

## Appendix D Specifications

### NovaTech NTZ-818 Zilog 18 pin Microcontoller Device Programmer

<b>Modes of Operation</b>	PC Control, Stand-Alone
<b>Computer Interface</b>	RS-232, 9600 baud via DB9 Female
<b>System Requirements</b>	Windows 95/98, free serial port, 3 MB hard drive disk space
<b>Devices Programmed</b>	ZiLOG 18 pin OTP microcontrollers <b>including...</b>  Z86E02/E04/E08 all revisions and option bits Z86E03/E06 all revisions and option bits <b>Excluding:</b> Z8PLUS (Z8E000/001/002/003)
<b>Functions</b>	Select Device/Revision/Option Bits Load Master Blank Check Program Verify
<b>Throughput</b>	300 - 450 devices per hour
<b>Size</b>	7.125" x 3.5" x 10"
<b>Weight</b>	3.75 lbs.
<b>Power Requirements</b>	90-264 VAC 50/60 Hz, 20 watts (NTZ-818)
<b>Environmental</b>	0 - 40° C

## NTZ-818 Users Manual

### Table of Contents

General Theory of Operation.....	2
Typical Programming Session.....	2
Supported Zilog Devices and Option Bits.....	6
Host Computer Control vs. Stand Alone.....	8
Host control - Windows GUI.....	8
Host Control - Dos terminal, or non PC application.....	9
Stand Alone mode .....	9
Precautions.....	11
Troubleshooting .....	12
Warranty.....	13
Technical Support.....	13
Appendix A - Interface Cable.....	14
Appendix B - ASCII Remote Control Commands.....	15
Appendix C - Options & Accessories.....	19
Appendix D - Specifications.....	20

P/N 50034-0001  
REV April, 1999

## General Theory of Operation

The NovaTech NTZ-818 is a low cost microcontroller gang programmer for ZiLOG Z8 18 pin parts except Z8Plus core. The programmer can burn up to eight chips at a time and can function in either stand alone mode or host computer control.

SOIC parts can be programmed using optional SOIC adapter sockets. Because of the width of the adapter sockets, the programmers can hold and program only 4 SOIC adapters at a time.

## Typical Programming Session

A typical programming session consists of the following steps:

1. **Select a Device Type and Mask Revision**
2. **Select option bits**
3. **Load user buffer with device data (either from a MASTER DEVICE or from an Intel .HEX file)**
4. **Insert Blank micro-controllers in ZIF sockets**
5. **Perform a Blank Check on installed devices and remove and segregate any non-blank devices**
6. **Program micro-controllers with desired code**
7. **Remove and label programmed micro-controllers segregate any non-verified devices.**

The steps differ slightly depending upon whether you're performing them from the Windows application program or the programmerskey-pad, but in general, they can be described as follows.

### 1. Select Device Type and Mask Revision

#### Device Type

Zilog 18 pin micro-controllers are available in two basic families the Z86E03/06 and the Z86E02/04/08. Each part offers a different available EPROM size and sometimes a different mix of features. The device type you select must match the device you intend to program.

#### Mask Revision

Over the years, changes and upgrades have been made to the Z8 micro-controller masks, providing bug fixes or added option bit support. Although all parts generally program the same way, some option bits are not supported on some Mask Revisions. The Mask Revision (also called the Top Mark) is printed next to the word Zilog on all DIP

## Appendix C Options & Accessories

<u>ITEM</u>	<u>Part No.</u>
Power Cord	50003-0001
DB9-DB9 Interface Cable	50014-0001
DIP-SOIC Adapter	50015-0001
User Manual	50034-0001

SFhh<CR> Fill user buffer with byte specified.

example: SF55<CR>

SRhhhhnn<CR> Patch user Ram location hhhh with byte nn.

example: SR00AA20<CR>  
Sets location \$00AA to \$20.

## PROGRAMMER FUNCTIONS

ZB<CR> Blank Check installed devices. Update device LEDs. Use 'QS' command to return results.

example: ZB<CR>

ZV<CR> Verify installed devices against user buffer. Update device LEDs. Use 'QS' command to return results.

example: ZV<CR>

ZM<CR> Move Device. Move the first non-blank, non-zero device into the user buffer. Display checksum. Update lights.

example: ZM<CR>

ZP<CR> Program Devices. Blank check all sockets. Program all devices with contents of user buffer and select option bits.

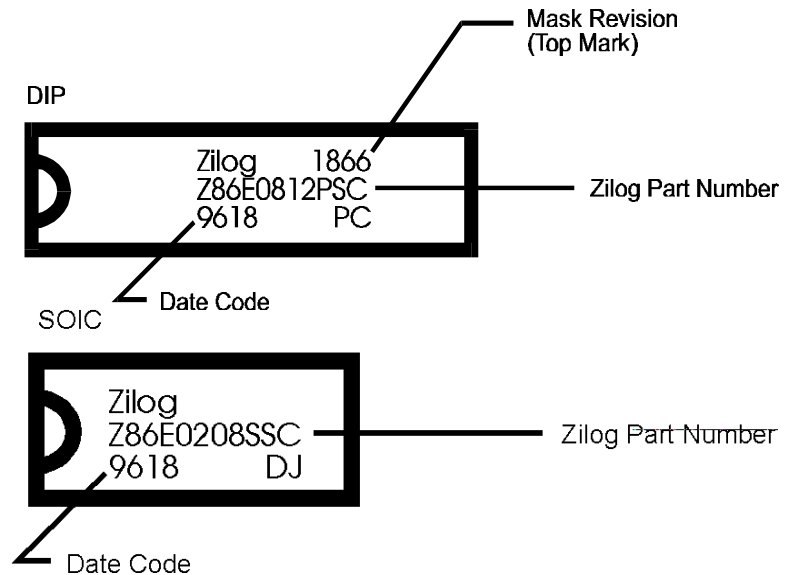
example: ZP<CR>

### A note on Syntax

In general, the syntax is extremely rigid. Commands may be upper- or lowercase, but no errant commas or leading spaces are allowed. Most numeric parameters are ASCII hex with no leading '\$' or '&H' required or allowed. Most numeric responses are ASCII hex with &H supplied.

ICs. Do not confuse the Mask Revision with the date code, which is typically printed on the third line and contains the year and week of production (e.g. 9618 = week 18 of 1996).

FIGURE 1



Use **Figure 2** (on Page 6) to help sort out Devices and Mask revisions with respect to available option bits. You may wish to make a copy of this chart and affix it to your programmer for quick reference.

**Important:** for Z86E04xxSEC/SSC or PEC and Z86E08xxSEC/SSC or PEC use Mask 1866 - even though it's not printed on IC. See page 6 for details

## 2. Select option bits

The option bits are non-executing configuration bits, typically write only, that modify the configuration of the part. They are burned into the chip after the EPROM array has been programmed with user code.

Many applications will not require any option bit programming at all. Not all options are available for all microcontrollers. Refer to **Figure 2** for a detailed index of which devices and revisions support which options.

The option descriptions below are intended only as a guide. Please refer to Zilog's product specification for the particular part you are using for a complete and accurate description of the option bits available for that particular part.

#### Option bit - ROM Protect

This option, when programmed, disables any further read access to the EPROM array. Basically, it is used to protect proprietary code from being copied. If you are using a Master IC to archive your production code, you should not program its ROM Protect bit, as this will disable copying of your Master IC.

*The unprogrammed state of this option is NOT Protected.*

#### Option bit - Low Noise (EMI)

Programming this option disables the internal clock divide-by-two (a major source of EMI) but subsequently doubles the effective execution speed. It also limits the slew rate of all external output transitions by switching in a series resistance to reduce harmonic emissions.

*The unprogrammed state of this option is Low Noise Disabled.*

#### Option bit - Autolatch Disable

The autolatches are intended to reduce target device power dissipation in the event unused inputs are left floating (unconnected). Unfortunately, an unwanted side effect is an additional temperature-sensitive impedance of between 50k to 500k on the input pin. In some applications this can cause problems. In these cases it is best to tie all undriven inputs to Vcc or Gnd and disable the Autolatches.

*The unprogrammed state of this option is Autolatch Enabled.*

#### Option bit - WDT Enable

This option permanently enables the watchdog timer. Use of the watchdog timer does not require you to program this bit - simply enable the Watchdog timer as required by the microcontroller (usually by executing a WDT instruction near the beginning of your code). However, an application that may unexpectedly 'get lost' before properly enabling the WatchDog timer (due to Power up transients or improper watchdog initialization) may benefit from the use of this option bit.

*The unprogrammed state of this option is Watchdog Timer Not Automatically Enabled.*

#### Option bit - EPROM Test Kill

This option is similar to ROM Protect, but is typically used for a different reason. When programmed, it disables any further access to

*example:* QS<CR>  
*response:* ZZZZBBPV<CR><LF>

*devices 1-4 are not installed  
devices 5&6 are blank (unprogrammed or inaccessible)  
device 7 is programmed but does not verify against user buffer.  
device 8 is programmed and verifies*

QT<CR> Query device Type. Programmer reports device type, mask, and options currently set.

*example:* QT<CR>

### SET COMMANDS

'?' Attention. Command used to initiate PCControl. Subsequently used to determine if programmer is busy (response: '\_') or ready to receive additional commands (response: '!')

SK<CR> Set Keypad Local. Used to terminate a PC Control session and return programmed to Keypad or Stand-alone control. (Command not supported on NTZ-418). '?' must be sent to return programmer to PC Control.

SOoo...o<CR> Set Options. Use to setup option bit programming. Send '0' for unprogrammed and '1' for programmed. Option bits are in order that they are reported by 'QT' command. Not all options are available for all parts. Invalid options will be silently ignored.

*example:* SO1111000000000000<CR>

*Set first four option bits. Leave rest unprogrammed.*

STddd,ddd<CR> Set device Type, Mask.

*example:* ST8608,1903<CR>

*example:* ST8604,0<CR>

## QUERY COMMANDS

QC*n*<CR> Query Checksum. Parameter *n* is the socket # 1-8 (or 0 for user buffer). Response is socket # followed by Checksum.

example: QC3<CR>  
response: 3 &HFC00<CR><LF>

*Programmer calculates and outputs Checksum of socket #3.*

QL<CR> Query Lights. Response is the status of the eight trinary LEDs.

example: QL<CR>  
response: rrrroooo<CR><LF>

*LEDs 1-4 are Red, 5-8 are off.*

QM<CR> Query Machine. Programmer responds with information about its hardware and firmware.

example: QM<CR>

QR*hhhh*<CR> Query RAM. Hex location *hhhh*. Programmer responds with HEX ASCII &H*hh*. Ram location queried is relative to start of programmer's user buffer.

example: QR03A0<CR>  
response: &HFF<CR><LF>

QS<CR> Query Sockets. Programmer reports information about what's in the sockets, based on last programmer function performed, using the following characters:

**Z** device is not installed or all Zeroes  
**P** device is programmed  
**V** device is verified against user buffer  
**B** device is blank (or access to EPROM Array is denied via option bit.)

ther the EPROM mode or Test mode. These modes are typically entered by applying high (greater than 8.5 volts) voltages on various device pins. In some applications, these voltages may be generated accidentally, causing the microcontroller to exit User mode (the code it is running in the target application) and enter either EPROM mode or Test Mode - modes it should only be in when in a device programmer or wafer test. By programming this option bit, you can prevent any access to EPROM or Test modes, regardless of any transient voltages on the Z8's pins. Refer to Zilog Data sheets for precautions.

*The unprogrammed state of this option is EPROM and Test modes both enabled and NOT killed.*

### Option bit - RC Oscillator

This option is available as an option bit for some of the Z8s. For others, it must be ordered from the factory as a different Mask Revision. For example, the 8602 can be ordered under mask revision *none* for crystal oscillator or revision 1903 for RC oscillator. When programmed, this option allows you to use a less expensive, though less predictable, RC oscillator, instead of a crystal or ceramic resonator, to generate the system clock.

*The unprogrammed state of this option is RC oscillator Disabled - must use a Crystal oscillator or Ceramic Resonator.*

### A final word about option bits

Option bits are typically 'write only'. This has the following implications:

You cannot duplicate the option bits from a Master IC, because you cannot read them.

You cannot verify directly that a particular option bit has been programmed.

Some of the option bits (ROM Protect and EPROM Test Kill) limit subsequent access to the EPROM array. Therefore, you cannot always use a microcontroller as a Master IC.

### - Warning -

*A microcontroller with either of these access limiting option bits may appear blank or zero to a programmer. Any attempt to re-program an already programmed device may render the target IC useless.*

NovaTech Z8 Gang Burner device support cross reference table											
Zilog Part	Z86E02	Z86E02	Z86E02	Z86E04	Z86E04	Z86E04	Z86E04	Z86E08	Z86E08	Z86E08	Z86E06
SL Rev	none	1903	1925	none	1866	1866	1903	none	1866	1903	none
In Production	yes	yes	yes	no	yes	yes	yes	no	yes	yes	yes
Rom Size	512	512	512	1024	1024	1024	1024	2048	2048	2048	1024
Gang burner Alias	8602	8602	8602	8604	8604	8604	8604	8608	8608	8608	8606
ROM Protect	Yes	Yes	Yes	Yes*	Yes	Yes	Yes	Yes*	Yes	Yes	Yes
Low Noise (EMI)	"RP Y"	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Auto Latch Disable	"LE Y"	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	FW
Permanent WDT	"AL d"	Yes	Yes	N/A	Yes	Yes	Yes	N/A	Yes	Yes	Yes
EPROM Test KILL	"ud Y"	Yes	Yes	N/A	Yes	Yes	Yes	N/A	Yes	Yes	N/A
RC Oscillator	"EP n"	Yes	Yes	N/A	Yes	Yes	Yes	N/A	Yes	Yes	N/A
	"RC Y"	Xtal	Yes	Xtal	Xtal	Xtal	RC	Xtal	Xtal	RC	Yes
	Alias if not prog.	Alias if not prog.	Notes:								
			1. If using a Plastic DIP extended temp. (PEC) or either SOIC package (SSC/SEC) the SL Mask may not be printed on the package.								
			2. Use none (0) for Z86E02 and 1866 for Z86E04 or Z86E08.								
			3. * with these older parts, programming ROM protect automatically got you Low EMI.								
			4. "N/A" means Not Available or Not Applicable								
			5. "FW" means this option is available via FW (not as an option bit)								
			6. Gang Burner Alias is the abbreviated name that will appear in Gang Burner's display								
			7. Option Aliases are similar abbreviations for display								
			8. Aliases may be a little cryptic because of seven-segment display.								

FIGURE 2

## Appendix B ASCII Remote Control Commands

### DISPLAY COMMANDS

DTdata<CR> Display Trinary. Light the 8 LEDs above the ZIF sockets. *Data* is lowercase 'r', 'g', or 'o' for RED, GREEN, and OFF respectively.

example: *DTrrrooggg<CR> will turn LEDs 1-3 red, 4 & 5 off, and 6-8 green.*

DP<CR> Display Piezo. No parameters. Beep the beeper.

example: *DP<CR>*

### FILE COMMANDS

All of the file commands require that a device type be selected first. Please refer to the ST command.

FDn<CR> File Download. Prepares programmer to receive a formatted object file containing user code that will ultimately be burned into the target devices. Parameter *n* = 1-9,1 is for Intel Hex type file (currently only supported file type).

example: *FD1<CR> then*

```
:10000000544553542D4F462D414E2D5A38453038C6
:10001000544849532D49532D504147452D2D2D31DD
:10002000303132333435363738394142434445462E
```

```
.
.
.
etc.
:00000003FD
:00000001FF
```

FUf,n<CR> File Upload. Parameter *f* is the socket # 1-8. (or 0 for user buffer). Parameter *n* is file type 1-9, currently only type 1, intel hex is supported.

example: *FU8,1<CR>*

*Programmer Uploads entire contents of Socket #8 in Intel hex format.*

## Appendix A Interface Cable

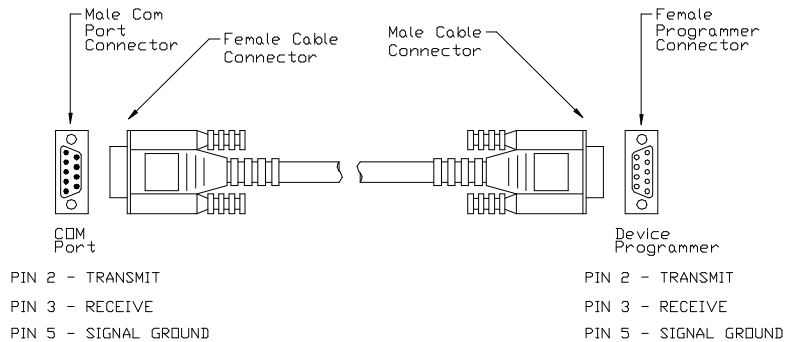
Attach the supplied DB-9 to DB-9 RS-232 interface cable to a spare COM port on your PC. The cable is straight through. You may need a '25 to 9' pin adapter, available from your local computer dealer.

All of the handshake lines are left floating. If this is a problem for your communications software, you may need to make your own cable.

Make sure your COM port is configured for:

9600 baud  
8 data bits  
No Parity  
1 Stop bit

If you are running the Windows GUI, Z8GANG.EXE, you don't have to worry about communication parameters. They are set up automatically by the program.



### 3. Load user buffer with device data

Next it is necessary to load the programmers internal user buffer with the appropriate user program or code that is destined for the blank microcontrollers. There can be two sources for this code:

A Master IC (a known working microcontroller containing user code without any EPROM restrictive option bits programmed).

An intel .hex file containing the user code. The address range of the user code must not exceed the program size available for that chip.

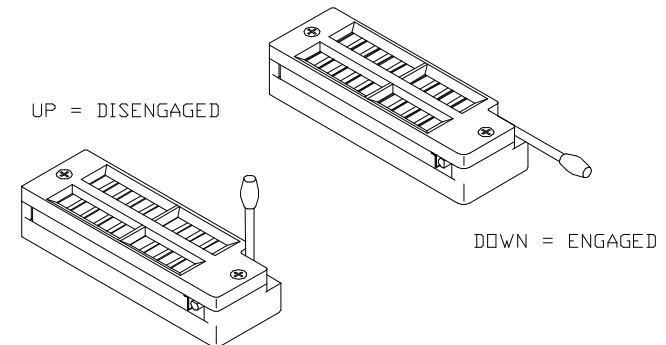
EPROM Size	Valid User Code Addresses
512 bytes	\$0000 - \$01FF
1024 bytes	\$0000 - \$03FF
2048 bytes	\$0000 - \$07FF

*etc.*

### 4. Insert Blank micro-controllers in ZIF sockets

Raise lever to open ZIFs. Install known blank micro-controllers of the appropriate type and Mask Revision. Lower ZIF lever to lock IC in place.

**FIGURE 3**



### 5. Perform a Blank Check on installed devices and remove and segregate any non-blank devices

After performing a blank check, all sockets with blank micro-controllers will light their corresponding green LEDs. If the LED is OFF or RED, the device was not detected or not blank. Remove these ICs and continue with remaining parts.

## 6. Program microcontrollers

Before Programming, all installed devices are blank checked again. If any non-blank devices are detected, programming will not start. Remove errant devices and try again.

During programming, which may take from ten seconds to one minute, the device LEDs will flicker, and the start switch LED will flicker RED. When programming is complete, the start LED will turn GREEN and all the devices will be verified. A GREEN light above the device indicates verify was successful. A RED light indicates the device did not verify. An LED that is OFF indicates that no device is detected in that particular socket. All devices that have properly verified are then programmed with the desired option bits. If either EPROM Test Kill or ROM Protect option was selected, subsequent verification of EPROM array is impossible.

## 7. Remove and label programmed microcontrollers and segregate any non-verified devices

Using the device LEDs as a guide, remove and label successfully programmed and verified devices (as indicated by GREEN LED) and segregate any device with problems (LED is RED or OFF). Note: any device which does not successfully verify after programming will not have its option bits programmed. This can help with troubleshooting, especially if option bits would have limited subsequent access to the EPROM array.

## Host Computer Control vs. Stand Alone

The included Windows 95/98 application program **Z8GANG.EXE** provides a convenient interface between you and the device programmer. File I/O, buffer editing, and a variety of programmer functions are provided by the application.

In a production environment, it is often inconvenient to dedicate a Windows PC just to run a piece of production equipment. This is why there is a stand alone mode on the NTZ-818. Using a 16 key keypad and a 4 digit LED display, you can perform all of the functions required to program microcontrollers.

## Host control - Windows GUI

The included program, **Z8GANG.EXE**, is a 16 bit Windows application that runs under either Windows 95 or 98. To install **Z8GANG.EXE** onto your hard drive, follow the directions printed on Disk One. Once the Z8GANG is installed, follow the on screen prompts, and step

## Warranty

The parts used to manufacture the NovaTech NTZ-818 programmers are under warranty for one year from date of purchase. To establish proof of purchase, we ask that you fill in and mail back the warranty card.

As new Zilog devices become available, Firmware and Control Program updates will be made available free of charge on our Web site at

**[www.novatech-industries.com](http://www.novatech-industries.com)**

Customers requiring actual upgrade diskettes will be accommodated, but a nominal shipping and handling charge will apply.

## Technical Support

NovaTech Industries provides telephone support for all of its products. Call **215-799-0123** for technical support between the hours of **9AM-5PM (EST) Monday-Friday**. To send a fax call **215-799-0125**. E-mail us at **[mail@novatech-industries.com](mailto:mail@novatech-industries.com)**

## Troubleshooting

Windows Application Z8GANG.EXE

1. Message: Could not communicate with gang burner. Entering local mode.

When this message appears, the program was unable to establish communications with the device programmer. This usually means the programmer is simply not turned on. Please refer to Communications troubleshooting below. You can use this mode as an opportunity to familiarize yourself with the application program without hooking up a programmer.

The NovaTech device programmers communicate over an RS-232 link with the following parameters:

- **9600 baud**
- **8 data bits**
- **no parity**
- **1 stop bits**

The Windows application program, Z8GANG.EXE is designed to operate with these settings. In addition, every time the application is executed, it automatically searches available COM ports for the device programmer, so no COM Port specification is required. If you are writing your own communications software, or using a terminal program to communicate directly with the device programmer, you will need to specify the communication parameters and COM Port. In either case, you should also check the following items:

Device Programmer is plugged in and turned on.  
Green Start Switch LED is blinking slowly. If NTZ-818,  
The characters '-818' will appear in the display.  
A serial cable is plugged in to the programmer, and it goes into a spare COM port on your PC.

*Non-Windows only:*

You have selected proper communications parameters and are plugged in to the proper COM port.

Finally, your programmer comes supplied with a 9-pin to 9-pin interface cable. Some older PCs have a 25-pin male connector on COM1. If you need to connect to this style connector, visit your local computer store and ask for a '25 to 9' adapter. This may be a small molded piece or a short cable with a 25-pin Female on one side and a 9-pin male on the other.

through the seven step sequence detailed above to produce your programmed Z8s.

## Host Control - DOS Terminal / non PC

In case you need to control your device programmers from a non-Windows environment, you may communicate directly with it via its built in serial port. **Appendix A** contains syntax for communicating with the device programmer over an asynchronous RS-232 line. Virtually every function accessible from the Windows GUI or Stand alone mode is accessible directly via the described ASCII command language.

## Stand Alone mode

Producing programmed parts in stand alone mode is almost identical to using the Windows control Program. The only exception is that the user code must be loaded from a Master Z8 that has been previously programmed. As detailed earlier in this manual, the Master Device should not have its ROM Protect or EPROM Test Kill option bit programmed.

*Note: When the Windows control program is running, a NTZ-818 will display -PC- in its LED display and its keypad will be inactive. Exit the control program to return to Stand Alone mode.*

## I/O in Stand Alone Mode

The keypad is used to enter Data (Device Type, Mask Revision, Option Bits) and execute Functions. The Start switch is used as a general purpose "Enter" or "Start" Key. The blinking green LED built in to the Start switch prompts you to hit the Start switch.

The LED display provides useful feedback during keyboard data entry. Note: The 4-digit LED has no problem with all numbers and most letters. However, because there are only 7 segments, interpreting some alphabetic characters is a little tricky.

The piezoelectric beeper is used to provide audible feedback of keyboard entry and programming status.

- Click** Short audible feedback of a key press
- Beep** Medium, pleasant audible feedback of a valid keyboard entry or successful termination of a function.
- Bop** Long, low, obnoxious audible feedback of an invalid keyboard entry or improper function termination.

The trinary Red/Green/Off LEDs provide useful information about the

status of the chips in the sockets immediately following a function.

#### Following a Blank Check:

Red Device not Blank  
Green Device Blank  
Off Device not detected

#### Following a Burn or Verify:

Red Device does not match User Buffer  
Green Device does match User Buffer  
Off Device not detected

#### Keyboard Functions in Stand Alone Mode

The keyboard functions are similar to those available with the Windows control program:

#### Select Device Type

Valid entries are 8602, 8604, 8608, 8603, 8606

#### Select Mask Revision

Valid entries are 0, 1866, 1903

**Important: for Z86E04xxSEC/SSC or PEC and Z86E08xxSEC/SSC or PEC use Mask 1866 - even though it's not printed on IC. See page 6 for details**

#### Select Option bits

Use the START switch to enter option bit programming immediately after setting the revision. If you wish to skip option bit programming entirely, press the SELECT key until the Device Type is re-displayed.

Once you select Option bit programming, use the START key to step through the options and the SELECT switch to toggle an option. Remember, the display is a little cryptic as it is difficult to display alphabetic characters on a seven segment display. Refer to the Option Bit Table frequently as you step through the options. The default for all valid option bits is unprogrammed.

#### Load Master

Place a Master Z8 into a ZIF socket. Remove ICs from any other sockets. Hit <Load> key then <START> switch. A green light over the Master indicates success. Verify the checksum displayed is as expected before continuing.

#### Blank Check

Remove Master Z8 and populate the ZIFs with blank Z8s. Ensure Device Type and Mask Revision printed on the ICs match what was pre-

viously set up. Hit <Blank> key then <START> switch. A green light over each IC indicates it is blank. Remove any non-blank devices before continuing.

#### Program

Hit <Program> key then <START> switch. The LEDs will flicker for a while. When all apparent activity ceases, each properly programmed and verified Z8 will light its corresponding green LED. Remove the programmed parts.

#### Precautions

##### Pin 1 Precaution

Observe pin 1 polarity on all ICs. Refer to diagram on the device programmer. Erratic operation may occur if one or more devices are inserted improperly.

##### ZIF Sockets

Lever up is disengaged (can deposit or withdraw IC) lever down is engaged (use to lock IC in place). Refer to Figure 3.

##### Power up and Power down

Do not leave ICs in the device programmer during power up or down. The programmer at all time attempts orderly sequencing of voltage to the sockets. Between functions, the sockets are not powered. Nevertheless, it is prudent to remove any microcontrollers when programmer is not in use.

##### ESD Precaution

Observe proper ESD handling precautions when handling ICs. Zilog recommends use of a grounding strap during device handling.

##### Option Bits

Some of the option bits on the older parts may not work as expected. For example, programming ROM Protect on the older Z86E04/E08s (no Mask revision) automatically results in Low Noise (EMI). Consult your local Zilog sales office if you have specific questions about the parts you are using.

Use ROM Protect and EPROM test kill on production parts only. Otherwise, you will make it extremely difficult to troubleshoot a code problem. Additionally, these parts may appear blank, but attempts at re-programming could destroy the parts.