGRAM)	14 ($\overline{CS}$)	15 (V_{BB})	16 (V_{GG})	22 (V_{CC})	23 (V_{CC})
Read	V_{CC}	V_{CC}	GND	V_{CC}	V_{GG}	V_{CC}	V_{CC}
Programming	GND	Program Pulse	GND	V_{BB}	Pulsed V_{GG} (V_{IL4P})	GND	GND

*The external lead connections to the MM1702A differ, depending on whether the device is being programmed or used in read mode. (See following table.) In the programming mode, the data inputs are pins 4—11 respectively.

Absolute Maximum Ratings (Note 1)

Storage Temperature	−65°C to +125°C
Power Dissipation	2W
Read Operation	
Input Voltages and Supply Voltages with Respect to V_{CC}	+0.5V to −20V
Program Operation	
Input Voltages and Supply Voltages with Respect to V_{CC}	−48V
Lead Temperature (Soldering, 10 seconds)	300°C

Read Operation DC Characteristics

T_A = 0°C to +70°C, V_{CC} = +5V ±5%, V_{DD} = −9V ±5%, V_{GG} = −9V ±5%, unless otherwise noted. Typical values are at nominal voltages and T_A = 25°C. (Note 2)

SYMBOL	PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS
I_{LI}	Address and Chip Select Input Load Current	V_{IN} = 0.0V			1	μA
I_{LO}	Output Leakage Current	V_{OUT} = 0.0V, $\overline{CS}$ = V_{CC} −2			1	μA
I_{DDO}	Power Supply Current	V_{GG} = V_{CC}, $\overline{CS}$ = V_{CC} −2 I_{OL} = 0.0 mA, T_A = 25°C, (Note 2)		5	10	mA
I_{DD1}	Power Supply Current	$\overline{CS}$ = V_{CC} −2, I_{OL} = 0.0 mA, T_A = 25°C		35	50	mA
I_{DD2}	Power Supply Current	$\overline{CS}$ = 0.0, I_{OL} = 0.0 mA, T_A = 25°C		32	46	mA
I_{DD3}	Power Supply Current	$\overline{CS}$ = V_{CC} −2, I_{OL} = 0.0 mA, T_A = 0°C		38.5	60	mA
I_{CF1}	Output Clamp Current	V_{OUT} = −1.0V, T_A = 0°C		8	14	mA
I_{CF2}	Output Clamp Current	V_{OUT} = −1.0, T_A = 25°C			13	mA
I_{GG}	Gate Supply Current				1	μA
V_{IL1}	Input Low Voltage for TTL Interface		−1.0		V_{CC}−4.1	V
V_{IL2}	Input Low Voltage for MOS Interface		V_{DD}		V_{CC}−6	V
V_{IH}	Address and Chip Select Input High Voltage		V_{CC}−2		V_{CC}+0.3	V
I_{OL}	Output Sink Current	V_{OUT} = 0.45V	1.6	4		mA
I_{OH}	Output Source Current	V_{OUT} = 0.0V	−2.0			mA
V_{OL}	Output Low Voltage	I_{OL} = 1.6 mA		−0.7	0.45	V
V_{OH}	Output High Voltage	I_{OH} = −100μA	3.5	4.5		V

Note 1: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and and functional operation of the device at these or at any other condition above those indicated in the operational sections of this specification is not implied. Exposure to Absolute Maximum Rating conditions for extended periods may affect device reliability.

Note 2: Power-Down Option: V_{GG} may be clocked to reduce power dissipation. The average I_{DD} will vary between I_{DD0} and I_{DD1} depending on the V_{GG} duty cycle (see typical characteristics). For this option, please specify MM1702AL.

Read Operation AC Characteristics

T_A = 0°C to +70°C, V_{CC} = +5V ±5%, V_{DD} = −9V ±5%, V_{GG} = −9V ±5%, unless otherwise noted.

SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS
Freq.	Repetition Rate			1	MHz
t_{OH}	Previous Read Data Valid			100	ns
t_{ACC}	Address to Output Delay		0.7	1	μs
t_{DVGG}	Clocked V_{GG} Set-Up (Note 1)	1			μs
t_{CS}	Chip Select Delay			100	ns
t_{CO}	Output Delay From $\overline{CS}$			900	ns
t_{OD}	Output Deselect			300	ns
t_{OHC}	Data Out Hold in Clocked V_{GG} Mode (Note 1)			5	μs

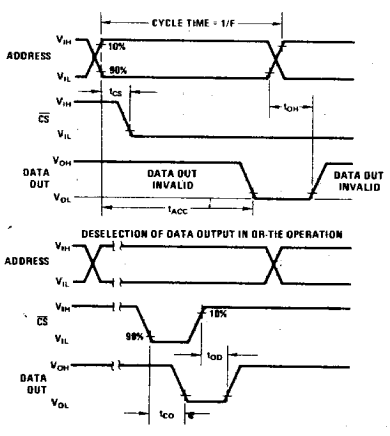
Capacitance Characteristics T_A = 25°C (Note 3)

SYMBOL	PARAMETER	CONDITIONS		MIN	TYP	MAX	UNITS
C_{IN}	Input Capacitance	All Unused	$V_{IN} = V_{CC}$		8	15	pF
C_{OUT}	Output Capacitance	Pins Are	$\overline{CS} = V_{CC}$		10	15	pF
C_{VGG}	V_{GG} Capacitance (Note 1)	At ac Ground	$V_{OUT} = V_{CC}$ $V_{GG} = V_{CC}$			30	pF

Note 3: This parameter is periodically sampled and is not 100% tested.

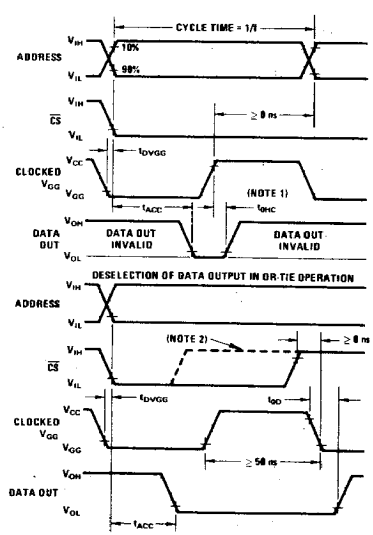
Read Operation Switching Time Waveforms

(a) Constant V_{GG} Operation



Conditions of Test:
Input pulse amplitudes: 0–4V, t_r, $t_f \leq 50$ ns. Output load is 1 TTL gate; measurements made at output of TTL gate ($t_{PD} \leq 15$ ns), C_L = 15 pF.

(b) Power-Down Option (Note 1)



Note 1: The output will remain valid for t_{OHC} as long as clocked V_{GG} is at V_{CC}. An address change may occur as soon as the output is sensed (clocked V_{GG} may still be at V_{CC}). Data becomes invalid for the old address when clocked V_{GG} is returned to V_{GG}.

Note 2: If $\overline{CS}$ makes a transition from V_{IL} to V_{IH} while clocked V_{GG} is at V_{GG}, then deselection of output occurs at t_{OD} as shown in static operation with constant V_{GG}.

Programming Operation DC Characteristics

$T_A = 25°C$, $V_{CC} = 0V$, $V_{BB} = 12V \pm 10\%$, $\overline{CS} = 0V$ unless otherwise noted.

SYMBOL	PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS
I_{LI1P}	Address and Data Input Load Current	$V_{IN} = -48V$			10	mA
I_{LI2P}	Program and V_{GG} Load Current	$V_{IN} = -48V$			10	mA
I_{BB}	V_{BB} Supply Load Current	(Note 5)		10	100	mA
I_{DDP}	Peak I_{DD} Supply Load Current	$V_{DD} = V_{PROG} = -48V$ $V_{GG} = -35V$ (Note 4)		200	300	mA
V_{IHP}	Input High Voltage				0.3	V
V_{IL1P}	Pulsed Data Input Low Voltage		−46		−48	V
V_{IL2P}	Address Input Low Voltage		−40		−48	V
V_{IL3P}	Pulsed Input Low V_{DD} and Program Voltage		−46		−48	V
V_{IL4P}	Pulsed Input Low V_{GG} Voltage		−35		−40	V

Note 4: I_{DDP} flows only during V_{DD}, V_{GG} on time. I_{DDP} should not be allowed to exceed 300 mA for greater than 100μs. Average power supply current I_{DDP} is typically 40 mA at 20% duty cycle.

Note 5: The V_{BB} supply must be limited to 100 mA max current to prevent damage to the device.

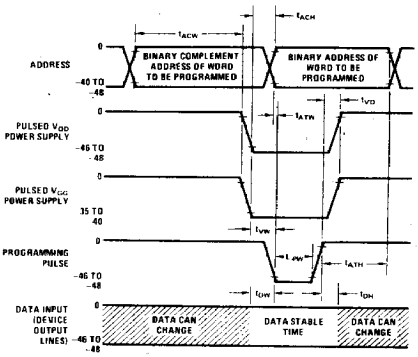
Programming Operation AC Characteristics

$T_A = 25°C$, $V_{CC} = 0V$, $V_{BB} = 12V \pm 10\%$, $\overline{CS} = 0V$ unless otherwise noted.

SYMBOL	PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS
	Duty Cycle (V_{DD}, V_{GG})				20	%
$t_{\phi PW}$	Program Pulse Width	$V_{GG} = -35V$, $V_{DD} = V_{PROG} = -48V$			3	ms
t_{DW}	Data Set-Up Time		25			μs
t_{DH}	Data Hold Time		10			μs
t_{VW}	V_{DD}, V_{GG} Set-Up		100			μs
t_{VD}	V_{DD}, V_{GG} Hold		10		100	μs
t_{ACW}	Address Complement Set-Up	(Note 6)	25			μs
t_{ACH}	Address Complement Hold	(Note 6)	25			μs
t_{ATW}	Address True Set-Up		10			μs
t_{ATH}	Address True Hold		10			μs

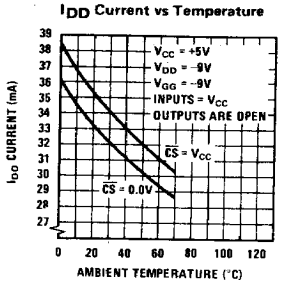
Note 6: All 8 address bits must be in the complement state when pulsed V_{DD} and V_{GG} move to their negative levels. The addresses (0–255) must be programmed as shown in the timing diagram until data reads true, then over-programmed 4 times that amount. (Symbolized by x + 4x.)

Programming Operation Switching Time Waveforms

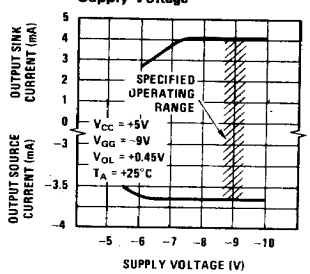


Typical Performance Characteristics

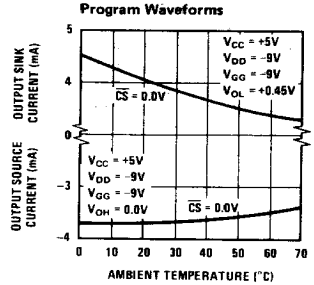
I_{DD} Current vs Temperature



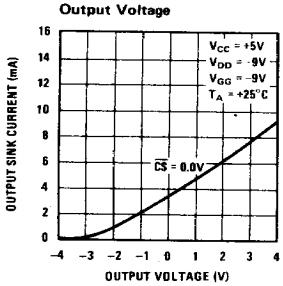
Output Current vs V_{DD} Supply Voltage



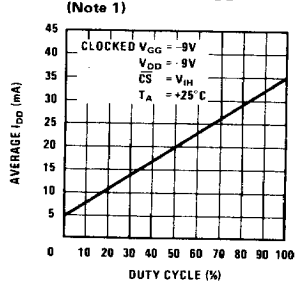
Program Waveforms



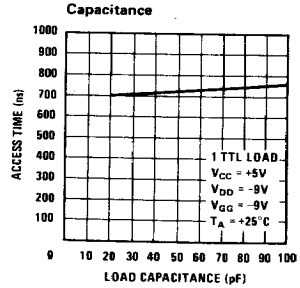
Output Sink Current vs Output Voltage



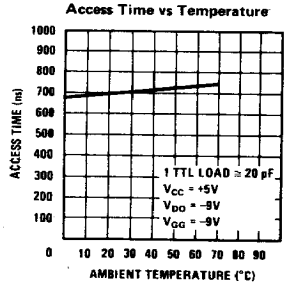
Average Current vs Duty Cycle for Clocked V_{GG} (Note 1)



Access Time vs Load Capacitance



Access Time vs Temperature



Operation of the MM1702A in Program Mode

Initially, all 2048 bits of the ROM are in the "0" state (output low). Information is introduced by selectively programming "1's" (output high) in the proper bit locations.

Word address selection is done by the same decoding circuitry used in the READ mode (see table for logic levels). All 8 address bits must be in the binary complement state when pulsed V_{DD} and V_{GG} move to their negative levels. The addresses must be held in their binary complement state for a minimum of 25μs after V_{DD} and V_{GG} have moved to their negative levels. The addresses must then make the transition to their true state a minimum of 10μs before the program pulse is applied. The addresses should be programmed in the sequence 0–255 for a minimum of 32 times. The eight output terminals are used as data inputs to determine the information pattern in the eight bits of each word. A low data input level (–48V) will program a "1" and a high data input level (ground) will leave a "0" (see table on page 4-4). All eight bits of one word are programmed simultaneously be setting the desired bit information patterns on the data input terminals.

During the programming, V_{GG}, V_{DD} and the Program Pulse are pulsed signals.

MM1702A Erasing Procedure

The MM1702A may be erased by exposure to high intensity short-wave ultraviolet light at a wavelength of 2537Å. The recommended integrated dose (i.e., UV intensity x exposure time) is 6W sec/cm^2. Examples of ultraviolet sources which can erase the MM1702A in 10 to 20 minutes are the Model UVS-54 and Model S-52 short-wave ultraviolet lamps manufactured by Ultra-Violet Products, Inc. (5114 Walnut Grove Avenue, San Gabriel, California). The lamps should be used without short-wave filters, and the MM1702A to be erased should be placed about one inch away from the lamp tubes. There exists no absolute rule for erase time. Establish a worst case time required with the equipment. Then over-erase by a factor of 2, i.e., if the device appears erased after 8 minutes, continue exposure for an additional 16 minutes for a total of 24 minutes. (May be expressed as x + 2x.)

9.7 1702 EPROM Specification

The following pages are Intel’s 1702 Silicon Gate MOS LSI PROM specification.

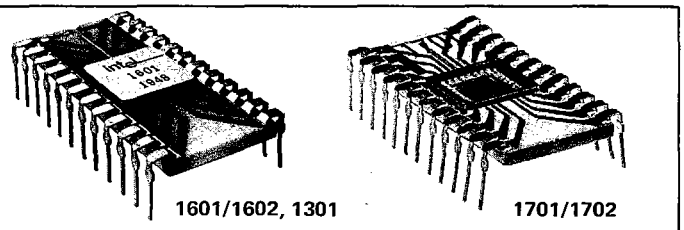
INTEL CORP. 3065 Bowers Avenue, Santa Clara, California 95051 • (408) 246-7501

1601/1701,1602/1702 ELECTRICALLY PROGRAMMABLE 1301 MASK PROGRAMMABLE

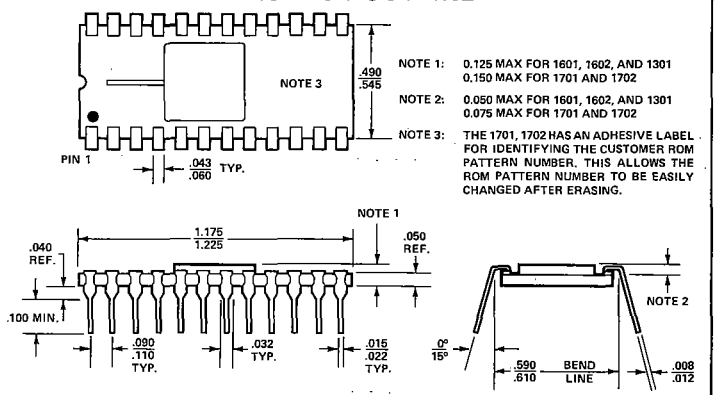
2048 BIT FULLY DECODED READ ONLY MEMORY

SEPTEMBER 1971

- Erasable and Field Reprogrammable (1701,1702)
- Field Programmable (1601,1602)
- All 2048 Bits Guaranteed Programmable – 100% factory tested (1601/1701,1602/1702)
- Inputs and outputs DTL and TTL compatible
- Static and Dynamic Operation (1601,1701,1301)
- Static Only Operation (1602,1702)
- OR-tie capability
- Simple Memory Expansion – Chip Select input lead
- 24 pin dual-in-line hermetically sealed ceramic package (1601,1602,1301)
- 24 pin dual-in-line, quartz lid ceramic package (1701,1702)



PACKAGE OUTLINE

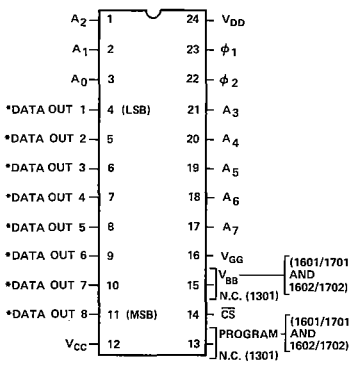


NOTE 1: 0.125 MAX FOR 1601, 1602, AND 1301
0.150 MAX FOR 1701 AND 1702

NOTE 2: 0.050 MAX FOR 1601, 1602, AND 1301
0.075 MAX FOR 1701 AND 1702

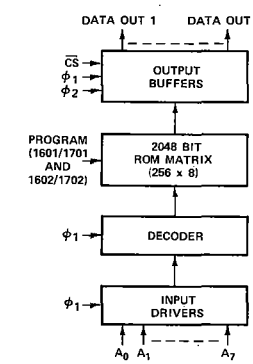
NOTE 3: THE 1701, 1702 HAS AN ADHESIVE LABEL FOR IDENTIFYING THE CUSTOMER ROM PATTERN NUMBER. THIS ALLOWS THE ROM PATTERN NUMBER TO BE EASILY CHANGED AFTER ERASING.

PIN CONFIGURATION



*THIS PIN IS THE DATA INPUT LEAD FOR THE 1601/1701 AND 1602/1702 DURING PROGRAMMING

BLOCK DIAGRAM



NOTE: LOGIC 1 AT INPUT AND OUTPUT IS A HIGH AND LOGIC 0 IS LOW.

The Intel 1601, 1602, 1701, and 1702 is a 256 word by 8 bit electrically programmable ROM ideally suited for uses where fast turnaround and pattern experimentation are important such as in prototype or in one of a kind systems. The 1601, 1602, 1701, and 1702 is factory reprogrammable which allows Intel to perform a complete programming and functional test on each bit position before delivery.

The four devices 1601, 1602, 1701, and 1702 use identical chips. The 1601 and 1701 is operable in both the static and dynamie mode while the 1602 and 1702 is operable in the static mode only. Also, the 1701 and 1702 has the unique feature of being completely erasable and field reprogrammable. This is accomplished by a quartz lid that allows high intensity ultraviolet light to erase the 1701 and 1702. A new pattern can then be written into the device. This procedure can be repeated as many times as required.

The 1301 is a direct replacement part which is programmed by a metal mask and is ideal for large volume and lower cost production runs of systems initially using the 1601/1701 or the static only 1602/1702.

The dynamic mode of the 1601/1701 and 1301 refers to the decoding circuitry and not to the memory cell. Dynamic operation offers higher speed and lower power dissipation than the static operation.

The 1601, 1602, 1701, and 1702 is fabricated with silicon gate technology. This low threshold technology allows the design and production of higher performance MOS circuits and provides a higher functional density on a monolithic chip than conventional MOS technologies.

STATIC, DYNAMIC, PROGRAMMING MODE PIN CONNECTIONS

To operate the 1601/1701, 1602/1702, 1301 in either a static, dynamic, or programming (1601/1701, 1602/1702) mode[1], the following external lead connections are required in addition to those shown by the pin configuration diagram on page 1.

PIN / MODE	13[2] (Program)	15[2] (V_{BB})	16 (V_{GG})	22 (ϕ_2)	23 (ϕ_1)
Static	V_{CC}	V_{CC}	V_{GG}	V_{CC}	V_{CC}
Dynamic (1601/1701, 1301)	$\emptyset_1$	V_{CC}	V_{CC}	$\emptyset_2$	$\emptyset_1$
Programming (1601/1701,1602/1702)	Program Pulse	V_{BB}	Pulsed V_{GG} (V_{IL4P})	GND	GND

Absolute Maximum Ratings*

Case Temperature Under Bias	0°C to +85°C
Storage 1601/1701, 1602/1702	−65°C to +125°C
Temperature: 1301	−65°C to +160°C
Soldering Temperature of Leads (10 sec)	+300°C
Power Dissipation	1 Watt
Static and Dynamic Operation: Input Voltages and Supply Voltages with respect to V_{CC}	+0.5V to −20V
Program Operation: Input Voltages and Supply Voltages with respect to V_{CC}	−50V

*COMMENT

Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or at any other condition above those indicated in the operational sections of this specification is not implied. Exposure to Absolute Maximum Rating conditions for extended periods may affect device reliability.

STATIC OPERATION FOR 1601/1701, 1602/1702 AND 1301

D.C. and Operating Characteristics for Static Operation

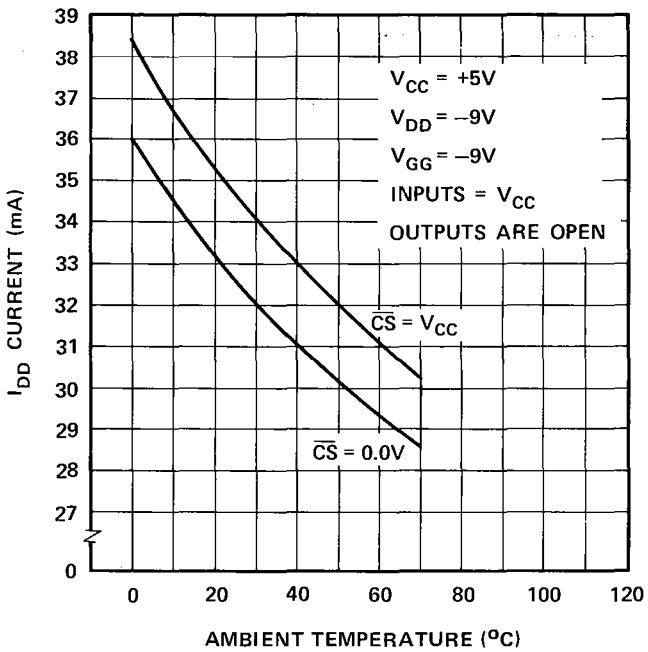
T_A = 0°C to 70°C, V_{CC} = +5V±5%, V_{DD} = −9V±5%, V_{GG}[3] = −9V±5%, unless otherwise noted.

SYMBOL	TEST	MIN.	TYP.[4]	MAX.	UNIT	CONDITIONS
I_{LI}	Address and Chip Select Input Load Current			1	µA	V_{IN} = 0.0V
I_{LO}	Output Leakage Current			1	µA	V_{OUT} = 0.0V, $\overline{CS}$ = V_{CC} −2
I_{DDO}	Power Supply Current		5	10	mA	V_{GG}=V_{CC}, $\overline{CS}$=V_{CC}−2 I_{OL}= 0.0mA, T_A = 25°C
I_{DD1}[5]	Power Supply Current		35	50	mA	$\overline{CS}$=V_{CC} −2 I_{OL} =0.0mA, T_A = 25°C
I_{DD2}[5]	Power Supply Current		32	46	mA	$\overline{CS}$=0.0 I_{OL} =0.0mA, T_A = 25°C
I_{DD3}[5]	Power Supply Current		38.5	60	mA	$\overline{CS}$=V_{CC} −2 I_{OL} =0.0mA , T_A = 0°C
I_{GG}	Gate Supply Current			1	µA	
V_{IL}	Address and Chip Select Input Low Voltage	V_{CC}−10		V_{CC}−4.2	V	
V_{IH}	Address and Chip Select Input High Voltage	V_{CC}−2		V_{CC}+.3	V	
I_{OL}	Output Sink Current	1.6	4		mA	V_{OUT} = 0.45V
I_{CF}	Output Clamp Current		8	13	mA	V_{OUT} = −1.0V
I_{OH}	Output Source Current	−2.0			mA	V_{OUT} = 0.0V
V_{OL}	Output Low Voltage		−.7	0.45	V	I_{OL} = 1.6mA
V_{OH}	Output High Voltage	3.5	4.5		V	I_{OH} = −100 µA

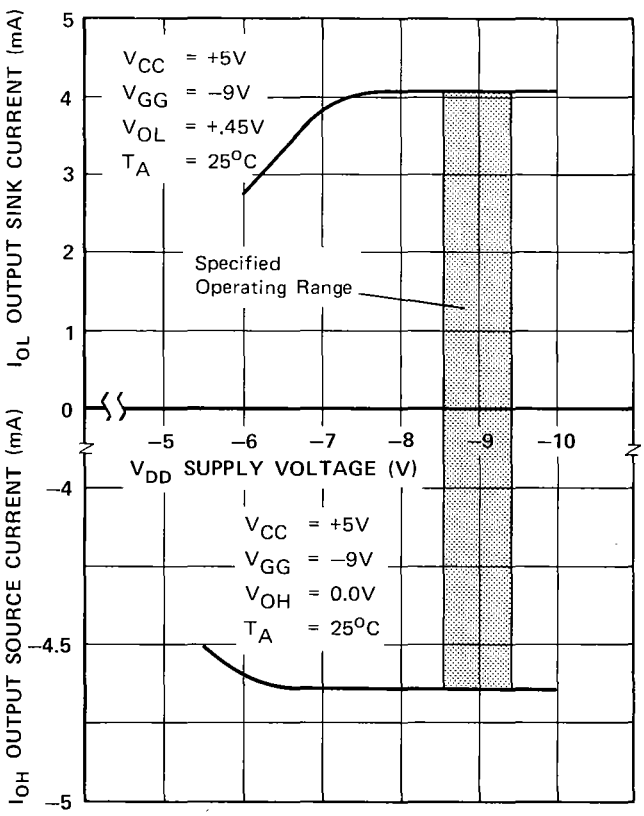
Note 1: In the programming mode, the data inputs 1–8 are pins 4–11 respectively. $\overline{CS}$ = GND.
Note 2: This external lead connection is only necessary on the 1601/1701 and 1602/1702. It may be unconnected on the 1301.
Note 3: V_{GG} may be clocked to reduce power dissipation. In this mode average I_{DD} increases in proportion to V_{GG} duty cycle. (See p. 5)
Note 4: Typical values are at nominal voltages and T_A = 25°C.
Note 5: Measured under continuous operation.

Typical Characteristics for Static Operation

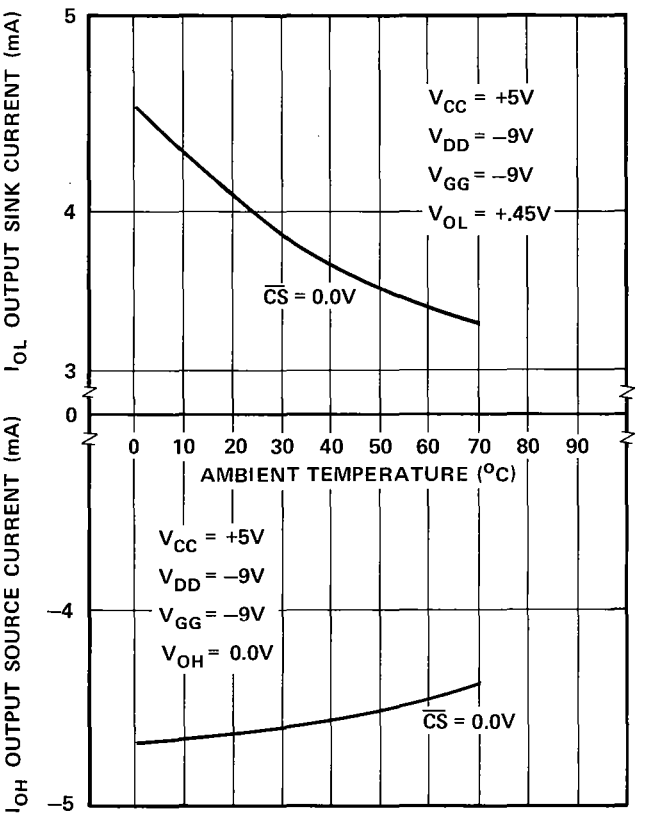
I_{DD} CURRENT VS. TEMPERATURE



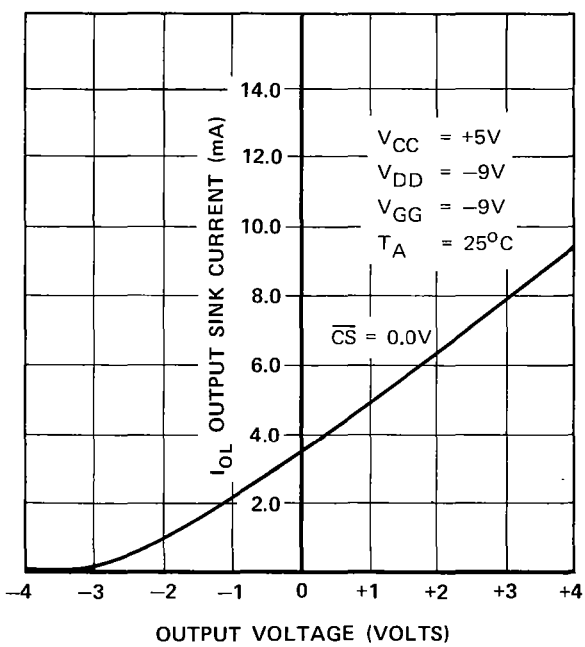
OUTPUT CURRENT VS. V_{DD} SUPPLY VOLTAGE



OUTPUT CURRENT VS. TEMPERATURE



OUTPUT SINK CURRENT VS. OUTPUT VOLTAGE



A.C. Characteristics for Static Operation

T_A = 0° C to +70° C, V_{CC} = +5V ±5%, V_{DD} = −9V ±5%, V_{GG} = −9V ±5% unless otherwise noted

SYMBOL	TEST	1601/1701, 1602/1702 MIN.	TYP.	MAX.	1301 MIN.	TYP.	MAX.	UNIT
Freq	Repitition rate			1			1.1	Mhz
t_{OH}	Previous read data valid			100			100	ns
t_{ACC}	Address to output delay		.700	1			.900	μs
t_{DVGG}	Clocked V_{GG} set up	0			1			μs
t_{CS}	Chip select delay			100			200	ns
t_{CO}	Output delay from $\overline{CS}$			900			500	ns
t_{OD}	Output deselect			300			300	ns
t_{OHC}	Data out hold in clocked V_{GG} mode (Note 1)			5			5	μs
C	Capacitance		See page 10			See page 10		

NOTE 1: The output will remain valid for t_{OHC} as long as clocked V_{GG} is at V_{CC}. An address change may occur as soon as the output is sensed (clocked V_{GG} may still be at V_{CC}). Data becomes invalid for the new address when clocked V_{GG} is returned to V_{GG}.

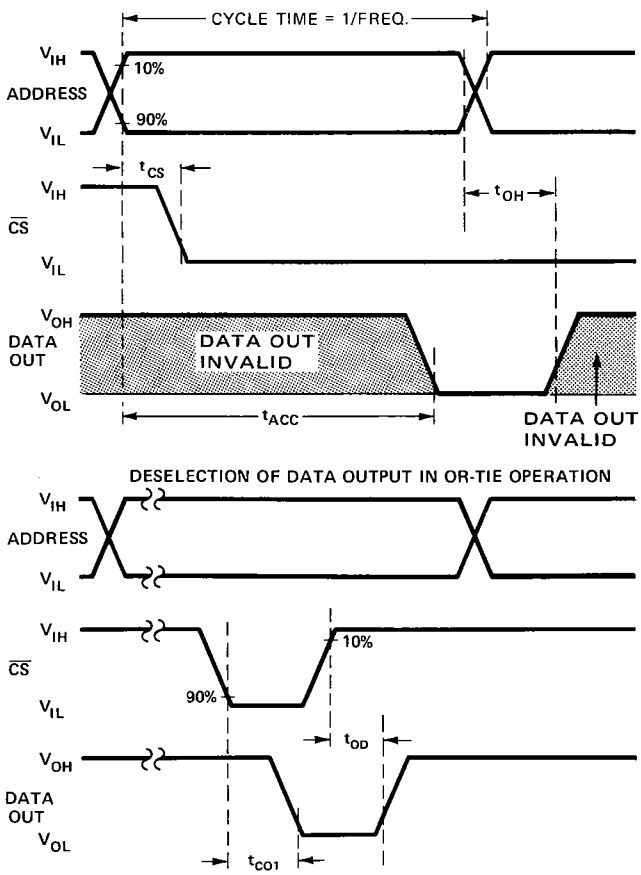
Switching Characteristics for Static Operation

Conditions of Test:

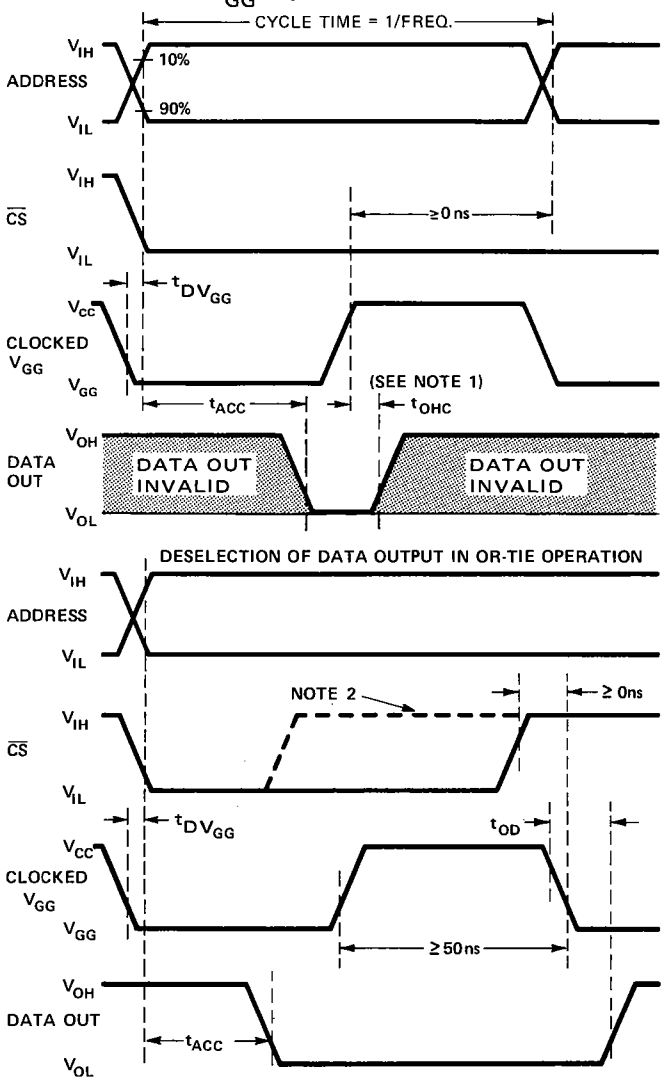
Input pulse amplitudes: 0 to 4V; t_R, t_F ≤50 ns

Output load is 1 TTL gate; measurements made at output of TTL gate (t_{PD} ≤15 ns)

A) Normal Operation (Constant V_{GG})



B) Clocked V_{GG} Operation

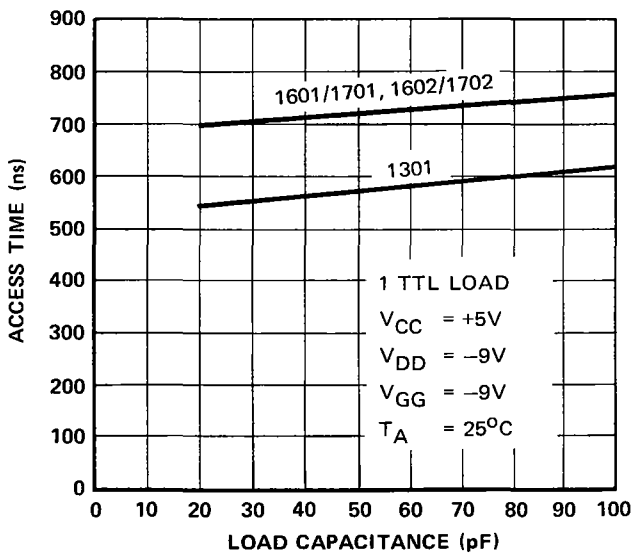


NOTE 1: The output will remain valid for t_{OHC} as long as clocked V_{GG} is at V_{CC}. An address change may occur as soon as the output is sensed (clocked V_{GG} may still be at V_{CC}). Data becomes invalid for the new address when clocked V_{GG} is returned to V_{GG}.

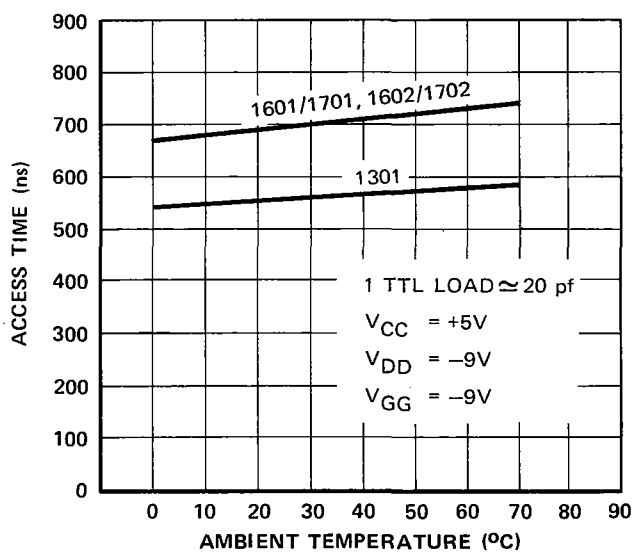
NOTE 2: If $\overline{CS}$ makes a transition from V_{IL} to V_{IH} while clocked V_{GG} is at V_{GG}, then deselection of output occurs at t_{OD} as shown in static operation with constant V_{GG}.

Typical Characteristics for Static Operation

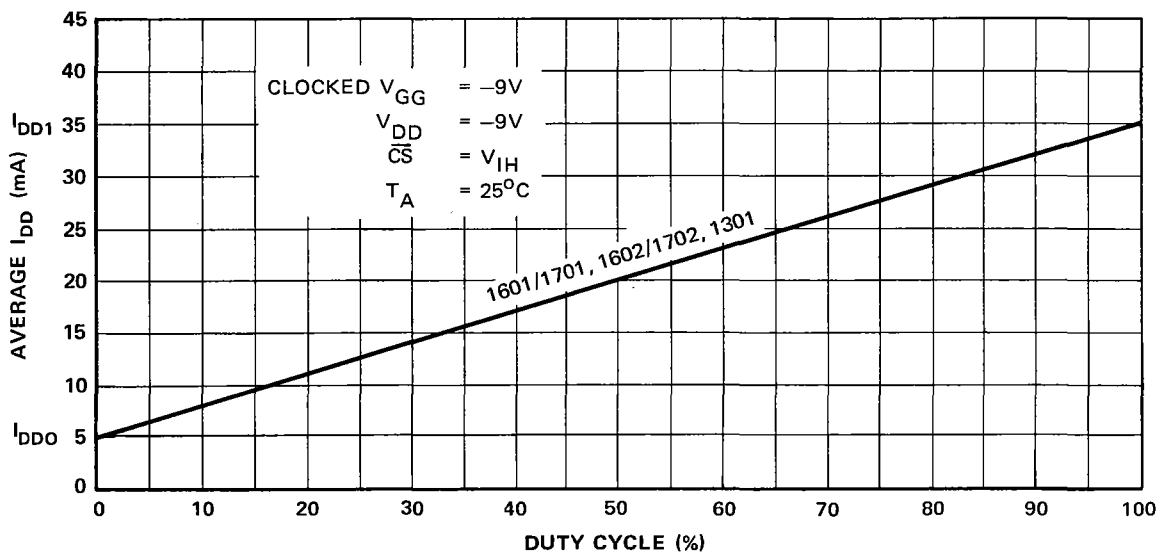
ACCESS TIME VS. LOAD CAPACITANCE



ACCESS TIME VS. TEMPERATURE



AVERAGE CURRENT VS. DUTY CYCLE FOR CLOCKED V_{GG}



DYNAMIC OPERATION FOR 1601/1701 AND 1301 ONLY

D.C. and Operating Characteristics for Dynamic Operation

T_A = 0°C to +70°C, V_{CC} = V_{GG} = +5V ±5%, V_{DD} = −9V ±5%, unless otherwise noted

SYMBOL	TEST	MIN.	TYP.	MAX.	UNIT	CONDITIONS
I_{LI}	Address and Chip Select Input Load Current			1	µA	V_{IN} = 0.0V
I_{LC}	Ø1, Ø2 Clock Leakage Current			10	µA	V_{ILC} = V_{CC} −14.5
I_{LO}	Output Leakage Current			10	µA	V_{out} = 0.0V $\overline{CS}$ = V_{CC} −2
I_{DDO}	Average Power Supply Current		5	10	mA	Clock Voltages = V_{IHC} T_A = 25°C, $\overline{CS}$ = V_{CC} −2
*I_{DD4}	Average Power 1601/1701		30	45	mA	$\overline{CS}$ = V_{CC} −2 1 Mhz rate; clock width at min spec value
	Supply Current 1301		28	40	mA	
	@ T_A = 25°C					
*I_{DD5}	Average Power 1601/1701			55	mA	
	Supply Current 1301			50	mA	
	@ T_{case} = 0°C					
V_{IL}	Address and Chip Select Input Low Voltage	V_{CC} −10		V_{CC} −4.2	V	
V_{ILC}	Ø1, Ø2 Input Low Voltage	V_{CC} −14.5		V_{CC} −12.6	V	
V_{IH}	Address and Chip Select Input High Voltage	V_{CC} −2		V_{CC} +.3	V	
V_{IHC}	Ø1, Ø2 Input High Voltage	V_{CC} −1		V_{CC} +.3	V	
I_{OL}	Output Sink Current	1.6			mA	V_{out} = 0.45V
I_{CF}	Output Clamp Current		8	13	mA	V_{out} = −1.0V
I_{OH}	Output Source Current	−2			mA	V_{out} = 0.0V
V_{OL}	Output Low Voltage		−.7	0.45	V	I_{OL} = 1.6mA
V_{OH}	Output High Voltage	3.5	4.5		V	I_{OH} = −100µA

*I_{DD} flows mainly during $t_{\phi 1PW}$. I_{DD} is directly proportional to ϕ_1 clock duty cycle.

A.C. Characteristics for Dynamic Operation

T_A = 0°C to +70°C, V_{CC} = +5V ±5%, V_{DD} = −9V ±5% unless otherwise noted

SYMBOL	TEST	1601, 1701			1301			UNIT
		MIN.	TYP.	MAX.	MIN.	TYP.	MAX.	
$t_{\phi 1PW}$	ϕ_1 Clock Pulse Width	0.260		2	0.260		2	μs
$t_{\phi 2PW}$	ϕ_2 Clock Pulse Width	0.140		2	0.140		2	μs
$t_{\phi D1}$	ϕ_2 delay from ϕ_1	0.150		2	0.150		2	μs
$t_{\phi D2}$	ϕ_1 delay from ϕ_2	0.05			0.05			μs
t_r, t_f	Clock Pulse Transition			50			50	ns
t_{ACC1}	Address to Output Access		450	650		450	650	ns
t_{ACC2}	Output Access from ϕ_2			130			130	ns
t_{CD}	Chip Select to ϕ_1 Overlap	0			0			ns
t_{DES}	Deselection of Data Output			150			150	ns
C	Capacitance	See page 10			See page 10			

Switching Characteristics for Dynamic Operation

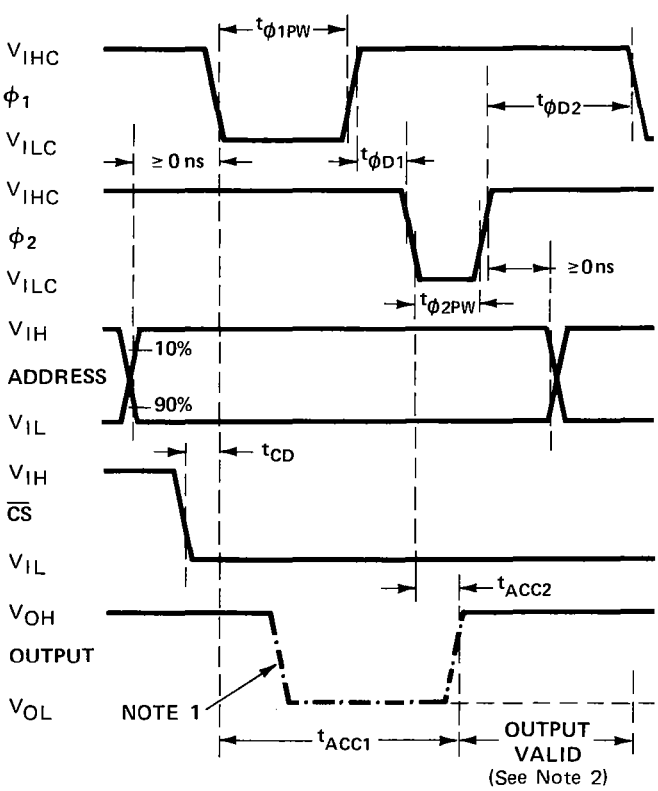
DYNAMIC OPERATION

Conditions of Test:

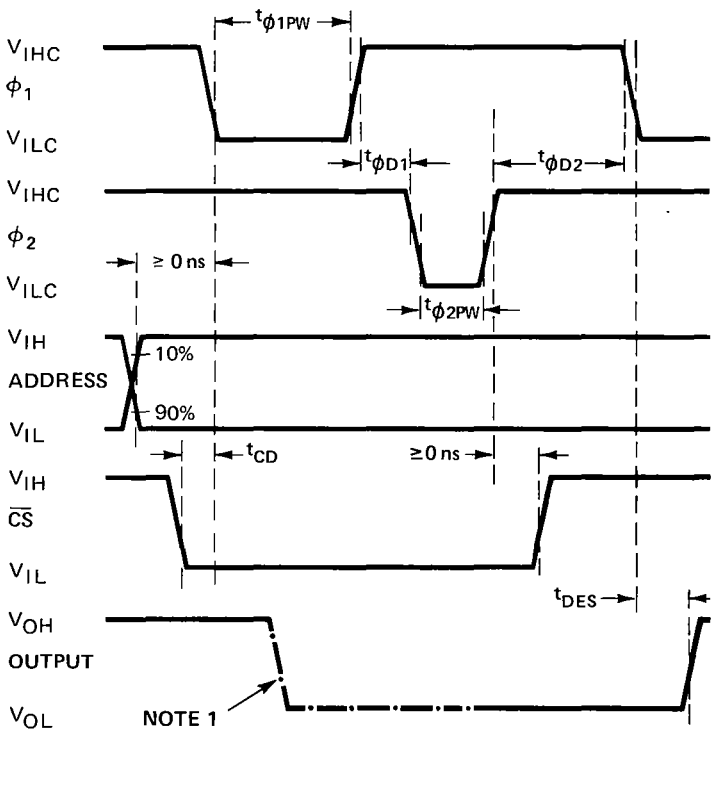
Input pulse amplitudes: 0 to 4V, Input pulse rise and fall times ≤50 nsec

Output load is 1 TTL gate; measurements made at output of TTL gate (t_{pd} ≤ 15 nsec)

A) Normal Operation



B) Deselection of Data Output In OR-tie Operation

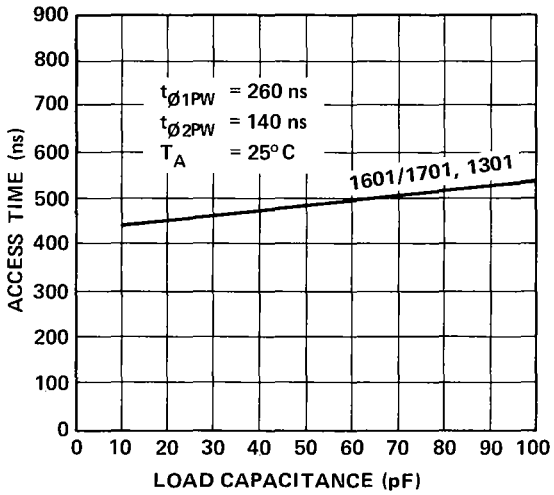


Note 1: An output low transition occurs for every ϕ_1 period independent of memory information.

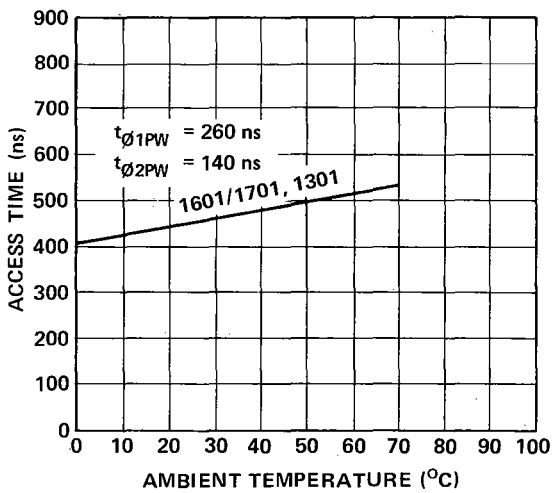
Note 2: Output will remain valid for 2 μsec as long as ϕ_1 does not occur.

Typical Characteristics for Dynamic Operation

ACCESS TIME VS. LOAD CAPACITANCE



ACCESS TIME VS. TEMPERATURE



PROGRAMMING OPERATION FOR THE 1601/1701 AND 1602/1702 ONLY

D.C. and Operating Characteristics for Programming Operation

$T_A = 25°C$, $V_{CC} = 0V$, $V_{BB} = +12V \pm 10\%$, $\overline{CS} = 0V$ unless otherwise noted

SYMBOL	TEST	MIN.	TYP.	MAX.	UNIT	CONDITIONS
I_{LI1P}	Address and Data Input Load Current	10			mA	$V_{IN} = -40V$
I_{LI2P}	Program and V_{GG} Load Current	10			mA	$V_{IN} = -48V$
I_{BB}	V_{BB} Supply Load Current		.05	1	mA	
I_{DDP}(1)	Peak I_{DD} Supply Load Current		750		mA	$V_{DD} = V_{prog} = -50V$ $V_{GG} = -35V$
V_{IHP}	Input High Voltage			0.3	V	
V_{IL1P}	Pulsed Data Input Low Voltage	−40		−48	V	
V_{IL2P}	Address Input Low Voltage	−40		−48	V	
V_{IL3P}	Pulsed Input Low V_{DD} and Program Voltage	−48		−50	V	
V_{IL4P}	Pulsed Input Low V_{GG} Voltage	−35		−40	V	

Note 1: I_{DDP} flows only during program period $t_{\phi PW}$.
Average power supply current I_{DDP} is typically 15 mA at 2% duty cycle.

A.C. Characteristics for Programming Operation

($T_{AMBIENT}$ = 25°C, V_{CC} = 0V, V_{BB} = + 12V ± 10%, $\overline{CS}$ = 0V unless otherwise noted

SYMBOL	TEST	MIN.	TYP.	MAX.	UNIT	CONDITIONS
	Duty Cycle			2	%	
$t_{\phi PW}$(1)	Program Pulse Width			20	ms	V_{GG}=−35V,V_{DD}=$V_{program}$=−48V
t_{DW}	Data Set Up Time	1			µs	
t_{DH}	Data Hold Time	1			µs	
t_{VD}	Pulsed V_{GG} and V_{DD} Supply Overlap	1			µs	
C	Capacitance	See page 10				

Note 1: Maximum duty cycle of $t_{\phi PW}$ should not be greater than 2% of cycle time so that power dissipation is minimized. To guarantee long term memory retention the program cycle should be repeated five times with $t_{\phi PW}$ = 20 msec or the equivalent thereof, e.g. 10 cycles of $t_{\phi PW}$ = 10 msec.

Switching Characteristics for Programming Operation

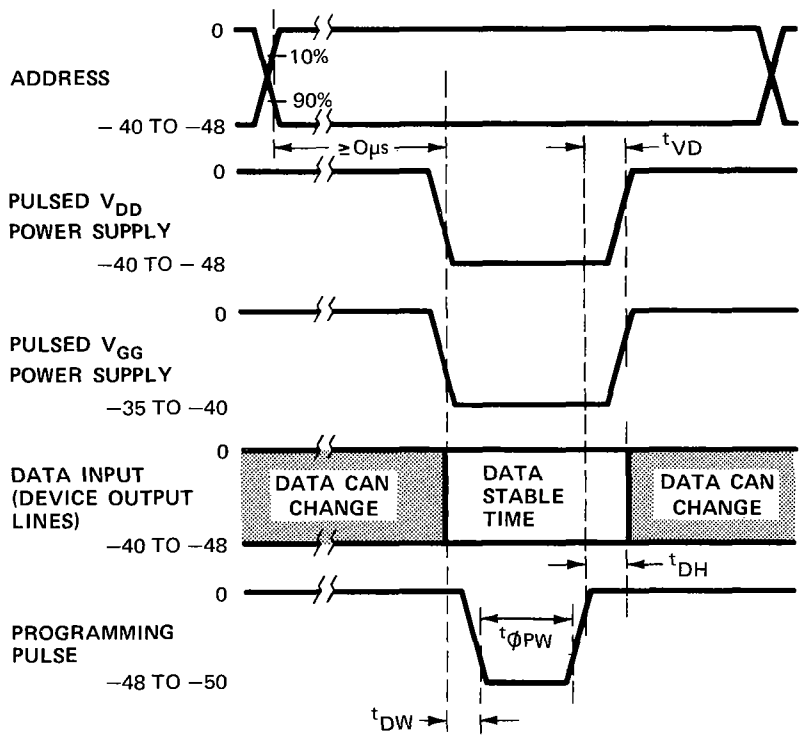
PROGRAM OPERATION

Conditions of Test:

Input pulse rise and fall times ≤250nsec

$\overline{CS}$ = 0V

PROGRAM WAVEFORMS



Programming Operation of the 1601/1701 and 1602/1702

When the Data Input for the Program Mode is:	Then the Data Output during the Read Mode is:
$V_{IL1P} = \sim -40V$ pulsed	Logic 1 = V_{OH} = 'P' on tape
$V_{IHP} = \sim 0V$	Logic 0 = V_{OL} = 'N' on tape

WORD \ ADDRESS	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	1
|	|	|	|	|	|	|	|	|
|	|	|	|	|	|	|	|	|
255	1	1	1	1	1	1	1	1

Address Logic Level During Read Mode: Logic 0 = V_{IL} ($\sim$.3V) Logic 1 = V_{IH} ($\sim$3V)

Address Logic Level During Program Mode:* Logic 0 = V_{IL2P} ($\sim -40V$) Logic 1 = V_{IHP} ($\sim$0V)

*The Logic Levels for the address inputs are inverted from the Logic Levels for the data inputs during the Program Mode.

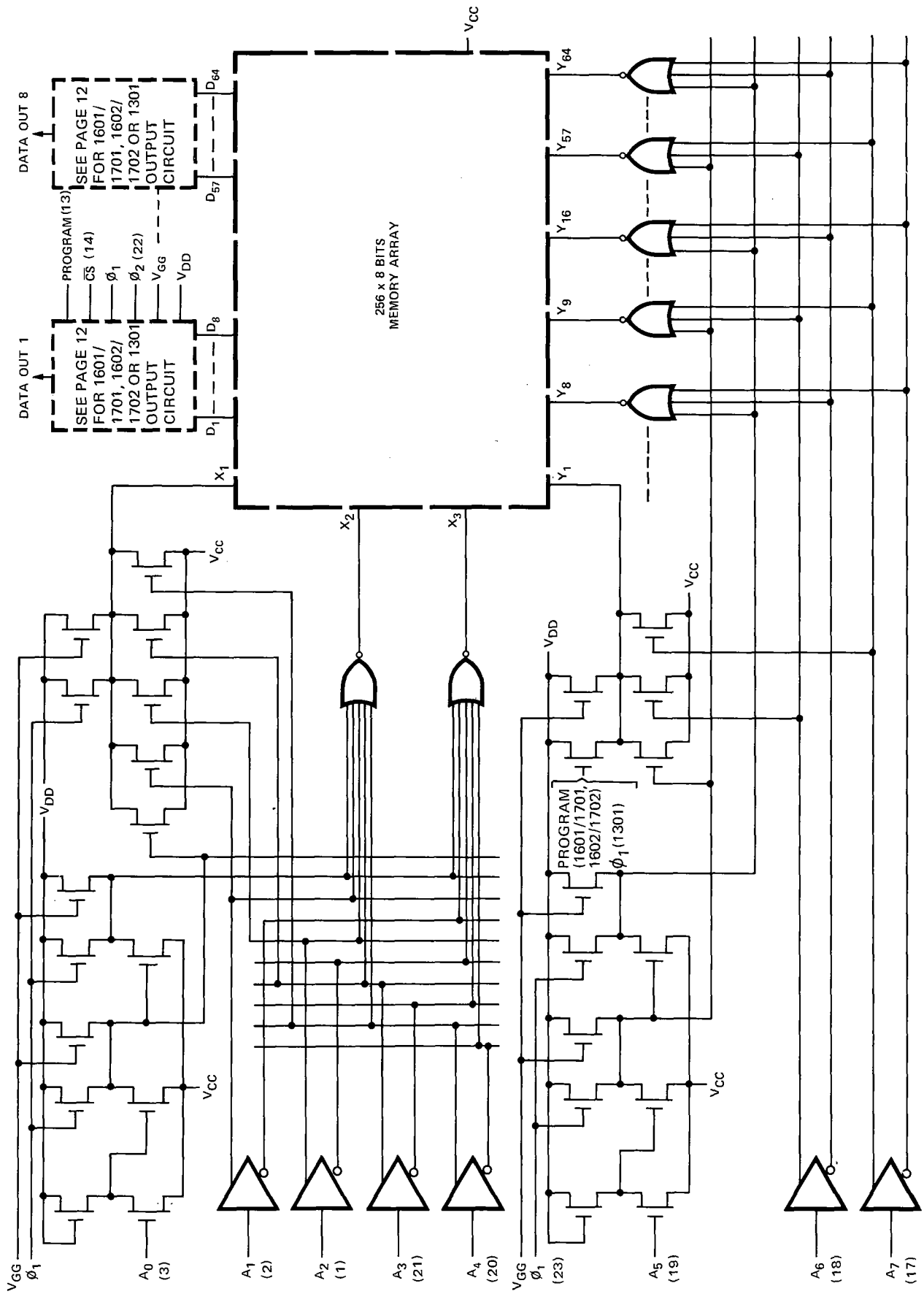
CAPACITANCE*
A.C. Characteristics, T_A = 25°C

SYMBOL	TEST	1601/1701, 1602/1702 MIN.	TYP.	MAX.	1301 MIN.	TYP.	MAX.	UNIT	CONDITIONS
C_{IN}	Input Capacitance		8	15		5	10	pF	$V_{IN}=V_{CC}$, $\overline{CS}=V_{CC}$
C_{out}	Output Capacitance		10	15		5	10	pF	$V_{out}=V_{CC}$, $\overline{CS}=V_{CC}$
$C_{\phi 1}$	ϕ_1 Clock Capacitance (includes pin 13)		35	55		20	30	pF	$V_{\phi 1}=V_{CC}$
$C_{\phi 2}$	ϕ_2 Clock Capacitance		9	15		7	15	pF	$V_{\phi 2}=V_{CC}$
$C_{V_{GG}}$	V_{GG} Capacitance (Clocked V_{GG} Mode)			30			30	pF	$V_{GG}=V_{CC}$

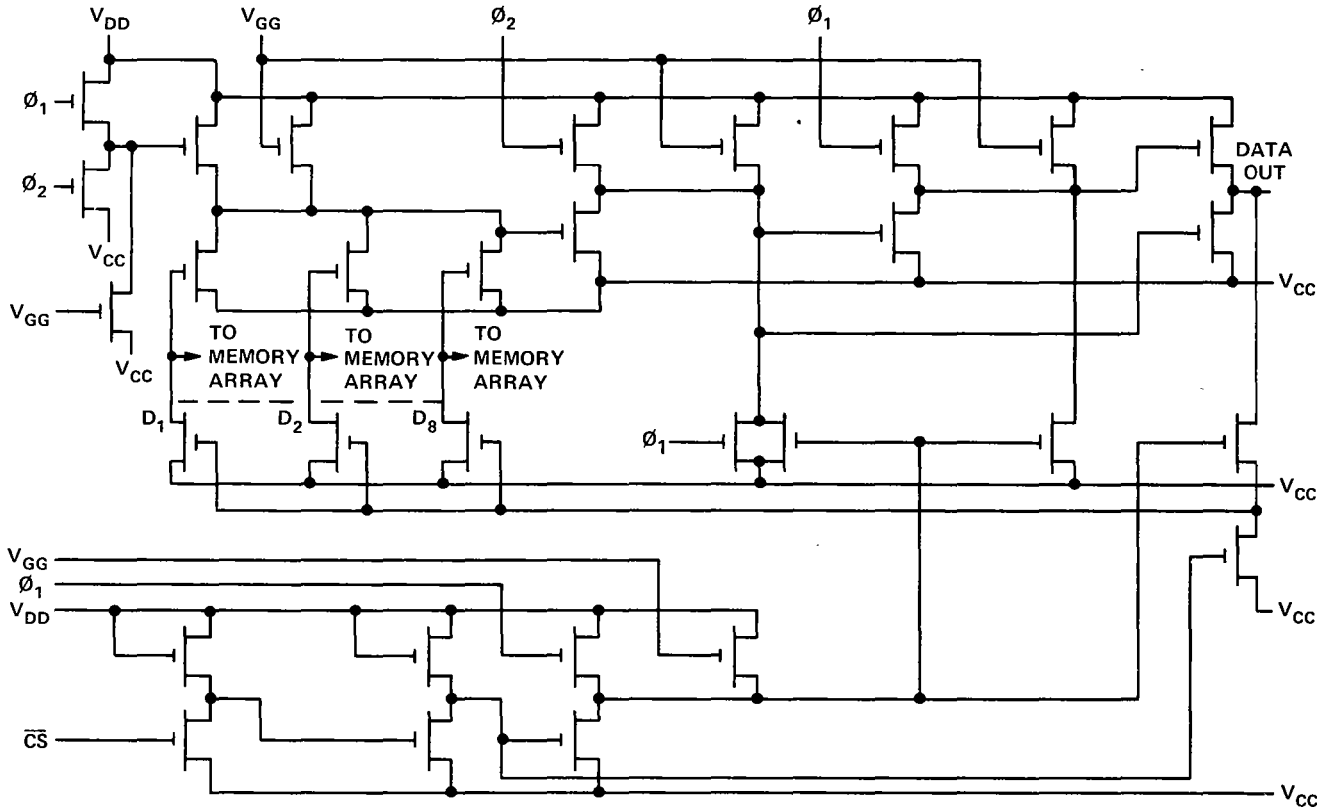
All unused pins are at A.C. ground

*This parameter is periodically sampled and is not 100% tested

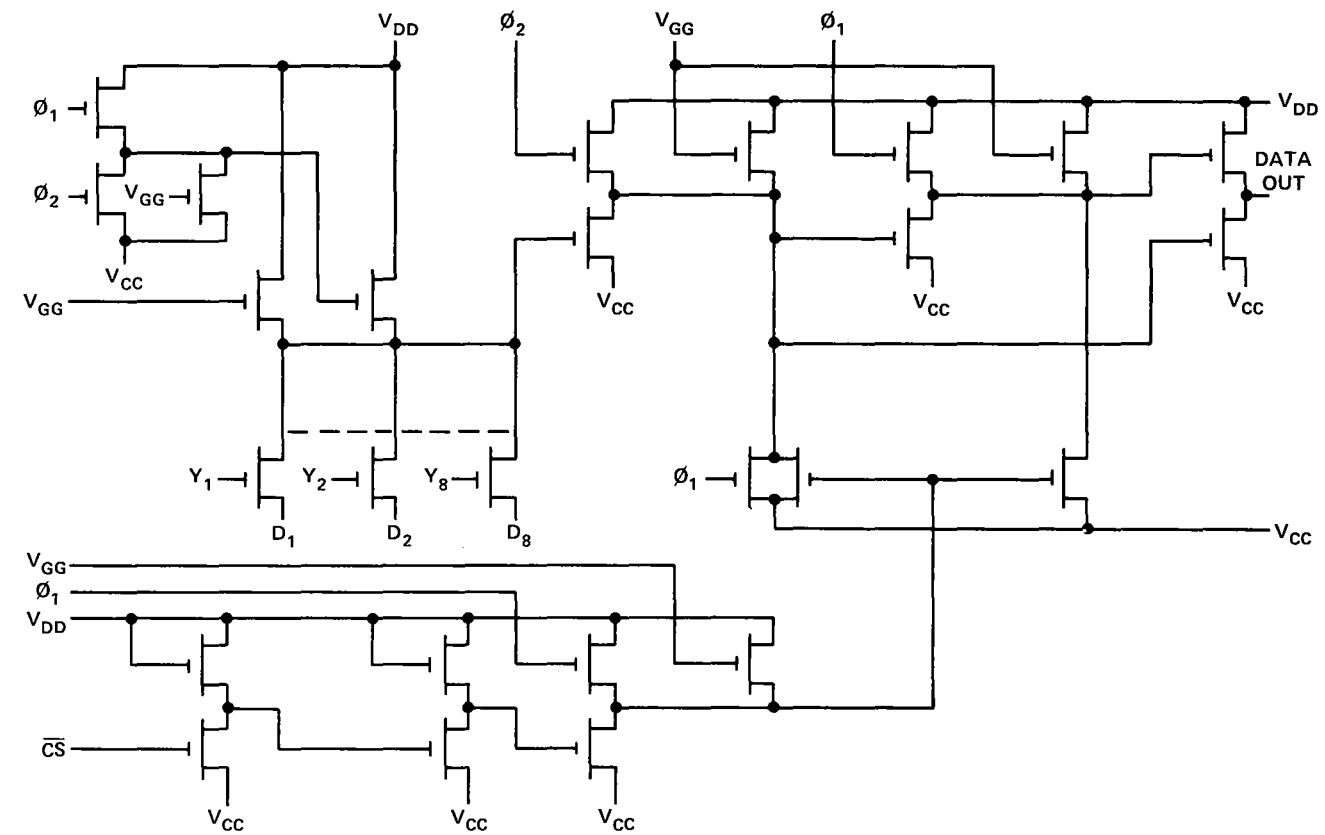
1601/1701, 1602/1702, 1301 Circuit Schematic



1 of 8 Output Circuits 1601/1701, 1602/1702



1 of 8 Output Circuits 1301



Application Information

I. OPERATION OF THE 1601/1701 AND 1602/1702 IN PROGRAM MODE

Initially, all 2048 bits of the ROM are in the "1" state (output high). Information is introduced by selectively programming "0"s (output low) in the proper bit locations.

Word address selection is done by the same decoding circuitry used in the READ mode (see table on page 10 for logic levels). The eight output terminals are used as data inputs to determine the information pattern in the eight bits of each word. A low data input level (−40V) will leave a "1" and a high data input level (ground) will allow programming of "0" (see table on page 10). All eight bits of one word are programmed simultaneously by setting the desired bit information patterns on the data input terminals. The duty cycle of the Program pulse (amplitude and width as specified on page 10) should be limited to 2%. The address should be applied for at lease 1 μsec before application of the Program pulse.

During the programming, V_{GG}, V_{DD} and the Program Pulse are pulsed signals.

II. MANUAL PROGRAMMING OF THE 1601, 1602, 1701, AND 1702

The 1601, 1602, 1701, or 1702, may be programmed by a machine such as the 7600 programmer or manually using a circuit similar to the one on pages 14 and 15. A parts list (pages 16 and 17) and the circuit board layout (pages 16 and 17) is also given. The circuit is capable of programming as well as reading the ROM. Programming takes approximately one hour.

Circuit Operation

1. In the read mode, the 1601, 1602, 1701, or 1702, is operated with V_{CC}=0 and V_{DD} =−14V (rather than +5 and −9V). The ROM is biased for static operation, and the sensed output signals from the ROM are used to drive transistors which in turn drive LED display devices. Input addresses are biased at levels of 0V and −5V for logic 1 and logic 0 respectively.

2. In the write mode, the 1601, 1602, 1701, or 1702, is operated in a pulsed mode. An astable multivibrator, running at about 2 pulses per second, drives a transistor which normally biases the negative regulator off. At each pulse, the negative regulator is allowed to apply −48 to −50 volts to the V_{DD} terminal of the ROM. Address and data voltages are derived using a simple emitter follower circuit as a regulator. V_{GG} during programming is derived using a zener diode and a resistive divider. In read mode, this circuit causes V_{GG} to equal V_{DD}. During each V_{DD} pulse, the program pulse is held off for about 2 μsec using a 100 pF capacitor, 47K resistor and 1N914 diode at the input of the program pulse driver circuit. The program pulse driver is biased to turn off prior to turn off of the V_{DD} pulse.

The entire circuit can be operated from a single −60V to −80V unregulated source. The +12V V_{BB} needed during programming is derived using a capacitive coupled circuit with a 12V zener regulator.

Program pulses average about 10 msec in length. At two per second, the circuit must be operated for 5 seconds to achieve a total of 100 msec of program pulses. The rate of 10 msec of program pulse every .5 second insures that a 2% programming duty cycle will not be exceeded.

Circuit Checkout (to be done before plugging in a Unit)

1. Set the read/write, 4PDT switch ($S1_A$ through $S1_D$) to the R(Read) position:
   (a) Adjust the (V_{DD}, RD) resistor so that V_{DD} reads −14 volts (T.P.1)

2. Set the read/write, 4PDT switch to the W(Write) position. Depress the "Write" push button and hold for the following sequence of adjustments. **They must be done in the order shown:**
   (a) Adjust the (V_{DD}, PR) resistor so that V_{DD} reads −49 volts (T.P.1).
   (b) Adjust the (V_{GG}, ADJ) resistor so that V_{GG} reads −36 volts (T.P.2).
   (c) Adjust the (V_{ADR}, ADJ) resistor so that V_{ADR} reads −40 volts (T.P.3).

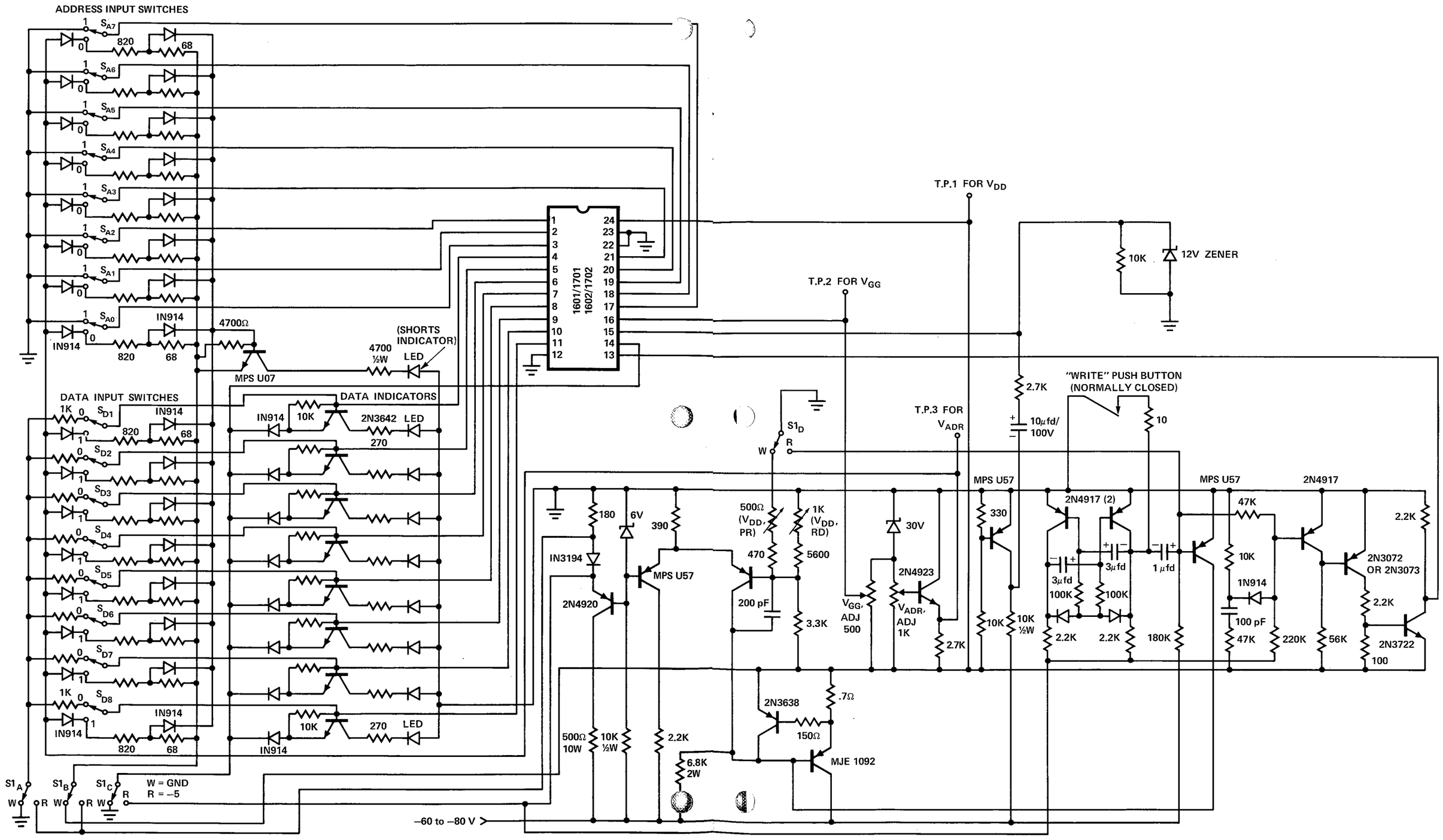
3. While the read/write switch is in the write position, the shorts indicator should be checked to see if there are any shorts on the data or address input pins. This would be indicated by the shorts indicator flashing when the "write" push button is depressed.

See page 18 for Programming Instructions For Manual Programmer

Intel cannot assume responsibility for the programming circuit described in this data sheet.

Circuit Schematic

1601/1701 BASIC MANUAL PROGRAMMER



PARTS LIST

Resistors	Quantity
0.7 Ω½W	1 ea.
10 Ω	1 ea.
68 Ω	16 ea.
100 Ω	2 ea.
270 Ω	8 ea.
330 Ω	1 ea.
390 Ω	1 ea.
680 Ω	1 ea.
820 Ω	16 ea.
1000 Ω	8 ea.
2000 Ω	1 ea.
2200 Ω½W	2 ea.
2200 Ω	3 ea.
2700 Ω	2 ea.
3.3K Ω	1 ea.
4.7K Ω	1 ea.
4.7K Ω½W	1 ea.
6.8K Ω2W	1 ea.
10K Ω	11 Ea.
10K Ω½W	2 ea.
47K Ω	2 ea.
56K Ω	1 ea.
100K Ω	2 ea.
180K Ω	1 ea.
220K Ω	1 ea.
500 Ω10W	1 ea.
500 Ω Trim Pot 1W	2 ea.
1000 ΩTrim Pot 1W	2 ea.

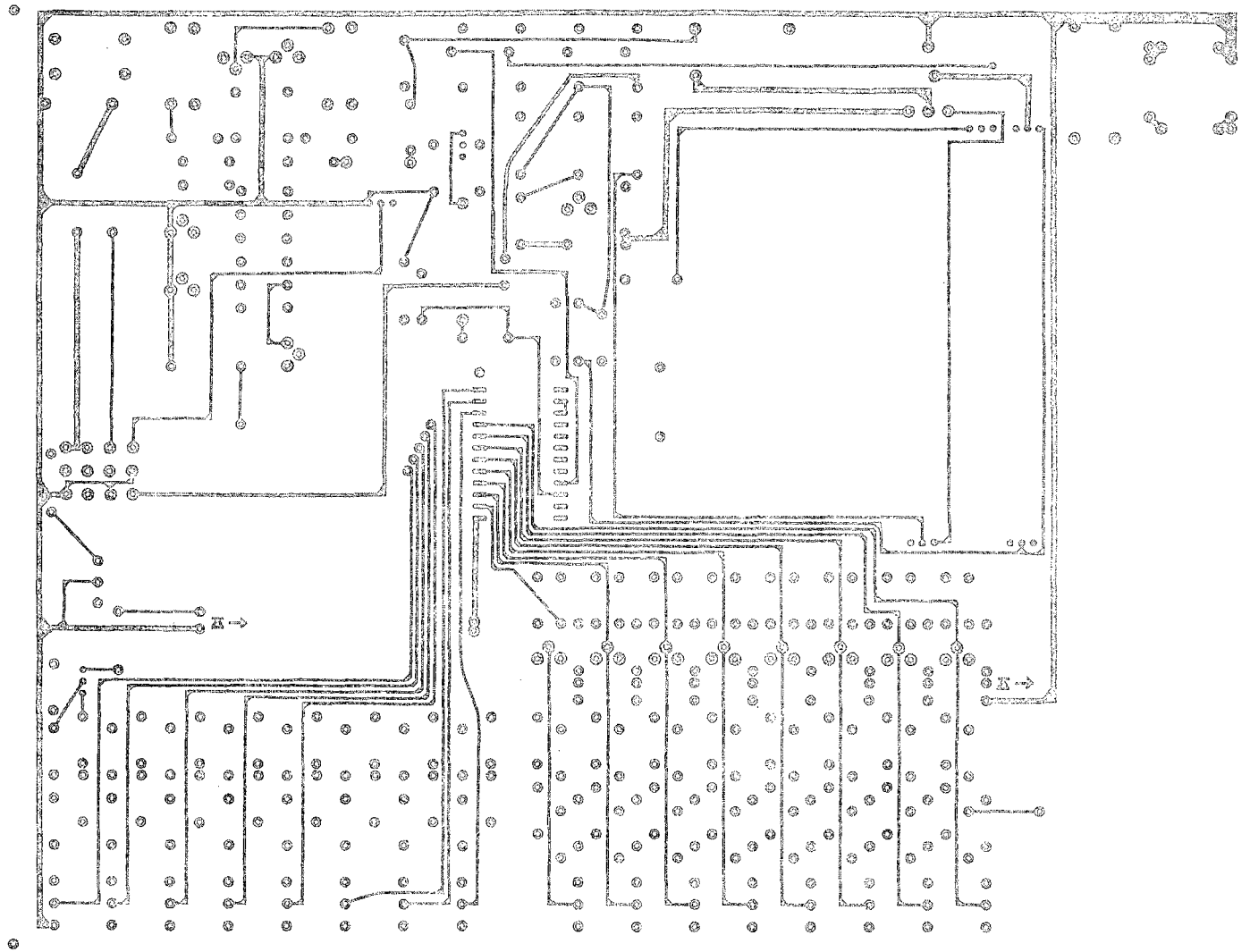
NOTE: All Resistors are 1/4 watt 10% unless marked otherwise

Capacitors	Quantity
10μf 100V	1 ea.
3.3μf 15V	2 ea.
1.0μf 35V	1 ea.
200 pF Mica	1 ea.
100 pF Mica	1 ea.

Semiconductors	Quantity
IN914	43 ea.
IN3194	1 ea.
IN5233 6.0V Zener	1 ea.
IN5242 12.0V Zener	1 ea.
IN5256 30.0V Zener	1 ea.
MJE1092	1 ea.
MPS-U57	4 ea.
MPS-U07	1 ea.
2N4920	1 ea.
2N4923	1 ea.
2N4917	3 ea.
2N3073	1 ea.
2N3722	1 ea.
MV-10B L.E.D.	9 ea.
2N3642	8 ea.
2N3638	1 ea.

Switches	Quantity
SPDT Toggle	16 ea.
4PDT Toggle	1 ea.
SPST Push Button N.C.	1 ea.

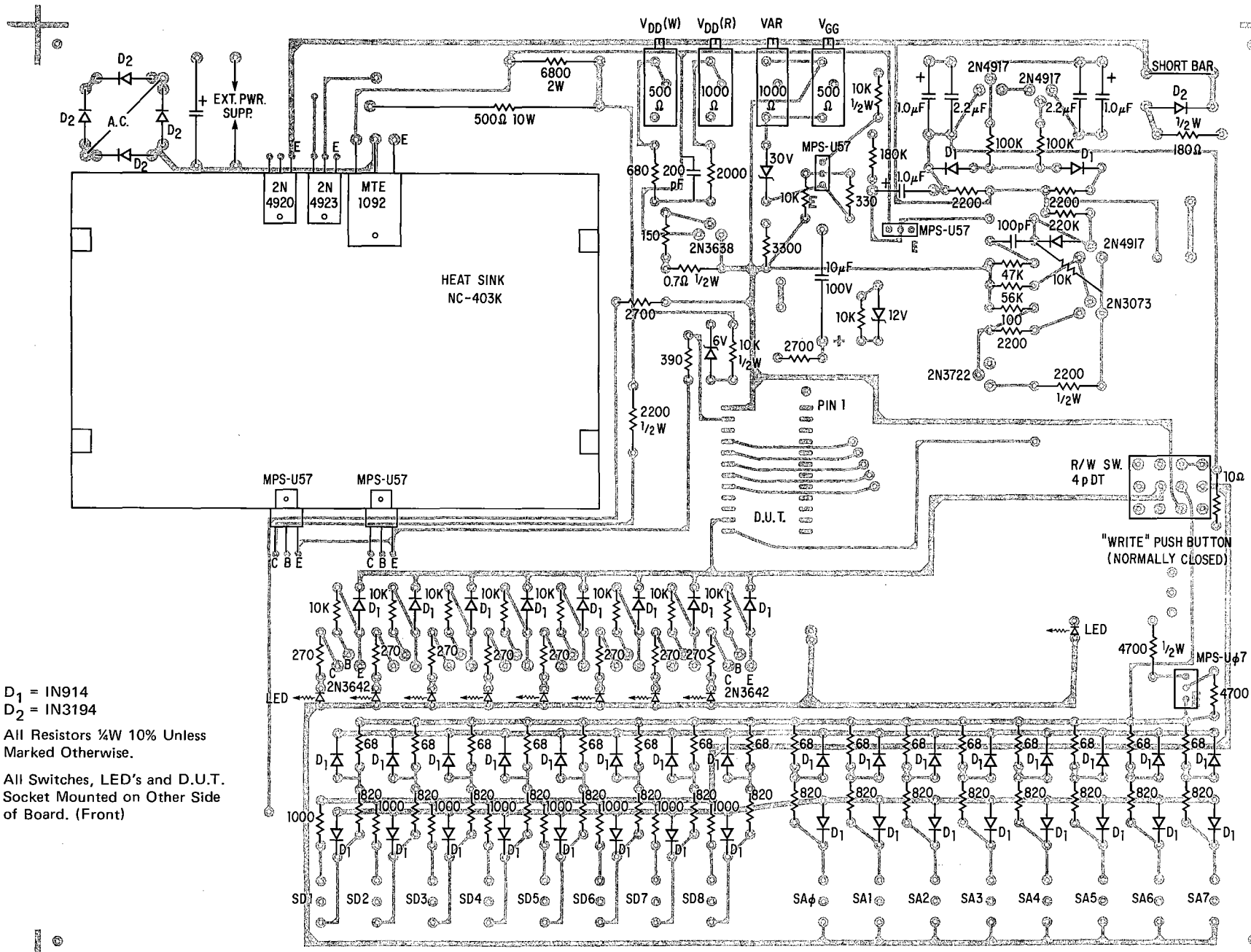
Miscellaneous	Quantity
Heat Sink NC403K	1 ea.
24 pin test socket	1 ea.
60-80V 1 amp Power Supply	1 ea.
P.C. Board (optional)	1 ea.



FRONT OF MANUAL PROGRAMMING ROM PC BOARD

SCALE: 65% OF ACTUAL BOARD SIZE

NOTE:
Only Switches, LED's and D.U.T.
Socket Mounted on This Side.



D_1 = IN914
D_2 = IN3194

All Resistors ¼W 10% Unless Marked Otherwise.

All Switches, LED's and D.U.T. Socket Mounted on Other Side of Board. (Front)

BACK OF MANUAL PROGRAMMING ROM PC BOARD
SCALE: 80% OF ACTUAL BOARD SIZE

Programming Instructions For Manual Programmer

1. Insert the device into the socket.
2. Turn on power.
3. Set the read/write, 4PDT switch ($S1_A$ through $S1_D$) to the W(Write) position.
4. Check for shorts in the data and address input pins by depressing the "write" push button and monitoring shorts indicator. If the indicator is flashing, attempts at writing should be discontinued until the problem is located. The first thing to be checked is to insure that the device has been correctly inserted in the socket.
5. Set the address inputs (toggle switches S_{A0} through S_{A7}) to the desired address (generally starting with address 00000000). The correct position for Logic "0" and Logic "1" is shown in the schematic.
6. Set the data inputs (toggle switches S_{D1} through S_{D7}) to the desired inputs for the selected address. The correct position for the Logic "0" and Logic "1" for the data input switches is shown in the schematic.
7. Then press the write push button and hold it for at least 5 seconds. Data has now been written.
8. To verify the data, set the read/write 4PDT switch to the R(Read) position. The data that has been written into the selected address will then be displayed on the LED's. A Logic "1" will be represented by a lit LED. A Logic "0" will be represented by a blank LED.
9. To write the next word, set the read/write, 4PDT switch to the W position and repeat steps 5 through 8.
10. The remaining words are then written into the device following the outlined sequence until the device is completely written into.

III. PROGRAMMING OF 1601/1701 AND 1602/1702 USING INTEL 7600 PROGRAMMER.

The 1601/1701 and 1602/1702 have been designed to facilitate rapid turnaround of custom patterns. Patterns supplied on paper tape are electrically programmed into the ROM by the 7600 programmer. Programmers are located at Intel, major distributors, and at many of our representatives in the U.S., Europe, and Japan. Programmers will also be available for sale to customers.

Programmer Description

The paper tape containing the custom pattern is loaded into the programmer which verifies the format and length of the data field. After loading, the programmer will write the data from the paper tape into the ROM (1601, 1701, 1602, or 1702).

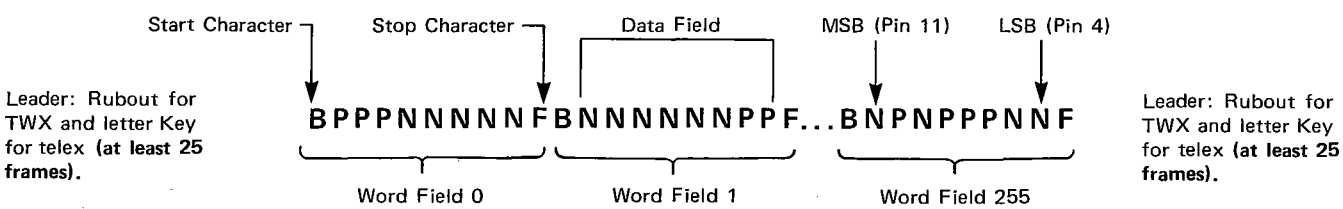
To insure that the ROM has been properly programmed, it is read out for every address and compared with the data of the paper tape. Reading is done in both the dynamic and static mode. An error will stop the read cycle at the bad word location. The displays on the programmer will indicate the bad word location and the 8 output bits of the ROM. For comparison the 8 input bits are also displayed.

A 1601, 1701, 1602, or 1702 is loaded again into the programmer and the programming cycle is repeated.

Tape Format

The 7600 programmer accepts 1" wide paper tape using 7 or 8 bit ASCII code, such as a model 33 ASR teletype produces. These programmers can also be used with the narrower 5 bit Telex tape. For such a tape to correctly program a 1601/1701 or 1602/1702, it must follow exactly the format rules below.

The format required by the 7600 is as shown below:



NOTE: Intel cannot assume responsibility for the programming circuit described in this data sheet.

ERRATA SHEET

(To replace Programming Instructions for Manual Programmer given on the top of Page 18 of the 1601/1701, 1602/1702, 1301 Data Sheet)

PROGRAMMING INSTRUCTIONS FOR MANUAL PROGRAMMER

1. Insert the device into the socket.
2. Turn on power
3. Set the read/write, 4PDT switch ($S1_A$ through $S1_D$) to the W (Write) position.
4. Set the address inputs (toggle switches S_{A0} through S_{A7}) to the desired address (generally starting with address 00000000). The correct position for Logic "0" and Logic "1" is shown in the schematic.
5. Set the data inputs (toggle switches S_{D1} through S_{D7}) to the desired inputs for the selected address. The correct position for the Logic "0" and Logic "1" for the data input switches is shown in the schematic.
6. Then press the write push button and hold it for at least 5 seconds. Data has now been written.

   Note: If the shorts indicator begins to flash, writing should be discontinued until the problem is located. The first thing to be checked is to insure that the device has been correctly inserted in the socket.

7. To verify the data, set the read/write 4PDT switch to the R (Read) position. The data that has been written into the selected address will then be displayed on the LED's. A logic "1" will be represented by a lit LED. A Logic "0" will be represented by a blank LED.

8. To write the next word, set the read/write, 4PDT switch to the W position and repeat steps 5 through 8.

9. The remaining words are then written into the device following the outlined sequence until the device is completely written into.

The format requirements are as follows:

1. There must be exactly 256 word fields in consecutive sequence, starting with word field 0 (all address lines low — refer to the table shown on page 10).

2. Each word field must consist of 10 consecutive characters, the first of which must be the start character B. Following the start character, there must be exactly 8 data characters (P's or N's) and ending with the stop character F. NO OTHER CHARACTERS, SUCH AS RUBOUTS, ARE ALLOWED ANYWHERE IN A WORD FIELD. If in preparing a tape, an error is made, the entire word field, including the B and F, must be rubbed out. Within the word field, a P results in a high level output, an N results in a low level output. The first data character corresponds to the desired output for data bit 8 (pin 11), the second for data bit 7 (pin 10), etc.

3. Preceding the first word field and following the last word field, there must be a leader/trailer length of at least 25 characters. This should consist of rubout punches (letter key for Telex tapes).

4. Between word fields, comments **not containing B's or F's** may be inserted. It is recommended that carriage return and line feed characters be inserted (as a "comment") just before each word field or at least between every four word fields. When these carriage returns, etc., are inserted, the tape may be easily listed on the teletype for purposes of error checking. The customer may also find it helpful to insert the word number (as a comment) at least every four word fields.

5. Included in the tape before the leader should be the customers complete Telex or TWX number and if more than one pattern is being transmitted, the ROM pattern number.

INTEL Telex No.	346372
INTEL TWX No.	910-338-0026
Japan's Telex No.	26364
Europe Telex No.	21060

IV. PARALLEL PROGRAMMING OF 1601/1701 AND 1602/1702.

When programming several ROMs in parallel, $\overline{CS}$ and V_{CC} should be at 0V for all 1601/1701 or 1602/1702 while the program pulse is applied only to the ROM being programmed. Pulsed V_{DD} and V_{GG} may be applied to all ROMs (both selected and unselected chips). However, if V_{DD} and V_{GG} are individually decoded, so that V_{DD} and V_{GG} are applied only to the ROM being programmed, multiplexing techniques may be used to permit programming an entire array in the same time that one part is programmed.

V. MASK PROGRAMMING OF 1301.

Tape Format

The custom patterns may be sent in on a Telex or submitted as a paper tape in a 7 or 8 bit ASCII code from model 33 teletype or TWX. The paper tape format is exactly the same as for the 1601/1701 and 1602/1702.

Truth Table

The custom patterns may be sent in on a truth table. Blank custom truth table forms are available upon request from Intel.

VI. 1701, 1702 ERASING PROCEDURE.

The 1701 and 1702 may be erased by exposure to a high intensity ultraviolet light source. A 1701 or 1702 placed 1 to 1.5 inches from a light source with an ultraviolet wavelength of 2537Å at an intensity of 10 mW/cm^2 (i.e. a dosage of 6W sec/cm^2) will be erased in 10 minutes. An example of a light source which is capable of producing the required ultraviolet wavelength and intensity is the Model R51 manufactured by Ultraviolet Products (San Gabriel, California).

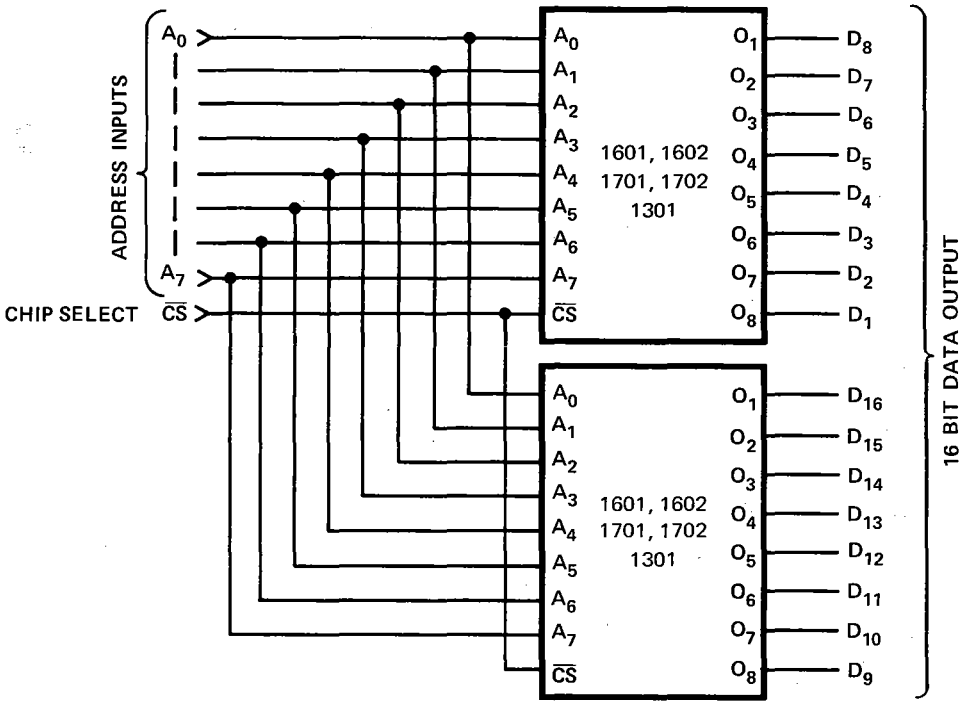
Application Information

POWER DISSIPATION CONSIDERATIONS

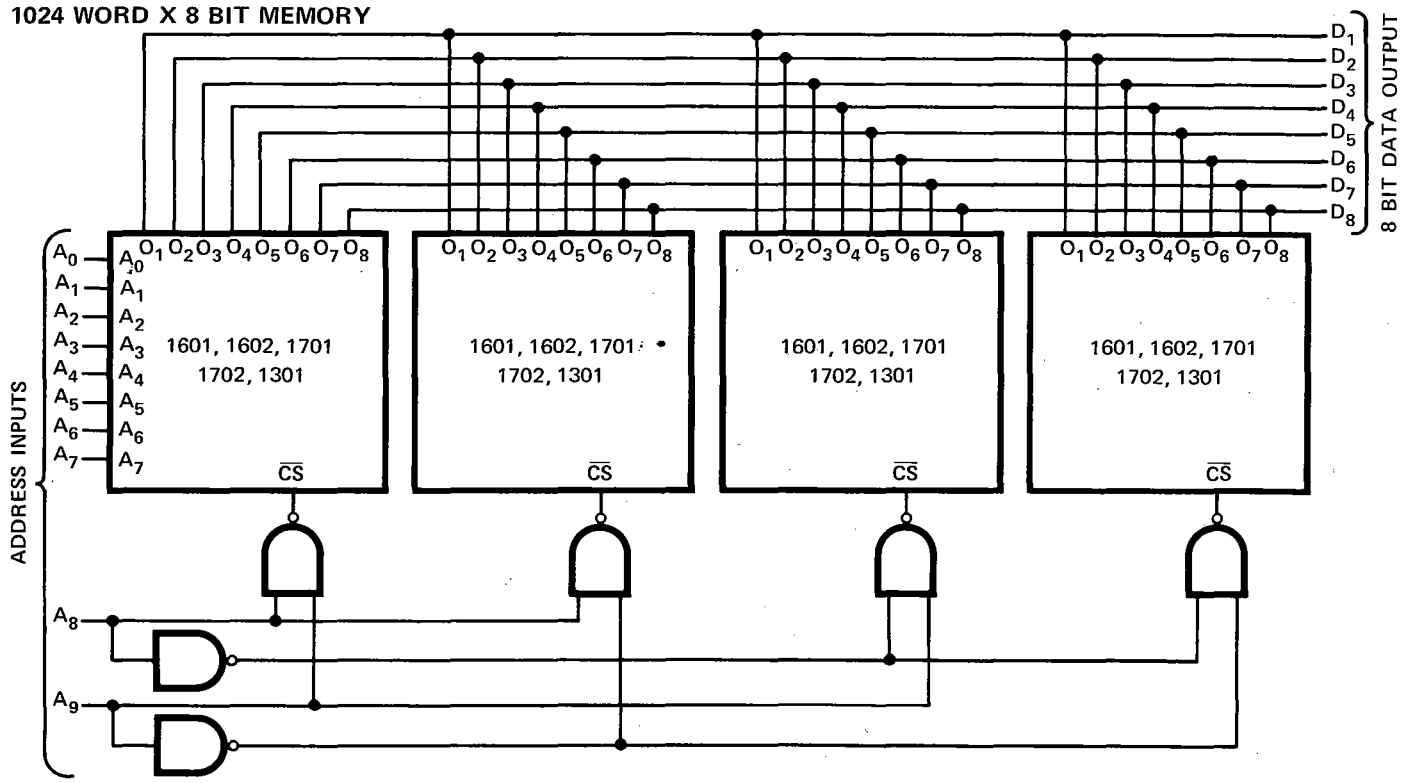
Peak power dissipation in the dynamic mode occurs when the ϕ_1 clock is low. At other times, two sources of power dissipation must be considered: A fixed current flow (from V_{CC} to V_{DD}) which corresponds to I_{DDO}, and the current flowing through the output terminals. These output currents continue to flow after the end of the cycle. To prevent these currents from flowing when the memory is inactive (yet has V_{DD} applied), the chip must be deselected before the clocks are deactivated. Deselection can be accomplished only by executing a memory cycle with $\overline{CS}$ in the deselect condition.

256 WORD BY 16 BIT ROM

In this example the 8 bit address bus A_0 - A_7 and $\overline{CS}$ lines are connected in parallel. A full 16 bit word is made up of 8 bits from each 1601/1701, 1602/1702 and 1301 package.



1024 WORD X 8 BIT MEMORY



INTEL CORP. 3065 Bowers Avenue, Santa Clara, California 95051 • (408) 246-7501

Printed in U.S.A.

7136/1