Table of Contents

Model 9800 User's Manual
Document number &9800.PUB

Date: December 30, 1987

***** READ THIS IF NOTHING ELSE *****

- The end of the programming socket marked bottom locates the ground pin of the chip. This means that pin 12 on a 24 pin part goes at the bottom. So does pin 14 on a 28 pin part.

- Apply AC power before putting devices into the programmer.

- Do not attempt to read a masked ROM without checking to see if Vpp is applied during reads (Verify mode) for that part number.

- See information about baud rates and cables if the 9800 fails to communicate.

- This document contains user information on the GTEK Model 9800 Gang Eprom Programmer. Its contents are proprietary and may not be reproduced in whole or in part without the express written consent of GTEK, Inc. The information in this manual is provided "As Is" without warranty of any kind, either expressed or implied. GTEK, Inc. does not assume any liability for damages. Technical information and specifications included in this document are subject to Change Without Notice.

INTRODUCTION

Congratulations. You now have, what we believe to be, the most cost effective gang eprom programmer on the market today. The design philosophy used on the 9800 allows for simple future expansion of capabilities. It may be used as a stand alone production programmer, or via its RS-232 interface from a host computer or terminal. All serial communications with the 9800 is in printable ASCII characters and it supports Intel and Motorola hex formats as well as simple block formats. Additionally, the 9800 supports the MCS–86 extended hex format, and Motorola's S record format with features for automatically split programming 2 Eproms for use in a true 16 bit data path. Resident features include facilities for making source to eprom content comparisons, erasure checks, formatted device listings, menu driven device selection, and more.

The 9800's interrupt driven type ahead buffer allows it to program and verify in real time, while data is being sent (transparent to the user, whose sole responsibility is to send and receive data). Three user selectable algorithms are available, a standard 50ms program cycle with post verification, adaptive algorithms and QuickPulse algorithms.

QuickPulse algorithms are automatically selected on the nmos parts that are capable of using them. Not all manufacturers' parts can be programmed with the QuickPulse algorithm. If a part doesn't program with the QuickPulse algorithm, switch to the adaptive algorithm with a "TI" command. MCM68764's and MCM68766's also use an adaptive algorithm. Adaptive algorithms typically offer a six fold improvement in programming time over the standard algorithm. QuickPulse algorithms are about 10 times faster on the 9800 over the 7956's adaptive algorithms. Extended diagnostics pinpoint the cause of any errors.

Throughput is greatly enhanced by using parts which can be programmed with the QuickPulse algorithm. QuickPulse can program an Intel P27256 in 24 seconds. The adaptive algorithm can program the same part in 164 seconds. The standard algorithm, (if it were available for this part) would take 1638 seconds! The QuickPulse algorithm is not for all brands of eproms but probably can be successfully used on most of the newer technology 12.5 volt parts.

The Model 9800 may be used without handshaking, or with XON/XOFF or hardware CTS/DTR handshake. Baud rate selection is done automatically through your interface program or PGMX. The 9800 defaults to 2400 baud on power-up. PGMX is a program provided with the 9800 to allow you to read and program eproms and other devices on a PC/XT/AT at baud rates up to 57,600. See Appendix B for the (E)(E)Prom types that may be programmed.

All voltages and pin configurations are set up by the onboard microprocessor and no personality modules are required. ROMs may be read safely only with certain eprom selections, such as i27512, i68766, F27C64, F27C256 and 27C32. See Appendix A.

Getting Started Quickly!

Note that whenever it says to insert a part in any of the below examples, you should put the part in so that the notch on the part is towards the TOP of the programmer or where pin 28 or pin 24 is marked on the case. AND close the handle!

Stand Alone Mode

To begin programming quickly (without reading the rest of the manual), follow these directions for the stand alone mode. The RS-232 mode will follow in the next section.

First, "Power Up" the programmer. Always make sure that no chips are in the sockets on the programmer before applying power. When you apply power, if the rotary dip switches are on "00" or "01", then you will not be able to use the 9800 until you make a valid menu selection, either through the RS-232 or by setting the rotary dip switches. If the rotary dips are set to say, "27", then the programmer will be set up for that part, which is a 2764A.

Look up the eprom part number of your "master" chip in the appendix of this manual, or on the program disk. The master chip is the part that will be in the master socket of the programmer.

The eprom part number will usually be prefixed with a manufacturers symbol or letters, with a number following, usually starting with a "27", like iP27256 or MBM27C256. There may be a letter after the "27" number like 2764A or 2716B.

This letter may affect what menu selection to make for that eprom type, so always look for that extra letter! It will usually be an A, B or C. Determine the setting that will be used from the part listing in the appendix of this manual or from the program disk.

If you are using an Intel 2764A part, the setting will be "50" on the two rotary dip switches on the front of the 9800 to use the QuickPulse algorithm. Remember that the menu selection of the part determines what "programming algorithm" is used and the "programming voltage". The programming algorithm is the set of instructions built into the 9800

that determine what voltages to put on what pin, at what time. The programming voltage is the level of elevated voltage that is to be applied to the pin selected by the programming algorithm.

Selection of the wrong part number might cause you to destroy either your master and/or your "slave" chips. The slave chips are the chips that fit in sockets E0-E7 to the right of the master socket.

Once you have determined what selection to make for the master part, look up the eprom part number of your "slave" chips if it is different from the master chip, the same way you did for the "master" chip above. Your slave parts must have the same menu selection number as the master. If they are different, then read the part with the right selection to a disk file and then copy the file to the same eprom part number as your slave chips. This will avoid any possible damage to the master or slave parts and ensure the programmability of the slave parts.

If the selection is the same for the slave and master parts, then set the eprom menu selection by dialing the number from the manual or the disk. If the Ready led is on and the Busy led is off, you may now begin "loading" the programmer with your parts. Remember that if the Busy led is on, power is applied to all sockets, and removing or installing eproms at that time may damage them and/or the programmer. Always make sure the Busy led is OFF before removing and installing eproms. Do not turn the programmer on or off while parts are in the sockets for the same reason.

Put the master chip in the socket to the left by itself, and from 1 to 8 slave chips in the sockets numbered from E0-E7. If you use less than 8 chips, It doesn’t matter which sockets you use.

Now, optionally, you can check the slave parts to see if they are blank. You don’t have to do this if you don’t want to, but if any errors occur during programming you will wonder whether or not the part was really erased or not.

To verify, press the Verify button momentarily and release it. The Ready led will go off and the Busy led will come on indicating that the programmer is doing something. After a short period of time, the programmer will give a medium length, medium tone beep to indicate

it is done. The Ready led will come on and the Busy led will go off. If any eprom is not erased, the led above the suspect eprom will be on, indicating that part is not erased.

If the programmer gives a burst of 5 short, high pitched beeps, and then the ready light comes on, this indicates that one or more of the parts in the master or slave sockets is shorted and should be removed.

In any case, if you got 5 short beeps, one or more of your parts are shorted and should not remain in the programmer. If you are in a production mode, you should set all those eproms aside to be checked later, and continue on with a fresh set of eproms. See the Diagnostics chapter for information on determining which eprom is bad.

If you get verify error indications on 1 or more chips during verify (which is accompanied by a short high pitched beep for each failure), you can stop the verify process by pressing the Verify button once. This will cause all leds to light, so make a note of which chip did not verify before you press the button. You can then put a fresh chip in that location to try verifying again.

When the verify cycle ends, a medium length beep is issued to indicate the process is complete and the Ready led comes on and the Busy led goes off. Errors are indicated by the led above the suspect part.

Now, If all chips are blank (no error leds lit), you can press the Copy button to copy the master chip to the slaves. The Ready led should go out and the Busy led will come on. Again, if you get a series of 5 short high pitched beeps after pressing the Copy button, this indicates an overload condition that should be corrected before continuing. See the previous paragraphs about what to do about this.

During the copy process, copying errors are indicated by the led above the suspect eprom. A short beep will be issued every time one comes on. Those parts failed to program and should be removed after programming is complete. If all the slave parts fail to program, a series of beeps for each failure is issued, and the Ready light comes on and the Busy light goes off.

Usually, if all the slave eproms fail (to copy or to verify), you may have the wrong eprom type selected on the rotary dip switch or one of the parts is defective. If the dip selection is correct then set the (1-8) parts aside for checking later. If you had the wrong eprom type selected, you may have damaged up to 9 parts (including the master).

At the end of the copy cycle the 9800 will issue a medium beep followed by the Ready led on and the Busy led off to indicate that it is done. If any parts failed to program, the led above each suspect part will be lit. Remove all parts except the master if you are going to be programming some more parts. If you are done programming parts, you should also remove the master part, because you can damage the master if you leave it in the socket and remove power from the programmer.

Getting started quickly with the RS-232 mode.

First, "Power Up" the programmer. Always make sure that no chips are in the sockets on the programmer before applying power. If you have the rotary dip set to "00" or "01", your default eprom type will be null, and the default prompter will indicate "<xxxx>". This reminds you to make an eprom type selection on power up.

Look up the eprom part number of your "master" chip in the appendix of this manual, or on the program disk. The master chip is the part that will be in the master socket of the programmer.

The eprom part number will usually be prefixed with a manufacturers symbol or letters, with a number following, usually starting with a "27", like iP27256 or MBM27C256. There may be a letter after the "27" number like 2764A or 2716B.

This letter will affect what menu selection to make for that eprom type, so always look for that extra letter! It will usually be an A, B or C. Determine the setting that will be used from the part listing in the appendix of this manual or from the program disk.

At this point, or at any time after you apply power to the programmer, for that matter, you can communicate with the programmer. Use the PGMX program to do this. See the section on PGMX for specific details

on initializing PGMX and communications. If you are going to be using PGMX to select eprom types, you should set the rotary dip switches to "00".

Remember that the menu selection of the part determines what "programming algorithm" is used and the "programming voltage". The programming algorithm is the set of instructions built into the programmer that determine what voltages to put where, when. The programming voltage is the level of elevated voltage that is to be applied to the pin selected by the programming algorithm.

Selection of the wrong part number might cause you to destroy either your master and/or your "slave" chips. The slave chips are the chips that fit in sockets E0-E7 to the right of the master socket.

Once you have determined what selection to make for the master part, look up the eprom part number of your "slave" chips if they are different from the master chip, the same way you did for the "master" chip above. Your slave parts must have the same menu selection number as the master part in most cases! If they are different, then you have to adapt the master part to the socket or make a copy of the part with PGMX to the same part number as your slave chips. This will avoid any possible damage to the master or slave parts and ensure the programmability of the slave parts.

If the selection is the same for the slave and master parts, then set the eprom menu selection (while communicating with the programmer with PGMX) by typing the letter M plus the letter indicated by the selection. If you already know what you are doing but forgot what selection to make, type M plus a <cr> to get a menu of parts to select from. Make sure you know what you are selecting if you use the M<cr> method of selecting parts. It is easy to find say a 27128 in the menu, but if you don't know that there are 2 different types of 27128's then you have a 50-50 chance of making the right selection. As it turns out, there are 3 selections for 27128, one uses 21 volts another uses 12.5 volts, and the third uses the 27256 algorithm and 12.5 volts (N27C128). If you know your part uses 12.5 volts, you can simply make that 12.5 volt 27128 selection (if you know which selection is the 12.5 volt part, selection "2" in the case of a 12.5 volt part and not "F").

Once the menu selection is made, it will stay that way until you lose power or you make another selection.

If the Ready led is on and the Busy led is off, you may now begin "loading" the programmer with your parts. Remember that if the Busy led is on, power is applied to all sockets, and removing or installing eproms at that time will damage them and/or the programmer. Always make sure the Busy led is OFF before removing and installing eproms.

Put from 1 to 8 slave chips in the sockets numbered from E0-E7. Even if you use less than 8 chips, it doesn't matter which sockets you use.

Now, optionally, you can check the slave parts to see if they are blank. You don't have to do this if you don't want to, but if any errors occur during programming you will wonder whether or not the part was really erased or not.

To verify through PGMX, type the letter U and then return(<cr>). This will cause the programmer to check the entire part. If you only want to check part of it, type the letter U and then the starting and ending addresses to check, or you can press the Verify button momentarily and release it. The Ready led will go off and the Busy led will come on indicating that the programmer is doing something. After a short period of time, the programmer will give a short Beep to indicate it is done. The Ready led will come on and the Busy led will go off. If any eprom is not erased, the led above the suspect eprom will be on, indicating that part is not erased.

If the programmer issues a burst of 5 short beeps, this indicates that one or more of the parts in the slave sockets is shorted and should be removed.

Remove any parts that have an led lit above it, reload the socket and try verifying again until you have a full load (1 to 8) parts. At this point you are ready to begin programming.

Type Control - F (hold down the control key and press the letter F) and you will get a prompter to enter command line. A minimum command line consists of a <cr>, which will return you to the 9800 commmand prompter. To program a file, the minimum command line would consist

of a filename and a <cr>. You can also specify options on the command line, but probably not when you are programming manually like this.

To program from an Intel Hex file from the Control-F "enter command line" prompt, enter the file name (format: filename.HEX). You don't have to specify an extension unless you want to program from a BINARY file (format: filename.ext). In most cases it is probably an Intel Hex file you are using. Remember the interface program PGMX can't handle any other format than Intel Hex or Binary.

After you enter the filename, and you hit <cr>, PGMX will look for a file by the name you specified on the disk and begin sending it to the programmer. It will show only the load address that is being processed (in Intel Hex format), or only the number of bits programmed in the Binary format.

If an error occurs while programming any particular chip, the led will light above the suspect part and that part is skipped from that point forward. If all the parts being programmed fail, PGMX will abort sending the file and issue the error message that was sent from the programmer. If there is a problem condition with any of the eproms such as overload on Vpp or any of the low address or data pins are not Tri-State, a short burst of 5 beeps is issued and control is returned to PGMX or DOS (depending on where you started from, in this case from PGMX). An error message of *SC err @ aaaa will be issued with a short burst of 5 beeps.

See the Diagnostics section for Overload information.

When programming is done, a short beep is issued and control is returned to PGMX if that's where you started from. Any parts not programmed properly will have the led lit above the suspect part.

At this point, you may then reload the programmer and begin the process again.

Examples:

Example Stand Alone- to program 8 i2764A's from a master i2764A made by Intel:

1. Make sure the 9800 has power applied to it.
2. Look the part number up in the manual. It says to use #27 on the rotary dip switches. Dial in 27 on the rotary dips.
3. Insert the master part into the master socket.
4. Insert the slave parts into the slave sockets.
5. Press the Verify button once, wait for the Ready led to come on. If you hear the indicator beep that indicates shorts, abort as previously described.
6. Remove any parts that have the led lit above it and replace with a fresh part. Go back to step 5 again (until you have a verify cycle where no parts fail).
7. Press the Copy button once, wait for the Ready led to come on. If you hear the indicator beep that indicates shorts, abort as previously described.
8. When the Ready led comes on, any parts that failed to program properly will have the led lit above the suspect part. Set that part aside to erase and try again later. If a part fails to program after you have erased it and tried it again, you can assume that part is BAD. Don't try to use it any more.
9. If you want to verify that the slave parts compare to the master part, press both the Verify and Copy buttons together and release them. Both the Ready and the Busy led will light until the process is complete. Any part that fails to compare to the master will have the led lit above the suspect part. This is a "double check" of the parts, since the programmer continuously makes comparisons of the master part to each slave part during the program process.
10. Remove the programmed parts from the programmer. If you are going to use a different eprom part type next time, go to step 2. If you are going to use the same eprom part type go to step 4. Otherwise you are done.

Example RS-232- to read 1 2764 made by Hitachi (21 volt pgm voltage) and then program 1 2764A to be used later as a master chip in the previous stand alone example. Remember that a 2764A is a 12.5 volt part and may as well be considered as a completely different part number even though generically (in operation) they are identical parts. BUT they don't program the same!!!

1. Apply power to the programmer before you insert any parts.
2. Look the part number up in the manual. It says to use #05 on the rotary dip switches or Menu Selection number "E". Since you are going to set the programmer through RS-232, turn the rotary dip switches to "00".
3. Communicate with the programmer if you are not already in communication (at the DOS prompt type PGMX<cr>). Type "ME" at the PGMX eprom prompt. You could also dial in "05" on the rotary dips to select that part, but it is easier to leave the setting at "00".
4. Insert the Hitachi 2764 into slave socket E7. This is the only slave socket that the programmer can read to a disk file from, or display any data from.
5. Press ^F (hold down the control key and press the letter F) to get the "enter command line -->" prompter and type:

enter command line -->**FILENAME [R<cr>**

where FILENAME is what you want the file to be called on the disk. It will automatically have an extension of ".HEX". <cr> means to press the "enter" or "return" key. "R" means to read the file in the Intel Hex format (OI to the programmer). PGMX will automatically open the disk file (if one exists already, it will not let you destroy it) and cause the programmer to begin sending the content of the eprom in the Intel Hex format, which is then put into the disk file. When the programmer is done sending, PGMX will close the file and return you to the eprom type prompter.

6. Remove the Hitachi part and insert the Intel part into any slave socket. You can use E7 again if you like, but you don't have to in this case.

7. **You must now change the eprom type!** Looking it up in the manual says to use "1" so type "M1" to select "i2764A>" or it might say "q2764A>" depending on what you had selected before. It is ok to use the "q" or "i" on any brand part (at the eprom type prompt), but you are more likely to get an error programming using the "q" in the prompter if it is not an Intel part. Since this example uses an Intel part, we won't change it, but if for instance it was a GI or AMD part, you might want to type "TI" to select the adaptive programming algorithm (i) rather than the QuickPulse algorithm (q).
8. This part has to be blank, so press the Verify button once (or press U and return) to see if it is blank (as in the previous stand alone example). If the led above the part comes on, the part is not blank. Repeat with another part until you find one that is blank.
9. If it is blank, press ^F (as before) to get the "enter command" prompter and type:

enter command line -->**FILENAME<cr>**

This will cause PGMX to look for FILENAME.HEX on the disk and begin sending it to the programmer. All empty slave sockets (7) will fail with an error led lit, and the single part will begin programming.

10. If the part fails during programming, the led above it will light and a short beep will be issued along with an error message of what went wrong and the location (like *WP err @0000 or *SC err @1abc). See the Diagnostics chapter in that case. Otherwise, if programming is completed without an error light above the part, the part is properly programmed and verified at this point. If the Ready led is on and the Busy led is off, then you can remove the eprom and go to the Stand Alone example to use the part as a master. Otherwise you are done. Note that you could have programmed up to 8 parts at this time instead of just 1.

Example RS-232- programming 8 chips from the file made by the previous example, using Fujitsu 27C64's.

1. Make sure power is applied before inserting any parts.
2. Communicate with the 9800 with PGMX.
3. Select the correct eprom type to use by looking up the part number in the manual. In this case it is menu selection "O". Type "MO" at the eprom type prompter:

<q2764A> **MO**
<F27C64>_

Notice that a <cr> is not necessary!

4. Insert the parts to be programmed into the slave sockets (E0- E7).
5. Press U<cr> to verify for blank on all the parts for the entire part. If there are any error leds lit, put a new part in the socket and press U<cr> again otherwise procede to next step. Note that pressing U on the computer keyboard and pressing the Verify or copy pushbutton are mutually exclusive. You cannot terminate a Verify or Copy pushbutton press with the "$" command, and you cannot terminate a U<cr> keypress with the Verify or Copy pushbutton.
6. Type ^F (hold down control and press F) to enter file name to send to the programmer.

enter command -->**FILENAME<cr>**

Will begin sending FILENAME.HEX to the programmer. If any error occurs, the led above the suspect part will be lit. When PGMX is finished, you are returned to the eprom part type prompter.

7- You are done. To program another set the same way, go back to step 3 or step 4.

Stand alone- set part types without being hooked to RS-232:

- Turn dip switches to the number indicated for the part in the manual: 50 for < q2764A> , 27 for < i2764A> , 28 for < i27128A> , 05 for <i2764>, 15 for <2764>, etc.

RS-232- set part types through RS-232 (and/or PGMX):

- turn dip switches to "00", select part as indicated by the manual. ME for <i2764> (and TQ for <q2764>), MF for <i27128>, M1 for <q2764A> (TI for i2764A), M2 for <q27128A>, etc...

You SHOULD read the rest of the manual to get specific details about some of the operations performed above, specifically the COMMANDS chapter, the DIAGNOSTICS chapter and the PGMX chapter.

COMMANDS

The following are the commands that can be used on the programmer. Most people that use PGMX will not have to use any of these commands except for the "Toggle" and "Menu" commands. They are given here so that people who can't use PGMX can still have all the functionality of PGMX. Commands that are taken care of transparently to the user by PGMX are so noted. That means "don't worry about this one" unless you don't have PGMX!

[H] = Handled by PGMX's command mode or Immediate mode

[N] = Not a function of PGMX, but can be used in Immediate mode

[I] = Immediate mode only

: Intel Hex Program [H]

When in the command state, receipt of a colon is interpreted as the lead character in an Intel hex record. The 9800 automatically enters the program mode and programs the data contained in the hex record at the address specified in the header of the hex record. The check sum is verified at the end of the hex record and the programmer then returns to the command state but does not reissue the command prompter unless the record happened to be the END record. This is done in anticipation of another hex record, i.e., all characters from the hex file, sent to the Model 9800 will be echoed back to the user with no additions or deletions. Power to the sockets is not turned off until an end record or error occurs. (Busy off, Ready on)

The error light above any socket which fails to program properly will light. Remember to clear the error lights before sending your file with the "X" command. A short beep will be issued each time an error occurs. If a data error, checksum error, or syntax error occurs during the file transfer, the programmer will issue the appropriate error message and abort back to the command state. A *WP or *NE error will be issued only if all eight slaves fail.

See the section on toggles and hex formats for clarification on how to program two devices for device use on a true 16 bit data bus. The segment base address register, maintained by the 9800, is automatically cleared when the end record is detected, or if any other command is executed other than the Intel Hex command. Remember that you do not have to "split" a hex file if you have a 27210 (16 bit data path).

S Motorola Hex Program [N]

This command functions precisely the same way that the Intel hex program command does, except the format is the Motorola S record format. Records may be of type S0, S1, S2, S3 OR S9.

/ Tektronix Hex Program [N]

When in the command state, receipt of a forward slash is interpreted as the lead character in a Tektronix hex block. The 9800 automatically enters the program mode and programs the data contained in the hex block at the address specified in the header of the hex block. The checksums are verified at the end of the hex block and the programmer then returns to the command state but does not reissue the command prompter unless the block happened to be the termination block. This is done in anticipation of another hex block, i.e., all characters from the hex file, sent to the Model 9800 will be echoed back to the user with no additions or deletions.

R Block Read (socket E7 only) [N]

(Don't confuse this command with PGMX's R command.)
The R command, followed optionally by beginning and ending addresses, causes the Model 9800 to output a continuous string of ASCII-HEX characters between the specified addresses. If no addresses are specified, the 9800 will output the entire contents of the selected device. The R command may be aborted at any time by sending a dollar sign, "$", to the programmer. The following example uses the eprom programmed in the example of the P command.

Example:
<2716> **R444,445<cr>**
3323
<2716>_
Note: The R command is primarily for automated reading of eproms.

If you execute the command line as shown in the above example, you will find that the data output overwrites the command line unless your terminal is in an auto line feed mode.
(Eg.3323_R444,445)

OI Intel Hex File Output (FROM E7) [H]

(This is the command that PGMX uses to Read a file.)
The OI command has the same command syntax as the R command. It differs in that the 9800 will output the device contents as an Intel hex file, including the end record, between the specified addresses inclusive or if no addresses are specified, the entire device. Again, the command may be aborted if desired with a dollar sign, "$".

OM Motorola Hex File Output [N]

The OM command functions precisely the same way the OI command does, except that the output is in the Motorola S record format.

OT Tektronix Hex File Output [N]

The OT command works the same way as the OM and OI command does, except that the output is Tektronix Hexadecimal Format.

L List Formatted Output [N]

The L command outputs the data, between optionally specified addresses, inclusive, in a formatted fashion similar to many dump utilities. If no addresses are specified, the entire contents will be listed and the command may be aborted with the dollar sign, "$". Each line of the listing includes the beginning address in ASCII-HEX, sixteen data bytes in ASCII-HEX and the ASCII REPRESENTATION of the data. Non printable bytes are replaced with periods in the ASCII representation field.

Example:

```
<2716>  L90,AF<cr>
0090  4845  4C4C 4FFF FFFF FFFF FFFF FFFF 99FF HELLO. . . . . .
00A0 FFFF FFFF FFFF FFFF FFFF FFFF AA55 AA55 . . . . . .
2716>_ [prompter indicates end of command]
```

Note: Unlike the R, OI, OT and OM commands, the L command will output a carriage return and line feed at the beginning of the listing. This is because the L command is primarily used when the host is functioning as a terminal and it would be irritating to have the first line of the listing overwrite the command line.

U Unerased (Blank) Check [H]

The U command works exactly the same way as pressing the Verify button. It checks the entire part for blank if a starting and ending address is not specified. If a part is not erased, the led above the suspect part will light, indicating that part is not erased. An empty socket will look like a blank part. If all 8 sockets are not erased, then the function fails with an *NE err @nnnn where nnnn=the last address checked that was not blank. The command can be aborted with a dollar sign, "$". If any error leds are lit and before this command is issued, then those leds are automatically cleared.

V Verify Erasure Check (from socket E7 only) [H]

The V command checks the cells between the optionally specified addresses for erasure, FF's or 00's as the device type dictates. If no addresses are specified, the entire device is checked. If a non erased cell is encountered, the led above that particular socket will light and a short beep will be sounded. Remember to reset the leds before issuing the "V" command (with an "X" command). Any messages refer to the Slave socket next to the Master socket. The process continues until the end address is reached or the command is aborted with a dollar sign, "$". The programmer is left in the "compare" mode if all 8 sockets are not blank. The following example uses the same eprom used in the P and R command examples.

Example:
<2716> **V<cr>**
33 @ 0444
23 @ 0445
<2716>_

M Menu Selection (dips=00) [H]

You may select the device you will be working with in 2 ways. The current device type always becomes part of the command prompter. Selecting a device establishes the programming algorithm to be used, as well as the device pinout, proper programming voltage and prompter.

Instructions:

A) Direct Selection Method (types 02-99):

- 1) Move selector switches to the position desired from the decimal column from the table in appendix B of this manual. If you are communicating with the programmer, the new current eprom type will be displayed every time you move the selector switch.

B) Software Selection Method:

- 1) Move selector to position "00" and leave it there.
- 2) Press "M" on keyboard.
- 3) Then press the code letter for the device you want or <CR> to get a displayed menu.

See the Appendix section on Manufacturer's cross reference to correlate your part number with the appropriate eprom type selection.

Software Selection Method:- This is an example only!

(dip switches set to 00)

<2732> **M<cr>**

```
EPROMSELECTIONMENU
     NMOS        NMOS        CMOS        EEPROM      W/ADAPT
A– 2758     G– AM2716B                  P– 5213     R– 874x–1K
B– 2716     H– AM2732B                  Y– I2816A   S– 874x–2K
C– 2732A    I– 2532                     3– I2817A   T– 874xH–1K
D– 2732A    +– TI2532A                  Q– X2816A   U– 874xH–2K
E– 2764     ^– TI2732A  "– N27C128     9– X2864A   V– 8751
1– 2764A    J– 2564     8– F27C256     4– X28256
F– 27128    K– 68766    .– N27C256                  W–8755
2– 27128A
Z– 27256    %– F27256
7– 27512
#– 27513
=– 27011
ENTER SELECTION >2
<i27128A>_
```

M<cr> Results in the 9800 giving you a menu of parts to select from. Refer to the appendix parts list for help in selecting the correct part. At that time, enter the menu selection number and the prompter will reflect the part number selection that you made, or dial in the right selection.

Note: The dip has precedence over the software select. If you set a device by software, you can reset it with the dip. Positions 02-99 have precedence over the hardware select position 01. If you select one of positions 02-99, the programmer will output the new selection to the terminal. To make a menu selection via RS-232 the switches have to be set to "00".

Also, an *RT err @ 0000 is issued if a menu selection is attempted but you had the rotary dip switches set to something other than "00". An M<cr> will allow you to see a menu selection, but will issue the *RT error when you make a selection.

TC Toggle Compare Mode [H,I]

The TC toggle command is used to turn the compare mode on and off. When in the compare mode, the command prompter is prefixed by a lower case c. The compare mode is used to compare the contents of a device against that of a source file. To use the compare mode, use the TC toggle to turn on the compare mode. Then use one of the various programming commands as if you were going to program the device. Instead of programming the device, the 9800 will make a comparison of the source byte to the contents of the device. If they are not the same, the comparison error will cause the led above that particular device will be lit and the programmer will continue to check the other eproms. See Diagnostics Section for details. The TR toggle or a Menu selection will cause the TC toggle to be reset as well as another TC command.

TD Toggle Dumb Algorithm Mode [H,I]

The TD toggle command is used to turn on/off the "standard" programming algorithm for the selected part. If their is no standard algorithm, then a *UV err @nnnn error message is issued.

TH Toggle High Byte Split Mode [H,I]

The TH toggle puts the 9800 into a split mode used for programming 2 eproms whose intended destination is for use in a true 16 bit data path environment. While in the TH split mode, the command prompter is prefixed by a lower case h indicating that the high (Odd Address) byte is set. See the TL and TR command below. Note that the split mode works on either Intel Hex type files or Motorola S record type files. It is not functional from the "P" command.

TL Toggle Low Byte Split Mode [H,I]

The TL toggle puts the 9800 into a split mode used for programming 2 eproms whose intended destination is for use in a true 16 bit data path environment. While in the TL split mode, the command prompter is prefixed by a lower case l indicating that the low (Even Address) byte is set. See TH command above and the TR command below. The split mode works on either an Intel Hex or Motorola S record Hex file, but not from the "P" command.

TI Toggle Intelligent Algorithm Mode [H,I]

The TI command turns the intelligent programming algorithm on/off. Typing TI for a device that does not use the intelligent algorithm will cause an error message *UV err @nnnn to be issued. Some parts default to the intelligent algorithm.

TN Toggle Checksum [H,I]

The TN command is used to generate a 16 bit checksum from the data in the eprom. This is the 16 bit sum of all the (8 bit) DATA bytes added together without carry. You may make a checksum between any two addresses by specifying the Hex starting and ending addresses. The checksum is calculated for all 9 sockets and then output to the user. See examples of this in the PGMX chapter.

TQ Toggle QuickPulse Algorithm Mode [H,I]

The TQ command selects the QuickPulse algorithm for the selected part. Some part types default to the QuickPulse algorithm and is the only algorithm supplied for that part, so typing TQ on those will result in a *SN err or a *UV err.

TR Reset TC, TH and TL Toggles [H,I]

The TR command resets the TH and TL and TC toggles. To reset these and any other toggles set, reselect the part type with the Menu command.

' ' Reissue Command Prompter [H,I]

Sending a space (ascii 32 char) to the programmer causes it to reissue the command prompter.

I Identify Serial Device

The Model 9800 will issue a serial number and re-output the command prompter in response to an I. This may be used by automated programs which need to have the prompter transmitted to them. Sending the 9800 a carriage return or a space yields almost the same results.

X Clear Error Lights and Return Version [H,I]

The X command is used to clear the error lights above the chips before programming or verifying a chip via software. It will also clear any segment register offset, issue a Logon message and the prompter. The X command will return the following:

<2716> **X**
GTEK , INC.
MODEL 9800 Vx.xx
COPYRIGHT 1987
<2716>_

And the leds will be turned off. When ordering accessories from GTEK, please remember to include the version and serial number.

$ Abort to prompter [H,I]

A $ sent to the programmer will abort most operations.

DIAGNOSTICS

General

Most diagnostics are handled by PGMX. The person that is using PGMX need only be concerned with the meaning of any error message that is issued by PGMX. Other information here is for persons not using PGMX.

1) All error codes to be issued by the 9800 are preceded by an asterisk, "*". This makes error trapping very easy.

2) When a non-fatal error, such as won't program, need erasing or compare occurs during programming, (WP, NE, CP) the error light for the associated chip lights.

3) FATAL errors are output on a real time basis, that is, they are output as soon as they are detected, and the programmer returns to the command state.

4) Fatal Error codes include the address at which the error occurred.

5) The error lights are cleared only when you issue the "X" command. This is so that you can issue more than one command, such as the Verify and Program, so that the error lights will be cumulative. All error lights turn on when a command is executing from the Verify or Copy pushbutton and the command is aborted by pushing either Copy or Verify. Note the position of any displayed error light before you abort a command this way.

Fatal Error Codes

***WP ERR @ nnnn Won't Program error:** This error is issued only in the event that all eight slave sockets have errors and the 9800 discovered that it could not change the data in the chips, even though the bits were not already set. When using the QuickPulse algorithm, you will not get any *NE errors, only *WP since the 9800 does not "pre-read" the cells prior to programming.

***NE ERR @ nnnn Needs Erasing error:** This error is issued only in the event that all eight slave sockets have errors and the 9800 discovered that it could not change the data in the chips, and the bits were already set. You will never get an NE error with the QuickPulse algorithm, because the 9800 does not pre-read cells to be able to tell that a bit was not set previous to programming.

***CP ERR @ nnnn ComParison error:** Issued during comparisons and verifies, but only if ALL eight slaves fail.

***DT ERR @ nnnn DaTa error:** The character that was sent is not valid hex data. (0-9 or A-F) This error message is issued as soon as it happens.

***CS ERR @ nnnn Check Sum error:** Issued if a checksum error is detected in a hex record. Only applies to Intel, Motorola, and Tek hex format program commands. This error message is issued as soon as it happens

***SN ERR @ nnnn SyNtax error:** An invalid command was issued to the programmer. This error message is issued as soon as it happens. See COMMANDS section.

***ST ERR @ nnnn STack error:** FIFO overflow. Reduce baud rate or see the interfacing section for handshaking methods. (The 9800 can take data at 300 bps with no handshake.) PGMX users may not be using the right RS-232 cable. This error message is issued as soon as it happens.

***UV ERR @ nnnn Un-aVailable error:** Issued in the event the user tries to use a function of the programmer that is not available for that particular device. This error message is issued as soon as it happens.

***RT ERR @ nnnn error:** Issued in the event the user tries to make a menu selection through the RS-232 (PGMX) and the rotary dip switches are not set to "00". This error message is issued as soon as it happens.

***SC ERR @ nnnn error:** Issued in the event that one of the low address or data lines are not Tri-State or Vpp is drawing excessive current. This error message is issued as soon as it happens.

Non-Fatal Errors

Non-fatal errors are indicated only by the LEDs, no message is output to the console. These errors are considered non fatal in that the process continues, i.e. you don't want to stop programming eight eproms just because one has failed.

During programming, the error lights indicate that the chip failed to program. The eprom may have needed erasing, may be no good, the wrong device type was selected or the device was missocketed.

During erasure verification, error lights mean that the chip is not erased.

During comparisons, error lights mean that the eprom contents differ from the source.

Overload Conditions

If a programming voltage overload condition occurs, the programmer will issue a short burst of 5 high pitched beeps, possibly indicating an overload on the programming voltage pin. An *SC error can occur at any time, but a problem with the low address or data lines will usually show up immediately.

The programmer will abort to the command state and issue a *SC ERR @ nnnn and light all the error lights, since the programming voltage is cut short during the last programming cycle.

Remove the shorted part(s) or make the proper selection for the chips you are programming or correct the condition that caused the overload and CAUTIOUSLY try them again. Wait a short period of time before attempting to program again. Pressing the Copy or Verify button more than twice without correcting the overload condition could possibly damage the 9800. Shorted Vcc pins will not cause the same kind of overload. Chips will usually fail on the first byte with this kind of overload, without the *SC beeps.

To find a part causing the problem out of 8 parts, take out 4 chips and try again. If it still errors out, take 2 chips out and try again. If it still errors out that means that one of the two parts is bad. Take 1 out and try again.

If it still errors out, that must be the bad chip; put the remainder of the chips back into the programmer and begin again. The Master chip could cause the problem if it is a 28 pin part.

Remember that the Master Socket may have programming voltage applied to pin 1. No other pin of the Master Socket gets programming voltage. This means that you may NOT copy from a 2764A to a 2764, unless you have taken action to prevent programming voltage from

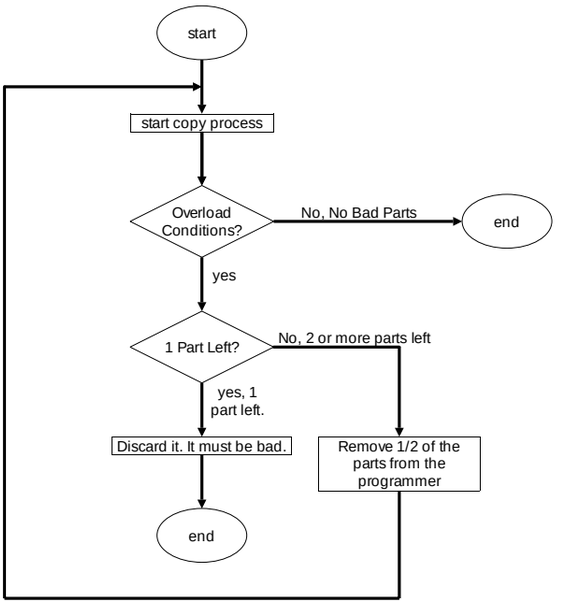


Figure 4.1

Finding a single bad part during the program process

being applied to pin 1 of the 2764A. You can also get around the problem by copying the 2764A to a file and then making a 2764 master chip.

INTERFACING NOTES

Persons using PGMX can ignore this chapter!

The Model 9800 is surprisingly easy to interface and there are several methods of handshaking which can be utilized if it is desired to operate at the higher baud rates. The following section describes some of the methods.

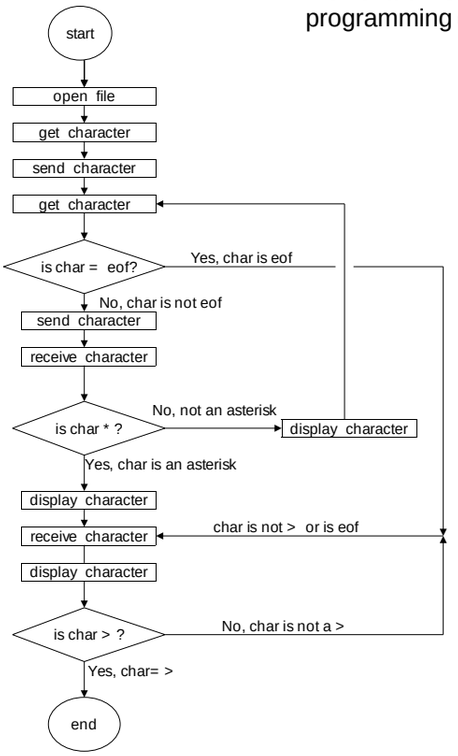
1. Software handshake. This is perhaps the easiest method of all. When you begin to send data to be programmed, send the first byte but don't wait for it to be echoed. That would effectively cut your communication rate in half. Instead, send the second byte, receive the first, send the third byte, receive the second, etc. This technique will allow you to program as fast as the algorithm in use permits. Some devices program faster, some slower! See an example of this in Fig. 5.1.

2. CTS/DTR hardware handshaking. The Model 9800 is configured as data terminal equipment, which means that the CTS (clear to send) line is an input to the programmmer which when pulled low forces the programmer to stop sending. On the other hand, the DTR (data terminal ready) line is an output from the programmer, which will go low when the buffer is about 50% full and high again when the buffer is about 38% full. If you are using hardware handshake and the DTR line goes low, you should stop sending to the 9800 within about 2 character periods (before XOFF is sent). The RTS line is pulled high whenever the programmer is plugged in. See Specifications for Cable.

3. Xon/Xoff software handshaking. If you do not monitor the DTR line, the 9800 will transmit an XOFF character if the buffer gets to be about 63% full. If an XOFF has been sent, an XON will be sent when the buffer level drops to about 25% full. Likewise, when the programmer is sending you data, you may send an XOFF character, which will stop the programmer from sending until it receives an XON character. XON's and XOFF's, are not put into the buffer, but are processed as soon as they are received. Even if you don't use XON/XOFF handshaking, you will find it useful when using the L, list command, to stop and start the data flow to your screen. XON and XOFF are the keyboard equivalents of control-Q and control-S respectively.

4. Please note that the 9800 may communicate at many different baud rates. To initialize at the new baud rate, send the 9800 a break signal (set the output data line on your computer to +12 volts) for 100 milliseconds, set the break to normal again (-12 volts) and wait 40 milliseconds. After the 40 milliseconds is up, send an 80H character to the 9800 at the new baud rate. The 9800 will begin reissuing the prompter in response to the space or return command when locked on again.

Figure 5.1
Flowchart showing a programming example.



AUTOMATION HINTS

When you automate the transfer of data from your computer to the 9800, you should examine the echoed characters to see if an asterisk, "*" has been sent. If you receive one, it means that an error message will follow and that the programmer will return to the command state. Any automation software should take this into account.

The effective addressing range of a device is determined by it's size. If a 2K byte device is being used, then it only has 11 significant address lines and only the lowest 11 bits of the address field are significant. Thus, as far as the 9800 is concerned, 000H is equivalent to 800H or F000H in a 2K device.

You don't need to compare the characters that are echoed to what you sent. The characters are echoed to the host as they are removed from the FIFO, and would not reflect a programming error. However, the 9800 will detect any programming error and the host need only trap the error message. The PGX utilities for CP/M and MSDOS based computers send echoed characters to the screen (console). PGMX, due to its high baud rates, does not attempt to display all the information being transferred unless you specify that with the "d" option on the command line. Error messages are displayed when they occur whether or not the "d" option is specified.

The programmer is in the command state after the prompter is sent. The prompter always ends with a '>'. You can use this character to let your program know that an R, OI, OM, OT, V, or L command has finished.

You should probably have one mode of operation where you communicate directly with the programmer (turn your computer into a terminal). This will give you easy use of the L, V, P, and M commands.

SPECIFICATIONS

DIMENSIONS: (H x W x D)
2.5" x 12.0" x 7.5"
(63.5mm x 304.8mm x 190.5mm)

POWER:
120VAC, 60HZ, 25 VA (240Vac, 50Hz, option)

INTERFACE:
DB25P - data terminal equipment (see below).
DATA WORD:
1 Start, 8 Data, 1 Stop, No parity

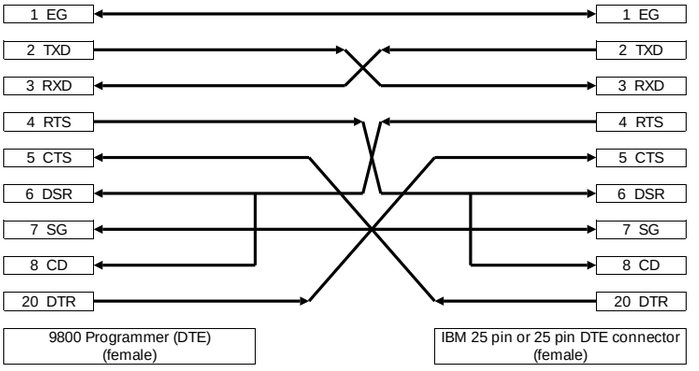
BAUD RATE:
Auto select 300-9600, 19200, 28800, 57600
(Rates above 9600 depend on your computer being able to keep up)

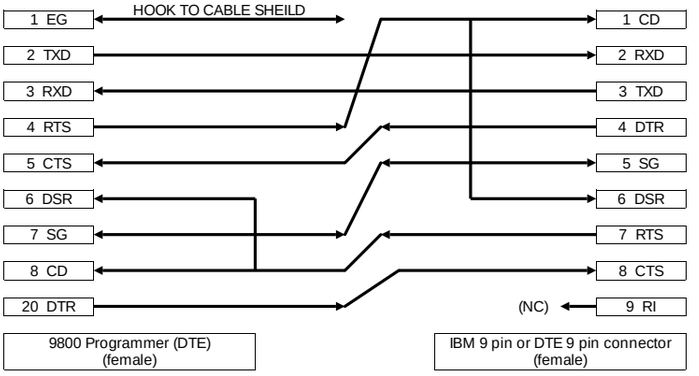
WEIGHT:
5 Pounds (2.4 KG)

OPERATING ENVIRONMENT:
45 - 95 DEG F. (7 - 35 DEG C.)
5% TO 95% non-condensing relative humidity

MAKING A CABLE:

9800 DTE	to	DTE	or	DCE
1- Equip Ground (EG)	> <	1 (EG)		1 (EG)
2- Transmit Data (TXD)	>	3 (RXD)		2 (TXD)
3- Receive Data (RXD)	<	2 (TXD)		3 (RXD)
4- Ready To Send (RTS)	>	6 (DSR)		4 (RTS)
5- Clear To Send (CTS)	<	20 (DTR)		5 (CTS)
6- Data Set Rdy (DSR)	>	4 (RTS)		6 (DSR)
7- Signal Ground (SG)	> <	7 (SG)		7 (SG)
20- Data Term Rdy (DTR)	>	5 (CTS)		20 (DTR)





HEX FORMATS

Intel Format

Data Record

Byte Number	
1	Colon (:)
2-3	Number of binary data bytes
4-5	Load address, high byte
6-7	Load address, low byte
8-9	Record type
10-x	Data bytes, 2 ASCII-HEX characters
x+ 1 - x+ 2	Checksum, two ASCII-HEX characters
x+ 3 - x+ 4	CR,LF

End Record

Byte Number	
1	Colon (:)
2-3	Record length, must be "00"
4-7	Execution address
8-9	Record type
10 - 11	Check sum
12 - 13	CR,LF

Extended Address Record (MCS-86 hex format)

Byte Number	
1	Colon (:)
2 - 3	Record length, should be "02"
4 - 7	Load address field, should be "0000"
8 - 9	Record type, must be "02"
10 - 13	USBA
14 - 15	Check sum
16 - 17	CR,LF

Start Address Record (MCS-86 hex format)

Byte Number	
1	Colon (:)
2 - 3	Record length, "04"
4 - 7	"0000"
8 - 9	Record type, "03"
10 - 13	8086 CS value
14 - 17	8086 IP value
18 - 19	Check sum
20 - 21	CR, LF

The checksum is the **two'scompliment** of the 8-bit sum, without carry, of all the data bytes, the two bytes in the load address, and the byte count.

MOTOROLA FORMAT

Comment Record

Byte Number	
1 - 2	"S0"
3 - n	Comment field
x+ 1 - x+ 2	CR,LF

Data Records

Byte Number	
1 - 2	"S1"
3 - 4	Number of data bytes + 3.
5 - 6	Load address, high byte.
7 - 8	Load address, low byte.
9 - x	Data bytes, 2 characters each.
x+ 1 - x+ 2	Checksum.
x+ 3 - x+ 4	CR,LF.

Byte Number	
1 - 2	"S2"
3 - 4	Number of data bytes + 4. (2 characters)
5 - 10	Load address, 24 bits (6 characters)
11 - x	Data bytes, 2 characters each.
x+ 1 - x+ 2	Checksum (2 characters).
x+ 3 - x+ 4	CR,LF.

Byte Number	
1 - 2	"S3"
3 - 4	Number of data bytes + 5.
5 - 12	Load address, 32 bits (8 characters)
13 - x	Data bytes, 2 characters each.
x+ 1 - x+ 2	Checksum
x+ 3 - x+ 4	CR,LF.

End Record

Byte Number	
1 - 2	"S9"
3 - 4	CR,LF.

In the above S records, the byte count includes the load address and checksum. Thus the byte count is equal to the number of data bytes plus the following; 3 for S1, 4 for S2 and 5 for S3 type records. The checksum is the **one's compliment** of the 8-bit sum, without carry, of the byte count, the two bytes of the load address, and the data bytes.

TEKTRONIX HEX FORMAT

Data Blocks

Byte Number	
1	Header (which is a forward slash- /)
2 - 5	Location counter which is 4 ascii-hex characters representing the load address of the data bytes.

6 - 7	Byte count which is 2 ascii hex bytes specifying the number of binary data bytes in the data field of the block.
8 - 9	First Checksum, which is 2 ascii-hex bytes specifying the HEX SUM of the values of the previous six digits. (location counter and the byte count)
10 - X	Binary data bytes which are each represented as 2 ascii-hex digits. (in other words 16 binary bytes are represented as 32 ascii-hex bytes.)
X+ 1 - X+ 2	Second Checksum. 2 ascii-hex bytes representing the SUM, modulo 256 of the binary values of the ascii data bytes. (8 bit sum without carry.)
X+ n	Always a carriage return. (CR)

Termination Block

Byte Number	
1	Header (forward slash /)
2 - 5	Transfer address which is the address for execution of code.
6 - 7	Byte count, always 00 for a termination block.
8 - 9	Checksum of the six digits that make up the transfer and byte count.
10	Always a carriage return. (CR)

Abort Block

Byte Number	
1	Header forward slash /
2	Header forward slash /
3 - X+ 69	Message up to 69 characters for error information etc.
X+ 70	Always a carriage return. (CR)

Example of Data block and 1 Abort block

/0000101001020304050607080 90A0B0C0D0E0F0038
//THIS IS AN ERROR MESSAGE HERE

Note: programmer will issue a *DT error on the second "/" mark and return to the command state without displaying the abort message...

Example... of Data block and 1 Termination block

/0000101001020304050607080 90A0B0C0D0E0F0038
/00000000

NOTE: Most terminals will display Tektronix data only on one line, since the format calls for only a carriage return at the end of a record.

USAGE OF GHEX.EXE

GHEX.EXE is a program provided for you to be able to convert a binary file into an INTEL.HEX file. This capability is built-in to the PGMX.COM program, but you may want to use it for convenience.

General usage is:

C> **GHEX filename.ext<cr>**

OR

C> **GHEX filename.ext offset <cr>**

Offset is an ASCII-HEX number (0-9 and/or A-F) that specifies where you want your code to begin in the HEX file.

C> **GHEX filetest.bin<cr>**

Will result in an INTEL.HEX file being created on your disk by the name filetest.hex. The load addresses begin at 0000H since no offset was specified. GHEX does not destroy the input file.

C> **GHEX filetest.bin AA55<cr>**

Will result in an INTEL.HEX file being created on your disk by the name filetest.hex, just like before except the load addresses start at AA55H.

GHEX is provided as a convenience now , since the PGMX program can handle transferring in Intel Hex or Binary code. PGMX will also handle any offsets into the code too.

One thing you have to remember when using GHEX is that any code that you run GHEX on should be an exact multiple of 128. If your binary file is not an exact multiple, then GHEX will fill out to an even multiple of 128 with nulls.

USING DEBUG.COM

You may use DEBUG.COM (supplied with PC-DOS) in conjunction with our GHEX.EXE to modify an INTEL.HEX file without worrying about the checksums in the INTEL.HEX file.

The following is a short tutorial to modify a 4K byte INTEL.HEX file with DEBUG. The procedure is to run DEBUG first.

C> **DEBUG<cr>**
—_

From the - prompter within DEBUG use the N command to specify the name of your INTEL.HEX file.

—**Nfilename.HEX<cr>**
—_

Use the L command to load the hex file with an offset (if it begins at 0000H). You must do this since if it starts loading at 0000H within the segment, it will overwrite your file control block at 5Ch.

—**L 100<cr>**
—_

The CX register now contains the number of bytes read into memory with an offset of 100. You may have to modify the CX register to properly reflect the correct number bytes you must write back to the disk. Remember that this is going to write from CS:CX when you issue the command.

—**RCX<cr>**
CX: **1000<cr>**
-_

Your data is now loaded into the memory of the computer at offset 100H. Use the E command to modify the bytes you need to modify. An example of modifying locations starting at 0A55H with data is shown. Locations A55H through A57H contain FFH.

—**EA55 01 02 03<cr>**
—_

Now specify a new file name to write to the disk with since you can't use an extension of HEX with the file you are writing. You want to call it a BIN or IMG file instead since that is what the data really is anyway.

—**NNEWFILE.BIN<cr>**
—_

Now you can use the Write command to write the new data to the disk. DEBUG will write an exact image of CS:CX bytes to the disk starting at an offset of 0100H bytes.

—**W<cr>**
Writing 1000H bytes
—_

Now use GHEX to make it an INTEL.HEX file, or use PGMX's binary file transfer.

Using Interface Program PGMX

Installation of PGMX

PGMX is a high speed communication program which runs on IBM PC/XT/AT's. It allows flexible manipulation, transmission and reception of Intel HEX files and binary files.

On the PGMX program disk you will have at least 3 programs: PGMX.COM, PINSTALL.COM and GHEX.EXE. PGMX is the program used to communicate with your 9800. PINSTALL is the program that you must run to install the serial drivers in PGMX so that you can communicate with the 9800. Other programs and document files are provided to allow conversion from Motorola format to Intel hex and other programs to split and interleave to and from 8, 16 and 32 bit binary formats.

If you try to run the PGMX program without installing the serial drivers, it will tell you to run the PINSTALL program. Remember that the PGMX license is a single user license.

Insert GTEK program disk in drive A: and copy the programs to your hard disk with:

C> **COPY A:*.***

This will copy all the programs on the GTEK disk over to the subdirectory that you are logged on to on your hard disk. If you don't have a hard disk, use DISKCOPY or COPY to the B: drive. Refer to the DOS manual for specific instructions on using the COPY command. The desired end result is a backing up of the original GTEK copy. Store the original program disk in a safe place.

Now you should insert the backup copy in the drive A: and/or go to the subdirectory where PINSTALL and PGMX are located. You must first run the PINSTALL program to install the serial drivers for PGMX.

C> **PINSTALL<cr>**

After the copyright and version number appears, you are asked to select a letter which corresponds to the type of installation you wish to perform.

Most people will probably select to set up to communicate at 19,200 baud on computer serial port COM1: or the selection for 19,200 baud on COM2:.

IRQ4 is used in conjunction with an interrupt service routine for COM1: when PGMX is invoked if you installed it for COM1:. This is a hardware line on your PC to give the system an interrupt whenever a character is received. If you know that something else in your computer is using this hardware interrupt line, then you should use the other com line, which uses IRQ3 (COM2:).

IRQ3 is also used in the same manner for COM2: when PGMX is invoked if you installed it for COM2:. If you know something in your system uses IRQ3 for interrupts, then you must use the other com port.

The next selection that you have to make is where your line printer is located, on parallel port 1, 2, or 3 (lpt1:, lpt2: or lpt3:). This has to be done so that PGMX knows where to send printed data.

After you have made that last selection, you are returned to the DOS command prompter and PGMX is set up to run under those conditions that you specified.

See the example for C>**PGMX<cr>** later in the manual.

OPERATION

PGMX is a "command driven program" as opposed to a "MENU driven program" which means that everything you do is done by entering a "command" on the command line instead of "selecting" the command from a menu. This makes the program very fast when you have learned what the commands are.

In most cases the commands are exactly the same command as what the programmer is expecting, so the selection of the command is somewhat intuitive.

There are 2 ways that commands may be given to PGMX:

1- From the PC or MS DOS command line.

2- From within PGMX.

Commands executed from DOS return to DOS upon completion. Commands executed from within PGMX return to PGMX upon completion. Command lines may be entered from within PGMX by depressing control F.

EXAMPLES:

C> **PGMX<cr>**

Enter PGMX and establish communication with the programmer (assuming everything is hooked up properly).

C> **PGMX FILENAME<cr>**

Results in communication being established with the programmer and sending FILENAME.HEX (Intel Hex Format) from the disk to the programmer. When PGMX is through, you are returned to the DOS system prompt.

C> **PGMX FILENAME [OPTIONS]<cr>**

Results in PGMX establishing communication with the programmer, and then performing according to selected options.

Programming the eprom in binary or Intel Hex format or Reading the eprom in the same formats may be accomplished by giving the proper options. OPTIONS are always enclosed in square brackets and separated by comma's. Invalid commands result in an appropriate and descriptive ERROR message.

VALID OPTIONS

R	read file. (default is program mode)
%ooooo	binary mode select (default is HEX)
@sssss - eeeee	Eprom bounds
Mx	menu selection
Tx	toggle command (3 max on command line)
Vsssss-eeeee	verify erasure
D	display data as it is being received from the 9800

MORE EXAMPLES

PGMX<cr> from the DOS command line establishes communication with the 9800, and after log-on displays the 9800 Command Prompter, which is the currently selected eprom type.

LOG ON MESSAGE EXAMPLE

(remember these are examples and your display may not be exactly like this one!)

C> **pgmx<cr>**
High Speed Interface Package Version 9.33
Copyright 1983, 1984, 1986, 1987 GTEK, INC.
All Rights Reserved, worldwide.
I/O Hardware Driver Vers 1.01 - IBM PC/AT
Serial port - COM1, 19,200 bps
Printer port - LPT1:

GTEK,INC.
MODEL 9800 V1.00
COPYRIGHT 1987

<xxxx>_

The programmer is ready and waiting for a command at this point. If you want to do a Menu command, pressing an **M** and the code necessary will select an eprom type or press **M<cr>** to get a menu:

2732> **M<cr>**

```
     NMOS        NMOS        CMOS        EEPROM      W/ADAPT
A–  2758    G–  AM2716B                 P–  5213    R–  874x–1K
B–  2716    H–  AM2732B                 Y–  I2816A  S–  874x–2K
C–  2732A   I–  2532                    3–  I2817A  T–  874xH–1K
D–  2732A   +–  TI2532A                 Q–  X2816A  U–  874xH–2K
E–  2764    ^–  TI2732A  "–  N27C128    9–  X2864A  V–  8751
1–  2764A   J–  2564     8–  F27C256    4–  X28256
F–  27128   K–  68766    .–  N27C256                W–8755
2–  27128A
Z–  27256   %–  F27256
7–  27512
#–  27513
=–  27011
EPROMSELECTIONMENU
Enter Selection —>2
i27128A>_
```

WARNING! Do not use this **example** to select parts from. Use Appendix B. Parts are removed and added from time to time!

Results in the programmer giving you a menu of parts to select from. Refer to the appendix parts list for help in selecting the correct part. At that time, enter the menu selection number and the prompter will reflect the part number selection that you made, or dial in the right selection.

i27128A> **TN< cr>**
C000 C000 C000 C000 C000 C000 C000 C000 C000
i27128A> _
(Master, E7, E6, E5, E4, E3, E2, E1, E0)

Results in the programmer giving you a 16 bit addition of all the 8 bit bytes of all the part, without carry. Blank 27128s give you C000 for the checksum.

i27128A>(**control-F**)

Control- generally means to press and hold the CONTROL key on your keyboard and press a command letter. Valid command letters are P, F and C. The ESCape key is also a valid control command key, but you do not hold the control key down to press ESC. The ESC key is a valid control character already. The escape control command may also be obtained by pressing CONTROL [on the IBM keyboard or by holding down the ALT key and entering 027 on the numeric keypad. Pressing and holding the CONTROL - C key for instance is represented by a caret and the letter that must also be pressed, eg. ^C.

The definitions of the CONTROL commands are:
^P -start sending / stop sending (toggle) data simultaneously to the printer.
^F -enter a command line. Examples follow.
^C -Abort most programmer commands and return to the DOS or PGMX command prompter. This command will work even though you may be in the process of programming, reading, verifying, etc., an eprom in the automated (control-F) mode.

ESC or ^[- Escape from program. This command is used as an alternative to control-alt-del and is not normally used. This is an EMERGENCY command and the results could be unpredictable.

USING control F

2716> **^F**

Enter Command line --->**FILENAME [@0-1FF,V,TN<cr>**

Results in PGMX doing a blank check on the eprom between 0 and 1FF inclusive. Then FILENAME.HEX is opened and any hex data falling between the specified boundaries is sent. During data transfer, PGMX displays the load addresses of the hex records that it is sending. Finally, the checksum is calculated between the specified addresses and displayed.

The options are always set off by an opening square bracket ([) and the ending square bracket (]) is optional. Invalid commands result in an error message and a return to the 9800 command prompter.

DEFINITIONS

Please note that the listed commands are generally passed on to the programmer unchanged except for the order in which they appear in the command line. PGMX will send the commands specified to the programmer in the following order:

1 - menu command
2 - toggle commands (except TN is done last)
3 - blank check or verify erasure
4 - program or read
5 - checksum (tn)

Some commands, particularly the "R" command, work differently from the 9800 command "R". The "%" and the "@" command are not valid commands for the 9800 except on the PGMX command line. They are used to give PGMX information, not the 9800. You may not specify any command more than once inside the brackets except the toggle commands, and you are allowed a maximum of 3 of those.

sssss = 24 bit starting address, Hex characters (0-9 and A-F).
eeeee = 24 bit ending address, Hex characters.
ooooo = 24 bit offset amount, Hex Characters

A delimiter is a dash (—), a comma (,), a space ( ), a carriage return, or a line feed (ascii characters 2Dh, 2Ch, 20h, 0Dh or 0Ah). Carriage return and line feed are re presented by a <cr> or <lf>.

A FILENAME is a valid DOS filename to be used by PGMX to look for a file on the disk. In the case where a percent (%) sign is specified, the filename specified will be taken literally. In other words you must be explicit and give the extension of the filename also. If the percent sign was not specified then PGMX will automatically supply a .HEX extension and look for a .HEX even if you specified an extension.

An EXT is a valid DOS extension for the filename in your directory. You are allowed to use any extension you wish here, (in the binary % mode) and the data will be sent to the programmer UNCHANGED. The EXT will only be valid when you have specified a percent sign (%) within the brackets.

AND REMEMBER!
The effective addressing range of a device is determined by it's size. If a 2K byte device is being used, then it only has 11 significant address lines and only the lowest 11 bits of the address field are significant. Thus, as far as the 9800 is concerned, 000H is equivalent to 800H or F000H in a 2K device.

VALID COMMANDS FOR PGMX

1- Any valid programmer command except OI, OM, OT, R.

2- @sssss-eeeee. An @ symbol followed by the starting address (ssss) followed by a dash (-) followed by the ending address (eeee) will cause PGMX to search through the specified FILENAME to find the specified locations inclusive to be sent to the 9800. In the case of a binary file (specified by a % on the same command line only), the @ symbol means that the data specified by the % sign (offset), will go to the ssss-eeee specified by the @ sign within the eprom, and eeee less ssss bytes will be sent. In the case of an Intel Hex file (no %), the @ symbol means that PGMX will search the Intel Hex file for data located between the start address (ssss) and the end address (eeee) inclusive, and send that data to the same locations within the eprom.

3- %ooooo. A percent sign (%) followed by an offset (you may omit specifying an offset of 0, but PGMX may warn you that you did not specify it, just in case you forgot) will cause PGMX to treat the EXTension you specified literally (and not add a .HEX extension). Any offset you specify (ooooo) will cause PGMX to scan up to that location in the file before sending any data to the 9800.

EXAMPLES:

To program 3 2716's from a binary file that contains 1093H bytes:

xxxx> **MB**
2716> **^F**
Enter Command line -->**TEST.BIN[%0,@0-7FF<cr>**

Causes PGMX to look for a file called TEST.BIN on the disk, and when found start sending from relative offset location 0 within TEST.BIN to locations 0 through 7FFh within the eprom. The number of bytes sent

is the number of bytes between 0 to 7FFh inclusive. If you don't specify boundaries, you will "Wrap Around" to location 000H at location 800H because you are still sending data to the programmer through PGMX.

2716> **^F**

Enter command line-->**TEST.BIN[%800,@0-7FF<cr>**

Causes PGMX to look for a file called TEST.BIN on the disk, and when found start sending from relative offset 800H from within TEST.BIN to locations 0 to 7FFh within the eprom.

2716> **^F**

Enter command line ->**TEST.BIN[%1000,@0-7FF<cr>**

Causes PGMX to look for a file called TEST.BIN on the disk and when found start sending from relative offset 1000H from within the TEST.BIN to locations 0 through 7FFh within the eprom. However, the program will terminate when it encounters the end of the file you are sending from, since there are only 94H bytes left in the file TEST.BIN left to send.

Reading an eprom to a disk file is accomplished with the 'R' option.

C> **pgmx filename[r<cr>**

Results in reading the selected eprom to the Intel hex disk file, FILENAME.HEX.

C> **pgmx filename[r,%<cr>**

Results in reading the selected eprom to a binary disk file whose name is FILENAME. (no extension was specified.). Notice an offset value included with the % has no meaning during a read operation. Use the @ command to read between specified locations within an eprom.

C> **pgmx [tn,ma< cr>**
or
2716> **^F**
enter selection —>**[tn,ma <cr>** (from within PGMX)

Results in selecting 2758 (note menu selection has side effect of resetting all toggles) and calculating the checksum.

ADVANCED EXAMPLE

C> **pgmx filename[mz,ts,u,tn,@20000-2FFFF**

Results in selecting 27256, split mode, doing a blank check, programming the eprom with hex data residing between the 20 bit addresses of 20000 and 2FFFF inclusive, and calculating it's checksum.

This particular file is big. Don't be afraid that PGMX has hung up. It has to check the load addresses of every record in the file, and it would take a minute before it reached records at load address 20000, unless the file was created with an "exotic" compiler in such a manner that segment records with apparently random addresses are placed at apparently random locations every few records in the file. No joke intended.

The boundaries specified cover a 64k range, but the eprom is only 32k. The reason for this is that in the split mode, the 2 eproms are considered as one eprom of twice the size. However, if an error message is issued during programming in the split mode, the address given by the error message is the physical address in the single eprom.

Batch file automation

Automating the process could be accomplished with a batch file such as this:

TEST.BAT

```
pgmx  test.bin[mb,u,@0-7ff,%%0,tn
pause remove eprom, insert new blank
pgmx  test.bin[u,@0-7ff,%%800,tn
pause remove eprom, insert new blank
pgmxtest.bin[u,@0-7ff,%%1000,tn
echo now you are done.
```

(use 2 percents (%%) in a batch file)

Error return codes for batch file processing:

These error return codes may be used by a calling batch file or process which drives a chip handler like those manufactured by EXATRON.

ERCODE=1 for any 9800 error messages (like *NE, or *WP)
ERCODE=2 for PGMX aborted by user with control-C
ERCODE= 5 for PGMX aborted by a disk error like "file not found"
or disk full or any command syntax error like "option error"
ERCODE=6 for PGMX when it was expecting a response from the 9800 and a timeout occured before any response was received.

ERROR.BATprogram:

```
echo off
pgmx %1
if errorlevel 6 goto :lostcom
if errorlevel 5 goto :syssner
if errorlevel 2 goto :abort
if errorlevel 1 goto :badpart
echo This part programmed ok.
goto :enbat

:lostcom
echo You have lost communications with the programmer
goto :enbat

:syssner
echo There is a disk system error or a syntatical error.
echo Example, PGMX cannot find the file you specifyed or
echo  you are trying to use a command that does not exist
echo  or if you are reading a file maybe the disk is full!
goto :enbat

:abort
echo Someone typed a control C while the file was transferring. The
echo   program has been aborted.
```

```
goto :enbat

:badpart
echo The Eprom programmer issued an error such as *WP or
echo *NE or *DT or any other error which it might issue.
echo In any case you should reject echo this part.
goto :enbat

:enbat 1
echo  done
```

The above batch file will allow you to automatically program an eprom and abort if there are any problems. Add to it any other commands or programs nececssary for your specific application.

Other programs available:

Note the following programs are written so you can compile them easily with QuickBasic. We don't guarantee these programs to be error free, but they should present no problem to the experienced user.

CBIN.BAS:
A program to calculate a checksum from a binary file. The file must contain the exact number of bytes that fits in the eprom for you to get the same checksum as the TN command will give you (unless you specify boundaries with TN).

CHEX.BAS:
A program to calculate a checksum from an Intel Hex file. The file must contain data for every byte in the eprom. A file that does not fully program the eprom will not give the same checksum as the TN command unless you know what part and how much of the eprom is not programmed.

INTR16.BASandINTR32.BAS
Programs to combine 2 (intr16) or 4 (intr32) 8 bit BINARY files into 1 binary file.

SPLIT16.BASandSPLIT32.BAS
Programs to split 1 BINARY file into 2 (split16) or 4 (split32) binary files to program sets of eproms. The 9800 already has a 16 bit split mode,

and it may be faster to split the 16 bit file with the 9800 since basic runs so slowly (unless it's compiled). You would have to split a 32 bit file with the basic program first to obtain 4 8 bit files.

S_TO_HEX.COM
Program to take a Motorola Hex file and convert it to an Intel Hex file. It takes input from the keyboard and outputs it to the console. To modify whole files, use the DOS redirection commands:

Example: C>S_TO_HEX <moto.mik >intel.hex
will take a Motorola mik or ptp file by the name of MOTO.MIK and convert it to an Intel hex file by the name intel.hex.

WARRANTY AND SERVICE

LIMITED WARRANTY

GTEK, INC., warrants to the original purchaser of this GTEK, INC., product that it is to be in good working order for a period of 1 year from the date of purchase from GTEK, INC., or an authorized GTEK, INC., dealer. Should this product, in GTEK, INC.'s opinion, malfunction during the warranty period, GTEK will, at its option, repair or replace it at no charge, provided that the product has not been subjected to misuse, abuse, or non GTEK authorized alterations, modifications, and / or repairs.

Products requiring Limited Warranty service during the warranty period should be delivered to GTEK with proof of purchase. If the delivery is by mail, you agree to insure the product or assume the risk of loss or damage in transit. You also agree to pre-pay the shipping charges to GTEK.

ALL EXPRESS AND IMPLIED WARRANTIES FOR THIS PRODUCT INCLUDING, BUT NOT LIMITED TO, THE WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, ARE LIMITED IN DURATION TO THE ABOVE 1 YEAR PERIOD. Some states do not allow limitations on how long an implied warranty lasts, so the above limitations may not apply to you.

UNDER NO CIRCUMSTANCES WILL GTEK, INC. BE LIABLE IN ANY WAY TO THE USER FOR DAMAGES, INCLUDING ANY LOST PROFITS, LOST SAVINGS, OR OTHER INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OF, OR INABILITY TO USE, SUCH PRODUCT. Some states do not allow the exclusion or limitation of incidental or consequential damages for consumer products, so the above limitations or exclusion may not apply to you.

THIS WARRANTY GIVES YOU SPECIFIC LEGAL RIGHTS, AND YOU MAY ALSO HAVE OTHER RIGHTS WHICH MAY VARY FROM STATE TO STATE.

The limited warranty applies to hardware products only.

SERVICE

For warranty service or non warranty service, contact GTEK, INC. at (601) 467-8048 to obtain an RMA (Return of Material Authorization number). We will need the serial number and date of purchase along with the invoice number or a copy of the old invoice. Send the programmer, freight prepaid to:

GTEK,INC.
RMA Number #####
399 Highway 90
Bay St. Louis, MS 39520

Be sure to include the RMA on the shipping label and in the package so we will know what to do with it. Out of warranty service charges are determined on an hourly labor plus materials basis.

PGX AND PGMX SOFTWARE LICENSE AGREEMENT

"This software is a proprietary product of GTEK, Inc. It is protected by copyright and trade secret laws. It is licensed (not sold) for use on a single micro-computer system, and is licensed only on the condition that you agree to this LICENSE AGREEMENT." GTEK, INC. provides this program and licenses its use worldwide. You assume responsibility for the use of this software to achieve your intended results, and for the installation, use and results obtained from the software.

LICENSE

The Licensee may:
a. use the program on a single machine;
b. copy the program into any machine readable or printed form for backup or modification purposes in support of your use of the program on the single machine;
c. modify the program and/or merge it into another program for your use on the single machine (Any portion of this program merged into another program will continue to be subject to the terms and condiions of this Agreement.): and,

d. transfer the program and license to another party if the other party agrees to accept the terms and conditions of this Agreement. If you transfer the program, you must at the same time either transfer all copies whether in printed or machine-readable form to the same party or destroy any copies not transferred; this includes all modifications and portions of the program contained or merged into other programs. You must reproduce and include the copyright notice on any copy, modification or portion merged into another program.

YOU MAY NOT USE, COPY, MODIFY, OR TRANSFER THE PROGRAM, OR ANY COPY, MODIFICATION OR MERGED PORTION, IN WHOLE OR IN PART, EXCEPT AS EXPRESSLY PROVIDED FOR IN THIS LICENSE. IF YOU TRANSFER POSSESSION OF ANY COPY, MODIFICATION OR MERGED PORTION OF THE PROGRAM TO ANOTHER PARTY, YOUR LICENSE IS AUTOMATICALLY TERMINATED.

TERM

The license is effective until terminated. You may terminate it at any other time by destroying the program together with all copies, modifications and merged portions in any form. It will also terminate upon conditions set forth elsewhere in this Agreement or if you fail to comply with any term or condition of this Agreement. You agree upon such termination to destroy the program together with all copies, modifications and merged portions in any form.

PGMX LIMITED WARRANTY

THIS PRODUCT IS NOT A CONSUMER PRODUCT WITHIN THE MEANING OF THE UNIFORM COMMERCIAL CODE AND APPLICABLE STATE LAW. THE PROGRAM IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU (NOT GTEK, INC.) ASSUME THE ENTIRE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION. SOME STATES DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO THE

ABOVE EXCLUSION MAY NOT APPLY TO YOU. THIS WARRANTY GIVES YOU SPECIFIC LEGAL RIGHTS AND YOU MAY ALSO HAVE OTHER RIGHTS WHICH VARY FROM STATE TO STATE.

GTEK, Inc. does not warrant that the functions contained in the program will meet your requirements or that the operation of the program will be uninterrupted or error free. However, GTEK, Inc. warrants the diskette(s) on which the program is furnished, to be free from defects in materials and workmanship under normal use for a period of ninety (90) days from date of delivery to you as evidenced by a copy of your receipt.

Licensee herein acknowledges that the software licensed hereunder is of the class which inherently cannot be tested against all contingencies by Licensor. Licensee acknowledges Licensee's obligation to test all programs produced by the licensed software to determine suitability and correctness prior to use.

LIMITATIONS OF REMEDIES

GTEK, Inc.'s entire liability and your exclusive remedy shall be:
1. the replacement of any diskette(s) not meeting GTEK's "Limited Warranty" and which is returned to GTEK, Inc. with a copy of your receipt, or

2. if GTEK, Inc. or the dealer is unable to deliver a replacement diskette(s) which is free of defects in materials or workmanship, you may terminate this Agreement by returning the program and your money will be refunded.

IN NO EVENT WILL GTEK, INC. BE LIABLE TO YOU FOR ANY DAMAGES, INCLUDING ANY LOST PROFITS, LOST SAVINGS OR OTHER INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE SUCH PROGRAM EVEN IF GTEK, INC. HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, OR FOR ANY CLAIM BY ANY OTHER PARTY.

SOME STATES DO NOT ALLOW THE LIMITATION OR EXCLUSION OF LIABILITY FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES SO THE ABOVE LIMITATION OR EXCLUSION MAY NOT APPLY TO YOU.

GENERAL

You may not substitute, assign or transfer the license or the program except as expressly provided in this Agreement. Any attempt otherwise to sublicense, assign or transfer any of the rights, duties or obligations hereunder is void.

This Agreement will be governed by the laws of the State of Mississippi.

Should you have any questions concerning this Agreement, you may contact GTEK, Inc. by writing to:

GTEK, Inc.
Sales and Service
P. O. Box 2310
Bay St. Louis, MS 39521–2310 U.S.A.

Appendix A- Introduction

Parts in the following list are listed by manufacturer that can be programmed on the 9800. In most cases you probably could use the "generic" selection of that part except for the notable exceptions of the 27256. Notice! On the model 9800, some menu selections are different from the other GTEK eprom programmers.

If you don't see your part on the list, you may send a data sheet to GTEK or try calling GTEK to see if we can tell you about a particular part. BE SURE to have a data sheet handy when calling unless you have not been able to obtain one, in which case we may or may not be able to tell you if it will program or how to program it.

GENERAL RULES

1- "A" and some "B" version parts program at lower voltages than the standard parts. If you try to Program, Verify, or List or Output an "A" or "B" part using a "standard" selection or the incorrect algorithm, the part will probably die within microseconds due to overvoltage on the programming pin. The part will appear to be OK and may even still contain any data that you had previously programmed in it, but the symptom will be *WP ERR @ nnnn. This goes for the MPU's also.

2- CMOS eproms generally use different algorithms to program than the NMOS parts, but if the voltage is the same, you might try the NMOS equivalent if you want to try programming the part adaptively (a lot faster).

3- ROMs are generally readable on the programmer if you take precautions to not use a selection that is going to use the Verify mode to read it. If you're not sure, simply use a spec sheet for the menu selection / part you would like to use and check the Vpp pin during reads (OI or L) to see if programming voltage appears there. This is done with NO part in the socket of course. Generally the CMOS part selections and the 27512 and 68766 do not use the Verify mode, only the Read mode. This may not always hold true on the 9800.

ROM equivalents of MPU's may only be read after modification of the programming socket of recalibration of the programmer. You must call GTEK for details of this.

4- ROMs may be masked to use what would be address lines on eproms as chip select lines. This means that they would address or enable the part in a low condition instead of a high condition as with an address line. This means that sections of the data might be swapped as you read it. It could also mean that the part has no eprom equivalent!

Manufacturer's Cross Reference vs Menu Selection

GTEK believes that the information contained in this list is correct. However, GTEK assumes no responsibility or liability for the accuracy of this list.

These Notes are beside the Eprom selections on the following pages:

1–Standard algorithm only is available for this part, which is typically 50ms. EEproms may use a 10ms algorithm as the standard algorithm.

2–Adaptive algorithm only is available for this part, which is typically 1ms X total number of cycles to program (15-25) + overprogram pulse of 3 or 4 times the total number of cycles.

3–Can use Standard or Adaptive algorithm. Some parts may not program with Adaptive algorithm. The selection shown is for the recommended algorithm. If your part will not program, try the alternative algorithm. The 2764A comes up with the Quick Pulse algorithm set. Try "TI" if it won't program at some location. The programmer will not let you select an algorithm that is not possible to use with the part, like the "dumb" algorithm for the 2764A for instance.

4–Use Model 482 adapter with this part and selection on the 7956 or the 9800 programmer. Uses Standard algorithm only. Use Model 481 with GTEK's other programmers.

5–Use Model 484 adapter with this part and selection. Uses Adaptive algorithm only. Programming the security byte on the 8742AH chip is accomplished by programming data FFH at location FF1FH.

6–Use Model 512 adapter with this part and selection. Uses Standard algorithm only. Programming the security byte on the 8751 or 8744 chip is accomplished by programming data 00H at location 0FFFFh. The data in location 0FFFh in the 8751 may be anything but zero, or else the security byte will not program.

7–This Fujitsu 12.5 volt algorithm selection is different from the Intel selection by the use of the -CE pin.

8–na

9–Use Model 756 adapter with this part and selection. Uses Standard algorithm only.

10–This CMOS part would normally use the standard algorithm. You can use the NMOS equivalent part selection to program this part Adaptively (TI command).

11–These parts are programmed using a 705 adapter (programmer) through programming a 2732 or 2732A on a GTEK (or any) programmer and then putting the 2732 into the 705, so the 68705 can copy the data from the 2732 into the Eprom of the 68705.

12–The "C" after the part number in this case deontes the case style. There are other "C" parts now available that program with Intelligent or quick pulse algorithms with 12.5 volts rather than the 21 volts of this part. Using the 21 volt selection will damage a 12.5 volt part beyond use. If you have a "C" part, call GTEK for details on programming it.

13–These parts may REQUIRE the Adaptive algorithm. TI started producing chips using a fast algorithm without changing their part numbers. You may not be able to determine which algorithm to use with these parts. To be safe, always use the Adaptive algorithm with these parts. Programming with the dumb algorithm might damage the part.

14–Use current model 512 adapter (REV-B) to program this part, using the Quick Pulse algorithm. Using Menu selection V will destroy the 87C51! Make SURE you use the M$ selection.

Use the following chart and your Intel Spec Sheet to program the security on an 87C51:

0000H - 0FFFH	Code Area. Put your program here...
2000H - 201FH	Encryption Data. Put the KEY bytes here...
6000H	Signature Byte. 89H = Intel...
6001H	Signature Byte. 57H = 87C51...
8000H	Lock Bit 1. Program FFH here for Lock 1.
E000H	Lock Bit 2. Program FFH here for Lock 2.

It’s not possible to read the encryption table. You can verify it by reading the programmed code and massage it with your encryption key. You can then compare the results with your original code that you programmed the part with. If Lock Bit 2 is set you will not be able to read or program the part at all, until you

erase it. If Lock Bit 1 is set then you can still read the part, but you can't program it any more until you erase it.
If you're using a model 512 adapter purchased before July of 1986, make sure that it is modified to be a REV-B for use with the Model 9800. If it is not modified you will not be able to program the encryption or lock bits. The modification does not affect the operation of the 512 with other GTEK gang programmers.

15–Comment not applicable to model 9800.

16–Use the Model 110 adapter on 27010 parts (32 pins). Use the Model 210 adapter on 27210 parts (40 pins).

17–Quick Pulse is the default algorithm for this part, although the adaptive programming algorithm is also available.

18–Quick Pulse is the only algorithm you can use with this part.

GTEK believes that the information contained in this list is correct. However, GTEK assumes no responsibility or liability for the accuracy of this list.

To select a part using a GTEK programmer, do the following: From the default power up prompter you type "M" and then you can either type the selection or hit return:

<xxxx> **ME**
<i2764> **M**

Menu appears here. Make your part selection.

enter selection --> **E**
<i2764>_

The part you select can usually use several different algorithms. Different GTEK models use different algorithms on power-up default. The 7228 uses the standard algorithm as default unless you change it on all parts capable of the adaptive algorithm, except for the parts that require the adaptive algorithm like the 2764A. For instance a 2764 in most cases can be programmed adaptively. On the 9800, you can usually speed up programming by also selecting the QuickPulse algorithm for that part (by typing TQ at the prompter).

<i2764> **Tq**
<q2764> **ME**
<i2764>

Some parts default to the QuickPulse algorithm, like 2764A for example:

<i2764> **M1**
<q2764A>_
and the Td command will not work for this part, since it can only use the Adaptive or QuickPulse algorithm. It is safe to use the "Ti" and "Tq" command for all parts that can use it. You can't select "Ti" or "Tq" if the part can't use it. The 9800 uses the QuickPulse algorithm default for the 2764A. (q2764A_)

In the list below under the menu selections, the suggested algorithm is shown. When using the interactive terminal mode or PGMX to select a part, a "ti" might also be shown. That means to also select the intelligent algorithm for that part. A "td" or a "tq" would mean to select that algorithm to program the part. The rotary dip switch method shows the direct selection of the correct algorithm.

AMD
EPROMS: N=nmos, C=cmos

Part #	Volts	Type	Menu	Select	Size	notes
AM2716	25.0	N	mb	53	2K	3
AM2716B	12.5	N	mg	20	2K	2
AM2732	25.0	N	mc	16	4K	3
AM2732A	21.0	N	md	17	4K	3
AM2732AP	21.0	N	md	17	4K	3
AM2732B	12.5	N	m4	23	4K	2
AM2764	21.0	N	me	05	8K	3
AM2764P	21.0	N	me	05	8K	3
AM2764A	12.5	N	m1,ti	27	8K	2
AM2764AP	12.5	N	m1,ti	27	8K	2
AM27128	21.0	N	mf	06	16K	3
AM27128A	12.5	N	m2,ti	28	16K	2
AM27256	12.5	N	mz,ti	52	32K	2
AM27512	12.5	N	m7	33	64K	2

EEPROM:

Part #	Volts	Type	Menu	Select	Size	notes

MPU'S:

Part #	Volts	Type	Menu	Select	Size	notes
8741	25.0	N	mr	10	1K	4
8742H	21.0	N	mu	13	2K	4
8748	25.0	N	mr	10	1K	4
8748H	21.0	N	mt	11	1K	4
8749	21.0	N	mu	13	2K	4
8749H	21.0	N	mu	13	2K	4

ATMEL
EPROMS: N=nmos, C=cmos

Part #	Volts	Type	Menu	Select	Size	notes
AT27256	12.5	N	mz,ti	52	32K	2
AT27C256	12.5	C	mz,ti	52	32K	2

CYPRESS
(E)PROMS: N=nmos, C=cmos

Part #	Volts	Type	Menu	Select	Size	notes

Future support
Note that you can do nearly the whole series Cypress Prom parts using addressing techniques to fit the part)

DALLASSEMICONDUCTOR
BATTERY BACKED STATIC RAM:
N=nmos, C=cmos

Part #	Volts	Type	Menu	Select	Size	notes
DS1220	TTL	N	mq	35	2K	1
DS1225	TTL	N	m9	39	8K	1

FUJITSU
EPROMS: N=nmos, C=cmos

Part #	Volts	Type	Menu	Select	Size	notes
MBM2764	21.0	N	me	05	8K	3
MBM27C64	21.0	C	me	05	8K	3
MBM27128	21.0	N	mf	06	16K	3
MBM27C128	21.0	C	mf	06	16K	3
MBM27256	12.5	N	m%	21	32K	7
MBM27C256	21.0	C	m8	37	32K	2
MBM27C256A	12.5	C	m%	21	32K	7
MBM27C512	12.5	C	m7,ti	33	64K	2
MBM27C1001	12.5	C	m=	60	128Kx8	2,16
MBM27C1024	12.5	C	m=	60	64Kx16	2,16

MPU'S:

8742H	21.0	N	mu	13	2K	4

GENERALINSTRUMENT

EPROMS: N=nmos, C=cmos

Part #	Volts	Type	Menu	Select	Size	notes
27C64	12.5	C	m1,ti	27	8K	2
27HC64	12.5	C	m1,ti	27	8K	2
27C128	12.5	C	m2,ti	28	16K	2
27256	12.5	C	mz,ti	52	32K	2
27C256	12.5	C	mz,ti	52	32K	2

HITACHI

EPROMS: N=nmos, C=cmos

Part #	Volts	Type	Menu	Select	Size	notes
HN482716G	25.0	N	mb	53	2K	3
HN482732G	25.0	N	mc	16	4K	3
HN482732AG	21.0	N	md	17	4K	3
HN482764G	21.0	N	me	05	8K	3
HN482764P	21.0	N	me	05	8K	3
HN27C64	21.0	C	me	05	8K	3
HN4827128P	21.0	N	mf	06	16K	3
HN27128A	12.5	N	m2	28	16K	2
HN27256G	12.5	N	mz,ti	52	32K	2
HN27512	12.5	N	m7,ti	33	64K	2
HN27C101	12.5	C	m=,ti	60	128K	2,16

EEPROMS:

HN58064P	TTL	N	m9	39	8K	1

INTEL

EPROMS: N=nmos, C=cmos

Part #	Volts	Type	Menu	Select	Size	notes
2758	25.0	N	ma	19	1K	3
2716	25.0	N	mb	53	2K	3
2732	25.0	N	mc,tq	55	4K	3,17
2732A	21.0	N	md,tq	56	4K	3,17
P2732A	21.0	N	md,tq	56	4K	3,17
2764	21.0	N	me,tq	48	8K	3,17
2764A	12.5	N	m1	50	8K	2,17
P2764A	12.5	N	m1	50	8K	2,17
27C64	12.5	C	m1	50	8K	2,17
87C64	12.5	C	m1	50	8K	2,17
27128	21.0	N	mf,tq	49	16K	3,17
27128A	12.5	N	m2	51	16K	2,17
27256	12.5	N	mz	52	32K	2,17
P27256	12.5	N	mz	52	32K	2,17
27C256	12.5	C	mz	52	32K	2,17
87C256	12.5	C	mz	52	32K	2,17
27512	12.5	N	m7	54	64K	2,17
P27512	12.5	N	m7	54	64K	2,17
27010	12.5	N	m=	61	128Kx8	16,17
27011	12.5	N	m=	61	128Kx8	17
27210	12.5	N	m=	61	64Kx16	16,17

EEPROMS: N=nmos, C=cmos

Part #	Volts	Type	Menu	Select	Size	notes
2816A	TTL	N	my	36	2K	1
2817A	TTL	N	m3	38	2K	1
2864	TTL	N	m9	39	8K	1

Intel MPU'S:

Part #	Volts	Type	Menu	Select	Size	notes
8741	25.0	N	mr	10	1K	4
8742H	21.0	N	mu	13	2K	4
8748	25.0	N	mr	10	1K	4
8748H	21.0	N	mt	11	1K	4
8749H	21.0	N	mu	13	2K	4
8751	21.0	N	mv	32	4K	6
8751H	21.0	N	mv	32	4K	6
87C51	12.5	C	m$		4K	14
8744H	21.0	N	mv	32	4K	6

OTHER:

8755	25.0	N	mw	29	2K	9

MITSUBISHI EPROMS: N=nmos, C=cmos

Part #	Volts	Type	Menu	Select	Size	notes
M5L2716K	25.0	N	mb,ti	53	2K	3
M5L2732K	25.0	N	mc	16	4K	3
M5L2764K	21.0	N	me	05	8K	3
M5L27128	21.0	N	mf	06	16K	3
M5M27C128	21.0	C	mf	06	16K	3
M5L27256	12.5	N	mz,ti	52	32K	2
M5M27C256K	12.5	C	mz,ti	52	32K	2
M5L27512	12.5	N	m7,ti	33	64K	2
M5M27C101K	12.5	C	m=,ti	60	128Kx8	2,16
M5M27C102K	12.5	C	m=,ti	60	64Kx16	2,16

MOTOROLA
EPROMS: N=nmos, C=cmos

Part #	Volts	Type	Menu	Select	Size	notes
MCM2716	25.0	N	mb	02	2K	3
MCM2532	25.0	N	mi	08	4K	1
MCM68732	25.0	N	mc	16	4K	3
MCM68764	25.0	N	mk	09	8K	2
MCM68766	25.0	N	mk	09	8K	2

EEPROM:

MCM2833	TTL	N	m9	39	4K	1
MCM2864	TTL	N	m9	39	8K	1

MPU'S:

MC68705P3	21.0	N	(note)		1K	11
MC68705P5	21.0	N	(note)		1K	11
MC68705R3	21.0	N	(note)		2K	11
MC68705R5	21.0	N	(note)		2K	11
MC68705U3	21.0	N	(note)		2K	11
MC68705U5	21.0	N	(note)		2K	11

NATIONAL
EPROMS: N=nmos, C=cmos

Part #	Volts	Type	Menu	Select	Size	notes
MM2716	25.0	N	mb	53	2K	3
NMC27C16	25.0	C	mb	53	2K	3
NMC27C32	25.0	C	mc	16	4K	3
NMC27C64	12.5	C	m1	50	8K	2
NMC27CP128	12.5	C	m"	44	16K	2
NMC27C256	12.5	C	m.	45	32K	2
NMC27C512	12.5	C	m7	54	64K	2

EEPROM:

NMC98C64A	TTL	N	m9	39	8K	1

NEC

EPROMS: N=nmos, C=cmos

Part #	Volts	Type	Menu	Select	Size	notes
uPD2716D	25.0	N	mb	53	2K	3
uPD2732D	25.0	N	mc	16	4K	3
uPD2732C	25.0	N	mc	16	4K	3,12
uPD2732AD	21.0	N	md	17	4K	3
uPD27C32D	21.0	N	md	17	4K	3
uPD2764D	21.0	N	me	05	8K	3
uPD2764C	21.0	N	me	05	8K	3,12
uPD27C64D	21.0	C	me	05	8K	3
uPD27C64C	21.0	C	me	05	8K	3
uPD27128D	21.0	N	mf	06	16K	3
uPD27128C	21.0	N	mf	06	16K	3,12
uPD27256D	21.0	N	m8	37	32K	2
uPD27256C	21.0	N	m8	37	32K	2,12
uPD27C256D	21.0	C	m8	37	32K	2
uPD27C256C	21.0	C	m8	37	32K	2,12
uPD27C1001D	12.5	C	m=	61	128K	16,17
uPD27C1024	12.5	C	m=	61	64Kx16	16,17

MPU'S:

8741	25.0	N	mr	10	1K	4
8742H	21.0	N	mu	13	2K	4
8748	25.0	N	mr	10	1K	4
8748H	21.0	N	mt	11	1K	4
8749H	21.0	N	mu	13	2K	4

OKI

EPROMS: N=nmos, C=cmos

Part #	Volts	Type	Menu	Select	Size	notes
MSM2764	21.0	N	me	05	8K	3
MSM27128	21.0	N	mf	06	16K	3

ROCKWELL

EPROMS: N=nmos, C=cmos

Part #	Volts	Type	Menu	Select	Size	notes
87C64	12.5	C	m1,ti	27	8K	2

SEEQ
EPROMS: N=nmos, C=cmos

Part #	Volts	Type	Menu	Select	Size	notes
5133	21.0	N	me	05	8K	3
5133H	21.0	N	me	05	8K	3
5143	21.0	N	mf	06	16K	3
27256	12.5	N	mz,ti	52	32K	3
27C256	12.5	N	mz,ti	52	32K	3

EEPROMS:

DQ2816A	TTL	N	my	36	2K	1
DQ2817A	TTL	N	m3	38	2K	1
DQ2864	TTL	N	m9	39	8K	1
DQ28C64	TTL	C	m9	39	8K	1
DQ28C256	TTL	C	m4	41	32K	1
5212	TTL	N	mp	34	1K	1
5213	TTL	N	mp	34	2K	1
52B13	TTL	N	mp	34	2K	1
52B23	TTL	N	m9	39	4K	1
52B33	TTL	N	m9	39	8K	1
52B13H	TTL	N	m9	39	2K	1
52B23H	TTL	N	m9	39	4K	1
52B33H	TTL	N	m9	39	8K	1

SIGNETICS
EPROMS: N=nmos, C=cmos

Part #	Volts	Type	Menu	Select	Size	notes
27C64	12.5	C	m1,ti	27	8K	2
87C64	12.5	C	m1,ti	27	8K	2
27C256	12.5	C	mz,ti	52	32K	2
87C256	12.5	C	mz,ti	52	32K	2

SGS
EPROMS: N=nmos, C=cmos

Part #	Volts	Type	Menu	Select	Size	notes
M2716	25.0	N	mb	53	2K	3
M2716P	25.0	N	mb	53	2K	3
M2732A	21.0	N	md	17	4K	3
M2732AP	21.0	N	md	17	4K	3
M2764	21.0	N	me	05	8K	3
M2764P	21.0	N	me	05	8K	3
M2764A	12.5	N	m1,ti	27	8K	2
M2764AP	12.5	N	m1,ti	27	8K	2
M27128A	12.5	N	m2,ti	28	16K	2
M27256	12.5	N	mz,ti	52	32K	2
M27512	12.5	N	m7,ti	33	64K	2

SMOS
EPROMS: N=nmos, C=cmos

Part #	Volts	Type	Menu	Select	Size	notes
27C64	21.0	C	me	05	8K	3
27128	21.0	N	mf	06	16K	2
27C256	12.5	C	mz,ti	52	32K	2

EEPROM:

2864	TTL	N	m9	39	8K	1

TEXASINSTRUMENTS

EPROMS: N=nmos, C=cmos

Part #	Volts	Type	Menu	Select	Size	notes
TMS2516	25.0	N	mb	53	2K	3
TMS2532	25.0	N	mi	08	4K	1
TMS2532A	21.0	N	m+	40	4K	2
TMS2732	25.0	N	mc	16	4K	13
TMS2732A	21.0	N	m^	42	4K	2
TMS27P32A	21.0	N	m^	17	4K	2
TMS2564	25.0	N	mj	14	8K	13
TMS2764	21.0	N	me	05	8K	13
TMS27P64	21.0	N	me	05	8K	13
TMS27C64	12.5	C	m1,ti	27	8K	2
TMS27C128	12.5	C	m2,ti	28	16K	2
TMX27PC128	12.5	C	m2,ti	28	16K	2
TMS27C256	12.5	C	mz,ti	52	32K	2
TMX27PC256	12.5	C	mz,ti	52	32K	2
TMX27C512	12.5	C	m7,ti	33	64K	2

(E)PROMS: (50nS PROM EQUIVALENT)

TOSHIBA

EPROMS: N=nmos, C=cmos

Part #	Volts	Type	Menu	Select	Size	notes
TMM2764D	21.0	N	me	05	8K	3
TMM2764DI	21.0	N	me	05	8K	3
TMM2764AD	12.5	N	m1,ti	27	8K	2
TMM2464AP	12.5	N	m1,ti	27	8K	2
TMM27128D	21.0	N	mf	06	16K	3
TMM27128DI	21.0	N	mf	06	16K	3
TMM27128AD	12.5	N	m2,ti	28	16K	2
TMM24128AP	12.5	N	m2,ti	28	16K	2
TMM27256D	21.0	N	m8	37	32K	2
TMM27256DI	21.0	N	m8	37	32K	2
TMM27256AD	12.5	N	mz,ti	52	32K	2
TMM24256AP/F	12.5	N	mz,ti	52	32K	2
TC57256D	21.0	N	m8	37	32K	2
TMM27512D	12.5	N	m7,ti	33	64K	2

VLSI
EPROMS: N=nmos, C=cmos

Part #	Volts	Type	Menu	Select	Size	notes
VT27C64	12.5	C	m1,ti	27	8K	2
VT27C128	12.5	C	m2,ti	28	16K	2
VT27256	12.5	C	mz,ti	52	32K	2

XICOR
EEPROMS: N=nmos, C=cmos

Part #	Volts	Type	Menu	Select	Size	notes
X2816A	TTL	N	mq	35	2K	1
X2864A	TTL	N	m9	39	8K	1
X28C64	TTL	C	m9	39	8K	1
X28256	TTL	N	m4	41	32K	1
X28C256	TTL	C	m4	41	32K	1

GTEK believes that the information contained in this list is correct. However, GTEK assumes no responsibility or liability for the accuracy of this list.

Rotary Menu Selection Vs Algorithm/Voltage

#=Item number Part=GENERIC part number
D=Dumb (standard) algorithm I=Intelligent (adaptive) algorithm
Q=Quick (QuickPulse) alg. DF=Default algorithm with RS–232
Tx=Toggle alg. that may be used (TD, TI and TQ) via RS–232
Vpp=Voltage used to program part with the selected algorithm.

#	Part #	Vpp	D	I	Q	DF	Menu Tx
02	2716	25.0	02	53	—	I	B (DI)
03	2732	25.0	03	16	55	I	C (DIQ)
04	2732A	21.0	04	17	56	I	D (DIQ)
05	2764	21.0	15	05	48	I	E (DIQ)
06	27128	21.0	18	06	49	I	F (DIQ)
07	27256	12.5	—	52	07	Q	Z (IQ)
08	2532	25.0	08	—	—	D	I
09	68766	25.0	—	09	—	I	K
10	874x–1K	25.0	10	—	—	D	R
11	874xH–1K	21.0	11	—	—	D	T
12	874x–2K	25.0	12	—	—	D	S
13	874xH–2k	21.0	13	—	—	D	U
14	2564	25.0	14	—	—	D	J
15	2764	21.0	15	05	48	I	E (DIQ)
16	2732	21.0	03	16	55	I	F (DIQ)
17	2732A	21.0	04	17	56	I	D (DIQ)

#	Part #	Vpp	D	I	Q	DF	Menu Tx
18	27128	21.0	06	18	49	I	F (DIQ)
19	2758	25.0	43	19	—	I	A (DI)
20	2716B	12.5	—	20	—	I	G
21	F27256	12.5	—	21	—	I	%
22							
23	2732B	12.5	—	23	57	I	H (IQ)
24							
25							
26							
27	2764A	12.5	—	27	50	Q	1 (IQ)
28	27128A	12.5	—	28	51	Q	2 (IQ)
29	8755	25.0	29	—	—	D	W
30							
31							
32	8751	21.0	32	—	—	—	V
33	27512	12.5	—	33	54	Q	7 (IQ)
34	5213	5.0	34	—	—	D	P
35	X2816A	5.0	35	—	—	D	Q
36	I2816A	5.0	36	—	—	D	Y
37	F27C256	21.1	—	37	—	I	8

#	Part #	Vpp	D	I	Q	DF	Menu Tx
38	I2817A	5.0	38	—	—	D	3
39	X2864A	5.0	—	39	—	I	9
40	TI2532A	21.0	—	40	—	I	+
41	X28256	5.0	—	41	—	I	4
42	TI2732A	21.0	—	42	—	I	^
43	2758	25.0	43	19	—	I	A (IQ)
44	N27C128	12.5	—	—	44	Q	"
45	N27C256	12.5	—	—	45	Q	.
46							
47							
48	2764	21.0	15	05	48	I	E (DIQ)
49	27128	21.0	18	06	49	I	F (DIQ)
50	2764A	12.5	—	27	50	Q	1 (IQ)
51	27128A	12.5	—	28	51	Q	2 (IQ)
52	27256	12.5	—	52	07	Q	Z (IQ)
53	2716	25.0	53	02	—	I	B (DI)
54	27512	12.5	—	33	54	Q	7 (IQ)
55	2732	25.0	03	16	55	I	C (DIQ)
56	2732A	21.0	04	17	56	I	D (DIQ)
57	2732B	12.5	—	23	57	I	H (IQ)

#	Part #	Vpp	D	I	Q	DF	Menu Tx
58	27513	12.5	—	58	59	Q	# (IQ)
59	27513	12.5	—	58	59	Q	# (IQ)
60	27011	12.5	—	60	61	Q	= (IQ)
61	27011	12.5	—	60	61	Q	= (IQ)
62							
63							
64							
65							
66							
67							
68							
69							
70							
71							
72							
73							
74							
75							