ME5204

MM4204 and MM5204 EPROM Programmer

By Martin Eberhard

ME5204

Martin Eberhard's

MM4204 & MM5204 EPROM Programmer

Rev A and B PC Boards, Rev 2.01 Firmware

Welcome

The ME5204 programmer is designed to program one of the more obscure EPROMs ever made. National Semiconductor's MM5204Q EPROM was only made for a few years, and no other manufacturer made a second-source equivalent part. Its pinout and programming voltages are unique, making it unsupportable by most 'universal' programmers. It was not used in many designs, though it was used in a few interesting products, such as early Sol-20 computers from Processor Technology. (If you bought this programmer, then you already know this.)

The ME5204 Programmer is a full-function EPROM programmer designed to program and read MM4204 and MM5204 PMOS EPROMs. The programmer connects to a computer with an RS-232C serial port at 9600 baud, requiring a terminal program that can send and receive text files. You can transfer files to and from the ME5204 Programmer in either Intel Hex format or Motorola S-record format. You can also edit the EPROM file while it is in the ME5204's buffer.

The revision A PC board requires a few simple rework steps. These changes were incorporated into the Rev B PC board.

A few words of caution: Be respectful of the voltages within the MM5204. Obviously, the incoming 120V or 240V are dangerous. Also be aware that there are 60V on the EPROM's pins while it is being programmed, 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 ME5204 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 ME5204 Programmer is delivered as a kit that must be assembled and adjusted before use. Simple electronics assembly skills and tools are required to assemble the printed circuit 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 ME5204. Have fun assembling!

-Martin Eberhard
21 October 2018

ME5204 Revision History

PCB	Firmware	Date	Change Notes
A	1.01	28 Dec 2011	Created, based on ME1702/A firmware version 1.07
A	2.00	15 Jan 2013	Rev 2.00 firmware supports firmware downloading via the serial port, in conjunction with ME Loader Kernel 1.00
A	2.01	18 Aug 2015	Add automatic address offset (AO command with no parameter). Improve some messages by adding ‘range’ when working with only a portion of the EPROM or buffer.
A & B	2.01	25 Sep 2018	Add R101 rework, Support Rev B boards, update Digikey part numbers, etc.

Contents

Section 1. ME5204 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 is not 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.

Also pay attention to the orientation of the two 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 ten TO-92 devices that are NOT 293904 transistors. These devices are installed first, to minimize the chance of the wrong device being installed.

All unlabeled 1/4W resistors are 3K-ohm, 5% resistors. Most of the rest are labeled to help you get the right resistors in the right locations.

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 socket is flush to the PC board.

If you are using a Rev A PC board then you will need to make one trace cut and install four rework components on the solder-side of the board. Make sure the cut trace is cut completely. Install the four components so that they are flush to the PC board, anfd their leads cannot accidentally touch adjacent component pins. (This change has been incorporated into the Rev B PC board.)

Four components (the ZIF socket and the 3 LEDs) are installed on the solder-side of the board, so that they will protrude through the Enclosure when assembled. 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. (Rework for Rev A PC Boards ONLY)** Cut the wide solder-side trace that goes from the '+' pin of C13 to pin 1 of V4 (the 7805). I recommend making two cuts about 1/8" apart. Then heat up the trace between the cuts with your soldering iron, and slide the little piece of trace off the board.

Step 2. Install the following 1/4 W, 1% resistors.

√	Qty	Locations	Value	Digikey Part Number
	3	R29,R30,R48	324 1%	324XBK-ND
	1	R27	976 1%	976XBK-ND
	2	R22,R26	2.1K 1%	RNF14FTD2K10CT-ND
	1	R47	2.8K 1%	2.80KXBK-ND
	1	R23	3.09K 1%	RNF14FTD3K09CT-ND
	3	R64,R83,R85	10K 1%	10.0KXBK-ND
	2	R82,R84	49.9K 1%	49.9KXBK-ND
	1	R63	78.7K 1%	78.7KXBK-ND

Step 3. Install the following 1/4 W, 5% resistors. (**Rev A PC Boards:** Note that R43, R50, and R66 are not labeled '1K' on the silkscreen. Also R41 is not labeled '12K'.)

√	Qty	Locations	Value	Digikey Part Number
	8	R5,R7,R9,R11,R13,R15,R17,R19	100 5%	S100QCT-ND
	2	R60,R70	330 5%	CF14JT330RCT-ND
	1	R25	910 5%	CF14JT910RCT-ND
	5	R43,R50,R59,R66,R78	1K 5%	CF14JT1K00CT-ND
	1	R24	5.6K 5%	CF14JT5K60CT-ND
	4	R1,R41,R44,R49	12K 5%	12KQBK-ND
	8	R51-R58	47K 5%	47KQBK-ND

Rev B PC Boards Only:

√	Qty	Locations	Value	Digikey Part Number
	1	R99	12K 5%	12KQBK-ND
	1	R100	1K 5%	CF14JT1K00CT-ND

Step 4. Install 3K-ohm, 5% resistors in the 55 remaining 1/4 W resistor locations. Note that two more 3K-ohm resistors will be installed later, as part of rework.

√	Qty	Locations	Value	Digikey Part No.
	55	R2-R4,R6,R8,R10,R12,R14,R16, R18,R20,R21,R28,R31-R40,R42,R45, R46,R61,R62,R65,R67-R69,R71-R77, R79-R81,R86-R98	3K 5%	3.0KQBK-ND

Step 5. Install the following diodes. **Be very careful** about orientation and also **be very careful** to put the correct diode in each location. It's a good idea to bend the leads such that the last 2 or 3 digits of the diode number will be readable when the diodes are soldered in place.

√	Qty	Locations	Value	Digikey Part Number
	3	D1-D3	Diode, 1N4148	1N4148TACT-ND
	4	D4-D7	Diode, 1N4004	1N4004-TPMSCT-ND

Rev B PC Boards Only:

√	Qty	Location	Value	Digikey Part Number
	1	Z1	Diode, 12V Zener, 1N4742A	1727-1946-1-ND

Step 6. Install 2 DIP sockets flush to the PC board, paying attention to orientation:

√	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 capacitors:

√	Qty	Locations	Value	Digikey Part Number
	6	C1,C2,C4,C5,C15,C16	0.1 uF, 100V	478-4855-ND
	8	C3,C6-C11,C14	1 uF, 25V	445-173583-1-ND

Rev B PC Boards Only:

√	Qty	Locations	Value	Digikey Part Number
	1	C17	0.001 uF, 100V	BC5112-ND

Step 8. Install the TO-220 package voltage regulators. Bend their leads such that they lie flat against the board and their mounting holes align with the PC board holes, and then solder them in place. Screw the two regulators down to the PC board, with the nuts on the regulators.

√	Qty	Locations	Value	Digikey Part Number
	1	V2	LM317	LM317MTGOS-ND
	1	V3	LM337	296-21577-5-ND
	2	V2, V3	Pan-head screw, 6-32,5/16"	H115-ND
	2	V2, V3	Nut, #6	H220-ND

Step 9. 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 stand straight, and have about 1/8 inch of lead between the PC board and their plastic bodies.

√	Qty	Locations	Value	Digikey Part Number
	1	V1	LM317L	LM317LZXTR-ND
	2	Q19,Q23	MPSA56	MPSA56-APMSCT-ND
	Alternate	Q19,Q23	MPSA55	MPSA55GOS-ND
	3	Q1,Q8,Q20	2N3906	2N3906-APCT-ND
	4	Q2,Q6,Q9,Q22	MPSA05	MPSA05-APMSCT-ND

Step 10. Install 30 2N3904 transistors in the following locations. Double-check their orientation. When installed, these components should stand straight, and have about 1/8 inch of lead between the PC board and their plastic bodies.

√	Qty	Locations	Value	Digikey Part Number
	30	Q3-Q5,Q7,Q10-Q18,Q21,Q24-Q39	2N3904	2N3904-APCT-ND

Step 11. Install 2 trim-pots in the following locations, and set them to the center of their ranges.

√	Qty	Locations	Value	Digikey Part Number
	1	VR2	1K ohm Trimpot	262UR102B-ND
	1	VR1	250 ohm Trimpot	201UR251B-ND

Step 12. Install two fuse-clips each at FS1 and FS2. 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
	4	FS1, FS2	Fuse clip	F4186-ND
	0	FS1, FS2	Alternate fuse clip	BK-6005-ND

Step 13. Install the two 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	C7,C8	470 uF 63V	493-1127-ND

Step 14. Install the LM7805 (TO-220) voltage regulator. It should first be attached to a heat sink loosely with a screw and a nut, with the nut against the TO-220 device. (Put a tiny bit of heat sink grease on the mating surface of the TO-220 device if you have some.) Then insert the assembly into the PC board and solder it in place, including soldering the heat sink’s pin. Finally, tighten the screw and nut thoroughly.

√	Qty	Loc.	Value	Digikey Part Number
	1	V4	Vertical TO-220 heat sink	HS368-ND
	1	V4	Pan-head screw, 6-32,5/16"	H115-ND
	1	V4	Nut, #6	H220-ND
	1	V4	LM7805	MC7805CT-BPMS-ND

Step 15. **(Rework: Rev A and Rev B PC Boards)** Install the following component on the solder-side of the PC Board. Keep it flush to the board, and be sure that the component leads cannot accidentally short to any adjacent pins.

√	Value	Ref.	Digikey No.	From	To
	47K ohm Resistor	R101	47KQBK-ND	R46 end closest to Q21	R47 end closest to Q21

Step 16. **(Rework: Rev A PC Board ONLY)** Install the following four components on the solder-side of the PC board. Keep them flush to the board, and be sure that the component leads cannot accidentally short to any adjacent pins. Make sure the zener diode is installed in the correct orientation.

√	Value	Ref.	Digikey No.	From	To
	1N4742A Zener Diode	Z1	1727-1946-1-ND	Anode (no stripe) to V4 pin 1	Cathode (stripe) to C13 '+' end
	3K ohm Resistor	R99	3.0KQBK-ND	V4 pin 1	V4 heatsink pin
	0.001 uF capacitor	C17	BC5112-ND	Q4 emitter	Q4 collector
	3K ohm Resistor	R100	3.0KQBK-ND	C1 pin farthest from board edge	R26 pin closest to board edge

Step 17. Install two 0.1" headers in the following locations. Be sure to seat them all the way against the PC board. (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	GND	1-pin header	609-3466-ND
	1	J2	6-pin 0,1" header	A31116-ND

Step 18. 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 farthest from 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

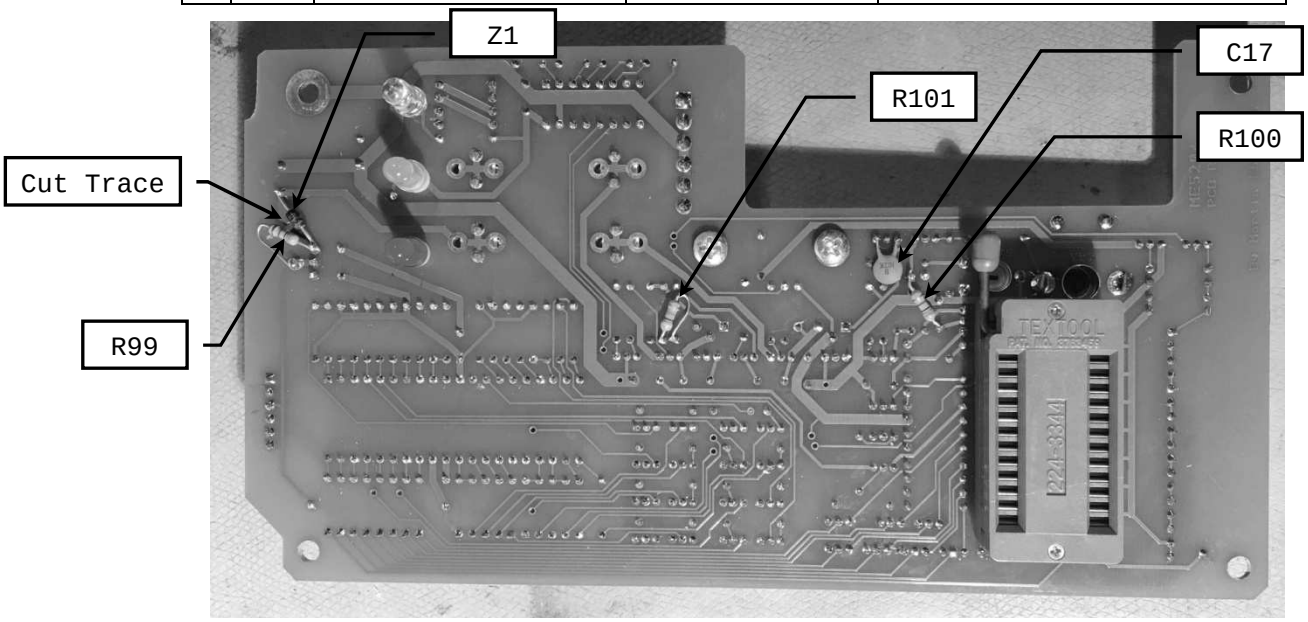
Step 19. **REVERSE-MOUNTED COMPONENT**: Install the Textool ZIF socket **on the solder-side of the PC board.** Install the socket with its handle toward the cut-out side of the PC board - the handle should be closest to the square pin-1 pad. It is **very important** to open the socket (handle perpendicular 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 24-pin ZIF	3M2402-ND

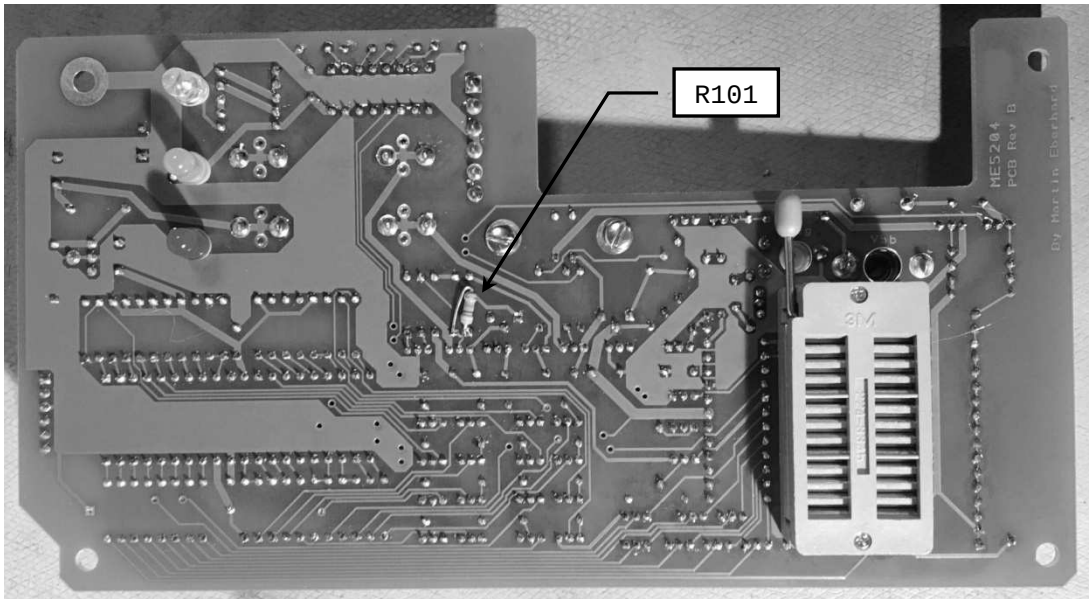
Step 20. **REVERSE-MOUNTED COMPONENTS**: Install the following 3 LEDs **on the solder-side of the PC board.** (Optionally, slide a stand-off onto each LED before inserting the LED in the PC board.) Pay attention to the LED orientation - the shorter lead on the LED (closest to the flat side of the LED's 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	LED standoff	8311K-ND
	1	LED1 **Solder Side**	Blue T1 LED	C503B-BCS-CV0Z0461-ND
	1	LED2 **Solder Side**	Green T1 LED	160-1130-ND
	1	LED3 **Solder Side**	Red T1 LED	160-1127-ND



Four Reverse-Mounted Components and five rework components installed on the solder-side of a rev A PC board



Four Reverse-Mounted Components and one rework component installed on the solder-side of a rev B PC board

Step 21. Insert two 3/4-amp fast-blow fuses in the fuse clips at FS1 and FS2. (These fuses help 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
	2	FS1, FS2	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	Digikey Part Number
	1	HM244-ND	1955-Series Instr. 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 ME5204, 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 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 3. 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 4. 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 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 5. 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 the following components, and three 9-inch pieces of AWG 20 or AWG 22 stranded wire (or 9 inches of 3-strand ribbon cable) to build the RS-232 Cable Subassembly:

√	Qty	Manufacturer/ Part No.	Description	Digikey Part No.
	1	AMP/09-50-3061	6-pos. 0.156″ conn. housing	WM2104-ND
	3	AMP/08-52-0072	AWG18-AWG24 contact	WM2302-ND
	1	3M/8R09-N001	9-pin female DSUB (DA-9)	3M10608-ND

If you do not use a real crimping tool for the AMP contacts, 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	AMP pin	Signal
	3	2	Data In (into the ME5204)
	2	1	Data Out (out of the ME5204)
	5	3	Ground

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 fuse, hidden inside a fuse drawer
4. The fuse drawer reverses to select either 120V or 240V operation

These four components are not electrically connected together - you must connect them with wires. Use AWG 16 or AWG 18 wire to construct the Inlet Subassembly, using the following parts:

√	Qty	Manufacturer/ Part No.	Description	Digikey Part Number
	1	Qualtek/ 765-00/001	Multifunction Inlet	Q306-ND
	1	TE Connectivity/ 61793-1	#6 Ring terminal	A29900CT-ND

Connect the Multifunction Inlet’s terminals together as follows. When you route a wire from one side of the inlet to the other, go the long way around its back (not the short way around its side) or you will not be able to insert the inlet into the hole you made in the box. Note that some of the wires have 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.

Some of the Multifunction Inlets do not have the terminals labeled clearly. When in doubt, refer to the Chassis Wiring Schematic in Section 9.3, and double-check with an ohm meter.

√	Inlet Pin	Inlet Pin	And also To	Wire Color	Length
	N	2A	--	White	As needed
	L	B	--	Black	As needed
	A	1B	--	Black	As needed
	1A	4	Free wire end	Black	8″
	2B	1	Free wire end	White	8″
	2	--	Free wire end	Brown	8″
	3	--	Free wire end	Orange	8″
	G		Ring Terminal	Green	3″

Step 3. Configure the Multifunction Inlet

√	Qty	Description	Digikey Part Number
	1	3/4A 1-1/4″ fast fuse	283-2631-ND

Pry the fuse drawer from the Multifunction Inlet with a screwdriver. Install the 3/4A fuse in the side that has the 110-120V arrow pointing to it. (You will probably need to remove and discard a small black plastic cover first - the inlet is designed for two different fuse sizes, and this cover is used for the shorter fuses.) Insert the drawer back in the inlet, such that the triangular pointer near the power cord inlet points to the 110V-120V arrow. (For 240 volts, put the fuse in the other side and install the drawer the other way.)

Step 4. Assemble the Components into the Chassis

√	Qty	Description	Digikey Part Number
	1	Inlet Subassembly	from Step 2 above
	2	3/8" 4-40 flat-head screw	501-1531-ND
	2	3/8" 4-40 pan-head screw	36-9901-ND
	4	4-40 nut	H220-ND
	2	3/8" 6-32 pan-head screw	
	2	#6 lock washer	H240-ND
	2	6-32 nut	H220-ND
	4	3/8" 4-40 self-tapping screw	
	3	#4 flat washer	5205820-3-ND
	1	0.25A 56V center-tapped Transformer, 120V/240V input	HM4690-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. If any of the plastic ribs on the inside of the box are in the way of the nuts, trim them off with a sharp knife.

☐ **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.

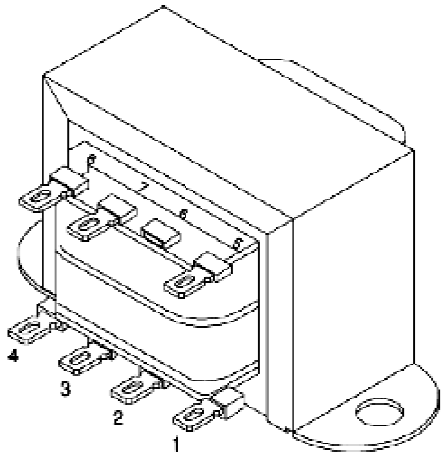
☐ **Step 4D:** Screw the transformer in place in the rear panel of the box, with its contacts toward the box opening. Use two 6-32 screws, two #6 lock washers, and two 6-32 nuts. If any of the ribs are in the way of the washers, trim them off with a sharp knife.

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.

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.



√	From Inlet Subassembly	T1 pin
	White wire	1
	Orange wire	2
	Brown wire	3
	Black wire	4

Step 6. Build the Isolated Wiring Harness

Use the following components, and some AWG 18 wire to create the isolated wiring harness, using the three unused locations in the 6-pin connector on the RS-232 harness.

√	Qty	AMP Part No.	Description	Digikey Part No.
	3	08-52-0072	AWG18-AWG24 contact	WM2302-ND

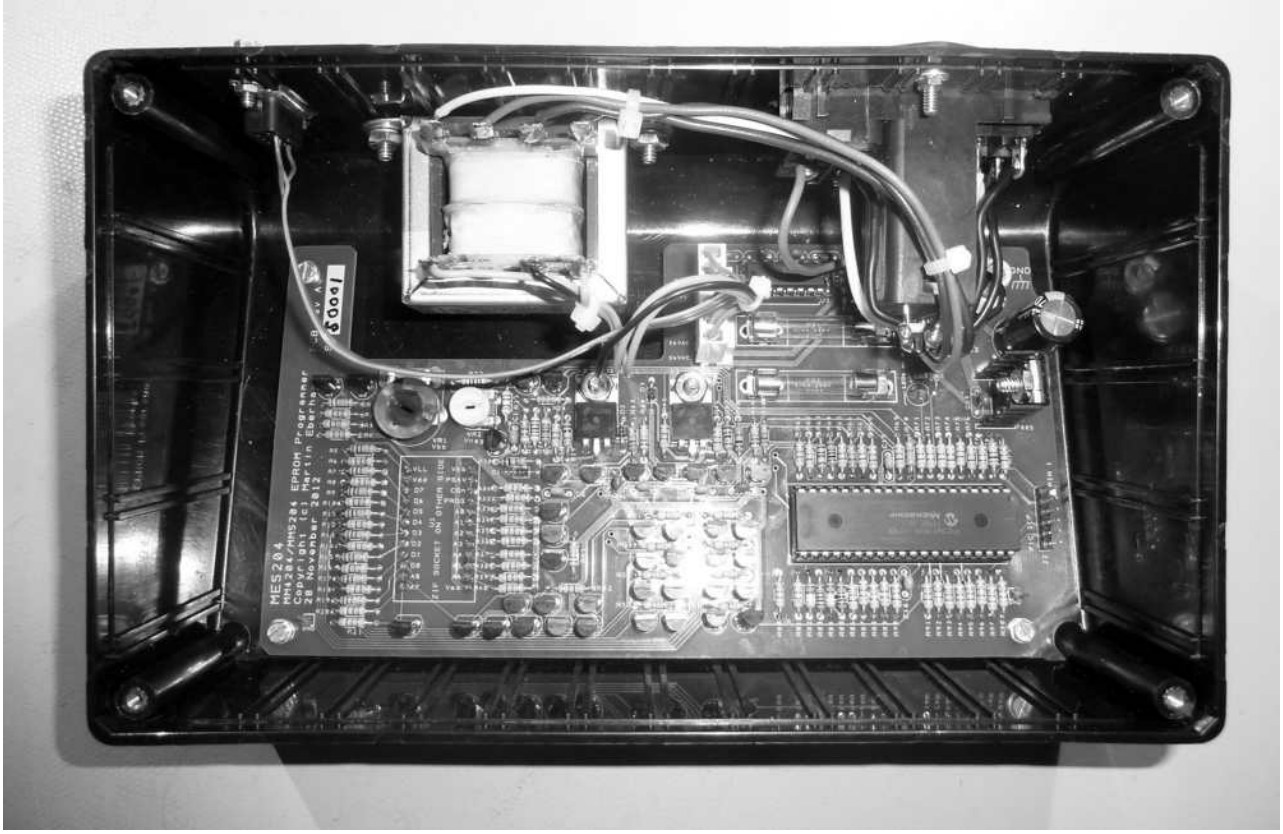
The AMP connector will plug into J1 on the PCBA. Make your wires long enough to reach comfortably, but not overly-long. Look at the PC board to determine which end is pin 1 - it is the farthest from the back panel. Double-check the wiring against the labels on the PC board.

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

√	From J1 pin	To T1 pin	Wire Color
	4	7	Black
	5	5	Yellow
	6	8	Yellow

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.

Double-check that you put the harness wires into the correct connector locations. Make sure the harness wires match the labels on the PC board silkscreen. (If you get this wrong, you will probably damage the board when you power it up!)



Inside of completed chassis (Rev A board shown)

Step 7. Inspect your work!

☐ Check for correct wiring, stray wire strands, loose crimps, etc.

Section 2. Checkout and Adjustment

Basic checkout requires a voltmeter and either a computer terminal (such as the most excellent Wyse WY-30[1]) 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 AC voltage between pins N and L of the Inlet. If they are not at line voltage, then your power cord is installed incorrectly or defective.

☐ **Step 3.** Measure AC voltage between pins 1B and 2A of the Multifunction Inlet (on the switch). If they are not line voltage, then check the location of the fuse in the fuse drawer. If this is correct, check your wiring.

☐ **Step 4.** Measure the AC voltage between 1A and 2B (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 line voltage across 1A and 2B.

☐ **Step 5.** Measure between pins 1 and 4 of the transformer, again looking for line voltage. Check your wiring if you do not see this.

☐ **Step 6.** Measure between pins 1 and 2 of the transformer. If you do not see line voltage, then you either have the fuse drawer installed backwards, or you have a wiring mistake. (Note: move the fuse to the other position if you turn the drawer over.) Measure also between pins 3 and 4 of the transformer. Again, you should see line voltage.

Step 7. Measure the following approximate voltages between pins of the 6-pin connector on the wiring harness. Correct defects before moving on.

√	From	To	AC Voltage
	5	6	56 VAC (may be as high as 68VAC)
	4	5	28 VAC (may be as high as 34VAC)
	4	6	28 VAC (may be as high as 34VAC)

2.2 Basic PCBA Checkout

☐ **Step 8.** Turn the power off, and plug the harness connector into J1. Hook the ground lead of your voltmeter to the GND pin near the lower-left corner of the PCBA. Turn the power on. The blue LED should light.

☐ **Step 9.** Measure the unregulated supply for the 5V regulator:

√	Measure	Measurement	Meaning
	V4 pin 1	24V to 34V	Correct operation
▨		>34V	Z1 (rework diode) installed backwards or wrong diode type?
		<24V	Short circuit in rework? Short in +5V circuit?

☐ **Step 10.** Measure DC voltage at U2 pins 11 and 32. Both should be +5V +/-0.25V. If not, power off and debug the logic supply.

[1] The Wyse Technology WY-30 was the first product that I designed professionally.

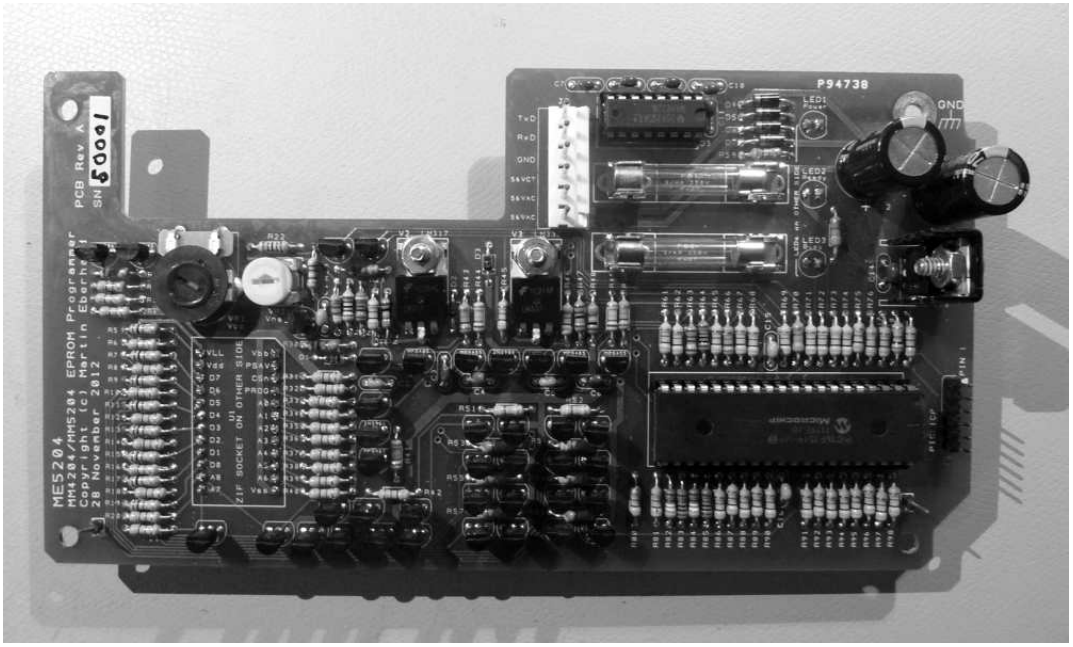
☐ **Step 11.** Measure DC voltage at both ends of the fuse FS1. You should see around -44V at both ends. If this voltage is only at one end, then the fuse has blown, probably because of an assembly problem on the board. Power down and debug as needed.

☐ **Step 12.** Measure the voltage at both ends of the fuse FS2. You should see around +40V at both ends. If this voltage is only at one end, then the fuse has blown, probably because of an assembly problem on the board. Power down and debug as needed.

Step 13. Power down and unplug the ME5204 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	Programmed PIC16F1519 microcontroller	PIC16F1519-I/P-ND (With ME5204 firmware)

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 ME5204 Programming Firmware, which can be loaded via the serial port - see section 8. If you are using the PIC that I supplied, then it has already been programmed with both the loader and the programming firmware.



Completed Rev A PC Board, Component Side

2.3 Microcontroller Bring-Up

Step 1. Plug a terminal (or a PC with a terminal program) into the ME5204 Programmer's serial port connector, making sure to connect the transmit signal (TxD, pin 3) from the ME5204 to receive signal of the terminal and the receive signal (RxD, pin 2) from the ME5204 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 in the ME5204 Programmer and turn it on. On the terminal screen, you should see a sign-on banner and a prompt like this:

```
*====================================*
*               ME5204               *
*====================================*
*   MM4204/MM5204 EPROM Programmer   *
*         By Martin Eberhard         *
*       Firmware Version  2.01       *
*====================================*

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 ME5204 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 Microcontroller-Assisted Checkout and Adjustment

NOTE: The following steps involve dialog with the ME5204'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 ME5204 Programmer are off, then turn them back on.

Step 1. Test Master Power-Off

☐ Measure the voltage at pin 3 of V2. This should be close to 0V. If not, debug the circuit that includes Q6 and Q19.

☐ Measure the voltage at pin 2 of V3. This should be close to 0V. If not, debug the circuit that includes Q22 and Q23.

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 pin 3 of V2

√	Measurement	Meaning	Debug
	37V to 43V	Correct operation	--
	0V to 37V	Problem	• Correct component in V1 and V2? • Problem with V1 circuit or V2 circuit? • Problem with Q6 & Q19 circuit?

Measure the voltage at pin 2 of V3

√	Measurement	Meaning	Debug
	-37V to -44V	Correct operation	--
	0V to -37V	Problem	• Correct component in V3? • Problem with V3 circuit? • Problem with Q22 & Q23 circuit?

Step 3. Test Low-voltage Regulated Outputs

Measure the following voltages, and debug as needed:

√	Measure	Correct Value	Debug if incorrect
	U1 Vss	5.0V +/- 0.25V	V2, Q5 circuits. Check values: R26, R27, R29
	U1 Vbb	3.5V to 6.5V *	V1, Q4 circuits. Check values: R25, R30
	V3 pin 3	-12.0V +/- 0.6V	V3, Q21 circuits, check values: R47, R48
	U1 Vdd	4.75V +/- 0.5V	Q2 circuit
	U1 PROG	4.75V +/- 0.5V	Q9 circuit
	U1 CSn	4.75V +/- 0.5V	Q7 circuit

* We will adjust Vbb soon.

Step 4. Adjust Vbb. Note that the Vbb adjustment is relative to Vss.

√	Measure	Adjust	Correct voltage
	U1 Vbb	VR1	(Vss + 0.15V) +/-0.05V

Step 5. Test Chip Select Signal

Monitor dialog:

```
>TC 1
  -CS Signal On
>
```

☐ U1 CSn and U1 Powersaver (pin 2) should both be less than 0.5V. If not, debug Q7 circuit.

Step 6. Test Vdd

Monitor dialog:

```
>TD 1
         Vdd On
>
```

☐ U1 Vdd should now be within 0.3V of Vneg (which is -12V, measured at V3 pin 3). If not, debug Q9 circuit.

Step 7. Test the Address Drivers

Monitor dialog:

```
>WA 0
>
```

☐ Measure each address pin on U1. They should all be less than 0.3V.

Monitor dialog:

```
>WA 1FF
>
```

☐ 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 nine address driver transistors as needed.

Step 8. Test the Data Drivers

Monitor dialog:

```
>TC 0
  -CS Signal Off
>WD 0
>
```

☐ Measure each data pin on U1. They should all be less than 0.3V.

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 01
>RD
Data Read: 01
```

If you get any value besides FF, double check that you typed WD 01, and then debug the data input transistor circuits as needed.

☐ Write other values, including 02, 04, 08, 10, 20, 40, and 80, using the WD command. Each time, follow with an RD command. You should read the same value that you wrote each time. Debug as needed.

Monitor dialog:

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

☐ 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 1FF
>TP 0
-PROG Signal off
>TD 0
         Vdd off
>TH 1
High Voltage 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?

Step 11. Test and Adjust High-Voltage Power Supply Outputs

Measure the following voltages, adjust and debug as needed.

√	Measure	Measurement	Meaning
	U1 Vss (pin 12)	13V +/- 0.6V	Correct operation
		>13.6V	Check value of R26
		5.3V to 12.4V	
		4.75V to 5.3V	Debug Q5 circuit
		0 to 4.75V	Debug V2 circuit
	U1 Vbb (pin 1)	13V +/- 0.5V	Adjust VR1 so that Vbb = Vss + 0.10V
		5.3V to 12.5V	Check value of R22
		13.5V to 15V	
		4.75V to 5.3V	Debug Q4 circuit
		0 to 4.75V	Debug V1 circuit
		>15V	Debug Q3 circuit
	U1 Vdd (pin 23)	Within 0.3V of Vss	Correct operation
		otherwise	Debug Q2 circuit
	Vneg (V3 pin 3)	34V to 38V	Correct operation
		<-38V	Check value of R24
		-13.5 to -34V	
		-11V to -13,5V	Debug Q20, Q21 circuit
		<11V	Debug V3 circuit, look for shorts
	U1 PROG (pin 4)	Within 0.3V of Vss	Correct Operation
		Otherwise	Debug Q8, Q9 circuit
	U1 CSn (pin 3)	Within 0.3V of Vss	Correct Operation
		Otherwise	Debug Q7 circuit
	U1 A0-A8	Within 0.3V of Vss	Correct operation
	U1 D0-D7	Within 0.8V of Vss	Correct operation

Step 12. Test and Adjust Pulsed Power Signal Vdd

Monitor Dialog

```
>TD 1
   Vdd on
>
```

Measure the following voltages, adjust and debug as needed.

√	Measure	Measurement	Meaning
	U1 Vdd (pin 23)	-36V +/- 1V	Adjust VR2 for -36V +/- 0.1V
▨		Otherwise	Debug Q1, Q2 circuit

Step 13. Test Pulsed Power Signal Vbb

Monitor Dialog

```
>TB 1
   Vbb on
>
```

Measure the following voltages, debug as needed:

√	Measure	Measurement	Meaning
	U1 Vbb (pin 1)	25 +/- 0.5V	Correct operation
▨		>25.5V	Correct value for R23?
		13.5V to 24.5V	Correct Vbb adjustment in step 4?
		12.5V to 13.5V	Debug Q3 circuit
		<12.5V	Debug V1 circuit, look for shorts

Step 14. Turn off pulsed power supply voltages

Monitor Dialog

```
>TB 0
   Vbb off
>TD 0
   Vdd off
>
```

Step 15. 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 12.75V +/- 0.5V. Logic low should read between 0V and 0.3V.

Step 16. Reset When Done

☐ Monitor Dialog

```
>RE
*====================================*
*               ME5204               *
*====================================*
*   MM4204/MM5204 EPROM Programmer   *
*         By Martin Eberhard         *
*       Firmware  Version 2.01       *
*====================================*

Type ? for command list

>
```

Step 17. Test with an EPROM

☐ Install an MM5204 EPROM in the ZIF socket, and repeat steps 1 through 16. This EPROM may get programmed while you test, so plan to erase it when you are done. Minimize the time you spend on steps 12 and 13, so that you do not leave Vdd or Vbb active for an extended time. This may damage the EPROM.

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 ME5204 Programmer, which will power-off all pins.

Section 3. Functional Testing

Power-off the ME5204 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 ME5204, and see that your terminal program can talk to it.

Note: The ME5204 Programmer uses an ‘Address Offset’ when uploading and downloading files. The Address Offset is set by the user (with the **AO** command), and defines an 8-bit offset for the high address byte in the hex files. During uploads, this address offset is added to the high address byte in the hex records. During downloads, the record data is only loaded into the buffer if the high address byte in the hex record minus the Address Offset is 00 or 01.

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 in these checks will generate a brief error message and bump the error count.

The record count, loaded record count (records with where the address high byte minus the Address Offset equaled 00 or 01), 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 ME5204 Programmer. The ME5204 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 ME5204) 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

You can always pause ME5204 transmission by sending an XOFF character, typically <control>S on your keyboard. Any key (including XOFF and XON) will restart transmission when paused.

Step 1. Display the Default Buffer Data

Monitor dialog:

```
>DB
000: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
010: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
020: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
040: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
050: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
060: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
070: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
080: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
090: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

```
0E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
100: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
110: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
120: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
130: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
140: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
150: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
160: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
170: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
180: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
190: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
1A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
1B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
1C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
1D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
1E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
1F0: 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 40 100
040: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
050: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
060: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
070: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
080: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
090: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
0A0: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
0B0: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
0C0: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
0D0: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
0E0: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
0F0: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
100: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
110: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
120: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
130: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
Buffer range checksum: 00
> FB AA
>Buffer filled with AA
>DB 32 81
032: AA AA AA AA AA AA AA AA AA AA AA AA AA AA
040: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
050: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
060: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
070: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
080: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
090: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
0A0: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
0B0: AA AA AA
Buffer range checksum: AA
>
```

(Note that you can display portions of the buffer by specifying the start address and the number of bytes to display.)

Step 3. Edit the Buffer

Monitor dialog:

```
>MB 110
110: AA 01 AA 02 AA 03 AA 04 AA 05 AA 06 AA 07 AA 08
118: AA 09 AA <control-C>

>DB 100 40
100: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
110: 01 02 03 04 05 06 07 08 09 AA AA AA AA AA AA AA
120: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
130: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
Buffer range checksum: B3
>
```

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 randomly setting the page address to 0x68 - you will see the result in the file.

Monitor dialog:

```
>AO 68
>Address Offset: 68
>UI          {start file capture before hitting Return}
:10680000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAE8
:10681000AAAAAAAAAAAAAAAA9AAAAAAAAAAAAAD8
:10682000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAC8
:10683000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAB8
:10684000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA8
:10685000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA98
:10686000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA88
:10687000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA78
:10688000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA68
:10689000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA58
:1068A000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA48
:1068B000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA38
:1068C000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA28
:1068D000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA18
:1068E000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA08
:1068F000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAF8
:10690000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAE7
:1069100001020304050607080 9AAAAAAAAAAAAA4
:10692000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAC7
:10693000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAB7
:10694000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA7
:10695000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA97
:10696000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA87
:10697000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA77
:10698000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA67
:10699000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA57
:1069A000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA47
:1069B000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA37
:1069C000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA27
:1069D000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA17
:1069E000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA07
:1069F000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAF7
:00000001FF
>
```

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

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 randomly setting the page address to 0x31 - you will see the result in the file.

Monitor dialog:

```
>AO 31
>Address Offset: 31
>US          {start file capture before hitting Return}

S1133100AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA1B
S1133110AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA0B
S1133120AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAFB
S1133130AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAEB
S1133140AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAADB
S1133150AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAACB
S1133160AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABB
S1133170AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAB
S1133180AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA9B
S1133190AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA8B
S11331A0AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA7B
S11331B0AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA6B
S11331C0AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA5B
S11331D0AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA4B
S11331E0AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA3B
S11331F0AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA2B
S1133200AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA1A
S113321001020304050607080 9AAAAAAAAAAAAAD7
S1133220AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAFA
S1133230AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAEA
S1133240AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAADA
S1133250AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAACA
S1133260AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABA
S1133270AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
S1133280AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA9A
S1133290AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA8A
S11332A0AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA7A
S11332B0AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA6A
S11332C0AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA5A
S11332D0AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA4A
S11332E0AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA3A
S11332F0AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA2A
S9030000FC
>
```

{Stop capturing and close the file on your computer}

Use a text editor to examine the file STEST.TXT, to make sure it transferred correctly, and to delete the '>' at the end.

Step 6. Test downloading files to the ME5204 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 ME5204.) Note that we set the Address Offset to match the base address in the hex file - otherwise nothing will get loaded into the buffer.

Monitor Dialog:

```
>FB 99
>Buffer filled with 99
>DB 25 44
25: 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
Buffer range checksum: A4
>AO 68
>Address Offset: 68
```

{Now, start sending the file INTEST.TXT to the ME5204}

```
>:10680000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAE8
:10681000AAAAAAAAAAAAAAAA9AAAAAAAAAAAAAAD8
:10682000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAC8
:10683000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAB8
:10684000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA8
:10685000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA98
:10686000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA88
:10687000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA78
:10688000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA68
:10689000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA58
:1068A000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA48
:1068B000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA38
:1068C000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA28
:1068D000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA18
:1068E000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA08
:1068F000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAF8
:10690000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAE7
:1069100001020304050607080 9AAAAAAAAAAAAAA4
:10692000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAC7
:10693000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAB7
:10694000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA7
:10695000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA97
:10696000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA87
:10697000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA77
:10698000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA67
:10699000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA57
:1069A000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA47
:1069B000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA37
:1069C000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA27
:1069D000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA17
:1069E000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA07
:1069F000AAAAAAAAAAAAAAAAAAAAAAAAAAAAAF7
:00000001FF
Records: 21, Bad Records: 00
20 records loaded into buffer with Address Offset: 68
>DB 100 100
100: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
110: 01 02 03 04 05 06 07 08 09 AA AA AA AA AA AA AA
120: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
130: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
140: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
150: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
160: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
170: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
180: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
190: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
1A0: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
1B0: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
1C0: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
1D0: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
```

```
1E0: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
1F0: AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
Buffer range checksum: 33
>
```

You can test with the file STEST.TXT the same way. Remember that its Address Offset 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 and cannot be programmed.
Is it inserted backwards? Is it really a MM4204 or MM5204?
Abort
>
```

3.2 EPROM Reading and Programming

Here, you need a blank MM5204 EPROM - preferably several of them. Known-good EPROMs would be nice. You will also want an EPROM eraser, as you will be filling EPROMs with junk.

Step 1. Low-voltage Operations

Power-on the ME5204 Programmer, and then install a blank MM5204 EPROM in the ZIF socket, with its pin 1 closest to the ZIF socket handle.
Monitor Dialog:

```
>EB
EPROM is blank
>

{or...}
Error Address: XXX  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
EPROM read into buffer
EPROM checksum:00
>DB
000: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
010: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
020: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
040: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
050: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
060: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
070: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
080: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
090: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

```
0E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
100: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
110: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
120: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
130: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
140: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
150: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
160: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
170: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
180: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
190: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
1A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
1B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
1C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
1D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
1E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
1F0: 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
085: 00 77 00 <control-C>
>EC
Error Address: 085  Buffer: 77  EPROM: 00
Fail
>
```

Step 2. High-Voltage Operations

First, create some interesting data.

Monitor Dialog:

```
>FB 55
>Buffer filled with 55
>MB
000: 55 1 55 2 55 4 55 8 55 10 55 20 55 40 55 80
008: 55 AA 55 <control-C>
>MB 19A
19A: 55 12 55 34 55 56 55 78 55 9A 55 BC
1B0: 55 DE 55 F0 55 <control-C>
>DB
000: 01 02 04 08 10 20 40 80 AA 55 55 55 55 55 55 55
010: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
020: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
030: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
040: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
050: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
060: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
070: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
080: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
090: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
0A0: DE F0 55 55 55 55 55 55 55 55 55 55 55 55 55 55
0B0: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
0C0: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
```

```
0D0: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
0E0: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
0F0: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
100: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
110: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
120: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
130: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
140: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
150: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
160: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
170: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
180: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
190: 55 55 55 55 55 55 55 55 55 55 12 34 56 78 9A BC
1A0: DE F0 55 55 55 55 55 55 55 55 55 55 55 55 55 55
1B0: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
1C0: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
1D0: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
1E0: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
1F0: 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+5P smart algorithm, which will program the EPROM until it matches the buffer, then over-program it 5 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. In the following example, the EPROM matched the buffer after the second pass.

Raise the handle of the ZIF socket, put a blank MM5204 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
P+5P 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+5P Smart Programming..
P=02. 5P 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 ME5204 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
EPROM read into buffer
EPROM checksum:3C
>DB
000: 01 02 04 08 10 20 40 80 AA 55 55 55 55 55 55 55
010: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
020: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
030: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
040: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
050: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
060: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
070: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
080: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
090: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
0A0: DE F0 55 55 55 55 55 55 55 55 55 55 55 55 55 55
0B0: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
0C0: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
0D0: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
0E0: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
0F0: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
100: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
110: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
120: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
130: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
140: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
150: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
160: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
170: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
180: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
190: 55 55 55 55 55 55 55 55 55 55 12 34 56 78 9A BC
1A0: DE F0 55 55 55 55 55 55 55 55 55 55 55 55 55 55
1B0: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
1C0: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
1D0: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
1E0: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
1F0: 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55 55
Buffer checksum: 3C
>
```

Congratulations, your ME5204 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 Vdd pulse width is always 1.05 mS mSec, with a 21% duty cycle. See the timing diagrams in section 7's Programming Timing subsections for details.

The MM5204 data sheet recommends the P+5P Smart Programming Algorithm. You can choose to program with a fixed number of programming cycles instead.

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.

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+5P' Smart Programming Algorithm

Setting the number of programming cycles to 00 (**PC 00**) selects the P+5P Smart Programming Algorithm.

Programming Phase: This algorithm first prints "P+5P 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 "5P over-programming". The EPROM is then programmed another 5 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.

4.3 Programming Time

Nominally, each pass through an entire MM5204 will take about 2.6 seconds. However, any byte that is to be programmed as 00 will be skipped, since this is the erased state of a MM5204 data byte. Skipped bytes take much less time.

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

Section 5. ME5204 Commands

5.1 EPROM Commands

The EPROM commands generally deal with the 512-byte EPROM, and the 512-byte buffer in the ME5204. 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>=000 is interpreted as <cnt>=200 hex. If you don't enter values for <beg> and <cnt>, they both default to 000. If you enter only one value, it is assumed to be <beg>, and <cnt> defaults to 000 (which means 200h). The EPROM and buffer addresses will wrap around to 000 once they pass 1FF.

>**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). 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 ME5204 (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 "EPROM read into buffer" followed by "EPROM checksum: XX".

>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".

>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+5P Smart Programming Algorithm. If you don't type a value for <cnt>, then 00 (P+5P Smart Programming Algorithm) is assumed.

?E Help with EPROM Commands

?E prints a help screen for these EPROM commands.

5.2 File Transfer Commands

>AO Automartic Address Offset Mode (default)

AO (with no parameter) selects automatic Address Offset mode. When downloading a file to the ME5204, the Address Offset will be taken from the high address byte of the first received record. When uploading a file from the ME5204, the Address Offset will be 00.

>AO <offset> Set Address Offset

AO sets the Address Offset for uploads and downloads to <offset>. The Address Offset is added to the upper address byte when uploading buffer data, and subtracted from downloaded Intel Hex records and Motorola S-Records.

See the **Address Offsets** subsection under **ME5204 Programmer Usage** for further details about the Address Offset, 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

Target buffer addresses are calculated by subtracting the Address Offset from the address in each record, and then incrementing after each data byte of the record has been handled.

If the record type is 00 (a data record), then all data whose target buffer address is between 000 and 1FF will be loaded into the buffer. Any data whose target address is outside this range will not be loaded.

If the record’s checksum does not match the computed checksum, then “? Csm” will be printed, and the error count will be 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

Target buffer addresses are calculated by subtracting the Address Offset from the address in each record, and then incrementing after each data byte of the record has been handled.

If the record type is S1 (a data record), then all data whose target buffer address is between 000 and 1FF will be loaded into the buffer. Any data whose target address is outside this range will not be loaded.

An S5 (record count) record will compare the ME5204'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 checksum in the record does not match the checksum computed by the ME5204, 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 3-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 ME5204, 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 (TH command), then Vss will be +5 volts when you type TM 1, and 0 volts when you type TM 0. If programming high voltage is on (via TH 1), then Vss will be 13 volts when on, and 0 volts when off.

>**TB <0/1> Test Vbb**

TB 1 turns on Vbb, though this is only visible when programming high-voltage is on (via TH 1). When programming high-voltage is on, Vbb will be at 25 volts. When off (TB 0), Vbb will be at 13 volts.

>**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 Vss, which depends on TM and TH.

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

TD 1 turns Vdd on, TD 0 turns Vdd off. Vdd is active low, so off means at Vss, which depends on TM and TH. When on, Vdd will be at 0 volts.

>**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 Vss, which depends on TM and TH.

>**TH <0/1> Test Programming High-Voltage**

TH 1 turns on the high-voltage programming supply, and puts Vss to 13 volts. TH 0 turns off programming high-voltage.

>**WA <val> Write Address**

WA writes <val> to the nine 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 Vss.

If you don’t enter a value for <val>, then 000 is written.

>**WD <val> Write Data**

WD writes <val> to the eight 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 Vss.

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 FF first.

Note also that the pins are pulled up to Vss. 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.

Use a jumper to test each pin - ground the pin and read the result.

>**RV Read Voltages**

RV reads the Vss, Vbb, and the negative supply, Vneg. Each of these pins has a resistor network. The 16-bit value printed is the direct result of the A/D converter. 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: Loader Kernel firmware revision, Page Address, EPROM Type, Programming Cycles/Smart Programming mode, and Echo State.

>**RE Reset ME5204 Programmer**

RE is just like power-cycling the ME5204.

>**?L View Firmware Loader Notes**

?L displays notes on loading new firmware into the ME5204 via the serial port. Firmware loading is discussed in a later section.

^S Pause Serial Port Output

Control-S (XOFF) tells the ME5204 to stop sending data. Any subsequent character (including XON, which is control-Q) will re-enable the ME5204 output, and that character will be discarded by the ME5204.

Section 6. ME5204 Programmer Usage

The previous sections walked you through the basic ME5204 Programmer operation. Here are some specifications, and the procedures for common EPROM operations.

6.1 120V and 240V Operation

The ME5204 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, such that 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, such that the triangular pointer near the power cord inlet points to the 220V-240V arrow.

To avoid damage to the ME5204, **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.

Female DA-9 Pin	Signal
2	Data Out (out of the ME5204)
3	Data In (in to the ME5204)
5	Ground

For a normal PC connection, you will need a standard straight-through male DA-9 to male DA-9 cable like this. Also, a standard USB-RS232 dongle (the kind with a male DA-9) will plug directly into the ME5204.

Male DA-9 Pin	Signal	Male DA-9 Pin
2	Data In (in to the ME5204)	2
3	Data Out (out of the ME5204)	3
5	Ground	5

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) ‘Ready’ LED is connected to the 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 ME5204 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 an EPROM. The software tolerances are set wide enough that you should never see a voltage error message with a correctly-functioning ME5204. Any voltage error message should cause you to debug the hardware.

Because of these sloppy tolerances, you cannot assume that the ME5204 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.

During reading, the ME5204 checks Vss 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 negative supply voltage, Vneg is checked for a reasonable voltage. If it is too low or too high, you will be given the option to continue.

During programming, the ME5204 checks Vss to see if it is roughly 13 volts. Vbb is similarly checked, to see if it is roughly 13 volts before each actual programming pulse. If either is too high, the ME5204 will panic, preventing programming. If either is too low, an error message will be printed, and you will be given the option to continue.

Vbb is also checked during teach programming pulse. If it is to far above 25 volts, the ME5204 will panic and abort the programming. If it is ever low during the programming pulse, the ME5204 will report this fact when programming has completed - whether or not the programming was successful.

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 ME5204 if this happens - otherwise you will have a hot socket, and will risk damaging EPROMS when you install or remove them.

6.5 Address Offsets

Simple Intel Hex files and Motorola S-record files have 2-byte addresses (represented as 4 hex digits). An MM5204 EPROM contains 512 bytes of data. Thus, the address in a hex file can be thought of as having 2 components: without any address offset, the high byte (first 2 hex digits) can either be 00 or 01, and the low byte (second 2 hex digits) represents the address within one half or the other of the EPROM.

The EPROM data may be a computer program intended to run at address 0000, or it may be intended to run 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 or 01 - these bytes will be the high byte of the intended target address.

You can tell the ME5204 what the address offsete is, using the **AO** (Address Offset) command. If you specify automatic Address Offset mode (by typing the **AO** command with no value), then the Address Offset for downloads will be set to the high address byte from the first received record of the file. Automatic Address Offset mode is the default after reset.

When you download a hex file to the ME5204, the Address Offset is subtracted from the high address bytes for the data in the file. Any data whose address high byte minus the Address Offset does not equal 00 or 01 will not be loaded into the buffer.

Note that it is possible for some of the data within a given record will get loaded into the buffer, and some of it will not. (This will happen if the record spans a 512-byte boundary.) Such records are not counted as loaded records in the reporting at the end of the file, though the data that fit into the buffer did get written into the buffer.

Note that if you have a hex file that spans several EPROMs, you can program each EPROM sequentially by first setting the Address Offset 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 Address Offset appropriately. Each time, only data whose address high byte minus the address offset equals 00 or 01 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 Address Offset before uploading the buffer. The Address Offset that you specify with the **AO** command will be added to the high address byte in every record uploaded. If automatic Address Offset mode is selected, then the Address Offset will be 000.

6.6 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 so-called P+5P Smart Programming Algorithm, which figures out how many passes it takes to program the EPROM and get a successful read-back, then programs it five times that many more times. To select the P+5P Smart Programming Algorithm, set the Programming Count to 0:

Monitor Dialog:

```
>PC 0
P+5P Smart programming algorithm enabled
>
```

Otherwise, you can simply tell the ME5204 Programmer how many passes to program the EPROM. For example, to program 32 decimal (20 hex) cycles, do this:
Monitor Dialog:

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

6.7 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 two 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 00}
>AO XX {XX is the Address Offset for the EPROM}
>Address Offset: XX
>   {send S-Record or Intel Hex file to the ME5204}
>EP
Make sure the EPROM is inserted correctly, with pin 1
closest to the socket handle. Ready to program (Y/N)? Y
P+5P Smart Programming..
P=02. 5P over-programming...........
Verifying
EPROM matches buffer
>
```

6.8 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}
>AO XX {XX is the Address Offset for the file}
Address Offset: 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**.

6.9 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
Make sure the EPROM is inserted correctly, with pin 1
closest to the socket handle. Ready to program (Y/N)? Y
P+5P Smart Programming..
P=02. 5P over-programming.........
Verifying
EPROM matches buffer
{insert another blank EPROM in the socket}
>EB
EPROM is blank
>EP
Make sure the EPROM is inserted correctly, with pin 1
closest to the socket handle. Ready to program (Y/N)? Y
P+5P Smart Programming...
P=02. 5P over-programming..............
Verifying
EPROM matches buffer
>
```

Section 7. ME5204 Theory of Operation

7.1 Architecture

The following is a very brief description of the ME5204 Programmer's circuit operation. For further detail, see the MM5204 specification.

The ME5204 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 MM5204 EPROM.

7.2 Isolation and Stepdown

Transformer T1 steps the line voltage down to 56VAC RMS.This is rectified by a bridge diode (D4-D7), and split into a positive and negative DC voltage, using the transformer's center-tap.The two DC supplies are smoothed with 470 uF caps (C12 & C13), large enough that they will charge to the AC line peak voltages, +39V and -39V. FS1 and FS2 protect these supplies from overcurrent.

7.3 Logic Supply

V4 regulates the +39V to produce +5V for the PIC microcontroller and the RS-232 interface. Rework zener diode Z1 is used to drop the unregulated, rectified voltage to well below 40V, to meet the LM7805's specification. Even with this rework, V4's input voltage is relatively high, between 24 and 30 volts. For this reason, it is normal for regulator V4 to get warm when the ME5204 is on.

7.4 Microcontroller-Controlled High-Voltage Supply

The MM5204 EPROM programming voltages are specified as Vbb = 0V & +12V, Vl = -13V, Vdd = 0 & -49V, and logic signals at 0V and -13V to the EPROM's inputs. These voltages are referenced to the EPROMs Vss, defined as 0V. When reading the EPROM, voltages are referenced to ground, defined as 5V below the voltage of the Vss pin, since the MM5204 does not have a ground pin.

To make interfacing simpler, the programming voltages are translated up by 13 volts, referenced to Vdd, when programming high-voltage is on. The voltages translate as follows:

Pin name	MM5204 Specified Voltage			Translated Voltage		
	Read	Ready[1]	Program	Read	Ready[1]	Program
Vss	5V ±5%	0V	0V	5V ±5%	13V±2%	13V±2%
Vdd	-12V ±5%	-2V to 0.5V	-49±1V	-12V ±5%	13V±2%	-36±0.1V
Vll	0V	0V to -14V	0V to -14V	0V	0V	0V
Vbb	5V ±5%	0V to 0.4V	12V ±0.6V	5V ±5%	13V±2%	25V±5%
PROG	5V ±5%	0V	-49±1V	5V ±5%	13V±2%	-36±0.1V
CSn	0V	0V	0V	0V	13V±2%	13V±2%
A<0:7> high	2.4V to 5V	0V	0V	3.5V to 5V	12.5V±5%	12.5V±5%
A<0:7> low	0 ±0.8V	-13 ±2V	-13 ±2V	0 to 0.5V	0 to 1V	0 to 1V
<0:7> high	(output)	-13 ±2V	-13 ±2V	(output)	0 to 1V	0 to 1V
D<0:7> low	(output)	0V	0V	(output)	12.5V±5%	12.5V±5%

1. Ready for programming

Vss is generated by an LM317 programmable voltage regulator (V2), with precision resistors R27 and R29 setting the 5V level, and R26 setting the 13V level. The microcontroller controls Q5 to select between these two voltages.

V1 and surrounding components serve as a programmable voltage regulator for Vbb, capable of providing 5V, 13V, and 25V, controlled by Q3 & Q4. All of the voltages are adjusted by VR1. The 13V output has the most restrictive requirement in the MM5204 spec, so adjustment should be made when the output is set for 13V.

V3 and surrounding components generate Vneg, providing the negative voltages used for Vdd and -PROG. This power supply provides either -12V or -36V, controlled by Q22 and Q23.

Vdd is pulled up to Vss when Q2 is off, and driven to at Vneg when Q2 is on. The microcontroller controls Q2 via its EN_VDD output.

Similarly, -PROG is pulled up to Vss when Q8 is off, and driven to at Vneg when Q8 is on. The microcontroller controls Q2 via its EN_PROGRAM output.

7.5 EPROM Read/Write Interface

Every MM5204 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 Vss (which may either be 5V or 13V) 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 off by the microcontroller.

Because the address and data drivers must operate at 13 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.

7.6 Microcontroller

All ME5204 Programmer operations are controlled by a 40-pin PIC16F1519 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 1024 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 16K of 14-bit program memory EEPROM, which holds the ME5204 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 Vneg, Vss, and Vbb, so that the PIC can measure these voltages and perhaps could warn the user if voltages are out of spec.

Ther PIC has been programmed with two pieces of firmware: the Loader and the Programming Firmware. See section 8 for details.

7.7 Voltage Measurement

Three A/D converters are implemented in the Microcontroller, measuring Vneg, Vss, and Vbb. They each connect to the power supply via a resistor divider. The A/D converters use the PIC's 5-volt Vss as a reference.

Vneg is equal to (5 - 44.35 X (ADC value)/resolution) volts

Vss and Vbb are equal to 5.99 X (ADC value)/resolution volts

The PIC's Vss 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 a properly adjusted power supply, but it is good enough to detect many potential failure modes, including 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 ME5204 from cascading into a larger failure, and also help prevent damaging an EPROM.

Here are the Vneg A/D result limits used by the firmware. The Minimum and Maximum Voltages are the extremes when all the tolerances are worst-case in either direction. The highlighted values are the limiting cases.

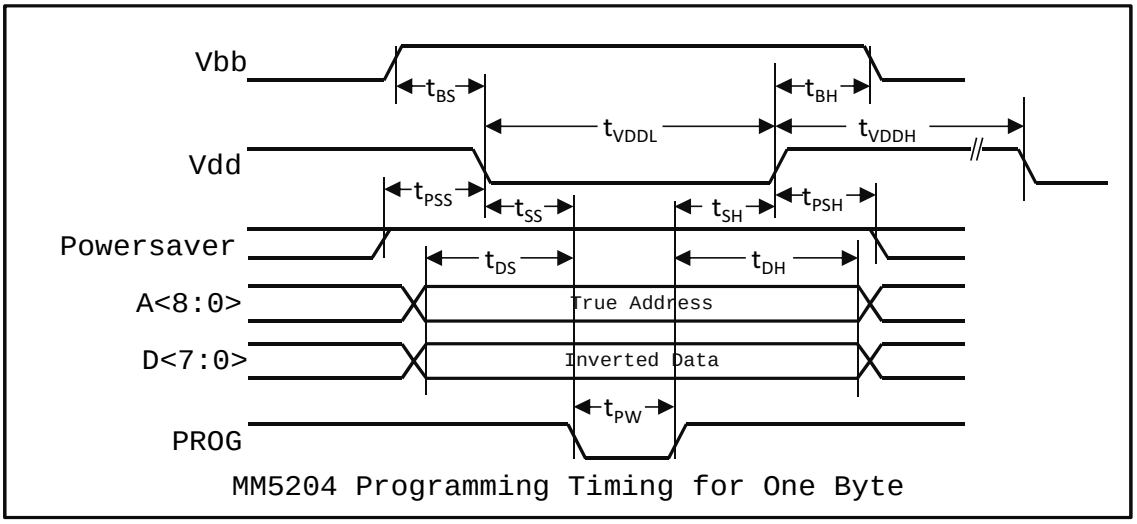
A/D high result	Minimum Voltage	Maximum Voltage	Note
03	>-41.64V	**>-36.20V**	Acceptable -36V programming voltage
22	**<-35.66V**	<-30.98V	
97	>-14.21V	**>-12.24V**	Acceptable -12V voltage
A3	**<-11.75V**	<-10.10V	
DF	>-0.87V	**>-0.59V**	Acceptable 0V (off) voltage
E4	**<0.29V**	<0.42V	

Here are the Vss and Vbb A/D result limits used by the firmware. The Minimum and Maximum Voltages 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	Maximum Voltage	Note
00	>0V	>0V	Acceptable 'off' voltage range
13	**<0.55V**	<0.62V	
26	>4.13V	**>4.72V**	Acceptable 5V (low voltage) Vss
2F	**<5.25V**	<6.00V	
66	>11.13V	**>12.72V**	Acceptable +13V Vss output
78	**<13.24V**	<15.13V	
C6	>21.64V	**>24.72V**	Acceptable 25V Vbb
E6	**<25.27V**	<28.88V	

7.8 MM5204 Programming Timing

Below is the timing specification for programming the MM5204 EPROM, compared to the timing of the ME5204 Programmer. 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, T_{VDD}



MM5204 Programming Timing for One Byte

Symbol	Parameter	MM5204 Spec		ME5204 (±10%)
		Min	Max	
T_{VDD}	T_{VDDL}/(T_{VDDH} + T_{VDDL})= Vdd Duty cycle		25%	21%
T_{PW}	PROG Pulse Width	0.5 mSec	5 mSec	1.0 mSec
T_{DS}	Data and Address Setup to PROG↓	40 uSec		1.06 mSec
T_{DH}	Data and Address Hold from PROG↑	0		3 mSec
T_{SS}	Pulsed Vdd Setup to PROG↓	40 uSec	100 uSec	60 uSec
T_{SH}	Pulsed Vdd Hold from PROG↑	1 uSec		2.5 uSec
T_{BS}	Pulsed Vbb Setup to Vdd↓	1 uSec		500 uSec
T_{BH}	Pulsed Vbb Hold from Vdd↑	1 uSec		2.5 uSec
T_{PSS}	Powersaver Setup to Vdd↓	1 uSec		>500 uS
T_{PSH}	Powersaver Hold from Vdd↑	1 uSec		>500 uS

7.9 Backwards EPROM Detection

Attempting to program an EPROM that is installed backwards can damage the ME5204. In particular, the fuse will blow, and one or two of the transistors might be damaged. Prior to programming, the ME5204 firmware attempts to detect an EPROM that is installed backwards, to try and avoid damage.

A backwards MM5204 reads as all bytes=FF. Reading a backwards MM5204 does not harm the ME5204 (and does not seem to harm the EPROM either). Note that unprogrammed bytes in a MM5204 read as 00.

Prior to programming, the ME5204 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 ME5204 from trying to program a backwards EPROM, as well as preventing damaging the EPROM.

Section 8. Downloading Firmware via the Serial Port

You can load new ME5204 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 ME5204.

The ME5204 firmware is divided into two components: the ME Loader Kernel (the Loader, which cannot be downloaded via the serial port) and the ME5204 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 ME5204 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 ME5204, or immediately after issuing a Reset command. You will see the Loader message, instead of the ME5204 sign-on banner:

```
ME Loader 1.0
```

When you see this message, the ME5204 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, me5204.hex) to the ME5204. 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 received (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. 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 ME5204.

If the load fails for any reason, 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 ME5204 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 ME5204 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.

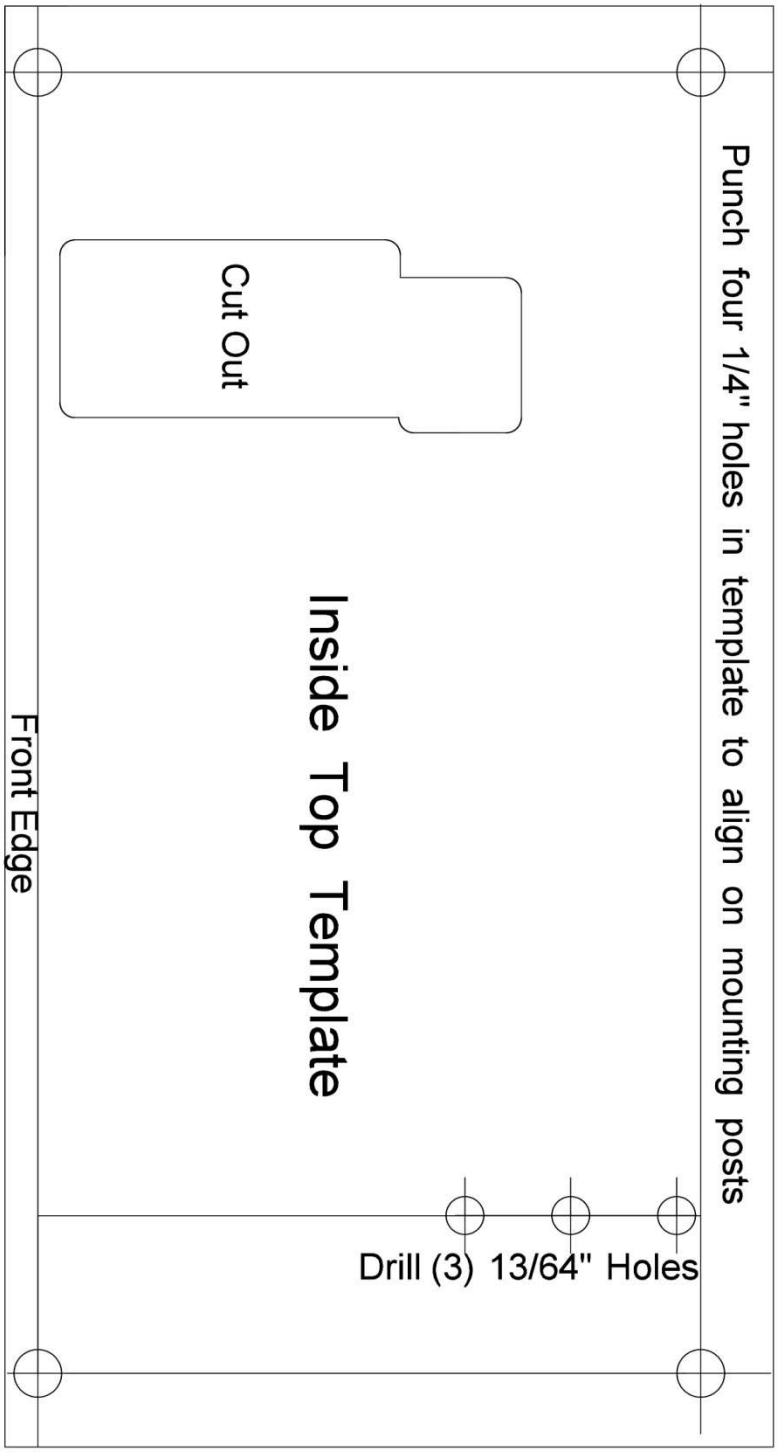
The PIC's **Linear RAM** is assigned as follows:

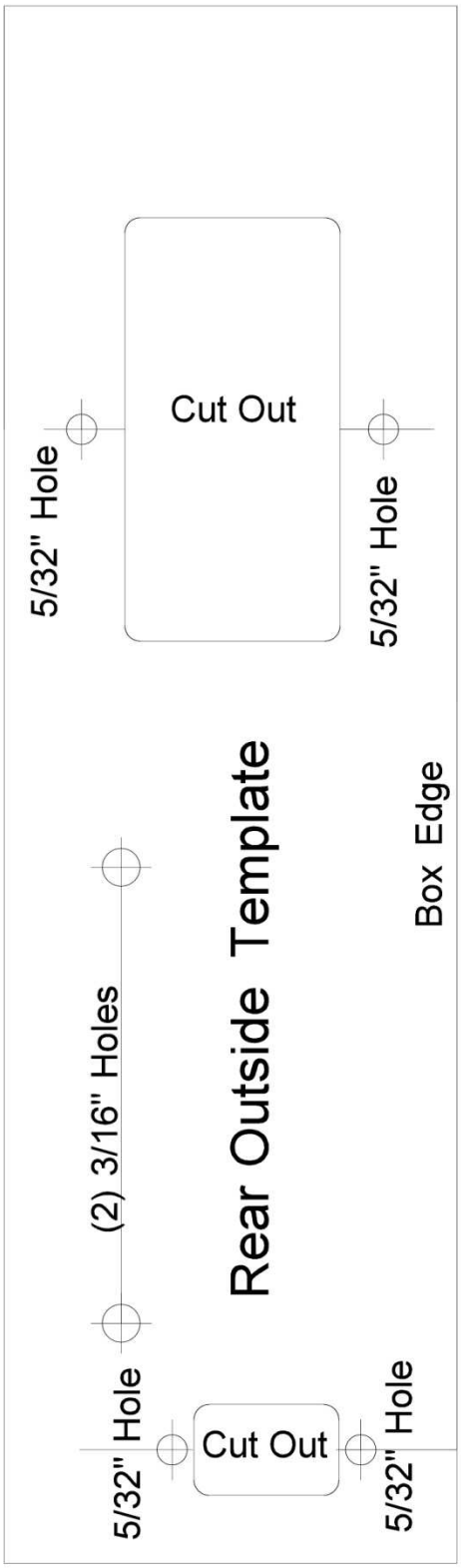
Address	Name	Function
0x2000	RAM_BUF	Available for 512-byte EPROM image buffer.
0x2200	RX_QUEUE	UART receive circular queue (128 bytes)
0x2280	TX_QUEUE	UART transmit circular queue (16 bytes)
0x2290-0x23EF	Available	Available for use by Programming Firmware

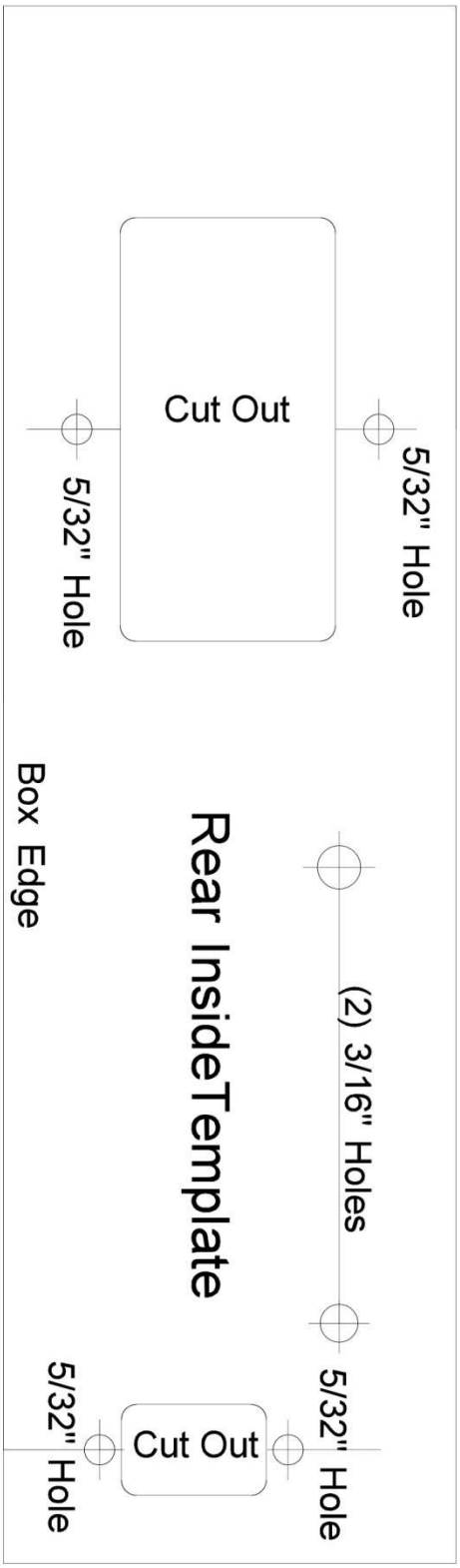
Section 9. Drawings

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**.




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


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

9.2 Bill of Materials

The following two pages are the complete Bill of Materials for the ME5204.

The Digikey part numbers and prices 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 ME5204 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 (where I used MPSA05 and MPSA56 transistors) need to tolerate at least 60V Vce. Most others must tolerate at least 40V Vce. Some (e.g. Q1) must pass 500 mA Ic with relatively low (less than 0.5V) voltage drop.
- If you replace the transformer with one that does not support both 120V and 240V inputs, then you should disable the appropriate circuit on the Multifunction Inlet Subassembly.

PCBA Components

NOTE: Components listed in assembly order

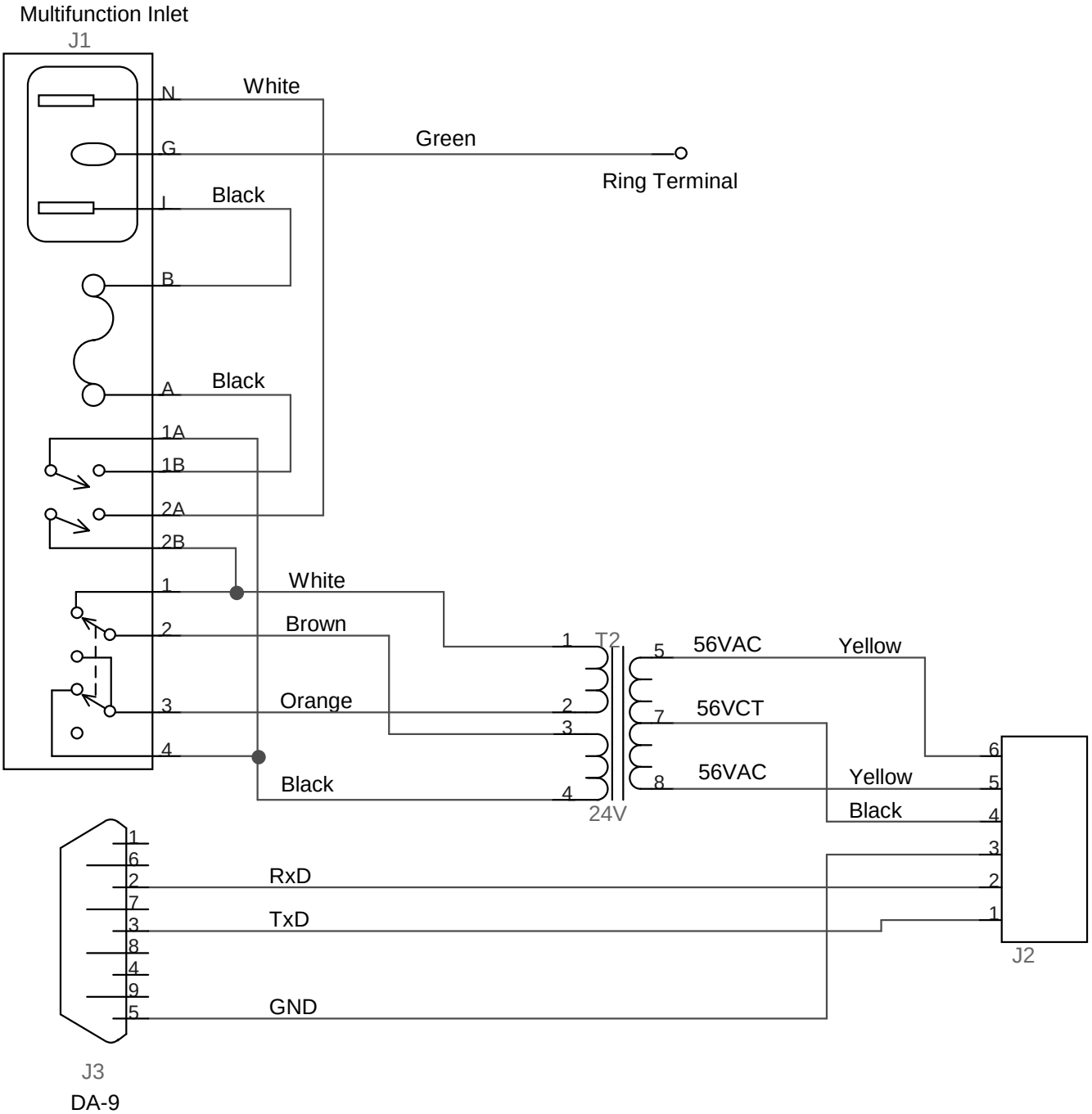
Qty	Component	Value	locations	Digikey Part Number
1	PC Board, rev A or B			
3	Resistor, 1/4W, 324 1%	324 1%	R29,R30,R48	324XBK-ND
1	Resistor, 1/4W, 976 1%	976 1%	R27	976XBK-ND
2	Resistor, 1/4W, 2.1K 1%	2.1K 1%	R22,R26	RNF14FTD2K10CT-ND
1	Resistor, 1/4W, 2.8K 1%	2.8K 1%	R47	2.80KXBK-ND
1	Resistor, 1/4W, 3.09K 1%	3.09K 1%	R23	RNF14FTD3K09CT-ND
3	Resistor, 1/4W, 10K 1%	10K 1%	R64,R83,R85	10.0KXBK-ND
2	Resistor, 1/4W, 49.9K 1%	49.9K 1%	R82,R84	49.9KXBK-ND
1	Resistor, 1/4W, 78.7K 1%	78.7K 1%	R63	78.7KXBK-ND
8	Resistor, 1/4W, 100, 5%	100	R5,R7,R9,R11,R13,R15,R17,R19	S100QCT-ND
2	Resistor, 1/4W, 330, 5%	330	R60,R70	CF14JT330RCT-ND
1	Resistor, 1/4W, 910, 5%	910	R25	CF14JT910RCT-ND
5	Resistor, 1/4W, 1K, 5%	1K	R43,R50,R59, R66,R78	CF14JT1K00CT-ND
1	Resistor, 1/4W, 5.6K, 5%	5.6K	R24	CF14JT5K60CT-ND
5	Resistor, 1/4W, 12K, 5%	12K	R1,R41,R44,R49, R99 [R99 is rework on Rev A PCB]	12KQBK-ND
9	Resistor, 1/4W, 47K, 5%	47K	R51-R58, R101	47KQBK-ND
56	Resistor, 1/4W, 3K, 5%	3K	R2,R3,R4,R6,R8,R10,R12,R14,R16,R18, R20,R21,R28,R31-R40,R42,R45,R46, R61,R62,R65,R67-R69, R71-R77,R79-R81,R86-R98, R100 [R100 is rework on Rev A PCB]	3.0KQBK-ND
1	Zener Diode, 12V, 1W, 1N4742A	1N4742A	Z1 (rework on Rev A PCB)	1727-1946-1-ND
3	Diode, 1N4148	1N4148	D1-D3	1N4148TACT-ND
4	Diode, 1N4004	1N4004	D4-D7	1N4004-TPMSCT-ND
1	Socket, 16-pin, low profile	16-pin DIP	U3	ED3046-5-ND
1	Socket, 40-pin, low profile	40-pin DIP	U2	A120355-ND
1	Capacitor, ceramic, 1000 pF, 100V	.001 uF	C17 (rework on Rev A PCB)	BC5112-ND
6	Capacitor, chip, 100V	0.1 uF	C1,C2,C4,C5,C15,C16	478-4855-ND
8	Capacitor, chip, 25V	1 uF	C3,C6-C11,C14	445-173583-1-ND
1	Regulator, Adjustable Pos., TO-220	LM317	V2	LM317MTGOS-ND
1	Regulator, Adjustable Neg., TO-220	LM337	V3	296-21577-5-ND
1	Regulator, Adjustable Pos., TO-92	LM317L	V1	LM317LZXTR-ND
2	Transistor, PNP, MPSA56	MPSA56	Q23 Q19	MPSA56-APMSCT-ND
3	Transistor, PNP, 2N3906	2N3906	Q1,Q8,Q20	2N3906-APCT-ND
4	Transistor, NPN, MPSA05	MPSA05	Q2,Q6,Q9,Q22	MPSA05-APMSCT-ND
30	Transistor, NPN, 2N3904	2N3904	Q3-Q5,Q7,Q10-Q18,Q21,Q24-Q39	2N3904-APCT-ND
1	Trimpot 1K	1K	VR2	262UR102B-ND
1	Trimpot 250	250	VR1	201UR251B-ND
4	Fuse Clip	Fuse Clip	FS1 & FS2 (both ends)	F4186-ND
0	Fuse Clip	Fuse Clip	(Alternate Part for FS1 & FS2)	BK-6005-ND
2	Capacitor,Electrolytic, 470 63V	470 uF 63V	C7,C8	493-1127-ND
3	Screw, 6-32, 5/16", Pan Head		V2-V4	H115-ND
3	Nut, 6-32		V2-V4	H220-ND
1	Heatsink, TO-220, vertical	Heatsink	V4	HS368-ND
1	LM7805	LM7805	V4	MC7805CT-BPMS-ND
1	Header, 6-pin, 0.1"	Hdr 6-0.1	J2	A31116-ND
1	Header, 6-pin, 0.156"	Hdr 6-0.156	J1	WM4624-ND
2	Test Pin	Hdr 1	GND	609-3466-ND
1	Socket, 24-pin ZIF	ZIF 24	U1 {REVERSE MOUNT}	3M2402-ND
3	Spacer, T1-3/4 LED, 0.12"	LED spacer	LED1-LED3	8311K-ND
1	LED, Red	LED Red	LED3 {REVERSE MOUNT}	160-1127-ND
1	LED, Green	LED Green	LED2 {REVERSE MOUNT}	160-1130-ND
1	LED, Blue	LED Blue	LED1 {REVERSE MOUNT}	C503B-BCS-CV0Z0461-ND
1	Tranceiver, RS232, DIP16	MAX232	U3	296-1402-5-ND
1	Microprocessor, PIC16F1519, DIP40	PIC16F1519	U2	PIC16F1519-I/P-ND
1	Fuse, 1-1/4", 3/4A, fast	3/4A fuse	FS1	283-2631-ND

Chassis Components

Qty	Part	Manufacturer	Mfg part no.	Supplier Part no.
1	0.25A 56V center-tapped Transformer, 120V/240V input	Hammond	186C56	HM4690-ND
1	slope-top cabinet	Hammond	1595EB	HM244-ND
1	Multifunction inlet	Qualtek	763-00/001	Q306-ND
1	Connector housing, 6-position .156	AMP	09-50-3061	WM2104-ND
6	Connector Contact	AMP	08-52-0072	WM2302-ND
1	Connector, 9-position, female, D-sub	Amphenol	DE09S064TLF	609-1525-ND
1	Ring Terminal, #6, star	TE Connectivity	F1759-ND	A29900CT-ND
1	Fuse, 1-1/4", 3/4A, fast			283-2631-ND
2	Screw, 6-32, 5/16", Pan Head			H115-ND
2	Flat washer, #6			H778-ND
2	Lock Washer, #6			H240-ND
2	Nut, 6-32			H220-ND
2	Screw, 4-40, 3/8", flat head			501-1531-ND
2	Screw, 40-40, 3/8", pan head			36-9901-ND
4	Nut, 4-40			H216-ND
4	3/8" self-tapping 4-40 screws, pan head			
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
	Various Hookup Wire			

9.3 Chassis Wiring Diagram

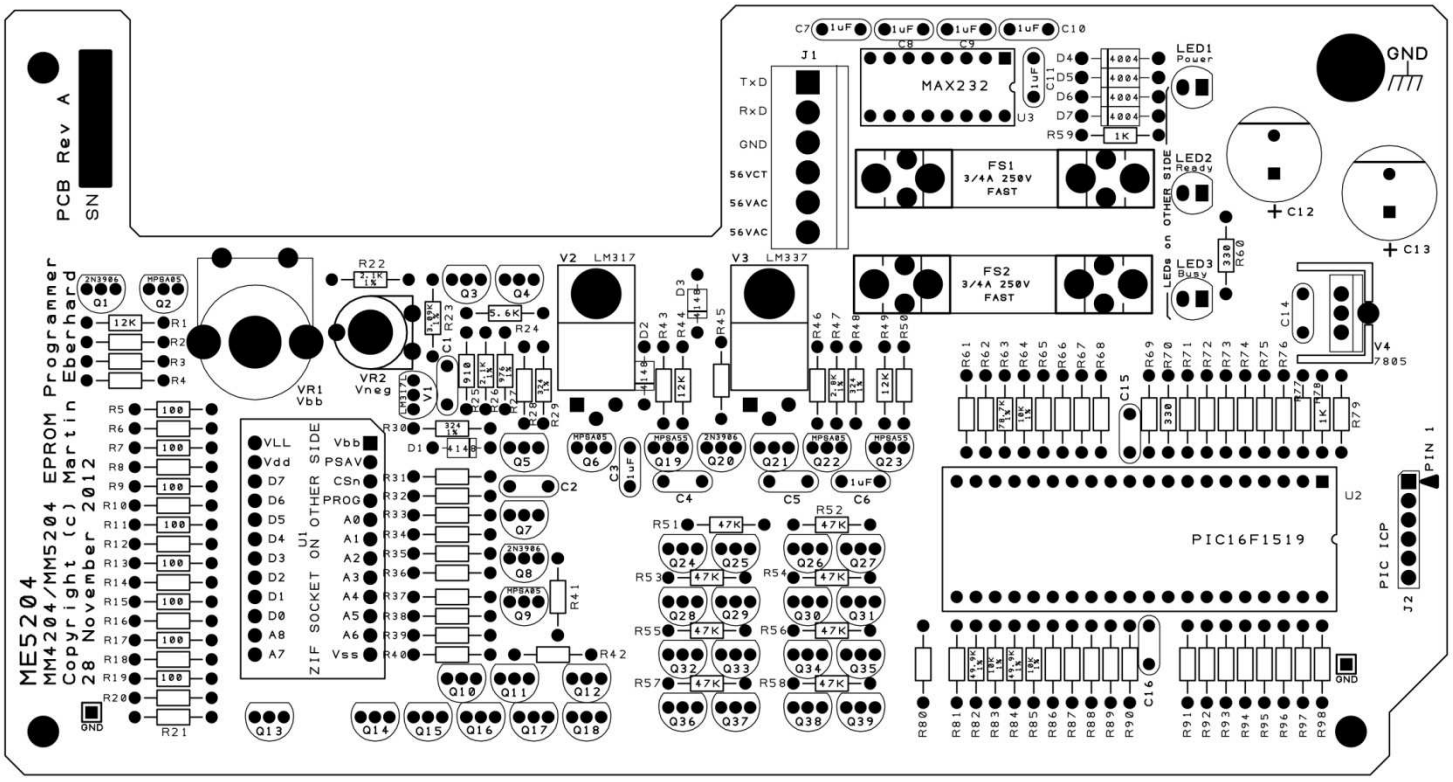
The following page is the schematic for the ME5204 chassis.



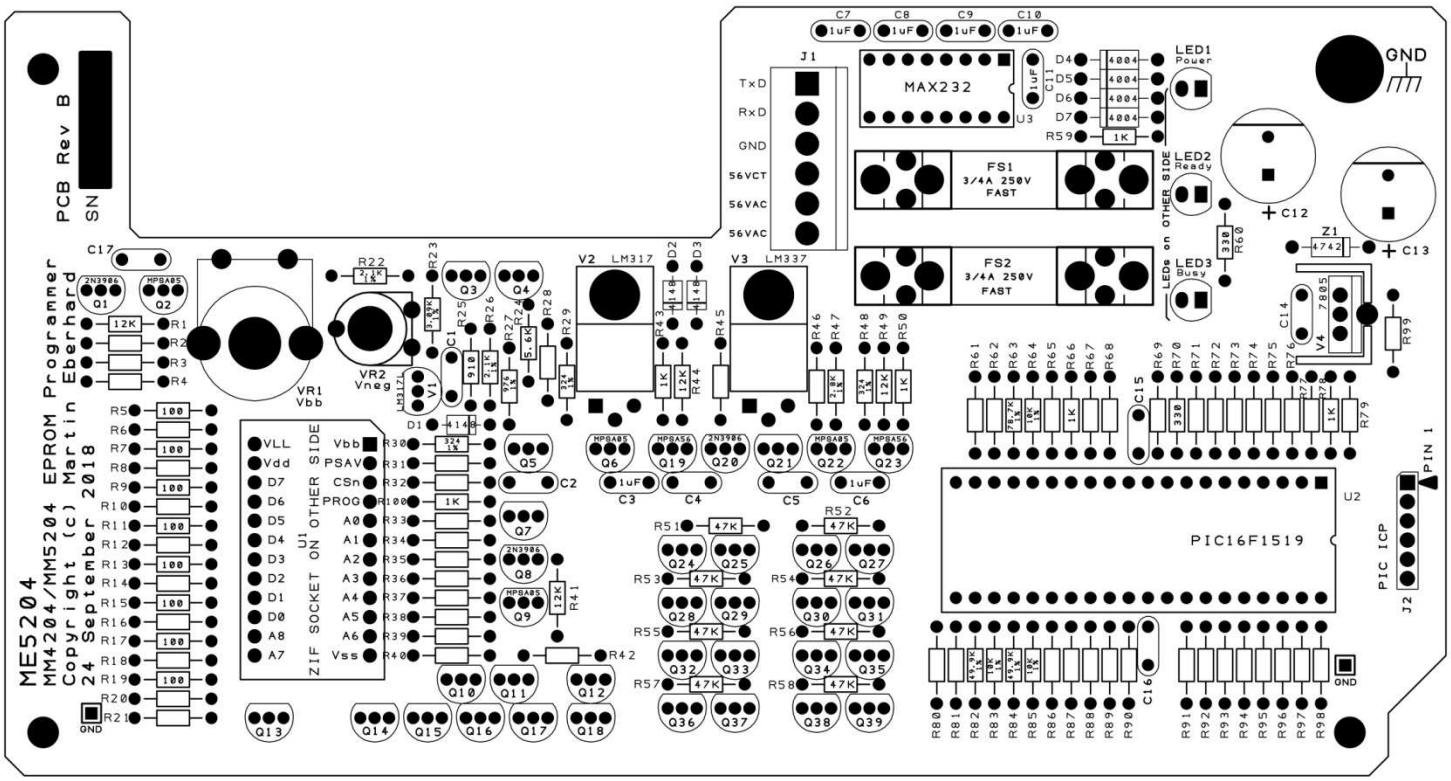
Title:	ME5204 EPROM Programmer		
Description:	Chassis Wiring		
Revision:		Drawn by:	M.Eberhard
Sheet:	1 of 1	Date:	15 January 2013

9.4 PCBA Component Placement

The following drawings the Rev A and Rev B printed circuit board outlines and silkscreen layers. This can help you find components on the printed circuit board assemblies.



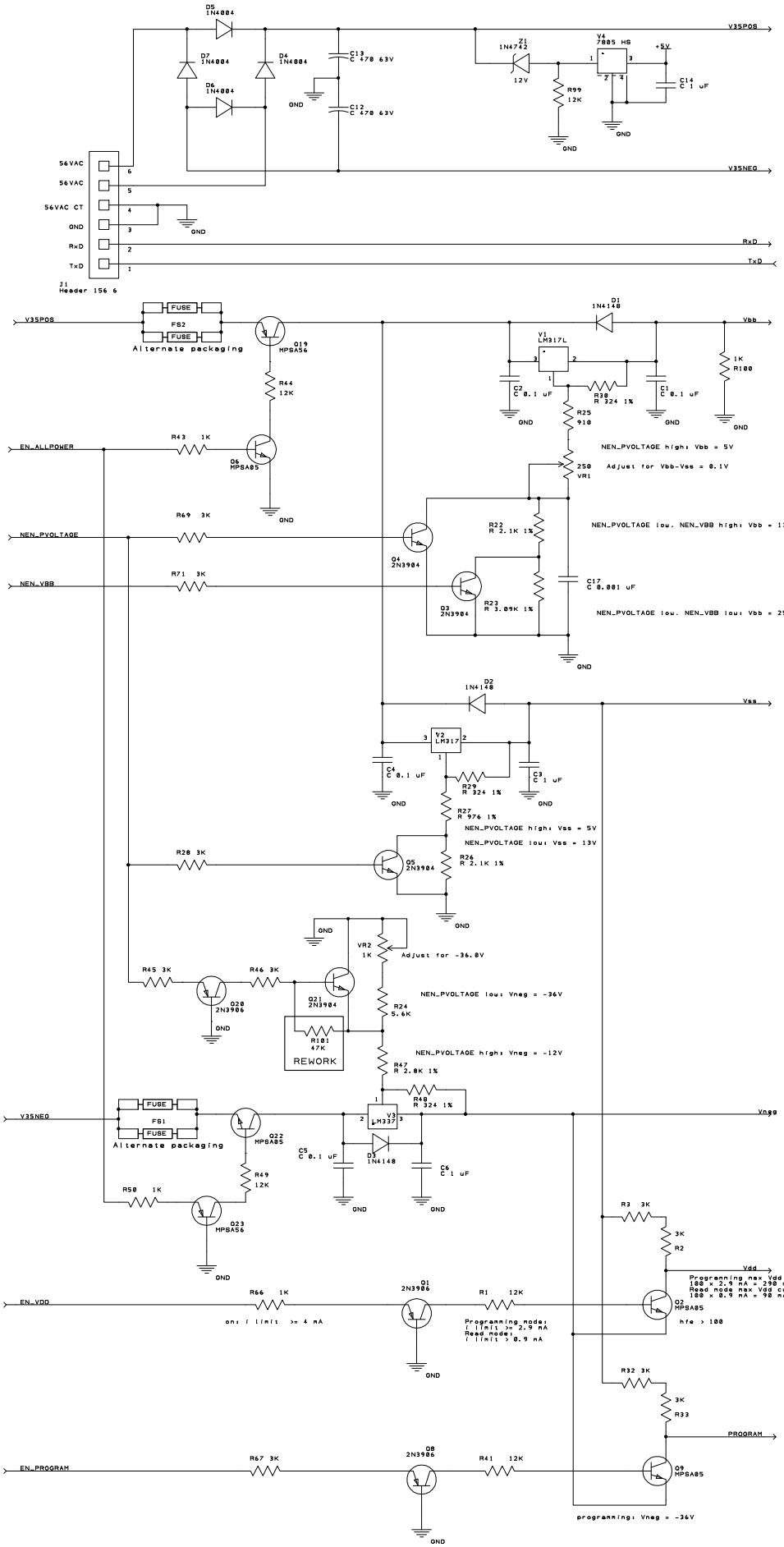
ME5204 Rev A PC Board Component Placement



ME5204 Rev B PC Board Component Placement

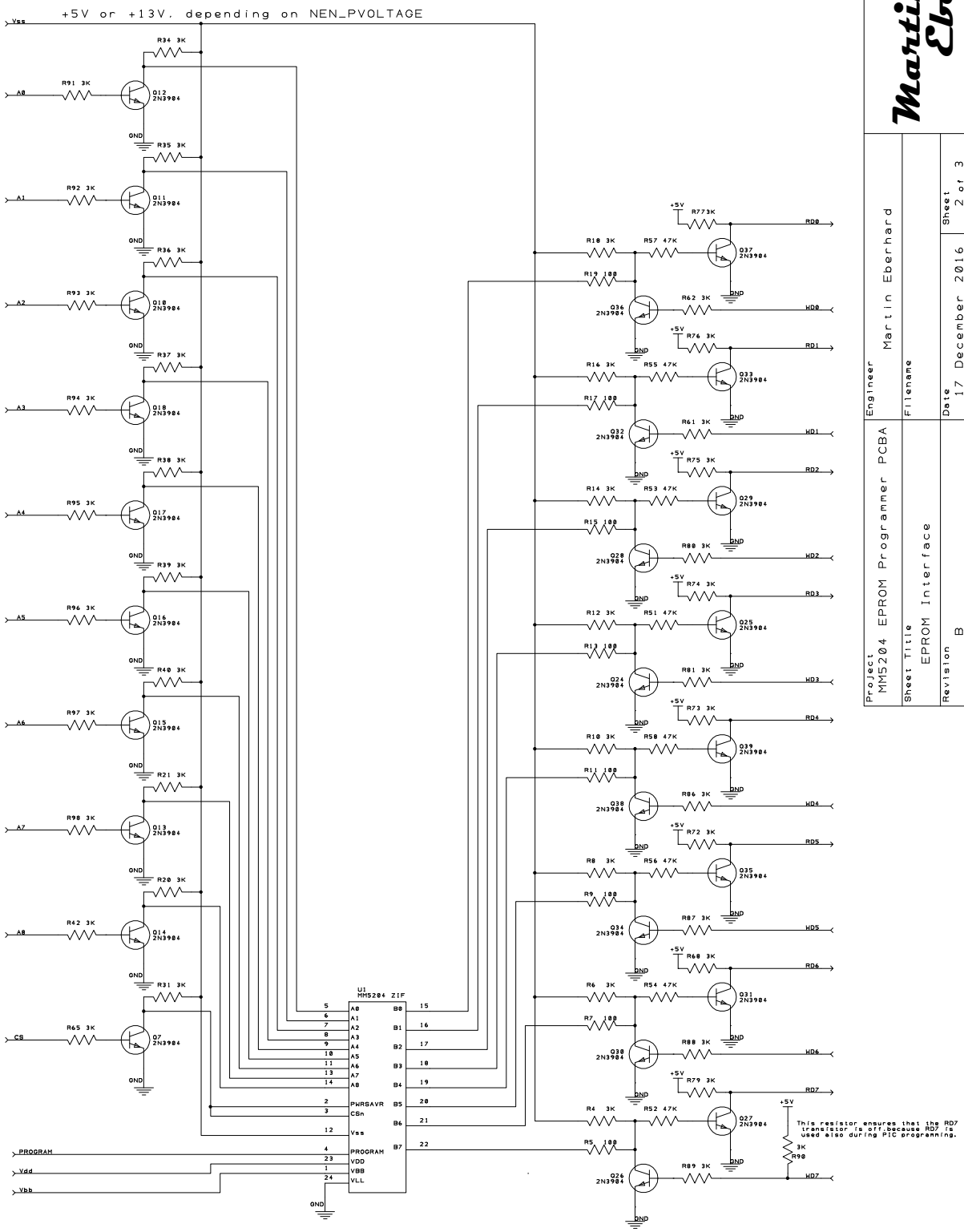
9.5 PCBA Schematics

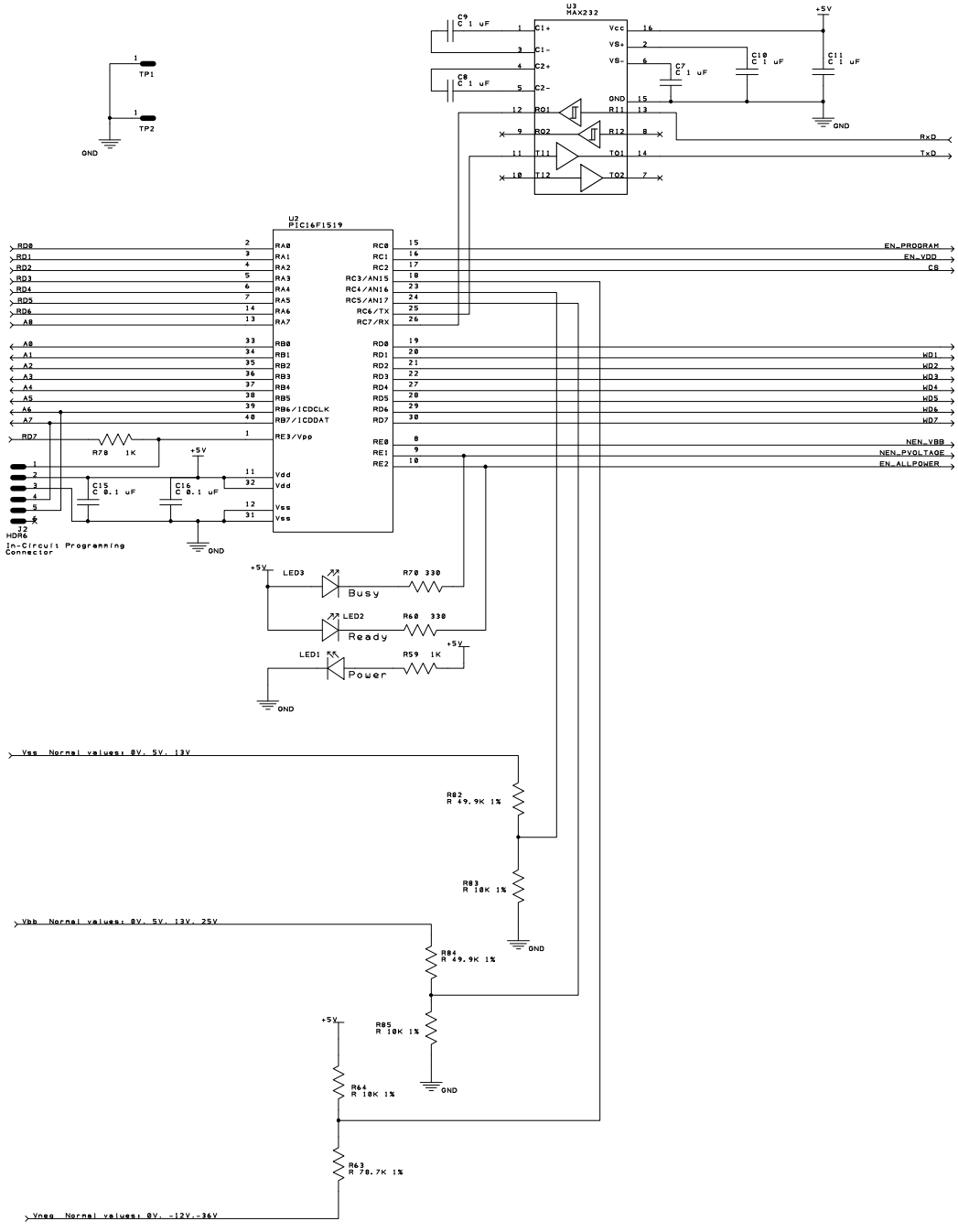
The following three pages are the schematics for the ME5204 Programmer’s printed circuit board assembly.



Martin Eberhard

Project MM5204 EPROM Programmer PCBA	Engineer Martin Eberhard	Sheet 1 of 3
Sheet Title Programmable Power Supply	Filename	Date 17 December 2016
	Revision B	


+5V or +13V, depending on NEN_PVOLTAGE
Vss
A0
A1
A2
A3
A4
A5
A6
A7
A8
CS
PROGRAM
Vdd
Vbb
GND
+5V
U1
MM5204 ZIF
PWRSAVR
CSn
VDD
VBB
VLL
B0
B1
B2
B3
B4
B5
B6
B7
RD0
RD1
RD2
RD3
RD4
RD5
RD6
RD7
WD0
WD1
WD2
WD3
WD4
WD5
WD6
WD7
This resistor ensures that the RD7 transistor is off,because RD7 is used also during PIC programming.
R90 3K
Project
MM5204 EPROM Programmer PCBA
Sheet Title
EPROM Interface
Revision
B
Engineer
Martin Eberhard
Filename
Date
17 December 2016
Sheet
2 of 3
Martin Eberhard



Martin Eberhard

Engineer: Martin Eberhard	Filename:	Sheet: 3 of 3
Project: MM5204 EPROM Programmer PCBA	Sheet Title: Microcontroller	Date: 17 December 2016
		Revision: B

9.6 MM5204 EPROM Specification

The following pages are National Semiconductor’s MMM5204 MOS EPROM specification.

National Semiconductor

MOS EPROMs

MM4204/MM5204
electrically programmable 4096-bit read only memory (EPROM)

general description

The MM4204/MM5204 is a 4096-bit static Read Only Memory which is electrically programmable and uses silicon gate technology to achieve bipolar compatiblity. The device is a non-volatile memory organized as 512 words by 8 bits per word. Programming of the memory is accomplished by storing a charge in a cell location by applying a −50V pulse. A logic input, "Power Saver," is provided which gives a 5:1 decrease in power when the memory is not being accessed.

- Standard power supplies 5.0V, −12V
- Static operation—no clock required
- Easy memory expansion—TRI-STATE® output Chip Select input ($\overline{CS}$)
- "Q" quartz lid version erasable with short wave ultraviolet light (i.e., 253.7 nm)
- Low power dissipation
- "Power Saver" control for low power applications

features

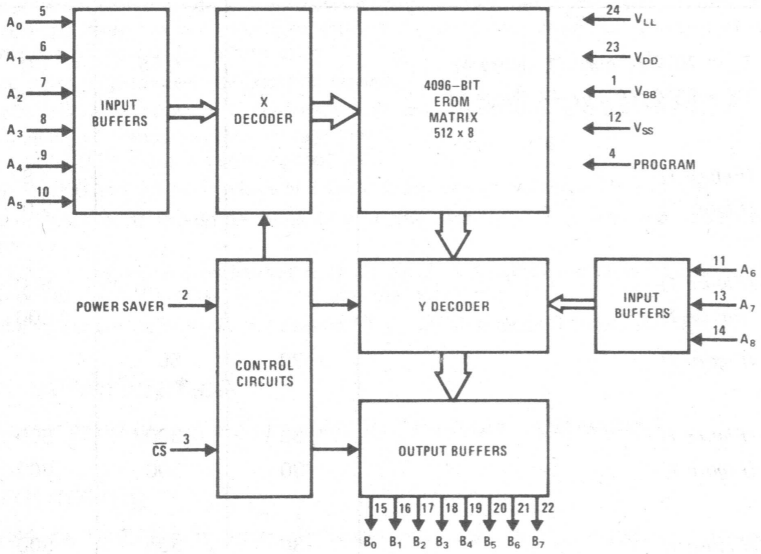
- Field programmable
- Fast program time: ten seconds typical for 4096-bits
- Fast access time
  - MM4204 1.25μs
  - MM5204 1μs
- DTL/TTL compatibility

applications

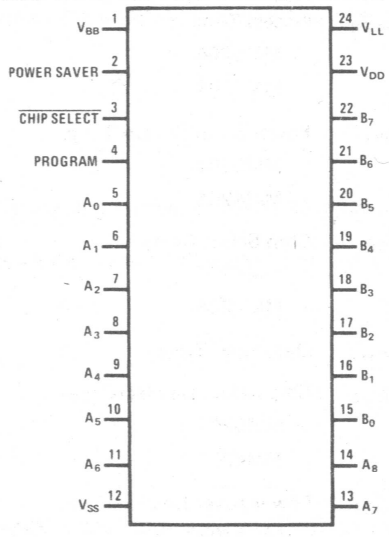
- Code conversion
- Random logic synthesis
- Table look-up
- Character generator
- Microprogramming
- Electronic keyboards

4

block and connection diagrams



Dual-In-Line Package



TOP VIEW

Order Number MM4204D or MM5204D
See Package 6

Order Number MM4204Q or MM5204Q
See Package 21

absolute maximum ratings (Note 1)

All Input or Output Voltages with Respect to V_{BB} Except During Programming	+0.3V to −20V
Power Dissipation	750 mW
Operating Temperature Range	
MM5204	0°C to +70°C
MM4204	−55°C to +85°C
Storage Temperature Range	−65°C to +125°C
Lead Temperature (Soldering, 10 seconds)	300°C

dc electrical characteristics

T_A within operating temperature range, V_{LL} = 0V, V_{BB} = PROGRAM = V_{SS}, MM4204: V_{SS} = 5.0V ±10%, V_{DD} = −12V ±10%, MM5204: V_{SS} = 5.0V ±5%, V_{DD} = −12V ±5%, unless otherwise noted.

PARAMETER		CONDITIONS	MIN	TYP (Note 7)	MAX	UNITS
V_{IL}	Input Low Voltage		V_{SS}−14		V_{SS}−4.2	V
V_{IH}	Input High Voltage		V_{SS}−1.5		V_{SS}+0.3	V
I_{LI}	Input Current	V_{IN} = 0V			1.0	μA
V_{OL}	Output Low Voltage	I_{OL} = 1.6 mA	V_{LL}		0.4	V
V_{OH}	Output High Voltage	I_{OH} = −0.8 mA	2.4		V_{SS}	V
I_{LO}	Output Leakage Current	V_{OUT} = 0V, $\overline{CS}$ = V_{IH}			1.0	μA
I_{DD}	Power Supply Current	MM5204 T_A = 0°C, $\overline{CS}$ = V_{IH}, Power Saver = V_{IL}		28	40.0	mA
		MM4204 T_A = 0°C, $\overline{CS}$ = V_{IH}, Power Saver = V_{IL}			50.0	mA
		MM5204 T_A = 0°C, $\overline{CS}$ = V_{IH}, Power Saver = V_{IH}		6.0	8.0	mA
		MM4204 T_A = 0°C, $\overline{CS}$ = V_{IH}, Power Saver = V_{IH}			10.0	mA
I_{SS}		MM5204 T_A = 0°C, $\overline{CS}$ = V_{IH}, Power Saver = V_{IL}			42	mA
		MM4204 T_A = 0°C, $\overline{CS}$ = V_{IH}, Power Saver = V_{IL}			52	mA
		MM5204 T_A = 0°C, $\overline{CS}$ = V_{IH}, Power Saver = V_{IH}			10	mA
		MM4204 T_A = 0°C, $\overline{CS}$ = V_{IH}, Power Saver = V_{IH}			12	mA

ac electrical characteristics

T_A within operating temperature range, V_{LL} = 0V, V_{BB} = PROGRAM = V_{SS}, MM4204: V_{SS} = 5.0V ±10%, V_{DD} = −12V ±10%, MM5204: V_{SS} = 5.0V ±5%, V_{DD} = −12V ±5%, unless otherwise noted.

PARAMETER		CONDITIONS	MIN	TYP (Note 7)	MAX	UNITS
t_{ACC}	Access Time					
	MM5204	T_A = 70°C, *(Figure 1)*, (Note 4)		0.75	1.0	μs
	MM4204	T_A = 85°C, *(Figure 1)*, (Note 4)			1.25	μs
t_{PO}	Power Saver Set-Up Time					
	MM5204	*(Figure 1)*			1.8	μs
	MM4204	*(Figure 1)*			2.0	μs
t_{CO}	Chip Select Delay					
	MM5204	*(Figure 1)*			500	ns
	MM4204	*(Figure 1)*			600	ns
t_{OH}	Data Hold Time	*(Figure 1)*	30	50		ns
t_{ODC}	Chip Select Deselect Time					
	MM5204	*(Figure 1)*	30	300	500	ns
	MM4204	*(Figure 1)*	30	300	600	ns
t_{ODP}	Power Saver Deselect Time					
	MM5204	*(Figure 1)*	30	300	500	ns
	MM4204	*(Figure 1)*	30	300	600	ns
C_{IN}	Input Capacitance (All Inputs)	V_{IN} = V_{SS}, f = 1.0 MHz, (Note 2)		5.0	8.0	pF
C_{OUT}	Output Capacitance (All Outputs)	V_{OUT} = V_{SS}, $\overline{CS}$ = V_{IH}, f = 1.0 MHz, (Note 2)		8.0	15	pF

programmer electrical characteristics

$T_A = 25°C$, $V_{SS} = \overline{CS}$ = Power Saver = 0V, V_{LL} = 0V to −14V, unless otherwise specified, (see *Figure 2*), (Note 5).

	PARAMETER	CONDITIONS	MIN	TYP (Note 7)	MAX	UNITS
I_{LD}	Data Input Load Current	V_{IN} = −18V			−10	mA
I_{ALD}	Address Input Load Current	V_{IN} = −50V			−10	mA
I_{LP}	Program Load Current	V_{IN} = −50V			−10	mA
I_{LBB}	V_{BB} Load Current				50	mA
I_{LDD}	V_{DD} Load Current	V_{DD} = PROGRAM = −50V			−200	mA
V_{IHP}	Address Data and Power Saver Input High Voltage		−2.0		0.3	V
V_{ILP}	Address Input Low Voltage		−50		−11	V
	Data Input Low Voltage		−18		−11	V
V_{DHP}	V_{DD} and Program High Voltage		−2.0		0.5	V
V_{DLP}	V_{DD} and Program Low Voltage		−50		−48	V
V_{BLP}	V_{BB} Low Voltage		0		0.4	V
V_{BHP}	V_{BB} High Voltage		11.4		12.6	V
V_{DD}	Pulse Duty Cycle				25	%
t_{PW}	Program Pulse Width		0.5		5.0	ms
t_{DS}	Data and Address Set-Up Time		40			μs
t_{DH}	Data and Address Hold Time		0			μs
t_{SS}	Pulsed V_{DD} Set-Up Time		40		100	μs
t_{SH}	Pulsed V_{DD} Hold Time		1.0			μs
t_{BS}	Pulsed V_{BB} Set-Up Time		1.0			μs
t_{BH}	Pulsed V_{BB} Hold Time		1.0			μs
t_{PSS}	Power Saver Set-Up Time		1.0			μs
t_{PSH}	Power Saver Hold Time		1.0			μs
t_R, t_F	V_{DD}, Program, Address and Data Rise and Fall Time				1.0	μs

Note 1: "Absolute Maximum Ratings" are those values beyond which the safety of the device cannot be guaranteed. Except for "Operating Temperature Range" they are not meant to imply that the devices should be operated at these limits. The table of "Electrical Characteristics" provides conditions for actual device operation.

Note 2: Capacitance is guaranteed by periodic testing.

Note 3: Positive true logic notation is used except on data inputs during programming
Logic "1" = most positive voltage level
Logic "0" = most negative voltage level

Note 4: t_{ACC} = 1000 ns + 25 (N-1) where N is the number of devices wire—OR'd together.

Note 5: The program cycle should be repeated until the data reads true, then over-programmed 5 times that number of cycles. (Symbolized as X + 5X programming).

Note 6: The EROM is initially programmed with all "0's." A V_{IHP} on any data input B0—B7 will leave the stored "0's" undisturbed, and a V_{ILP} on any data input B0—B7 will write a logic "1" into that location.

Note 7: Typical values are for nominal voltages and $T_A = 25°C$, unless otherwise specified.

erase specification

The recommended dosage of ultraviolet light exposure is 6W sec/cm^2.

programming

The MM4204/MM5204 is normally shipped in the unprogrammed state. All 4096-bits are at logic "0" state. The table of electrical programming characteristics and *Figure 2* give the conditions for programming of the device. In the program mode the device effectively becomes a RAM with the 512 word locations selected by address inputs A0—A8. Data inputs are B0—B7 and write operation is controlled by pulsing the Program input. Since the EROM is initially shipped with all "0's," a V_{IHP} on any data input B0—B7 will leave the stored "0's" undisturbed and a V_{ILP} on any data input B0—B7 will write a logic "1" into that location.

programming (cont.)

National offers programmer options with both the IMP16-P and the PACE IPC-16P Microprocessor Development Systems.

Microprocessor System	Programmer Part Number
IMP16-P	IMP-16P/805
IPC-16P	IPC-16P/805

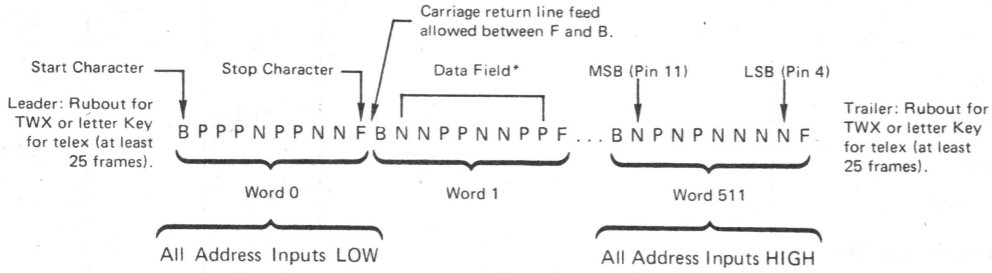
Contact the local sales office for further information. There are also several commercial programmers available such as the Data I/O Model V.

Most National distributors have programming capabilities available. Those distributors should be contacted directly to determine which data entry formats are available.

In addition, data may be submitted to National Semiconductor for factory programming. One of the following formats should be observed:

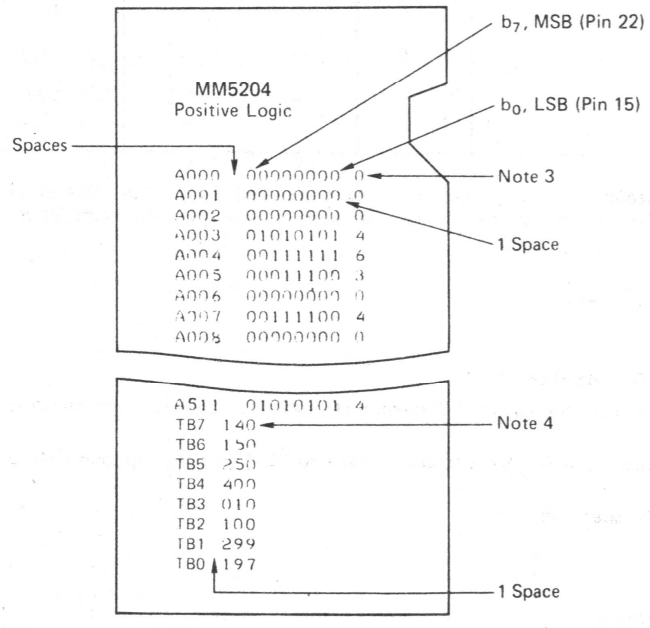
preferred format

The custom patterns may be sent in on a Telex or submitted as a paper tape in a 7-bit ASCII code from model 33 teletype or TWX. The paper tape should be as the following example:



*Data Field: Must have only P's or N's typed between B and F. No nulls or rubouts. Must have exactly eight P and N characters between B and F. Any characters except B and F may be typed between the F stop character and the B start character. If an error is made in preparing a tape the entire word including the B and F start and stop characters must be rubbed out. Data for exactly 512 words must be entered beginning with word 0.

alternate format [Punched Tape (Note 1) or Cards]



Note 1: The code is a 7-bit ASCII code on 8 punch tape. The tape should begin and end with 25 or more "RUBOUT" punches.

Note 2: The ROM input address is expressed in decimal form and is preceded by the latter A.

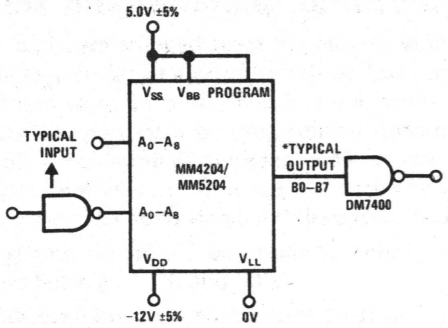
Note 3: The total number of "1" bits in the output word.

Note 4: The total number of "1" bits in each output column or bit position.

erasing procedure

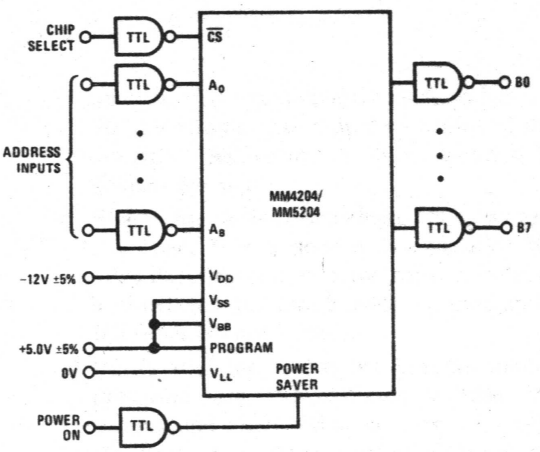
The MM4204Q/MM5204Q may be erased by exposure to short-wave ultraviolet light—253.7 nm. There exists no absolute rule for erasing time or distance from source. The erasing equipment output capability should be calibrated. Establish a worse case time required with the equipment. Then over-erase by a factor 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. Examples of UV sources include the Model UVS-54 and Model S-52 manufactured by Ultra-Violet Products, Inc. (5114 Walnut Grove Avenue, San Gabriel, California). The lamps should be used without short-wave filters. The MM4204/MM5204 should be placed about one inch away from the lamp for about 20—30 minutes.

ac test circuit

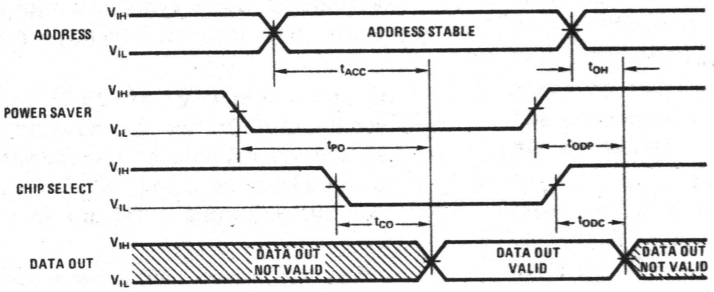


*t_{ACC}, t_{OH}, t_{CO}, and t_{OD} measured at output of MM4204/MM5204.

typical application



switching time waveforms



Note: All times measured with respect to 1.5V level with t_R and $t_F \leq 20$ ns.

FIGURE 1. Read Operation

programming waveforms

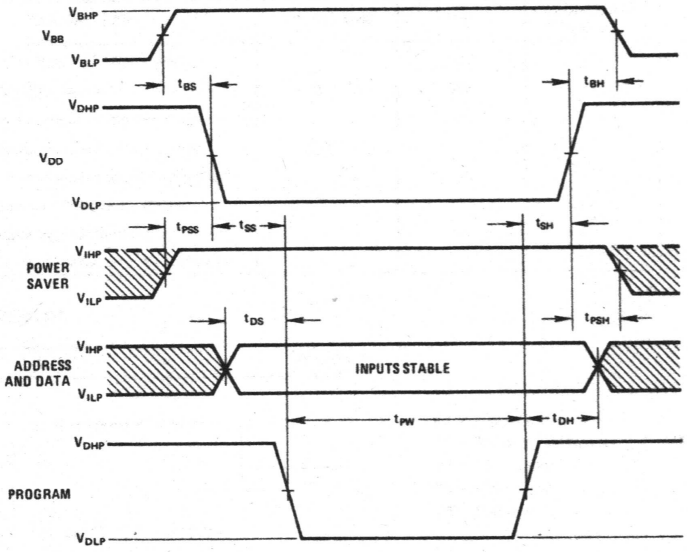


FIGURE 2. Programming Waveforms