WEIGH-TRONIX


# SimPoser™ Software
# User's Manual

04/13/00 SIMPOSR2.P65 PN 29587-0018e1 Printed in USA

# Table of Contents

# Introduction

## About This Manual

This manual covers the information you need to install and operate the *SimPoser*™ software package.

Major sections of this manual are headed by titles in a black bar like *Introduction* above. Subheadings appear in the left column. Instructions and text appear on the right side of the page. You will occasionally see notes, tips, and special instructions in the left column. This information will usually pertain to text in the opposite column.

## SimPoser™ and the WI-130 WDAC

*SimPoser*™ software works exclusively with the WI-130 WDAC (**W**eight **D**ata **A**cquisition **C**ontroller). The word SimPoser™ comes from two root words—Simulator and Composer. These two words describe the strengths of this program. Built into the software is a computer simulation of the WI-130. SimPoser lets you compose application programs and configuration setups for the WI-130. You then test an application on the simulator. This makes quick fixes easy and assures identical function when you download the program to a WI-130.

Application programming is done in the WT-BASIC computer language. These application programs and the configuration are saved in a computer file and can be recalled simply by opening that file. Because of this, you can design, buy or trade application programs.

Downloading a program to the WI-130 is as easy as clicking a button on the computer screen. The information you send to the WI-130 is instantly active and the WI-130 is ready to perform the task you have set it up to do. The program becomes part of the WI-130's permanent memory and cannot be lost due to power failure. If you download a new program to the WI-130, the old program is replaced with the new program. No chips to change and no long turnaround times. Each program can take the place of expensive software/hardware specials. Turnaround times can be measured in hours or days instead of weeks or months.

Examples of basic application programs are available which can do batching, checkweighing, inbound/outbound weighing, and other operations. These programs may require additional changes to meet your exact application requirements.

# Simposer™ Software

## Minimum and Recommended Computer Requirements

SimPoser is PC based and requires a certain level of computer power to function. With the minimum setup listed below, the system will work but slowly. The recommended configuration will run the program very well.

### Minimum Configuration

- IBM® compatible AT®/PC with an Intel 486 DX2-50MHZ microprocessor
- 8 megabytes of RAM
- 120 megabyte hard disk drive (program takes up a minimum of 5 meg.)
- 3.5" 1.44 megabyte floppy disk drive
- VGA color monitor
- Mouse
- Dos 5.0 or later
- Microsoft Windows® 3.1 running in enhanced mode

### Recommended Configuration

- IBM® compatible AT®/PC notebook computer with an Intel Pentium 90 microprocessor
- 16 megabytes of RAM
- 120 megabyte hard disk drive (program takes up a minimum of 5 meg.)
- 3.5" 1.44 megabyte floppy disk drive
- VGA color monitor
- 14.4K baud fax/modem
- Mouse
- Dos 5.0 or later
- Microsoft Windows® 3.1 running in enhanced mode, Win95, Win98 or WinNT
- QMODEM Communications program.

This section of the manual is divided into the following sections:

- SimPoser Installation
- Starting SimPoser
- Commands Overview
- SimPoser Operation

## SimPoser Installation

Place the SimPoser diskette in the disk drive and in Windows 3.1 click on File - Run - a:\setup, or in Windows 95 click on Start - Run - a:\setup. Follow all the on-screen prompts as the software is installed.

## Starting SimPoser

1. Once SimPoser is installed, double click the icon with the left mouse button. A progress bar appears and when it reaches 100% the SimPoser program is done loading. A license message appears, and upon accepting this toolbar is displayed:

*You can move the toolbar anywhere on your screen by clicking and dragging the title bar.*

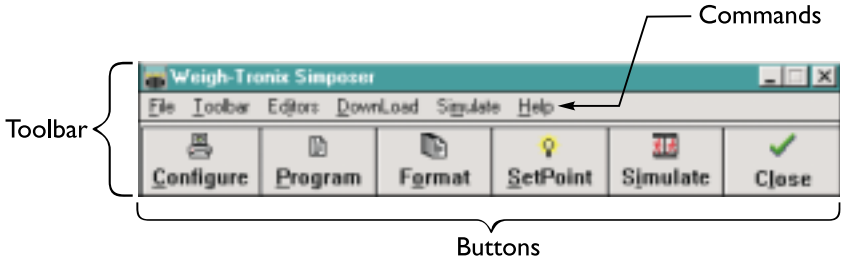



**Figure 1**
SimPoser's toolbar

## Commands Overview

The toolbar consists of six commands and six buttons. Some of these commands are common to all Windows programs and cause a drop down menu to appear. Some commands access the special features of the SimPoser program. A quick overview for each command is given below. Specific instructions appear in the *Operations* section of this manual. For help with Windows operations see your Windows documentation.

**File** — This drop down menu lets you create, open, save and print files and exit the program. This menu also contains a list of the last nine files you have opened. This allows you to click on a file name and open it quickly.

**Toolbar** — This allows you to select a small version of the toolbar.

**Editors** — This is a navigation aid for accessing the different editing windows.

**Download** — Click on this command, then **COM 1** or **COM 2** to download the active application program to the WI-130.

**Simulate** — Click on this command, then *Start* to bring up the WI-130 simulator on the computer screen. All the parameters and instructions you have created and saved will be active and running on the display. This allows you to test your programs before downloading to the WI-130.

**Help** — Click on this command or press F1 to find indexed help documentation.

## Toolbar Buttons Overview

The toolbar has six buttons. You select the function you want by clicking on the appropriate button with the mouse cursor. Below are brief descriptions of each button's function. Complete instructions are in the next section, *SimPoser Operation*.

**Configure button**

When you click on this button with your mouse, a tabbed, dialog box appears containing the customizable features you can set to suit your needs. Below is a list of items in this dialog box:

- Parameters
- Units
- Key Enable
- Display Values
- Display Modes
- Analog Output
- Bargraph
- Counting
- Misc
- Time Out
- Motion/AZT
- Filters
- ROC
- Serial Ports

**Program button**

The Program button opens a WT-BASIC text editing window. Use this window to create application programs using the WT-BASIC computer language. In these programs you can configure the system to run a batching program, setup a checkweighing function, design special graphics for use on the display, and much more.

**Format button**

Click on this button to see a screen for setting up print formats. Design your custom format, choose the format # (from 1 to 16) and save it by clicking on the Save button. Any or all of these formats can be recalled in the program you create in the Program window. For example, you might use this feature for IN and OUT tickets for truck loading/unloading situations, spreadsheet reports for managers, or ISO documentation.

**Setpoint**

Click on this button to bring up a dialog box for configuring setpoints.

**Simulate**

Click on this button to bring up the WI-130 simulator on the computer. This does the same thing as the Simulate command described earlier.

**Close**

Click on this button to exit the SimPoser program.

**The WI-130 can be sealed for legal for trade use and the software protected from change by a hardware connection on the main board. If P19 is jumpered, the system is sealed and programs cannot be downloaded or altered. If P19 is not jumpered the system is not sealed and programs can be downloaded from the SimPoser software.**

# SimPoser Operation

## Toolbar Commands

Following are specific instructions for each of the commands on the SimPoser toolbar.

**File**

*This manual assumes that you know the basic Windows™ procedures. If not, see your Windows™ documentation.*

1. Click on **File**. . .

The drop down menu shown in Figure 2 appears. With this menu you can call for a new file, open an existing file, save a file you are working on, save a file under a new name, print, or exit the SimPoser program. There is also a file history list which holds up to nine of the most recently opened file names

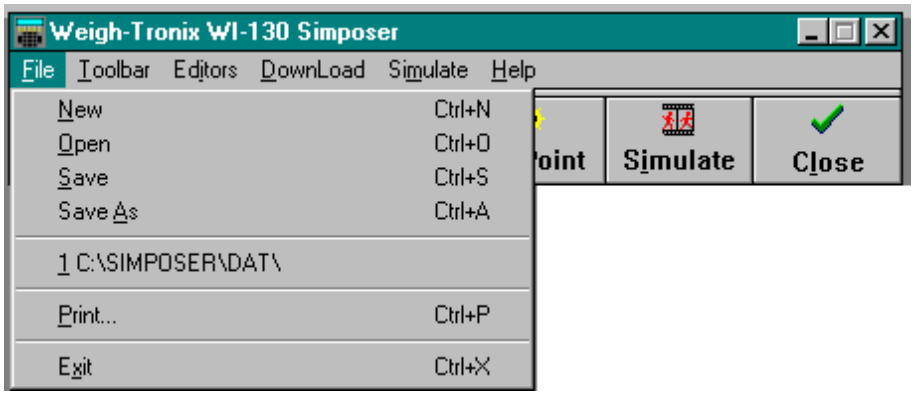
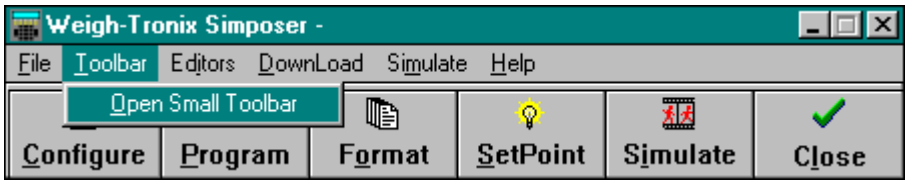



**Figure 2**
File menu

**Toolbar**

2. Click on the **Toolbar** command. . .

The following menu is shown.




Click on **Open Small Toolbar** to replace the large toolbar with the smaller one shown below. . .


Notice that the small toolbar does not have the command line. You can return to the large toolbar at any time by clicking on the button on the right side of the toolbar. You cannot close the SimPoser program from this toolbar. You must first return to the larger version. The other buttons on the small toolbar do the same things as their larger counterparts.

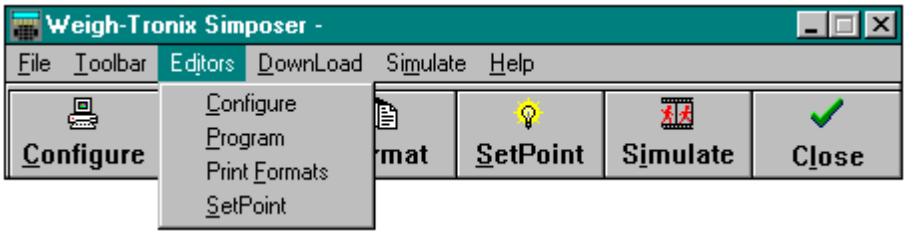
*If the SimPoser software freezes up or crashes while you are working on an application program, you may be able to recover it even if you have not saved it using the SAVE command. If you have perfromed a download or simulation, the SimPoser program creates a temporary copy of the application in C:\SIMPOSER\SIM\TEMPCFG.CFG. To recover the program, open this file in SimPoser and do a FILE,SAVE AS,{filename} and use the file name you want for your application program.*

***Helpful Hints:***
*1. SAVE often!*
*2. Do not have any other Windows® programs running.*
*3. SAVE often!*

**Editors**

3. Click on the **Editors** command. . .

The following is displayed:

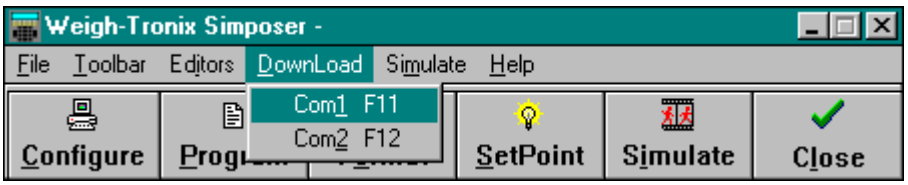



Click on the editing window you want. This is a duplication of the buttons and is handy for navigation of open windows.

**Download**

4. Click on the **Download** command. . .

The following menu is displayed:




*COM1 or COM 2 under Download refer to the serial output from your computer, NOT the serial ports on your WI-130*

*HINT: If you have an application with continuous output, you should disable it before you download.*

When you have created a custom program, use this command to choose which Com port to use to download the program to your WI-130. F11 and F12 keys can be used as hot keys for these functions.

**Simulate**

Click on the **Simulate** command. . .

A simulation of the WI-130 appears on screen. It will behave the same way as a real WI-130 loaded with the program you have active in SimPoser. Use this to test your program before downloading to the real WI-130.

The following computer key strokes take the place of pressing front panel keys when using the simulation:

*If you are going to print from the simulator mode you need to add this line to your autoexec.bat file on your computer and reboot before trying to print:*
*SET WTPORT 2=1*
*What this means is that WTPORT2 is the simulated TT-830 serial port number 2 and =1 is the communication port from your computer.*
***Never set both WTPORTS to the same computer COM port number.***

| COMPUTER KEY STROKE | = | FRONT PANEL KEY |
|---|---|---|
| ALT + S | | SELECT |
| ALT + U | | UNITS |
| ALT + P | | PRINT |
| ALT + T | | TARE |
| ALT + Z | | ZERO |
| ALT + X | | EXIT |
| ALT + C | | CLEAR |
| . | | DECIMAL |
| ESC | | ESCAPE |
| ENTER | | ENTER |
| F1 | | SOFT KEY #1 |
| F2 | | SOFT KEY #2 |
| F3 | | SOFT KEY #3 |
| F4 | | SOFT KEY #4 |
| F5 | | SOFT KEY #5 |

F6-F10 are accessible through a remote keyboard or the simulator.

Move the mouse to change weight on the simulator.

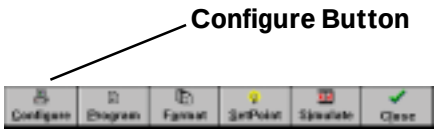
- Move the mouse to the right to increase weight value
- Move the mouse to the left to decrease weight value
- To add more weight with shorter mouse movements, click and hold down the left mouse button while moving the mouse to the right
- To add small increments of weight, hold down the right mouse button, while moving the mouse to the right
- To exit the simulation, press both mouse buttons at the same time or press **ALT** + **X**

**Help**

Click on the Help command to bring up an indexed help manual on your computer screen.

This is the last item on the command line of the SimPoser toolbar. The next section describes the toolbar buttons.

## Toolbar Buttons




Click on the first toolbar button, **Configure**. The dialog box in Figure 3 appears.

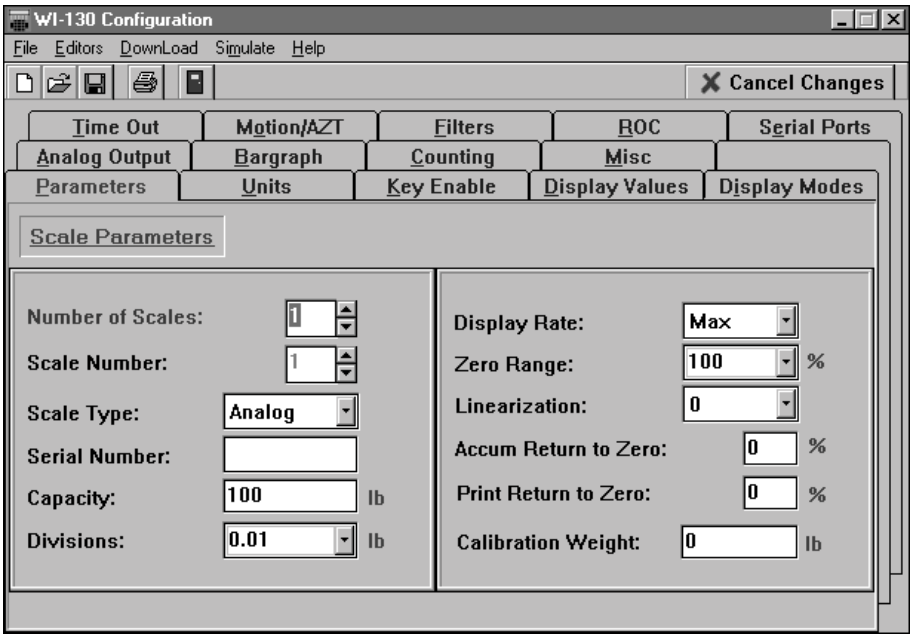



**Figure 3**
Configure dialog box

*TIP*
*Combo Box = A Windows feature which allows you to type in a value or select one from a drop down list.*

*Text Box = A box into which you type a value or word.*

This dialog box contains many tabs representing different areas of scale function. Click on a tab to bring that function into view. The first tab is **Parameters**.

*Parameters tab*

Following is a brief description of each of the scale parameter items you see in Figure 3.

*Number of Scales* Select the number of scales connected to your WI-130.

*Scale Number* Select the scale platform you want to configure.

*Scale Type* Select analog or Quartzell® weight sensor.

*Serial No.* A text box for you to enter in a serial number. This applies only to Quartzell® weight sensors.

*Capacity* Set the capacity for the chosen scale.

*Divisions* Set the division size for the chosen scale platform. Values must be a multiple or submultiple of 0, 1, 2, or 5. If you type an incorrect value, the program will automatically select the closest correct value.

*A display rate of 0.1 means the display is updated only once every 10 seconds. Choosing MAX updates the display faster than the eye can follow.*

*Display Rate* Set the display update rate (the number of times/second the display is updated.) Choose values between .1 (slowest, once every ten seconds) and MAX (fastest) updates per second.

*Zero Range %* Select the percentage of scale capacity you can zero.

*Linearization* Choose a number from -10 to +10 to pull the center point of span back to a linear value.

*By default, when the* **PRINT** *key is pressed, a print operation and an accumulation take place. If you do not want the accumulation to occur, a WT-BASIC program assigning only the DOPRINT command to the* **PRINT** *key needs to be downloaded to the WI-130. A WT-BASIC program can also define an ACCUM. soft key and assign accumulation to that key only.*

*Accum Return to Zero %* — To accumulate weight, the weight must be above this percentage of scale capacity and stable. Before you can perform another accumulation operation the weight must return to zero.

*Print Return to Zero %* — To print, the weight must be above this percentage of scale capacity and stable. Before you can perform another print operation the weight must return to zero.

*Calibration Weight* — The amount of test weight used to calibrate the scale. We recommend entering the test weight most commonly used to calibrate this scale capacity by your organization. (Minimum of 25% of capacity.)

*Units tab*

The next tab is **Units**. This is shown in Figure 4.

*Setpoint and configuration parameters such as capacity, bargraph, checkweigher, and analog output values are based on the calibration unit, not on displayed unit of measure.*

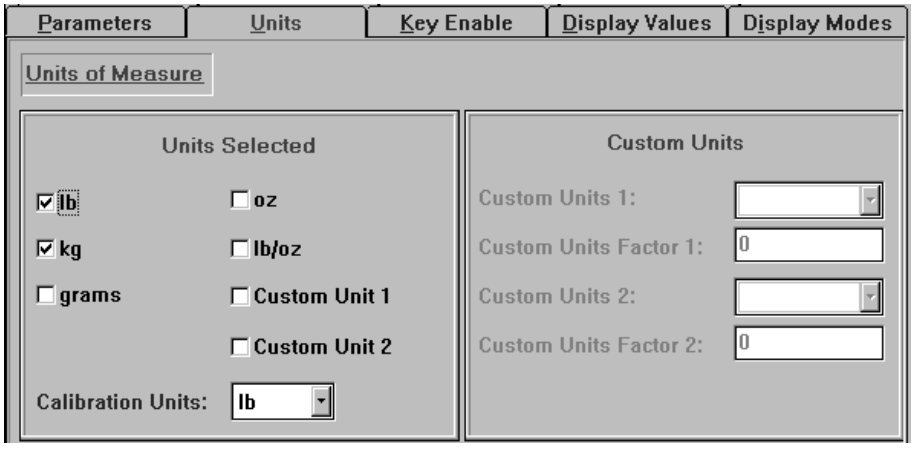



**Figure 4**
Units dialog box

Following is a brief description of each of the unit of measure items you see in Figure 4.

*lb, kg, grams, oz, lb/oz, Custom Unit 1 & 2* — Select the units of measure you want to use. Those you enable will be available to you as the **UNITS** key is pressed on the WI-130. Conversion factors are preassigned by the factory for lb, kg, grams, oz, and lb/oz.

*The custom units conversion factor is the number to be multipied by the weight (in calibration units) to get the desired custom unit. Example: 1 lb = 5 inches of a certain steel rod. Custom unit is inches. Calibration unit is lb. Conversion factor is 5. With six lbs of weight on the scale, 30 inches would be displayed. (Six lbs x 5 = 30 inches of steel)*

*Calibration Unit* Select the unit of measure used in the calibration of your scale. Choose from lb, kg, grams, or oz.

*Custom Units* If you select a custom unit, you can choose from a list of possible units or create your own name for a unit of measure. You must also enter in a conversion factor for that custom unit of measure based on your calibration weight unit.

*Key Enable tab*

The next tab is **Key Enable**, shown in Figure 5. This dialog box lets you enable or disable the keys listed. You can also enable or disable the autotare function or the keyboard tare.

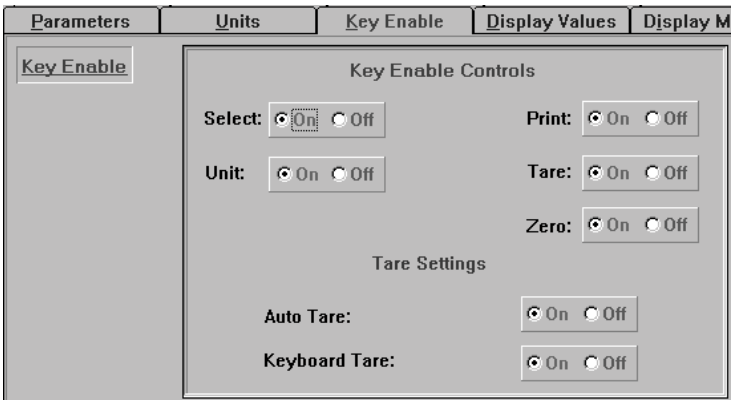



**Figure 5**
Key Enable dialog box

*Display Values tab*

**Display Values** is the next tab, shown in Figure 6.

*ROC stands for Rate Of Change.*

*Min Wt = Minimum captured weight*
*Max Wt = Maximum or peak captured weight.*
*Variable = a value defined by WT-BASIC programming using the keyword or command (SHOWVAR)*

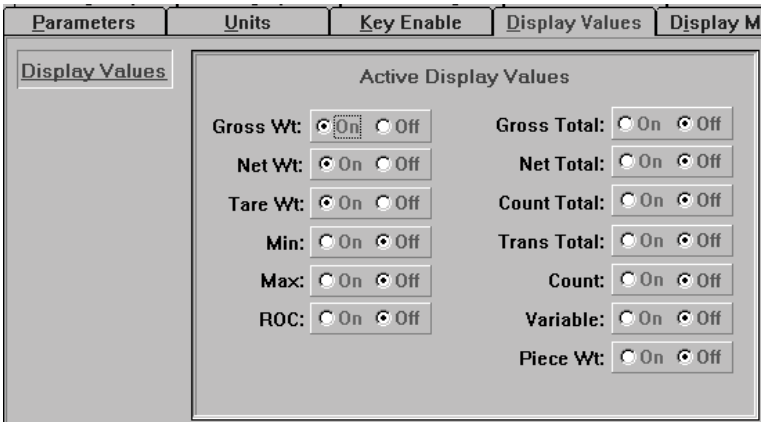



**Figure 6**
Display Values dialog box

From this dialog box, enable the types of display values you wish to be active and available during normal weighing operations. The display values you choose will show up on your screen as you repeatedly push the **SELECT** key on the front panel of the WI-130 during normal operation.

*Display Modes tab*

**Display Modes** is the next tab, shown in Figure 7.

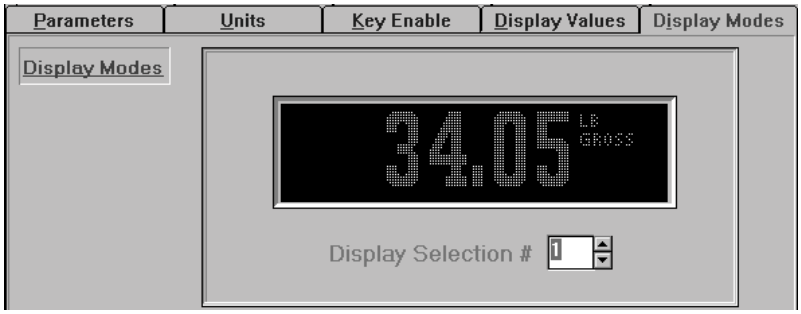



**Figure 7**
Display Modes dialog box

*See Appendix 1 for samples of displays.*

There are many display modes available. This dialog box lets you scroll through and select the style of display you want for your application. Displays vary in size of text, numbers, type of graphing, and soft key availability.

*Analog Output tab*

**Analog Output** is the next tab, shown in Figure 8.

*There are thirteen basis choices, however if the Analog Output pcboard is not installed, set Selection Basis to Disable.*

*Example:*
*4mA-20mA output*
*Minimum value = 0 lbs = 4mA output.*
*Max value = 1000 lbs = 20mA output.*

*Adjustments to the actual output of the analog output pcboard are only allowed through the front panel controls of the WI-130 and are dependent on the actual weight on the scale. Therefore, in the example above, a 1000 lb weight must be placed on the scale to allow adjustment of maximum value.*

*Offset Adjust and Span Adjust may have values between ±5000 counts.*

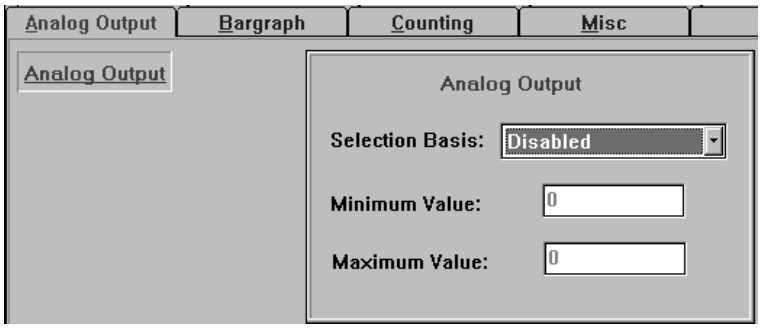



**Figure 8**
Analog Output dialog box

The Analog Output dialog box allows the user to select the parameters used with the optional Analog Output card as defined by calibration unit of measure.

**Selection Basis**
The Selection Basis combo box contains the active display values upon which the output of the analog output will be based.

**Minimum Value**
The Minimum Value text box allows the user to enter the lowest value that will be represented by the Analog Output.

**Maximum Value**
The Maximum Value text box allows the user to enter the highest value that will be represented by the Analog Output.

*Bargraph tab*

The next tab is **Bargraph**, shown in Figure 9.

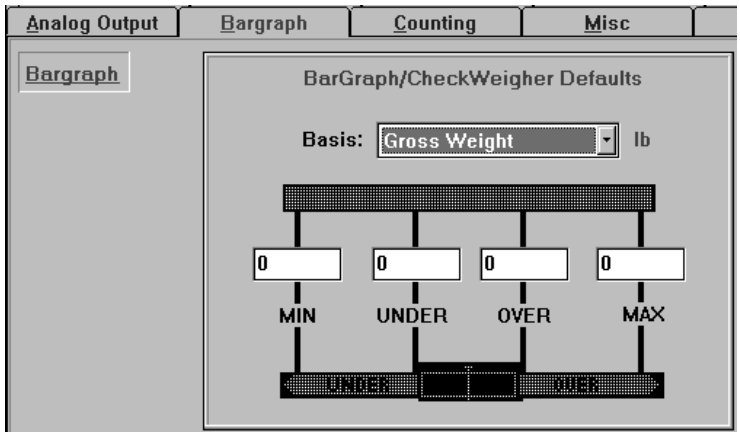



**Figure 9**
Bargraph/Checkweigher dialog box

Use this dialog box to enter parameters relating to bargraph or checkweigher functions, if the WI-130 is configured to operate in these modes. To operate in bargraph or checkweigher mode, the proper display mode must be selected.

**Basis**
Use the combo box to select one of thirteen choices for the Basis upon which the bargraph or checkweigher will be operating. It also is associated with the selected calibration unit selected in the Units dialog box.

**Min**
Enter the Minimum value of the Basis selection for which the bargraph or check-weigher will begin registering movement. If the WI-130 is set up with a bargraph, this value will determine when the graph on the display will begin moving. If the WI-130 is set up with a checkweighing display, the "Under" portion of the check-weigher graph will begin receding when this value is exceeded. The Minimum value can be set to any value, including negative values.

**Under**
Enter the Under value of the Basis selection. This value applies only to Check-weigher configurations and represents the Lower Acceptance Value for the checkweigher application. At this point, the "Under" portion of the checkweigher graph will disappear and the needle in the "Accept" range will be at it's extreme left position.

**Over**
Enter the Over value of the Basis selection. This value applies only to Check-weigher configurations and represents the Upper Acceptance Value for the checkweigher application. At this point, the needle in the "Accept" range will be at it's extreme right position and the "Over" area will not yet be visible.

**Max**
Enter the Maximum value of the Basis selection for which the bargraph or check-weigher will end registering movement. At this point, both the bargraph and checkweigher modes reach their maximum position and do not register further movement.

*Counting tab*

**Counting** is the next tab, shown in Figure 10.

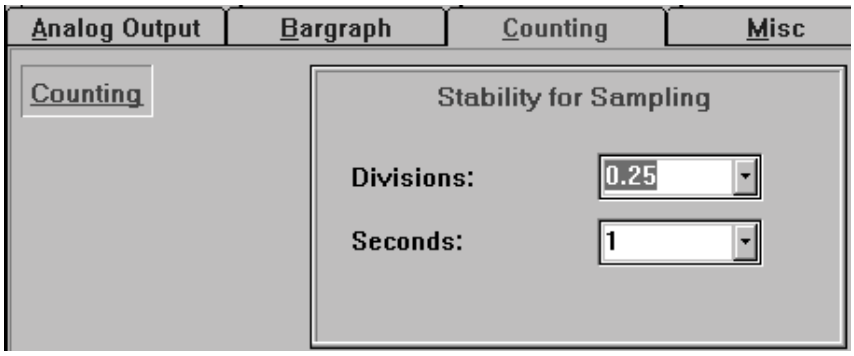



**Figure 10**
Counting dialog box

An application program with counting operation is required and then the Counting dialog box allows the user to select parameters relating to the stability of the scale during the parts sampling process.

*Motion and AZT settings do not affect the sampling process, but they do affect the counting process.*

**Divisions**

Select the number of scale divisions for stability. During the parts sampling process, this parameter determines how many divisions motion can occur on the scale while allowing the sampling process to occur. The smaller the number of divisions, the more stable the scale will need to be before the WI-130 will go into sampling mode. If the stability window is exceeded, the sampling process cannot occur and no piece weight is established, therefore no counting can occur.

**Seconds**
Select the number of seconds the weight must be within the Divisions range before the WI-130 will go into Sampling mode. In the setting above, the display must remain stable within 0.25 scale Divisions for one Second before sampling will occur and thus establish an accurate piece weight.

*Misc tab*

**Misc** is the next tab, shown in Figure 11.

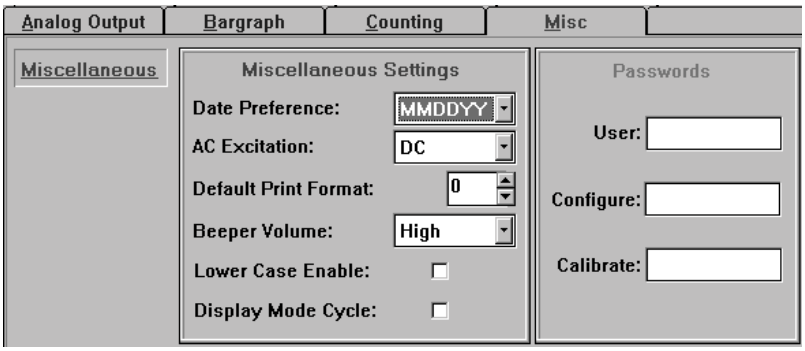



**Figure 11**
Miscellaneous dialog box

Record miscellaneous parameters for the configuration system in this dialog box. The first four selections use the combo box for entry and the last three use the Text box. The following items are included:

**Date Preference**

*The time and date may be sent in a variety of formats to the display, printer or computer. Format examples are: AM/PM, 24 hour, numerical reference, or spelled day and month.*

Allows you to set the WI-130 System Clock in Month-Day-Year format or Day-Month-Year format.

**AC Excitation**

You can select the analog weight sensor excitation to be a DC level of 10 volts or one of three frequencies for AC excitation which is useful for reducing weight shifts due to extreme temperature changes.

**Default Print Format**

*Default format 0 outputs the following information:*

**G 120000 LB**
**T 40000 LB**
**N 80000 LB**

Allows you to select which one of the 16 print formats you want to designate as the format used when the Print key is pressed. The default is format 0 and it is only sent to port #1.

**Beeper Volume**

Allows you to turn the WI-130 internal beeper off, or select from three volume levels.

**Lower Case Enable**

If you enable this option, letters in the soft key labels and all factory defined messages may be displayed in lowercase. If not enabled, all labels and messages are displayed in upper case.

**Display Mode Cycle**

*Display modes requiring BASIC text will show blank screen space if no WT-BASIC program exists to support screen text.*

Enable this mode for product demonstration of the display modes. If enabled, pressing the decimal key on the front panel causes the display to cycle through the display modes. Deselect this option to disable it and when using the unit as a weight indicator.

**Passwords**

The next three boxes allow you to change the following passwords:

User - (Default is 111.) Allows access to basic user parameters through the WI-130 keyboard.

Configure - (Default is 2045.) Allows access to configuration parameters.

Calibrate - (Default is 30456.) Allows access to WI-130 calibration routines.

*Time Out tab*

**Time Out** is the next tab, shown in Figure 12.

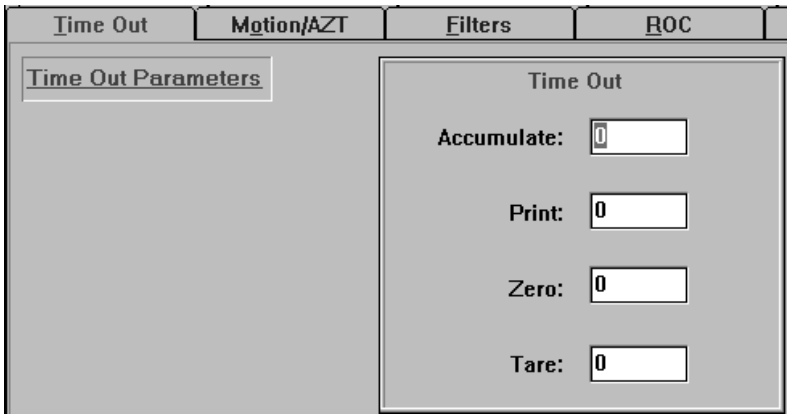



**Figure 12**
Time Out dialog box

Use this dialog box to set Accumulate Timeout, Print Timeout, Zero Timeout, Tare Timeout. This is the amount of time the WI-130 will wait for motion to cease and perform the function after the corresponding key is pressed and/or the event is queued up.

**Example:** If Zero Timeout is set to 3 seconds, when the **ZERO** key is pressed the unit will zero the scale if there is no motion. If there is motion and motion ceases within 3 seconds the unit will zero the scale. If motion doesn't cease the key press is aborted.

The same idea applies to the other three parameters in this dialog box.

*Motion/AZT tab*

**Motion/AZT** is the next tab. The dialog box is shown in Figure 13.

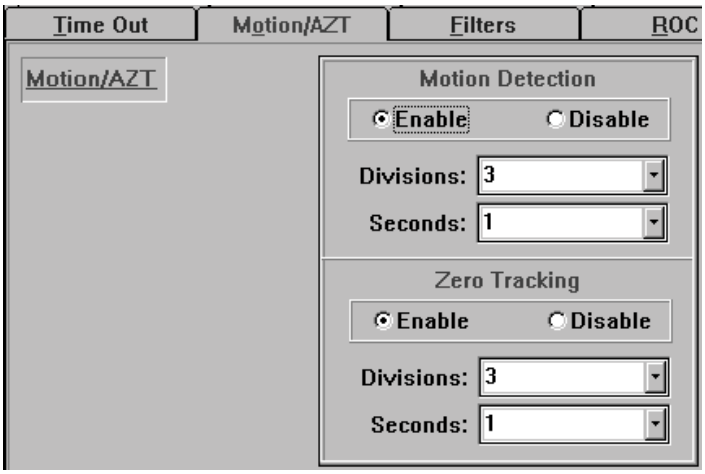



**Figure 13**
Motion Detection and AZT dialog box

In this dialog box you enable or disable motion detection and Automatic Zero Tracking or AZT.

If you enable motion detection you can set the motion detection window size in divisions and the time window in seconds. The default for motion detection is three divisions and one second.

For AZT the division size you pick defines a range above and below zero. When scale weight is inside this range for the number of seconds you picked, ½ of the weight will be zeroed. The indicator will repeat removing ½ the weight every X seconds. X being the number of seconds you have picked. This will be repeated as long as the condition exists or until the indicator display reaches zero. The default is three divisions and one second.

*Filters tab*

**Filters** is the next tab. This dialog box is shown in Figure 14.

*Harmonizer threshold is based on actual weight in calibration units, not division size.*

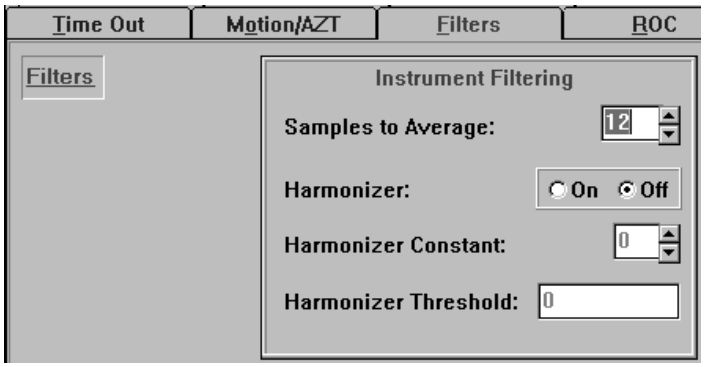



**Figure 14**
Filtering dialog box

This dialog box is used to configure the Harmonizer® filter. The Harmonizer can be customized to minimize interference from environmental conditions.

*The Harmonizer Constant choices are 0-10 in the SimPoser program but it may be best to make the selection in the "real world" through the front panel.*

The first combo box is called Samples to Average. The A-D weight conversion happens 60 times per second in the WI-130. In this combo box, set the number of conversions you want to average. For example, if you pick 15, the unit will average the weight values from the last 15 conversions every 1/60th of a second and uses that value for displayed data.

The next choice you have is for turning the Harmonizer filtering on or off. If you turn the Harmonizer filtering on, you need to set the Harmonizer Constant. This is a value from 0 to 10. Set the number low for small vibration problems and higher for more dampening effect.

*Pounds is the default calibration unit.*

The purpose of the Harmonizer Threshold is so the indicator will respond quickly to large weight changes by disabling Harmonizer temporarily. The Harmonizer Threshold is the amount of weight change, based on calibration units, beyond which the Harmonizer will be temporarily disabled. For example, if you set this to 10, a weight change greater than 10 pounds will disable the Harmonizer until the weight change during the sample time drops within the 10 pound threshold, then Harmonizer turns back on.

*ROC tab*

**ROC** is the next tab. This is the Rate of Change dialog box and it is shown in Figure 15.

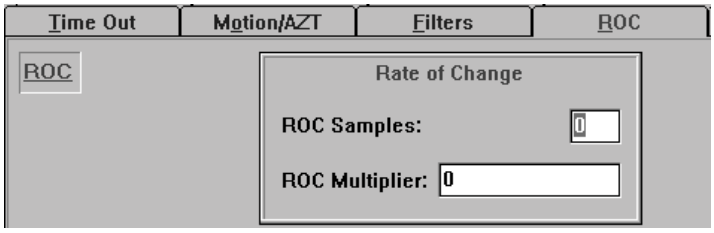



**Figure 15**
Rate of Change dialog box

The Rate of Change dialog box allows the user to set up a WI-130 Indicator to calculate Rate of Change for flow rate, or weight/time, applications.

**ROC Samples**

ROC Samples is the number of samples over which the rate of change of weight is determined. The WI-130 converts weight from A to D at 60 times per second. If ROC Samples is set to 60, the WI-130 is determining the rate of weight change over one full second.

*ROC is the rate of material flowing on or off the scale. If the flow of material is constant, the value displayed is zero. If the flow increases, the value is positive. If the flow decreases, the value is negative.*

**ROC Multiplier**

The ROC Multiplier allows you to enter a conversion factor to translate weight to some other unit of measure, such as gallons/hour or tons/minute, etc. or some other weight unit based upon the active unit of measure during a specific time. There is an example on the next page.

$$\frac{\text{Cal Unit}}{\text{Custom Unit weight in Calibration Units}} = \frac{1}{8} = 0.125$$

**ROC Examples:**

If pounds is your calibration unit, pick a sample value of 60 and a multiplier of 1. The display will show the rate of change in pounds/second.

For gallons of water/second set the sample value at 60 and the multiplier to 0.125. Water = 8 lbs/gallon (8 lbs is close enough for our example) so their are 0.125 gallons per pound. See formula to the left.

To get gallons/minute, do not change the sample size but rather multiply the 0.125 by 60 to get a value equal to gallons/minute (7.5). The display will then show you a rate of change in gallons per minute. (This is the flow over the last second not over a whole minute's time.)

*Serial Ports tab*

**Serial Ports** is the last tab and the serial ports dialog box is shown in Figure 16.

*Consult your peripheral device manual for proper serial port parameter selections.*

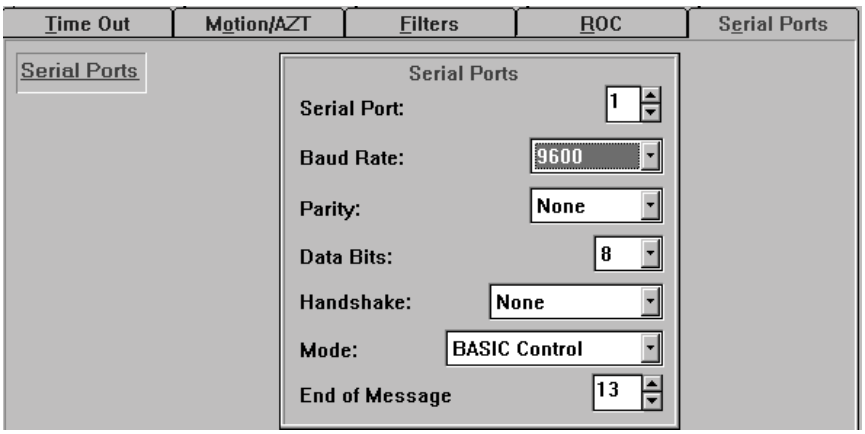



**Figure 16**
Serial ports dialog box

The Serial Ports dialog box allows the user to set the parameters for the two WI-130 serial ports. Each parameter is described below.

**Serial Port**

This selection switches between Serial Port 1 and Serial Port 2. Use the up/down arrow box to select the port or position the cursor inside the text box and type the number directly into the box.

**Baud Rate**

Use the combo box to select the Baud Rate from the list of selections.

Caution: If you use a baud rate above 19.2k, you need to use an error detection or correction protocol as well. Baud rate choices available are:

| | |
|---|---|
| 300 | 9600 |
| 1200 | 19200 |
| 2400 | 38400 |
| 4800 | 56700 |

**Parity**

Use the combo box to select the Parity setting from the list of selections. Choices available are shown in bold below:

| | Stop Bits | Data Bits | Parity |
|---|---|---|---|
| **None** | 1 or 2 | 7 or 8 | None |
| **Odd** | 1 or 2 | 7 | Odd |
| **Even** | 1 or 2 | 7 | Even |
| **Set** | 2 | 7 | None |
| **Clear** | 1 | 8 | None |

**Data Bits**

Use the combo box to select the Data Bits from the list of selections. Choices are 7 or 8.

*CTS is a hardware handshake (ready/busy) which requires two extra wires in your cable.*

*Xon/Xoff is a software handshake requiring no additional hardware.*

*Software must support this protocol in all devices.*

**Handshake**

Use the combo box to select the Handshake protocol from the list of selections.

Selections include:

| | |
|---|---|
| None - | No Handshake protocols are selected. |
| CTS - | Clear to Send protocol is selected. |
| Xon/Xoff - | Xon/Xoff protocol selected |
| Both - | Both CTS and Xon/Xoff protocols selected. |

**Mode**

Use the combo box to select the mode from the list of selections. The following mode selections are available:

BASIC Control - Control of the serial port is through the WT-BASIC program executing in the WI-130. When BASIC Control is selected, the End of Message box appears. Select or enter the ASCII value for the end of message character to denote the end of the serial transmission. For example, setting the end of message character to 13 would indicate that the transmission would end on the reading of a "carriage return" (ASCII character 13).

*A keyboard is an input device that a WI-130 is listening to or receiving data from. You may share a serial port with a printer that just listens or receives data from the WI-130.*

Keyboard - Control of the serial port is through an attached keyboard. As in Basic Control above, with this selection, an End of Message character must be entered.

Disabled - The serial port is turned off.

Multidrop - The serial port is configured in RS-485 Multidrop mode. When selected, the Address of the WI-130 in the multidrop loop must be indicated.

**End of Message (EOM)**

Enter the decimal number that represents the ASCII character the WI-130 expects as a signal for end of data transmitted to it from a communicating device such as a PLC or computer. When an EOM is received a COM1_MESSAGE or COM2_MESSAGE event is queued in the WT-BASIC program.You must then write BASIC code for the COM1_MESSAGE or COM2_MESSAGE event containing the GETCOM$ BASIC command.

**Program Button**

The next button on the SimPoser toolbar is **Program**. Figure 17 shows the screen which appears when you click on this button with your mouse.

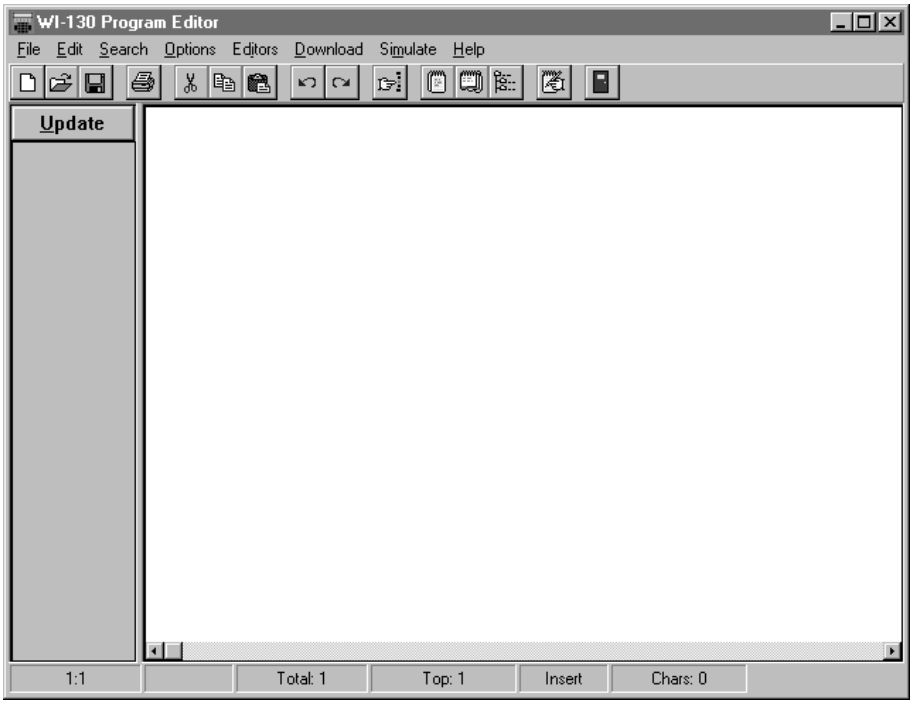



**Figure 17**
Program editor window

*If you forgot to save your file but have either downloaded to a WI-130 indicator or have tested your file by using the simulator, a copy of your .cfg file exists in C:\SIMPOSER\SIM\TEMPCFG.CFG*

You use the program editor window to enter the code for a WT-BASIC program. When you save the program you create in this window, all the setpoints, print formats, and configuration information you have selected are saved as a whole in a .cfg file. This file can be run in the simulator mode or downloaded to the WI-130.

WT-BASIC programming is the key feature of the WI-130's power and flexibility. Using the programming system in conjunction with the Setup configurations, print formats and the Setpoint configuration allow the WI-130 to be adapted to a wide variety of user defined applications.

**Program Editor Window Commands**

*The Program Editor window can be maximized by clicking the maximize button in the upper right corner of the window or be resized to suit your needs by clicking and dragging a corner of the window.*

The program editor window has several commands and buttons. The commands in this window are:

- File
- Edit
- Search
- Options
- Editors
- Download
- Simulate
- Help

## File

The file command operates the same way as the file command on the main toolbar but it also lets you access the Windows print setup dialog box and print the text currently in the Program Editor window.

## Edit

The edit command allows you to cut, copy, paste, and delete text in the editor window. You can also undo or redo actions and select all text in the window.

## Search

This command drops down a menu which helps you find and replace text, This can be very handy when the program is very long and you need to find a particular section for changes. Hot keys for these functions are F2 for Find, F3 for Find next, and F4 for Replace. When you access the Find function, the dialog box has a place to type in the text you want to find. You can cause the program to look for only exact matches to what you type in or words that contain the letters you type in.

This command also has a **Go to line** feature. This allows you to move to any line of the program simply by typing the line number in a popup dialog box.

The last feature in the search command is called **Program Errors**. This is used to retrieve a list of errors in your program. If you simulate a program and it contains an error or errors, SimPoser creates an error file. You can retrieve this file by clicking on **Program Errors** in this drop down menu or clicking the program errors button on the tool bar (  ). A dialog box pops up listing the error and the line it is on. This error file is automatically deleted when you load a new configuration file or test an updated version.

Use the **Set Marker** command to set place markers in the program. Use the **Go to Marker** command to find a previously set marker in the program. This is helpful when trying to find a routine you are currently creating.

*Use the* **Ctrl +Tab** *keys to place tabs in your text. Pressing the tab key alone selects the buttons on the button bar of the program editor window.*

## Options

This command lets you customize the editor window and its function. Each item is briefly described below.

Auto-indentation — If you enable this function with a checkmark, your lines of text will automatically indent the same as the previous line of text.

Font — Click on this option to bring up a Font popup box. Use this to choose what type font, font style and type size is used in the editor window.

Tabs — This option lets you set three types of tabs and customize the tab size.

- Fixed Tabs - Pressing **Ctrl + Tab** keys moves the cursor to the next tab position. This inserts spaces not true tabs.
- Real Tabs - Pressing **Ctrl + Tab** keys moves the cursor to the next tab position. This inserts real tabs into the text, not spaces.
- Smart Tabs - Pressing **Ctrl + Tab** keys aligns the cursor on the current line to the position of the next closest text on the previous line. This inserts spaces not true tabs.
- Tab Size - Choose a tab size.

## Editors

Click this command to access the other editing windows without returning to the main toolbar.

## Download

Click this command to download the program to your WI-130. You are given a choice of which port to use. The F11 and F12 keys are hot keys for downloading to Com1 and Com2 respectively.

## Simulate

Choose this command to start the WI-130 simulation using the program you are working on.

## Help

Choose this command to open help documentation on SimPoser operation.

## Program Editor Window Buttons

There are several buttons on the button bar of the editor window. Below is a description of each.

*New File Button*
*Open File Button*
*Save File Button*

These three buttons are for opening a new file, opening an existing file or saving a file you are currently working on.

*Print Button*

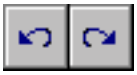
This button is for printing the program. The following dialog box appears when you click on this button:


You can insert a header and footer into the printout. You can also choose what information to print: Configuration; Program; Print Formats; and Set Points.

*Cut Button*
*Copy Button*
*Paste Button*

These three buttons are for cutting, copying and pasting text.

*Undo Button*
*Redo Button*



These two buttons are for undoing and redoing an action.

*Find Text Button*

This button brings up a dialog box for finding text.

*Toggle Event List Buttton*

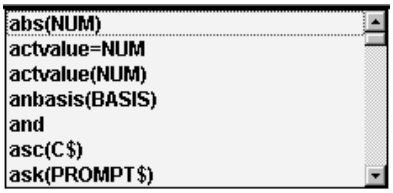
This button is for toggling the event list on and off. Use the elevator button or the arrow keys to scroll through the entire list of events available to you.

This is a list of predefined events and subroutines you can use in your program. Double click an event name to place it at the end of your program. If the event name already exists in your program, double clicking the event name will cause the event to be found and highlighted in the program. After the event name is placed in your program you need to fill in your particular commands.

*Teaching BASIC programming language is beyond the scope of this manual. There are many good reference books on the subject. All the WT-BASIC commands available to you in the SimPoser program are listed in* Appendix 2: The WT-BASIC Interpreter Command Set. *Use these commands to build your program.*

*Toggle Key Word List Button*

This button is for toggling the WT-Basic keyword list on and off. Below are some of the items in this list. The entire list is found in *Appendix 2* of this manual.



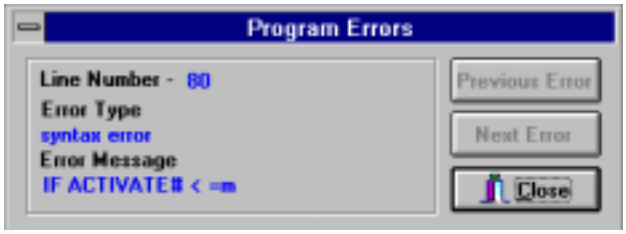

When you double click a selection, it will appear in your program where your cursor is located. You will need to replace the syntax placeholders with your values or strings.

*Toggle Sub Routine List*

This is a list containing currently used event names and self-created subroutine names that are currently in your program. The Update window on the editor screen will appear or disappear when this key is pressed. By double clicking on an event or subroutine name in this list, your cursor is relocated to the beginning of that subroutine.

*Program Errors Button*

This button is for bringing up the program error dialog box shown below. This box will only exist after an error is found in the program during simulation. When the simulation is canceled, this box will appear in the Program Editor window. If your program has multiple errors, you can move through them using the Previous Error and Next Error buttons. You can close this window by clicking the Close button. You can make it reappear by clicking the Program Error button again or clicking on Program Errors under the Search command at the top of the Program Editor window.



**Format Button**

The next button on the toolbar is **Format**. Use this to control the way a particular printer connected to a WI-130 will print a document or bar code label. The program is capable of 16 output formats.

Click the format button and the format editing window shown in Figure 18 appears.

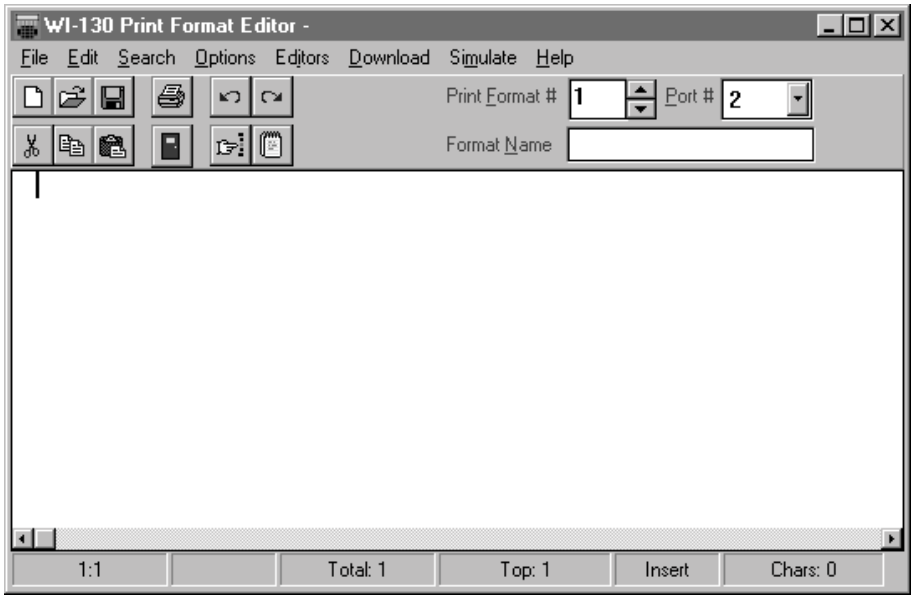
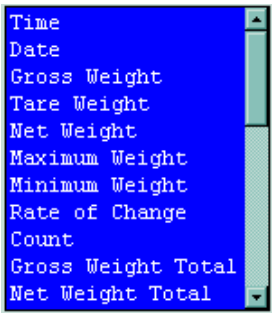
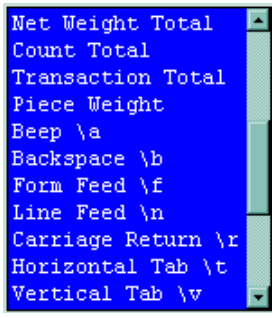


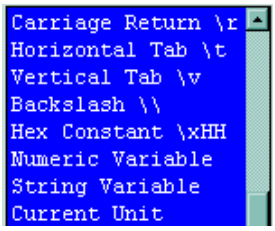

**Figure 18**
Print format editing window

The commands along the top of the window are the same as those described in *Program Editor Window Commands*.

The buttons are also the same as those in the program editor window with one exception. The lower right button brings up a list of terms. The list is shown below.




Choose the number of the print format you want to create or work with. Choose which port you want it printed from and give the format a name by typing it in the Format Name box. You may use any number of print formats, from one to 16, with each configuration file. You do not have to use them in order: it is possible to use Format 1, Format 9 and Format 15, for example, or any other combination you desire.

*In a WT-BASIC program, actual values that can change during the process are represented by "VARIABLE NAMES".*

*In WT-BASIC there are variables that are predefined. These are referred to as "system variables", such as "gross weight", "count", etc. These are available under TERMS. You may also create and use your own variables in WT-BASIC.*

*To get the current unit of measure label to print after a weight value, place the system variable {curunit$} after the callout.*

**\S** *may be used in a print format to comment a section out or prevent a carriage return from being sent. Any character to the right of a \S will not be printed.*

For printed tickets, you use the editor to lay out a format in logical order. The example in Figure 19 shows a typical truck scale ticket format, with item legends on the left side of the ticket and the actual values for the variables to be printed (in brackets) on the right side. As you can see in the adjacent sample ticket, the format looks very much like the actual ticket that is printed.

**Example**

**Format**

```
{Cname$}

Operator: {opid$}

Truck ID:  {trID%,6.0}
Date    :  {date$}
Time In :  {trTIMEIN$}
Time Out:  {trTIMEOUT$}

Gross   : {trGROSS#,6.0} {curunit$}
Tare    : {trTARE#,6.0} {curunit$}
--------------------

Net     : {nET#,6.0} {curunit$}

Signature:



-----------------------------------
\r\r\r
```

**Actual Ticket**

```
HIGHLAND STONE

OPerator: JDB

TruckID :     69
Date    : 12-31-99
Time In : 09:33:23
Time Out: 12:59:59

Gross   : 85280 lb
Tare    : 10500 lb
-----------------

Net     : 74700 lb

Signature :


-------------------------
```

**Figure 19**
Print format example

*Printing Titles and Legends*

For fixed legends or titles, type the information in position in the edit area of the screen, exactly as you want to see it on the finished ticket. Use the space bar to move to the desired position and the **ENTER** key for additional lines. Lay out the ticket to match the design you desire. Once you have some of the information laid out you may move around the screen using the cursor arrow keys on your keyboard.

*Printing Actual Values*

The actual values that will be printed on the ticket when run with the WI-130 require a different method of placement. Values from the WI-130 must be surrounded with brackets ( { } ) to tell the Print Format that this information will be coming from data stored, generated or collected by the WI-130. Information such as scale weights, accumulated total, transaction counts, piece weights, part counts, and Variables must be enclosed in brackets in the Print Format. This is done automatically when using the TERMS list box shown on the previous page.

*Selecting Actual Values*

An alternative to typing in actual values is to use the terms list. Click the terms list button (  )to make the list appear.

1. Place the cursor in the position on the screen where you want to place a new value.

2. Double click with the left mouse button on the value item you want to place in your format.

   The term, in its correct syntax, is placed into the print format.

3. Repeat steps 1 and 2 until you have the terms you want in the window.

4. To remove the list box, click the right mouse button with the mouse pointer inside the list box or click on the terms button in the tool bar.

*Depending on the type of printer you use, you may have to use the following commands to make it respond correctly:*
*\r "carriage return"*
*or*
*\n "line feed"*

The list contains several "format codes" such as backspace, line feed, carriage return, beep and others designed for special functions when the ticket is printed.

Two selections, "Numeric Variable" and "String Variable" require that you replace the words 'Numeric Variable' or 'String Variable' with the actual variable name that you created in WT-BASIC.

Defining the way numeric variables are printed is accomplished by the following syntax:

**SYNTAX:** **{VARIABLE,WIDTH.PRECISION}**

See the example to the left or the print format example on the previous page.

*Example: {GROSS, 3.2}*

*This will print the gross weight as follows: 500.01*

Width and Precision are optional expressions. You do not have to use them. By default, a numeric variable is right-justified. Use *width* to define a minimum width of the numeric variable. Use a negative width to left-justify the numeric variable. Use *Precision* to designate the number of positions to be printed to the right of the decimal point.

When defining a string variable, width is used to define a minimum width. If the string variable does not take the minimum width to print, spaces are printed to fill the gap. String variables are Left Justified by default and will print Right Justified if a negative width is used.

*When TIME$ or DATE$ are used in a program's print statement, you may use the following syntax:*
**TIME$(n)** *or* **DATE$(n)**

TIME$ and DATE$ are the two string system variables that have the following optional syntax in a print format:

**{TIME$,0.n}**

When a TIME$ is printed, (n) is used to tell the system what format of TIME$ to print. A value of 0, 1, 2, or 3 is used in the (n) expression to print TIME$ in the following formats:

**TIME FORMATS**

| | | |
|---|---|---|
| **0 =** | **24 Hour format with seconds** | **18:00:00** |
| **1 =** | **AM/PM format with seconds** | **1:00:00 AM** |
| **2 =** | **24 Hour format without seconds** | **18:00** |
| **3 =** | **AM/PM format without seconds** | **1:00 AM** |

**{DATE$,0.n}**

When a DATE$ is printed, (n) is used to tell the system what format of DATE$ to print. A value of 0, 1, 2, or 3 is used in the (n) expression to print DATE$ in the following formats:

| **DATE FORMATS** | **M/D/Y mode** | **D/M/Y mode** |
|---|---|---|
| **0 Numbers** | **03-14-99** | **(14/03/99)** |
| **1 Spelled Month** | **Mar 14, 1999** | **(14 Mar, 1999)** |
| **2 Numbers with Day of Week** | **Mon 03-14-99** | **(Mon 14/03/99)** |
| **3 Spelled Month with Day of Week** | **Mon Mar 14, 1999** | **(Mon 14 Mar, 1999)** |
| **4 Numbers with 4 digit year** | **03-14-1999** | **(14/03/1999)** |

**Setpoint Button**

The next button on the SimPoser toolbar is **Setpoint**. Press this button and the setpoint window shown in Figure 20 appears.

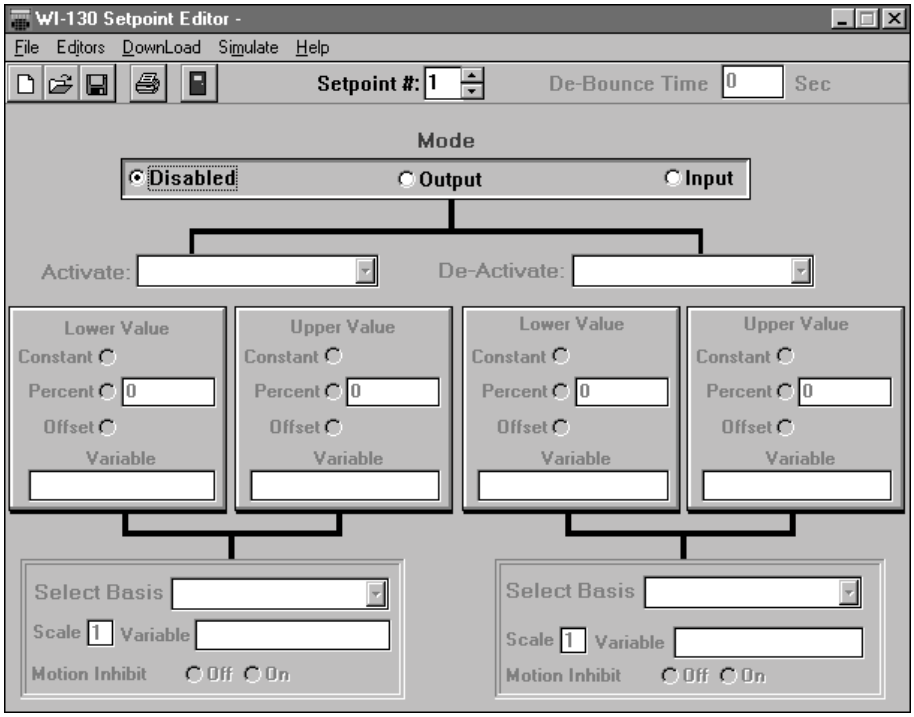



**Figure 20**
Setpoint window

*PLEASE NOTE: Applications using setpoints should be handled and tested with great care to assure that the system operates in conformance with the stated objectives and design parameters of the system. Always completely test all parameters to the fullest! Manual back up controls should always be installed on the most critical components to assure that the system can be adjusted and/or shut down manually, should the system malfunction or deviate from the proper operation sequence.*

The Setpoint window allows you to define the condition you wish to monitor or detect thus causing the WI-130 program to trigger a new event (SetPoint Act or Deact).

The Setpoint capabilities of the WI-130 range from very simple applications such as turning on an alarm when a weight value is reached, up to very sophisticated batching applications specifically tailored to a customer's requirements. The setpoint system is designed for almost limitless flexibility, to allow the application designer to use the system in conjunction with WT-BASIC programming to solve a wide variety of scale user requirements.

Not limited to batching, however, the WI-130 Setpoint System is adept at solving all types of application needs, such as truck scale control, checkweighing systems, conveyor systems, PLC applications or any other situation where external feed-back and control is required.

Use the setpoint window to configure up to 32 setpoints in the WI-130 System. This dialog screen provides a visual representation of the configuration of a particular setpoint so that you can quickly glance at the form and see how the setpoint is configured.

The screen is separated into the following divisions:

- Setpoint #
- Mode
- De-Bounce Time In Seconds
- Activate/De-Activate Condition - Output Setpoint Only
- Lower/Upper Values - Output Setpoint Only
- Select Basis - Output Setpoint Only

Each of these areas of the screen is explained below.

*Setpoints are commonly used as outputs without I/O modules for program interactions like changing the display, continous output, or diagnostic flags.*

***SetPoint #***

Determines which setpoint is currently active. Use the up/down arrow box to select the active setpoint or enter the number directly into the text box. You may not enter a number smaller than 1 or greater than 32.

***Mode***

Mode determines what level of operation the setpoint is in. The possible choices are:

- Disabled - Setpoint not active
- Input - Setpoint receives input from external device
- Output - Setpoint sends signal to external device or setpoints are commonly used as outputs without I/O modules for program interactions like changing the display, continous output, or diagnostic flags.

***Disabled***

This is the default state for a setpoint and means that the setpoint is not in use with the current configuration.

***Input***

Select Input to receive a signal from an external device, such as a switch. The state of an Input setpoint can be detected using commands in the WT-BASIC programming system. When this mode is selected, the only parameter to choose is De-Bounce Time. De-Bounce time is the time in seconds allotted to receive switch activations, to prevent receiving a double signal. For example, if a De-Bounce time of 1 is entered in the text box, the Input setpoint will only receive one activation input per second, no matter how many activations of the switch occur during the 1 second period.

***Output***

Select Output to send a signal to an external device to activate or deactivate the device. Output requires setting various parameters to achieve the proper control of external equipment.

***De-Bounce***

When the setpoint is used as an output, the debounce time can be used as a timer interval for events to occur by selecting IMMEDIATE in the Activate and Deactivate drop down boxes.

For example, with Activate/Deactivate both set to immediate, and De-Bounce set to 1.0 seconds, the setpoint activates or turns on for one second, then deactivates or turns off for one second, then turns back on for one second. The time between activate events is two seconds.

***Activate/Deactivate Condition***

Each Output setpoint must have a condition that activates and deactivates the setpoint. By clicking the combo box next to Activate and Deactivate, a list is displayed for you to select this condition. You must select both an Activate and Deactivate Condition in order for the setpoint to work properly. Make sure you fill in both sides of the form.

The following conditions are available:

- Above
- Below
- Inside
- Outside
- Motion
- No Motion
- Center of Zero
- Not Center of Zero
- BASIC Control
- Immediate
- Accum Operation
- Print Operation
- Zero Operation
- Tare Operation

*In addition to the conditions defined by the setpoint window, you may also force the setpoint on or off from your BASIC program by using the BASIC commands SETPT ON or SETPOINT OFF.*

Selection of the above conditions determine the additional parameters that will be required to make the setpoint functional. Each of these will be described in the following sections and each section will apply to both Activate and Deactivate.

**1. Above/Below**

Above/Below indicates that the setpoint will activate or deactivate when the scale reading is either Above or Below the selected value. When either of these selections is made, the "Lower Value" box on the screen will become active.

Lower Value

Indicates the value that the setpoint will activate on. Use the mouse to click on the selection in the Lower Value box.

**Constant -** Indicates that the value will be a fixed value, using the same standard as the calibration Unit of Measure selected for the system, such as 5000 lbs. When Constant is selected, the text box to the right (Value box) becomes active. Enter the constant value in this box.

**Percent -** Indicates that the value will be a percentage of a Variable amount. When selected, the Variable text box is enabled. Enter the percentage in the box to the right of the form (Value box) and the Variable Name in the box under "Variable". Variables can be used in the WT-BASIC program to control operations of the setpoint system. You may enter both positive and negative percentage amounts and amounts greater than 100% (i.e. 1000%)

**Offset -** Indicates that the value of a Variable from your BASIC program, plus this offset value in cal units, will control this setpoint.

For example, the value may be 100 lbs. more than the value currently contained in variable "TestAmount" (a variable defined in the WT-BASIC program). For this example, you would enter 100 in the Value text box and the name "TestAmount" in the Variable text box. You may use both positive and negative Offset amounts.

***Select Basis***

The Basis defines the Weight or other value that the selected setpoint will activate on. By clicking the Combo box next to Select Basis, a list is displayed for you to select from. Depending on your selection, additional selections in the Select Basis box will be enabled for you to select from.

*Gross Weight, Net Weight, Tare Weight, Minimum Weight, Maximum Weight, Rate of Change, Gross Weight Total, Net Weight Total, Count*

If one of the above Basis selections is made, you need to make the following additional selections from the Select Basis box:

Scale Number - Enter the scale number that this selection will apply to in the Text box.

Motion Inhibit - Click On to enable Motion Inhibit, or Off to disable Motion Inhibit.

***Count Total, Transaction Total, Piece Weight***

If one of the above Basis selections is made, you need to make the following additional selections from the Select Basis box:

Motion Inhibit - Click On to enable Motion Inhibit, or Off to disable Motion Inhibit.

***Variable***

If the above Basis selection is made, you need to make the following additional selection from the Select Basis box:

Variable - Enter the name of a Variable used in the WT-BASIC program for this application.

Motion Inhibit - Click On to enable Motion Inhibit, or Off to disable Motion Inhibit.

**2. Inside/Outside**

Inside/Outside indicates that the setpoint will activate or deactivate when the scale reading is either Inside or Outside a range of selected values. When either of these selections is made, both the "Lower Value" and "Upper Value" boxes on the screen will become active allowing you to enter a range of values to meet this criteria. Make your selections in the same manner as described in *1. Above/Below*, but make sure you fill in selections for both the Lower and Upper Value boxes. Both will become active when either of these selections is made.

**3. Motion, No Motion, Center of Zero, Not Center of Zero**

The setpoint will activate or deactivate when any one of these chosen conditions is met. The only condition that has to be selected is the Scale Number. Enter the number in the adjacent text box.

**4. BASIC Control, Immediate, Accum Operation, Print Operation, Zero Operation, Tare Operation**

The setpoint will activate or deactivate when any of the above operations are in control or are activated, as in the case of the **Zero** key being pressed and the zero operation successfully completed. There are no additional parameters to be set when one of these conditions is selected. The entire form below Activate/De-Activate will become disabled.

**Example 1:**

Following are several examples of how setpoints operate. There is much more you can do with setpoints once you familiarize yourself with all the variables and conditions you can use to trigger the setpoints.

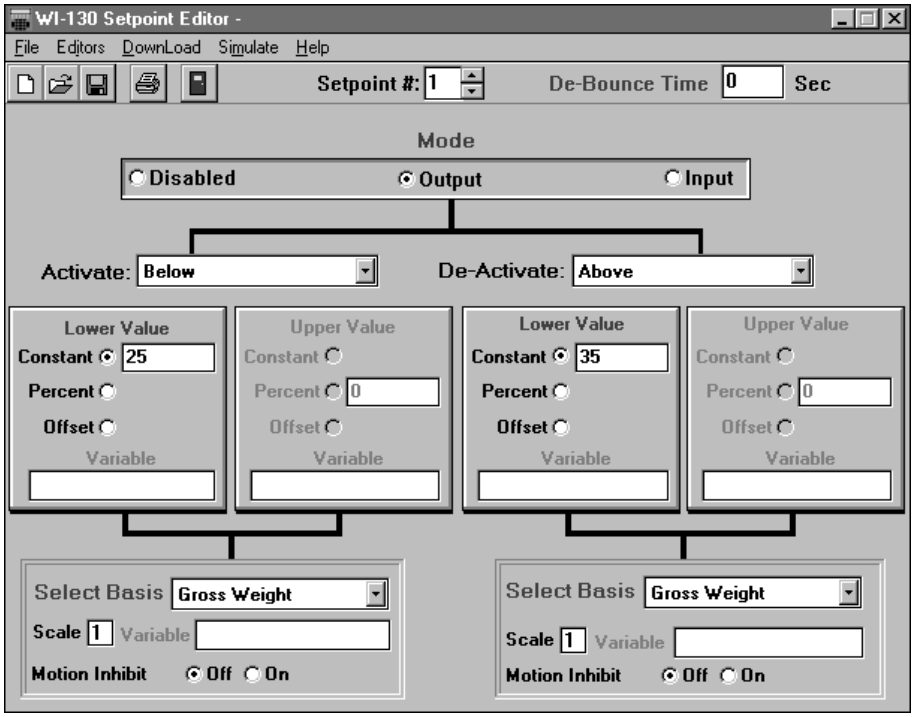



**Figure 21**
Example 1

In example 1, setpoint 1 is an output. The setpoint will activate below a constant value of 25 lbs gross weight. When the gross weight goes above 35 pounds, the setpoint will deactivate. Weight readings are coming from scale #1 and the motion inhibit is off.

**Example 2:**

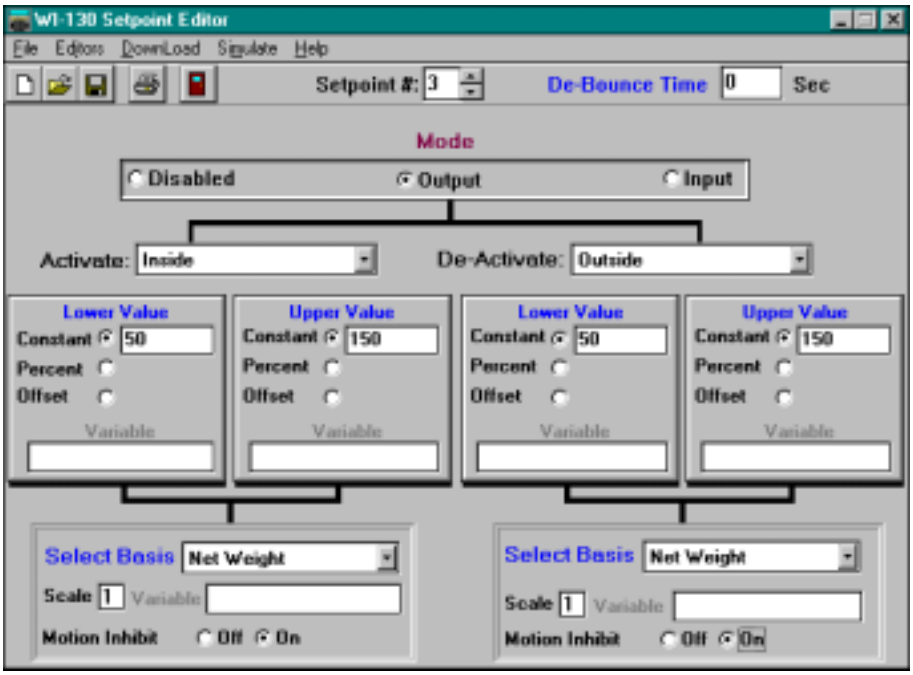



**Figure 22**
Example 2

Figure 22 shows the setpoint window for example 2. In this example we have chosen Setpoint 3 as an output. The setpoint will activate when the net weight value is between 50 and 150 pounds and will deactivate outside of this range.

**Example 3**

In this example we will use two setpoints #1 (see Figure 23) and #2 (see Figure 24).

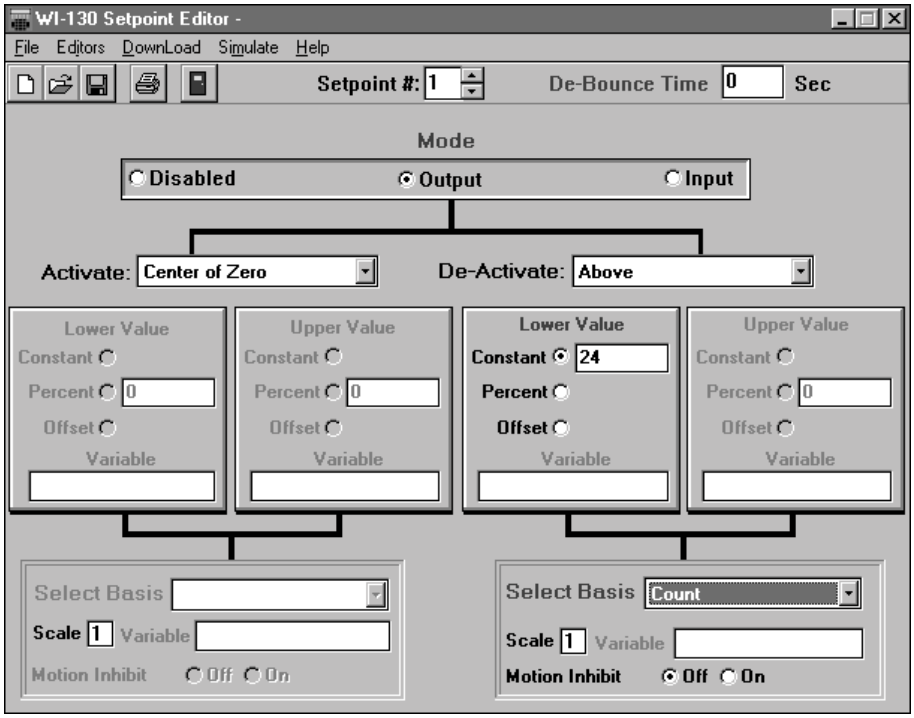



**Figure 23**
Example 3-Setpoint #1

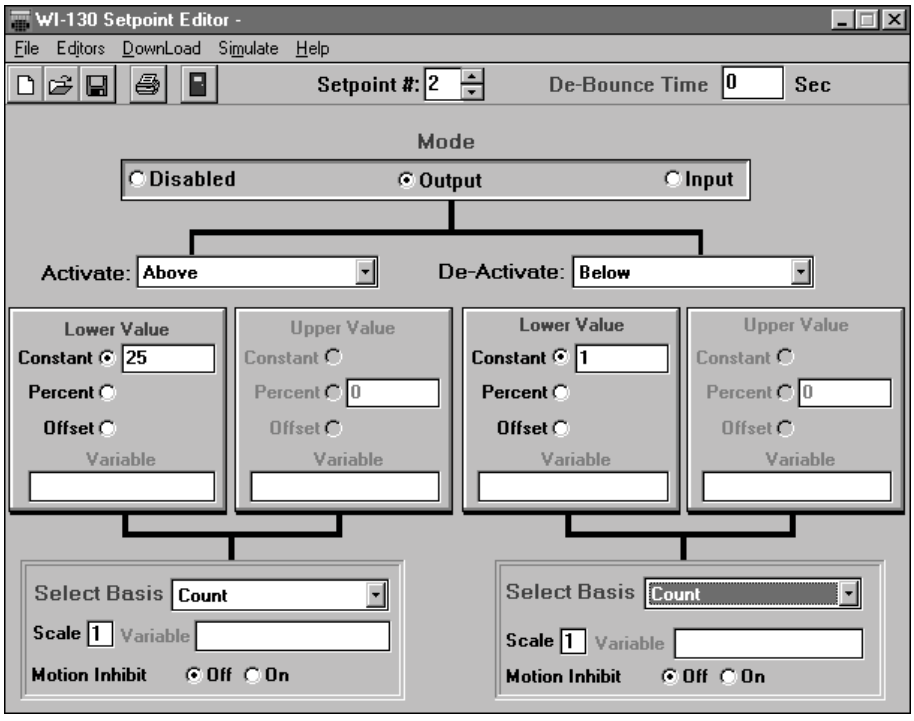



**Figure 24**
Example 3 - Setpoint #2

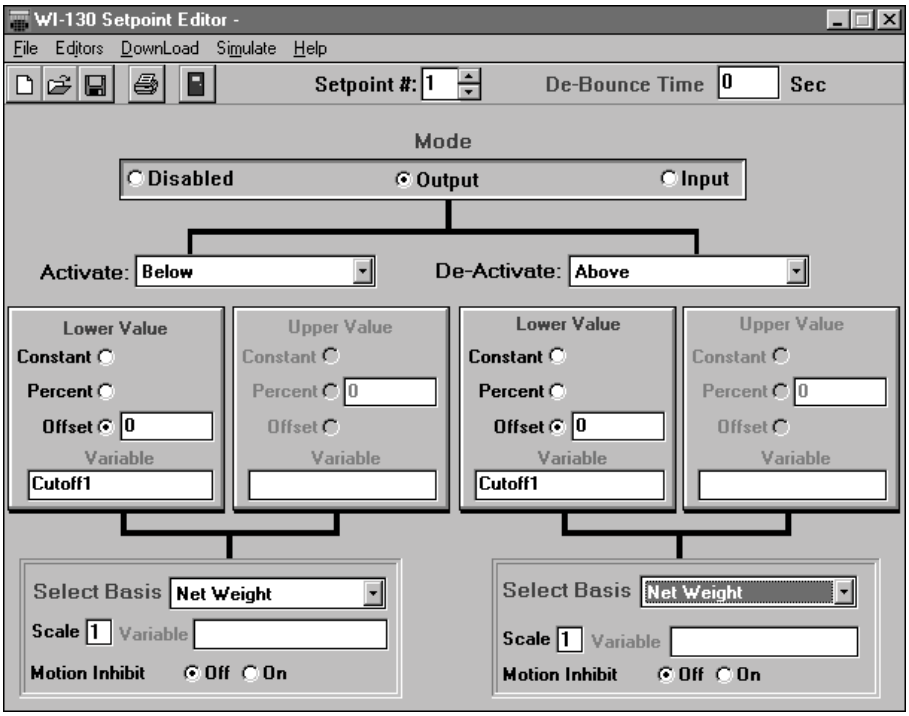
In this third example we are running a **very** simplified batching sequence. We are opening a valve or gate to drop gumballs into a hopper on a scale. We want to count out 25 gumballs and dump the hopper, then repeat the process. (Keep in mind that to run a batching procedure in real life it will probably be necessary to use a WT-BASIC program in conjunction with the setpoint configuration.)

Setpoint 1 controls the valve or gate to allow gumballs to roll in to the hopper. It is set up as an output and activates when the gross weight is at center of zero. The gumballs will roll into the hopper until the count is over 24. The valve will shut and the next setpoint (#2) which controls the dump gate on the scale hopper will activate since the count is now 25 or above. When the count on the scale hopper drops below 1, the gate closes and the fill valve reopens.

As was stated earlier, this is an extremely simplified batching program. With a WT-BASIC program you can design very sophisticated batching sequences.

**Example 4**

Figure 25 shows two screens using variables. The explanation for these is below.



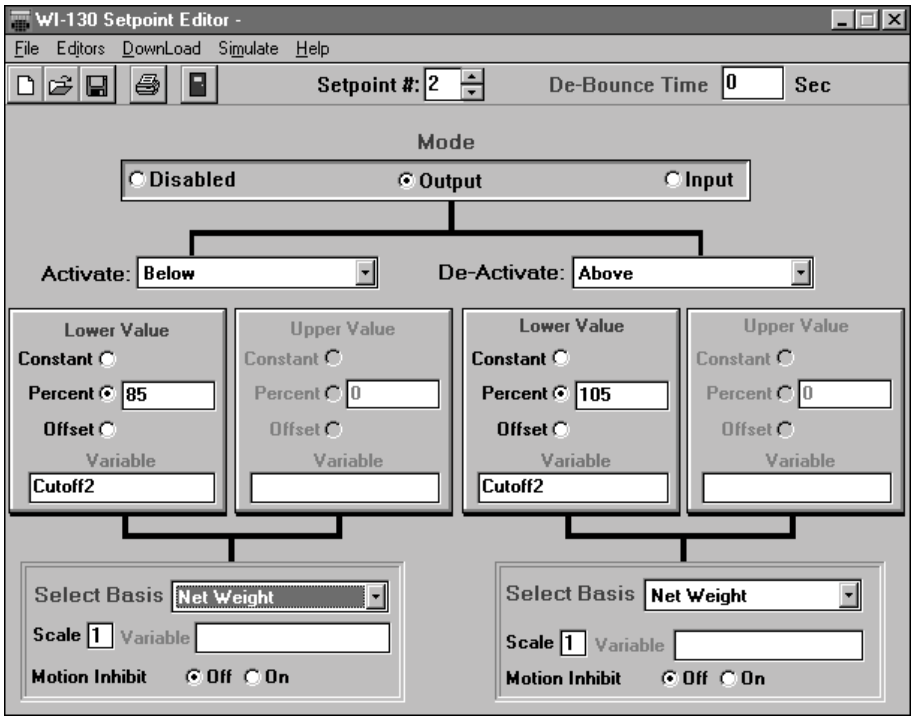
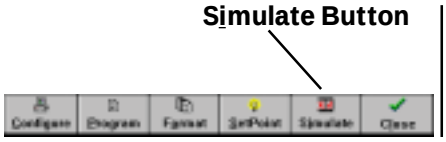



**Figure 25**
Example 4

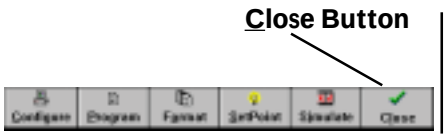
In this fourth example we are controlling two setpoint outputs as cutoffs, similar to the way older indicators have worked. The first screen in Figure 25 refers to Cutoff1 as the variable name for control of setpoint #1. The second screen refers to Cutoff2 as the variable name for control of setpoint #2. Both variable names will have values assigned to them in the WT-BASIC program via the front panel.

Setpoint #1 activates when the net weight is below an offset of 0 of the variable Cutoff1. It deactivates when the net weight is above an offset of 0 of the variable Cutoff1.

Setpoint #2 activates when the net weight is below 85% of the variable Cutoff2. It deactivates when the net weight is above 105% of the variable Cutoff2.




The next toolbar button is **Simulate**. Click this button to start the WI-130 simulation. This is the same as clicking the Simulate command on the command line. This was covered earlier in this manual.




The last toolbar button is **Close**. Click this to close your SimPoser program. If you have made any changes in a program, a dialog box will pop up asking if you want to save the changes.

# Application Program Summaries

## Batching

| | |
|---|---|
| 2spd1ing.cfg | Two speed single ingredient batching application. |
| 4ingbatc.cfg | 4 Ingredient single speed batching application. |
| Flow8.cfg | Demo using ROC to control flow rate (using analog output) for Mechanical device control. |
| Flow9.cfg | Demo using ROC to control flow rate(using setpoints) “LB/HR”. |
| Jogbatch.cfg | Jog softkey example. |
| Wi-1106.cfg | Multi-scale (3-scales), dual cutoffs, single speed. |

## Checkweighing

| | |
|---|---|
| Chkweig2.cfg | Advanced checkweigher program with set points. |
| Setchk1.cfg | Simple checkweigher program. |

## Counting

| | |
|---|---|
| Count5.cfg | Simple counting scale using the Dribble sample method. Printing –sample size, piece weight, net weight, and count total. |
| Count6.cfg | Simple counting scale offering both Bulk & Dribble sample methods. Printing –Barcode labels out of Port number 2 using print Formats 4, 5, and 6 for a label printer. |

## InMotion

| | |
|---|---|
| Inmo5.cfg | Simple conveyor scale application. |
| Firminmo.cfg | Firmware level inmotion system application |

## Lift Truck

| | |
|---|---|
| 11894_0B.cfg | Simulcast only. Simple counting application. ID entry, sampling, reverse sampling, pcwt entry, WP-23x printout, PC comma delimited printout, quick count. |
| 11896_0C.cfg | Simulcast only - 500 channel accumulation application. Reporting, WP-23x printout, PC comma delimited printout. |
| 11897_0B.cfg | Simulcast only - 500 accumulator, multiple field database. This application is shipped with every standard Simulcastä . |
| 11897A0B.cfg | Simulcast only. Same as 11897_0B, but port 1 prints to the printer. |

## Miscellaneous

| | |
|---|---|
| 1corner3.cfg | Application for corner balancing weight sensors through a J-box. |
| Fish.cfg | Fish swimming on screen. |
| Fishscr.cfg | Screen saver application of fish swimming on screen. |
| Harmhelp.cfg | Application for the setup of filtering values. |
| Lamptst.cfg | Application for the testing of all dots on display. |
| Sys_err.cfg | Example application for using SUB SYSTEM_ERROR and the ERR keyword. |
| Tr-icon1.cfg | Trade show truck scale bitmap example. |

## Printers

| | |
|---|---|
| Conout.cfg | Provides continuos output of GROSS weight out port 1. |
| Enq_cr | Remote request from a PC application |
| Keithley.cfg | Configuration for RS-485 to Analog Output converters. |
| Orion.cfg | Eltron Orion sample label printer application. |
| Orion1.cfg | Eltron Orion sample label printer application. Examples from Orion printer price sheet. |
| Rs485.cfg | RS-485 multi-drop example application. |
| Tm_295a.cfg | Epson TM-295 ticket printer example. |

## Rail (Track Scales)

| | |
|---|---|
| Wt-line4.cfg | Standard Weigh-Line application. |

## RD (Remote Display)

| | |
|---|---|
| Rd4000_6.cfg | RD-4/6000 with 6 digit display sample. |
| Rd4000_8.cfg | RD-4/6000 with 8 digit display sample. |
| Rd-125.cfg | RD-125 sample application. |

## Sales Demos

| | |
|---|---|
| Chkweigh.cfg | Checkweigher sales demonstration. |
| Demoapps.cfg | Multiple programs for sales demonstrations. |
| Partct.cfg | Part Counter sales demonstration. |
| Slide.cfg | Slideshow for sales demonstration. |
| Specs.cfg | WI-130 specification slideshow for sales demonstrations. |
| Tareiso.cfg | Multi-Channel tare database with ISO-9000 tracking information. |
| Trkinout.cfg | Inbound/Outbound sales demonstration. |
| Wi110.cfg | WI-110 with 10 tare registers. |
| Wi110bat.cfg | WI-110 with dual cutoffs. |
| Wi120.cfg | WI-120 emulation sales demonstration. |

## Truck Scale

| | |
|---|---|
| 10544f.cfg | Axle weigh truck scale. |
| 9459d.cfg | GTN or 70 foot axle weigh scale. |
| Freetrk5.cfg | Inbound/Outbound truck scale application example. |
| Freetrk6.cfg | Dual indicator Inbound/Outbound truck scale application example. |
| Tare100.cfg | Multi-Channel tare database. |

**Microsoft® Word documents explaining these programs are available in the C:\wt\wi130\docs folder or the folder you designated during installation.**

# Appendix 1: Display Samples


#1


#2


#3


#4


#5


#6


#7

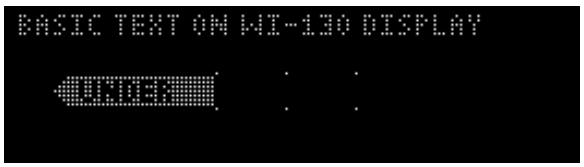

#8


#9


#10


#11


#12




#13


#14


#15


#16


#17


#18

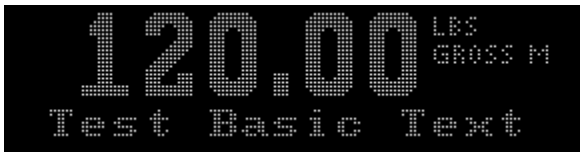

#19


#20


#21


#22


#23




#24


#25


#26


#27


#28


#29


#30


#31


#32


#33

The following are multi-scale displays. If all the lines are not used for scales, they are available for Basic text. #34 and 35 are small basic text. #36 & 37 are large basic text.


#34


#35


#36


#37


#38


#39

# Appendix 2: WT-BASIC Interpreter Command Set

## WT-BASIC

*If you hold in the **CLEAR** key when powering up the WI-130, the WT-BASIC program which is currently resident in the indicator will be temporarily disabled until the next power up.*

*This is used when troubleshooting to eliminate the BASIC program as the source of a problem.*

The following pages contain the WT-BASIC inpterpreter command set you use to create programming for the WI-130.

This command set goes well beyond the normal BASIC language by adding many commands not found in the original. This expanded WT-BASIC language makes programming the WI-130 more flexible than was possible with the original BASIC language. It adheres closely to the QBASIC language included in current versions of MS-DOS. Syntax examples are included where necessary in the following pages.

There is a difference between WT-BASIC and QBASIC. In QBASIC a main program body runs and calls out subroutines and performs some task. In WT-BASIC, you design what are called Event Handlers. These are BASIC subroutines that activate only upon the occurrence of some event. What is an event? It can be any indicator or scale related activity such as

- pressing a soft key
- pressing a hard key
- on a time interval
- reaching a certain gross weight
- scale stability
- scale motion
- serial port data
- setpoint activation or deactivation
- input switch state change
- etc.

The idea is to handle the event then exit the subroutine so that other event handlers can be called. You can create your own names for subroutines that are not triggered by an event but can be called from other subroutines that are triggered (or handled) by an event.

The actual list is very long, but you should get the idea that you can tie a WT-BASIC program to many events. The program you write can cause an unlimited number of things to occur, whether for prompting for data, opening valves, printing tickets or sending data to a computer. It all depends on your imagination and your particular application.

See the included examples for an idea of where to start. For those new to programming or those accustomed to other languages, a book on QBASIC is a good place to begin.

## Event Handlers

**SYNTAX:** SUB *Eventname*
commands
.
.
END SUB

Anytime one of the following events is triggered in the instrument, program execution will be transferred to the corresponding sub procedure (if the subroutine procedure exists). If more than one event is queued up, they are executed in the order they were tripped.

*Think of a queue as a list of many items. (i.e. event names)*

**SYSTEM_STARTUP** If a SYSTEM_STARTUP procedure exists in the WT-BASIC program, it will always get executed at the time of startup.

**SETPT1_ACT**
•
•
**SETPT32_ACT** These events execute when the SetPoint turns on.

**SETPT1_DEACT**
•
•
**SETPT32_DEACT** These events execute when the SetPoint turns off.

**SYSTEM_TIMER** Called on interval determined by Settimer command. This may generate multiple queues in the event buffer.

**SYSTEM_ TIMER2** Called on interval determined by Settimer command. This will generate one queue (see note) in the event buffer.

**F1_KEY**
.
**F5_KEY** For soft keys. These events execute when the key is pressed on the WI-130 or an external keyboard

**F6_KEY**
.
.
**F10_KEY** Only on an external keyboard or in the simulator.

Defining one of these blocks the normal default operation from occuring when you press a key on the front panel of the WI-130. Use the **doxxx** commands to make the regular function work. See **doxxx**.

**ZERO_KEY** This event will be activated if the ZERO key is pressed on the WI-130.

**UNITS_KEY** This event will be activated if the UNITS key is pressed on the WI-130.

**TARE_KEY** This event will be activated if the TARE key is pressed on the WI-130.

**PRINT_KEY** This event will be activated if the PRINT key is pressed on the WI-130.

**SELECT_KEY** This event will be activated if the SELECT key is pressed on the WI-130.

| | |
|---|---|
| **ZERO_OPER** | This event will be activated if the DOZERO command is successfully processed. |
| **UNITS_OPER** | This event will be activated if the DOUNITS command is successfully processed. |
| **TARE_OPER** | This event will be activated if the DOTARE command is successfully processed. |
| **PRINT_OPER** | This event will be activated if the DOPRINT command is successfully processed. |
| **SELECT_OPER** | This event will be activated if the DOSELECT command is successfully processed. |
| **ACCUM_OPER** | This event will be activated if the DOACCUM command is successfully processed. |
| **ZERO_ABORT** | This event will be activated if the DOZERO command is not successfully processed. |
| **UNITS_ABORT** | This event will be activated if the DOUNITS command is not successfully processed. |
| **TARE_ABORT** | This event will be activated if the DOTARE command is not successfully processed. |
| **PRINT_ABORT** | This event will be activated if the DOPRINT command is not successfully processed. |
| **ACCUM_ABORT** | This event will be activated if the DOACCUM command is not successfully processed. |
| **COM1_MESSAGE** | This event is activated when a complete message has arrived in the COM1 serial port input buffer. A "Complete Message" is determined by the End of Message Character in the Serial Ports Configuration. |
| **COM2_MESSAGE** | Same as COM1_MESSAGE only for Port #2. |

## Variables

*See the list of variables under Terms in Print Format to see the usable system variables in print formats. All system variables can be used in the subroutines of your program.*

A variable is a word you create which represents values (numbers or letters) which continually change (vary) during the course of your running program. The variables can be numeric variables or string variables.

**Numeric Variable**

A numeric variable represents a group of numbers that mathematic functions may be performed on.

**String Variable**

A string variable may represent a group of letters, numbers or a combination of the two that math functions can not be performed on.

**System Variable**

These variables have already been predefined by Weigh-Tronix in the WI-130 WT-BASIC command set to make your programming easier. These system variables are double precision floating decimal point unless otherwise noted.

These system variables may be used in a print format or accessed in a program.

**SYNTAX:** *X*=GROSS
or
PRINT GROSS

*Expanded definitions for each of these system variables appears in Appendix 7.*

**COUNT**
**GROSS**
**NET**
**TARE***
**MAXPEAK***
**MINPEAK***
**ROC**
**MOTION**
**CZERO**
**OVERLD**
**UNDERLD**
**GROSSTOT***
**NETTOT***
**COUNTTOT***
**TRANSTOT***
**PCWT***
**RAWGROSS**
**CURSCALE***
**TIME$(n)**
**DATE$(n)**
**DIVISION**
**CAPACITY**
**DISPLAY**
**CURUNIT**
**CURUNIT$**

*Returns a negative one (-1) for a true condition and a zero (0) for a false condition.*

Refer to the *Print Format* section of this manual for more information on time and date strings.

This is the active displayed numeric value (i.e., gross, tare, net, count, etc.). See **doselect**.

* These can also be assigned to and from,
i.e. Tare = 10.0 (This number **must** be in calibration unit of measure)
x=tare (returns the tare in current unit of measure)

These may not be directly used with the input statement.

WRONG - Input "Enter tare", tare

RIGHT - x = tare
Input "Enter tare", x
tare = x

## Variable Commands

*MUSTDIM is very helpful when debugging code by detecting spelling errors of variable names.*

**MUSTDIM** Mandates that all variables used in program must be dimensioned before being used.

**Variable Types**

Variable types can be defined by DIM statements. Below are the different kinds of DIM commands.

*X is a variable name in these examples.*

*Numeric variables only.*

| | |
|---|---|
| **DIM** X# **or DIM** X | This uses the system default of 15 digit, double precision, floating point. (i.e., 6 would be 6.0000) |
| **DIM** X% | A regular integer with no decimal point |
| **DIM** X! | Single precision floating point |
| **DIM** X& | Long integer (no decimal point) |
| **DIM** X$ | 16 character string variable (default) |
| **DIM** X$ * 10 | 10 character string variable (64 characters max) |

## Logical Operators

*Logical operators return a true (-1) or false (0). Any non-zero value is true.*
*Example:*
*14 AND 2 results in a -1 or true.*
*14 AND 0 results in a 0 or false.*

**For an expression using a logical operator to be considered true, the following conditions must be met:**

AND - *both parts of the expression must be true*
OR - *either part of the expression must be true*
XOR - *both parts of the expression must be the opposite*
EQV - *both parts of the expression must be the same*
IMP - *first part true, second part false*
NOT - *inverse of expression or opposite condition*

## Arithmetic Operators

(In order of precedence)

**-** Negation
***** Multiplication
**/** Division
**+** Addition
**-** Subtraction

Operations within parenthesis are performed first. Inside the parenthesis, the usual order of precedence is maintained.

## Relational Operators

= Equal to
> Greater than
< Less than
<= Less than or equal to
>= Greater than or equal to
<> Not equal to
>< Not equal to

## Command Statements

*Any keyword with a printable message in quotation marks must have a space separating the keyword from the message.*

*EXAMPLE: Print "Hello"*
*Space*

**PRINT** To output data to the screen.
**SYNTAX:** PRINT "*THIS WILL GO TO THE DISPLAY*"
The area of the display dedicated to WT-BASIC messages is determined by the active display mode.

**PRINT #** To output data to a serial port.
**SYNTAX:** PRINT #1, "*THIS WILL GO OUT OF PORT #1*"
PRINT #2, "*THIS WILL GO OUT OF PORT #2*"

**FMTPRINT(n)** Calls a predefined print format (n = 1 to 16) that will be sent out a serial port, as specified in destination port settings.

**INPUT** To receive data input from the keypad or a remote keyboard.
**SYNTAX:** INPUT "Enter id", ID$
The specified prompt is shown on the display.

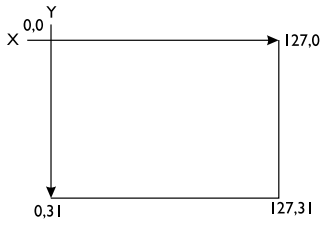
*These input options remain active until turned off by using INPUTOPT(0,0,0)*

**INPUTOPT** This stands for input options. This keyword will allow the following features in an input statement:

**SYNTAX:**
inputopt(noecho, timeout, noclear)

| | |
|---|---|
| noecho | If noecho is set to "0" then the data entered during an input statement will be visible.<br>If noecho is set to "1" or "-1" then the data entered during an input statement will appear as an asterisk "*". This will be useful for password protecting softkeys. |
| timeout | If timeout is set to "0", the input statement will wait for an **ENTER** or **ESC** key depression.<br>If timeout is set to a number, then the input statement will timeout after that amount of time if a key is not hit. The timeout feature is reset each time a key is hit. |
| noclear | If noclear is set to "0" the variable used for the input statement will be cleared upon data entry.<br>If noclear is set to "1" or "-1" the variable used for the input statement will NOT be cleared upon data entry, and the character/number entered will be concatenated to the existing data. |

*The WI-130 has a display height of 32 dots and a width of 128 dots. Coordinate 0,0 is the upper leftmost dot.*




**DOT(*X,Y*)** Draws a dot on the screen at coordinates *X,Y*.

**CIRCLE(*X,Y,R*)** Draws a circle on the screen starting at coordinates X,Y, using the value of R as the radius (in pixels).

**LINE(X1,Y1,X2,Y2)** Draws a line on the screen from coordinates X1, Y1 to X2, Y2.

## Program Loops

*All forms of **IF.. commands** are used to to make a decision regarding program flow based on the result of an expression.*

*Example of an expression:*
*(A>B)*

*Examples of statements:*
*PRINT "BIG NUMBER"*
*or*
*PRINT "LITTLE NUMBER"*
*or*
*C = (A*B)+ d*

**IF. .THEN (singular)**

**SYNTAX:**
IF expression THEN statement

**IF. . THEN . .ELSE (singular)**

**SYNTAX:**
IF *expression* THEN *statement* ELSE *statement*

**IF. .THEN (block)**

**SYNTAX:**
IF expression THEN
statement
statement
statement
} Block of statements
END IF

**IF. .THEN.. ELSE (block)**

**SYNTAX:**

```
IF expression THEN
        statement
        statement      } Block of statements
        statement
ELSE
        statement
        statement      } Block of statements
        statement
END IF
```

**END IF** Used to stop a block of statements.

**ELSEIF** This command allows absolute detection of a condition.

**SYNTAX:**

```
IF expression THEN
        statement
        ELSEIF expression THEN
                statement
        ELSEIF expression THEN
                statement
ELSE
        statement
END IF
```

*"Statement label" is any name or number used to identify the next "line of code" or statement to be executed.*

**GOTO** To branch out of the normal program sequence of instructions to a specified instruction identified by a "statement label".

**SYNTAX:** GOTO statement label

*FOR. . NEXT loops only count up, not down.*

**FOR. . NEXT** To execute a series of instructions a specified number of times in a loop. Must also use "NEXT" command.

**TO** Used in FOR. . NEXT commands

**EXIT FOR** Used to terminate a FOR. . NEXT loop upon a certain condition prior to the loop expiring.

**SYNTAX:**

```
FOR variable = X TO Y
        statement
IF expression THEN EXIT FOR
NEXT variable
```

**WHILE. . WEND** To execute a series of statements in a loop as long as a given condition is true.

**EXIT WHILE** Used to terminate a WHILE. . WEND loop upon a certain condition prior to the loop expiring.

**SYNTAX:**

```
WHILE  expression
       statement
IF. . expression THEN EXIT WHILE
       statement
WEND
```

## Subroutine Controls

It is usually better to use **CALL** instead of **GOSUB**

**GOSUB** To branch to a subroutine.

**RETURN** To return from a subroutine.

**SUB. .** Defines a subroutine procedure.

**END SUB** Ends a procedure.

**EXIT SUB** Used to terminate a subroutine upon a certain condition prior to the subroutine expiring.

**SYNTAX:**

```
SUB PrintRoutine
       statement(s)
IF expression THEN EXIT SUB
       statement(s)
END SUB
```

**CALL** Transfers control to a second subroutine procedure. After the second subroutine is done, control automatically transfers back to the next line after the CALL command in the previous subroutine.

**SYNTAX:** CALL *PrintRoutine*

**END** Terminates the program.

**CLS** Clears the display of any WT-BASIC text. Not necessary before a DISPMODE command.

**REFRESH** Updates the displayed data while in a WT-BASIC program loop.

**DISPMODE** Sets the active display mode. The display modes may be viewed in accompanying software or in *Appendix 1* of this manual. There are more than 30 displays to choose from. **(n)** is the number representing the display mode.

**SYNTAX:** DISPMODE(n) or DISPMODE=n

*Use REM or ' for describing operations, setup of the instrument, temporarily disabling a command, etc.*

**REM** Allows for comments to be placed in the body of a program.

**SYNTAX:**

```
REM   THIS LINE IS A COMMENT IN THIS PROGRAM
```

or

```
      THIS LINE WITHOUT A REM WILL BE INTERPRETED AS A COMMAND
```

**'** (apostrophe) This also works in place of REM.

*Pay close attention to the location of spaces used with keywords that have a message inside of quotation marks.*

**BEEP** Sounds the beeper in the instrument.

**SLEEP(n)** Causes the program to halt processing for **n** number of seconds.
**SYNTAX:** SLEEP(10) (Causes processing to halt for 10 seconds)

**KEY 1,"keyname"** Names a softkey on the screen.
**KEY 2,"keyname"** **SYNTAX:** KEY 1, "START"
**KEY 3,"keyname"** KEY 2, "STOP"
**KEY 4,"keyname"** KEY 3, "SET UP"
**KEY 5,"keyname"** KEY 4, "MORE"

**ACTVALUE** As a command ACTVALUE sets the displayed numeric data on the indicator to a different active value.
**SYNTAX:** ACTVALUE(n) n=0 to 13

As a system variable ACTVALUE returns number (n) which represents the currently displayed active value per table below.
**SYNTAX:** n=ACTVALUE n=0 to 13

| | | | |
|---|---|---|---|
| 0 = Gross | 4 = Max | 8 = Count Total | 12=Piece Weight |
| 1 = Net | 5 = Rate of Change | 9 = # Total Trans. | 13=ADC |
| 2 = Tare | 6 = Gross Total | 10=Count | |
| 3 = Min | 7 = Net Total | 11=Variable | |

**GETCONV** Gets the conversion factor of the currently displayed unit measure.
**SYNTAX:** X = GETCONV

**GRBASIS(n,[varname$])** This command sets the basis for the bar graph representation on the display. (n = 0 to 13) Varname$ only applies when the graph basis is number 11, variable.

| | | | |
|---|---|---|---|
| 0 = Gross | 4 = Max | 8 = Count Total | 12=Piece Weight |
| 1 = Net | 5 = Rate of Change | 9 = # Total Trans. | 13=ADC |
| 2 = Tare | 6 = Gross Total | 10=Count | |
| 3 = Min | 7 = Net Total | 11=Variable | |

**SETBAR(*Minvalue,MaxValue)*** Sets the values for the bar graph. The (MIN) is the minimum value where the graph starts. The (MAX) is the maximum value where the graph stops.
**SYNTAX:** SETBAR(MIN,MAX)

**SETCHECK** Sets the values for the checkweigher graph. The (MIN) is the minimum value where the graph starts. The (MAX) is the maximum value where the graph stops. (UNDER) is the lowest acceptable value for the checkweigher graphic display. (OVER) is the highest acceptable value for the checkweigher graphic display.
**SYNTAX:** SETCHECK(MIN,UNDER,OVER,MAX)

**CALCSTAT** Calcstat is a function that calculates a number of statistical values based on a series of numbers.

**SYNTAX:** CALCSTAT(START,COUNT,DEST)

Parameters

***Start***- Start is the beginning memory location of a series of values to calculate the statistical analysis on.

***Count***- Count is the number of sequential memory locations from the previous stated START parameter that have values stored to calculate the statistical analysis on.

***Destination***- Destination is the starting memory location to record the results of the statistical analysis. There must be eight sequential memory locations available for the results of CALCSTAT to be recorded in. The results will be in the following order starting at the *Destination* memory location:

Average
Minimum
Maximum
Average Deviation
Standard Deviation
Standard Variance
Skewness
Curtosis

*There are two timer events available for use in the WI-130. They are SYSTEM_TIMER and SYSTEM_TIMER2. Use SETTIMER to regulate their frequency of occurrence.*

**SETTIMER** This command causes the SYSTEM_TIMER event to occur.

**SYNTAX:** SETTIMER(timernum,seconds)

Example: SETTIMER(1,0.5) This will activate the event system_timer every ½ second. With zero seconds, this disables the timer event specified. This is the default state of the Timer Events upon power up.

SETIMER(1,0) This shuts the timer off.

**TIMER** This command returns the number of seconds (in hundredths) from midnight until the present time of day.

**SYNTAX:** x = TIMER

**UNIXTIME** This command returns the number of seconds since 1970.

**SYNTAX:** x = UNIXTIME

**ANBASIS*(n[,varname$])-*** This command sets the basis for the analog output PC board. ***(n)*** can be equal to any of the following values: (varname$ only applies when the analog basis is #11 - variable)

| | | | |
|---|---|---|---|
| 0 = Gross | 4 = Max | 8 = Count Total | 12 = Piece Weight |
| 1 = Net | 5 = Rate of Change | 9 = # Total Trans. | 13 = ADC |
| 2 = Tare | 6 = Gross Total | 10 = Count | |
| 3 = Min | 7 = Net Total | 11 = Variable | |

**SETANLOG*(MIN, MAX[,SPAN ADJUST, OFFSET ADJUST])*** - This command sets the minimum and maximum analog output limits. The optional parameters Span Adjust and Offset Adjust are used a preliminary setting for the analog output. These parameters are 0 by default and have a typical value between 5000 and -5000.

*Fine tuning of the actual analog output pc board signal should be adjusted in the field, or real world, from the front panel of the WI-130.*

**SHOWVAR** This lets you display a numeric value on the WI-130 where the weight value usually appears along with custom labels or legends.

**SYNTAX:**

SHOWVAR (varname$,legend1$,legend2$,precision)

**See Appendix 3 for an example of this command.**

**PCWTZERO** This command zeros the scale according to the Count Stability Values as defined in the Configuration forms. This is the recommended method of zeroing the scale before a sampling process is started. If a key is hit before counting stability is reached, the zeroing process is aborted.

**PCWTSAMP** This function determines a pieceweight based on the weight currently on the scale divided by (n) as the number of pieces in the sample size. The sample will be taken upon reaching Count Stability as defined in the Configuration forms. If a key is hit before counting stability is reached, the sampling process is aborted.

**SYNTAX:** PCWTSAMP(n)

## Do... Commands

The following DO..... commands are used when the corresponding WI-130 front panel key control has been taken over through the use of a WT-BASIC event. By issuing the DO..... command, the system initiates the appropriate operation, such as zeroing the scale.

**DOZERO** This command zeroes the scale if no motion is detected.

**DOUNITS** This command switches the display to the next valid unit of measure.

**DOTARE** This command tares the scale if no motion is detected.

**DOPRINT** This command prints the selected default print format as previously configured if no motion is detected.

**DOACCUM** This command requests an Accumulate event. The weight must be stable within the motion window parameters for the accumulation to occur.

**DOSELECT** This command switches the display to the next valid active displayable value (gross, net, tare . . .).

## Command Functions

**RESETMIN** RESETMIN command resets the value of the system variable MINIMUM to the current value on the scale platform.

**RESETMAX** RESETMAX command resets the value of the system variable MAXIMUM to the current value on the scale platform..

**INKEY$** Returns one character read from the remote keyboard or from the WI-130 front panel keypad as a string. Returns an empty string " ", if no character is available.

**SYNTAX:** X$=INKEY$

*INKEY$ is typically used in a program loop for detecting a specific key depression.*

*Also used to clear the value of "LASTKEY" or "KEYHIT"*

**KEYHIT** This command is used in program loops to detect a key hit on the front panel of the WI-130. This command only recognizes numeric keys on the front panel of the WI-130.

**SYNTAX:** x = KEYHIT

The variable **x** is returned as a true, negative one (-1), or a false, zero (0) depending on whether a key is hit or not.

*The average weight used in these calculations is based upon the system variable GROSS weight and is affected by filtering settings from the configuration menu.*

**LASTKEY**

| *Last Key* | *Values Returned* |
|---|---|
| *F1* | *15104* |
| *F2* | *15360* |
| *F3* | *15616* |
| *F4* | *15872* |
| *F5* | *16128* |
| *F6* | *16384* |
| *F7* | *16640* |
| *F8* | *16896* |
| *F9* | *17152* |
| *F10* | *17408* |
| *SELECT* | *7936* |
| *UNITS* | *5632* |
| *PRINT* | *6400* |
| *TARE* | *5120* |
| *ZERO* | *11264* |
| *ESCAPE* | *27* |
| *ENTER* | *13* |
| *CLEAR* | *11776* |
| *.* | *46* |
| *0* | *48* |
| *1* | *49* |
| *2* | *50* |
| *3* | *51* |
| *4* | *52* |
| *5* | *53* |
| *6* | *54* |
| *7* | *55* |
| *8* | *56* |
| *9* | *57* |

**LASTKEY** LASTKEY returns the ASCII value of the last key pressed on the keyboard as a numeric value. See table (LASTKEY) in left column.

**AVGSTART** Starts averaging for inmotion systems.

**AVGSTOP** This returns the average weight (**X**) on the scale since the last AVGSTART command or for the last 4 seconds, whichever is shorter.

**SYNTAX:** **X**=AVGSTOP

**ABS(n)** Returns the absolute value of the expression.

**ATN(n)** Returns the arctangent of the expression.

**SQR(n)** Returns the square root of the expression.

**INT(n)** Returns the integer value of the expression.

**FIX(n)** Truncates the expression to a whole number.

**CINT(n)** Rounds numbers with fractional portions to the next whole number or integer.

**SGN(n)** Returns the sign of the expression. 1, 0, or -1

**SIN(n)** Calculates the trigonometric sine of the expression, in radians.

**COS(n)** Calculates the cosine of the range of the expression.

**TAN(n)** Calculates the trigonometric tangent of the expression, in radians.

**LOG(n)** Calculates the natural logarithm of the expression.

**EXP(n)** Returns *e* to the power of *x*.

**ROUND(x,y)** Rounds *X* to the nearest *y*.

**SYNTAX:** Z=**Round(X,Y)**
With X=10.04 and Y=0.05
The value of Z is now set to 10.05

**STR$(n)** Converts the value of an expression to a string.

**SYNTAX:** Z$=**STR$(X)**
With X= 5000
The string Z$ is now set to “5000”
not the numeric value of 5000

**MID$(C$,n,[x])** This command copies part of an existing string (**C$**) and makes a new string. The new string starts at the **n**th character of the orignal string (**C$**) and continues for **x** number of characters.

**SYNTAX:** Z$=**MID$(C$,n,[x])** ( [ ]=optional )
With **C$** =”The WI-130 Indicator"
and **n** = 5 and x = 6
MID$ returns the new string Z$ or “WI-130”.

**LEFT$(C$,x)** This command copies the leftmost characters of an existing string (**C$**) and makes a new string. The new string starts at the leftmost character of the orignal string and continues for **x** number of characters to the right.

**SYNTAX:** Z$=**LEFT$(C$,x)**
With **C$** =”The WI-130 Indicator"
and **x** = 3
LEFT$ returns the new string Z$ or “The”.

**RIGHT$(C$,n)** This command copies the rightmost characters of an existing string (**C$**) and makes a new string. The new string starts at the rightmost character of the orignal string and continues for **x** number of characters to the left.

**SYNTAX:** Z$=**RIGHT$(C$,x)**
With **C$** ="The WI-130 Indicator"
and **x** = 9
RIGHT$ returns the new string Z$ or "Indicator".

**CHR$(n)** Converts an ASCII code to its equivalent character.

**SYNTAX:** Z$ = CHR$(n)

*Converts the numeric value of X to a string as defined by width and precision parameters.*

**FORMAT$** Format$(x,width.prec) Returns x as string with a width and precision the same as in the print formats. See *Format* section in this manual for more information on width and precision.

**SYNTAX:** Z$ = FORMAT$(x,w.p)

**LCASE$(C$)** Converts the *C$* to lower case characters.

**SYNTAX:** Z$ = LCASE$(C$)

**UCASE$(C$)** Converts the *C$* to upper case characters.

**SYNTAX:** Z$ =UCASE$(C$)

**HEX$(n)** Creates a new string that represents the hexadecimal value of the numeric decimal value.

**SYNTAX:** Z$ = HEX$(n)

**LTRIM$(C$)** Strips *C$* of any leading spaces.

**SYNTAX:** Z$ = LTRIM$(C$)

**RTRIM$(C$)** Strips *C$* of any trailing spaces.

**SYNTAX:** Z$ = RTRIM$(C$)

*Julian date is the number of days since some day in the ancient past. One reference point is Julian day 2,440,000 is May 23, 1968. This allows any date to be represented or stored as a value rather than a string.*

*Julian dates start at noon.*

*All 4 digits of the year must be used to ensure correct J values beyond the year 2000.*

**JULDATE$** x$=JULDATE$(y) Where **y** is a julian date, this will return the date as the string **x$**.

**JULIAN** y=JULIAN(x$) Where **x$** is the date to convert, this will return the value **y** as the julian date.

**VAL(C$)** Returns the numerical value of string *C$*.

**SYNTAX:** Y = VAL(C$)

**LEN(C$)** Returns the number of characters in *C$*.

**SYNTAX:** Y = LEN(C$)

**ASC(C$)** Returns the numeric value that is the ASCII code for the first character of *C$*.

**SYNTAX:** Y = ASC(C$)

**INSTR(C$,D$)** Searches for the first occurrence of string *D$* in *C$*, and returns the position.

**SYNTAX:** Z=INSTR(C$,D$)
C$="WI-130 INDICATOR"
D$="N"
Z=**INSTR(C$,D$)**
The value of Z is now set to 9

*See the Enhanced Ask feature in the* New Keywords *section*

**ASK(MSG$)** Ask is a function that allows you to prompt the operator for a yes/no response. The ASK(MSG$) function returns a negative one (-1) for a YES response and a zero (0) for a NO response.

**SYNTAX:** ASK(MSG$)

An example would be as follows:

```
IF ASK("New Truck, Add?") THEN                    'YES
  PRINT "STORED NEW TARE"
ELSE                                              'NO
  PRINT "CANCELLED STORAGE OF NEW TARE"
END IF
```

**GETPORT** Returns the numeric value (x) that represents the current protocol of the port selected from the tables below.

**SYNTAX:** x=getport(port,y)

Pick **y** from the table below (1 to 10)

| | |
|---|---|
| 1 | baud rate |
| 2 | parity |
| 3 | databits |
| 4 | handshake |
| 5 | mode |
| 6 | eom character |
| 7 | RTS only used in setport |
| 8 | CTS only used in getport |
| 9 | transmit buffer free size (max 512) |
| 10 | receive buffer count (512 is full) |
| 11 | number of messages in the receive buffer |

| Baud | Parity | Databits | Handshake | Mode |
|---|---|---|---|---|
| 1=300 | 0=none | 7=7 | 0=none | 0=Basic Control |
| 2=1200 | 1=odd | 8=8 | 1=CTS | 1=Keyboard |
| 3=2400 | 2=even | | 2=XonXoff | 2=Disable |
| 4=4800 | 3=set | | 3=both | 3=Multidrop |
| 5=9600 | 4=clear | | | |
| 6=19200 | | | | |
| 7=38400 | | | | |
| 8=56700 | | | | |

*EOM - End of Message character as set in the configuration menu.*

| EOM | CTS |
|---|---|
| See ASCII table in ***Appendix 5*** | 0=off<br>1=on |

*Input buffer size is 512 characters. You may define the length of your string variable in a DIM statement. If the length is not defined, it defaults to 16 characters. Storage of strings in memory is limited to 16 characters per location.* ***See Appendix 3 for a sample routine called GETCOM$.***

**SETPORT** SETPORT(port,y,z)

Pick **y** from the table below (1 to 7)
Pick **z** from the corresponding list below.

| | |
|---|---|
| 1 | baud rate |
| 2 | parity |
| 3 | databits |
| 4 | handshake |
| 5 | mode |
| 6 | eom character |
| 7 | RTS only used in setport |

| Baud | Parity | Databits | Handshake | Mode |
|---|---|---|---|---|
| 1=300 | 0=none | 7=7 | 0=none | 0=Basic Control |
| 2=1200 | 1=odd | 8=8 | 1=CTS | 1=Keyboard |
| 3=2400 | 2=even | | 2=XonXoff | 2=Disable |
| 4=4800 | 3=set | | 3=both | 3=Multidrop |
| 5=9600 | 4=clear | | | |
| 6=19200 | | | | |
| 7=38400 | | | | |
| 8=56700 | | | | |

| EOM | RTS |
|---|---|
| See ASCII table in ***Appendix 5*** | 0=off<br>1=on |

**GETCOM$(*portnum,<maxchars>)* -** GETCOM$() reads one message from the specified Com-Port. The function reads from the first character in the input buffer to one of the following:

**a) End of Message Character** (As specified in the Serial Port configuration Menu in the SimPoser)

**b) No more characters in the input buffer.**

**c) Maximum characters reached** (As specified in the optional GETCOM$ parameter)

**d) Maximum string size reached**

**SYNTAX**: X$=GETCOM$(*portnum,<maxchars>)*
*portnum represents serial port* 1 or 2
*maxchars (*optional) represents the maximum number of characters to read and assign to the string (X$).

*WARNING: Using CONTOUT may cause system performance problems. i.e.-execution speed may be slowed.*

**CONTOUT** This command will activate the continuous output of the chosen format. Destination port is selected from the format menu.

**SYNTAX:** CONTOUT(FMT,RATE)
FMT = format print number
RATE = update rate

Example: CONTOUT(1,2) This will output format 1 twice per second.

**SETPTON(*n*)** This command will activate SETPOINT *n.*

**SETPTOFF(*n*)** This command will deactivate SETPOINT *n.*

**ISSETPT** Tells you whether the setpoint has been activated.

**SYNTAX:** Z = ISSETPT(x) x = 1 to 32

The variable **Z** is returned as a true, negative one (-1), or a false, zero (0) depending on the condition of the setpoint.

**CURUNIT** CURUNIT can be used as a command or a system variable. CURUNIT can be used to return the current selected system UNIT. The CURUNIT function will return a 0 for lbs, 1 for kgs, 2 for grams, 3 for oz, 4 for lb oz, 5 for custom 1 and 6 for custom 2.

**SPACE$(*n*)** SPACE$(*n*) will return a string with the specified number of spaces.

**PLEN(C$)** **SYNTAX:** X = PLEN(C$)
PLEN returns a numeric value (X) which represents the width of a string (C$) in dots for proportional fonts.

**CURUNIT$** This system variable returns the current unit of measure string 'lb' or 'kg', etc. See **Format** menu for more information.

**UNIT$** The command UNIT$(0) returns 'lb' and UNIT$(1) returns 'kg', etc.

**RANDOM** Generates a random decimal number between 0 and **(n)**.
**SYNTAX:** x = RANDOM(n)
The variable **x** represents the generated decimal number.
The variable **n** represents the largest decimal number you want to use.

**EVENTNUM** This command returns the variable (**x**) which identifies the event name. The variable (**x**) can then be used in the EVENTRDY and EVENTCLR commands.
**SYNTAX**: x = EVENTNUM(NAME$)
The variable (**x**) is the EVENTNUM.
**Name$** is the event name.

**EVENTRDY** This command detects the existence of an event name in the event queue.
**SYNTAX**: Z = EVENTRDY(x)
The variable **Z** returns true, negative one (-1), or false, zero (0).
The variable **x** represents the value of the event name from the command EVENTNUM.

**EVENTCLR** This clears the oldest instance of EVENTNUM from the event queue.
SYNTAX: EVENTCLR(x)
The variable **x** represents the value of the event name from the command EVENTNUM.

## Permanent Memory Storage

*There are three types of memory:*
- *Standard*
- *Expanded*
- *Memory Card*

*Standard memory has locations 0-1023 for numeric storage and 0-511 for string storage.*

*Expanded memory has locations 1024-8191 for numeric storage and 512-4095 for string storage.*

*The memory card has locations 8192-73727 for numeric storage and 4096-36863 for string storage.*

The WI-130 has 1024 storage locations and 512 sixteen character string storage locations available in permanent memory. These memory locations are reusable and will remain through a power loss. Expanded memory options increase these limits. See note at left. Four commands are available for accessing these memory locations:

**STORE**(*n,*3456.89*)* Stores 3456.89 in the *n* numeric memory location.

**STORE$**(*m,*"Hello"*)* Stores "Hello" in the *m* string memory location, 16 characters maximum per location.

x=**RECALL**(*n)* Recalls the *n* numeric memory location.

C$=**RECALL$**(*m)* Recalls the *m* string memory location.

**FIND**(VAR,x,y) Searches numeric memory slots between x and y for a variable and returns the memory location. Returns a -1 if not found. Returns a -2 if not valid.

**SYNTAX:** Z = FIND (VAR,x,y)

*If you do not have the memory installed, the location returns a zero.*

**FINDSTR**(C$,x,y) Searches numeric memory slots between x and y for a string and returns the memory location. Returns a -1 if not found. Returns a -2 if not valid.

**SYNTAX:** Z = FINDSTR (C$,x,y)

*See* Appendix 3 *for a sample subroutine.*

**SHOWSETP** Shows setpoints in the upper right of the display. Use only for troubleshooting.

**CALDATA(x)** Returns calibration information for ISO purposes.
x=1 Span Factor
x=2 Zero Count

**TONE(x)** Turns the tone on.
x = 1/Frequency x Offset

**TONEOFF** Turns the tone off.

**VERSION$(x)** Returns the version of the firmware and W-T Basic program.
x=1 Indicator type, SimPoser version, time, date of download, License #, customer name
x=2 Firmware date and time
x=3 Part Number
x=4 Revision Letter

*These appear only in firmware dated after 1-11-96 or after Rev. A.*

## New System Events (Added in Rev. E of the manual)

**SYSTEM_SETUP**

This event is queued by first issuing the setpwd keyword. The programmer should then hold in the ESCAPE key for 5 seconds and enter the password defined by the setpwd command. If the passwords match, SUB SYSTEM_SETUP will be queued. If the passwords don't match, the prompt will be aborted.
Example:

```
SUB SYSTEM_STARTUP
word$="130"
setpwd(1,word$)
END SUB

SUB SYSTEM_SETUP
dispmode =10
Cls
Print "HELLO WORLD!!!"
END SUB
```

**ENTER_KEY**

This event is queued when the ENTER key is pressed. This event is not queued from the "input" or "ask" keywords.
Example:

```
SUB SYSTEM_STARTUP
dispmode=10
END SUB

SUB ENTER_KEY
print "HELLO WORLD"
END SUB
```

**CLEAR_KEY**

This event is queued when the CLEAR key is pressed. This event is not queued from the "input" keyword.

```
SUB SYSTEM_STARTUP
dispmode=10
END SUB

SUB CLEAR_KEY
print "HELLO WORLD"
END SUB
```

**ESC_KEY**

This event is queued when the ESC key is pressed. This event is not queued from the "input" or "ask" keywords.
Example:

```
SUB SYSTEM_STARTUP
dispmode=10
END SUB

SUB ESC_KEY
print "HELLO WORLD"
END SUB
```

*Keyboard Tare under CONFIG must be disabled.*

**NUMERIC_KEY & ENTRY_KEY**

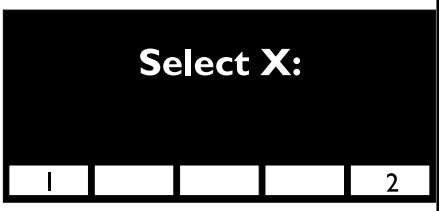
This event is queued when a key is pressed on the front panel numeric keypad. The keypress is then ported directly into the keyboard buffer, which allows for Reverse Pollish Notation input applications. SUB ENTRY_KEY works the same as NUMERIC_KEY, but is used for the remote keyboards.

Example:

```
SUB SYSTEM_STARTUP
dispmode=17
call Level0
END SUB

SUB F1_KEY
if level=10 then call InPN
END SUB

SUB NUMERIC_KEY
call Level0
x$=x$+inkey$
cls
print x$
END SUB

SUB ENTRY_KEY
call Level0
x$=x$+inkey$
cls
print x$
END SUB

SUB LEVEL0
key 1,"P/N"
key 2,"EDIT"
key 3,"SAMPLE"
key 4,"LOT#"
key 5,"REPORT1"
level=10
END SUB

'*** Input Routines
enter a part number
SUB INPN
cls
print "P/N: ",x$
pn$=x$
x$=""
call Level0
END SUB
```

**SYSTEM_ERROR** Event Is queued when an error occurs in your program. Use the key word **ERR** to return the number of the error.

## New or Enhanced Key Words (Added in Rev. E of the manual)




**ASK** The ask command will now accept 2 new optional arguments which allow the user to change the legends of the F1 key and the F5 key. Example:

```
SUB SYSTEM_STARTUP
dispmode=10
if ask("Select X:","1","2") then x=2 else x=1
cls
print "X=",x
END SUB
```

**CAPTURE** The capture command allows storage of weight readings into RAM for analysis at varying rates. There are multiple commands built into the capture keyword as outlined below.

0. Pauses the capture.
1. Starts the data capture.
2. Configures the Data capture.
3. Returns the number of data points the capture command has stored.

**SYNTAX:**

capture(2,xstart,xdatapt,xrate,xch,xs) OR
capture(n)

The following application outlines how to use the capture command with a single scale.

```
SUB SYSTEM_STARTUP
dispmode=17
key 1,"START"
key 2,"SETUP"
key 3,"OFF"
key 4,"STOP"
key 5,"RE-STRT"
xstart=0          'default starting memory location
xdatapt=1600      'take 1600 data points
xrate=60          'take samples at 60Hz
xch=1             'take readings from scale#1
xs=32             'setpoint 32 may control the activation
                  or 'deactivation of the capture feature,
                  but isn't required
END SUB

'start the data capture
SUB F1_KEY
capture(2,xstart,xdatapt,xrate,xch,xs)
setpton(32)
```

```
capture(1)
settimer(2,1)
END SUB

‘change the default capture settings
SUB F2_KEY
input “Start:”,xstart
if lastkey=27 then exit sub
input “DataPT:”,xdatapt
if lastkey=27 then exit sub
input “RATE: “,xrate
if lastkey=27 then exit sub
input “Channel:”,xch
if lastkey=27 then exit sub
input “SETPT:”,xs
if lastkey=27 then exit sub
capture(2,xstart,xdatapt,xrate,xch,xs)
‘capture(2,0,1600,60,1,32)
END SUB

‘turn the data capture setpoint off, essentially you paused
the data ‘capture
SUB F3_KEY
setptoff(32)
settimer(2,0)
END SUB

‘turn the data capture off this resets the data capture
SUB F4_KEY
setptoff(32)
capture(0)
settimer(2,0)
END SUB

‘clear ram to start over again
SUB SELECT_KEY
store(0,0,8000)  ‘store 8000 zeros starting at address 0
END SUB

‘show the data capture status to the display
sub system_TIMER2
capture(3,8100)  ‘tell capture where to store the index
print RECALL(8100)  ‘get the index
end sub

‘restart the data capture again
sub F5_key
setpton(32)   ‘re-enable the capture
settimer(2,1)
end sub
```

**CHECKSUM**

**SYNTAX:** x=checksum(x$,n)

The checksum keyword has been enhanced to include more that 1 algorithm. The following methods are now available:

1. 32 bit sum
2. 8 bit xor
3. CCITT 16-bit CRC
4. XMODEM CRC

Example:

```
SUB F1_KEY
Dispmode=10
X$="123456abc"
X=checksum(x$,3)
Cls
Print x$,x
END SU
```

**CHR$**

**SYNTAX:** Z$=CHR$(n)

CHR$ has been enhanced to accept any number from 0 – 255.

**ASC**

**SYNTAX:** Y=ASC(C$)

ASC has been enhanced to return any ASCII character from (0-255)

**FIND**

**SYNTAX:** LOC=FIND(Value,Start,Stop,[opt])

The FIND keyword has a new optional argument. The new selection tells the find keyword what type of find to perform. The following list outlines the new selections. Returns the value -2 if not found and -1if not valid.

OPT
0. Equal to (default)
1. Less than
2. Less than or equal to
3. Greater than
4. Greater than or equal to
5. Maximum
6. Minimum
7. Pass Through. Returns the first location which passes through a given data point.

Example:

```
SUB SYSTEM_STARTUP
dispmode=16
for i=1 to 200
 store(i,i)
next
store(201,23) 'for pass through test
END SUB
```

```
'equal to
SUB F1_KEY
print find(23,1,200,0)
print "SHOULD BE 23"
END SUB

'less than
SUB F2_KEY
print find(23,1,200,1)
print "SHOULD BE 1"
END SUB

'less than or equal to
SUB F3_KEY
print find(23,1,200,2)
print "SHOULD BE 1"
END SUB

'greater than
SUB F4_KEY
print find(23,1,200,3)
print "SHOULD BE 24"
END SUB

'greater than or equal to
SUB F5_KEY
print find(23,1,200,4)
print "SHOULD BE 23"
END SUB

'Max
SUB SELECT_KEY
print find(23,1,200,5)
print "SHOULD BE 200"
END SUB

'Min
SUB UNITS_KEY
print find(23,1,200,6)
print "SHOULD BE 1"
END SUB

'Pass through
SUB PRINT_KEY
print find(23,1,201,7)
print "SHOULD BE 24"
END SUB

'Selection not valid, should return -2
SUB TARE_KEY
print find(23,1,200,8)
print "SHOULD BE -2 SELECTION NOT VALID"
END SUB

'value not found, should return -1
SUB ZERO_KEY
print find(203,1,200)
print "SHOULD BE -1, Target Not Found"
END SUB
```

**FINDSTR**

**SYNTAX:** LOC=FINDSTR(VALUE$,START,STOP,[OPT])
The findstr keyword has two new optional arguments. The first selection is a case sensitive switch. If the "nonsensitive" switch is present the “type” argument **MUST** follow. The following list outlines the new selections. Returns the value -2 if not found and -1if not valid.

OPT

0. Equal to (default)
1. Less than
2. Less than or equal to
3. Greater than
4. Greater than or equal to
5. Maximum
6. Minimum
7. Pass Through.
8. Partial String Find. The first memory location which contains the search string is returned.
9. Inverted Partial String Find. The first memory location which contains a part of the search string is returned.

```
SUB SYSTEM_STARTUP
dispmode=16
store$(0,"123ABC")
store$(1,"123abc")
store$(2,"444rrr333eee")
store$(3,"444RRR333EEE")
END SUB

SUB F1_KEY
print findstr("123ABC",0,10)
END SUB

SUB F2_KEY
print findstr("123abc",0,10)
END SUB

SUB F3_KEY
print findstr("444rrr333eee",0,10)
END SUB

SUB F4_KEY
print findstr("444RRR333EEE",0,10)
END SUB

SUB F5_KEY
print findstr("123ABC",0,10,1,0)
END SUB

SUB F6_KEY
print findstr("123abc",0,10,1,0)
END SUB
```

```
SUB F7_KEY
print findstr("rrr",0,10,1,8)
END SUB

SUB F8_KEY
print findstr("EEE",0,10,1,8)
END SUB

SUB F9_KEY
print findstr("RRR",0,10,0,8)
END SUB

SUB F10_KEY
print findstr("eee",0,10,0,8)
END SUB
```

**INMOTION**

**SYNTAX:**

INMOTION(startIO,stopIO,startLocation,quantity,minVal,maxVal)

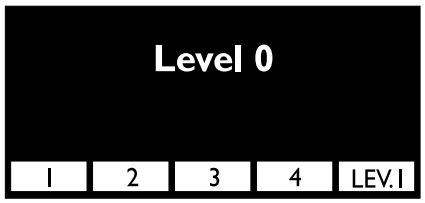
The INMOTION keyword will react at the A/D conversion rate of 60Hz. The entrance photoeye will be detected within 1/60$^{th}$ of a second, the averaging will start, the exit photoeye will be detected within 1/60$^{th}$ of a second, and the averaging will stop. The minimum weigh time has been decreased to 17milliseconds. This would return 1 A/D conversion for the weight reading. Over, Under, Accept, and Reject setpoints have been included in this keyword also.

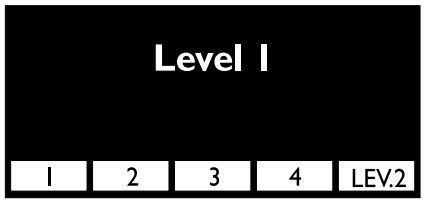
Example:

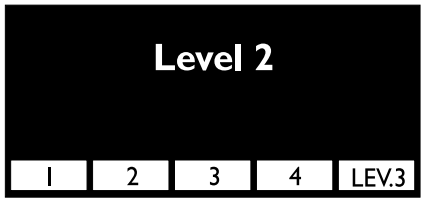
```
'Note: Accept, Reject, Over, and Under setpoints MUST be
configured for 'BASIC control Activate and Deactivate.

'inmotion(startIO,stopIO,startLocation,quantity,minVal,maxVal)

'startIO - IO location for the input to start the weighment.
Positive
'value indicates start on activate, negative indicates start on
'deactivate.

'stopIO - IO location for the input to stop the weighment.
Positive
'value indicates stop on activate, negative indicates stop on
deactivate.

'startLocation - Beginning Store/Recall location to store
values from in
'motion weighing.
```

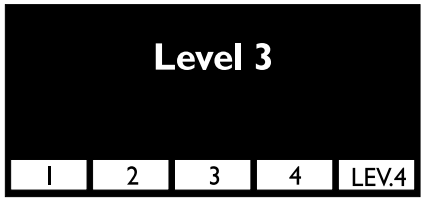
```
'quantity - Number of values to store beyond startLocation
before
'wrapping around back to startLocation.

'startLocation + quantity - New insertion pointer for store/
recalls

'minVal - Minimum (inclusive) acceptable gross weight.

'maxVal - Maximum (inclusive) acceptable gross weight.
'stopIO+1 = Accept Setpoint
'stopIO+2 = Reject Setpoint
'stopIO+3 = Over Setpoint
'stopIO+4 = Under Setpoint

'The Store/Recall value at recall(startLocation+quantity)
holds the 'value

SUB SYSTEM_STARTUP
dispmode=16
store(0,0,1023)
minval=0
maxval=50
inmotion(3,-4,0,1000,minval,maxval)
END SUB

SUB SETPT4_DEACT
wt=RECALL(x)
fmtprint(1)
x=x+1
if x>1000 then x=0
END SUB

SUB F1_KEY
inmotion(3,-4,0,1000,minval,maxval)
END SUB

SUB F2_KEY
input "Enter Min: ",minval
END SUB

SUB F3_KEY
input "Enter Max: ",maxval
END SUB
```

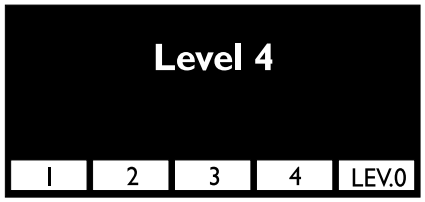
**Bitmaps showing softkey labels:**




**MENU**

**SYNTAX:**
MENU"key1","routine1", . . . . "key5","routine5"

This new keyword allows the user to create a menu system without creating it with a series of if/then statements. The keyword requires the softkey legend and the subroutine menu event which is queued for execution upon the keypress. Menu may be passed upto 10 softkey labels and sub routines. If a function is not defined for a given softkey, the default event handler, for that softkey will be queued, if it exists.

Example:

```
SUB SYSTEM_STARTUP
dispmode=17
Call Level0
END SUB

SUB LEVEL0
cls
print “level=0”
menu “1”,”R1”,”2”,”R2”,”3”,”R3”,”4”,”R4”,”Lev.1”,”LEVEL10"
END SUB

SUB LEVEL10
cls
print “level=10”
menu“5”,”R5”,”6”,”R6”,”7”,”R7”,”8”,”R8”,”Lev.2”,”LEVEL100"
END SUB

SUB LEVEL100
cls
print “level=100”
menu“1”,”R1”,”2”,”R2”,”3”,”R3”,”4”,”R4”,”Lev.3”,”LEVEL1000"
END SUB

SUB LEVEL1000
cls
print “level=1000”
menu“1”,”R1”,”2”,”R2”,”3”,”R3”,”4”,”R4”,”Lev.4”,”LEVEL10000"
END SUB

SUB LEVEL10000
cls
print “level=10000”
menu “1”,”R1”,”2”,”R2”,”3”,”R3”,”4”,”R4”,”Lev.0”,”LEVEL0"
END SUB

SUB R1
cls
r=1
```

```
print “Routine”;r;” HERE”
END SUB
SUB R2
cls
r=2
print “Routine”;r;” HERE”
END SUB
SUB R3
cls
r=3
print “Routine”;r;” HERE”
END SUB
SUB R4
cls
r=4
print “Routine”;r;” HERE”
END SUB
SUB R5
cls
r=5
print “Routine”;r;” HERE”
END SUB
SUB R6
cls
r=6
print “Routine”;r;” HERE”
END SUB
SUB R7
cls
r=7
print “Routine”;r;” HERE”
END SUB
SUB R8
cls
r=8
print “Routine”;r;” HERE”
END SUB
SUB R9
cls
r=9
print “Routine”;r;” HERE”
END SUB
SUB R10
cls
r=10
print “Routine”;r;” HERE”
END SUB
SUB R11
cls
```

```
r=11
print “Routine”;r;” HERE”
END SUB
```

**RAWNET** Rawnet is calculated from Rawgross-tare.

**RAWTARE** Rawtare is calculated by converting the tare to A/D counts. Then the tare(in counts) is multiplied by the calibration span factor. The result is not rounded to the nearest division.

**SETPWD** This keyword allows the user to set the password, from BASIC, for the SUB SYSTEM_SETUP event handler.
Example:

```
SUB SYSTEM_STARTUP
dispmode=16
setpwd(1,“130”)
END SUB

SUB SYSTEM_SETUP
cls
print “You just executed the”
print “SYSTEM_SETUP CODE!!”
END SUB
```

*First parameter must be “1”.*
*In this example “130” is the number entered as the password.*

**** Integer Divide Operator
This operator divides 2 numbers and returns the Integer portion of the computation.
Example:

```
SUB SYSTEM_STARTUP
dispmode=10
Cls
Print gross \ 3
END SUB
```

*Space needed on both sides of the "\"*

**^** Power of Operator
The result of the Power of operator(^) is the product of multiplying the first operand by itself, the second operand times.
Example:

```
SUB SYSTEM_STARTUP
dispmode=10
Cls
Print (2^8);”=”;(2*2*2*2*2*2*2*2);”=256”
END SUB
```

**m** or **MOD**

Modulus Operator

The result of the modulus operator (m) is the remainder when the first operand is divided by the second.

The “m” must always be lower case and must always have a space before and after it.

*Space needed on both sides of the "m" or "mod."*

Example:

```
SUB SYSTEM_STARTUP
dispmode=10
cls
Print (12345678 m 7);"=2"    'mod 7
Print (12345678 mod 7);"=2"    'mod 7
END SUB
```

**SYNTAX:**

**INPUT**

The input statement has been enhanced to accept a variable for the prompt.

Example:

```
SUB SYSTEM_STARTUP
dispmode=10
Prompt$="PROMPT: "
Input Prompt$,value$
Print value$
END SUB
```

**SYNTAX:**

**Multi-Scale System Variables:**

**SYNTAX:** X=GROSS(5)  Gross weight from scale 5.

An optional parameter has been added to each system variable. For example, gross returns the gross weight for the current scale. Where gross(1) returns the gross weight for scale 1 and gross(0) returns the total of all active scales. System variables which have this new feature include gross, tare, net, rawgross, rawnet, rawtare, display, maxpeak, minpeak, ROC, and count. Motion, Underload, Overload, and center of zero are included also, but since they are system flags, their totals are returned differently. If any scale is in motion, overloaded or underloaded, the total for these values will return true. All scales must be at Center of zero for the total to return true. These values are accessible through the print formats also.

Example:

```
Print format #1
{gross(1),6.0},{gross(2),6.0},{gross(0),6.0}\r\n\S

'this example assumes 2 scales are enabled
SUB SYSTEM_STARTUP
dispmode=6
Settimer(2,1)
Contout(1,1)
END SUB
```

*If a scale, greater than the number of active scales is accessed, the value for scale 1 will be returned.*

```
SUB SYSTEM_TIMER2
Cls
Print gross(1)
Print gross(2)
Print gross(0)
END SUB
```

**ERR**

The system variable that returns the number of the error from the following table:

0 = no error
1 = syntax error
2 = unbalanced paren.
3 = no expression present
4 = equals sign expected
5 = not a variable
6 = Label table full
7 = duplicate label
8 = undefined label
9 = THEN expected
10 = TO expected
11 = too many nested FORs
12 = NEXT without FOR
13 = too many nested GOSUBs
14 = RETURN without GOSUB
15 = double quotes needed
16 = string expected
17 = variable name too long
18 = var not defined (DIM)
19 = too many nested WHILEs
20 = WEND without WHILE
21 = Division by Zero
22 = EEPROM Sentinal
23 = RAM Sentinal

*See the sample application, **Sys_err.cfg**, for error detection that came with your SimPoser installation disk.*

# Appendix 3: Subroutine Examples

## Sample : GETCOM$

This is an example of using GETCOM2 for data storage:

```
SUB SYSTEM
dim X$*32
dim SPACE$*32
ROOM$="<32-Blanks>"
DISPMODE(17)
KEY 1,"RECALL"
END SUB
```

This section reserves room in memory by dimensioning the variables X$ and SPACE$ each to 32 characters. Display mode is set to 17 (See *Appendix 1*). KEY labels softkey 1 as RECALL.

```
SUB COM2_MESSAGE
X$ = GETCOM$(2)
X$ = X$+LEFT$(SPACE$,(32-LEN(X$))
A$=LEFT$(X$,16)
B$=RIGHT$(X$,16)
STORE$(1,A$)
STORE$(2,B$)
END SUB
```

This section retrieves a message from serial port #2 and then makes it 32 characters long and splits this into two 16-character string variables (A$, B$), and stores them in locations 1 and 2.

```
SUB F1_KEY
C$=RECALL$(1)
D$=RECALL$(2)
X$=C$+D$
PRINT X$
END SUB
```

This section recalls the two 16-character strings from memory, combines them and prints them to the display.

## Sample : SHOWVAR

```
SUB SYSTEM_STARTUP
dim WT
dim VAR$
dim LEGEND1$
dim LEGEND2$
dim PREC
VAR$="WT"
LEGEND1$="kg"
LEGEND2$="brutos"
PREC=2
SHOWVAR(VAR$,LEGEND1$,LEGEND2$,PREC)
SETTIMER(2,0.5)
ACTVALUE(11)
END SUB
```

This subroutine reserves memory by dimensioning the variables of the program. It assigns strings to the variables LEGEND1$ and LEGEND2$ which are used in the SHOWVAR command. SETTIMER causes the SUB SYSTEM_TIMER2 event to occur every ½ second. ACTVALUE allows the variable WT to be displayed where the weight normally is displayed on the WI-130 display.

```
SUB SYSTEM_TIMER2
WT=GROSS
END SUB
```

The SUB SYSTEM_TIMER2 subroutine assigns the system variable GROSS into the variable WT.

## Sample: STORE and RECALL

```
SUB SYSTEM_STARTUP
MUSTDIM
dim WORD$
dim LOC
dim PASSWORD
dim GUESS
dim Y
dim VNUM
dim X
WORD$=""
PASSWORD=111
DISPMODE(17)
KEY 1, "NEXT"
KEY 2, "STORE"
KEY 4, "CLRMEM"
KEY 5, "PREV"
LOC=RECALL(1000)
END SUB

SUB F2_KEY
LOC=LOC+1
input "NAME",WORD$
STORE$(LOC,WORD$)
INPUT "PHONE#",VNUM
STORE(LOC,VNUM)
STORE(1000,LOC)
END SUB

SUB F1_KEY
X=X+1
IF X>LOC THEN X=LOC
WORD$ = recall$ (X)
VNUM = RECALL (X)
PRINT WORD$;" ";VNUM
END SUB

SUB F4_KEY
GUESS=0
INPUT "PASSWORD?",GUESS
IF PASSWORD=GUESS THEN
 FOR Y=0 TO 1024
 STORE(Y,0)
 STORE$(Y,"")
 NEXT
 LOC=0
END IF
END SUB

SUB F5_KEY
X=X-1
IF X<0 THEN X=0
WORD$ = recall$ (X)
VNUM = RECALL (X)
PRINT WORD$;" ";VNUM
END SUB
```

This program gives you an example of storage and retrieval of names and numbers from memory.

## Variable analog output basis sample program

By setting the variable x to a number between 0-10 the 'voltage analog output settings will be set to this variable. If x = 2 then the analog output value=2 volts

```
SUB SYSTEM_STARTUP
mustdim
dim output
anbasis(11,"output")
setanlog(0,10)
output=0
END SUB

SUB F1_KEY
output=2
END SUB

SUB F2_KEY
output=4
END SUB

SUB F3_KEY
output=6
END SUB

SUB F4_KEY
output=8
END SUB

SUB F5_KEY
output=10
END SUB
```

```
SUB SYSTEM_STARTUP
mustdim
dim cutoff1
dim cutoff2
dim disp
cutoff1=RECALL(1023)
cutoff2=RECALL(1022)
END SUB

SUB F1_KEY
disp=dispmode
dispmode=6
input "Enter Cutoff#1",cutoff1
if lastkey=27 then
 dispmode=disp
 exit sub
end if
```

```
input “Enter Cutoff#2”,cutoff2
if lastkey=27 then
 dispmode=disp
 exit sub
end if
dispmode=disp
store(1023,cutoff1)
store(1022,cutoff2)
END SUB



‘print screen of setpoints
‘remote zero and remote tare examples
SUB SETPT1_ACT
dozero
END SUB

SUB SETPT2_ACT
dotare
END SUB




‘old way for continuous output
format #1
\x02 G {SEND$} LB\r\n

SUB SYSTEM_STARTUP
DISPMODE=10
ZERO$=”000000"
SETTIMER(1,0.5)
END SUB

SUB SYSTEM_TIMER
GROSS$=STR$(ABS(GROSS))
X=6-LEN(GROSS$)
TEMP$=LEFT$(ZERO$,X)+GROSS$
  IF GROSS<0 THEN
   SEND$=”-”+TEMP$
  ELSE
   SEND$=”+”+TEMP$
  END IF
FMTPRINT(1)
END SUB



‘new way for continuous ouput
‘max is 10 times a second
```

```
SUB SYSTEM_STARTUP
contout(1,2)                ‘format #1 output twice a second
END SUB



‘do a print screen of format 1 and serial port??
‘when a enquire EOM is received
‘print format#1 will be sent
SUB COM2_MESSAGE
fmtprint(1)
END SUB



‘multi scale axle example application with four RD-4000’s with Version 6.0
SUB SYSTEM_STARTUP
dispmode=35
dim s1
dim s2
dim s3
dim tot
END SUB

‘starts the continuous output to the daisy chained RD-4000’s
SUB F1_KEY
settimer(2,1)
END SUB

‘stops the continuous output to the daisy chained RD-4000’s
SUB F2_KEY
settimer(2,0)
END SUB

‘sub routine to interface to RD-4000 software Version 6.0
‘Version 6.0 will allow 10 RD-4000’s to be daisy chained togther
‘with seperate addresses for seperate weight displays.
‘Address 0 will send the same data to all RD-4000’s connected to the serial port
SUB SYSTEM_TIMER2
tot=0
curscale=1
s1=gross
if s1<=0 then s1=ABS(s1)
print #2,”!A1";s1
curscale=2
s2=gross
if s2<=0 then s2=ABS(s2)
print #2,”!A2";s2
curscale=3
```

```
s3=gross
if s3<=0 then s3=ABS(s3)
print #2,"!A3";s3
tot=s1+s2+s3
print #2,"!A4";tot
END SUB

SUB PRINT_KEY
fmtprint(1)
END SUB

SUB ZERO_KEY
curscale=1
dozero
curscale=2
dozero
curscale=3
dozero
curscale=1
END SUB

'if to much motion occurs the zero will be aborted after 5 tries
SUB ZERO_ABORT
x=x+1
if x<5 then
 call ZERO_KEY
else
 dispmode=22
 print " ZERO ABORTED!!!"
 print "MOTION ON SCALE!!"
 print "  TRY AGAIN!!!!"
 sleep(3)
 dispmode=35
 x=0
end if
END SUB
```

# Appendix 4: Error Messages

| | Message | Meaning |
|---|---|---|
| **When Saving** | Drive not Ready | No disk in drive |
| | Disk Write Protected | Write protect tab on diskette is in place. |
| **Edit Menu** | Invalid Line Number | A line number that doesn't exsist.(Goto Line# in Edit Menu) |
| | Search Item Not in this Program! | You tried to find something that wasn't there. |
| **Downloading** | Begin Transfer.....Bad Block | Cabling problem has occured, intermittent connection. |
| **Other** | Subscript Out of Range | Restart SimPoser. |
| | Out of Memory | 1. You have too many applications running in Windows®.<br>2. Your SimPoser configuration file is too large. |
| | Permission Denied | Do a FILE, SAVE AS c:\simposer\dat then a FILE, SAVE AS A: |
| | Disk not ready | Make sure disk is inserted properly. |
| **Non-Mouse Users:** | ALT-F6 | Gets you back to the main screen if you were in either SETPOINTS or CONFIGURE. |
| **WT-BASIC Errors** | syntax error | A keyword or command was spelled wrong or used improperly. |
| | unbalanced paren | A paraenthesis is missing. |
| | no expression present | Missing part of an expression. |
| | equals sign expected | Missing a space before a quotation mark , a system variable is mistakenly used as a regular variable, or variable name is too long. |
| | not a variable | Trying to assign a value to a command statement. |
| | Label table full | May have too many subroutines or not enough memory. Try adding the extended memory option. |
| | duplicate label | You've duplicated an event name. |
| | undefined label | Statement label in a GOTO is needs to be defined. |
| | THEN expected | Part of IF THEN statement is missing. |

| Message | Meaning |
|---|---|
| TO expected | TO missing in a FOR. . .NEXT loop. |
| too many nested FORs | To many levels of FOR. . .NEXT loops. |
| NEXT without FOR | FOR. . NEXT loop missing the FOR. |
| too many nested GOSUBs | To many levels of GOSUB. |
| RETURN without GOSUB | GOSUB. . .RETURN loop misisng the GOSUB. |
| double qoutes needed | Quotation marks missing. |
| String Expected | A string variable has no string assigned to it or a $ is missing. |
| Variable Name Too Long | Variable name has too many characters. |
| Var Not Defined (DIM) | Variable name is probably misspelled or not dimensioned. |
| too many nested WHILEs | To many levels of WHILE statements. |
| WEND without WHILE | WHILE. . .WEND without a WHILE. |
| Division by Zero | You cannot divide by a value of zero. |

# Appendix 5: ASCII Chart

| Code # | Control Character | Code # | Control Character | Code # | Control Character | Code # | Control Character |
|---|---|---|---|---|---|---|---|
| 0 | NUL | 33 | ! | 66 | B | 99 | c |
| 1 | SOH | 34 | " | 67 | C | 100 | d |
| 2 | STX | 35 | # | 68 | D | 101 | e |
| 3 | ETX | 36 | $ | 69 | E | 102 | f |
| 4 | EOT | 37 | % | 70 | F | 103 | g |
| 5 | ENQ | 38 | & | 71 | G | 104 | h |
| 6 | ACK | 39 | ' | 72 | H | 105 | i |
| 7 | BEL | 40 | ( | 73 | I | 106 | j |
| 8 | BS | 41 | ) | 74 | J | 107 | k |
| 9 | HT | 42 | * | 75 | K | 108 | l |
| 10 | Line Feed | 43 | + | 76 | L | 109 | m |
| 11 | VT | 44 | , | 77 | M | 110 | n |
| 12 | Form Feed | 45 | - | 78 | N | 111 | o |
| 13 | Carriage Return | 46 | . | 79 | O | 112 | p |
| 14 | S0 | 47 | / | 80 | P | 113 | q |
| 15 | S1 | 48 | 0 | 81 | Q | 114 | r |
| 16 | DLE | 49 | 1 | 82 | R | 115 | s |
| 17 | DC1 | 50 | 2 | 83 | S | 116 | t |
| 18 | DC2 | 51 | 3 | 84 | T | 117 | u |
| 19 | DC3 | 52 | 4 | 85 | U | 118 | v |
| 20 | DC4 | 53 | 5 | 86 | V | 119 | w |
| 21 | NAK | 54 | 6 | 87 | W | 120 | x |
| 22 | SYN | 55 | 7 | 88 | X | 121 | y |
| 23 | ETB | 56 | 8 | 89 | Y | 122 | z |
| 24 | CAN | 57 | 9 | 90 | Z | 123 | { |
| 25 | EM | 58 | : | 91 | [ | 124 | \| |
| 26 | SUB | 59 | ; | 92 | \ | 125 | } |
| 27 | ESC | 60 | < | 93 | ] | 126 | ~ |
| 28 | FS | 61 | = | 94 | ^ | 127 | Delete |
| 29 | GS | 62 | > | 95 | _ | | |
| 30 | RS | 63 | ? | 96 | ` | | |
| 31 | US | 64 | @ | 97 | a | | |
| 32 | Space | 65 | A | 98 | b | | |

# Appendix 6: Alphabetical Listing of WT-BASIC Commands

# Appendix 7: System Values Definitions

*See Appendix 2 for more information about system variables.*

The following are definitions for the system values list:

**MULTI-SCALE SYSTEM VARIABLES**

An optional parameter has been added to each system variable. For example, gross returns the gross weight for the current scale. Where gross(1) returns the gross weight for scale 1 and gross(0) returns the total of all active scales. System variables which have this new feature include gross, tare, net, rawgross, rawnet, rawtare, display, maxpeak, minpeak, ROC, and count. Motion, Underload, Overload, and center of zero are included also, but since they are system flags, their totals are returned differently. If any scale is in motion, overloaded or underloaded, the total for these values will return true. All scales must be at Center of zero for the total to return true. These values are accessible through the print formats also.

Example:
Print format #1
{gross(1),6.0},{gross(2),6.0},{gross(0),6.0}\r\n\S

'this example assumes 2 scales are enabled
SUB SYSTEM_STARTUP
dispmode=6
Settimer(2,1)
Contout(1,1)
END SUB

SUB SYSTEM_TIMER2
Cls
Print gross(1)
Print gross(2)
Print gross(0)
END SUB

**COUNT** Returns the count value of the current scale (count=net/pcwt)

**GROSS** Returns the gross value of the current scale rounded off by division size.

**NET** Returns the net (net=gross - tare) value on the current scale round by division size.

**TARE** Returns or sets the tare value on the current scale. The display will not switch to net mode automatically. Use the ACTVALUE command. Rounded by division size.

**MAXPEAK** Returns or sets the highest value on the current scale rounded by division size.

| | |
|---|---|
| **MINPEAK** | Returns or sets the lowest value on the current scale rounded by division size. |
| **ROC** | Returns the rate of change of the current scale per configured parameters. |
| **MOTION** | Motion flag. Returns -1 if motion, 0 if no motion on the current scale. |
| **CZERO** | Center of Zero flag. Returns -1 if center of zero, 0 if not on the current scale |
| **OVERLD** | Overload flag. Returns -1 if overload, 0 if not on the current scale. (gross up to 120% of capacity) |
| **UNDERLD** | Underload flag. Returns -1 if underload, 0 if not on current scale. (gross up to -120% of capacity) |
| **GROSSTOT** | Returns or sets the gross total used by the DoAccum function to total the gross weights. (Grosstot=Ø to reset) |
| **NETTOT** | Returns or sets net total used by the DoAccum function to total the net weights. |
| **COUNTTOT** | Returns or sets count total used by the DoAccum function to total the count. |
| **TRANSTOT** | Returns or sets transaction total used by the DoAccum function to total the number of accumulations. |
| **PCWT** | Returns the pieceweight value of the current scale. (pcwt = 0.0017) |
| **RAWGROSS** | Returns the unrounded gross weight of the current scale. |
| **CURSCALE** | Returns or sets the current active scale on a multiscale system. |
| **TIME$(n)** | Returns or sets the internal time. |
| **DATE$(n)** | Returns or sets the internal date. |
| **DIVISION** | Returns or sets the division size of the current scale. |
| **CAPACITY** | Returns the capacity of the current scale. |
| **DISPLAY** | Returns the current system variable being displayed. (not the actual number on the display) |
| **CURUNIT** | Returns or sets the value of the current units.<br>0=lbs 2=grams 4=lb oz 6=custom2<br>1=kg 3=oz 5=custom1 |
| **CURUNIT$** | Returns the label of the current unit of measure for the current scale. |
| **RAWNET** | Rawnet is calculated from Rawgross-tare. |
| **RAWTARE** | Rawtare is calculated by converting the tare to A/D counts. Then the tare(in counts) is multiplied by the calibration span factor. The result is not rounded to the nearest division. |

**ERR**

**SYNTAX:**

X=ERR

or

PRINT ERR

Use ERR in SUB SYSTEM_ERR system event to detect and display the last system error that occurred.

The system variable that returns the number of the error from the following table:

0 = no error
1 = syntax error
2 = unbalanced paren.
3 = no expression present
4 = equals sign expected
5 = not a variable
6 = Label table full
7 = duplicate label
8 = undefined label
9 = THEN expected
10 = TO expected
11 = too many nested FORs
12 = NEXT without FOR
13 = too many nested GOSUBs
14 = RETURN without GOSUB
15 = double quotes needed
16 = string expected
17 = variable name too long
18 = var not defined (DIM)
19 = too many nested WHILEs
20 = WEND without WHILE
21 = Division by Zero
22 = EEPROM Sentinal
23 = RAM Sentinal

**Weigh-Tronix**
1000 Armstrong Dr.
Fairmont, MN 56031 USA
Telephone: 507-238-4461
Facsimile: 507-238-4195
e-mail: industrial@weigh-tronix.com
www.weigh-tronix.com

**Weigh-Tronix Canada, ULC**
217 Brunswick Blvd.
Pointe Claire, QC H9R 4R7 Canada
Telephone: 514-695-0380
Facsimile: 514-695-6820

WEIGH-TRONIX
Weighing Products & Systems