Avery Weigh-Tronix

Model 1310 SimPoser® Software User's Manual

Printed in USA Weigh Bar® is a registered trademark of Weigh-Tronix Inc.

03/02/05 1310SIMPOSER_U.P65 PN 29804-0015A e8

Table of Contents

Introduction

About This Manual

This manual covers the information you need to install and operate the Model 1310 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.

Model 1310 SimPoser™

Attention

Some of the scale functions listed below cannot be simulated on your computer. You will need to download the program to an indicator for testing purposes.

Functions:
Multiscale inputs
Quartzell
SensorComm
Networks
Memory management
Setpoints

Model 1310 SimPoser™ software works exclusively with the Model 1310 indicator. 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 Model 1310. Model 1310 SimPoser lets you compose application programs and configuration setups for the Model 1310. You then test an application on the simulator. This makes quick fixes easy and assures identical function when you download the program to a Model 1310.

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 Model 1310 is as easy as clicking a mouse button. The information you send to the Model 1310 is instantly active and the Model 1310 is ready to perform the task you have set it up to do. The program becomes part of the Model 1310's permanent memory and cannot be lost due to power failure. If you download a new program to the Model 1310, the old program is replaced with the new program. There are 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.

Model 1310 SimPoser® Software

Minimum and Recommended Computer Requirements

Model 1310 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 Pentium II, 233 Mhz microprocessor
- 128 megabytes of RAM
- 120 megabyte hard disk drive (program takes up a minimum of 50 meg.)
- CD-ROM Drive
- VGA color monitor
- Mouse
- Microsoft Windows® 95

Recommended Configuration

- IBM® compatible AT®/PC notebook computer with an Intel Pentium III microprocessor
- 256 megabytes of RAM
- 120 megabyte hard disk drive (program takes up a minimum of 50 meg.)
- CD-ROM Drive
- VGA color monitor
- Internet access
- Mouse
- Microsoft Windows® Win95, Win98, WinNT, Win2000, or WinXP

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

- SimPoser Installation
- Starting Model 1310 SimPoser
- Commands Overview
- SimPoser Operation

Model 1310 SimPoser Installation

Place the Model 1310 SimPoser CD in the CD drive. The CD should autorun. Click on the **Install M1310 Simposer** button. Key in the company name and key code and follow all the on-screen prompts.

If your CD does not autorun, double-click the CDInstall.exe file contained on the CD from your Windows Explorer. Click on the **Install M1310 Simposer** button. Key in the company name and key code and follow all the on-screen prompts.

Starting Model 1310 SimPoser

1. Once Model 1310 SimPoser is installed, double click the icon with the left mouse button. A license message appears and upon accepting, this toolbar is displayed:

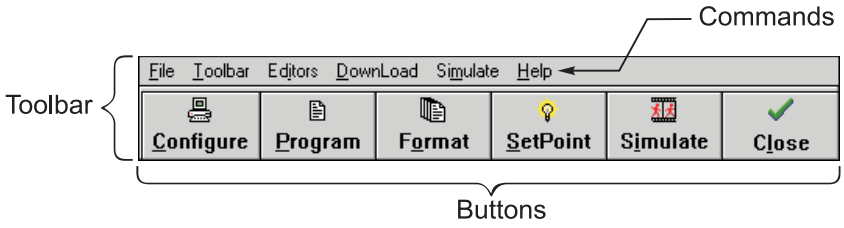


Figure 1
Model 1310 SimPoser's toolbar

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 Model 1310 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 Model 1310. COM1 and COM 2 refer to the serial output from your computer, NOT the serial ports on your Model 1310.

Com 3 through Com 5 have been added in case you are using a USB to serial converter.

HINT: If you have an application with continuous output, you should disable it before you download. To do this, power down the indicator, hold in the **CLEAR** *key and power the unit back up. After the words* Basic Disabled *are displayed, release the* **CLEAR** *key, then download your program.*

Simulate

Click on this command, then *Start* to bring up the Model 1310 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 Model 1310.

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, *Model 1310 SimPoser Operation*.

Configure button

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

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

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 specialized counting program, setup a truck in/out system, 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 32) 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 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 Model 1310 simulator on the computer. This does the same thing as the Simulate command described earlier.

Close

Click on this button to exit the Model 1310 SimPoser program.

The Model 1310 can be sealed for legal-for-trade use and the software protected from change by a switch located under the nylon plug on the rear panel.

Model 1310 SimPoser Operation

Toolbar Commands

Following are specific instructions for each of the commands on the Model 1310 SimPoser toolbar.

File

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 Model 1310 SimPoser program. There is also a file history list which appears after you've opened a file. This list will hold up to nine of the most recently opened file names. You can click on the name of the file to open it.

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

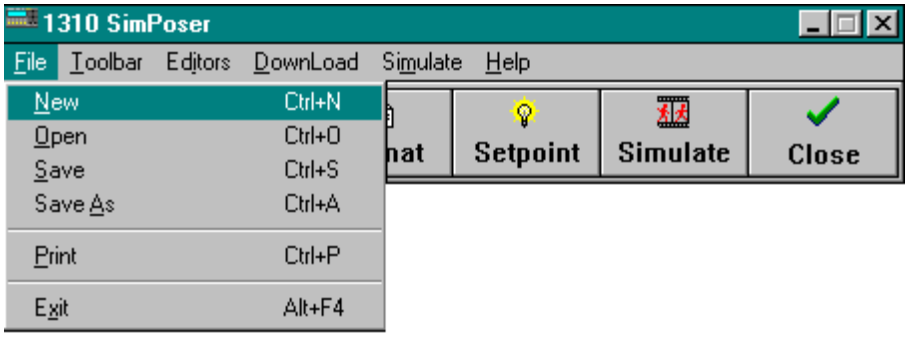


Figure 2
File menu

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

Toolbar

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

The following is shown.

If the Model 1310 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 performed a download or simulation, the Model 1310 SimPoser program creates a temporary copy of the application in C:\wt\1310\SIM\tempcfg.310. To recover the program, open this file in Model 1310 SimPoser and do a FILE,SAVE AS,{filename} and use the file name you want for your application program.

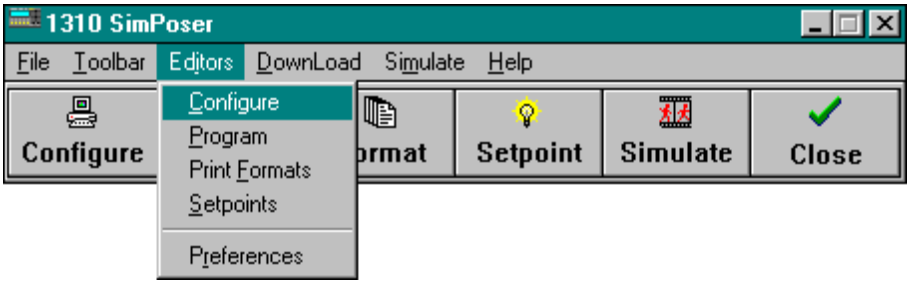
Click on **Open Small Toolbar** to replace the large toolbar with the smaller one shown below. . .

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 far right side. You cannot close the Model 1310 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.

Editors

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

The following is displayed:



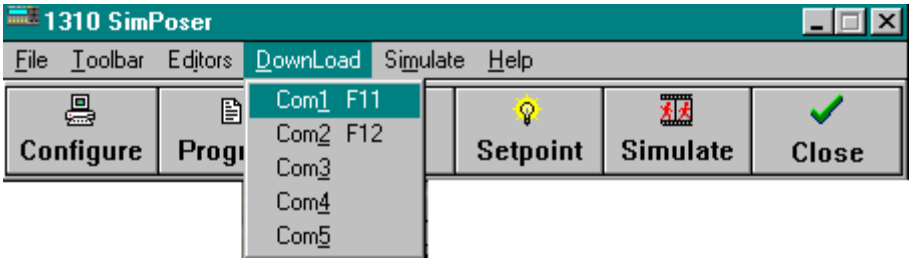
Click on the editing window you want. This is a duplication of the buttons and is handy for navigation of open windows.

The **Preferences** menu item allows you to configure the SimPoser program so that a **Tools** command appears on all the SimPoser toolbars. For more information about using this function see a Windows or DOS manual.

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 Model 1310

HINT: If you have an application with continuous output, you should disable it before you download. To do this, power down the indicator, hold in the **CLEAR** *key and power the unit back up, then release the* **CLEAR** *key.*

When you have created a custom program, use this command to choose which Com port to use to download the program to your Model 1310. F11 and F12 keys can be used as hot keys for these functions.

Simulate

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

A simulation of the Model 1310 appears on screen. It will behave in a manner similar to a real Model 1310 loaded with the program you have active in Model 1310 SimPoser. Use this to test your program before downloading to the real Model 1310. **Always verify function of a program on a live indicator.**

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 Model 1310 serial port number 2 and =1 is the communication port from your computer.
Never set both WTPORTS to the same computer COM port number.

Some of the scale functions listed below cannot be simulated on your computer. You will need to download the program to an indicator for testing purposes.

Functions:
Multiscale inputs
Quartzell
SensorComm
Networks
Memory management
Setpoints

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

COMPUTER KEY STROKE	EVENT QUEUED
ALT-S	SELECT_KEY
ALT-Z	ZERO_KEY
ALT-T	TARE_KEY
ALT-C	CLEAR_KEY
ALT-U	UNITS_KEY
ALT-P	PRINT_KEY
ALT-X	EXIT
ALT-Page Up	Increment contrast
ALT-Page Down	Decrement contrast
ENTER	ENTER_KEY
ESCAPE	ESC_KEY
1..9	NUMERIC_KEY
A..Z	ENTRY_KEY
F1..F10	F1..F10_KEY
.	DECIMAL POINT

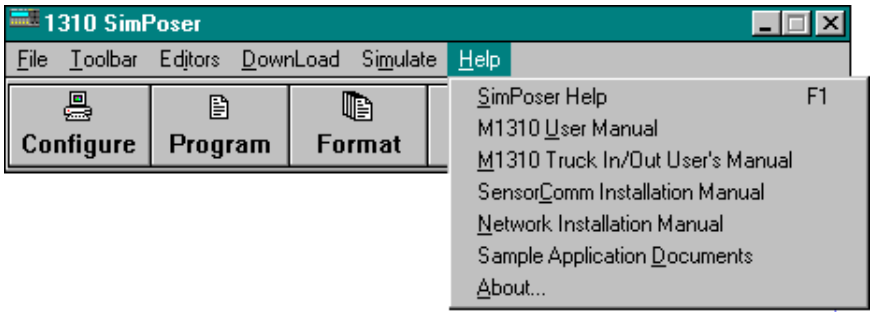
F6-F10 are accessible through a remote keyboard or the simulator. All the above are available from a PS2-style keyboard attached to your 1310 indicator.

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 and you will see the drop down list shown below:



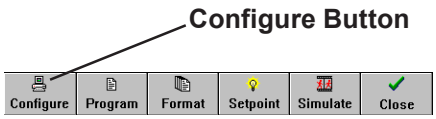
Click on the manual name under ***Help*** to bring up that manual in PDF format.

Click on Sample Application Documents to open Windows Explorer. Double click the appropriate folder and the document will open.

Click *About* to see the part number, revision level and serial number of the SimPoser software.

This is the last item on the command line of the Model 1310 SimPoser toolbar. The next section describes the toolbar buttons.

Toolbar Buttons



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.

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

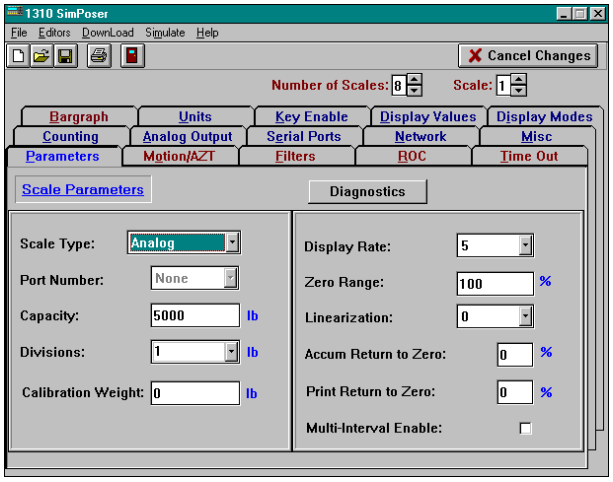


Figure 3
Configure dialog box

At the top of the screen are listed the number of scales connected and the active scale number. Use these boxes to set the number of scales you have and choose which scale you are configuring.

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 item you see in the Parameters tab in Figure 3.

Scale Type Select Analog, Quartzell® , Pulse Counter, SensorComm™, Position X or Position Y (see note at left). If you choose SensorComm, a **setup** button appears, Figure 3a. Click this button to see the dialog box in Figure 3b. Follow the steps following Figure 3b to configure the SensorComm.

When you configure a scale for Pulse Counter, you will be able calibrate the pulses to a given unit of measure. The GROSS keyword will return the calibrated pulses. Setpoints, print formats and networks will use the GROSS keyword for the current scale to access the counter's value.

Also, a Quadrature Enable box appears when you enable a PULSE COUNTER scale. Quadrature encoders are used to create a pulse when an object passes by the encoder, as well as the direction which it is traveling. For example: Clockwise motion, could increment the pulse counter, where counter clockwise motion decrements the pulse counter.

Port No. For Quartzell® only. Select the serial port in the Model 1310 to which you want to connect the Quartzell. Connect to port #1 (TB26B), port #3 (TB28B), or port 4 (TB29B).

Attention

The Traxle application requires you to create 3 active scales even though you are using only one physical truck scale. Scale 1 must be configured as SensorComm, Scale 2 must be Position X and Scale 3 must be Position Y.

Traxle uses position sensing to compute individual axle weights.

See Traxle Configuration ***section for instructions.***

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 1, 2, or 5. If you type an incorrect value, the program will automatically select the closest correct value.

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.)

Attention

You must have a Traxle enabled Model 1310 indicator to use the Traxle application.

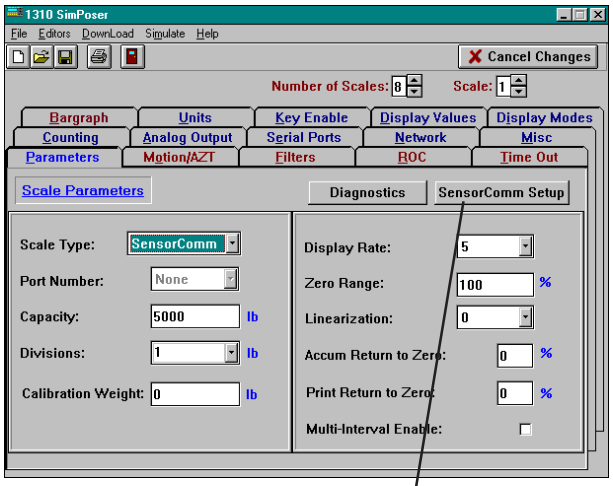


Figure 3a
setup button

The information you enter into the dialog boxes in Figure 3b is transferred to connected SensorComm j-boxes when you download the configuration file to your indicator. This information configures each SensorComm j-box.

*Cell **Factors** will only be sent if this box is checked. You can find the data for cell factors on a tag attached to the cable of each Weigh Bar. Use 1 as a default value if the information is not available.*

SensorComm connects to the Model 1310 via a serial Port. Select that Port Number here.

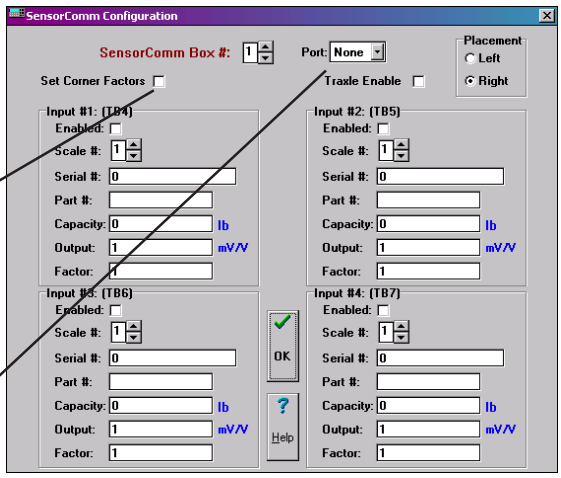
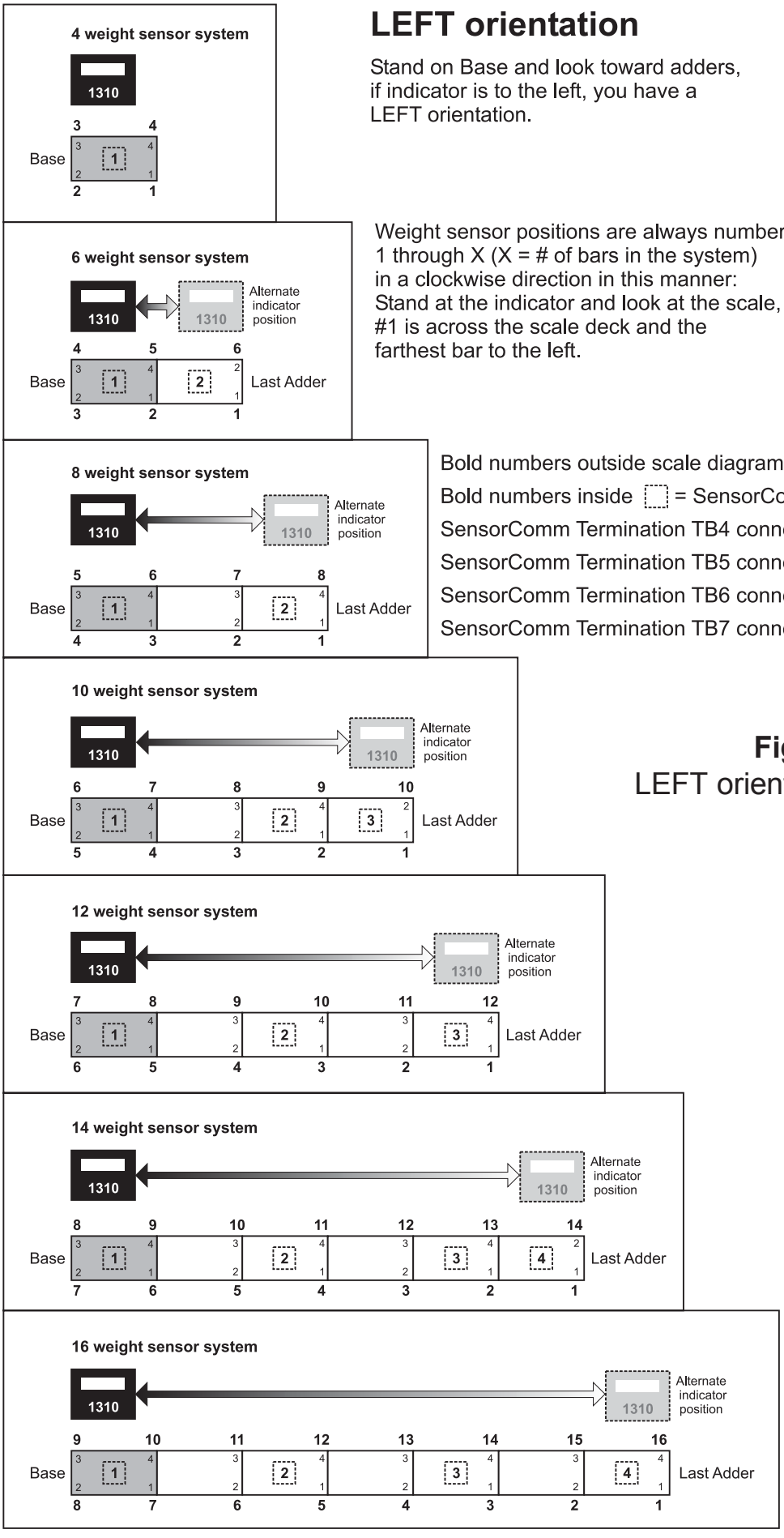


Figure 3b
SensorComm Configuration dialog box

LEFT orientation

Stand on Base and look toward adders, if indicator is to the left, you have a LEFT orientation.



Weight sensor positions are always numbered 1 through X (X = # of bars in the system) in a clockwise direction in this manner: Stand at the indicator and look at the scale, #1 is across the scale deck and the farthest bar to the left.

Bold numbers outside scale diagrams = Weight sensor number

Bold numbers inside ⬚ = SensorComm J-Box Number

SensorComm Termination TB4 connects to Sensor #1

SensorComm Termination TB5 connects to Sensor #2

SensorComm Termination TB6 connects to Sensor #3

SensorComm Termination TB7 connects to Sensor #4

Figure 3c
LEFT orientation numbering

RIGHT orientation

Stand on Base and look toward adders, if indicator is to the right, you have a RIGHT orientation.

Weight sensor positions are always numbered 1 through X (X = # of bars in the system) in a clockwise direction in this manner: Stand at the indicator and look at the scale, #1 is across the scale deck and the farthest bar to the left.

Bold numbers outside scale diagrams = Weight sensor number

Bold numbers inside ⬚ = SensorComm J-Box Number

SensorComm Termination TB4 connects to Sensor #1

SensorComm Termination TB5 connects to Sensor #2

SensorComm Termination TB6 connects to Sensor #3

SensorComm Termination TB7 connects to Sensor #4

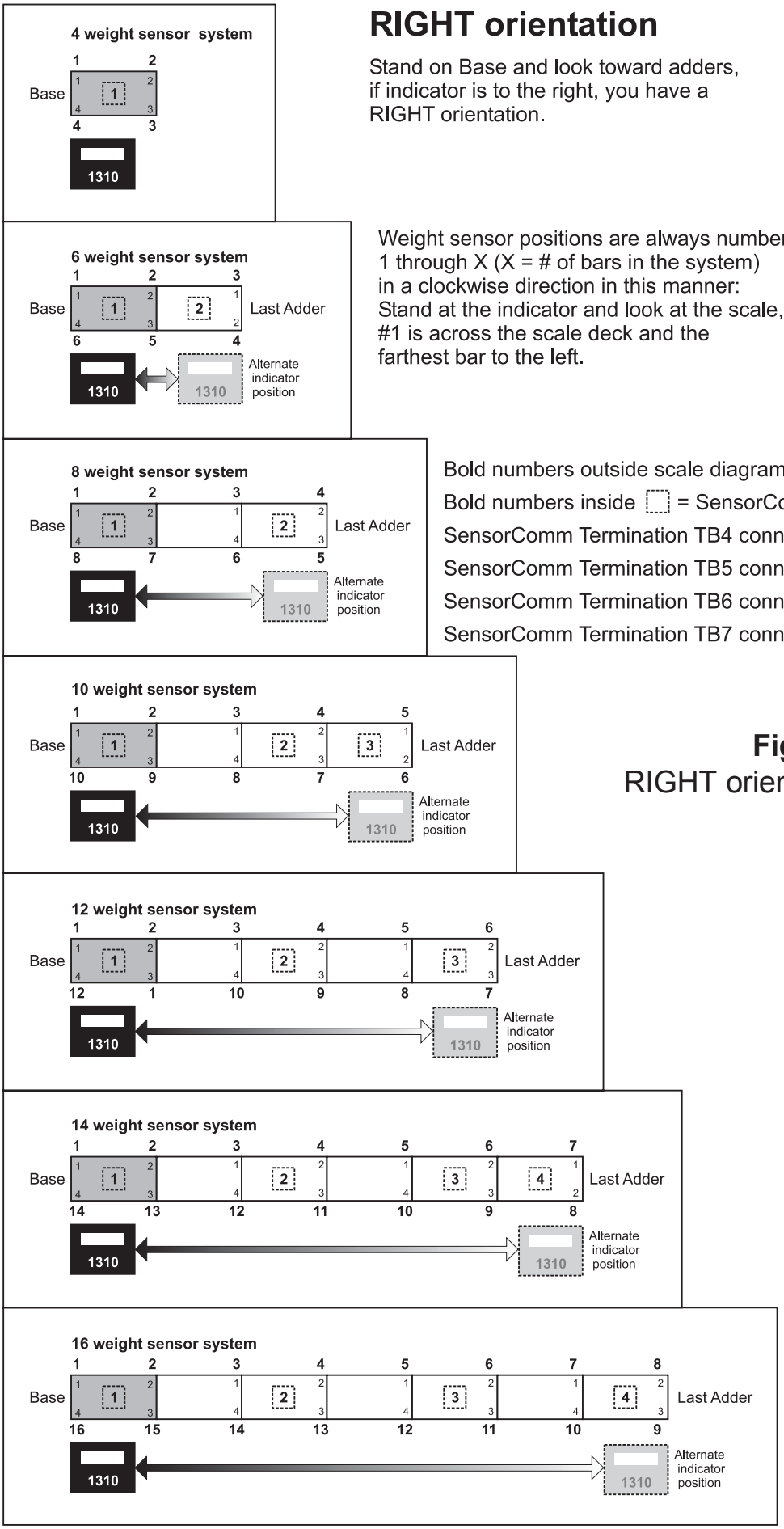
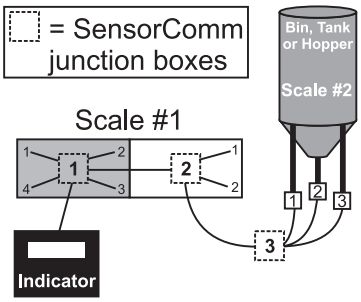


Figure 3d
RIGHT orientation numbering



Shown above is a two scale daisychain. Weight sensors #1-3 in scale #2 are attached to SensorComm #3. These three weight sensors would be assigned to Scale #2.

The six weight sensors in scale #1 should be assigned to Scale #1.

The Model 1310 needs to know where it is located in relationship to the scales and to the SensorComm hardware in the scale so it can communicate properly.

If you do not enable Traxle, the first setting you make should be *Placement*. Is your setup a RIGHT or LEFT orientation? See Figures 3c and 3d to determine which you have.

You must configure each SensorComm junction box and each weight sensor connected to that junction box. Refer to Figures 3c or 3d. Fill out the following for each weight sensor:

Enabled Check this box if there is a weight sensor connected to this input (TB4, TB5, TB6, or TB7)

Scale # SensorComm junction boxes in separate scales can be daisychained. Identify which scale a weight sensor is associated with in this dialog box. See note at left.

Serial # Enter the serial number of the weight sensor.

Part # Enter the part number of the weight sensor.

Capacity Enter the capacity of the individual weight sensor.

Output Enter the precise mV/V output of each weight sensor.

Factor Enter the span factor. This is listed on each Weigh Bar.

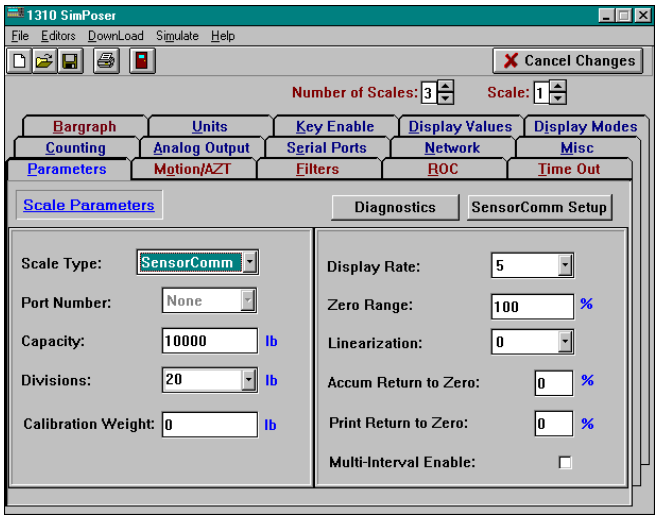
Traxle Configuration

The Traxle application requires you to create 3 active scales even though you are using only one physical truck scale. Scale 1 must be configured as SensorComm, Scale 2 must be Position X and Scale 3 must be Position Y.

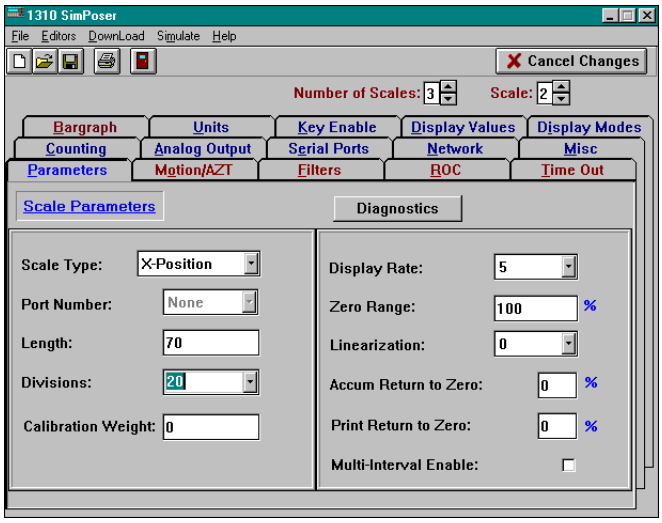
Traxle uses the position X and Y capabilities to compute individual axle weights.

Follow these steps to set up your truck scale for Traxle operation:

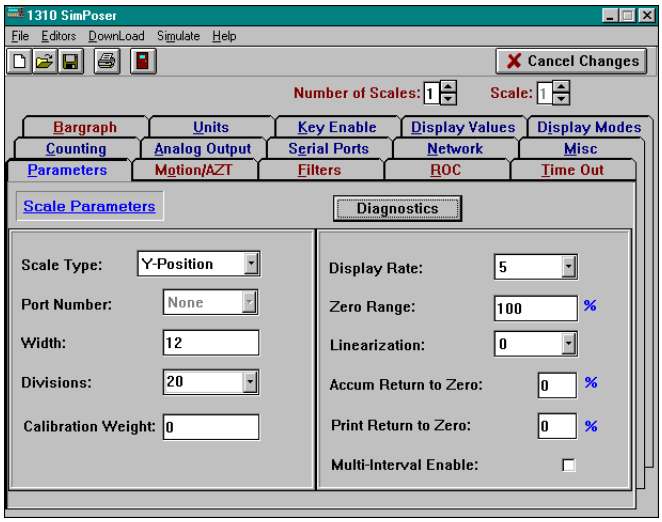
1. Under *Scale Parameters*, set the *Number of Scales* to 3, select *Scale* #1, choose SensorComm as the *Scale Type*, then set the rest of the parameters to your system needs. See example below:



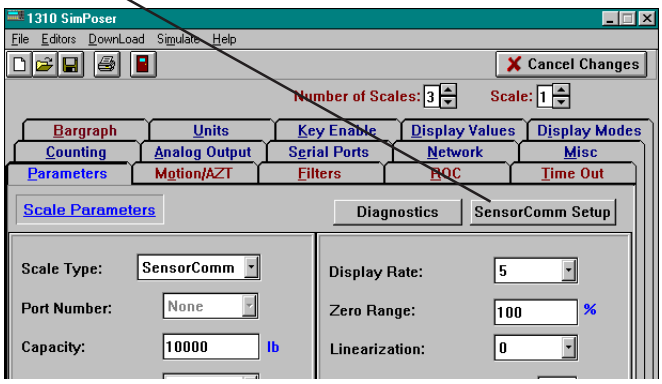
2. Select *Scale* #2, select X-Position for *Scale Type*, and enter the length of your scale, in feet, under *Length*. See example below:



3. Select *Scale* #3, select Y-Position for *Scale Type*, and enter the width of your scale, in feet, under *Width*. See example below:

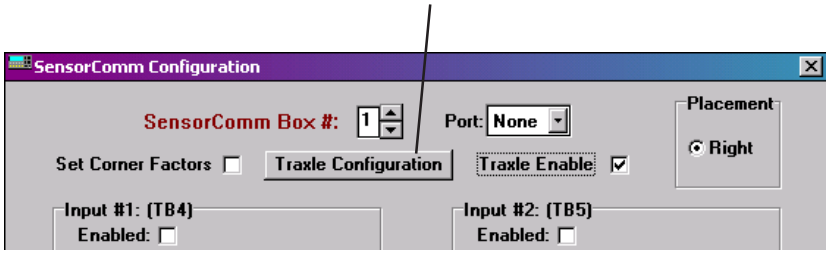


4. Select *Scale* #1 again. Click the **SensorComm Setup** button located here.

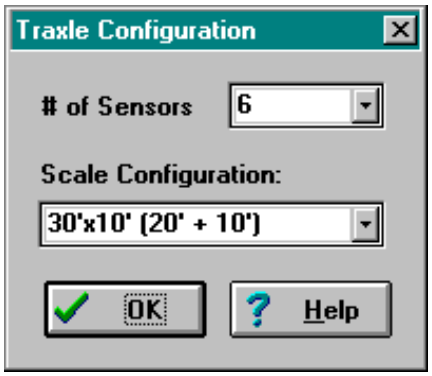


Traxle applications require a RIGHT orientation to the system.

If this is a Traxle application, select *Traxle Enable*. If it is a Traxle application, the placement choice is limited to just right orientation. The left orientation choice disappears from the dialog box. If you enable Traxle, a Traxle Configuration button appears. See illustration below:



Click the **Traxle Configuration** button and the following dialog box appears:



Choose the # of weight sensors in your Traxle system, then choose your scale configuration from the drop down list. The scale configuration choices are reproduced below. Click **OK** to accept.

6 Sensors	
Total L x W	**Deck sizes**
30' x 10'	20'+ 10'
35' x 10'	23' +12'
40' x 10'	2 x 20'
47' x 10'	2 x 23'
50' x 10'	2 x 25'
30' x 11'	20'+ 10'
35' x 11'	23' +12'
40' x 11'	2 x 20'
47' x 11'	2 x 23'
50' x 11'	2 x 25'
30' x 12'	20'+ 10'
35' x 12'	23' +12'
40' x 12'	2 x 20'
47' x 12'	2 x 23'
50' x 12'	2 x 25'
30' x 13'	20'+ 10'
35' x 13'	23' +12'
40' x 13'	2 x 20'
47' x 13'	2 x 23'
50' x 13'	2 x 25'

8 Sensors	
Total L x W	**Deck sizes**
60' x 10'	3 x 20'
70' x 10'	3 x 23'
75' x 10'	3 x 25'
70' x 10'	3 x20' +10'
60' x 11'	3 x 20'
70' x 11'	3 x 23'
75' x 11'	3 x 25'
70' x 11'	3 x20' +10'
60' x 12'	3 x 20'
70' x 12'	3 x 23'
75' x 12'	3 x 25'
70' x 12'	3 x20' +10'
60' x 13'	3 x 20'
70' x 13'	3 x 23'
75' x 13'	3 x 25'
70' x 13'	3 x20' +10'

10 Sensors	
Total L x W	**Deck sizes**
80' x 10'	3 x 23'+ 10'
80' x 10'	4 x 20'
93' x 10'	4 x 23'
100' x 10'	4 x 25'
80' x 11'	3 x 23'+ 10'
80' x 11'	4 x 20'
93' x 11'	4 x 23'
100' x 11'	4 x 25'
80' x 12'	3 x 23'+ 10'
80' x 12'	4 x 20'
93' x 12'	4 x 23'
100' x 12'	4 x 25'
80' x 13'	3 x 23'+ 10'
80' x 13'	4 x 20'
93' x 13'	4 x 23'
100' x 13 '	4 x 25'

12 Sensors	
Total L x W	**Deck sizes**
100' x 10'	5 x 20'
116' x 10'	5 x 23'
124' x 10'	5 x 25'
100' x 11'	5 x 20'
116' x 11'	5 x 23'
124' x 12'	5 x 25'
100' x 12'	5 x 20'
116' x 12'	5 x 23'
124' x 12'	5 x 25'
100' x 13'	5 x 20'
116' x 13'	5 x 23'
124' x 13'	5 x 25'

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 Model 1310. A WT-BASIC program can also define an ACCUM. soft key and assign accumulation to that key only.*

The default print format is 0 and is transmitted out of port #1. (Format 0 = Gross, Tare, Net)

Display Rate	Set the display update rate (the number of times/second the display is updated.) Choose values between .1 (slowest, once every 10 seconds) and 10 (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. Used for single point calibration. For multipoint calibration method, reference the *Service Manual*.
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.
Multi-interval	To allow the division size to be doubled at the half capacity point. This selection is useful for truck/track scale applications.

Diagnostics

Attention

Some of the scale functions listed below cannot be simulated on your computer. You will need to download the program to an indicator for testing purposes.

Functions:
Multiscale inputs
Quartzell
SensorComm
Networks
Memory management
Setpoints

There is a Diagnostics button on the Scale Parameters tab. See Figure 4 below.

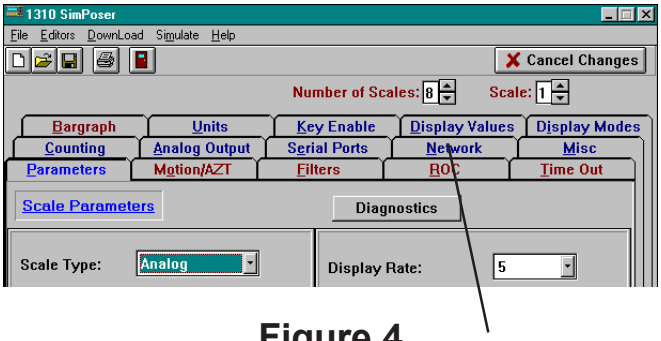
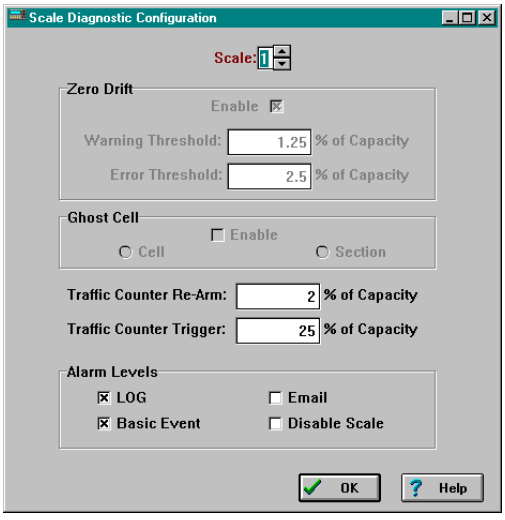


Figure 4
Diagnostics button

Click this button and you will see the following dialog box.



Use this dialog box to select a scale and set all the available diagnostics in the dialog box.

Zero Drift Enable Select this to enable and configure deadload drift analysis.

This allows you to set a warning level as a percentage of scale capacity. If the deadload weight change exceeds this set percentage, a warning is displayed on the screen. You can also set an error level as a percentage of scale capacity. If the deadload weight change exceeds this set percentage, the scale returns an error.

Ghost Cell Select *Cell* (bin, tank, hopper) or *Section* (**truck scale applications only**) to enable the ghost weight sensor option.

If you enable the *Cell* ghost option and a weight sensor on a scale system fails, you will see a display similar to this example:

GHOST appears on the display with the number of the last cell affected. Scale weight will be estimated using the inputs from the remaining functioning weight sensors. This is useful if you must keep an operation functioning, although at a reduced accuracy, for a period of time until a replacement can be installed.

If you enable the *Section* ghost option and a weight sensor in a section fails, GHOST appears on the display with the number of the cell affected. Scale weight will be estimated using the inputs from the remaining functioning weight sensors. This is useful if you must keep an operation functioning, although at a reduced accuracy, for a period of time until a replacement can be installed. See note at left.

*Only **one** failed weight sensor can be ghosted in truck scale applications. Once Ghost (section) is enabled, the system is no longer "**legal**" or "**trade approved**." The audit counters will be incremented to track this change.*

In tank, hopper, and deck scale applications the weight applied must be in a constant position, i.e. the center of gravity must be constant for Ghost (cell) to work properly. Up to three out of four cells may be "Ghosted", but the accuracy and stability decreases as the number of active cells decrease.

Traffic Counter Use the Traffic Counter boxes to set the parameters for counting traffic across a scale. Set the low weight as a percentage of capacity at which the system will rearm and be ready for the next count. The trigger amount is the weight above which the weight must rise before a weighment is counted.

Alarm Levels Select what you want to happen if any errors occur. You can choose to LOG it, cause a BASIC event, send an email or disable the scale.

When you are done setting all the info for each scale, click the OK button.

Motion/AZT tab

Motion/AZT is the next tab. The dialog box is shown in Figure 5.

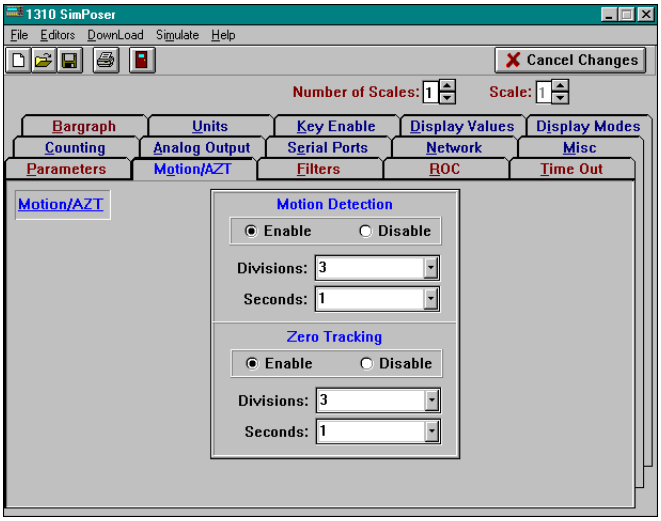


Figure 5
Motion Detection and AZT dialog box

Center of Zero is defined as 0 ±¼ of a scale division.

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 center of zero. The default is three divisions and one second.

Filters tab

Filters is the next tab. This dialog box is shown in Figure 6.

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

Harmonizer should not be used with a Quartzell Base.

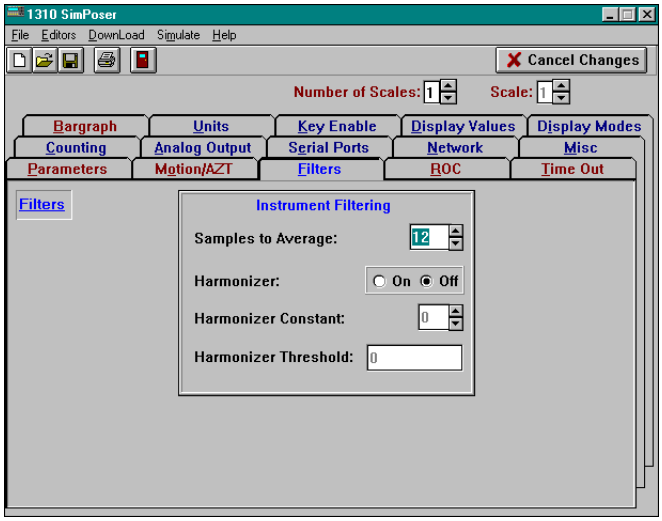


Figure 6
Filtering dialog box

Use this to set up the Harmonizer™ filtering. A full explanation of the Harmonizer™ is given below.

Samples to Average is the number of conversions you want to average. For example, with an analog base, if you pick 30, the unit will average the weight values from the last 30 conversions or ½ second and uses that value for displayed data.

On an analog base, the Samples to Average should be set between 12 and 60. **With a QDT base the value should be set to 1.**

The Harmonizer Constant choices are 0-10 in the Model 1310 SimPoser program but it may be best to make the selection in the "real world" through the front panel.

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. Typical values are between 1 and 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. Harmonizer Threshold is the amount of weight change, in calibration units, beyond which the Harmonizer will be temporarily disabled. For example, if you set this to 10 lbs, a change in weight greater than 10 lbs between samples will disable the Harmonizer until the weight change during the sample time drops below 10 pounds.

We suggest starting values for a typical analog base of:

48	Sample to Average
ON	Harmonizer
4	Harmonizer Constant
0	Harmonizer Threshold

ROC tab

ROC is the next tab. This is the Rate of Change dialog box and it is shown in Figure 7.

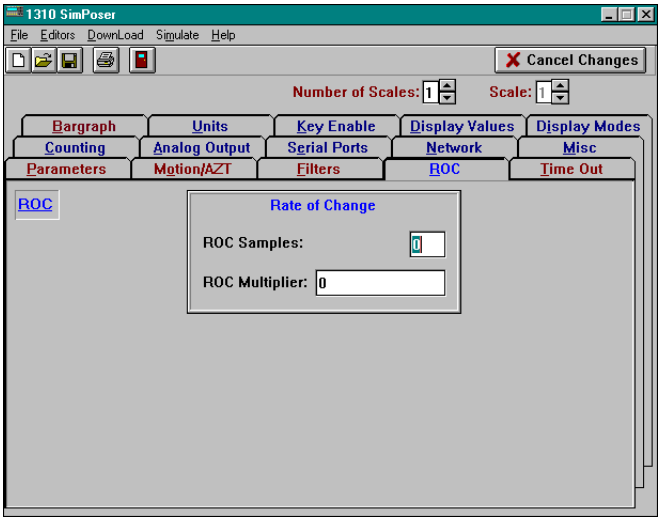


Figure 7
Rate of Change dialog box

The Rate of Change dialog box allows the user to set up a Model 1310 Indicator to calculate Rate of Change for flow rate, or weight/time, applications.

ROC is the rate of material flowing on or off a 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 Samples
ROC Samples is the number of samples over which the rate of change of weight is determined. The Model 1310 converts analog weight signals at 60 times per second. If ROC Samples is set to 60, the Model 1310 is determining the rate of weight change over one full second. For remote Quartzell bases, set Samples to 50.

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.

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.)

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

Time Out tab

Time Out is the next tab, shown in Figure 8.

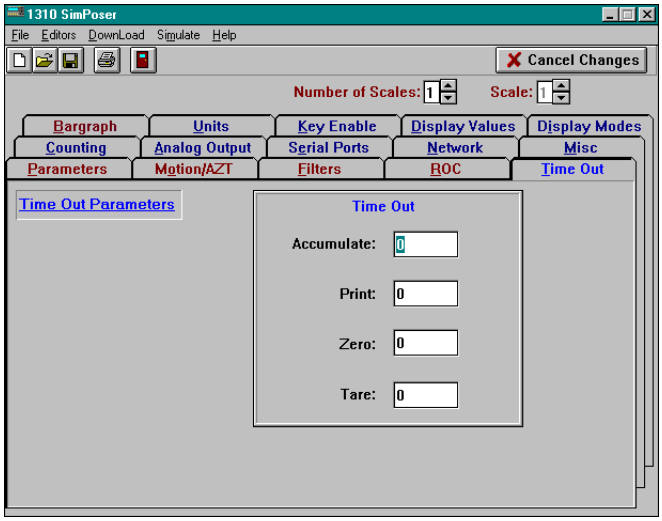


Figure 8
Time Out dialog box

Use this dialog box to set Accumulate Timeout, Print Timeout, Zero Timeout, Tare Timeout. This is the amount of time the Model 1310 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.

Counting tab

Counting is the next tab, shown in Figure 9.

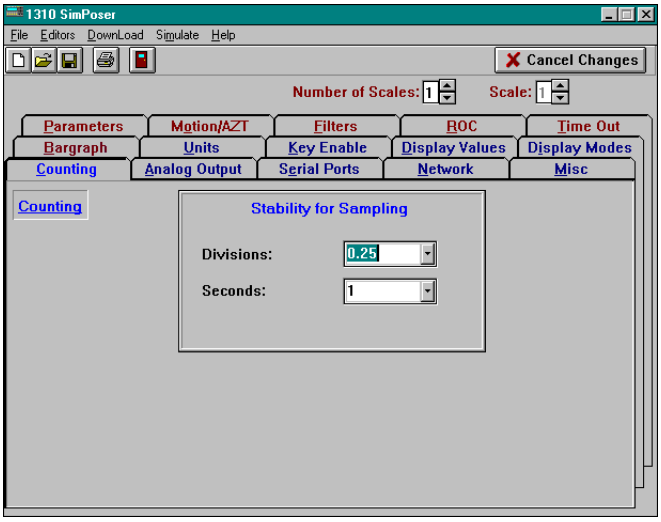


Figure 9
Counting dialog box

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

Divisions

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

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 Model 1310 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 Model 1310 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.

Analog Output tab

Analog Output is the next tab, shown in Figure 10.

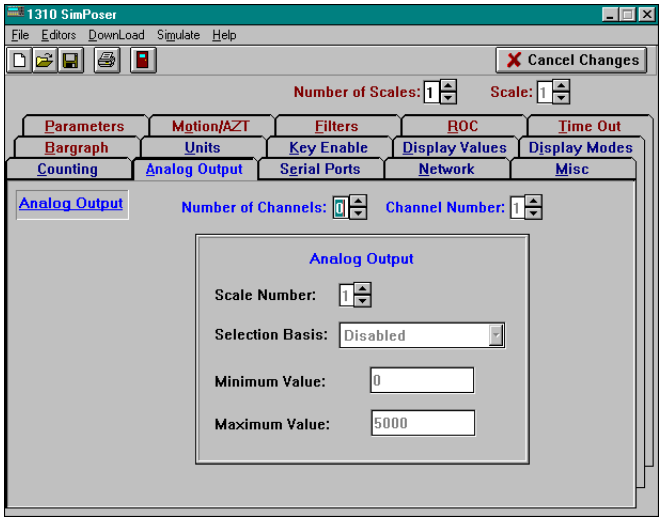


Figure 10
Analog Output dialog box

Analog Output Basis list:
Disabled
Gross Weight
Net Weight
Tare Weight
Minimum Weight
Maximum Weight
Rate of Change
Gross Weight Total
Net Weight Total
Count Total
Transaction Total
Count
Variable
Piece Weight.

In this dialog box you set the following:

1. The number of analog output channels you have
2. The channel you want to configure
3. The scale number you want the output based on. If you will be controlling the channels on the card from your BASIC program, set this value to Gross Wt.
4. What the analog output will be based on (see the list at left)
5. The basis value which will cause the minimum output from the analog board.
6. The basis value which will cause the maximum output from the analog board.

Serial Ports tab

Serial Ports is next. The serial ports dialog box is shown in Figure 11.

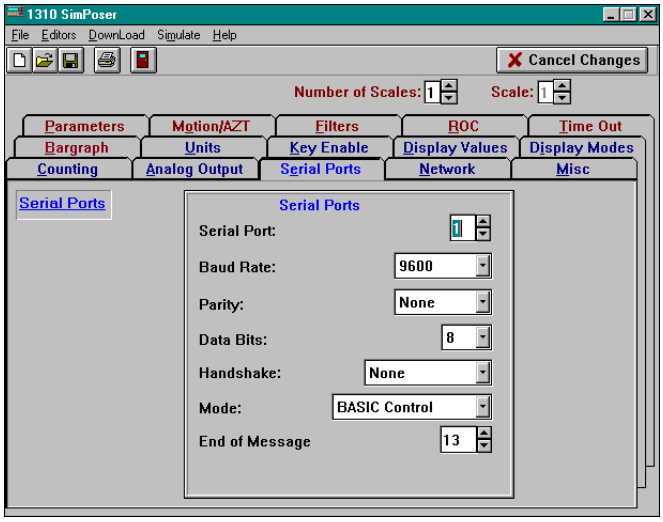


Figure 11
Serial ports dialog box

The Serial Ports dialog box allows the user to set the parameters for the four Model 1310 serial ports. Each parameter is described below.

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

Serial Port
This selection switches between Serial Port 1,2,3 or 4 . 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
	115000

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

Attention

Some of the scale functions listed below cannot be simulated on your computer. You will need to download the program to an indicator for testing purposes.

Functions:
Multiscale inputs
Quartzell
SensorComm
Networks
Memory management
Setpoints

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 Model 1310. 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 Model 1310 is listening to or receiving data from. You may share a serial port with a printer that just listens or receives data from the Model 1310.

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.

Multi-Drop- When an RS-485 network is used multi-drop mode automatically enables the RS-485 line drivers while transmitting. It tri-states the drivers when the system is done transmitting. No standard protocol is implemented. The programmer must write their own.

*Enquire Mode must be enabled and the EOM character **MUST** be set to a 5 to operate with the Truck Weigh software program.*

Computer Mode- For future use.

Enquire Mode - When the EOM character is received, the default print format will be transmitted if all motion criteria is met. (Sub COMx_Message will not be queued.)

End of Message (EOM)
Enter the decimal number that represents the ASCII character the Model 1310 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, COM2_MESSAGE, COM3_MESSAGE, or COM4_MESSAGE event is queued in the WT-BASIC program. You must then write BASIC code for the COM1_MESSAGE, COM2_MESSAGE COM3_MESSAGE, or COM4_MESSAGE event containing the GETCOM$ BASIC command.

Port 2 has a port level setting. Port 2 can be configured for RS-232 or 20mA current loop.

Network tab

Network is the next tab, shown in Figure 12.

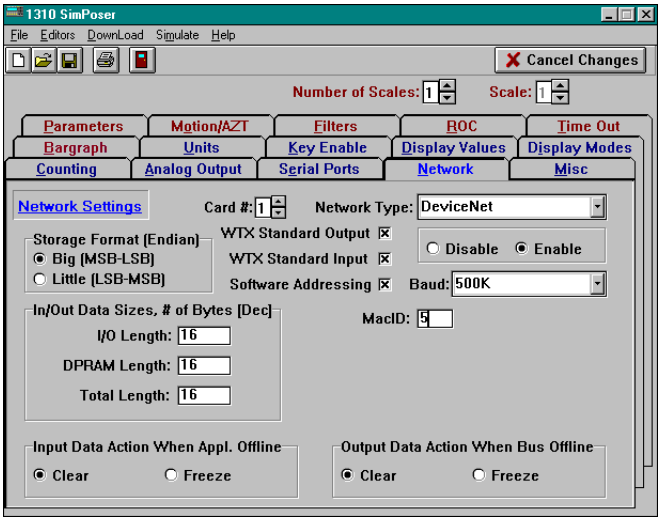


Figure 12
Network dialog box

Use this tab to configure your network card. In this box you can set the following:

1. Card number to configure
2. Network type. Figure 12 shows DeviceNet™ as an example.
3. Enable or disable WTX (Weigh-Tronix) standard output (see table at left)
4. Enable or disable WTX (Weigh-Tronix) standard input (see table at left)
5. Enter the appropriate MacID address for the network module used.
6. Disable or enable the network card
7. Configuration for the network card (data size) memory (all three values should be identical). The numbers for input and output memory represent the default data type for a given network. Using the BASIC command MAP, will require you to increase the allocated memory. Example: 16 Bytes will be declared for DeviceNet.
8. Input Data Action when Appl. (Application) off-line:

   Clear - sets all memory to zero (0) if the network goes off-line

   Freeze - leaves the memory alone if the network goes off-line
9. Output Data Action when Network Bus off-line:

   Clear - sets all memory to zero (0) if the network goes off-line

   Freeze - leaves the memory alone if the network goes off-line
10. Storage Format-Big Endian or Little Endian. This defines the order of data storage (MSB-LSB or LSB-MSB).

WT Standard Network Input

Address	*Value*
0	*reqZero*
2	*reqTare*
4	*reqPrint*
6	*reqAccum*
8	*setTare*
10	*setCurrentUnits (see CURUNIT)*

Req is motion inhibited.
Set is not motion inhibited.

WT Standard Network Output

Address	*Value*
0	*Gross*
2	*Net*
4	*Tare*
6	*Motion*
8	*Center of zero*
10	*Active value (see ActValue keyword*
12	*Overload-Underload*

Outputs: Gross, Net, Tare return the current active numeric value.

Motion and Center of Zero return a True (1) or False (0).

ActValue returns 0-13, which represents the current active value on the display.

Overload-Underload returns 0 = Okay, 1 = Under, 2 = Over

***For more detailed information on network connections and programming please reference the* Model 1310 NETWORK INSTALLATION GUIDE P/N 29806-0013.**

Misc tab

Misc is the next tab, shown in Figure 13.

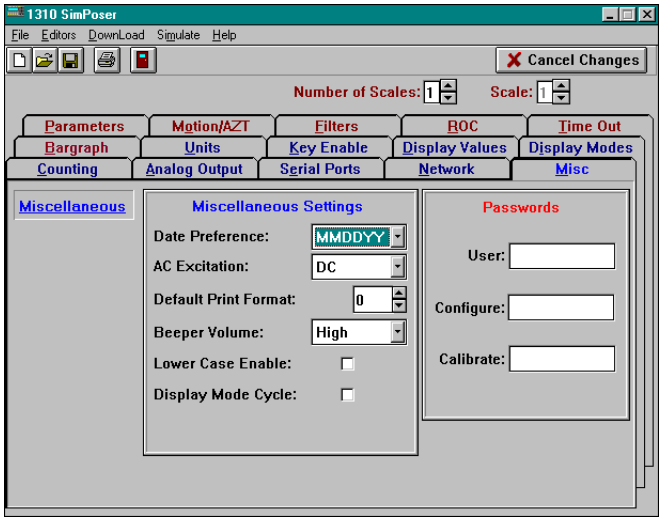


Figure 13
Miscellaneous dialog box

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:

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.

Date Preference
Allows you to set the Model 1310 System Clock in Month-Day-Year format, Day-Month-Year format or Year-Month-Day 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 format 0 outputs the following information:
G 12 LB
T 4 LB
N 8 LB

Default Print Format
Allows you to select which one of the 32 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 Model 1310 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 modes requiring BASIC text will show blank screen space if no WT-BASIC program exists to support screen text.

Display Mode Cycle
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 Model 1310 keyboard.

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

Calibrate - (Default is 30456.) Allows access to Model 1310 calibration routines.

Bargraph tab

The next tab is **Bargraph**, shown in Figure 14.

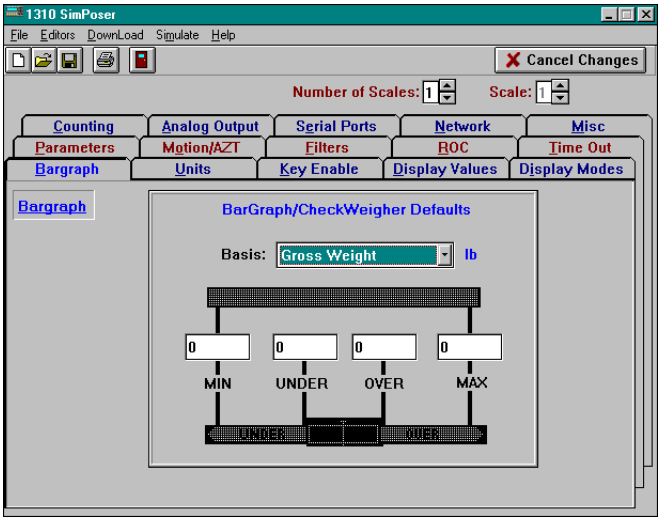


Figure 14
Bargraph/Checkweigher dialog box

Attention

Graph basis must be configured for each scale individually when more than one scale is enabled. Select the scale number then key in the MIN, UNDER, OVER and MAX values.

Use this dialog box to enter parameters relating to bargraph or checkweigher functions, if the Model 1310 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 checkweigher will begin registering movement. If the Model 1310 is set up with a bargraph, this value will determine when the graph on the display will begin moving. If the Model 1310 is set up with a checkweighing display, the "Under" portion of the checkweigher 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 Checkweigher 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 Checkweigher 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 checkweigher will end registering movement. At this point, both the bargraph and checkweigher modes reach their maximum position and do not register further movement.

Units tab

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

Figure 14 shows an example of setting 2000 lb = 1 ton for Custom Unit 1.

The next tab is **Units**. This is shown in Figure 15.

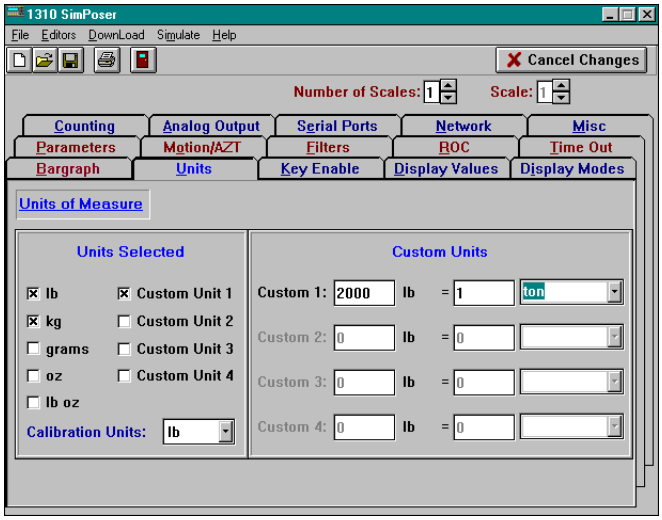


Figure 15
Units dialog box

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, 3, 4
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 Model 1310. Conversion factors are preassigned by the factory for lb, kg, grams, oz, and lb/oz.

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, enter the number of calibration units equal to a number of custom units and choose a label for the custom unit or type in your own label.

Key Enable tab

The next tab is **Key Enable**, shown in Figure 16. This dialog box lets you enable or disable the keys listed. You can also enable or disable the autotare function or the keyboard tare.

Perform an Auto Tare by placing a container on the scale and pressing the yellow **TARE** *key.*

The Keyboard Tare allows you to enter numbers, via the keypad, of known tare values.

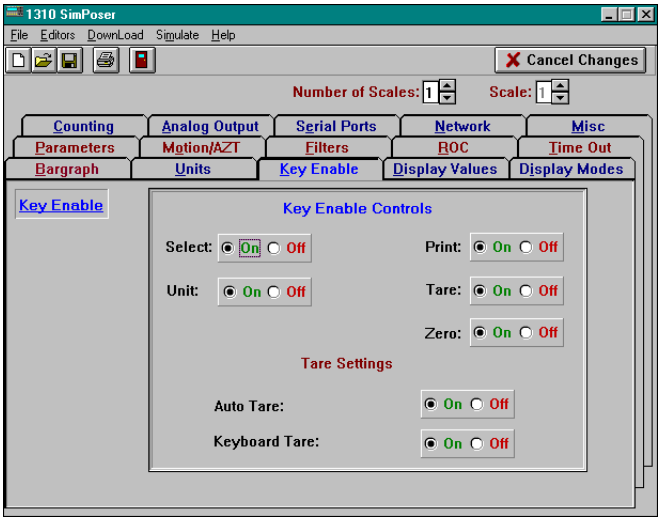


Figure 16
Key Enable dialog box

Display Values tab

Display Values is the next tab, shown in Figure 17

ROC stands for Rate Of Change.

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

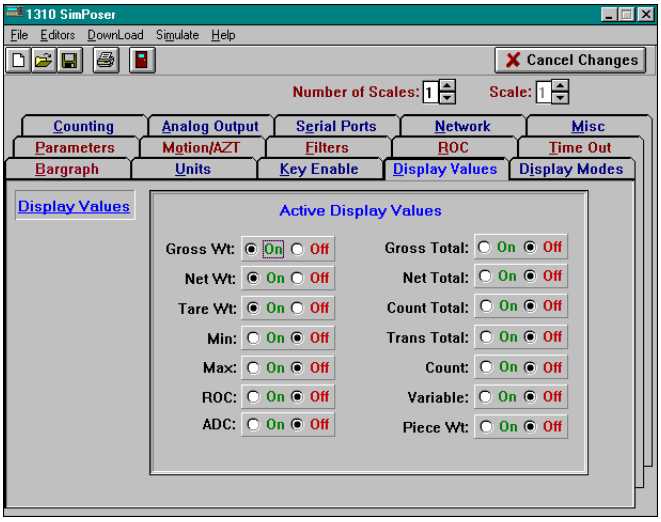


Figure 17
Display Values dialog box

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 Model 1310 during normal operation.

Display Modes tab

Display Modes is the next tab, shown in Figure 18.

See Appendix 1 for samples of displays.

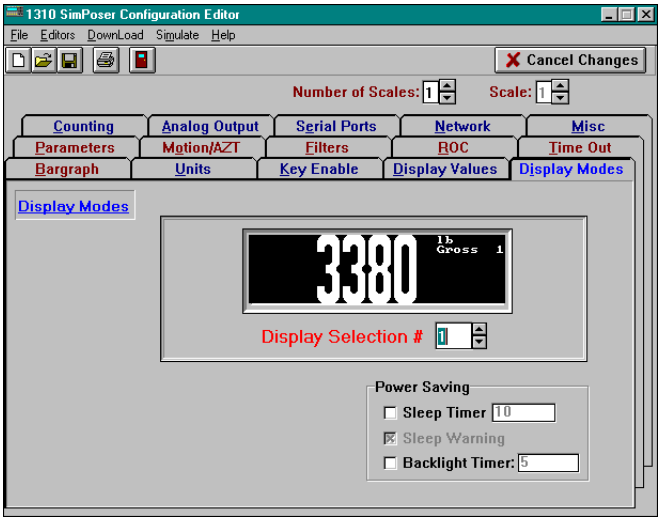


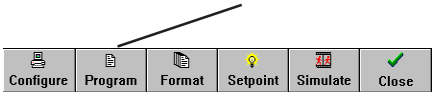
Figure 18
Display Modes dialog box

There are over 90 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.

You can set the sleep timer and sleep warning in this dialog box. Enable the timer by clicking on the box, then type in the amount of idle time which will cause the unit to go into sleep mode. If you enable sleep mode you can enable or disable the sleep warning beeper. If enabled the beeper will sound several times before the indicator goes to sleep.

You can also set a backlight timer. Click the box to enable or disable and enter a number of minutes in the window. The backlight will shut off after this set time. A press to any key on the display will turn the backlight on.

Program Button



The next button on the Model 1310 SimPoser toolbar is **Program**. Figure 19 shows the screen which appears when you click on this button with your mouse.

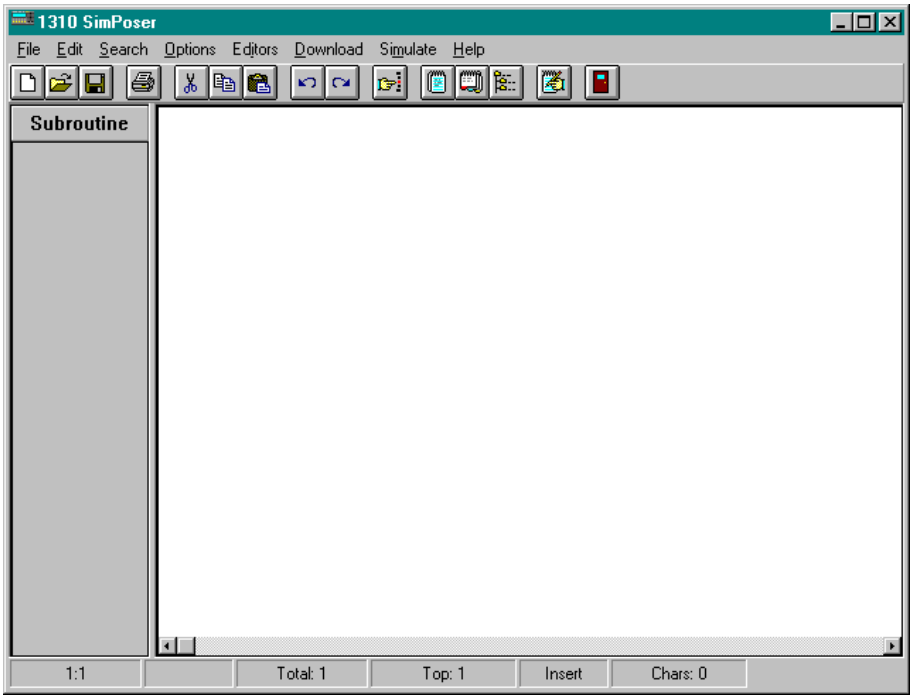


Figure 19
Program editor window

If you forgot to save your file but have either downloaded to a Model 1310 indicator or have tested your file by using the simulator, a copy of your .310 file exists in C:\wt\1310\sim\tempcfg.310

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 .310 file. This file can be run in the simulator mode or downloaded to the Model 1310.

WT-BASIC programming is the key feature of the Model 1310's power and flexibility. Using the programming system in conjunction with the Setup configurations, print formats and the Setpoint configuration allow the Model 1310 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, F4 for Replace, F5 for Go to Line and F6 for Program Errors. 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, Model 1310 SimPoser creates an error file. You can retrieve this file by clicking on **Program Errors**, or F6, in this drop down menu or clicking the program errors button on the tool bar. See button at left. 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.

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 **Tab** keys moves the cursor to the next tab position. This inserts spaces not true tabs.

Real Tabs - Pressing **Tab** keys moves the cursor to the next tab position. This inserts real tabs into the text, not spaces.

Smart Tabs - Pressing **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 Model 1310. 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. Com3, 4 and 5 are used to communicate via USB to serial adapters which assign a com # to the USB port.

Simulate

Choose this command to start the Model 1310 simulation using the program you are working on.

Help

Choose this command to open help documentation on Model 1310 SimPoser operation.

Attention

Some of the scale functions listed below cannot be simulated on your computer. You will need to download the program to an indicator for testing purposes.

Functions:
Multiscale inputs
Quartzell
SensorComm
Networks
Memory management
Setpoints

Program Editor Window Buttons

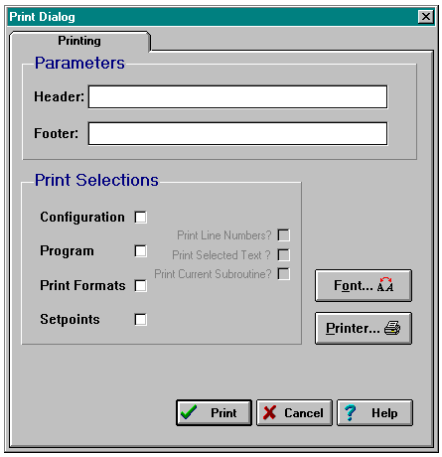
There are several buttons on the button bar of the editor window. Below is a description of each.

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:

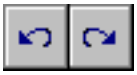


You can insert a header and footer into the printout. You can also choose what information to print: Configuration; Program; Print Formats; and Set Points.

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

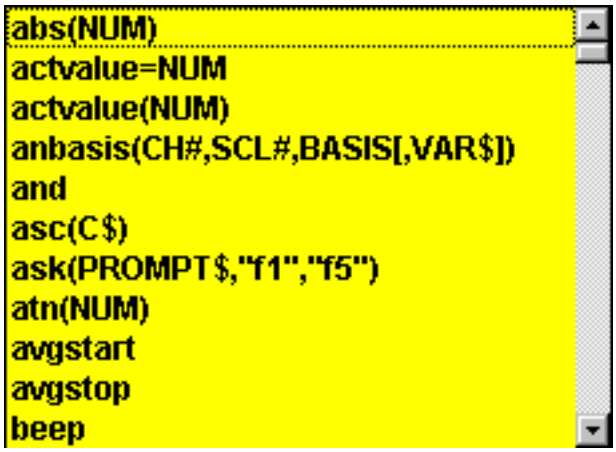
This button is for toggling the event list on and off. Use the elevator button or the arrow keys to scroll through the entire list of events available to you.

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 Model 1310 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

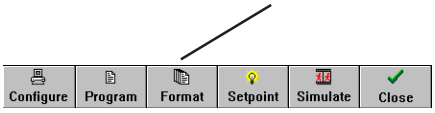
This is a list containing currently used event names and self-created subroutine names that are currently in your program. The Update window on the editor screen will appear or disappear when this key is pressed. By double clicking on an event or subroutine name in this list, your cursor is relocated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

Program Errors Button

This button is for bringing up a program error dialog box. 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, or F6, again or clicking on Program Errors under the Search command at the top of the Program Editor window.

If the line referenced in the error dialog box appears to be OK, click the line before and after it or check the Print Format for misspelled variable names.

Format Button



Ports 13 & 14 pertain to the modem option.

PORT #	*Serial connection*
1, 2, 3, 4	*COMM ports*
13, 14	*Modem card*
51, 52, 53, 54, 55, 56, 57, 58, 59	*Ethernet- Raw Sockets #1-9 for option card in location number ONE.*
61, 62, 63, 64, 65, 66, 67, 68, 69	*Ethernet- Raw Sockets #1-9 for option card in location number TWO*
70, 80	*Ethernet IT*

Please reference the Model 1310 Network Installation *manual (PN 29806-0013) for functionality.*

Ports 70 & 80 for Ethernet -IT: File system which is used by the FTP, HTTP and SMTP Ethernet protocols.

The next button on the toolbar is **Format**. Use this to control the way a particular printer connected to a Model 1310 will print a document or bar code label. The program is capable of 32 output formats.

Click the format button and the format editing window shown in Figure 20 appears.

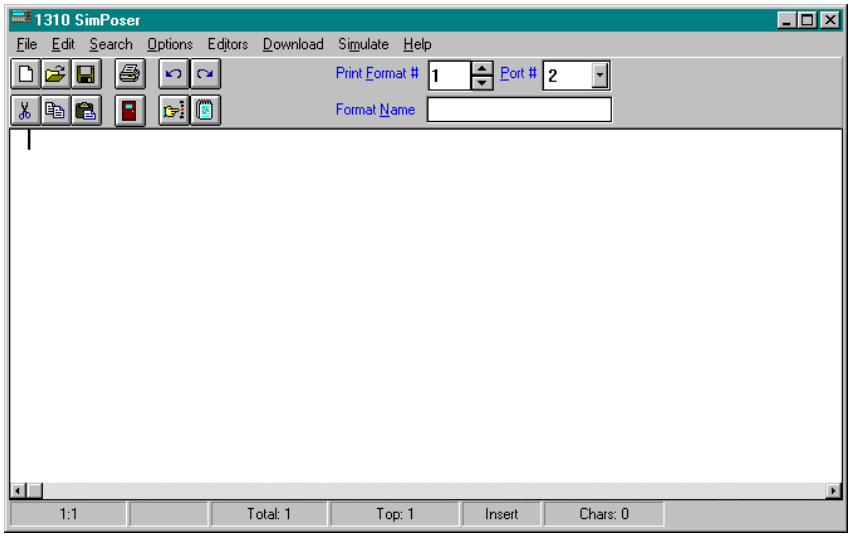
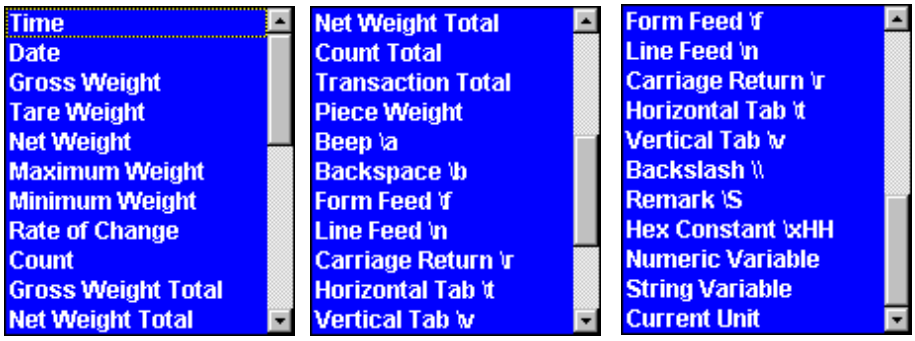


Figure 20
Print format editing window

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 32, 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. This is a case sensitive command or term.*

For printed tickets, you use the editor to lay out a format in logical order. The example in Figure 21 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 21
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 Model 1310 require a different method of placement. Values from the Model 1310 must be surrounded with brackets ({ }) to tell the Print Format that this information will be coming from data stored, generated or collected by the Model 1310. Information such as scale weights, accumulated total, transaction counts, piece weights, part counts, and ID names. These 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, 6.2}

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:

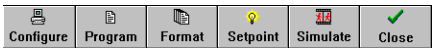
TIME FORMATS

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, 3, or 4 is used in the (n) expression to print DATE$ in the following formats:

Date Formats	Description	MM/DD/YY	DD/MM/YY	YY/MM/DD
0	Numbers with 2-digit year	06-14-99	14-06-99	99-06-14
1)	Spelled Month	Jun 14, 1999	14 Jun, 1999	1999 Jun 14
2)	Numbers with day of week	Mon 06-14-99	Mon 14-06-99	Mon 99-06-14
3)	Spelled Month with day of week	Mon Jun 14, 1999	Mon 14 Jun,1999	1999 Mon Jun 14
4)	Numbers with 4-digit year	06-14-1999	14-06-1999	1999-06-14

Setpoint Button



The next button on the Model 1310 SimPoser toolbar is **Setpoint**. Press this button to configure setpoint operation. Depending on what things you choose, the window will display the parameters you need to set. Figures 22, 23 and 24 show the window as it first appears and as it may appear as you select different parameters.

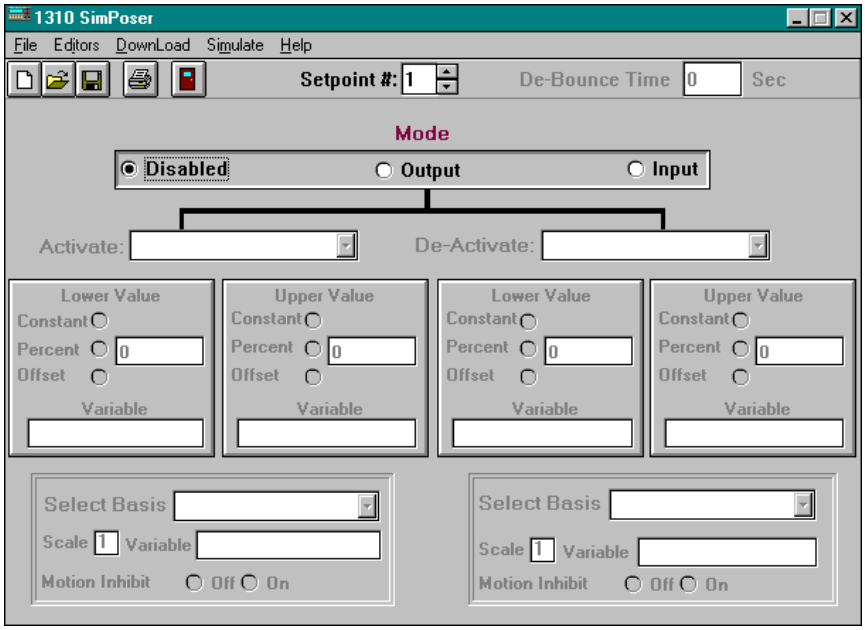


Figure 22
Setpoint window w/setpoints disabled

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.

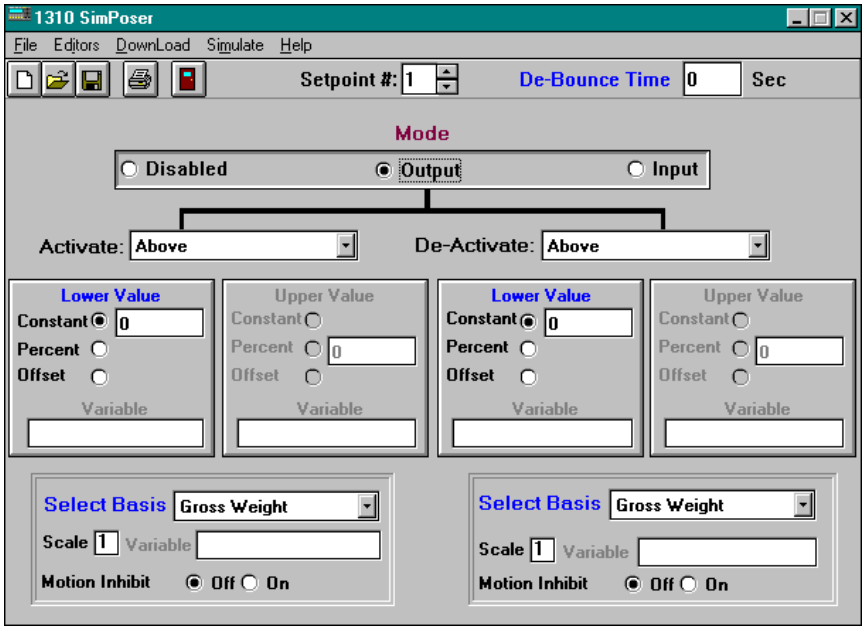


Figure 23
Setpoint window w/Output enabled

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

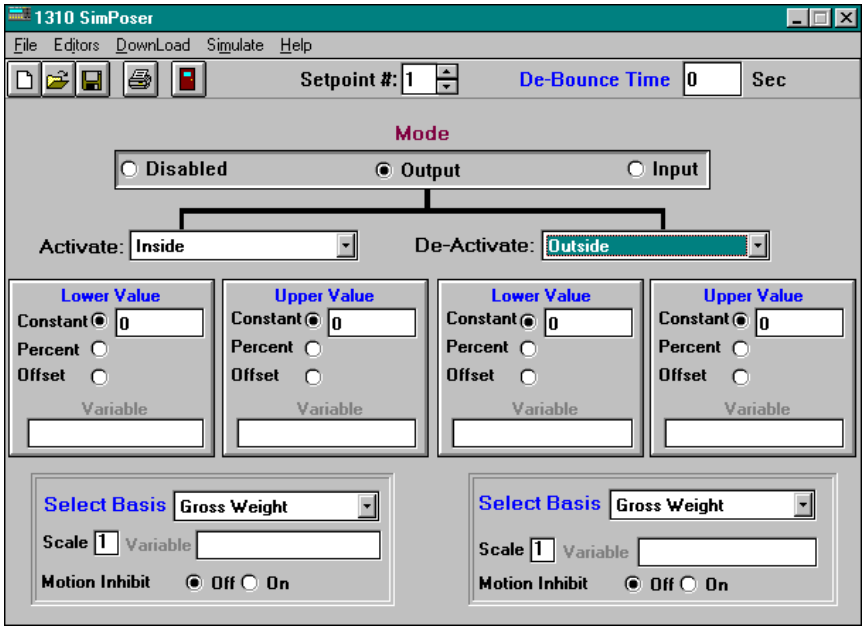


Figure 24
Setpoint window with different parameters

The Setpoint window allows you to define the condition you wish to monitor or detect thus causing the Model 1310 program to trigger a new event (SetPoint Act or Deact).

The Setpoint capabilities of the Model 1310 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 Model 1310 Setpoint System is adept at solving all types of application needs, such as truck scale control, check-weighing systems, conveyor systems, PLC applications or any other situation where external feedback and control is required.

Use the setpoint window to configure up to 64 setpoints in the Model 1310 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.

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 64.

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, continuous 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
- Tare Key
- Select Key
- Print Key
- Units Key
- Zero Key
- Equal To
- Logical AND
- Logical OR
- Logical XOR

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.

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 SETPT OFF.

1. Above/Below/Equal To

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 or minus (if a negative number is used) this offset value in cal units, will control this setpoint.

*You **CAN'T** use structures or arrays for a setpoint variable when using percent and offset conditions. You **MUST** copy the structure or array element into a temporary variable first.*

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
No selections need to be made.

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.

Attention

*You **CAN'T** use structures or arrays for a setpoint variable basis. You **MUST** copy the structure or array element into a temporary variable first.*

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.

5. LOGICAL AND, LOGICAL OR, LOGICAL XOR
The setpoint will activate or deactivate based upon the logical state of the other setpoints.

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.

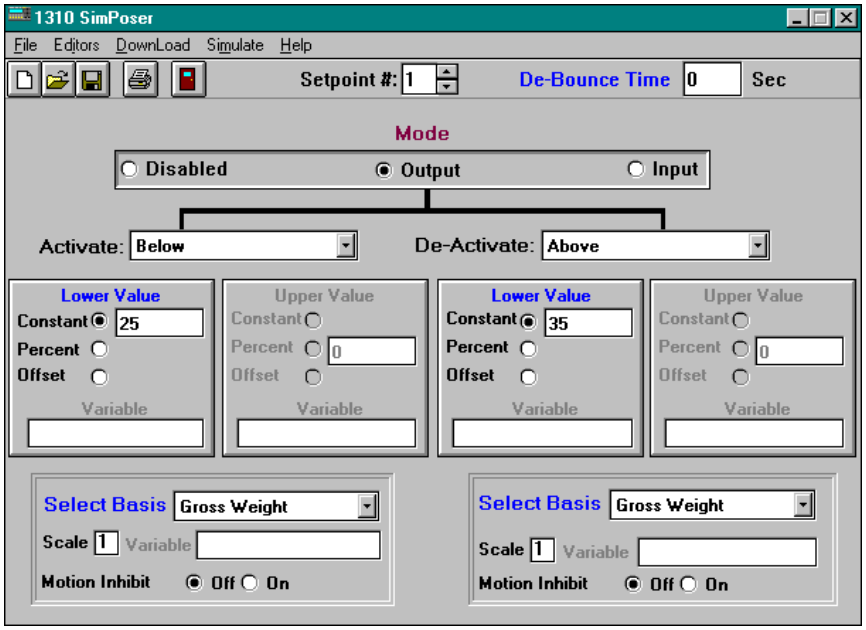


Figure 25
Example 1

In Figure 25, 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:

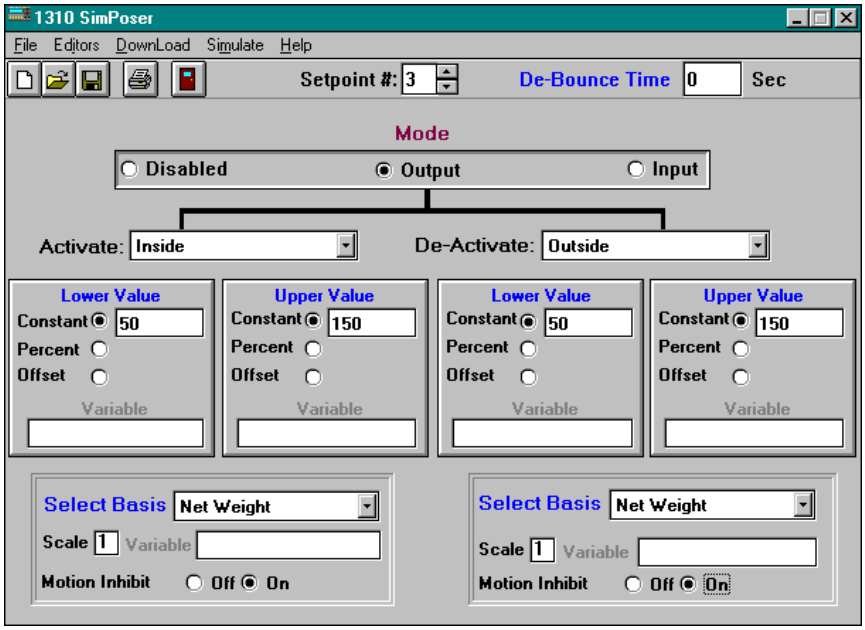


Figure 26
Example 2

Figure 26 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 27) and #2 (see Figure 28).

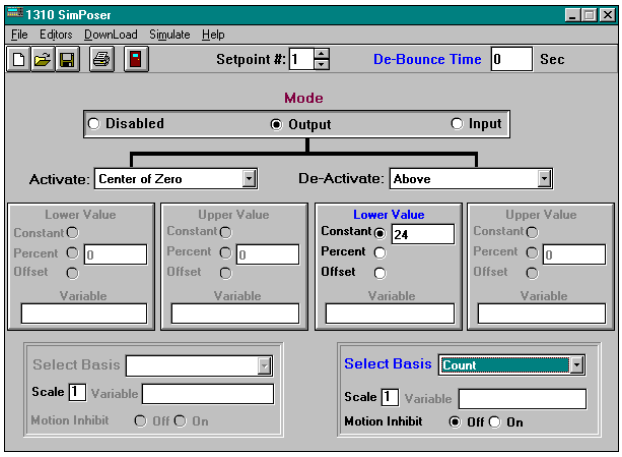


Figure 27
Example 3-Setpoint #1

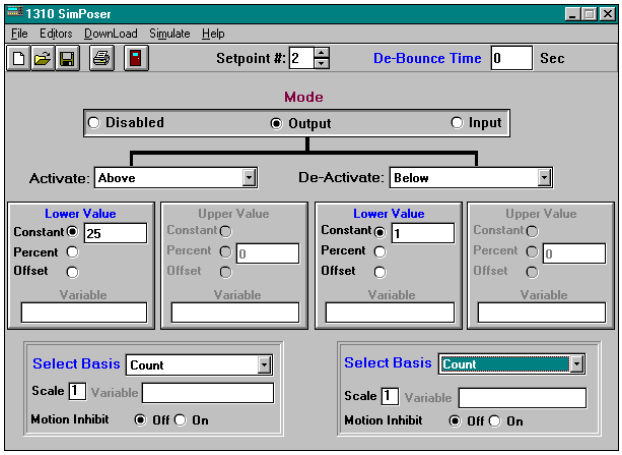


Figure 28
Example 3 - Setpoint #2

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 29 shows two screens using variables. The explanations for these two screens are below.

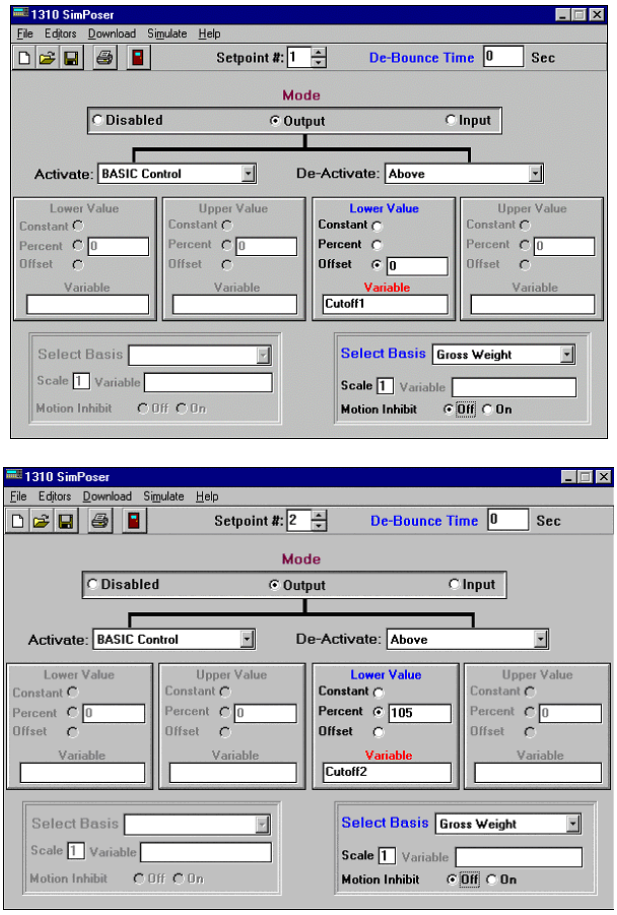


Figure 29
Example 4

In this fourth example we are controlling two setpoint outputs as cutoffs, similar to the way older indicators have worked in the past. The first screen in figure 29 refers to Cutoff1 as the variable name for the control of setpoint#1. The second screen refers to Cutoff2 as the variable name for the 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 basic program instruction (basic control) to start the batch is issued from a key press on the front panel. This key press tells setpoint#1 to turn on.

Setpoint#1 deactivates when the gross weight on scale#1 reaches or exceeds the value the variable Cutoff1 holds.

Setpoint#2 activates when in the basic program the sub routine SETPOINT#1 DEACTIVATE runs. The instruction (basic control) to turn on setpoint#2 is issued and setpoint#2 turns on.

Setpoint#2 deactivates when the gross weight on scale#1 reaches or exceeds the value the variable Cutoff2 holds by 105% of that value.

Example 5

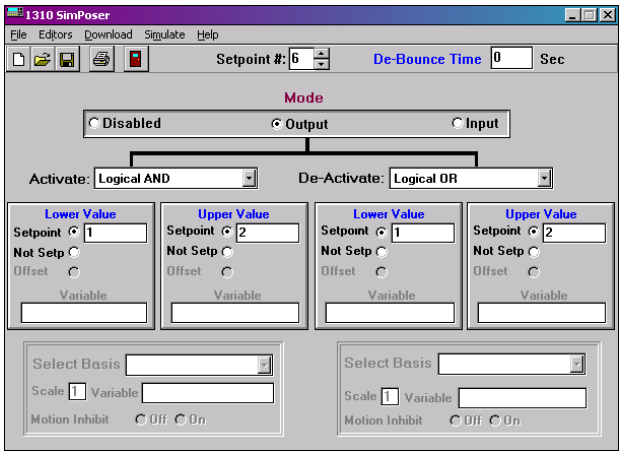
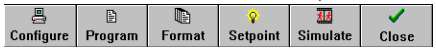


Figure 30
Example 5

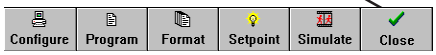
In this example, shown in Figure 30, we use the logical AND and OR conditions. Setpoint #6 will activate when Setpoints 1 AND 2 are on and deactivate when either Setpoint 1 OR 2 is off.

Simulate Button



The next toolbar button is **Simulate**. Click this button to start the Model 1310 simulation. This is the same as clicking the Simulate command on the command line. This was covered earlier in this manual.

Close Button



The last toolbar button is **Close**. Click this to close your Model 1310 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.

Sample Application Program Summaries

These sample application programs can be used as templates to create a custom solution.

Batching	
2spd1ing.310	Two speed single ingredient batching application.
4ingbatc.310	4 Ingredient single speed batching application.
Flow9.310	Demo using ROC to control flow rate(using setpoints) “LB/HR”.
Jogbatch.310	Jog softkey example.
Wi-1106.310	Multi-scale (3-scales), dual cutoffs, single speed.
Checkweighing	
Chkweig2.310	Advanced checkweigher program with set points.
Setchk1.310	Simple checkweigher program.
Counter (Pulse counter)	
cntr.310	Example using pulse counter as data collection.
cntr_scl.310	Example using pulse counter in a batching system.
Counting	
Count5.310	Simple counting scale using the Dribble sample method. Printing –sample size, piece weight, net weight, and count total.
Count6.310	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.
Ethernet	
enet_IT2.310	Demonstrates email, web page generation, file creation, and file deletion and file I/O.
winsock.310	Demonstrates and interfaces to an Ethernet-Raw Sockets server. Example servers may be found in the Tools directory.
InMotion	
EXT_inmo.310	Standard application; uses externally mounted OPTOs
Inmo5.310	Simple conveyor scale application.
INT_inmo.310	Standard application; uses internally mounted OPTOs
Miscellaneous	
Bkltcont.310	Application to demonstrate how to control the backlight and contrast.
Dbase.310	Application to demonstrate the new arrays and structures.
Dispscl.310	Application to demonstrate how to reorder the scales on the display.
Graphscl.310	Application to demonstrate how to use multiple graphs.
Sys_err.310	Example application for using SUB SYSTEM_ERROR and the ERR keyword.
Tare100.310	Application for tare channel store/recall based on an alphanumeric ID.

Microsoft® Word documents explaining these programs are available in the C:\wt\WPI1310\docs folder or the folder you designated during installation.

Printers

Contout.310	Provides continuous output of GROSS weight out port 1.
Enq_cr.310	Remote request from a PC application.
Enquire.310	Remote Enquire recognition and data transfer.
modem.310	Example of how to configure the modem for WT-BASIC mode or Diagnostic mode.
modem1.310	Demonstrate more modem functionality.
Orion.310	Zebra LP2443 and Eltron Orion sample label printer application.
Orion1.310	Zebra LP2443 and Eltron Orion sample label printer application. Examples from Orion printer price sheet.
Rs485.310	RS-485 multi-drop example application.
Tm_295a.310	Epson TM-295 ticket printer example.

Rail (Track Scales)

W-line6.310	Standard Weigh-Line application.

RD (Remote Display)

Rd-125.310	RD-125 sample application.
Rd4000_6.310	RD-4/6000 with 6 digit display sample.
Rd4000_8.310	RD-4/6000 with 8 digit display sample.
Rd4100.310	RD-4100 remote display sample.

Sales Demos

Chkweigh.310	Checkweigher sales demonstration.
Demoapps.310	Demonstrate the versatility and the capability of the Model 1310 for solving truck scale, checkweigher, and printer applications.
Partct.310	Part Counter sales demonstration.
S_demo_8.310	Sales demonstration application, includes standard truck IN/OUT slide show, specifications, 8 scale/8 graph batching system.
S_demo_1.310	Sales demonstration application, includes standard truck IN/OUT slide show, specifications, 1 scale/8 graph batching system.
Slide.310	Slideshow for sales demonstration.
Specs.310	Model 1310 specification slideshow for sales demonstrations.
Tareiso.310	Multi-Channel tare database with ISO-9000 tracking information.
Trkinout.310	Truck scale IN/OUT demonstration.
Wi110.310	WI-110 with 10 tare registers.
Wi110bat.310	WI-110 with dual cutoffs.
Wi120.310	WI-120 emulation sales demonstration.

Standard Application

14835_0d.310	Inbound/Outbound truck scale
14835B0A.310	The standard multiple ID/tare and accumulator application that ships with the 1310 indicator.
16206_0A.310	Dual indicators for Inbound/Outbound truck scale (Primary)
16206_1A.310	Dual indicators for Inbound/Outbound truck scale (Secondary)
14835A0C.310	Traxle truck scale application.

Truck Scale

10544f.310	Axle weigh truck scale.
9459d.310	GTN or 70 foot axle weigh scale.
Freetrk5.310	Inbound/Outbound truck scale application example.
Freetrk6.310	Dual indicator Inbound/Outbound truck scale application example.
Traxle.310	Traxle sample application.

Appendix 1: Display Samples

A scale number will appear on the display if multiple scales are configured. The samples below are shown with lower case text enabled.

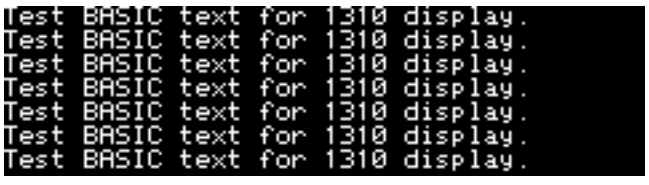
Display Mode #1

Display Mode #2

Display Mode #3

Display Mode #4

Display Mode #5



Display Mode #6

Display Mode #7

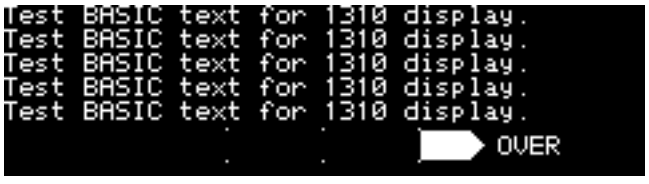
Display Mode #8

Display Mode #9

Display Mode #10

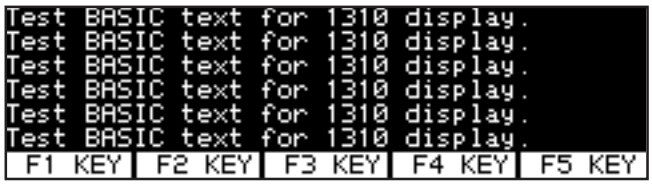
Display Mode #11

Display Mode #12



Display Mode #13

Display Mode #14



Display Mode #15

Display Mode #22

Display Mode #16

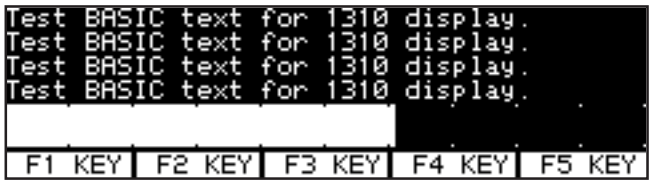
Display Mode #23

Display Mode #17

Display Mode #24

Display Mode #18

Display Mode #25

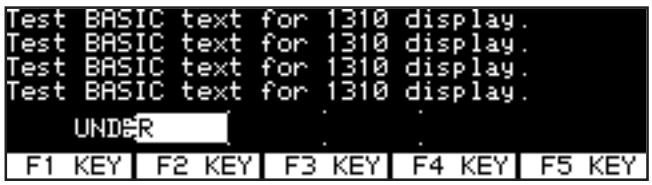


Display Mode #19

Display Mode #26

Display Mode #20

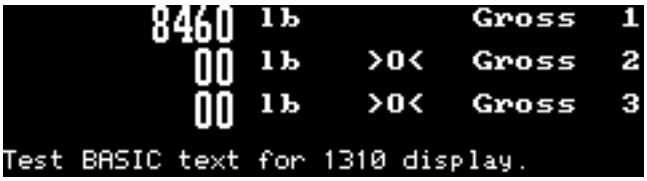
Display Mode #27



Display Mode #21

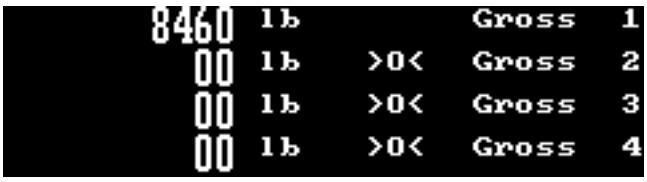
Display Mode #28

Display Mode #29



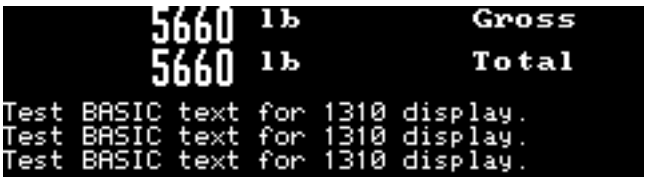
Display Mode #34 w/3 scales enabled

Display Mode #30



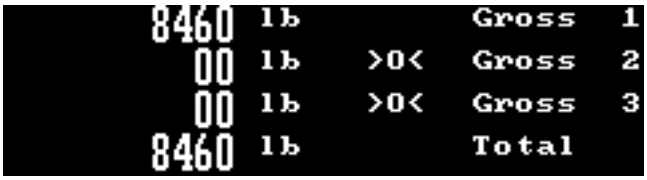
Display Mode #34 w/4 scales enabled

Display Mode #31



Display Mode #35 w/1 scale enabled

Display Mode #32

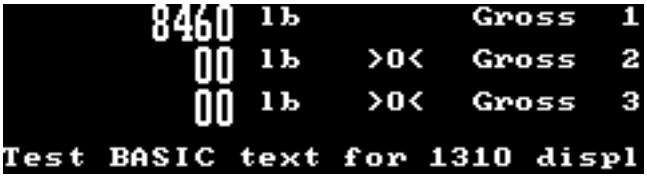


Display Mode #35 w/3 scales enabled

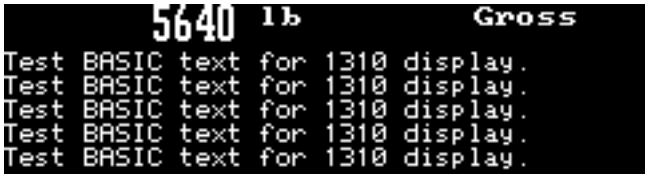
Display Mode #33

Display Mode #36 w/1 scale enabled

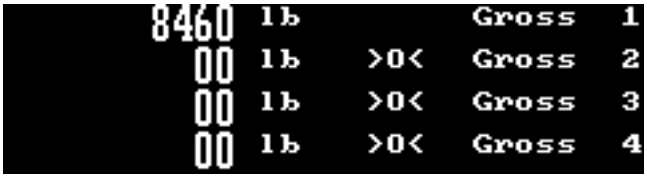
The following are multi-scale displays. If all the lines are not used for scales, they may be available for Basic text.



Display Mode #36 w/3 scales enabled

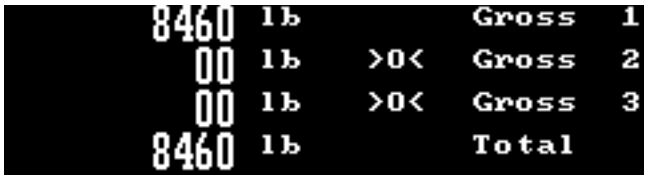


Display Mode #34 w/1 scale enabled



Display Mode #36 w/4 scales enabled

Display Mode #37 w/1 scale enabled



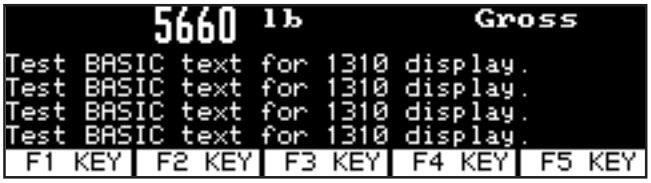
Display Mode #37 w/3 scales enabled

Display Mode #38 w/1 scale enabled

Display Mode #38 w/2 scales enabled

Display Mode #39 w/1 scale enabled

Display Mode #39 w/2 scales enabled



Display Mode #40 w/1 scale enabled

Display Mode #40 w/3 scales enabled

Display Mode #41 w/1 scale enabled

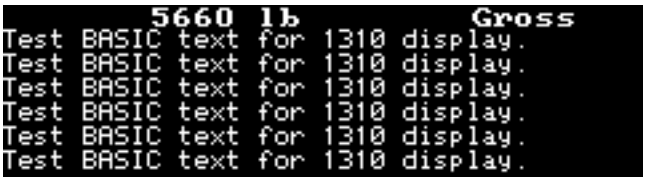
Display Mode #41 w/2 scales enabled

Display Mode #42 w/1 scale enabled

Display Mode #42 w/3 scales enabled

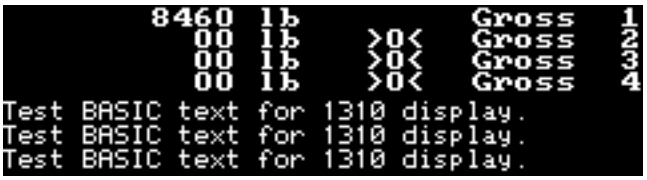
Display Mode #43 w/1 scale enabled

Display Mode #43 w/2 scales enabled



Display Mode #44 w/1 scale enabled

Display Mode #46 w/4 scales enabled

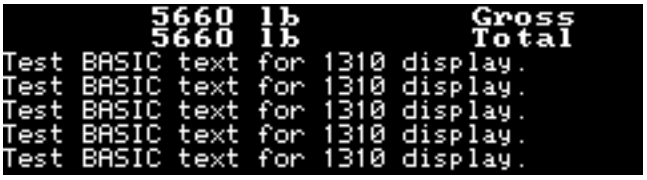


Display Mode #44 w/4 scales enabled

Display Mode #46 w/8 scales enabled

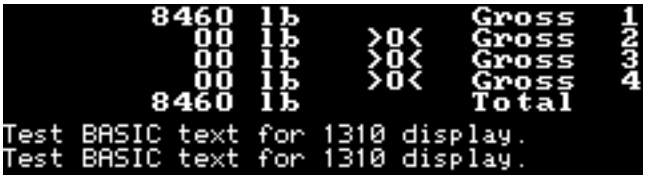
Display Mode #44 w/8 scales enabled

Display Mode #47 w/1 scale enabled



Display Mode #45 w/1 scale enabled

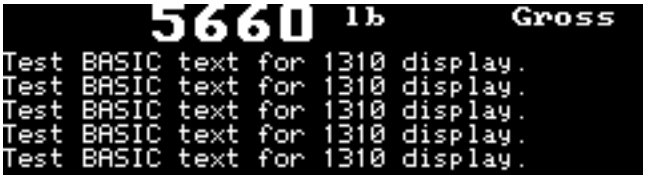
Display Mode #47 w/4 scales enabled



Display Mode #45 w/4 scales enabled

Display Mode #47 w/7 scales enabled

Display Mode #45 w/7 scales enabled



Display Mode #48 w/1 scale enabled

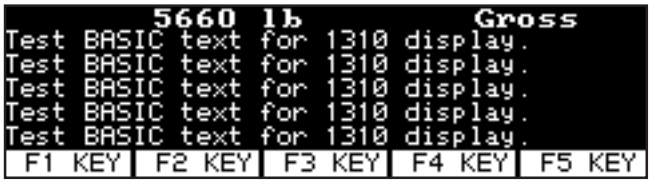
Display Mode #46 w/1 scale enabled

Display Mode #48 w/3 scales enabled

Display Mode #48 w/4 scales enabled

Display Mode #49 w/1 scale enabled

Display Mode #49 w/3 scales enabled



Display Mode #50 w/1 scale enabled

Display Mode #50 w/4 scales enabled

Display Mode #50 w/6 scales enabled

Display Mode #51 w/1 scale enabled

Display Mode #51 w/4 scale enabled

Display Mode #51 w/5 scales enabled

Display Mode #52 w/1 scale enabled

Display Mode #52 w/4 scales enabled

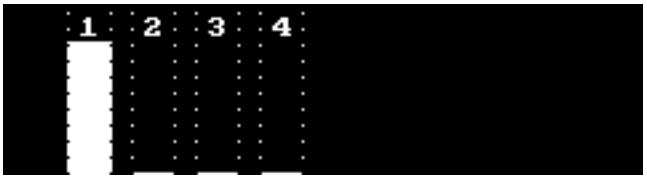
Display Mode #52 w/6 scales enabled

Display Mode #53 w/1 scale enabled

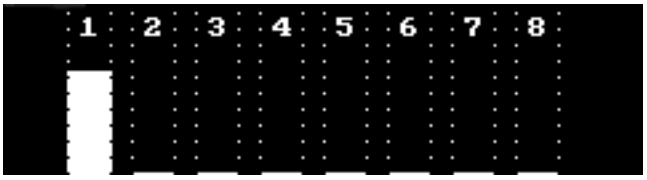
Display Mode #53 w/4 scales enabled

Display Mode #53 w/5 scales enabled

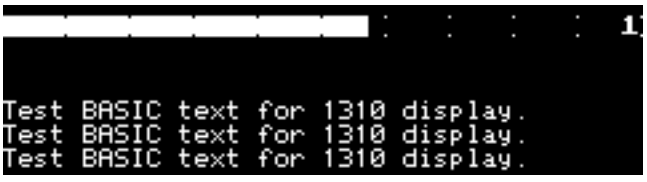
Display Mode #54 w/1 scale enabled



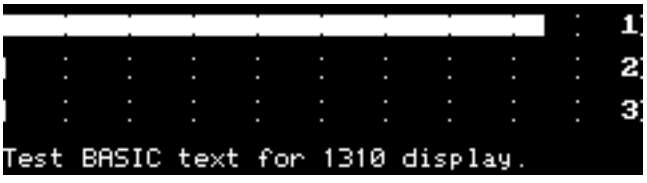
Display Mode #54 w/4 scales enabled



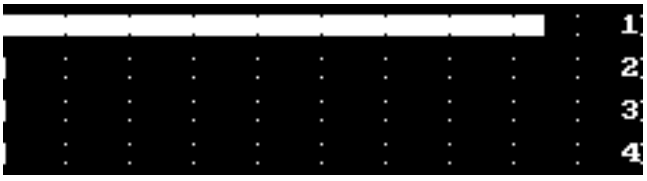
Display Mode #54 w/8 scales enabled



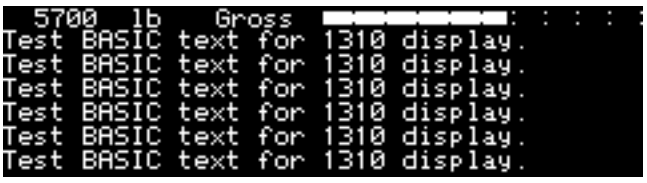
Display Mode #55 w/1 scale enabled



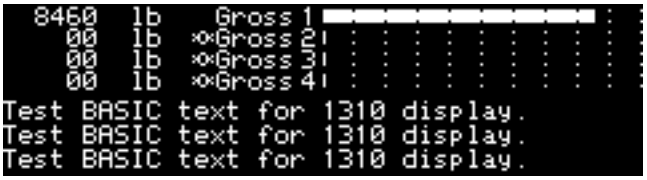
Display Mode #55 w/3 scales enabled



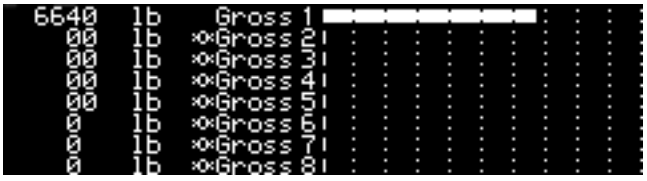
Display Mode #55 w/4 scales enabled



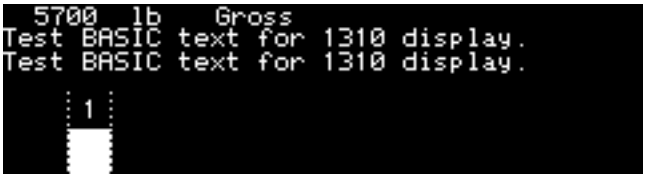
Display Mode #56 w/1 scale enabled



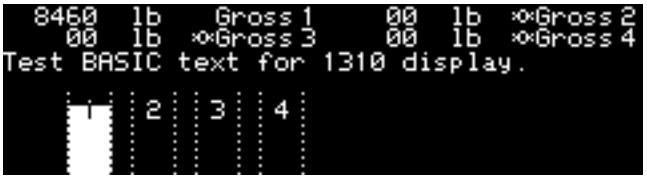
Display Mode #56 w/4 scales enabled



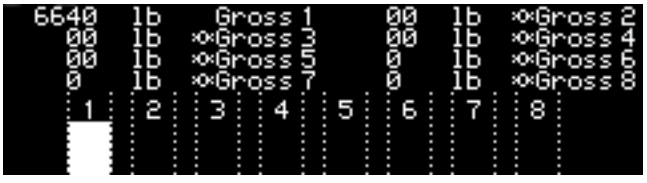
Display Mode #56 w/8 scales enabled



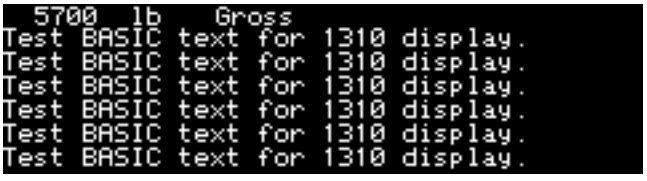
Display Mode #57 w/1 scale enabled



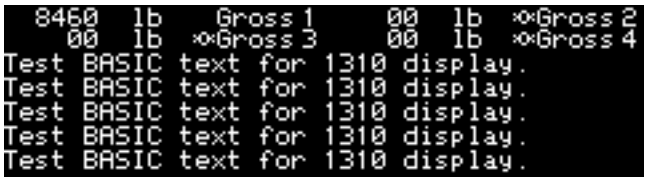
Display Mode #57 w/4 scales enabled



Display Mode #57 w/8 scales enabled

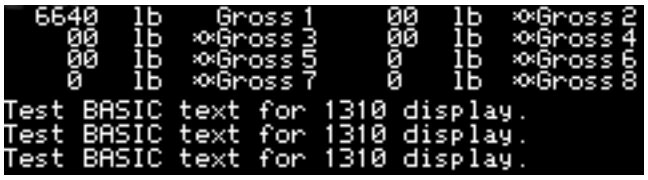


Display Mode #58 w/1 scale enabled

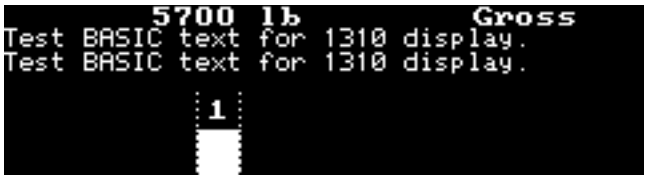


Display Mode #58 w/4 scales enabled

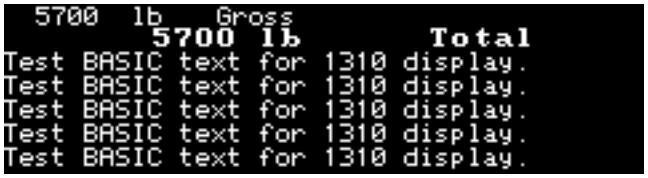
Display Mode #60 w/8 scale enabled



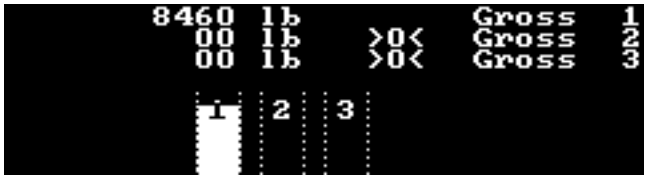
Display Mode #58 w/8 scales enabled



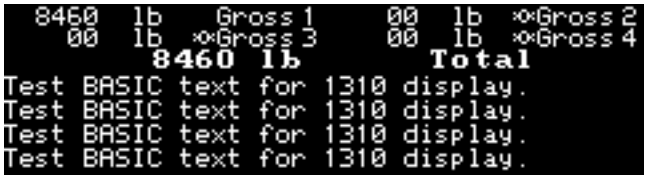
Display Mode #61 w/1 scale enabled



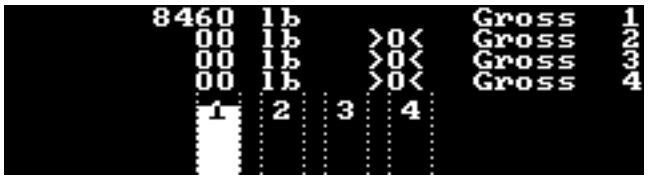
Display Mode #59 w/1 scale enabled



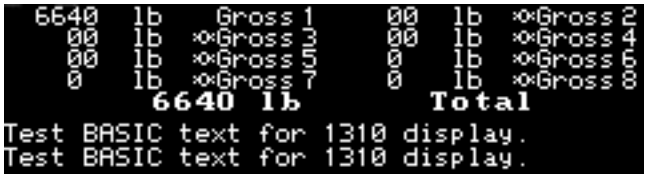
Display Mode #61 w/3 scales enabled



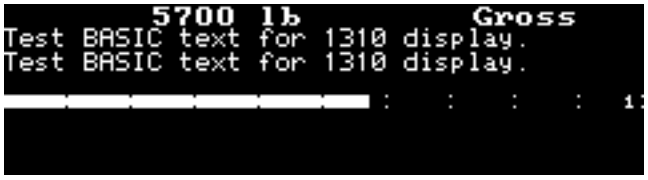
Display Mode #59 w/4 scales enabled



Display Mode #61 w/4 scales enabled

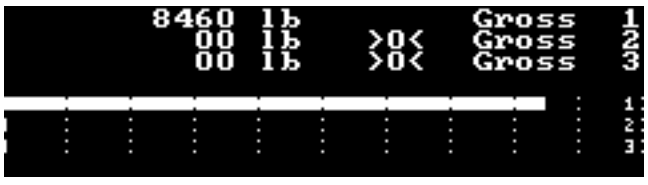


Display Mode #59 w/8 scales enabled



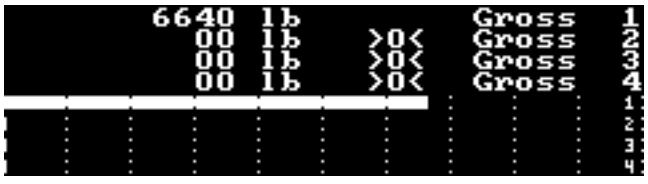
Display Mode #62 w/1 scale enabled

Display Mode #60 w/1 scale enabled



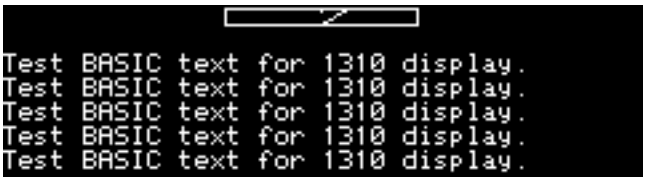
Display Mode #62 w/3 scales enabled

Display Mode #60 w/4 scales enabled

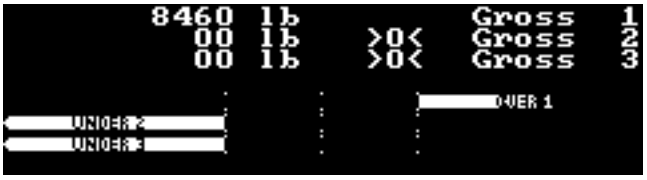


Display Mode #62 w/4 scales enabled

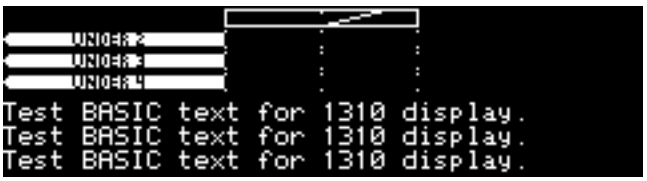
Display Mode #63 w/1 scale enabled



Display Mode #66 w/1 scale enabled



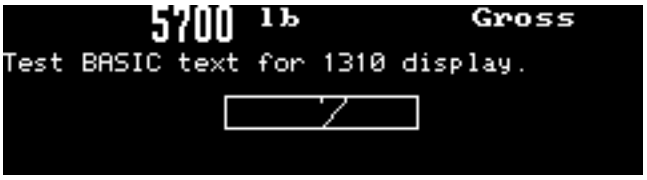
Display Mode #63 w/3 scales enabled



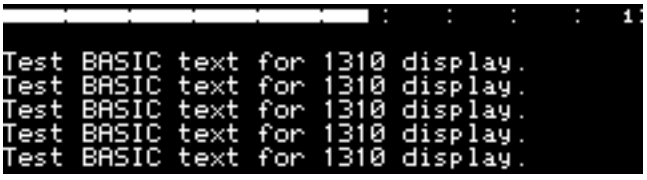
Display Mode #66 w/4 scales enabled

Display Mode #63 w/4 scales enabled

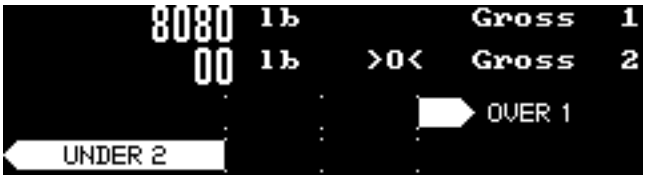
Display Mode #66 w/8 scales enabled



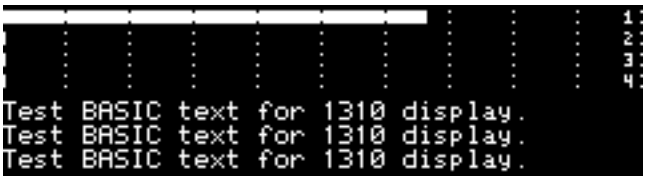
Display Mode #64 w/1 scale enabled



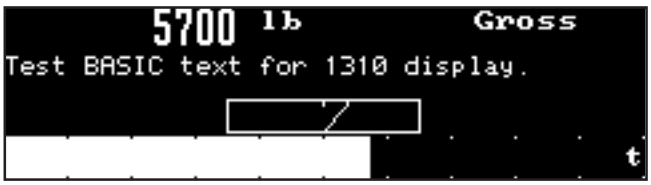
Display Mode #67 w/1 scale enabled



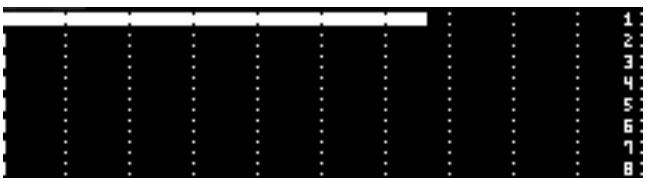
Display Mode #64 w/2 scales enabled



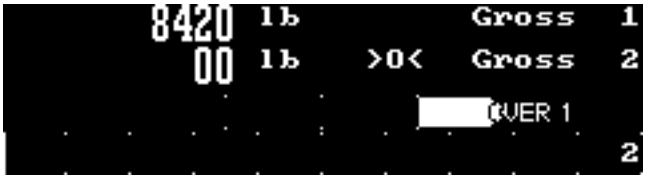
Display Mode #67 w/4 scales enabled



Display Mode #65 w/1 scale enabled



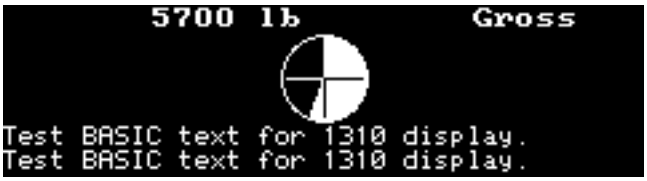
Display Mode #67 w/8 scales enabled



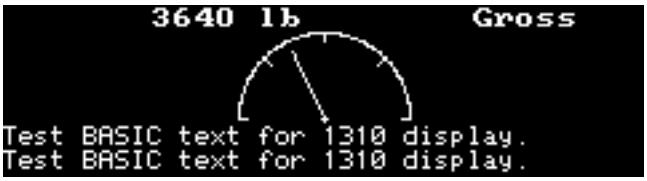
Display Mode #65 w/2 scales enabled

The following displays are all single scale displays. Checkweigher examples may show Over, Under, or Accept conditions.

The scale # appears in some of these examples because more than one scale is configured.



Display Mode #68



Display Mode #75

Display Mode #69

Display Mode #76

Display Mode #70

Display Mode #77

Display Mode #71

Display Mode #78

Display Mode #72

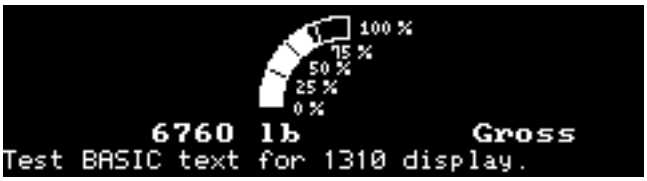
Display Mode #79

Display Mode #73

Display Mode #80

Display Mode #74

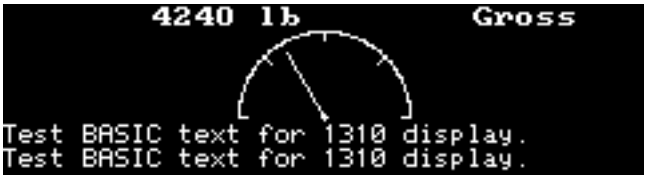
Display Mode #81



Display Mode #82

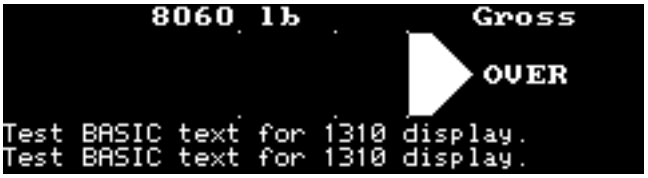
Display Mode #89 in an UNDER condition

Display Mode #83

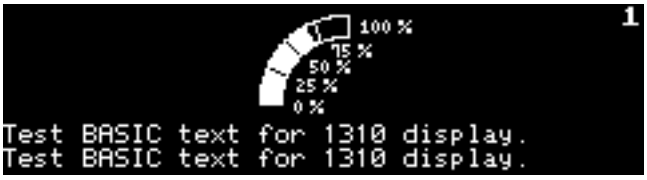


Display Mode #89 in the ACCEPT range

Display Mode #84

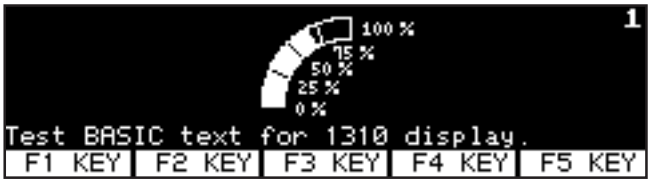


Display Mode #89 in an OVER condition



Display Mode #85

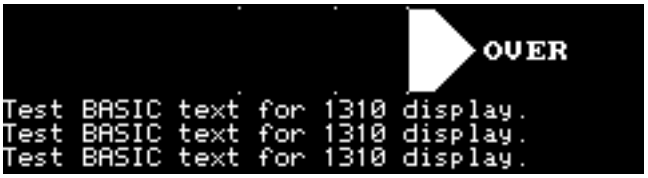
Display Mode #90 in an OVER condition



Display Mode #86

Display Mode #91 in an OVER condition

Display Mode #87



Display Mode #92 in an OVER condition

Display Mode #88

Display Mode #93 in an OVER condition

Display Mode #94 in an OVER condition

Display Mode #95 in an OVER condition

1310 Display Descriptions

Below are the descriptions of each display mode.

While many of the multi-scale display modes can display up to 8 scale weight values, their unique features (Total display, BASIC text, Softkeys) will be lost when exceeding the "**# of Scales**" value listed.

*# of Display #	*BASIC Scales	Text	*Softkeys	Graph	Weight Value Font Size	*Total	Description
1	1	none	No	No	3 x 8	No	1 scale
2	1	none	Yes	No	3 x 6	No	1 scale
3	1	none	Yes	Hor. bar	2 x 4	No	1 scale
4	1	none	Yes	Hor. bar	3 x 4	No	1 scale
5	1	none	Yes	Checkweigh	2 x 4	No	1 scale
6	1	Small	No	No	none	No	1 scale
7	1	Small	No	No	2 x 2	No	1 scale
8	1	Small	No	No	2 x 4	No	1 scale
9	1	Small	No	No	3 x 4	No	1 scale
10	1	Small	No	No	3 x 6	No	1 scale
11	1	Small	No	Hor. bar	2 x 4	No	1 scale
12	1	Small	No	Hor. bar	3 x 4	No	1 scale
13	1	Small	No	Checkweigh	none	No	1 scale
14	1	Small	No	Checkweigh	2 x 4	No	1 scale
15	1	Small	Yes	No	none	No	1 scale
16	1	Small	Yes	No	2 x 4	No	1 scale
17	1	Small	Yes	No	3 x 4	No	1 scale
18	1	Small	Yes	No	3 x 6	No	1 scale
19	1	Small	Yes	Hor. bar	none	No	1 scale
20	1	Small	Yes	Hor. bar	2 x 2	No	1 scale
21	1	Small	Yes	Checkweigh	none	No	1 scale
22	1	Small	No	No	none	No	1 scale
23	1	Large	No	No	2 x 4	No	1 scale
24	1	Large	No	No	3 x 6	No	1 scale
25	1	Large	No	Hor. bar	none	No	1 scale
26	1	Large	No	Checkweigh	none	No	1 scale
27	1	Large	Yes	No	none	No	1 scale
28	1	Large	Yes	Hor. bar	none	No	1 scale
29	1	Large	Yes	No	2 x 2	No	1 scale
30	1	Large	Yes	No	2 x 4	No	1 scale
31	1	Large	Yes	No	3 x 6	No	1 scale
32	1	Large	Yes	Hor. bar	2 x 2	No	1 scale
33	1	Large	Yes	Checkweigh	none	No	1 scale
34	4	Small	No	No	1 x 2	No	4 Scale multi-scale mode (Small text available with fewer scales)
35	3	Small	No	No	1 x 2	Yes	3 Scale multi-scale mode w/Total (Small text available with fewer scales)
36	4	Large	No	No	1 x 2	No	4 Scale multi-scale mode (Large text available with fewer scales)
37	3	Large	No	No	1 x 2	Yes	3 Scale multi-scale mode w/Total (Large text available with fewer scales)
38	2	Small	No	No	2 x 3	No	2 Scale multi-scale mode w/Small Text
39	2	Large	No	No	2 x 3	No	2 Scale multi-scale mode w/Large Text
40	3	Small	Yes	No	1 x 2	No	3 Scale multi-scale mode w/Softkeys (Small text available with fewer scales)
41	2	Small	Yes	No	1 x 2	Yes	2 Scale multi-scale mode w/Total w/Softkeys (Small text available with fewer scales)

*# of Display #	*BASIC Scales	Text	*Softkeys	Graph	Weight Value Font Size	*Total	Description
42	3	Large	Yes	No	1 x 2	No	3 Scale multi-scale mode w/Softkeys (Large text available with fewer scales)
43	2	Large	Yes	No	1 x 2	Yes	2 Scale multi-scale mode w/Total w/Softkeys (Large text available with fewer scales)
44	8	Small	No	No	1 x 1	No	8 Scale multi-scale mode (Small text available with fewer scales)
45	7	Small	No	No	1 x 1	Yes	7 Scale multi-scale mode w/Total (Small text available with fewer scales)
46	8	Large	No	No	1 x 1	No	8 Scale multi-scale mode (Large text available with fewer scales)
47	7	Large	No	No	1 x 1	Yes	7 Scale multi-scale mode w/Total (Large text available with fewer scales)
48	4	Small	No	No	2 x 2	No	4 Scale multi-scale mode (Small text available with fewer scales)
49	4	Large	No	No	2 x 2	No	4 Scale multi-scale mode (Large text available with fewer scales)
50	6	Small	Yes	No	1 x 1	No	6 Scale multi-scale mode w/Softkeys (Small text available with fewer scales)
51	5	Small	Yes	No	1 x 1	Yes	5 Scale multi-scale mode w/Total w/Softkeys (Small text available with fewer scales)
52	6	Large	Yes	No	1 x 1	No	6 Scale multi-scale mode w/Softkeys (Large text available with fewer scales)
53	5	Large	Yes	No	1 x 1	Yes	5 Scale multi-scale mode w/Total w/Softkeys (Large text available with fewer scales)
54	8	small	No	Vert. bars	None	No	8 Scale Vertical bar graphs
55	4	small	No	Hor. bars	None	No	4 Scale Horizontal bar graphs
56	8	small	No	Hor. bars	small (side/side)	No	8 Scale multi-scale mode w/Horizontal bar graphs (Small text available with fewer scales)
57	8	small	No	Vert. bars	small (side/side)	No	8 Scale multi-scale mode w/Vertical bar graphs (Small text available with fewer scales)
58	8	small	No	No	small (side/side)	No	8 Scale multi-scale mode w/Small Text
59	8	small	No	No	small (side/side)	Yes	8 Scale multi-scale mode w/Total (in 1 x 1 font) w/Small Text
60	8	small	No	No	small (side/side)	Yes	8 Scale multi-scale mode w/Total (in 1 x 1 font) w/Small Text w/Softkeys
61	4	small	No	Vert. bars	1 x 1	No	4 Scale multi-scale mode w/Vertical bar graphs (Small text available with fewer than 3 scales)
62	4	small	No	Hor. bars	1 x 1	No	4 Scale multi-scale mode w/Horizontal bar graphs (Small text available with fewer than 3 scales)
63	4	small	No	Checkweigh	1 x 1	No	4 Scale multi-scale mode w/Checkweigh graphs (Small text available with fewer than 3 scales)
64	2	small	No	Checkweigh	1 x 2	No	2 Scale multi-scale mode w/Checkweigh graphs (Small text available with 1scale)
65	2	small	No	Checkweigh & bar	1 x 2	No	2 Scale multi-scale mode w/ 1 Checkweigh & 1 bar graph (Small text available with 1scale)
66	8	small	No	Checkweigh	None	No	8 Scale Checkweigh graphs (Small text available with fewer than 7 scales)
67	8	small	No	Hor. bars	None	No	8 Scale Horizontal bar graphs (Small text available with fewer than 7 scales)
68	1	small	No	Pie Chart	1 x 1	No	Single Scale mode w/Pie Chart graph w/ 2 lines of small text
69	1	Large	No	Pie Chart	1 x 1	No	Single Scale mode w/Pie Chart graph w/ 2 lines of Large text
70	1	small	Yes	Pie Chart	1 x 1	No	Single Scale mode w/Pie Chart graph w/ 1 line of small text w/Softkeys
71	1	small	No	Pie Chart	None	No	Single Pie Chart graph w/ 3 lines of small text
72	1	small	Yes	Pie Chart	None	No	Single Pie Chart graph w/ 2 lines of small text w/Softkeys
73	1	Large	No	Pie Chart	None	No	Single Pie Chart graph w/ 3 lines of Large text
74	1	Large	No	Pie Chart	None	No	Single Pie Chart graph w/ 2 lines of Large text w/Softkeys
75	1	small	No	Meter Gauge	1 x 1	No	Single Scale mode w/Meter Gauge graph w/ 2 lines of small text
76	1	Large	No	Meter Gauge	1 x 1	No	Single Scale mode w/Meter Gauge graph w/ 2 lines of Large text
77	1	small	Yes	Meter Gauge	1 x 1	No	Single Scale mode w/Meter Gauge graph w/ 1 line of small text w/Softkeys
78	1	small	No	Meter Gauge	None	No	Single Meter Gauge graph w/ 3 lines of small text

*# of Display #	*BASIC Scales	Text	*Softkeys	Graph	Weight Value Font Size	*Total	Description
79	1	small	Yes	Meter Gauge	None	No	Single Meter Gauge graph w/ 2 lines of small text w/Softkeys
80	1	Large	No	Meter Gauge	None	No	Single Meter Gauge graph w/ 3 lines of Large text
81	1	Large	No	Meter Gauge	None	No	Single Meter Gauge graph w/ 2 lines of Large text w/Softkeys
82	1	small	No	Curved bar	1 x 1	No	Single Scale mode w/Curved bar graph w/ 1 line of small text
83	1	Large	No	Curved bar	1 x 1	No	Single Scale mode w/Curved bar graph w/ 1 line of Large text
84	1	None	Yes	Curved bar	1 x 1	No	Single Scale mode w/Curved bar graph w/ Softkeys
85	1	small	No	Curved bar	None	No	Single Curved bar graph w/ 2 lines of small text
86	1	small	Yes	Curved bar	None	No	Single Curved bar graph w/ 1 line of small text w/Softkeys
87	1	Large	No	Curved bar	None	No	Single Curved bar graph w/ 2 lines of Large text
88	1	Large	No	Curved bar	None	No	Single Curved bar graph w/ 1 line of Large text w/Softkeys
89	1	small	No	L. Checkweigh	1 x 1	No	Single Scale mode w/Large Checkweigh graph w/ 2 lines of small text
90	1	Large	No	L. Checkweigh	1 x 1	No	Single Scale mode w/Large Checkweigh graph w/ 2 lines of Large text
91	1	small	Yes	L. Checkweigh	1 x 1	No	Single Scale mode w/Large Checkweigh graph w/ 1 line of small text w/Softkeys
92	1	small	No	L. Checkweigh	None	No	Single Large Checkweigh graph w/ 3 lines of small text
93	1	small	Yes	L. Checkweigh	None	No	Single Large Checkweigh graph w/ 2 lines of small text w/Softkeys
94	1	Large	No	L. Checkweigh	None	No	Single Large Checkweigh graph w/ 3 lines of Large text
95	1	Large	No	L. Checkweigh	None	No	Single Large Checkweigh graph w/ 2 lines of Large text w/Softkeys

Appendix 2: WT-BASIC Interpreter Command Set

WT-BASIC

*If you hold in the **CLEAR** key when powering up the Model 1310, the WT-BASIC program which is currently resident in the indicator will be temporarily disabled until the next power up.*

This is used when trouble-shooting to eliminate the BASIC program as the source of a problem or to disable continuous output so you can download a new program.

Attention

Some of the scale functions listed below cannot be simulated on your computer. You will need to download the program to an indicator for testing purposes.

Functions:
Multiscale inputs
Quartzell
SensorComm
Networks
Memory management
Setpoints

The following pages contain the WT-BASIC interpreter command set you use to create programming for the Model 1310.

This command set goes beyond the normal BASIC language by adding many commands found only in this expanded WT-BASIC language. This makes programming the Model 1310 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 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 Subroutines

Anytime one of the following events is triggered in the instrument, program execution is transferred to the corresponding sub procedure (if the subroutine procedure exists). If more than one event is triggered, they are executed in the order they were tripped. The syntax for a subroutine is identified by the word “SUB” followed by the name of the event subroutine. The subroutine is ended by the command “END SUB”

Some events have default actions that are executed when the event is triggered. The default action is not executed if a subroutine exists in the BASIC program. An example is ZERO_KEY. If the subroutine ZERO_KEY is in the BASIC program the code there is executed instead of the firmware level ZERO function. i.e.:

```
SUB  ZERO_KEY
   x=4
   y=3
   PRINT X,Y
END SUB
```

ACCUM_ABORT
This event is activated if the DOACCUM command is not successfully processed.

ACCUM_OPER
This event is activated if the DOACCUM command is successfully processed.

CLEAR_KEY
This event is activated whenever the CLEAR key is pressed.

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 primary receive line.

COM3_MESSAGE
Same as COM1_MESSAGE only for Port #3 primary receive line.

COM4_MESSAGE
Same as COM1_MESSAGE only for Port #4 primary receive line.

ENTER_KEY
An event is activated whenever the ENTER key is pressed.

When defining this subroutine the key buffer must be emptied using either INKEY$ or an INPUT statement. If the key buffer is not cleared, the indicator may hang.

ENTRY_KEY
This event is activated whenever any key on the remote keyboard is pressed (except function keys).

ESC_KEY
An event is activated whenever the ESCAPE key is pressed.

Fx_KEY
For softkeys. These events queue when the front panel key is pressed or an external keyboard key is pressed. (x; 1-5)

F6 to F10
Only available on an external keyboard or in the simulator. (x; 6-10)

Attention

When defining this subroutine the key buffer must be emptied using either INKEY$ or an INPUT statement. If the key buffer is not cleared, the indicator may hang.

NUMERIC_KEY
This event is activated whenever any numeric key (0-9) is pressed.

PRINT_ABORT
This event is activated if the DOPRINT command is not successfully processed.

PRINT_KEY
This event will be activated if the **PRINT** key is pressed on the Model 1310.

PRINT_OPER
This event is activated if the DOPRINT command is successfully processed.

SELECT_KEY
This event will be activated if the **SELECT** key is pressed on the Model 1310.

SELECT_OPER
This event is activated if the DOSELECT command is successfully processed.

SETPTx_ACT
These events queue when the SetPoint turns on. There is one subroutine for each setpoint. (x; 1-64)

SETPTx_DEACT
These events queue when the SetPoint turns off. There is one subroutine for each setpoint. (x; 1-64)

SYSTEM_ERROR
Whenever a BASIC error occurs, this event is queued and the error that triggered this event is held in ERR. See also ERR.

SYSTEM_SETUP
This event is queued by first issuing the setpwd keyword. The programmer should then hold in the **ESCAPE** key for five 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$="135"
setpwd(1,word$)
END SUB
```

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

SYSTEM_STARTUP
If a SYSTEM_STARTUP procedure exists in the WT-BASIC program, it always gets executed at the time of startup or upon exiting CAL or CONFIG menus.

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

SYSTEM_ TIMER2
This event is activated on interval determined by Settimer command. This will generate one queue in the event buffer.

SYSTEM_ TIMER3
This event is activated on interval determined by Settimer command. This will generate one queue in the event buffer.

SYSTEM_ TIMER4
This event is activated on interval determined by Settimer command. This will generate one queue in the event buffer.

TARE_ABORT
This event is activated if the DOTARE command is not successfully processed.

*Defining this blocks the normal default operation from occurring when you press a key on the front panel of the Model 1310. Use the **doxxx** commands to make the regular function work. See **doxxx**.*

TARE_KEY
This event is activated if the TARE key is pressed.

TARE_OPER
This event is activated if the DOTARE command is successfully processed.

Default: Changes the display value to NET if the display value is Gross and Tare does not equal 0 or changes the display value to GROSS if the display value is NET and Tare = 0.

UNITS_KEY
This event will be activated if the UNITS key is pressed on the Model 1310.

UNITS_OPER
This event is activated if the DOUNITS command is successfully processed.

ZERO_ABORT
This event is activated if the DOZERO command is not successfully processed.

*Defining this blocks the normal default operation from occurring when you press a key on the front panel of the Model 1310. Use the **doxxx** commands to make the regular function work. See **doxxx**.*

ZERO_KEY
This event is activated if the ZERO key is pressed.

ZERO_OPER
This event is activated if the DOZERO command is successfully processed.

BASE_OPER
If the dobase keyword was successful, this event is queued.

NETWORK_ABORT
If the initialization procedure fails on power up, this event is queued. NetErr will contain the error corresponding to the failure.

*Check the **neterr** keyword for the status change. **clearerr** can be used to reset the **neterr** keyword so multiple status changes can be viewed.*

NETWORK_OPER
If the initialization procedure succeeds on power up, this event it queued. NetErr will be set to 0.

NETWORK_STATUS
If the network status changes, this event is queued.

SOCK*x*_MESSAGE
This event will be queued when the EOM character is received over the Ethernet network. The *x* represents the card number that has been selected. (x=1 or 2)

NET*x*_OFFLINE
When the network cable is damaged or disconnected, the network goes offline and this event will be queued. The *x* represents the card number that has been selected. (x=1 or 2)

NET*x*_ONLINE
When the network cable is connected, the network goes on-line and this event will be queued. The *x* represents the card number that has been selected. (x=1 or 2)

SOCK*x*_OFFLINE
When the server is shut down, or the connection is lost to the server. The socket goes offline, and this event will be queued. The *x* represents the card number that has been selected. (x=1 or 2)

SOCK*x*_ONLINE
When the server makes a connection to the specified socket, the network goes on-line, and this event will be queued. The *x* represents the card number that has been selected. (x=1 or 2)

MODEM_MESSAGE
Similar to COMX_MESSAGE. When a packet is received via the modem, and the End of Message character is received, this event will be queued. Use ports 13 or 14 for the modem. GETCOM$ will retrieve the message.

MODEM_STATUS
This event will be queued when the modem changes status (Ex: OK, Connected, Busy or No Carrier). See Modem keyword to retrieve status messages.

SCOMM_ERROR
When an error occurs on a SensorComm system, this event will be queued. See SCOMMERR keyword for error messages.

The following 4 events are Traxle specific system events:

STOP1_OPER
This system event is queued if the Traxle(1) keyword was successful.

STOP1_ABORT
This system event is queued if the Traxle(1) keyword failed.

STOP2_OPER
This system event is queued if the Traxle(2) keyword was successful.

STOP2_ABORT
This system event is queued if the Traxle(2) keyword failed.

DOWNLOAD_OPER
If the data download operation is completed (data sent to the 1310) successfully, this event is queued.

DOWNLOAD_ABORT
If the download is unsuccessful/terminated prematurely, this event is queued.

UPLOAD_OPER
If the data upload operation is completed (data sent from the 1310) successfully, this event is queued.

UPLOAD_ABORT
If the upload is unsuccessful/terminated prematurely, this event is queued.

SensorComm/Ghost

GHOST_OPER
This event is queued when the Ghost feature is engaged.

Email

SYSTEM_EMAIL
This event is queued if the Email or Error configuration is enabled and a system error occurs.

Variables

Weigh-Tronix strongly recommends the use of the MUSTDIM command in the SYSTEM_STARTUP event subroutine.

The user defined variables can be numeric variables or string variables. The variables can be defined by just referencing them. The type of the variable defined is dependant on the last character of the name of the variable. A variable can be defined also by the command "**DIM *VARIABLE NAME***". For purposes of debugging, the command **MUSTDIM** can be placed in the WT-BASIC program forcing all variables to be defined exclusively through the dimension command.

Numeric Variable

A numeric variable represents a group of numbers only that mathematical functions may be performed on.
A numeric variable is assigned using "**=**" i.e. **X = 42.**
All variables without a special dimensioning character or with a "**#**" character on the end becomes **a 15 digit, double precision, floating point**.
By using a "**%**" character, the variable is dimensioned as a regular **integer**.
An "**!**" character dimensions the variable as a **single precision floating point**.
The "**&**" character dimensions the variable as **Long Integer**.

Numeric array

Use Structures for multi-dimensional arrays.
Numeric Variable Values:
Double floating:
$2.2E^{-308}$ to $1.7E^{308}$

Single floating:
$1.0E^{-37}$ to $1.0E^{37}$

Integer:
-32767 to 32767

Long Integer:
-2,147,483,647 to 2,147,483,647

A numeric array is like a list of numeric variables. Unlike variables, arrays cannot be defined by simply referencing them even if a **MUSTDIM** command is not present in the WT-BASIC program. They must be defined using a **DIM** or **DIM1** statement before they are referenced. The size of the array is specified in the **DIM/DIM1** command by using brackets. Arrays can be 15 digit double precision floating point, single precision floating point, integer, or long integers depending on the special dimensioning character at the end of the array name. An array is assigned using "=" much like variable assignments. However the array index must be included in the assignment statement. Array indexes start at zero and go to (array size – 1).

EXAMPLE:

```
DIM a[10]    -defines the size of array "a" as10 elements double prec.
a[0] = 20    -assigns 20 to the first element of array a.
a[9] = 21    -assigns 21 to the last element of array a.
```

DIM1 B%[100] This statement defines the size of array "B" as100 elements of integers. Using DIM1 instead of DIM places the array in battery-backed RAM so its value is saved through a power-down. The values are automatically recalled on power up if the DIM1 statements are not rearranged.

If you modify your existing program by adding new DIM1 or DEFINE1 statements, data stored in memory will become invalid unless you add the new DIM1 or DEFINE1 after existing similar commands.

*If DIM1 and DEFINE1 are used, and the order of the DIM1/DEFINE1 declarations change, memory **MUST** be cleared prior to using those variables.*

Attention

*You **CAN'T** use structures or arrays for a setpoint variable basis or when using percent and offset conditions. You **MUST** copy the structure or array element into a temporary variable first.*

String Variable

A string variable may represent a group of letters, numbers or a combination of the two that no math functions can be performed on.

A string variable is assigned using "**=**" using quotes to surround the default string. i.e. **X$ = "TARE".**

A standard **string** variable can store a string 16 characters or less and is dimensioned by the character "**$**".

To define a shorter or longer string, the string variable can be succeeded by an "*****" and an integer (1-64). The integer defines the length of the string. i.e. "**DIM X$*10**" defines a string X$ with storage space for 10 characters.

String array

A string array is a list of string values. Like numeric arrays, string arrays must be defined using **DIM**/**DIM1** before they are referenced. String arrays, like string variables, must have the "**$**" character at the end of their name and the "*****" character specifies the length of the strings.

EXAMPLE:

```
DIM a$*32[100]
a$[99] = "SIMPLE"
```

The first statement defines the size of array "a$" as 100 strings. Each string is 32 characters long. The second statement assigns the value "SIMPLE" to the last element of the array a$.

Special Rules For Assigning Arrays

If two arrays are the same type (string, numeric) and the same size, an entire array can be assigned the values in the second array using one statement. Also, a range of elements can be specified in an assignment to set more than one array element to the same value with one statement.

EXAMPLE:

```
DIM a1[100]                DIM B1$[100]
DIM a2[100]      OR        DIM B2$[100]
a1[] = a2[]                B1$[] = B2$[]
```

After this statement is executed, every element in **a1** will be the same as the corresponding element in **a2** OR every element in **B1$** will be the same as the corresponding element in **B2$**.

Setting all the elements of an ARRAY to the same value:

a1[0,100] = 0

After the first statement is executed every element in a1 will be set to zero.

B1$[0,100]="GOOD"

After the first statement is executed, every element in B1$ will be set to GOOD.

a1[x,y] = 0 OR B2$[x,y]="BAD"

In the second statement x specifies the starting element and y specifies the number of consecutive elements to set to zero or set to BAD.

Structures

The variable and/or array list between the TYPE and ENDTYPE keyword commands, defines the STRUCTURE.

The programmer can specify a list of variables and arrays to place together in a group using a **TYPE** statement. Then the programmer can define a structure or array of structures of this type that will contain every element in the list using **DEFINE/DEFINE1** statements. DEFINE places the structure in volatile RAM (the values will be lost at power-down). Using DEFINE1 places the structure in battery-backed RAM (saved during power-down and automatically recalled on power up if the DIM1 statements are not rearranged).

EXAMPLE:

```
TYPE database
  string$*6
  integer%
  long&
  single!
  doubles[10]
ENDTYPE


DEFINE db as database
DEFINE1 dbase[100] as database
```

The first statement defines a single structure of type database. The second statement defines an array of 100 database structures.

To reference an element of a structure the name of the structure and the element within the structure must both be specified. The structure name and element name are separated by a ".".

EXAMPLE:

```
db.string$ = "AAAAA"
dbase[0].integer% = 1
```

System Variable

These variables have already been predefined by Weigh-Tronix in the Model 1310 WT-BASIC command set.

Items that bear an asterisk(*) can be set through a WT-BASIC (TARE=100) statement, although they can't be part of an input statement. The rest of the values are set only through firmware level calls. Also, when these are set, they **must** be in calibration unit of measure. When they are recalled the system returns the variable in current unit of measure.

All system variables have the ability to include a scale number in parenthesis after the keyword to select a scale number to access. If the parenthesis is not there the current scale is assumed. Scale#=0 returns the total of all scales.

Example: CAPACITY(3) returns the current capacity of scale three.

Items that bear an asterisk(*) can be set through a WT-BASIC equivalent statement, although they can't be part of an input statement.

All system variables have the ability to include a scale number in parenthesis after the keyword to select a scale number to access. If the parenthesis is not there the current scale is assumed. Scale#=0 returns the total of all scales.

System flags (CZERO,, MOTION, OVERLD, UNDERLD) will return a true (-1) or a false (0) value depending on their current condition.

ADCCNTS
Returns filtered raw Counts (not scaled to current unit of measure) from the currently selected base. Counts for Quartzell and Analog Base have different ranges.

CAPACITY
Returns the capacity of the currently selected scale.

COUNT
Is the actual calculated count value based on the Net Weight.
Count = Net/Pcwt.

COUNTTOT*
Returns or sets count total used by the DoAccum function to total the count.

SYNTAX: x=Countot (x is count total) or Countot=2365

CURSCALE* or CURSCALE=
Returns or sets the current active scale number selected. (0 through 8)

SYNTAX**:** Curscale=3 Sets scale 3 as the current scale

or

x=Curscale x is the current active scale

CURUNIT*
Returns or sets the value of the current units.

SYNTAX: Curunit=x (x is unit)

0=lbs	2=grams	4=lb oz	6=custom2	8=custom4
1=kg	3=oz	5=custom1	7=custom3	

CURUNIT$
Returns the label of the current unit of measure for the current scale, i.e. Kg, LB.

CZERO
Center of Zero flag. Returns -1 if center of zero, 0 if not center of zero (on the current scale)

DATE$(n)*
Returns or sets the internal date. (n) is the format syntax.

n	Description	Example
0	Numbers with 2-digit year	07-28-03
1	Spelled month	Jul 28, 2003
2	Numbers with day of week	Mon 07-28-03
3	Spelled month with day of week	Mon Jul 28, 2003
4	Numbers with 4 digit year	07-28-2003

ACTVALUE Table

#0	Gross
#1	Net
#2	Tare
#3	Minimum
#4	Maximum
#5	Rate of Change
#6	Gross Total
#7	Net Total
#8	Count Total
#9	Transaction Total
#10	Count
#11	Value
#12	Piece Weight
#13	A to D Counts

System flags (CZERO,, MOTION, OVERLD, UNDERLD) will return a true (-1) or a false (0) value depending on their current condition.

DIVISION*

Returns or sets the division size of the currently selected scale.

SYNTAX: x=Division (x is division size) **or** Division=5

DISPLAY

Returns the current numeric value being displayed.

GROSS

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

GROSSTOT*

Returns or sets the gross total used by the DoAccum function to total the gross weights. (Grosstot=Ø to reset)

SYNTAX: x=Grosstot (x is gross total) **or** Grosstot=1000

MAXPEAK*

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

SYNTAX: x=Maxpeak (x is new value)

MINPEAK*

Returns or sets the lowest value on the current scale rounded by division size.

SYNTAX: x=Minpeak (x is new value)

MOTION

Motion flag. Returns -1 if motion, 0 if no motion on the current scale.

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

NETTOT*
Returns or sets net total used by the DoAccum function to total the net weights.

SYNTAX: x=NETTOT (x is net total)

OVERLD
Overload flag. Returns -1 if overload, 0 if not on the current scale. (gross up to 120% of capacity)

PCWT*
Returns or sets the pieceweight value of the current scale.
(pcwt = 0.0017) **or** x=pcwt

RAWGROSS
Returns the unrounded gross weight of the current scale.

RAWNET
Rawnet is calculated from Rawgross-tare.

RAWTARE
This tare value represents the current tare in its highest resolution, which is **NOT** rounded to the nearest division.

ROC
Returns the rate of change of the current scale per configured parameters.

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.

SYNTAX: x=TARE (x is tare value)

n	*Time Formats*
0	*24 hr with seconds (18:00:00)*
1	*AM/PM with seconds (1:00:00 AM)*
2	*24 hr w/o seconds (18:00)*
3	*AM/PM w/o seconds (1:00 AM)*

TIME$(n)*
Returns or sets the internal time. (n) is the format syntax. See table at left.

TRANSTOT*
Returns or sets transaction total used by the DoAccum function to total the number of accumulations.

SYNTAX: x=TRANSTOT (x is transaction total) **or** TRANSTOT=65

UNDERLD
Underload flag. Returns -1 (true) if underload, 0 if not on current scale. (gross up to -120% of capacity)

Operators

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

Logical Operators

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

The arithmetic operators are listed in order of precedence. Multiplication, division, Integer Modulo, and Integer Divide have same precedence. Addition and subtraction have the same precedence. Items with the same precedence are evaluated left to right. Operations within parenthesis are performed first. Inside the parenthesis, the usual order of precedence is maintained.

() Parenthesis

- Negation

^ Exponent or 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
```

***** Multiplication

/ Division

m or **MOD** Modulus Operator (Requires a space on each side of operator)

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

Example: 10 divided by 3 = 3 with a remainder of 1. The modulus is 1.

The “m” must always be lower case and must always have a space before and after it.
Example: Returns the value of 1

```
SUB SYSTEM_STARTUP
dispmode=10
cls
VALUE=(10 m 3)
VALUE1=(20MOD3)
PRINT "The remainder is"; VALUE, VALUE1
END SUB
REM The VALUE is one
```

\ Integer Divide (Requires a space on each side of operator)

This operator divides 2 numbers and returns the Integer portion of the computation.
Example: 10 divided by 3 = 3 with a remainder of 1 (3 is the result)

```
SUB SYSTEM_STARTUP
dispmode=10
Cls
VALUE=(10\3)
PRINT "THE PRODUCT IS"; VALUE
END SUB
REM The VALUE is three
```

+ Addition

- Subtraction

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

Input/Output

Maximum of 40 characters/ display line for small font. Maximum of 30 characters/ display line for large font. Both fonts are fixed width.

PRINT

To output data to the screen. The area of the display dedicated to WT-BASIC messages is determined by the active display mode.

SYNTAX: PRINT “THIS WILL GO TO THE DISPLAY”

PRINT #*x*

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”

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

Example: Print “Hello”

Space

FMTPRINT(n)

Calls a predefined print format (n = 1 to 32) that will be sent out a serial port, as specified in destination port settings. See *Format Button* section in this manual.

INPUT(Prompt$,Var)

To receive data input from the keypad or a remote keyboard. The specified prompt is shown on the display. The input softkey interface is selected by the variable type to be input. **The prompt message in quotation marks must be followed by a comma, then your variable name.**

SYNTAX: INPUT “Enter id”, ID$ Alphanumeric entry

OR

INPUT "Enter#", VALUE Numeric entry only

These input options remain active for all input statements 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.

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.

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.

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.

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

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

INKEY$

Returns one character read from the remote keyboard or from the Model 1310 front panel keypad as a string. Returns an empty string " ", if no character is available.

SYNTAX: X$ = INKEY$

KEYHIT

This command is used in program loops to detect a key hit on the front panel of the Model 1310. This command only recognizes numeric keys on the front panel of the Model 1310.

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.

LASTKEY

LASTKEY returns the ASCII value of the last key pressed on the keyboard as a numeric value.

SYNTAX: x = LASTKEY

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

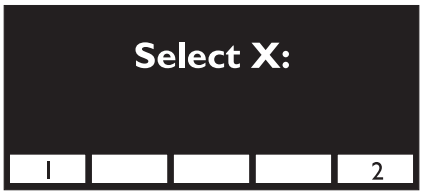
LASTKEY1

lastkey1 returns the ASCII value for the last key pressed on the front panel keypad. The ASCII value of lastkey1 resets itself to 0 after it has been assigned or printed.

Example:

```
SUB ENTER_KEY
dispmode=6
x=lastkey1
x1=lastkey1
print x,x1
END SUB
```

Displayed sample of above SUB ENTER_KEY routine showing the enter key as the last key press with its character value 13 which is reset to a 0 value after printed to the display screen.



ASK(MSG$) (PROMPT$[”F1”,”F5”])

ASK is a function that allows you to prompt the operator for a Yes/No type response. The ASK command accepts 2 optional arguments which allow the user to change the legends of the F1 key and the F5 key. The ASK(MSG$) function returns a negative one (-1) or true for a YES response and a zero (0) or false for a NO response.

SYNTAX: ASK("MSG$", "TRUE", "FALSE")

An example would be as follows:

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

TIMER

This command returns the number of seconds (in hundredths) since midnight of the current day.

UNIXTIME

This command returns the number of seconds (in hundredths) since midnight January 1, 1970.

SYSDATA (scalenum,type)

This command returns the system counters for a specific scale.

Type: 1 = Traffic counter
2 = Overload counter
3 = Underload counter

```
x=sysdata(1,2) REM Returns the overload counter for scale #1.
```

ERR

This keyword returns the error number corresponding to a given error which occurred in a WT-BASIC application. The following error numbers and errors are available. Err returns 0 if no error has occurred since power-up. The last error remains until a new error has occurred or until power-down.

SYNTAX**:**

X=ERR
or
PRINT ERR

0=	"No error"	20=	“WEND without WHILE”
1=	“syntax error”	21=	“Division by Zero”
2=	“unbalanced parenthesis”	22=	"EEPROM Sentinel”
3=	“no expression present”	23=	“Ram Sentinel”
4=	“equals sign expected”	24=	“Unbalanced brackets”
5=	“not a variable”	25=	“Array not defined”
6=	“Label table full”	26=	“Array index error”
7=	“duplicate label”	27=	“Duplicate TYPE define”
8=	“undefined label”	28=	“AS expression”
9=	“THEN expected”	29=	“Undefined structure”
10=	“TO expected”	30=	“Redefined structure”
11=	“too many nested FORs”	31=	“No structure member”
12=	“NEXT without FOR”	32=	“Non-volatile RAM full”
13=	“too many nested GOSUBs”	33=	“Array size mismatch”
14=	“RETURN without GOSUB”	34=	“Array type mismatch”
15=	“double quotes needed”	35=	“Structure index error”
16=	“String Expected”	36=	“Duplicate array name”
17=	“Variable Name Too Long”	37=	“Duplicate variable”
18=	“Var Not Defined (DIM)”	38=	“Extra array elements”
19=	“too many nested WHILEs”	39=	“Extra Structure Records”

CLEARERR(errtype)

Resets the err or neterr keywords to zero. Specify which one to clear.

errtype =	1: err Keyword	(Application error)
	2: Neterr (1)	(Network Card 1 error)
	3: Neterr (2)	(Network Card 2 error)
	4: SJBOX (1)	(SensorComm error)

SYNTAX:

```
SUB NET1_ONLINE
  ClearErr(2) ‘clear card#1 error, network online again.
END SUB
```

CLRDATA(scalenum,type)

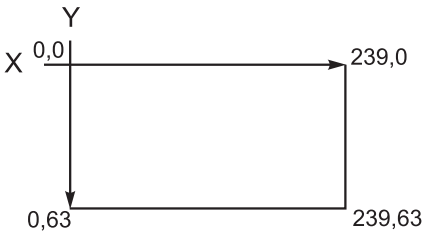
Clears the system counter for a specific scale.

Type: 1 = Traffic counter
2 = Overload counter
3 = Underload counter

clrdata(1,3) REM Clears the underload counter for scale #1.

Graphics

The Model 1310 has a display height of 64 dots and a width of 240 dots. Coordinate 0,0 is the upper leftmost dot.



Graphics can only be used when a display mode is chosen which does not contain a weight display. The upper left-hand corner of the display is (0,0). The lower right-hand corner is (239,63). The “X” references the horizontal coordinates (0-239). “Y” references the vertical coordinates (0-63).

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.

X1,Y1 are the starting point

X2,Y2 are the stopping point

Structure Control

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

*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*

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

‘ (apostrophe)
This also works in place of REM.

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
END IF

IF. .THEN.. ELSE (block)
SYNTAX:

IF *expression* THEN
    statement
    statement
    statement
ELSE
    statement
    statement
    statement
END IF

END IF
Used to stop a block of statements.

WARNING
GOTOs and GOSUBs are not recommended

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

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

GOTO

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

SYNTAX: GOTO *statement label*

YOU ARE STRONGLY URGED NOT TO USE THIS COMMAND DUE TO COMPLICATIONS IT CAN CAUSE IF NOT USED CORRECTLY.

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

FOR. .TO. .NEXT

To execute a series of instructions a specified number of times in a loop. Must also use “NEXT” command.

EXIT FOR

Used to terminate a FOR. . NEXT loop upon detection of a certain condition prior to the loop expiring naturally.

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 detection of a certain condition prior to the loop expiring naturally.

SYNTAX:
WHILE *expression*
    statement
IF *expression* THEN EXIT WHILE
    statement
WEND

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

GOSUB . . . RETURN

To branch to a subroutine and go back to the original subroutine.

RETURN

To return from a subroutine back to original subroutine.

SUB . . . END SUB

Defines a subroutine procedure.

EXIT SUB

Used to terminate a subroutine upon detection of a certain condition prior to the subroutine expiring naturally.

SYNTAX:
    SUB *subname*
        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

Display Control

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(n)
Sets the active display mode. The display modes may be viewed in accompanying software or in Appendix 1 of this manual. There are over 90 displays to choose from. (n) is the number representing the display mode.

SYNTAX: DISPMODE(n) or DISPMODE=n

KEY x,"keyname"
Names a function key on the screen. (x; 1-5)
SYNTAX: KEY 1, "START"
KEY 2, "STOP"
KEY 3, "SET UP"
KEY 4, "MORE"

Pay close attention to the location of spaces used with keywords that have a message inside of quotation marks. A space usually precedes the first quotation mark.

BACKLITE
Turns the backlight of the display on and off.
SYNTAX: BACKLITE(n) **or** `BACKLITE=n
0=OFF
1=ON

CONTRAST
Sets the contrast for the display.
SYNTAX: CONTRAST(n)
n=CONTRAST n=0 to 63

ACTVALUE
As a command ACTVALUE sets the displayed numeric data on the indicator to a different active value. See list at left.

SYNTAX: ACTVALUE(n) n=0 to 13
or
ACTVALUE=n

As a system variable ACTVALUE returns number (n) which represents the currently displayed active value per list at left.

SYNTAX: n=ACTVALUE n=0 to 13

Active VALUE Table

#0	Gross
#1	Net
#2	Tare
#3	Minimum
#4	Maximum
#5	Rate of Change
#6	Gross Total
#7	Net Total
#8	Count Total
#9	Transaction Total
#10	Count
#11	Value
#12	Piece Weight
#13	A to D Counts

ANBASIS(CH#,SCL#,BASIS[,VAR$])
CH# is the numeric value representing the analog output channel or board number in the system (CH#=1-8).

SCL# is the numeric value representing the system scale number this analog output channel is based on (SCL#=1-8).

BASIS is the active value in the system this analog output channel is based on (BASIS=0-13).

If #11 is used then the optional parameter VAR$ must be used.

0=Gross	4=Max	8=Count Total	12=Piece Weight
1=Net	5=ROC	9=# Total Tans.	13=ADC
2=Tare Value	6=Gross Total	10=Count	
3=Min	7=Net Total	11=Variable	

VAR$ is a numeric value or numeric variable that the analog output is based on. This is implemented by one of the following methods in your WTbasic program.

ANBASIS(3,2,**11**,**"**numeric_variable_name**"**)

Or

VAR$=" numeric_variable_name"
ANBASIS(2,3,**11**,**VAR$**)

SETANLOG(CH#,SCL#,MIN,MAX[,SPAN ADJ%,OFFSET ADJ%])

CH# is the numeric value representing the analog output channel or board number in the system (CH#=1-8).

SCL# is the numeric value representing the system scale number this analog output channel is based on (SCL#=1-8).

MIN is the minimum value represented by the analog output channel (voltage or current). This normally represents empty scale on the control room display panel.

MAX is the maximum value represented by the analog output channel (voltage or current). This normally represents a full scale on the control room display panel.

SPAN ADJ%,OFFSET ADJ% are optional parameters to compensate for the loading characteristics of the device in the control room that will respond to the analog output signal (voltage or current). **OFFSET ADJ%** affects the minimum value sent to the control room. **SPAN ADJ%** affects the maximum value sent to the control room. These values are typically between 0% and 100%. It is recommended that these adjustments be made on site via the front panel configuration menu. However if done in your basic program remember to store the values and then recall them for use on power up.

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

GRBASIS(n,[var$],scale)

This command sets the basis for the graph representation on the display. (n = 0 to 13) See list at left. "var$" only applies when the graph basis is set to "variable" (n = 11). "Scale" (1 to 8) specifies the corresponding scale number to which the basis setting is applicable (only valid for number of scales enabled).

SYNTAX: GRBASIS(11,"VALUE")
OR
VAR$="VALUE"
GRBASIS(11,VAR$)

SETBAR(*min,max,scale*)

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. "Scale" (1 to 8) specifies the corresponding scale number to which the basis setting is applicable (only valid for number of scales enabled).

In the SHOWVAR syntax the precision value is the number of digits on the right side of the decimal point for your displayed numeric value. A value of (-1) will put zeros on the display up to the number of digits used for your division size (i.e. a division size of 10 would place 00 on the display when your WtvalueVar is zero). Remember to to select a display mode that will allow you to see all the digits.

SETCHECK(*min,under,over,max,scale*)

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. (SCALE) (1 to 8) specifies the corresponding scale number to which the basis setting is applicable (only valid for number of scales enabled).

SHOWVAR (VAR$,LEG1$,LEG2$,PREC[,line#])

This lets you display a numeric value on the display where the weight value usually appears along with custom labels or legends. See note at left.

Two examples are shown below:

```
SUB F1_KEY
leg1$="Gross"
leg2$="kg"
wt=6543
var$="wt"
showvar(var$,leg1$,leg2$,0)
actvalue=11
END SUB
```

```
SUB F5_KEY
leg1$="net"
leg2$="lb"
wt=1234
showvar("wt",leg1$,leg2$,0)
actvalue=11
END SUB
```

SYNTAX: shown with optional parameters for line#, motion, and center of zero. If “mot” or “cz” have a value of 1 the motion and center of zero annunciators are shown, if they have a value of zero they are not shown.

SHOWVAR(VAR$,LEG1$,LEG2$,PREC[,LINE#,MOT$,CZ$])

or

SHOWVAR(“WTvalueVar”,LEG1$,LEG2$,PREC[,LINE#,”MotionVar”,”CzeroVar”])

```
SUB F3_KEY
leg1$="tare"
leg2$="ton"
mot=0
cz=1
wt=0
showvar("wt",leg1$,leg2$,-1,1,”mot”,”cz”)
actvalue=11
END SUB
```

If you have multiple SHOWVARs, you must have an equal number of scales enabled. Multi-scale display modes must be used. Using Line# will replace a weight value line with the SHOWVAR value.

SETSHOW(numLines)

This keyword configures the display for the number of lines to be used by multiple showvar statements.

numLines = 1 to 8

DISPSCL (1,2,3,4,5,6,7,8)

Allows the order OR total OR absence of a scale in a multi-scale display mode to be selected/specified. Also allows for the total to be displayed in a single-scale display mode.

SYNTAX DISPSCL(n1,n2,n3,n4,n5,n6,n7,n8)

n1 = scale number of 1st/top scale to be displayed
n2 = scale number of 2nd scale to be displayed
n3 = scale number of 3rd scale to be displayed
n4 = scale number of 4th scale to be displayed
n5 = scale number of 5th scale to be displayed
n6 = scale number of 6th scale to be displayed
n7 = scale number of 7th scale to be displayed
n8 = scale number of 8th scale to be displayed

NOTE: set “nx” to “0” to display total; set “nx” to “-1” to blank out this and all subsequent scale displays. "nx is n1.....n8"

SHOWSETP

Shows setpoints in the upper right of the display. Use only for troubleshooting. **This command should not be left in a customer's final program version.**

MENU

SYNTAX:

MENU"key1","routine1", ... "key5","routine5"

MENU allows the user to create a menu system without creating it with a series of if/then or “key” statements. The keyword requires the softkey legend and the subroutine which is queued for execution upon the keypress. Menu may be passed up to 5 softkey labels with sub routines. If a function is not defined for a given softkey, nothing happens.

```
SUB SYSTEM_STARTUP
dispmode=16
menu "setup","setup_values","","","","","","","start","start"
END SUB
```

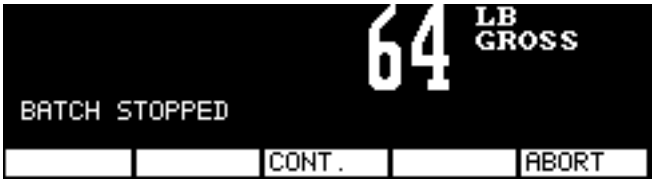
Brings up this display:


```
SUB start
cls
Print "     batching"
menu "stop","stop","","","","","","","","","",""
END SUB
```

Brings up this display:


```
SUB stop
cls
Print "  batch stopped"
menu "","","","","cont.","continue","","","abort","abort"
END SUB
```

Brings up this display:

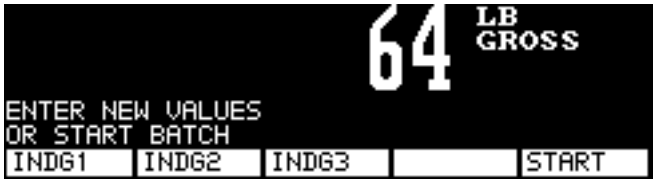



```
SUB continue
call start
END SUB

SUB abort
call setup_values
END SUB
```

```
SUB setup_values
cls
print "enter new values"
print "or start batch"
menu "ingd1","ingd1","ingd2","ingd2","ingd3","ingd3","","","start","start"
END SUB
```

Brings up this display:

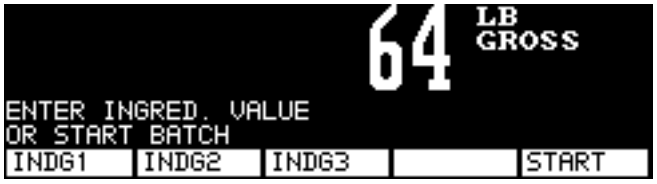



```
SUB ingd1
cls
print "enter ingred. value"
print "or start batch"
END SUB

SUB ingd2
cls
print "enter ingred. value"
print "or start batch"
END SUB

SUB ingd3
cls
print "enter ingred. value"
print "or start batch"
END SUB
```

All three of the previous subroutines will show this display:



MENU1

Very similar to the standard MENU command used for nested menu structures using the soft keys. This command when used properly will avoid interaction of keys and allows you to provide alternate functionality of the labeled keys on the front panel.

SYNTAX:

MENU1“ESC”,”ENTER”,”ZERO”,”UNITS”,”TARE”,”PRINT”,”SELECT”,”NUMERIC”,”ENTRY”,”CLEAR”

MENU1 “sub routine name that works on the ESCAPE key press”, “sub routine name that works on the ENTER key press”,… “sub routine name that works on the NUMERIC key press - meaning the pressing of any numeric character”, “sub routine name that works on the ENTRY key press - meaning the pressing of any non-numeric character”, “sub routine name that works on the CLEAR key press”

Serial Port Hardware Control

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 list below (1 to 13)

1 baud rate
2 parity
3 databits
4 handshake
5 mode
6 EOM character
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
12 Serial Blocking
13 Download Blocking

X is the return value from the table below

Baud	Parity	Databits	Handshake	Mode	CTS
1=300	0=none	7=7	0=none	0=Basic Control	0=off
2=1200	1=odd	8=8	1=CTS	1=Keyboard	1=on
3=2400	2=even		2=XonXoff	2=Disable	
4=4800	3=set			3=Multidrop	
5=9600	4=clear			4=Computer mode	
6=19200				5=Enquire mode	
7=38400					
8=56700					
9=115000					

Each serial port's transmit and receive 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 list below (1-15)
Pick z from the corresponding table below.

1 baud rate
2 parity
3 databits
4 handshake
5 mode
6 EOM character
7 RTS (only used in setport)
12 Serial Blocking
13 Download Blocking
15 Clears the receive buffer

Baud	Parity bits	Data	Handshake	Mode	CTS	RTS	Serial Blocking
1=300	0=none	7=7	0=none	0=Basic Control	0=off	0=off	0=on
2=1200	1=odd	8=8	1=CTS	1=Keyboard	1=on	1=on	1=off
3=2400	2=even		2=XonXoff	2=Disable			
4=4800	3=set			3=Multidrop			
5=9600	4=clear			4=Computer mode			
6=19200				5=Enquire mode			
7=38400							
8=56700							
9=115000							

Enquire Mode must be enabled and the EOM character ***MUST*** *be set to a 5 to operate with the Truck Weigh software program.*

SETPORT(port,5,5) sets enquire mode.
SETPORT(port,6,5) sets enquire EOM to 5

GETCOM$(port,[<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, 2, 3 or 4, (13 or 14 for modem)

maxchars (optional) represents the maximum number of characters to read and assign to the string (X$).

CLEARCOM (Port)

Clears serial transmit and receives buffers.

CONTOUT(fmt,rate)

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

This command activates 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 outputs format 1 twice per second.

SETPTON(n)

This command will activate SETPOINT n.

SETPTOFF(n)

This command will deactivate SETPOINT n.

ISSETPT(x)

Tells you whether the setpoint is 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.

TONE(x)

Turns the speaker on where x is a value between 0 and 65535.

TONEOFF

Turns the speaker off.

BEEP

Sounds the beeper in the instrument.

GETITEM (“ITEM”, INDEX,SCALE)

Get item is used to recall configuration parameters to be used by BASIC. The GETITEM request has 3 parameters. The first parameter is a two letter code that is used by the downloader to configure the indicator. The second parameter is used to request a particular data item from that list. The third parameter is used to choose a particular scale used by only certain configuration parameters if they fetch multiscale data. The following shows the context for using GETITEM and the two letter configuration codes that are supported.

GETITEM(“XX”, n)

“XX” - Is the configuration parameter to get the value from (See list below for an explanation of the configuration parameters available).

Supports “PF”, “PS”, “UN”, “KY”, “VT”, “MS”, “GS”, “DS”

n - Is the number of the configuration parameter to recall. The first configuration parameter is 1 and the second is 2 and so on. Any item that is numerical in nature is recalled, string parameters and undefined parameters return -2.

PF
Print format port. PF, n returns the port number used for print format n.

PS
Configures the power saving modes of the scale.

1) Power saving shutoff (-1 enable, 0 disable)
2) Power saving inactivity timer (integer entry 0 to 60 minutes)
3) Shutdown warning (-1 enable, 0 disable)

UN
Configures the number of active scales and the unit of measures.

1) Number of active Scales (1-3 scales (Local only is 1))
2) Calibration units - Selectable 0-lbs to 6-Custom unit 2
3) Pounds enable (-1 enable, 0 disable)
4) Kilograms enable (-1 enable, 0 disable)
5) Grams Enable (-1 enable, 0 disable)
6) Ounces enable (-1 enable, 0 disable)
7) Pound-Ounces enable (-1 enable, 0 disable)
8) Custom unit 1 enable (-1 enable, 0 disable)
9) Custom unit 2 enable (-1 enable, 0 disable)
10) Custom unit 3 enable (-1 enable, 0 disable)
11) Custom unit 4 enable (-1 enable, 0 disable)

KY
Key enable configuration.

1) SELECT key enable
2) UNITS key enable
3) PRINT key enable
4) TARE key enable
5) ZERO key enable

#0	Gross
#1	Net
#2	Tare
#3	Minimum
#4	Maximum
#5	Rate of Change
#6	Gross Total
#7	Net Total
#8	Count Total
#9	Transaction Total
#10	Count
#11	Value
#12	Piece Weight
#13	A to D Counts

VT
Display value enable.

1) Gross enable (-1 enable, 0 disable)
2) Net enable (-1 enable, 0 disable)
3) Tare enable (-1 enable, 0 disable)
4) Min enable (-1 enable, 0 disable)
5) Max enable (-1 enable, 0 disable)
6) Rate of change enable (-1 enable, 0 disable)
7) Gross total enable (-1 enable, 0 disable)
8) Net total enable (-1 enable, 0 disable)
9) Count total enable (-1 enable, 0 disable)
10) Transaction count enable (-1 enable, 0 disable)
11) Count enable (-1 enable, 0 disable)
12) Variable enable (-1 enable, 0 disable)
13) Pc. Wt. Enable (-1 enable, 0 disable)
14) ADC Enable (-1 enable, 0 disable)

MS
Miscellaneous configurations.

1) AC Excitation (0=DC, 1=360, 2=600, 3=1200)
2) Default print format (0-32)
3) Date format (0 MMDDYY, 1 DDMMYY, 2 YYMMDD)
4) Beeper volume (0 disabled, 1 low, 2 medium, 3 high)
5) Place holder
6) Place holder
7) Place holder
8) Place holder
9) Small font - enable lower case (0 disable, -1 enable)
10) Enable switching display mode with hidden key (0 disable, -1 enable)

GS
Scale specific graph configuration.

1) Graph minimum SYNTAX GETITEM(“GS”,1,[scale# - 1])
2) Graph below SYNTAX GETITEM(“GS”,2,[scale# - 1])
3) Graph above SYNTAX GETITEM(“GS”,3,[scale# - 1])
4) Graph maximum SYNTAX GETITEM(“GS”,4,[scale# - 1])
5) Graph basis SYNTAX GETITEM(“GS”,5,[scale# - 1])

DS
Selected display mode.

1) Power-up display mode number (1-95 integer)

GETITEM(“XX”, n, sc)

“XX” - Is the configuration parameter to get the value from (See list below for an explanation of the configuration parameters available).

Supports “TS”, “MC”, “ZS”, “FS”, “RS”, “SC”, “CT”, “CU”, “CF”

n - Is the number of the configuration parameter to recall. The first parameter for these configurations are scale number. The parameter immediately following the scale number is 1 and the next is 2 and so on. Any item that is numerical in nature is recalled. String parameters and undefined parameters return -2.

sc - Is provided to choose a scale from which to recall the parameter from. If a scale number is not provided the Local scale (0) is assumed.

EXCEPTIONS:

GETITEM(“PF”,f) - Returns the port number to which the format number ‘f’ is configured to output to.

GETITEM(“UN”,10) Returns a true (-1) if more than one unit of measure is enabled, otherwise returns false (0).

TS
Key time out configuration for **a specific** scale.

1) Accumulator time out (Float in seconds)
2) Print time out (Float in seconds)
3) Zero time out (Float in seconds)
4) Tare time out (Float in seconds)

MC
Motion detection configuration for **a specific** scale.

1) Motion detection enable (-1 enable, 0 disable)
(If motion detection is disabled Range and Delay are set to zero. Remainder of line is read, but not taken)
2) Range (float in divisions 0.1-20.0)
3) Delay (float in seconds 0.05-10.0)

ZS
Zero tracking configuration for a **specific** scale.

1) Zero tracking enable (-1 enable, 0 disable)
(If zero tracking is disabled Range and Delay are set to zero. Remainder of line is read, but not taken)
2) Range (float in divisions 0.1-20.0)
3) Delay (float in seconds 0.05-10.0)

FS
Filtering configuration for a **specific** scale.

1) Unused
2) Samples to average (1 to 60)
3) Harmonizer enable (-1 enable, 0 disable)
(Disabling the Harmonizer sets the constant value to 0)
4) Harmonizer constant (0 through 10)
5) Harmonize threshold (float weight in cal. unit of measure)

RS
Rate of change configuration for a **specific** scale.

1) Rate of change samples (0 through 60)
2) Rate of change multiplier (float)

SC
Configures a single scale.

1) Scale Type (0-Analog, 1-Quartzell, 2-Pulse Counter, 3-SensorComm, 4-X position, 5-Y position)
2) Quartzell port number
3) Capacity (Float- weight in cal. unit of measure)
4) Division (Float - weight in cal. unit of measure)
5) Update rate (0-5.0 Hz, 1-2 Hz, 2-1 Hz, 3-0.5 Hz, 4-0.25 Hz, 5-0.1 Hz, 6-10.0 Hz,)
6) Zero range (0-100%)
7) Linearization setting
8) Accumulator return to zero (0 to 100%)
9) Print return to zero (0 to 100%)
10) Cal Wt (float - weight in cal. unit of measure)
11) Multi-interval setting

CT
Configures counting settings.
1) Piece weight motion settings
2) Piece weight motion time

CU
1=Custom unit 1 conversion factor
2=Custom unit 2 conversion factor
3=Custom unit 3 conversion factor
4=Custom unit 4 conversion factor

CF
1=Custom Unit 1 Cal wt value
2=Custom Unit 1 Custom unit value
3=Custom Unit 2 Cal wt value
4=Custom Unit 2 Custom unit value
5=Custom Unit 3 Cal wt value
6=Custom Unit 3 Custom unit value
7=Custom Unit 4 Cal wt value
8=Custom Unit 4 Custom unit value

UNIT$

The command UNIT$(0) returns ‘lb’ and UNIT$(1) returns ‘kg’, etc. (For LB-OZ unit of measure, OZ is returned due to all weight variables return ounces when the LB-OZ unit of measure is selected.)

CALDATA(x, [y])

y = scalenum 1-8

Returns the following calibration information for the current scale, unless the scale is specified. This is for ISO purposes.

x=1 calfactor
x=2 calzero
x=3 Julian caltime
x=4 cal weight entered
x=5 cal weight on display after cal span

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 ; not used
x=3; Firmware Part Number
x=4; Firmware Revision Letter
x=5; Serial number of the indicator
x=6; Serial number of SimPoser which downloaded current file
x=7; Company name of SimPoser license holder
x=8; Xilinx part number
x=9; Xilinx revision number

SETPWD(n,p$)

Sets the system password only.

The value n selects the password to set (1 - System password)

The string p$ is the string containing the new password.

Syntax for setting the system password:

```
P$ = “PASSWORD”
SETPWD(1,P$)
```

Timer/Event/Sleep Control

There are four timer events available for use in the Model 1310. They are SYSTEM_TIMER, SYSTEM_TIMER2, SYSTEM_TIMER3, and SYSTEM_TIMER4. Use SETTIMER to regulate their frequency of occurrence.

SHUTDOWN
This command causes the scale to disable power and shutdown. There are no parameters for this function.

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 0.5 second. With zero seconds, this disables the timer event specified. This is the default state of the Timer Events upon power up.

SETTIMER(1,0.1) Minimum value is 0.1 seconds.

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

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(x)
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(0) will clear the entire event queue.

EVENTCLR(x)
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.

SLEEP(n)
Causes the program to halt processing for n number of seconds.

SYNTAX: SLEEP(10) (Causes processing to halt for 10 seconds)

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

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

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.

String Functions

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$) and makes a new string. The new string starts at the nth character of the original string (C$) and continues for x number of characters.

SYNTAX: Z$=MID$(C$,n,[x]) ([ ]=optional)
With C$ =”The Model 1310 Indicator”
and n = 5 and x = 10
MID$ returns the new string Z$ or “Model 1310”.

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 original string and continues for x number of characters to the right.

SYNTAX: Z$=LEFT$(C$,x)
With C$ =”The Model 1310 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 original string and continues for x number of characters to the left.

SYNTAX: Z$=RIGHT$(C$,x)
With C$ =”The Model 1310 Indicator”
and x = 9
RIGHT$ returns the new string Z$ or “Indicator”.

CHR$(n)
Converts an ASCII code to its equivalent character. See Appendix 5.

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. Positive width is Right justified. Negative width is Left justified.

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 argument.

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 Julian 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$=”Model 1310 INDICATOR”
D$=”N”
Z=INSTR(C$,D$)
The value of Z is now set to 9

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

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

SYNTAX: X = PLEN(C$)

CHECKSUM(DATA$, TYPE)
Calculates a checksum value for the given string (C$). The checksum calculation type (n) is and integer value representing the type of checksum to calculate. The possible values for n are shown below:

1 = 32 Bit Sum
2 = 8 bit XOR
3 = CCITT method CRC16
4 = XMODEM method CRC16
5 = CRC-16

The value returned is 32 bits, but not all calculation methods use the whole resolution.

Weighing Functions

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

DOUNITS
This command switches the display to the next valid unit of measure for all scales. (If other scales are not on the same unit of measure the current scale is incremented to the next unit of measure and the rest of the scales are forced to that same unit of measure)

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

DOPRINT
This command prints the selected default print format as previously configured if no motion is detected. The default print format is not printed if PRINT_KEY subroutine appears in the program.

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 . . .).

DOBASE
This command switches to the next configured and enabled scale.

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.

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.***

AVGSTART[(scale number)]

Starts averaging for in-motion systems. The optional parameter, scale number, specifies the scale to average. If this parameter is not specified, the current scale is averaged.

AVGSTOP[(scale number)]

This returns the average weight (X) on the scale since the last AVGSTART command or for the last 256 weight conversions, whichever is shorter. The optional parameter scale number specifies the scale number to stop averaging on. If a scale number is not given, the it is assumed to be the current scale.

SYNTAX: X=AVGSTOP

INMOTION(startIO,stopIO,startLocation,quantity,min,max[,scale number])

This keyword configures the firmware level inmotion system. This is different from the AVGSTART and AVGSTOP keywords.

Explanation of parameters:
startIO - IO location (setpoint) for the input to start the weighment. Positive value indicates start on activate, negative indicates start on deactivate.
stopIO - IO location (setpoint) 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
min - Minimum (inclusive) acceptable gross weight.
max - Maximum (inclusive) acceptable gross weight.
Scale number – this is an optional parameter that specifies which scale to read weight on. If this parameter is not specified, weight values are taken from the current scale.
The Store/Recall value at recall(startLocation+quantity) holds the value

stopIO+1 = Accept Setpoint
stopIO+2 = Reject Setpoint
stopIO+3 = Over Setpoint
stopIO+4 = Under Setpoint

Note: These setpoints must be configured for Basic control Activate and Deactivate.

Example:

```
SUB SYSTEM_STARTUP
 dispmode=1
 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
```

GETCONV

Gets the conversion factor of the currently displayed unit measure.

SYNTAX: X = GETCONV

PCWTSAMP(SAMPLEsize)

This keyword takes the sample size and tries to calculate a piece weight based on the weight on the scale.

PCWTZERO

This keyword performs a finer zero operation based on the counting criteria set in the counting configuration

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(0) - stop capture
CAPTURE(1) - start capture
CAPTURE(2, xstart, xdatapt, xrate, xch, xs) - configure capture

xstart - starting memory location (store/recall number)
xdatapt - number of data points to take
xrate - sampling rate (max 60Hz)
xch - scale number to take readings from
xs - setpoint number that can control the activation or deactivation of the capture feature.
Enter "0" to disable the need for setpoint activation.

CAPTURE(3, xstatus) - store current status.

The status stores the store/recall address of the next data point. To calculate the number of samples taken, subtract the status character from the start address (xstart).

xstatus - the store/recall address to place the status.

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 ‘deactiva
                tion 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
```

Memory

Option boards start at 8192 and 4096 respectively.

Option	*#s*	*Strings (16 char)*
1MB	*65536*	*32768*
4MB	*262144*	*131072*
5MB	*327680*	*163840*
8MB	*524288*	*262144*

In addition to the Main board.

Attention

*It is strongly recommended that STORE and RECALL keywords are **NOT** used in conjunction with DIM1 and DEFINE1 keywords, due to misalignment of data.*

If a memory option cared is installed, all DIM1 and DEFINE1 variables will be mapped to the memory card, thus leaving the main RAMs open STORE/RECALLs.

If these are used in the SimPoser, simulator mode may not reflect indicator operation.

There are 8192 numeric storage locations and 4096 sixteen character string storage locations available in nonvolatile memory. These memory locations are reusable and will remain through a power loss. Four commands are available for accessing these memory locations:

Locations start at 0 numerics go from 0 to 8191
strings go from 0 to 4095

STORE(loc, #value[,hiloc])

Stores #value in the loc numeric memory location. If a number for hiloc is included, a fill is performed thereby storing the value in all locations from loc to loc+hiloc.

STORE$(loc, "string value"[,hiloc])

Stores “string value” in the loc string memory location, 16 characters maximum per location. If a number for hiloc is included, a fill is performed, thereby storing the string value in all locations from loc to loc+hiloc.

RECALL(loc)

Recalls the loc numeric memory location and assigns the value stored there to your variable (X).

Syntax: X = RECALL(loc)

RECALL$(loc)

Recalls the loc string memory location and assigns the value stored there to your variable (C$).

SYNTAX: C$=RECALL$(loc)

DIM1 and DEFINE1

These can be used in place of DIM and DEFINE. If DIM1 and DEFINE1 are used, the values of the variable, array, or structure are placed in battery-backed RAM. This is the same memory that STORE and STORE$ use. Therefore, it is strongly recommended that if store/store$/recall/recall$ are being used in a program, DIM1 and DEFINE1 are not used. Also, all DIM1 and DEFINE1 commands must occur in the same order every time the WT-BASIC program is run for the correct values to be recalled. It is recommended that all DIM1 and DEFINE1 commands be placed in the SYSTEM_STARTUP subroutine. If the order must be changed, a CLEARMEM command (below) should be used to clear the battery-backed RAM.

CLEARMEM(memType)

Clears all battery backed RAM.

memType = 0 for Main Board
memType = 1 for expansion memory.

MEMLEFT

Returns the number of bytes remaining in battery-backed RAM.

MEMOPT

Returns the memory option(s) that are currently installed.

0 = main board only
1 = 1MB expansion SRAM
2 = 4MB expansion SRAM
3 = 5MB expansion SRAM
4 = 8MB expansion SRAM

SIZEOF

Returns the size of a defined type.

EXAMPLE:

```
TYPE dbase
  part$
  netwt
  desc$
ENDTYPE

x  = sizeof(dbase)
```

x will be set to 42 (17 bytes for each string (16 characters + 1 null terminator) and 8 bytes for the double).

UPLOAD(port, "struct", "type"[, start, number])

See STRUCTURES for information on using this keyword.

Upload is used to send the data stored in an array of structures out of a serial port. Port specifies the serial port, "struct" is the name of the array of structures. "type" is the name of the type used to define the structure. Start is an optional parameter that specifies the index to start at (if not present, the upload starts at index 0). Number is an optional parameter that specifies how many records to upload (if not present, the upload sends all records).

EXAMPLE:

```
TYPE database
  var1
  var2
  array1[2]
  var3
ENDTYPE
DEFINE dbase[100] as database
UPLOAD(1,"dbase","database",0,100)

Data sent out of serial port #1:
dbase[0]var1,var2,array1[0],array1[1],var3
dbase[1]var1,var2,array1[0],array1[1],var3
          .
          .
          .
dbase[99]var1,var2,array1[0],array1[1],var3
```

*Items **inside** of square brackets '[ ]' are optional only if a **comma** is the first character shown in the syntax example.*

When your storing data in battery backed RAM, DOWNLOAD will overwrite existing data but to be sure any extra data in memory is deleted first, use CLEARMEM prior to using DOWNLOAD.

DOWNLOAD(port,"struct","type")

Download is used to import data into an array of structures through a serial port. Port specifies the serial port, "struct" is the name of the array of structures and "type" is the name of the type used to define the structure. After the download command is given to the indicator, the data should be sent into the serial port in the order the structure was defined in the WT_BASIC program. Commas should be used to separate elements. A semicolon can be used to signal the end of an array if the entire array is not being sent. Records should be separated be a carriage return.

EXAMPLE:

Code:

```
TYPE database
  var1
  var2
  var3
  array1[2]
  array2[2]
ENDTYPE
DEFINE dbase[100] as database
DOWNLOAD(1,"dbase","database")
```

Data sent into serial port #1:

```
dbase[0]var1,var2,var3,array1[0],array1[1];array2[0],array2[1]
dbase[1]var1,var2,var3,array1[0],array1[1],array1[2],array2[0]
          .
          .
          .
dbase[99]var1,var2,var3,array1[0],array1[1],array1[2],array2[0]
```

See note at left.

You must send two consecutive carriage return/line feeds (CRLF) to mark the end of the data download.

FIND(TARGET,START,STOP[,TYPE])

Searches numeric memory slots between START and STOP for a TARGET variable and returns the memory location. Returns a (-1) if not found.

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

If x > y the keyword will look backwards through the specified memory locations.

Optional Argument ‘type’:

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

type - provides optional search patterns. If type is not provided, the default is to find an equal value (type = 0) The search options are as follows; type =

0 - Find an equivalent to the given value (=)
1 - Find the first value that is less than the given value(<)
2 - Find the first value that is less than or equal to the given value (<=)
3 - Find the first value that is greater than the given value (>)
4 - Find the first value that is greater than or equal to the given value (>=)
5 - Find the first largest value within the given range (VAR is ignored) (max)
6 - Find the first smallest value within the given range (VAR is ignored) (min)
7 - Find the pass through value within the given range (pass through)
14 - Find the first location that is not equal to VAR

Pass through is defined as the index of the first value from (x+1) to (y) that is greater than the given value where VAR > value(x), or the index of the first value from (x+1) to (y) that is less than the given value where VAR < value(x) If VAR = value(x) then x = x + 1.

If x > y the keyword will look backwards through the specified memory locations.

FINDSTR(TARGET$,START,STOP[,case,type])

Searches numeric memory slots between START and STOP for an equivalent string and returns the memory location. Returns a (-1) if not found.

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

Optional Arguments 'type' and 'nocase':

SYNTAX: Z = FINDSTR (VAR,x,y,nocase,type)

nocase - if set to true (nonzero) the strings are compared after being converted to all upper case letters. The default is False (0).

type - provides optional search patterns. If type is not provided, the default is to find an equivalent string (type = 0) The search options are as follows; type =

0 - Find an equivalent string to the given string (=)
1 - Find the first string that is less than the given string(<)
2 - Find the first string that is less than or equal to the given string (<=)
3 - Find the first string that is greater than the given string (>)
4 - Find the first string that is greater than or equal to the given string (>=)
5 - Find the first largest string within the given range (VAR is ignored) (max)
6 - Find the first smallest string within the given range (VAR is ignored) (min)
7 - Find the pass through string within the given range (pass through)

Pass through is defined as the index of the first string from (x+1) to (y) that is greater than the given string where VAR > value(x), or the index of the first string from (x+1) to (y) that is less than the given string where VAR < value(x) If C$ = string(x) then x = x + 1.

8 - Find the string that contains the given string (Contains)
9 - Find the string that is contained by the given string (is contained by)
10 - Find the first string that is less than the given string(<). This selection differs from #1 in that the search criteria is determined by the length of the string, then by the normal string compare conventions.
11 - Find the first string that is less than or equal to the given string (<=). This selection differs from #2 in that the search criteria is determined by the length of the string, then by the normal string compare conventions.
12 -Find the first string that is greater than the given string (>). This selection differs from #3 in that the search criteria is determined by the length of the string, then by the normal string compare conventions.
13 -Find the first string that is greater than or equal to the given string (>=).This selection differs from #4 in that the search criteria is determined by the length of the string, then by the normal string compare conventions.
14 -Find the first location that is not equal to VAR.

See sample applications for STRUCTURE key word examples.

SEARCH(name, start, stop, value, type)
SEARCH(name$,start,stop,value,nocase,type)

SEARCH is used to search an array or array of structures for a given value and return the array index once found. The name can be any one of the following forms:

arrayname
structurename.arrayname
structurename[0].arrayname
Structurename[].variablename

In the first three cases, an array is searched. In the last case, an array of structures is searched and the specified element (variablename) is compared to the target value. The start parameter specifies the index at which to

start the search. The stop parameter specifies the index at which to stop searching. The value parameter specifies what to search for. The optional parameter type specifies the type of search to perform (same as type for find and findstr). If the optional parameter nocase (string searches only) is set to a non-zero value, the strings are compared after being converted to all upper case letters.

SORT(name, reverse)
SORT(name$,reverse,nocase)

The **SORT** command is used to sort an array or an array of structures. The name parameter can be any of the following forms:

Arrayname
Structurename.arrayname
Structurename[0].arrayname
Structurename[].variablename

In the first three cases, an array is sorted. In the last case, an array of structures is sorted according to the value of the structure element (variablename). If the optional parameter, reverse, is nonzero, the array (or structure array) will be sorted in reverse order. The second optional parameter, nocase, parameter applies only when sorting by a string value. If nocase is set to a nonzero value, the object will be sorted without regard to case. Normally, uppercase characters have a lower value that lowercase characters.

SensorComm

SCOMERR(1, index)

Returns the specific error for a SensorComm scale and the last ten errors that have occurred on that scale.

Chain# = 1 Always set to 1
Index = 1-10

SCOMNUM(Sensorcomm, Scale#)

Returns how many errors have occurred on this SensorComm scale.

SCOMDIAG(opt, ScomBox, value)

Scomdiag returns or updates individual sensor counts or SensorComm Box voltages. This can only be done with SensorComm enabled. The (opt) option parameter specifies either a value to return or update. To get current data, an update should be issued just prior to requesting a value.

*SCOMDAIG **must not** be used more than one time per second to refresh diagnostic values.*

Opt = 1 - Update the A to D counts for all the sensors from a SensorComm Box. The sensor number (value) is not needed and should not be included for this option (opt) value.

Example: **NEWvalue = SCOMDIAG(opt,ScomBox)**
or
Example: **NEWvalue = SCOMDIAG(1,1)**

Opt = 2 - Return the A to D counts for a specific sensor. The SensorComm Box and sensor number (value) must both be specified as shown here for sensors 1-3 on SensorComm Box 1.

example: **cellONEvalue = SCOMDIAG(opt,ScomBox,value)**
or
Example: **cellONEvalue = SCOMDIAG(2,1,1)**
cellTWOvalue = SCOMDIAG(2,1,2)
cellTHREEvalue = SCOMDIAG(2,1,3)

Opt = 3 - Update the values of the supply voltages for a SensorComm Box. The (value parameter) is not needed and should not be included for this option (opt).

Example: **NEWvalue = SCOMDIAG(opt,ScomBox)**
or
Example: **NEWvalue = SCOMDIAG(3,1)**

Opt = 4 - Return the values of the supply voltages for a SensorComm Box. The SensorComm Box and voltage (value) must both be specified as shown here for the supply voltages from SensorComm Box 1. (value) is a number from 1-4, representing voltage in (+15.0VDC) or lower on long cable runs, minus and plus excitation voltages(+/-5.0VDC), and the regulated logic supply (5.0VDC)

#1 = +15V unregulated input
#2 = +5V excitation
#3 = -5V excitation
#4 = +5V logic supply

Example: **NEWvalue = SCOMDIAG(opt,ScomBox,value)**

Examples: **Vin15VDC = SCOMDIAG(4,1,1)**
EXCTminus5VDC = SCOMDIAG(4,1,2)
EXCTplus5VDC = SCOMDIAG(4,1,3)
LOGICplus5VDC = SCOMDIAG(4,1,4)

Traxle

Traxle(type[,memLoc])
Where type is the option for the keyword and memLoc is an optional argument for where to store the results.

Traxle(0,memLoc)
Configures the system to store the axle results starting at memLoc). Note axle weights are stored, followed by the total.

Traxle(1)
Aquires the data for the stop#1

Traxle(2)
Aquires the data for the stop#2

Pulse Counter

Mode1: IndexPin
GND = Run
+5V = Reset/Stop
Mode2: IndexPin
GND Pulses = Reset
Couter Keyword = Pause or Run
+5V = Do nothing

Counter(option[,channel,value])
This keyword allows the configuration and retreival of data from the pulse counter option board. Jumpers must be properly set on the PC Board for software control. See the *Service Manual* for jumper location information.

Counter(0[,mode])
Initializes **all** counter boards

mode = 1, Free run mode
X=Counter(0,1)
mode = 2, Pause mode
X= Counter(0,2)

Counter(1,chNum)
Returns the counts for chNum.

chNum = channels 0-15
X=Counter(1,1) returns the current # of pulses for channel 1

Counter(2,chNum,counts)
Sets the pulse counter register for chNum to the values contained in the counts variable.

X=Counter(2,0,0) sets ch1 to zero pulses
X=Counter(2,1,0) sets ch2 to zero pulses

Counter(4,boardNum,state)
Allows the user to pause or stop all channels on a board from counting pulses.

boardNum = 0, 1, 2

0=All boards
1=Board 1 (ch# 1-8)
2=Board 2 (ch# 9-16)

State = 0 - Run
State = 1 - Paused

Counter(5,howMany,memLoc)
Allows the user to trap "howMany" number of counter channels and stores them starting at "memLoc".

howMany = 1-16
memLoc follows the numeric memory limitations of your system.

Network

Some of the scale functions listed below cannot be simulated on your computer. You will need to download the program to an indicator for testing purposes.

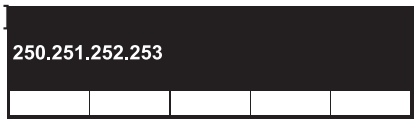
Functions:
Multiscale inputs
Quartzell
SensorComm
Networks
Memory management
Setpoints

Card Type:
Profibus = 1
Interbus = 16
DeviceNet = 37
Modbus Plus = 64
ControlNet = 101
ModBus/TCP = 128
EtherNet - Raw Sockets = 128
10/100Mb Ethernet-IP = 131
10/100Mb Ethernet-IT = 131
10/100Mb ModBus/TCP = 131
10/100Mb Ethernet - Raw Sockets = 131

Example Program:

```
SUB SYSTEM_STARTUP
   a=getnet(1,19,0)
   b=getnet(1,19,1)
   c=getnet(1,19,2)
   d=getnet(1,19,3)
print a;".";b,".";c,".";d
END SUB
```

Display will show:



Example:
IP is 250.251.252,253

Option	Returns
0	250
1	251
2	252
3	253

NETERR(cardnum)

This keyword returns an error number associated with a network card.

cardnum = 1 or 2

Return value:
0 = no error
1 = not enabled
2 = card type configured does not match card installed
3 = initialization failed
4 = offline

SYNTAX:

```
SUB NET1_OFFLINE
  Dipsmode=6
  Cls
  Print "Network Error: ";neterr(1)
  Print "Please call your service technician"
END SUB
```

GETNET

Returns the numeric value (x) that represents the current setting of the network card selected from the tables below.

SYNTAX: x=getnet(cardnum,y[,option])

Pick 'y' from the table below (1 to 17)
1: Card type (see note at left)
2: Not used
3: Network card serial number
4: Not used
5: Input memory I/O size
6: Input memory RAM size
7: Input memory total size
8: Output memory I/O size
9: Output memory RAM size
10: Output memory total size
11: Software revision
12: Boot loader software revision
13: Not used
14: Card enable (0 = disable, 1 = enable)
15: Input memory reset state (0 = clear, 1 = freeze)
16: Output memory reset state (0 = clear, 1 = freeze)
17: WT standard output enable state (0 = disable, 1 = enable)
18: Not used
19: Returns the IP octete specified by 'option.'
20: Returns the SUBNET MASK octete specified by 'option.'
21: Returns the Gateway octete specified by 'option.'
22: Returns the Host for SMTP Server IP octete specified by 'option.'
23: Returns the sockets port address.
24: Returns the socket number.
25: Returns the SMTP server IP octete specified by 'option.'
26: Returns the word swap configuration.

For y values 19-23, & 25 see *Option* table to the left.

setnet(1,13,1)

Reinitialize must be executed after any setnet command that effects the card settings or data output from the fieldbus network card.

SETNET

Sets the numeric value (x) which represents the current setting of the network card selected from the tables below.

SYNTAX: setnet(cardnum,y,x)[,a,b,c]

Pick 'y' from the table below:

5: Input memory I/O size
6: Input memory RAM size
7: Input memory total size
8: Output memory I/O size
9: Output memory RAM size
10: Output memory total size
13: Reinitialize network card
14: Card enable (0 = disable, 1 = enable)
15: Input memory reset state (0 = clear, 1 = freeze)
16: Output memory reset state (0 = clear, 1 = freeze)
17: WT standard output enable state (0 = disable, 1 = enable)
19: Sets the IP address. Example: IP 10.5.0.7 x=10 a=5 b=0 c=7
20: Sets the SUBNET MASK. Example: SUBNET MASK 255.255.255.0 x=255 a=255 b=255 c=0
21: Sets the Gateway. Example Gateway 0.0.0.0 x=0 a=0 b=0 c=0
22: Set Host IP or SMTP server IP Address Example: Host IP x=192 a=168 b=1 c=100
23: Sets the socket port address.
24: Sets the socket number
26: Sets the word swap data format for long and floating point data.

MAP(card#,sclNum,exLoc,intLoc,cmd,I_O[,"var"])

card#	Defines the card number for the system. 1 or 2 is used.
sclNum	Defines which scale should output its value. 1-8 is used.
exLoc	Defines the external memory location to store the information.
IntLoc	Defines the internal memory location for the network card.
Cmd	Defines the command/system variable to transmit/receive via the network. See table for definitions.
I_O	Defines if the command is inbound data or outbound data. 1=Inbound, 0=Outbound
"Var"	Optional argument to send a WT-BASIC variable via the network.

The map keyword allows the user to direct a given system variable to the network automatically. The system variables that are mapped to the network are updated at 10Hz.

Map Command Definitions:

Inbound Data to Model 1310	Command#:	Outbound Data from Model 1310:	Command#:
Request Zero	0	Actvalue(Gross=0…ADC=13)	0-13
Request Tare	1	Motion	100
Request Print	2	Center of Zero	101
Request Accumulation	3	Actvalue#(Gross=0…ADC=13)	102
Set Tare	4	Over/Underload	103
WT-BASIC Variable	5	Current Unit(0=lb, 1=kg…)	104
Set Current Units	6	WT-BASIC Variable	105
Setpoint Bank	110-117	Setpoint Bank	110-117
Fmtprint(150=1…181=32)	150-181	Setpoint	200-263
Setpoint	200-263	Recall Memory	1000-9191
WT-BASIC Event#	400-591		
Store Memory	1000-9191		

***For more detailed information on network connections and programming please reference the* Model 1310 NETWORK INSTALLATION GUIDE P/N 29806-0013.**

WT-BASIC example for 3 scale system transmitting gross weight only:

```
SUB SYSTEM_STARTUP
Map(1,1,0,0,0,0) 'map scale#1 gross weight
Map(1,2,2,1,0,0) 'map scale#2 gross weight
Map(1,3,4,2,0,0) 'map scale#3 gross weight
END SUB
```

10/100Mb Ethernet Keywords

If you are connecting to the 1310 via web browser, such as Microsoft Internet Explorer®, do the following to be sure you are getting current information, not cached information.

Under Tools>Internet Options> General>Temporary Internet files>Settings:

*Set **Check for newer versions of stored pages** from **Automatically** to **Every visit to the page**.*

If you use another browser, follow the steps necessary to turn off the page caching option.

File System Keywords:
The 1310 indicator's 10/100Mb Ethernet option card has a file system, which allows applications to read, write, and delete files. These files can be used for HTML code, data, or error logs. An email maybe sent, to notify someone of errors, or that data is ready to be extracted.

FOPEN(fileName$,mode[,boardNum])
This keyword allows the programmer to create/open a file on the Ethernet 10/100Mb option board. You may open the file, open it for read only purposes, or open it as append(write access, in addition to what is already in the file). This keyword MUST be issued prior to using a print formats whose destination ports are set to 70 or 80.

Syntax: handle = fopen(fileName$,mode)
handle = numeric return value used for further file operations
FileName$ = This is the name of the file you wish to create a file handle for.
Mode = Type of File access
Mode Values:
0=open existing file for reading
1=open new file for writing
2=open existing file for appending
BoardNum = an optional argument that tells the system which board to access, default=1.
BoardNum Values:
1=Card 1
2=Card 2

FREAD(fileName$,size,strLoc[,boardNum])
This keyword allows the programmer to read data from the Ethernet 10/100Mb option board. You may open the file for read access, then issue the fread keyword to read data from the file. This is useful when a remote server places a data file via FTP in the indicator. You can then read the data and set target weights, piece weights, load local database values, etc.

Syntax: fread(handle,size,strLoc)
Handle = This is the file handle generated by the *fopen* keyword.
Size = The size of the string you wish to read in.
StrLoc = The location in store/recall memory to store the string you are reading from the file.
BoardNum = an optional argument that tells the system which board to access, default=1.
BoardNum Values:
1=Card 1
2=Card 2

After fread is called, the fsize keyword holds the number of characters read, if the file is less than the number requested the end of the file has been reached. The ferror keyword holds the value of any error that occurred in the read (0 = success)

FWRITE(handle, size,strLoc[,boardNum])
This keyword allows the programmer to write data to the Ethernet10/100Mb option board. You may open the file for append access, then issue the fwrite keyword, or call a print format whose destination port is directed to Port 70 & 80. Data will appear in the file that you specify. Port 70 will create a file handle for card 1. Port 80 will create a file handle for card 2. Data, HTML code, or error logs can be generated using these keywords.

Syntax: fwrite(handle,size,strLoc) or fmtprint(1)
Handle = This is the file handle generated by the *fopen* keyword.
Size = The size of the string you wish to write to disk.
Strloc = Store/recall location for the string you wish to send to the file.
BoardNum = an optional argument that tells the system which board to access, default=1.
BoardNum Values:
1=Card 1
2=Card 2

After fwrite is called, the fsize keyword holds the number of characters written. The ferror keyword holds the value of any error that occurred in the read (0 = success)

Attention

It is strongly recommended that all files be closed using the FCLOSE keyword prior to viewing, copying or deleting the file form WT-BASIC or a PC via FTP.

FCLOSE(handle[,boardNum])
This keyword allows the programmer to close a given file on the Ethernet 10/100Mb option board. You will use the file handle generated by the *fopen* keyword.

Syntax: fclose(handle)
Handle = This is the file handle generated by the *fopen* keyword.
BoardNum = an optional argument that tells the system which board to access, default=1.
BoardNum:
1=Card 1
2=Card 2

The ferror keyword holds the value of any error that occurred in the read (0 = success)

FDELETE(fileName$[,boardNum])
This keyword allows the programmer to delete a given file on the Ethernet 10/100Mb option board. You will use the file name that was used by the *fopen* keyword. To delete the file, it must not be open. If the file has been opened, issue the fclose command before deleting the file. The ferror keyword holds the value of any error that occurred in the read (0 = success)

FERROR
This keyword returns the most recent error from a file access. This keyword is not card specific.

Syntax: fileError = ferror

Errors:
0=success
1=invalid IP address or subnet mask
3=no free socket
12=failed to open file or file not found
13=Invalid file handle or filename
14=invalid open mode
15=no email server configured

FSIZE
This keyword returns the most recent number of bytes read from or written to a file from a read or write operation. It also holds the size of a file just after it is opened.

Syntax: fileSize = fsize

E-Mail Keywords

SENDMAIL(to$,from$,subject$,textBody$)
This keyword allows the programmer to send an email via an SMTP server. The SMTP server's address is configured via the SimPoser Configuration, or the setnet keyword. This address must be an IP address. To determine the IP address of a named mail server ping the server from a computer (e.g. ping mail.wtx.com) and use the IP address displayed.

Syntax:

mailError = sendmail("don@hotmail.com","1310@weigh-tronix.com","Errors Present","Goto http://1310.1.com/errorlog.htm")
mailError = any error that occurred during the mail send.

mailError:
(-1)=not valid (ignore this response)
0=success
1=invalid IP address or subnet mask
3=no free socket
15=no email server configured

to$ = the email address of the recipient
from$ = the email address of the sender, also used as the reply-to address in case anyone replies to the email message. This ideally is set to someone's email address that should receive any replies. The 1310 will not receive emails.
subject$ = The subject of the email
textBody$ = The body of the email. As shown in the example, this can be a hyperlink to a web page in the scale indicator that contains more details for the message.

GETSOCK$(*x*):
This keyword allows the user to fetch a message from the receive buffer for the Ethernet option. GetSock will return all characters up to and including the End of Message character, outlined on the SimPoser network configuration tab.
Syntax:

```
SUB SOCK1_MESSAGE
  Msg$=getSock$(1)
END SUB
```

SOCKSTAT(card#, socket#):
This keyword returns the status of the network and returns the number which represents this status. The *x* represents the card number that has been selected.
Syntax:

```
SUB SOCK1_OFFLINE
  Dipsmode=6
  Cls
  Print “Socket Error: “;sockstat(1)
  Print “Please call your service technician”
END SUB
```

Reference #s for Cards and Sockets:
Card # can be #1 or #2.
Socket # can be #1-#9.

SOCKSTAT values:

0 = Not Active	1 = Listening
2 = Connecting(OFFLINE)	3 = Connected(ONLINE)
4 = Connection Refused	5 = Connection TimeOut
6 = Connection Failed	

SOCKERR(card#, socket#):
This keyword returns the last socket error that has occurred. The *x* represents the card number that has been selected.

Syntax:

```
SUB SOCK1_OFFLINE
  Dipsmode=6
  Cls
  Print “Socket Error: “;sockstat(1)
  Print “Please call your service technician”
END SUB
```

SOCKERR values:

1= Invalid IP-address or SUBNET mask.	2 = Invalid socket type
3= No free socket	4 = Invalid socket
5 = Not connected	6 = Command Failed
7 = Invalid data size	8 = Invalid fragment type
9 = Fragment Error	10 = Invalid timeout time
11 = Can’t send more	

See program example on next page.

Example Program

```
SUB SYSTEM_STARTUP
 dispmode=40
 settimer(2,0.5) ‘twice a second transmission.
END SUB

SUB SYSTEM_TIMER2
 Call SendGross1
END SUB

‘continuous output of the gross weight.
‘demonstrates serial print and ethernet print
SUB SENDGROSS1
 IF NOT MOTION THEN
  msg$ = “G “+format$(gross,6.2)+” lb”+chr$(13)
 else
  msg$ = “G “+format$(gross,6.2)+” lb M”+chr$(13)
 END IF
 print #52, msg$  ‘print msg$ out card#1, socket#2
END SUB

‘subroutine to emulate key presses on the
‘front panel via ethernet or serial port
SUB PROCESSCOM
 If left$(msg$,3)=”!F1" then
  CALL f1_KEY
 ELSEIf left$(msg$,3)=”!F2" then
  CALL f2_KEY
 ELSEIf left$(msg$,3)=”!F3" then
  CALL f3_KEY
 ELSEIf left$(msg$,3)=”!F4" then
  CALL f4_KEY
 ELSEIf left$(msg$,3)=”!F5" then
  CALL f5_KEY
 ELSEIf left$(msg$,3)=”!SK” then
  DoSelect
 ELSEIf left$(msg$,3)=”!UK” then
  DoUnits
 ELSEIf left$(msg$,3)=”!PK” then
  DoPrint
 ELSEIf left$(msg$,3)=”!TK” then
  DoTare
 ELSEIf left$(msg$,3)=”!ZK” then
  DoZero
 Else
```

```
  Cls
  Msg$=left$(msg$,len(msg$)-1) ‘remove EOM character
  Print msg$
 End if
End Sub

‘socket#1 parser
SUB SOCK1_MESSAGE
 Beep
 msg$=getsock$(1)
 CALL ProcessCom
END SUB

SUB SOCK1_OFFLINE
 Cls
 Print “Socket Offline”
END SUB

SUB SOCK1_ONLINE
 Cls
 Print “Socket Online”
END SUB
```

MODEM

Modem(cmd, PortNum, AAMode, dialTimeOut, Phone$, User$[,mode])

Cmd = 1-5
PortNum = 1,2,3,4,13(PC/104 Com3),14(PC/104 Com4)
AAMode = 0=Non-Auto Answer Mode, 1=Auto Answer mode
DialTimeOut = Amount of time the system retries before re-initializing. 1000 is equal to 1 second.
Phone$ = String representation of the phone number the system will dial.
Example: (9,15072384461)
User$ = String of modem commands. AT commands to change volume, etc…
Optional Argument:
Mode = 0: WT-BASIC mode, functions just like a serial port for transmitting data.
Mode = 1: Diagnostic mode, a menu system is echoed to the modem to select a given diagnostic function.

Comma’s create pauses, as with standard modems.

See the Service Manual for Hyperterminal settings.

See Modem.310 for examples of these functions.

Examples of Commands 1-5:
Initialize for WT-BASIC Mode:
modem(1,14,1,4000,” 9,,,15072384461",””)
Initialize for Diagnostic Mode:
modem(1,14,1,4000,” 9,,,15072384461",””,1)
Dial
modem(2,”9,,,15072384461")
Hangup
modem(3)
GetState
modem(4,100) ‘stores the state to numeric location 100
GetModem
modem(5,100) ‘stores the modem message to string location 100

Modem Response Numbers:

Initialize	*= 1*
Set auto-answer	*= 2*
Set user settings	*= 3*
Port ready (OK)	*= 4*
Dialing	*= 5*
Error	*= 6*
Connected	*= 7*
Disconnected (no carrier)	*= 8*

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 Model 1310 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.

Sample Setpoint (Cutoff) Application

```
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
```

Sample Continuous Output

```
‘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 output
‘max is 10 times a second
SUB SYSTEM_STARTUP
contout(1,2)                ‘format #1 output twice a second
END SUB
```

Sample Enquire via Serial Port

```
‘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
```

Sample for Backlite and Contrast

```
‘Sample application to demonstrate the syntax use ‘and functionality of the
“CONTRAST” and “BACKLITE” keywords
SUB SYSTEM_STARTUP
     mustdim
     dim mode
     dim contlevel
     mode = 50
     dispmode(mode)
     contlevel = 32
     contrast(contlevel)
     menu “CONT+”,”bright_lite”,” CONT-dark_lite”,””,””,””,””,”BCKLITE”,”no_lite”
     settimer(2,0.1)
END SUB

‘routine to increase display contrast
SUB bright_lite
     if contlevel < 64 then contlevel = contlevel + 1
     contrast(contlevel)
END SUB

‘routine to decrease display contrast
SUB dark_lite
     if contlevel > 0 then contlevel = contlevel - 1
     contrast(contlevel)
END SUB

‘routine to turn display backlight ON or OFF
     SUB no_lite
     if backlite = 0 then backlite(1) else backlite(0)
END SUB

‘routine to display contrast &/or backlight setting choices
‘and update the current contrast value (1-64) to the display
SUB SYSTEM_TIMER2
     cls
     dispmode(mode)
     print “       Current contrast level =”;contrast
     print “      “;chr$(34);”CONT+”;chr$(34);” = increase contrast”
     print “      “;chr$(34);”CONT-”;chr$(34);” = decrease contrast”
     print “      “;chr$(34);”BCKLITE”;chr$(34);” = backlight ON/OFF”
END SUB
```

Sample for DISPSCL

```
‘Sample application to demonstrate the syntax use ‘and functionality of the
“DISPSCL” keyword

SUB SYSTEM_STARTUP
      mustdim
      dim mode
      dim newscale
      curscale = 1
      newscale = curscale
      mode = 15
      Menu “demo”,”DISPSCL_SELECT”,””,””,””,””,””,””,””,””,””
END SUB

‘routine to display “DISPSCL” demo choices
SUB DISPSCL_SELECT
      cls
      dispmode(mode)
      print
      print “           Select a “;chr$(34);”DISPSCL”;chr$(34)
      print “   display order keyword demonstration”
      print “        from the choices below:”
      menu “NORMAL”,”NORMAL”,”REVERSE”,”REVERSE”,”EVENODD”,
            “EVEN_ODD”,” TOTAL”,”TOTAL”,” SCALE”,”SCALE”
END SUB

‘routine to display scales in “NORMAL” (top-down increment) order
SUB NORMAL
      dispscl(1,2,3,4,5,6,7,8)
      dispmode(44)
      sleep(5)
      call DISPSCL_SELECT
END SUB

‘routine to display scales in “REVERSE” (top-down decrement) order
SUB REVERSE
      dispscl(8,7,6,5,4,3,2,1)
      dispmode(44)
      sleep(5)
      call DISPSCL_SELECT
END SUB

‘routine to display scales in “EVEN/ODD” (top-down) order
SUB EVEN_ODD
      dispscl(2,4,6,8,1,3,5,7)
      dispmode(44)
      sleep(5)
      call DISPSCL_SELECT
END SUB

‘routine to display the total of all scales in a single-scale display mode
SUB TOTAL
      dispscl(0,-1,-1,-1,-1,-1,-1,-1)
      dispmode(8)
      sleep(5)
      call DISPSCL_SELECT
END SUB

‘routine to select the active scale
SUB SCALE
      newscale = curscale
      input “Enter Scale #:”, newscale
      if newscale > 0 and newscale < 9 then curscale = newscale
END SUB
```

Sample for GRAPH

```
‘Sample application to demonstrate the syntax use and functionality of the scale-
specific bargraph and checkweigher system variables

SUB SYSTEM_STARTUP
      mustdim
      dim mode
      dim scale
      dim min
      dim max
      dim above
      dim below
      dim basis
      curscale = 1
      scale = curscale
      mode = 63
      call GRAPH_MSG
END SUB

‘routine to display graph set softkey prompt message
SUB GRAPH_MSG
      cls
      dispmode(mode)
      print “ “;chr$(34);”F1";chr$(34);” = set graph(s)/”;chr$(34);”F5";chr$(34);” = set
      scale#”
      menu “  SET”,”set_scale”,””,””,””,””,””,””,” SCALE”,”pick_scale”
END SUB

SUB set_scale
      input “Enter Scale #:”,scale    ‘select scale # for graph settings
      if scale > 0 and scale < 3 then
      curscale = scale
      else
      exit sub
      end if
      ‘get the graph minimum value for the selected scale
      min = getitem(“GS”,1,(scale-1))
      ‘get the graph maximum value for the selected scale
      max = getitem(“GS”,4,(scale-1))
      ‘get the graph under value for the selected scale
      below = getitem(“GS”,2,(scale-1))
      ‘get the graph over value for the selected scale
      above = getitem(“GS”,3,(scale-1))
      ‘get the graph basis for the selected scale
      basis = getitem(“GS”,5,(scale-1))
      input “Enter Min value:”,min
      input “Enter Max value:”,max
      input “Enter Below value:”,below
      input “Enter Above value:”,above
      input “Enter Basis #:”,basis
      grbasis(basis,scale)
      setcheck(min,below,above,max,scale)
      setbar(min,max,scale)
END SUB

‘routine to select the active scale
SUB pick_scale
      scale = curscale
      input “Enter Scale #:”, scale
      if scale > 0 and scale < 3 then
      curscale = scale
      else
      scale = curscale
      end if
END SUB
```

Sample for DBASE

```
'Sample application to demonstrate the syntax use
'and functionality of structures/arrays
'and associated commands/keywords
SUB SYSTEM_STARTUP
 MUSTDIM
 DIM i
 DIM keyin$
 DIM tempid%
 DIM tempname$*16
 DIM tempaddr$*40
 DIM tempnum
 DIM temp$

 'use TYPE statement to setup a 100 record per element "database"
structure
 TYPE database
  last$*16
  first$*16
  addr$*40
  phone
  faxnum
  idnum%
 ENDTYPE

 'use DEFINE1 to define an array for a 100 customer database in
NON-VOLATILE RAM
 DEFINE1 cust[100] AS database

 'define a memory sentinel variable in NON-VOLATILE RAM
 DIM1 mem_sentinel%

 'define a serial port # selection variable in NON-VOLATILE RAM
 DIM1 portnum%

 'define a  database ADD/EDIT/DELETE/XFER Menu variable in NON-
VOLATILE RAM
 DIM1 pword$

 DIM memory
 memory = MEMOPT ' detect the physical memory available

 if mem_sentinel% <> 1 then 'setup data for initial database

  dispmode=6
  cls
  print
  print "   Initializing memory...."
  refresh
  sleep(3)

  CLEARMEM(memory) '* CAUTION: THIS IS DONE TO CLEAR ALL
NON-VOLATILE RAM IN INDICATOR ONLY! *

  portnum% = 2 'set the default Reports/Upload/Download of database
info to port #2
```

```
 'set the default setup password for the database ADD/EDIT/DELETE
as well as the database XFER Menu
  pword$ = "1310"

  '* THE FOLLOWING IS A PRE-SET LIST OF SAMPLE DATA *
  cust[0].last$    = "Smith"
  cust[0].first$   = "John"
  cust[0].addr$ = "456 Holly Ln. Tinseltown CA 55555"
  cust[0].phone    = 4445551234
  cust[0].faxnum   = 4445554321
  cust[0].idnum%   = 456

  cust[1].last$    = "Brown"
  cust[1].first$   = "James"
  cust[1].addr$ = "123 Motown Drive Detroit MI 48201"
  cust[1].phone    = 1115555678
  cust[1].faxnum   = 1115558765
  cust[1].idnum%   = 123

  cust[2].last$    = "Johnson"
  cust[2].first$   = "Bob"
  cust[2].addr$ = "789 Shady Ave. Mayberry NC 56031"
  cust[2].phone    = 7775559012
  cust[2].faxnum   = 7775552109
  cust[2].idnum%   = 789
 '* END OF A PRE-SET LIST OF SAMPLE DATA *'

  mem_sentinel% = 1 'set memory sentinel for all subsequent power-
ups
 end if

 'ensure the default Reports/Upload/Download of database info is set
to port #2
 if portnum% < 1 or portnum% > 4 then  portnum% = 2

 'set the the database XFER Menu password from NV RAM value
 SETPWD(1,pword$)

 call MAIN_MENU
END SUB

'routine for access to the Database XFER Menu
SUB SYSTEM_SETUP
 call XFER_MENU
END SUB

'routine to display the MAIN menu options
SUB MAIN_MENU
 cls
 dispmode(16)
 print "      Main Customer Database Menu"
 print "       Select from options below..."
' menu "RECALL","RECALL_MENU","ADD","CUST_ADD",
"EDIT","CUST_EDIT","DELETE","CUST_DELETE","MORE","MORE_MENU"
 menu "RECALL","RECALL_MENU","ADD","CUST_ADD","EDIT",
"CUST_EDIT","DELETE","CUST_DELETE","PWORD","SET_PASSWORD"
END SUB
```

```
'routine to display the RECALL function menu options
SUB RECALL_MENU
 cls
 dispmode(16)
 print "      Customer Database Search Menu"
 print "       Select from options below..."
 menu " ID#","CUST_ID","L_NAME","CUST_LAST","F_NAME",
"CUST_FIRST","PHONE#","CUST_PHONE"," EXIT","MAIN_MENU"
END SUB

'routine to display the XFER (port # set + data I/O) menu options
SUB XFER_MENU
 cls
 dispmode(16)
 print "  Customer Database Data Transfer Menu"
 print "      Select from options below..."
 menu "PORT #","PORTNUM_MENU","EXPORT","REPORT_MENU",
"IMPORT","GET_DATA","CLEAR","CLEAR_DATA","EXIT","MAIN_MENU"
END SUB

'routine to display the serial port # (for database I/O) selection menu
options
SUB PORTNUM_MENU
 'ensure the default Reports/Upload/Download of database info is set
to port #2
 if portnum% < 1 or portnum% > 4 then  portnum% = 2
 cls
 dispmode(15)
 print
 print "         Select a Serial Port #"
 print "    For Reports, Upload, or Download"
 print "        of database information"
 print "      (Port #";portnum%;" Currently Selected)"
 menu "PORT #1","SET_PORT1","PORT #2","SET_PORT2","PORT
#3","SET_PORT3","PORT #4","SET_PORT4"," EXIT","XFER_MENU"
END SUB

'routine to display the EXPORT (report output) menu options
SUB REPORT_MENU
 cls
 dispmode(16)
 print "          Customer Report Menu"
 print " Select Sorted order from options below"
 print "      (Port #";portnum%;" Currently Selected)"
 menu "
ID#","CUST_REPORT","L_NAME","L_NAME_REPORT","F_NAME",
"F_NAME_REPORT","PHONE#","PHONE_REPORT","EXIT","XFER_MENU"
END SUB
```

```
'routine to select a customer record via customer ID# entry
SUB CUST_ID
 tempid% = 0
 input "Customer ID#:",tempid%
 i = SEARCH(cust[].idnum%,0,99,tempid%,0)
 cls
 dispmode(15)
 if (tempid% > 0) AND (i > -1) then
  print "Customer ID#: ";cust[i].idnum%
  print cust[i].first$;" ";cust[i].last$
  print cust[i].addr$
  print "Phone#: ";cust[i].phone
  print "Fax  #: ";cust[i].faxnum
 else
  print
  print
  print "           NO RECORD FOUND"
  sleep(3)
  cls
  call RECALL_MENU
 end if
END SUB

'routine to select a customer record via customer LAST NAME entry
SUB CUST_LAST
 tempname$ = ""
 input "Last Name#:",tempname$
 i = SEARCH(cust[].last$,0,99,tempname$,-1,0)
 cls
 dispmode(15)
 if (tempname$ <> "") AND (i > -1) then
  print "Customer ID#: ";cust[i].idnum%
  print cust[i].first$;" ";cust[i].last$
  print cust[i].addr$
  print "Phone#: ";cust[i].phone
  print "Fax  #: ";cust[i].faxnum
 else
  print
  print
  print "           NO RECORD FOUND"
  sleep(3)
  cls
  call RECALL_MENU
 end if
END SUB
```

```
'routine to select a customer record via customer FIRST NAME entry
SUB CUST_FIRST
 tempname$ = ""
 input "First Name#:",tempname$
 i = SEARCH(cust[].first$,0,99,tempname$,-1,0)
 cls
 dispmode(15)
 if (tempname$ <> "") AND (i > -1) then
  print "Customer ID#: ";cust[i].idnum%
  print cust[i].first$;" ";cust[i].last$
  print cust[i].addr$
  print "Phone#: ";cust[i].phone
  print "Fax  #: ";cust[i].faxnum
 else
  print
  print
  print "           NO RECORD FOUND"
  sleep(3)
  cls
  call RECALL_MENU
 end if
END SUB

'routine to select a customer record via customer PHONE # entry
SUB CUST_PHONE
 tempnum = 0
 input "Phone(w/o -,(,)):",tempnum
 i = SEARCH(cust[].phone,0,99,tempnum,0)
 cls
 dispmode(15)
 if (tempnum > 0) AND (i > -1) then
  print "Customer ID#: ";cust[i].idnum%
  print cust[i].first$;" ";cust[i].last$
  print cust[i].addr$
  print "Phone#: ";cust[i].phone
  print "Fax  #: ";cust[i].faxnum
 else
  print
  print
  print "           NO RECORD FOUND"
  sleep(3)
  call RECALL_MENU
 end if
END SUB
```

```
'password-protected routine to ADD a customer record to the database
SUB CUST_ADD
 call PASSWORD_ENTRY
 inputopt(0,0,0)
 if not open then
  call MAIN_MENU
  exit sub
 end if
 cls
 dispmode(15)
 print
 print "Note: ";MEMLEFT;" bytes available in NV RAM"
 print "       (";SIZEOF(database);" bytes used/record)"
 print "       Press ENTER to continue..."
 i = lastkey1
 i = 0
 while (i <> 13) 'wait for ENTER keypress
  i = lastkey1
  keyin$ = inkey$
  refresh
 wend
 cls
' tempid% = 0
 i = lastkey1
 i = 0
 input "Enter New ID#:",tempid%
 if lastkey1 = 27 then 'ESCape keypress, exit
  cls
  call MAIN_MENU
  exit sub
 end if
 i = SEARCH(cust[].idnum%,0,99,tempid%,0)
 cls
 dispmode(15)
 if (i > -1) then
  print
  print
  print "          RECORD ALREADY EXISTS"
  print "         (USE 'EDIT' TO CHANGE)"
  sleep(3)
  call MAIN_MENU
  exit sub
 elseif (tempid% <= 0) then
  print
  print
  print "          INVALID ID# ENTERED!"
  sleep(3)
  call MAIN_MENU
  exit sub
 end if
 i = SEARCH(cust[].idnum%,0,99,0,0) 'find index of first "0" ID#
 if i > -1 then
  cust[i].idnum% = tempid%
  tempname$ = ""
  input "Last Name:",tempname$
  cust[i].last$ = tempname$
  tempname$ = ""
  input "First Name:",tempname$
  cust[i].first$ = tempname$
```

```
 tempaddr$ = ""
 input "Address:",tempaddr$
 cust[i].addr$ = tempaddr$
 tempnum = 0
 input "Phone#:",tempnum
 cust[i].phone = tempnum
 tempnum = 0
 input "Fax  #:",tempnum
 cust[i].faxnum = tempnum
 print "Customer ID#: ";cust[i].idnum%
 print cust[i].first$;" ";cust[i].last$
 print cust[i].addr$
 print "Phone#: ";cust[i].phone
 print "Fax  #: ";cust[i].faxnum
 end if
END SUB

'password-protected routine to EDIT an existing customer record in the
database
SUB CUST_EDIT
 call PASSWORD_ENTRY
 inputopt(0,0,0)
 if not open then
  call MAIN_MENU
  exit sub
 end if
' tempid% = 0
 i = lastkey1
 i = 0
 input "Customer ID#:",tempid%
 if lastkey1 = 27 then 'ESCape keypress, exit
  cls
  call MAIN_MENU
  exit sub
 end if
 i = SEARCH(cust[].idnum%,0,99,tempid%,0)
 cls
 dispmode(15)
 if (tempid% <= 0) OR (i < 0) then
  print
  print
  print "          RECORD DOESN'T EXIST"
  print "           (USE 'ADD' TO ADD)"
  sleep(3)
  call MAIN_MENU
  exit sub
 else
  tempname$ = cust[i].last$
  input "Last Name:",tempname$
  cust[i].last$ = tempname$
  tempname$ = cust[i].first$
  input "First Name:",tempname$
  cust[i].first$ = tempname$
  tempaddr$ = cust[i].addr$
  input "Address:",tempaddr$
  cust[i].addr$ = tempaddr$
  tempnum = cust[i].phone
  input "Phone#:",tempnum
  cust[i].phone = tempnum
```

```
  tempnum = cust[i].faxnum
  input "Fax  #:",tempnum
  cust[i].faxnum = tempnum
  print "Customer ID#: ";cust[i].idnum%
  print cust[i].first$;" ";cust[i].last$
  print cust[i].addr$
  print "Phone#: ";cust[i].phone
  print "Fax  #: ";cust[i].faxnum
 end if
END SUB

'password-protected routine to DELETE a customer record from the
database
SUB CUST_DELETE
 call PASSWORD_ENTRY
 inputopt(0,0,0)
 if not open then
  call MAIN_MENU
  exit sub
 end if
 tempid% = 0
 i = lastkey1
 i = 0
 input "Customer ID#:",tempid%
 if lastkey1 = 27 then 'ESCape keypress, exit
  cls
  call MAIN_MENU
  exit sub
 end if
 i = SEARCH(cust[].idnum%,0,99,tempid%,0)
 cls
 dispmode(15)
 if (tempid% <= 0) OR (i < 0) then
  print
  print
  print "          RECORD DOESN'T EXIST"
  sleep(3)
  call MAIN_MENU
  exit sub
 else
  cust[i].idnum% = 0
  cust[i].last$ = ""
  cust[i].first$ = ""
  cust[i].addr$ = ""
  cust[i].phone = 0
  cust[i].faxnum = 0
  print
  print
  print "            RECORD DELETED"
  sleep(3)
  call MAIN_MENU
 end if
END SUB
```

```
'send (UPLOAD) a list of customer records out of a serial port
'in increasing order per Customer ID#
SUB CUST_REPORT
 sort(cust[].idnum%,0) 'sort based on increasing Customer ID
' UPLOAD(portnum%,"cust","database",0,100) 'send all sorted cus-
tomer data records (0-99) out port
 i = SEARCH(cust[].idnum%,0,99,0,3) 'search for first record with
customer ID greater than 0
 UPLOAD(portnum%,"cust","database",i,(100 - i))  'send only non-zero
customer ID sorted data records out port
 print #portnum%, chr$(13);chr$(10);chr$(13);chr$(10) 'send 2 sets of
CR/LF's out port
 cls
 dispmode(15)
 print
 print "Sending Customer Records out of port ";portnum%;":"
 print "  (in order of ascending ID numbers)"
 print "       (Press ESCape to abort)"
 sleep(2)
END SUB

'send (UPLOAD) a list of customer records out of a serial port
'in alphabetical order per Customer last name
SUB L_NAME_REPORT
 'sort based on Customer last name (alphabetical and NOT case
sensitive)
 sort(cust[].last$,0,-1)
 'UPLOAD(portnum%,"cust","database",0,100) 'send all sorted cus-
tomer data records (0-99) out port
 i = SEARCH(cust[].last$,0,99,"",-1,3) 'search for first record with a
non-null last name
 UPLOAD(portnum%,"cust","database",i,(100 - i))  'send only non-null
last name sorted data records out port
 print #portnum%, chr$(13);chr$(10);chr$(13);chr$(10) 'send 2 sets of
CR/LF's out port
 cls
 dispmode(15)
 print
 print "Sending Customer Records out of port ";portnum%;":"
 print "  (in alphabetical order of last names)"
 print "       (Press ESCape to abort)"
 sleep(2)
END SUB
```

```
'send (UPLOAD) a list of customer records out of a serial port
'in alphabetical order per Customer first name
SUB F_NAME_REPORT
 'sort based on Customer first name (alphabetical and NOT case
sensitive)
 sort(cust[].first$,0,-1)
' UPLOAD(portnum%,"cust","database",0,100) 'send all sorted cus-
tomer data records (0-99) out port
 i = SEARCH(cust[].first$,0,99,"",-1,3) 'search for first record with a
non-null first name
 UPLOAD(portnum%,"cust","database",i,(100 - i))  'send only non-null
first name sorted data records out port
 print #portnum%, chr$(13);chr$(10);chr$(13);chr$(10) 'send 2 sets of
CR/LF's out port
 cls
 dispmode(15)
 print
 print "Sending Customer Records out of port ";portnum%;":"
 print " (in alphabetical order of first names)"
 print "       (Press ESCape to abort)"
 sleep(2)
END SUB

'send (UPLOAD) a list of customer records out of a serial port
'in increasing order per Customer phone number
SUB PHONE_REPORT
 'sort based on increasing Customer phone number
 sort(cust[].phone,0)
' UPLOAD(portnum%,"cust","database",0,100) 'send all sorted cus-
tomer data records (0-99) out port
 i = SEARCH(cust[].phone,0,99,0,3) 'search for first record with a
phone # greater than 0
 UPLOAD(portnum%,"cust","database",i,(100 - i))  'send only non-zero
phone # sorted data records out port
 print #portnum%, chr$(13);chr$(10);chr$(13);chr$(10) 'send 2 sets of
CR/LF's out port
 cls
 dispmode(15)
 print
 print "Sending Customer Records out of port ";portnum%;":"
 print "  (in order of ascending area codes)"
 print "       (Press ESCape to abort)"
 sleep(2)
END SUB

'routine to set XFER options (UPLOAD/DOWNLOAD) serial port
variable to 1
SUB SET_PORT1
 portnum% = 1
 call PORTNUM_MENU
END SUB

'routine to set XFER options (UPLOAD/DOWNLOAD) serial port
variable to 2
SUB SET_PORT2
 portnum% = 2
 call PORTNUM_MENU
END SUB
```

```
'routine to set XFER options (UPLOAD/DOWNLOAD) serial port
variable to 3
SUB SET_PORT3
 portnum% = 3
 call PORTNUM_MENU
END SUB

'routine to set XFER options (UPLOAD/DOWNLOAD) serial port
variable to 4
SUB SET_PORT4
 portnum% = 4
 call PORTNUM_MENU
END SUB

'routine to clear ALL stored customer database information
SUB CLEAR_DATA
 cls
 dispmode(6)
 print
 print "     WARNING! ALL CUSTOMER DATABASE"
 print "      INFORMATION WILL BE LOST!!!"
 print "      (Press ESCape to abort OR"
 print "       Press ENTER to continue)"
 i = lastkey1
 i = 0
 while (i <> 27) AND (i <> 13) 'wait for an ESCape OR ENTER
keypress
  i = lastkey1
  keyin$ = inkey$
  refresh
 wend
 if i = 27 then 'ESCape key was pressed; abort
  cls
  call XFER_MENU
  exit sub
 elseif i = 13 then 'ENTER key was pressed; set to import new data...
  cls
  print
  print "        CLEARING CUSTOMER RECORDS"
  print "            (Please Wait...)"
  refresh

  tempid% = mem_sentinel% 'retain memory sentinel value
  tempnum = portnum%      'retain port # setting
  temp$ = pword$          'retain the ADD/EDIT/DELETE/XFER MENU
password

  CLEARMEM(memory) '* CAUTION: THIS IS DONE TO CLEAR ALL
NON-VOLATILE RAM IN INDICATOR ONLY! *

  mem_sentinel% = tempid% 'restore memory sentinel value
  portnum% = tempnum      'restore port # setting
  pword$ = temp$          'restore the ADD/EDIT/DELETE/XFER MENU
password
```

```
  sleep(3)
  cls
  print
  print
  print "     ALL CUSTOMER RECORDS DELETED"
  sleep(3)
 end if
 call XFER_MENU
END SUB

'routine to set up for receipt (DOWNLOAD) of new customer database
records
SUB GET_DATA
 'call CLEAR_DATA
 cls
 dispmode(6)
 print
 print
 print "     Ready for New Customer Records:"
 print "      (Port #";portnum%;" Currently Selected)"
 print "        (Press ESCape to abort)"
 refresh
 DOWNLOAD(portnum%,"cust","database") 'receive new customer
records...
 cls
 call XFER_MENU
END SUB

'routine executed after a successful UPLOAD
SUB UPLOAD_OPER
 cls
 dispmode(6)
 print
 print "    Customer Records Upload Complete:"
 print "       (Press ENTER to continue)"
 i = lastkey1
 i = 0
 while (i <> 13) 'wait for ENTER keypress
  i = lastkey1
  keyin$ = inkey$
  refresh
 wend
 cls
 call REPORT_MENU
END SUB

'routine executed after an unsuccessful UPLOAD
SUB UPLOAD_ABORT
 cls
 dispmode(6)
 print
 print "     Customer Records Upload Aborted:"
 print "        (Press ESCape to Exit)"
 i = lastkey1
 i = 0
 while (i <> 27) 'wait for ESCape keypress
  i = lastkey1
  keyin$ = inkey$
  refresh
```

```
 wend
 cls
 call REPORT_MENU
END SUB

'routine executed after a successful DOWNLOAD
SUB DOWNLOAD_OPER
 cls
 dispmode(6)
 print
 print "   Customer Records Download Complete:"
 print "       (Press ENTER to continue)"
 i = lastkey1
 i = 0
 while (i <> 13) 'wait for ENTER keypress
  i = lastkey1
  keyin$ = inkey$
  refresh
 wend
 cls
 call XFER_MENU
END SUB

'routine executed after an unsuccessful DOWNLOAD
SUB DOWNLOAD_ABORT
 cls
 dispmode(6)
 print
 print "    Customer Records Download Aborted:"
 print "        (Press ESCape to Exit)"
 i = lastkey1
 i = 0
 while (i <> 27) 'wait for ESCape keypress
  i = lastkey1
  keyin$ = inkey$
  refresh
 wend
 cls
 call XFER_MENU
END SUB

'password-protected routine to change the ADD/EDIT/DELETE/XFER
MENU password
SUB SET_PASSWORD
 call PASSWORD_ENTRY
 inputopt(0,0,0)
 if not open then
  call MAIN_MENU
  exit sub
 end if
 i = lastkey1
 i = 0
 input "New Password:",temp$
 if lastkey1 = 27 then 'ESCape keypress, exit
  cls
  call MAIN_MENU
  exit sub
 end if
```

```
 pword$ = temp$   'store new password in NV RAM
 SETPWD(1,pword$) 'set also for the database XFER Menu

 cls
 print
 print "     The ADD/EDIT/DELETE/XFER MENU"
 print "     password is: ";chr$(34);pword$;chr$(34)
 sleep(5)
 cls
 call MAIN_MENU
END SUB

'routine to prompt for and check the ADD/EDIT/DELETE/XFER MENU
password
SUB PASSWORD_ENTRY
 open = 0
 temp$ = ""
 cls
 dispmode(6)
 inputopt(1,10,0)
 input "Setup Password:",temp$
 inputopt(0,0,0)
 if ucase$(temp$) = ucase$(pword$) then
  open = -1
 else
  open = 0
  print
  print "             INVALID PASSWORD!"
  print "              (ACCESS DENIED)"
  sleep(3)
  cls
 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 exist. (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 occurred, intermittent connection.
Other	Subscript Out of Range	Restart Model 1310 SimPoser.
	Out of Memory	1. You have too many applications running in Windows®. 2. Your Model 1310 SimPoser configuration file is too large.
	Permission Denied	Do a FILE, SAVE AS c:\Model 1310 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.

Message	Meaning
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.
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 missing the GOSUB.
double quotes 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.

For ERR = 24-37

ERR#	Message	Meaning
24	Unbalanced brackets	Array bracket missing
25	Array not defined	Array needs to be defined using DIM or DIM1
26	Array index error	Invalid array index
27	Duplicate TYPE define	Duplicate TYPE definition
28	AS expression	Invalid use of DEFINE. . .AS or DEFINE1. . .AS. Array of structure definition **OR** AS expected
29	Undefined structure	Structures not defined (need to use TYPE. . .ENDTYPE
30	Redefined structure	Structure already exists
31	No structure member	The structure member does not exist
32	Non-volatile RAM full	Non-volatile memory is full
33	Array size mismatch	Arrays of different sizes used or compared
34	Array type mismatch	Arrays of different types used or compared
35	Structure index error	Invalid array of structures index
36	Duplicate array name	Array name already defined
37	Duplicate variable	Variable name already defined
38	Extra array elements	More array elements have been received than were specified by the DOWNLOAD command
39	Extra Structure Records	More structure records have been received than were specified by the DOWNLOAD command

Appendix 5: ASCII Chart

Code #	Cont. Char.	Print Char.	Hex
0	NUL		00
01	SOH	☺	01
02	STX	☻	02
03	ETX	♥	03
04	EOT	♦	04
05	ENG	♣	05
06	ACK	♠	06
07	BEL		07
08	BS		08
09	HT		09
010	LF	LF	0A
011	VT	♂	0B
012	FF	FF	0C
013	CR	CR	0D
014	S0	♫	0E
015	S1	☼	0F
016	DLE	4	10
017	DC1	3	11
018	DC2	ø	12
019	DC3	Ø	13
020	DC4	ß	14
021	NAK	§	15
022	SYN		16
023	ETB	—	17
024	CAN	↑	18
025	EM	↓	19
026	SUB	→	1A
027	ESC	←	1B
028	FS	—	1C
029	GS	—	1D
030	RS	5	1E
031	US	6	1F
032	SP		20
033	!	!	21
034	"	"	22
035	#	#	23
036	$	$	24
037	%	%	25
038	&	&	26
039	'	'	27
040	(	(	28
041	)	)	29
042	*	*	2A
043	+	+	2B
044	,	,	2C
045	-	-	2D
046	.	.	2E
047	/	/	2F
048	0	0	30
049	1	1	31
050	2	2	32
051	3	3	33
052	4	4	34
053	5	5	35
054	6	6	36
055	7	7	37
056	8	8	38
057	9	9	39
058	:	:	3A
059	;	;	3B
060	<	<	3C
061	=	=	3D
062	>	>	3E
063	?	?	3F
064	@	@	40
065	A	A	41
066	B	B	42
067	C	C	43
068	D	D	44
069	E	E	45
070	F	F	46
071	G	G	47
072	H	H	48
073	I	I	49
074	J	J	4A
075	K	K	4B
076	L	L	4C
077	M	M	4D
078	N	N	4E
079	O	O	4F
080	P	P	50
081	Q	Q	51
082	R	R	52
083	S	S	53
084	T	T	54
085	U	U	55
086	V	V	56
087	W	W	57
088	X	X	58
089	Y	Y	59
090	Z	Z	5A
091	[	[	5B
092	\	\	5C
093	]	]	5D
094	^	^	5E
095	_	_	5F
096	`	`	60
097	a	a	61
098	b	b	62
099	c	c	63
0100	d	d	64
0101	e	e	65
0102	f	f	66
0103	g	g	67
0104	h	h	68
0105	i	i	69
0106	j	j	6A
0107	k	k	6B
0108	l	l	6C
0109	m	m	6D
0110	n	n	6E
0111	o	o	6F
0112	p	p	70
0113	q	q	71
0114	r	r	72
0115	s	s	73
0116	t	t	74
0117	u	u	75
0118	v	v	76
0119	w	w	77
0120	x	x	78
0121	y	y	79
0122	z	z	7A
0123	{	{	7B
0124	|	|	7C
0125	}	}	7D
0126	~	~	7E
0127	DEL	□	7F
0128	NA	Ç	
0129	NA	ü	
0130	NA	é	
0131	NA	â	
0132	NA	ä	
0133	NA	à	
0134	NA	å	
0135	NA	ç	
0136	NA	ê	
0137	NA	ë	
0138	NA	è	
0139	NA	ï	
0140	NA	î	
0141	NA	ì	
0142	NA	Ä	
0143	NA	Å	
0144	NA	É	
0145	NA	æ	
0146	NA	Æ	
0147	NA	ô	
0148	NA	ö	
0149	NA	ò	
0150	NA	û	
0151	NA	ù	
0152	NA	ÿ	
0153	NA	Ö	
0154	NA	Ü	
0155	NA	¢	
0156	NA	£	
0157	NA	¥	
0158	NA	₧	
0159	NA	ƒ	
0160	NA	╩	
0161	NA	í	
0162	NA	ó	
0163	NA	ú	
0164	NA	ñ	
0165	NA	Ñ	
0166	NA	ª	
0167	NA	º	
0168	NA	¿	
0169	NA	⌐	
0170	NA	¬	
0171	NA	½	
0172	NA	¼	
0173	NA	¡	
0174	NA	«	
0175	NA	»	
0176	NA	░	
0177	NA	▒	
0178	NA	▓	
0179	NA	┃	
0180	NA	┤	
0181	NA	╡	
0182	NA	╢	
0183	NA	╖	
0184	NA	╕	
0185	NA	╣	
0186	NA	║	
0187	NA	╗	
0188	NA	╝	
0189	NA	╜	
0190	NA	╛	
0191	NA	┐	
0192	NA	└	
0193	NA	┴	
0194	NA	┬	
0195	NA	├	
0196	NA	─	
0197	NA	┼	
0198	NA	╞	
0199	NA	╟	
0200	NA	╚	
0201	NA	╔	
0202	NA	╩	
0203	NA	╦	
0204	NA	╠	
0205	NA	═	
0206	NA	╬	
0207	NA	╧	
0208	NA	╨	
0209	NA	╤	
0210	NA	╥	
0211	NA	╙	
0212	NA	╘	
0213	NA	╒	
0214	NA	╓	
0215	NA	╫	
0216	NA	╪	
0217	NA	┘	
0218	NA	┌	
0219	NA	█	
0220	NA	▄	
0221	NA	▌	
0222	NA	▐	
0223	NA	▀	
0224	NA	α	
0225	NA	ß	
0226	NA	Γ	
0227	NA	π	
0228	NA	∑	
0229	NA	σ	
0230	NA	µ	
0231	NA	τ	
0232	NA	Φ	
0233	NA	Θ	
0234	NA	Ω	
0235	NA	δ	
0236	NA	∞	
0237	NA	ø	
0238	NA	ε	
0239	NA	∩	
0240	NA	≡	
0241	NA	±	
0242	NA	≥	
0243	NA	≤	
0244	NA	⌠	
0245	NA	⌡	
0246	NA	÷	
0247	NA	≈	
0248	NA	°	
0249	NA	•	
0250	NA	·	
0251	NA	√	
0252	NA	ⁿ	
0253	NA	²	
0254	NA	■	
0255	NA		

Appendix 6: Alphabetical Listing of WT-BASIC Commands

'
-
*
/
( )
\
+
-
=
>
<
<=
>=
< >
> <
^
ABS
ACCUM_ABORT
ACCUM_OPER
ACTVALUE
ADCCNTS
ANBASIS
AND
ASC
ASK(MSG$)
ATN
AVGSTART
AVGSTOP
BASE_OPER
BACKLITE
BEEP
BITMAP
CALCSTAT
CALDATA
CALL
CAPACITY
CAPTURE
CHECKSUM
CHR$
CINT
CIRCLE
CLEARCOM
CLEARERR
CLEAR_KEY
CLEARMEM
CLRDATA
CLS
COM1_MESSAGE
COM2_MESSAGE
COM3_MESSAGE
COM4_MESSAGE
CONTOUT
CONTRAST
COS
COUNT
COUNTER
COUNTTOT
CURSCALE
CURUNIT
CURUNIT$
CZERO
DATE$
DEFINE
DEFINE1
DIM
DIM1
DISPLAY
DISPMODE
DISPSCL
DIVISION
DOACCUM
DOBASE
DOPRINT
DOSELECT
DOT
DOTARE
DOUNITS
DOWNLOAD
DOWNLOAD_ABORT
DOWNLOAD_OPER
DOZERO
ELSE
ELSEIF
END
END IF
END SUB
END TYPE
ENTER_KEY
ENTRY_KEY
EQV
ERR
ESC_KEY
EVENTCLR
EVENTNUM
EVENTRDY
EXIT FOR
EXIT SUB
EXIT WHILE
EXP
Fx_KEY
F6- F10
FCLOSE
FDELETE
FERROR
FIND
FINDSTR
FIX
FMTPRINT
FOPEN
FOR . . TO. . NEXT
FORMAT$
FREAD
FSIZE
FWRITE
GETCOM$
GETCONV
GETITEM
GETNET
GETPORT
GETSOCK
GHOST_OPER
GOSUB . . RETURN
GOTO
GRBASIS

GROSS
GROSSTOT
HEX$
IF . . THEN. . ELSE
IMP
INKEY$
INMOTION
INPUT
INPUTOPT
INSTR
INT
ISSETPT
JULDATE$
JULIAN
KEY X, "keyname"
KEYHIT
LASTKEY
LASTKEY1
LCASE$
LEFT$
LEN
LINE
LOG
LTRIM$
m
MAP
MAXPEAK
MEMLEFT
MEMOPT
MENU
MENU1
MID$
MINPEAK
MOD
MODEM
MODEM_MESSAGE
MODEM_STATUS
MOTION
MUSTDIM
NET
NETERR
NET1_OFFLINE
NET1_ONLINE
NET2_OFFLINE
NET2_ONLINE
NETTOT
NETWORK_ABORT
NETWORK_OPER
NETWORK_STATUS
NEXT
NUMERIC_KEY
NOT
OR
OVERLD
PCWT
PCWTSAMP
PCWTZERO
PLEN
PRINT
PRINT #x
PRINT_ABORT
PRINT_KEY
PRINT_OPER
RANDOM
RAWGROSS
RAWNET
RAWTARE
RECALL
RECALL$
REFRESH
REM
RESETMAX
RESETMIN
RETURN
RIGHT$
ROC
ROUND
RTRIM$
SCOMERR
SCOMM_ERROR
SCOMDIAG
SCOMNUM
SEARCH
SELECT_OPER
SELECT_KEY
SEND MAIL
SETANLOG
SETBAR
SETCHECK
SETNET
SETPORT
SETPTOFF
SETPTON
SETPT_ACT
SETPT_DEACT
SETPWD
SETSHOW
SETTIMER
SGN
SHOWSETP
SHOWVAR
SHUTDOWN
SIN
SIZEOF
SLEEP
SOCK1_MESSAGE
SOCK1_OFFLINE
SOCK1_ONLINE
SOCK2_MESSAGE
SOCK2_OFFLINE
SOCK2_ONLINE
SOCKERR
SOCKSTAT
SORT
SPACE$
SQR
STOP1_ABORT
STOP1_OPER
STOP2_ABORT
STOP2_OPER
STORE
STORE$
STR$
SUB . . END SUB
SYSDATA
SYSTEM_EMAIL
SYSTEM_ERROR
SYSTEM_SETUP
SYSTEM_STARTUP
SYSTEM_TIMER
SYSTEM_TIMER2
SYSTEM_TIMER3

SYSTEM_TIMER4
TAN
TARE
TARE_ABORT
TARE_KEY
TARE_OPER
THEN
TIME$
TIMER
TO
TONE
TONEOFF
TRANSTOT
TRAXLE(type[,memLoc])
TRAXLE(0,memLoc)
TRAXLE(1)
TRAXLE(2)
UCASE$
UNDERLD
UNIT$
UNITS_KEY
UNITS_OPER
UNIXTIME
UPLOAD
UPLOAD_ABORT
UPLOAD_OPER
USER1...10_EVENT
VAL
VERSION$
WEND
WHILE . . WEND
XOR
ZERO_ABORT
ZERO_KEY
ZERO_OPER

Avery Weigh-Tronix

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

Avery Weigh-Tronix Canada, ULC
217 Brunswick Boulevard
Pointe Claire, QC H9R 4R7 Canada
Telephone: 514-695-0380
Toll free: 800-561-9461
Facsimile: 514-695-6820
www.weigh-tronix.ca