Scale Basic

Reference and Tutorial

Version 4.2E Date 9/6/2001

Table of Contents

Introduction

Scale Basic in a programming language that is used to modify the functions of a weight indicator. This makes it possible for the weight indicator to be customized to fit a wide variety of weighing applications

The first section of this manual describes how to use **EZ Link,** a computer program that runs on a PC computer under Microsoft's 'Windows' operating system. EZ Link connects the indicator to a PC computer. It is used to configure the weight indicator's parameters and to enter Scale Basic functions using the Windows graphical interface. Use the EZ Link section to learn how to connect the indicator to a PC computer and how to use EZ Link to configure, program, and test the weight indicator.

The **Event Driven Programming** section of the manual begins with a discussion of program design and then goes on to describe event driven programming. Scale Basic functions are controlled by a computer operating system called an "event driven executive". This means that the operation of the indicator is controlled by a combination of 'events' and 'functions'. The operating system scans for events. When an event is detected the operating system activates a scale basic function.

The **Scale Basic Tutorial** section is a tutorial on the Scale Basic language. It is a step by step demonstration on how to implement programs in Scale Basic.

The last section is a **Reference** on Scale basic. This section describes the Scale Basic instructions, built in functions, condition codes, and event types.

EZ Link

EZ Link is a Microsoft Windows compatible program that you can use to configure, program, and test the indicator. The Windows graphical user interface makes it easy to enter and view setup parameters and Scale Basic programs. The setup data can be saved to disk and it can be written to and read from the indicator.

EZ Link Requirements

- IBM PC compatible computer/laptop
- Windows 3.1 or Windows 95
- Approximately 1.6mb hard disk space
- An available serial communications port on the PC
- A cable to connect the PC to the indicator.

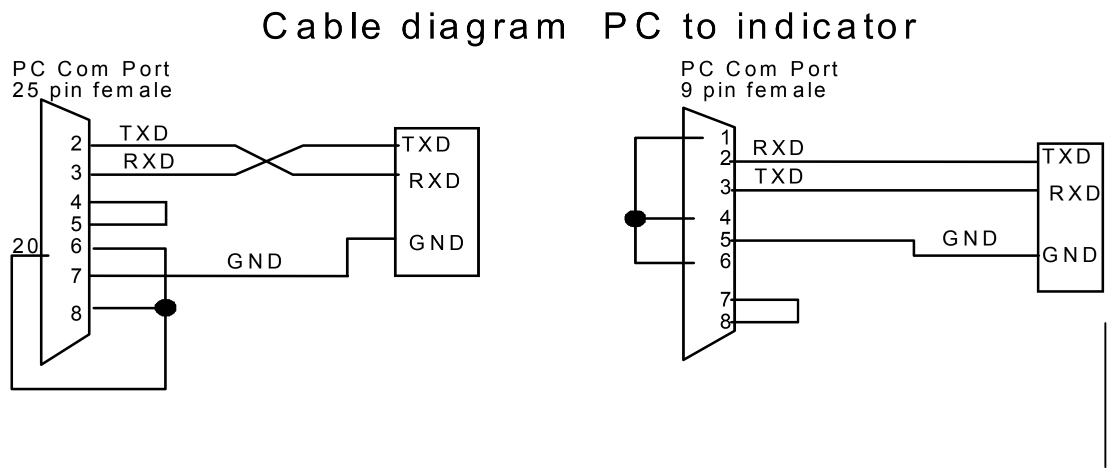
EZ Link Installation

The EZ Link program is available in 3 1/2 inch floppy diskette. It is compatible with Windows 3.1 / Windows 95.

1. Insert the EZ Link disk into diskette drive A.
2. Start the installation program:
   If you are using **Windows 3.1** select **Run** from the Program Manager File Menu.
   Type **A:\SETUP** and then click **OK**.

   If you are using **Windows 95**, select **Start**, then **Run** from the Taskbar.
   Type **A:\SETUP** and then click **OK**.
3. Follow the instructions on the screen.

Connect a cable between the indicator (Port 2) and the PC. Use the following diagram for signal connections. NOTE: There are 2 types of PC communications port connectors; 25pin female (left hand diagram) and 9pin female (right hand diagram).



EZ Link Configuration

The indicator's communications port 2 [Com Port 2] defaults to 9600 baud, 8 bits, no parity. The EZ Link program defaults to the same values. If the defaults have not been changed, then configuration is not required. Check the indicator's Com Port 2 parameters.

Parameter 25 Serial Port 2: Mode

Mode Number	Mode Description
1	7 data bits, no parity
2	7 data bits, even parity
3	7 data bits, odd parity
4	8 data bits, no parity
5	8 data bits, even parity
6	8 data bits, odd parity

Parameter 26 Serial Port 2: Baud Rate

Baud Number	Baud Description
1	9600 Baud
2	4800 Baud
3	2400 Baud
4	1200 Baud
5	300 Baud
6	150 Baud

Configure the EZ Link communication parameters to the same values as the indicator's communication parameters.

1. Double-click on the EZ Link Icon (located in the EZ Link directory) to start the EZ Link program.
2. In the **Utilities** menu, select **Setup PC Com Port Parameters**. The PC Com Port Setup menu is activated.
3. Select the port number that the communications cable is attached to.
4. Select the same baud rate, parity, data bits, and stop bits that are set into the indicator.
5. Enter the setup parameters below.

Com Port________ Baud Rate ________ Data Bits _________ Parity _________

EZ Link Test

Use this procedure to test the connection between the indicator and the PC.

On the PC:
If you are using Windows 95, access the Hyperterm program from the Start menu / Programs / Accessories / Hyper Terminal. In Windows 3.1 use the Terminal program. Configure the Hyperterm program using the parameters entered above for the indicator.

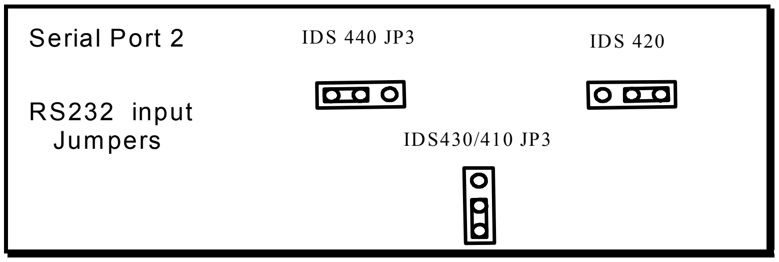
On the indicator:
Hold the CLEAR key down and press the ENTER key to enter the configure mode. Enter CFG 69. The indicator prompts “diA xx” where xx is the currently selected test number.

Test 4

Enter test number 4 and then press the ENTER key. This test displays serial data as it is received by the indictor’s serial communications port 2. The numeric display has limited alpha display capability, however numeric and some upper case alpha characters are legible. Type numeric data on the PC’s keyboard.

If there is no change on the indicator’s display:

1. Is the Hyperterm program configured for the correct PC Com Port (the one your communications cable is plugged into)? Check the Com port number used in Hyperterm and confirm that it is the same port that the cable is plugged into at the back of your computer.
2. Is the indicator receiving the data? Use a volt meter between pins 3 and 5 on the indicators port 2 terminal . The volt meter should indicate a negative voltage (approximately -5V to -12V). Type data on the PC keyboard. The volt meter should deviate when data is typed. If not, then the cable is wired incorrectly.
3. Is the RS232/RS485 jumper in the correct position for port 2?



If all data being received is unintelligible:
Are the configuration parameters for PC Terminal program the same as the parameters for the indicator (baud rate, data bits, and parity)?

Test 5

Press the indicator’s CLEAR key to exit diagnostic test 4. Enter test number 6 and then press the ENTER key. This test transmits data out of the indicator’s serial communications port 2. The display prompts: “OUT 0”.   Press the indicator’s ENTER key to send data. Serial Port 2 transmits “1234567890ABCDEF” and prompts “OUT 1” to indicate that 1 transmission has occurred. The transmitted data should appear on the PC terminal screen.

If data does not appear:
Is the indicator sending the data? Use a volt meter between pins 2 and 5 on the indicators terminal port 2. The volt meter should indicate a negative voltage (approximately -5V to -12V). Press the ENTER key on the indicator (while diagnostic 4 is active). The volt meter should deviate when data is transmitted. If not, the indicator’s RS232 transmitter is defective. Try changing the transmitter IC (U1).

Is the PC receiving the data? Check the cable wiring.

Event Driven Programming

This section provides an introduction to event driven, computer program design.

Program Design

The first step in computer programming is to create a ‘program design’. The first mistake that a programmer can make is to give insufficient attention to designing a program. Give extra attention to this phase of programming; you will be well rewarded for your effort. Programming will be easier and it will be less likely that you will have to start over again because the program doesn’t do what the customer wants.

There are many books written on program design. This manual uses the following program design process:

1. describe the application. Many times this will be provided by your customer.
2. list the results (outputs) are to be accomplished.
3. list the inputs are needed to get the results.
4. describe the program functions that are needed.

The above process should be done with close consultation with the user of the program (your customer). Usually, the customer describes the problem and the programmer takes notes and asks questions. Then the programmer writes a draft of the program design asking the customer questions as they occur. Then the customer is given a copy of the program design for approval. At this time, the programmer should go over the design with the customer to make sure that everything is covered.

Example: a customer presents the following application:

Application Description

The operator places containers on the scale and fills them with resin. An ‘Under-weight’ light turns on while the container weight is below the target weight. There are 3 sizes of container, each is filled with a different amount of resin. If the container is filled with too much resin, turn on an ‘Overweight’ light until enough resin is removed for the container to be within specifications.

Results / Outputs

Container filled with the proper amount of resin (3 sizes of container).
An ‘Under-weight’ light.
An ‘Over-weight’ light.

Inputs

Target weights (3).
Target weight select.
Start operation.

Sequence of operation

Enter target weight data: Use the indicator's built in memory register input function.

The operator sets memory registers 1, 2, and 3 for container 1, 2, and 3.

Select a target weight. Use the F1 key to initiate target weight selection. The display prompts "Con". The operator enters the container number that will be filled.

Monitor weight for over/under:

use Power On Start to initiate underweight scan by activating user function1.
Function 1 turns on Setpoint Monitor 1, and turns Under-weight light (relay 1).
Setpoint 1 triggers when the scale is no longer under weight. It activates function 2.
Function2 turns on Setpoint Monitor 2 (overweight scan) and turns on relay2, and turns off relay 1.

Setpoint 2 triggers when the scale is no longer over weight. It activates function 1.

REVIEW:
It is unlikely that the operator can fill to the exact target weight. (neither over or under).
After consulting with the customer, it is determined that the fill tolerance should be within 2% of the target weight.

MODIFIED DESIGN:

Application Description

The operator places containers on the scale and fills them with resin. An 'Under-weight' light turns on while the container weight **is more than 2%** below the target weight. There are 3 sizes of container, each is filled with a different amount of resin. If the container is filled with too much resin **(more than 2% over target weight)**, turn on an 'Overweight' light until enough resin is removed for the container to be within specifications.

Sequence of operation

Enter target weight data: Use the indicator's built in memory register input function. The operator sets memory registers 1, 2, and 3 for container 1, 2, and 3.

Select a target weight. Use the F1 key to initiate target weight selection. The display prompts "Con". The operator enters the container number that will be filled. **The over weight setpoint and underweight setpoints are calculated by adding/subtracting 2% from the target weight.**

COMMENTS:
The changes to the design are shown above in bold lettering.

Notice how each step in the program design is a refinement of the previous step.

Also note that often, missing specifications and other problems don't become apparent until well into the design process. Many times the approach will have to be changed.

Event Driven Programming

Event Driven programming is useful for ‘real time’ applications. Real time applications are those which depend on real events that happen when they happen. For example, a start switch closure, setpoint trip level, process monitoring, are real events that the computer monitors but the timing of the events are not controlled by the computer.

The most widely available programming model is the ‘sequential programming’ method. Sequential programs work best in applications where actions take place in a step by step manner. The next step in the process follows from the previous step.

In sequential programs for real time systems, the program structure starts with a main loop which scans for selected events. An event then activates a subroutine. The subroutine will execute while scanning for events that may occur while the subroutine is active. If an event occurs in a subroutine, then another subroutine is activated and whatever events that need scanning are scanned.

In an Event Driven Programming environment, events are automatically scanned for by the operating system. When an event is detected it triggers a program function. For example: part of the indicator’s event loop is to scan the keyboard. Each key is assigned a Scale Basic function to execute when the event scanner detects a key-press. Each Scale Basic function is short and executes quickly. Thus, the structure of an event driven program is: event - action, event - action, event - action, ….

In event driven program design, the program structure is centered around event-action pairs. The event scanning is taken care of in the operating systems event scanner.

The tutorial that follows uses the techniques described above for program design, and then completes the programs, using techniques of program implementation.

Program Implementation Steps:

1. Identify events and select the functions that they will activate.
2. Write program functions.
3. Test program.
4. Modify program as needed.
5. Repeat steps 3 & 4 as needed.

Design Template

To help in the design process, we will use the following design template. The first 4 sections (Application Description, Outputs, Inputs, Sequence of operation are used in program design. The remaining sections are used in program implementation.

[Application Title]

Application Description. This is a general description as given by the customer.

Outputs

list the outputs that will be produced by this application.

Inputs

list the inputs required by this application.

Sequence of operation

describe the sequence of operation of this application.

Event Function Comments

list the events used (keyboard keys, setpoint monitors, timers, communications ports)

Function Instructions Comments

write the scale basic functions needed by this application.

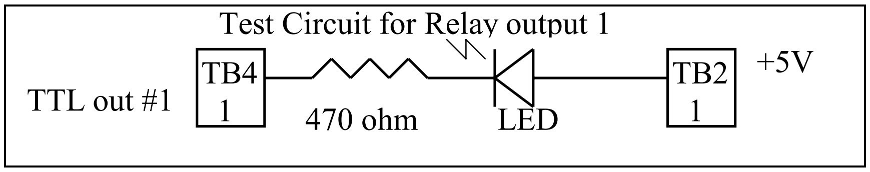
Parameter Used for

list resources used by this program (include memory registers and configuration parameters).

Scale Basic Tutorial

The following section assumes that the indicator is connected to a PC computer using EZ Link. If you are not connected, then use the keyboard entry techniques described in the user's manual.

Some of the examples used in the tutorial use the TTL Outputs (Relay outputs). Use the following circuit to view the results of the Relay out commands.



Getting Started

Most instruction books on computer languages start off with the most simple of programs: how to display "HELLO WORLD" on the computer screen.

Application Description

Display "HELLO" on the indicators numeric display.

Results / Outputs

HELLO message displayed on numeric display.

Inputs

F1 key to initiate program.

Sequence of operation

Press the F1 key. The indicator displays HELLO.

Use the above design criteria to define the events than need to be configured. The **Inputs** section of the design lists the F1 key as the event that starts the Hello program.

Event	Function	Comments
F1 key	User Function 1	The F1 key initiates the program by starting function1.

Use the sequence of operation to define User Function 1. The F1 key event triggers a function that displays "HELLO". From the list of Scale Basic Instructions we find the **Prompt** instruction is used to display messages.

Function Instructions Comments

Function	Instruction	Operand1	Operand2	Operand3	Comment
F1:	Prompt	HELLO			Display HELLO
	End				End of function

Use the template above to enter and test the program. The program can be extracted from the **Event** and **Function** sections of the template. (If you are using keyboard entry, enter the data listed in parenthesis ()).

1. Invoke the EZ Link program.
2. From the **Event** section: Set **Keyboard Events** / **F1 key** = User1 (parameter 57 = 1)
3. From the Function section: **Scale Basic** / **User 1** enter the following program:
   (parameter 72, Fn. 1) (232, 72, 69, 76, 76, 79, 255)

Instruction	Operand1	Operand2	Operand3	Comment
Prompt	HELLO			Display HELLO
End				End of function

4. In EZ Link select File, Save As, then type in "test1" then click OK. This saves the program onto disk.
5. In EZ Link select the UP (Upload) button. Click on file "test1.sf" then click on OK. If the upload function is not working, make sure that the indicator is in idle mode (not in configure mode). Try pressing the CLEAR key several times then try again. If it still doesn't work, then do the EZ Link test procedure described in this manual.
6. Press the F1 key on the indicator. "HELLO" should appear on the display.
7. Press the Enter key several times. Notice how the "HELLO" message remains on the display. Press the Clear key. The Gross weight should appear on the display.

The above program presents the programmer with a decision. The program 'works' but the user of the program may get stuck, not knowing to press the Clear key. Good programming practice assumes that the user will always do the wrong thing. To fix the above program, use the **Get key** command to wait for any key press, then force the display to a known state. Use the **Display [register]** instruction to display the gross weight after the **Get key** instruction.

Modify the function in step 3 above:

Instruction	Operand1	Operand2	Operand3	Comment
Prompt	HELLO			Display HELLO
Get key				**Wait for key-press**
Display	**Gross**			**Display gross weight**.
End				End of function

Do steps 4 through 7 above.

Arithmetic and Registers

There are 3 types of registers: general purpose registers (Memory1…15), permanent storage registers (Fixed43…50), and special purpose registers (Gross, Tare, Net, Id1, …).

The general purpose registers (**Memory1…15**) are used for temporary storage and for calculations. The Memory registers are stored in RAM memory, they can be read and written to. If power is lost, the data in the registers is lost. Scale Basic instructions Add, Sub, Mul, Div, Copy, Sign, Compare, Dp adjust, Set, and Get data can use the Memory registers.

The permanent storage registers (**Fixed43…50**) are used for permanent storage of numbers. These registers are stored in EA-ROM memory, they retain their value if power is lost. They are set by using the configure function for parameters 43…50 (EZ Link Fixed Registers). It is NOT good practice to change these registers using Scale Basic; the changes will be lost on power up or when the configure function is invoked.

The next program prompts the operator for 2 numbers, adds the numbers together, and displays the sum. It uses registers Memory1 and Memory2 to store the operator's input, and Memory3 to store the sum.

Application Description

Enter 2 numbers from the keyboard, add them together, display the sum.

Results / Outputs

Display the sum of 2 numbers.

Inputs

F1 key to begin program
Operator data entry of 2 numbers.

Sequence of operation

Press the F1 key to begin program
Enter up to 6 digits, then press the Enter key.
Enter up to 6 digits, then press the Enter key.
Add inputs to obtain sum.
The sum is displayed.

The **Inputs** section lists the F1 key to start the program and the input of 2 numbers. The input of the 2 numbers will be done in Scale Basic, the F1 key is an event that can be configured.

Event	Function	Comments
F1 key	User1	begin summing program.

Use the sequence of operation to define User Function 1. The F1 key event triggers a function that requires entry of 2 numbers. From the list of Scale Basic Instructions we find the **Get Data [r]** instruction is used to enter data. The next step is to add the 2 numbers together. The **Add [r] [a] [b]** instruction is used. Finally, the result is displayed. The **Display [r]** instruction is used.

NOTE: the program listing that follows is formatted in a more compact form than the HELLO program above. Instead of listing the instructions under Instruction, Operand 1, Operand 2, and Operand 3, the instructions and operands are listed together separated by commas.

Function	Instructions	Comments
User1	Get data, Memory1	Memory1 = Get operator input
	Get data, Memroy2	Memory2 = Get operator input
	Add, Memory3, Memory1, Memory2	Memory3 = Memory1 + Memory2
	Display, Memory3	Display sum
	End	End of function

The above function uses Memory1, 2, and 3. The memory registers used are listed in the **Parameters** section of the design template. This section of the template is used to manage resources such as memory registers, and to list parameters that need to be configured such as Fixed registers or print labels.

Parameter	**Used for**
Memory1	operator input 1
Memory2	operator input 2
Memory3	sum

Use the template above to enter the program (configure F1 key and User1 function). Upload the program, then press the F1 key to execute the program.

1. Invoke the EZ Link program.
2. Click on **Reset to Defaults** to clear previous example.
3. Set **Keyboard Events** / **F1 key** = User1
4. **Scale Basic** / **User 1** enter the following program:

Function	Instructions	Comments
User1	Get data, Memory1	Memory1 = Get operator input
	Get data, Memroy2	Memory2 = Get operator input
	Add, Memory3, Memory1, Memory2	Memory3 = Memory1 + Memory2
	Display, Memory3	Display sum
	End	End of function

6. In EZ Link select File, Save As, then type in "test2" then click OK. This saves the program onto disk.
7. In EZ Link select the UP (Upload) button. Click on file "test2.sf" then click on OK.
8. Press the F1 key on the indicator.

This program could be improved by changing the display to Gross mode after the operator presses a key. The same Display instruction can be used to change the display from Memory3 to the Gross register. However, if the Display Gross instruction immediately follows the Display Memory 3 instruction, the operator will not see the sum because it will be replaced by the Gross weight. The **Get key** instruction can be used to pause the program until a key is pressed.

Function	**Instructions**	**Comments**
User1	Get data, Memory1	Memory1 = Get operator input
	Get data, Memory2	Memory2 = Get operator input
	Add, Memory3, Memory1, Memory2	Memory3 = Memory1 + Memory2
	Display, Memory3	Display sum
	Get key	**Wait for key-press**
	Display, Gross	**Display Gross weight**
	End	End of function

Try changing the Add instruction to Sub, Mul, and Div. Also, try more complex math functions. The math functions work on any register. Try using other registers (Gross, Tare, Net, Fixed…).

Setpoint Monitors

Purpose1: to monitor 2 registers and activate a scale basic function when the lower register is greater than or equal to the upper register (trigger when: lower register >= upper register).

Purpose2: to monitor a condition code and activate a scale basic function when the condition is true or false. Set the upper register to True or False then the lower register to a condition code.

Remarks: Setpoint monitors are activated using the Scale Basic instruction: Setpoint On [x]. There are 16 setpoint monitor records that contain the following data:

Upper register [P0]	the upper value register, setpoint triggers when lower >= upper
Lower register [P1]	the lower value register
Scale Basic Function [P2}	the scale basic function to execute when lower >= upper

The next program uses 2 events: F1 key to start things off, and a setpoint monitor. This program also uses a TTL output (Relay 1). NOTE: use test circuit shown at beginning of this section.

Application Description

Drums are placed on the scale. Tare the scale, and then fill drums with material. The fill amount is fixed at 500 lb.

Outputs

Relay output 1 used to fill valve.

Inputs

F1 key starts operation
setpoint amount (500 lb)

Sequence of operation

Operator presses F1 key [Keyboard Event]
Tare the weight on the scale.
Turn on fill valve.
Monitor weight until Net >= setpoint amount [Setpoint Event]
Turn off fill valve.

From the **Inputs** section above we can fill in the **Events** section of the design template. The F1 key is used to start the fill process. Use it to activate User function 1.

The setpoint amount presents us with a new type of event, the **Setpoint Monitor**. Recall from the previous section of this manual titled “Event Driven Programming” that the program structure is event, action, event, action….

The first event is the F1 key. The action is to begin the fill process (tare the scale, turn on fill valve, turn on setpoint monitor). When the setup is complete, the process must wait for the next event (Net weight >= setpoint amount).

The Setpoint Monitor is used to generate the next event. When the setpoint event occurs, it activates user function 2 which turns off the fill valve.

Event	**Function**	**Comments**
F1 key	User 1	Start Fill operation
Setpoint 1	User 2,	trigger when Net >= Fixed43 (setpoint amount)

Configure the Setpoint 1 event with the Upper register = Fixed43 and the Lower register = Net weight register. This will cause the setpoint to trigger when the Net weight >= (greater than or equal to) the value in Fixed 43. Configure the Setpoint 1 function to execute = User 2.

Register Fixed43 is chosen because the fill amount is fixed. Data in the Fixed registers is written to EA-ROM which remains unchanged by power loss.

Function	**Instructions**	**Comments**
User 1	Copy, Tare, Gross	Tare the scale
	Relay on, 1	Turn on fill valve
	Set pt on, 1	Turn on setpoint monitor 1
	End	End of function
User 2	Relay off, 1	Turn off fill valve
	End	End of function

Parameter	**Used for**
Fixed 43	setpoint amount.

Use the template above to enter the program.

1. Invoke the EZ Link program.
2. Click on **Reset to Defaults** to clear previous program.
3. Set **Keyboard Events** / **F1 key** = User1
4. In **Setpoint Events** / **Setpoint 1** set Upper = Fixed43, Lower = Net, Function = User2
5. In **Fixed Registers** / **Fixed43** enter 500.
6. **Scale Basic** / **User 1** enter the following program:

Function	Instructions	Comments
User 1	Copy, Tare, Gross	Tare the scale
	Relay on, 1	Turn on fill valve
	Set pt on, 1	Turn on setpoint monitor 1
	End	End of function

5. **Scale Basic** / **User 2** enter the following program:

Function	Instructions	Comments
User 2	Relay off, 1	Turn off fill valve
	End	End of function

6. In EZ Link select File, Save As, then type in “test3” then click OK. This saves the program onto disk.

7. In EZ Link select the UP (Upload) button. Click on file “test3.sf” then click on OK.

8. Press the F1 key on the indicator. Press the Gross/Net key to verify that the tare weight has been read. The LED indicator should be on (Relay out 1). Add 1000 lb. to the scale using the scale simulator. The LED light should turn off.

The first improvement for this program is to automatically change the display mode to Net display.

Function	Instructions	Comments
User 1	Copy, Tare, Gross	Tare the scale
	Relay on, 1	Turn on fill valve
	Set pt on, 1	Turn on setpoint monitor 1
	Display, Net	**Display net weight**
	End	End of function

The next improvement is to get the target weight from the keyboard. First, change the setpoint monitor:

Event	**Function**	**Comments**
F1 key	User 1	Start Fill operation
Setpoint 1	User 2,	trigger when Net >= **Memory1** (setpoint amount)

Configure the Setpoint 1 event with the Upper register = **Memory1** and the Lower register = Net weight register. This will cause the setpoint to trigger when the Net weight >= the value in **Memory 1**. Configure the Setpoint 1 function to execute = User 2.

Next, modify function User 1:

Function	Instructions	Comments
User 1	**Get data, Memory1**	**Get target weight**
	Copy, Tare, Gross	Tare the scale
	Relay on, 1	Turn on fill valve
	Set pt on, 1	Turn on setpoint monitor 1
	Display, Net	Display net weight
	End	End of function

Timers

Purpose 1: to trigger a scale basic function after a set time interval.
Purpose 2: to wait an interval of time inside a scale basic function.
Remarks: Timer event monitors are activated using the Scale Basic instruction: Timer on [t]. When a timer is activated, the time interval is set into the timers countdown register. The countdown register decrements by 1 every 0.1 seconds. When the countdown reaches 0 the

Scale Basic function is executed. There are 5 timer records that contain the following data:

Time interval	time interval in tenths of a second (x0.1sec). Max = 6553.0 seconds.
Scale Basic Function	the scale basic function to execute when time-out occurs.

Timers 1-4 must be reactivated with a Timer on instruction to begin again. Timer 5 is an auto-reload timer.

Example:
As often happens when developing an application, the customer or the programmer notices that something has been left out. In the case of the Setpoint Monitors application above, the customer wants an automatic printout of the Gross, Tare, and Net weights of the batch. If the print instruction is placed after the Relay off instruction in User function 2, the printout will be inaccurate. The scale will be in motion. What we need is a time-out to allow the scale to settle. Configure Timer 1 by clicking the **Timer Events** button in EZ Link.

Timer 1 **Time** 50 **Resume** set timer 1 = 5.0 seconds, execute Resume function.

The timer activates the Resume function after a 5 second time-out. Modify User 2 function as follows:

```
User 2      Relay off, 1          Turn off fill valve
            Timer on, 1           Turn on timer 1
            Suspend                    Wait until Resume (from timer 1)
            Gosub, Print2              Print page format 2
            End                   End of function
```

The new User 2 function uses the Suspend instruction in combination with Timer1's Resume function.. The purpose of the Suspend/Resume combination is to provide a suspension of a scale basic function until an event occurs. In this case, User function 2 turns off relay 2, turns on timer 1, and then suspends processing until timer 1 times-out and executes the Resume function.

Timer5, If/Else/End if, Flags

Timer5 automatically reloads after it counts down to 0. This provides more accurate timing of repetitive events (such as pulse outputs, speed calculations, interval timing, etc.). Timer5 continues cycling until the Timer off, 5 instruction is executed.

The **If/Else/End if** instructions are used to test condition codes. If the condition code is true, then the instructions after the **If** instruction are executed. If the condition code in not true, then the instructions after the **If** instruction are skipped until an **Else** or **End if** instruction is encountered.

The **Flags** are condition codes that programmer can set or reset. The **Flags** are used to reflect a user defined state or status. In this case, **Flag1** is used to indicate the state of the LED (on/off).

Example Timer 5:

1. Set **Keyboard Events** / **F1 key** = User1
2. In **Timer Events** / **Timer5** set time = 2, function = User2

3. **Scale Basic** enter the following program:

Function	Instructions	Comments
User 1	Timer on, 5	turn on timer 5.
	End	end of function
User 2	If, Flag1	if LED flag
	Flag off, 1	turn off LED flag
	Relay off, 1	turn off LED
	Else	Else
	Flag on, 1	turn on LED flag
	Relay on, 1	turn on LED
	End if	End if
	End	End of function

4. Upload the program to the indicator. Press the F1 key. The LED flashes at 0.2 second intervals.

In the above program, the F1 key activates User 1 function, which turns on Timer 5. When timer5 times out, it simultaneously reactivates itself and activates User 2 function. The result is that User 2 function is continuously activated every 0.2 seconds. User 2 function turns the LED on if it was off, or turns the LED off if it was on.

Tutorial Conclusion

The tutorial is intended to get you started in Scale Basic programming. The next steps to take to become proficient in Scale Basic programming are:

- Read through the Reference section of this manual. Become familiar with the scale basic instructions.
- Try modifying the example programs used in the tutorial. Use instructions that were not used in the tutorial.
- Become familiar with the programs in the Application Library. Use the Application Library programs as starting points for your own program.
- Practice modifying the Application Library programs.
- Use the Scale Basic Design Template (see the appendix) to create your own application programs.

Reference

The reference is divided into 3 sections:

1. Instruction Reference: details each Scale Basic instruction and provides examples.
2. Condition Codes: details the condition codes and how they are used.
3. Built in Functions: the built in functions that can be used with Gosub and Goto instructions.

Instruction Reference

Add [r] = [a] + [b] 200 [r] [a] [b]

Purpose: add registers [a] & [b], put the result into register [r].
Remarks: add any two registers together, and put the results into a third register. Sets condition codes Positive, Negative, and Zero to reflect results.
Example:

```
Add,  Memory1, Memory2, Memory3        Memory1 = Memory2 + Memory3
Add,  Memory5, Memory5, Net            Add Net weight to register 5.
```

All off 229

Purpose: turn off all setpoints, relays, and timers.
Remarks: this instruction is usually used to turn off everything when an error occurs, or to make sure everything is off at the end of a process.

Example:

```
All off                    Turn off all setpoints, relays, and timers.
```

Beep (n) 233 [n]

Purpose: Sound the beeper [n] times.

Compare [a] - [b] 208 [a] [b]

Purpose: compare 2 registers. The condition codes (Positive, Minus, and Zero) are set as a result of subtracting register [a] from register [b], therefore:

Positive =	True if register a > register b
Negative =	True if register a < register b
Zero =	True if register a = register b

Remarks: the Compare instruction does not change the value of any registers, it only changes the value of the Positive, Negative, and Zero condition codes.

Example: register Fixed43 contains a setpoint. Turn off relay 1 if Net > Fixed43

```
Compare, Net, Fixed43
If, Positive               If Net > Setpoint Fixed43
Relay off, 1                    turn off relay 1
End if                     End if
```

```
    Test for Net >= Fixed43
                Compare, Net, Fixed43
                If not, Negative            If Net >= Setpoint Fixed43
                Relay off, 1                        turn off relay 1
                End if                      End if
```

Copy[To] [From] 206 [to] [from]

Purpose: copy the contents of register [from] to register [to]
Remarks: copy does not affect the condition codes.
Example: copy, Memory1, Gross Copy the gross weight into memory register 1

Example: copy, Memory1, Time Copy the time and date to memory register 1
Example: copy, Stime, Memory1 Copy the stored time and date to stime register for printing.
For more on time and date storage see page 49.

Dec [r] 205 [r]

Purpose: subtract 1 from a register. Set condition codes Positive, Negative, and Zero to reflect results.
Remarks: the decimal point is ignored. If Memory1 = 0.05, then Dec Memory1 = 0.04 This instruction is useful in creating loops that must be executed a fixed number of times.

Example: the following is an example of using the Dec instruction to implement a For/Next loop

```
                Set, Memory1, 25            For loop count  = 25
                Loop1                       Do
                        :
                <Do loop stuff here>
                        :
                Dec, Memory1                        loop count = loop count - 1
                If, Positive                While loop count > 0
                Next1                       Next loop
                End if                      End of For/Next loop
```

Display [r] 218 [r]

Purpose: to display the contents of a register. The registers contents continue to be displayed until another Display command is issued or the Gross/Net key on the keyboard is pressed.
Remarks: If the Gross/Net key is re-programmed to some function other than the Gross/net function, then the Gross/Net key will not clear the Display [ ] command.

Example: Display, Id1 Display ID register 1 (normally used for totals)

Div [r] = [a] / [b] 203 [r] [a] [b]

Purpose: divide register [a] by register [b], put results into register [r].

Remarks: sets condition codes Positive, Negative, and Zero to reflect results. The decimal position of the result before the divide determines the decimal position of the result after the divide. Divide by 0 sets the result to 0. NOTE: it is preferable to do a multiply instead of a divide (see Mul example).

Example: convert net weight pounds to tons. Register Fixed43 is set to 2000.

```
Dp adjust, Memory1, 2          Set memory1 decimal position = 0.00
Div, Memory1, Net, Fixed43     Memory1 = Net / Fixed43
```

Dp adjust [r] 212 [r]

Purpose: set the decimal position of a register.
Remarks: if the new decimal position is less than the previous decimal position, the result is rounded up by adding 5 to the most significant digit being dropped. Do not use a decimal position greater than 9.

Example:

```
Memory1 contains 0.425
Dp adjust, Memory1, 2          Set memory decimal position to 2 (0.00)
```

The result = (0.425 + 0.005) / 10 = 0.43

Else (see if instruction) 251

End if (see If instruction) 240

End 255

Purpose: to end a function.
Remarks: every function should have an End statement

Example:

```
Relay off, 1      Turn off relay 1
End               End of function.
```

Erase id [r] 236 [r]

Purpose: erase an Id in memory. Use the Id number in the register to find the Id and erase it.
Remarks: condition code Positive is set true if the Id is found, condition code Zero is set true if the Id is not found.

Example: Get an Id number from the keyboard, then erase the ID from memory.

```
Prompt, "ID"          Prompt the user to enter an Id number
Get data, Memory1     Get an Id number. Put it into Memory1
Erase id, Memory1     Find the Id record pointed to by Memory1. Erase it.
```

Error msg (error no.) 239 [nnn]

Purpose: to display user definable error messages on the display in the form “Err xxx” where xxx is a number between 1 and 255.

Remarks: Use this instruction to alert the operator that an error condition has occurred. The message is displayed until the operator presses any key.

Example: register Fixed43 is set to the maximum weight allowed in a weigh hopper.

```
Compare, Gross, Fixed43
If, Positive            If Gross weight > Maximum
All off                         turn off all setpoint monitors, relays, and timers
Error msg, 9                    display "Err   9"
End                             Exit function
End if                  End if over-weight error
```

Flag on / Flag off (flag no.) 210 [n]

Purpose: to turn on/off a general-purpose flag. NOTE: Flag off, 0 = turn off all flags.
Remarks: there are 9 general-purpose flags that can be set on/off using Flag on / Flag off instructions. The status of the flags can be tested using the Flag[x] condition code. Flags are useful for communicating between functions.

Example: Use the F1 key to start a timer, use the F2 key to stop the timer and display the elapsed time in seconds.

The F1 key activates user function1.

```
Flag on, 1                      Turn on general purpose flag 1
Set pt on, 1                    Turn on setpoint monitor 1
Set, Memory1, 0                 Memory1 = 0
End                             End of function
```

The F2 key activates user function 2.

```
Flag off, 1                     Turn off general-purpose flag 1.
Set Memory2, 60                 memory2 = 60
Div Memory3,Memory1,Memory2     memory3 = time / 60
Display, Memory3                Display Memory3
End                             End of function
```

Setpoint monitor 1 trips every scan cycle (60 times a second) and activates function3.

```
Inc, Memory1                    Memory1 = Memory1 + 1
If, Flag1                       If general flag 1 is on
Set pt on, 1                            re-activate setpoint monitor 1
End if                          End if
End                             End of function
```

Get data [r] 231 [r]

Purpose: get data from the keyboard and put it into a register.
Remarks: if the instruction immediately preceding the Get data instruction was the Prompt instruction, the prompt is used for prompting, otherwise "ENTER" is used for prompting.

The prompt is displayed for 0.7 seconds, then if the register is non-zero, the register data is displayed. If the register data is zero, then the prompt is displayed until the operator enters data. Condition codes Positive, Negative, and Zero are set to reflect data input. Press the Clear key to abort data entry and leave the register unchanged.. NOTE: use the Dp adjust instruction for data entry with decimal points.

Example: get the percent of moisture. The Get data instruction automatically prompts “EntEr”:

```
    Set, Memory1, 0                    Memory1=0.  The “EntEr” prompt remains until
                                               data entry.
    Dp adjust, Memory1, 2              Set the decimal position of Memory register 1
    Get data, Memory1          Get the kg/liter into Memory1
```

To use a prompt other than ENTER, use the prompt instruction.

```
    Dp adjust, Memory1, 2              Set the decimal position of Memory register 1
    Prompt, “PERCEN”           display “PErcEn”
    Get data, Memory1          get the percentage.
```

An alternative method to prompt for data uses a time delay to display the prompt, then the Get data instruction uses it’s default “ENTER” prompt. Assume Timer 1 is configured for 1 second delay:

```
    Dp adjust, Memory1, 2              Set the decimal position of Memory register 1
    Prompt, PERCEN                     display “PErcEn” on the display
    Timer on, 1                turn on timer 1 (1 seconds delay).
    Loop1                      While timer 1 on
    If, Timer1                         <keep looping>
    Next1
    End if                     End while
    Get data, Memory1          get setpoint.  Prompt is replaced by “EntEr”.
    End                        End of function
```

Get key 230

Purpose: stop processing until the operator presses a key.
Remarks: this instruction is usually used when displaying a message that requires operator acknowledgment before proceeding.

Example:
```
        Prompt, START              display “StArt”
        Get key                    wait for any key-press
```

Get id [r] 234 [r]

Purpose: use the number in register [r] to open an Id in memory.
Remarks: Condition code Positive is set true if the Id is found, condition code Zero is set true if the Id is not found.

Example:
```
        Set, Memory1, 25
```

```
Get id, Memory1          Open Id 25
If, Zero                 If the Id was not found
Error msg, 8                     display error message 8
End if                   End if
End                      End of function
```

Gosub (function no.) 243 [nnn]

Purpose: to execute a function (subroutine) from within a function and return to the calling function at the instruction after the Gosub instruction.

Remarks: a subroutine can be called from within a subroutine (this is called nesting). The maximum level of nesting is 8; i.e. function1 → function6 → function7 → function2 … maximum = 8 levels.

Example: in a bulk-weigh function the scale is zeroed, the display is set to Net mode, and relay1 is turned on.

```
Gosub, Zero              Zero the scale
Gosub, Set net           Set the display to net mode
Relay on, 1              Turn on relay 1
End                      End of function
```

Goto (function no.) 244 [nnn]

Purpose: transfer execution from one function to another.

Remarks:

Example:

```
If, Flag4                If flag4 is on
Goto, 7                          Go to function 7
End if                   End if
Set pt, On, 2            Turn on setpoint 2.  This instruction is not
                         executed if Goto was executed.
```

If (condition) / Else / End if 241 [nnn] / 251 / 240

Purpose: to control program flow using a condition code.

Remarks: the instructions after the if statement are executed if the condition code is true, the instructions after the Else statement are executed if the condition code is false. The Else part of the instruction is optional. The End if statement ends the If statement.

Example: the F1 key (activates function1) is used to toggle the display between net weight and memory1 (total). Flag1 is used to determine the current state of the display, Flag1 = True = display total.

```
If, Flag1                If current mode = display total
Flag off, 1                      turn off flag 1
Display Net                      set to display Net weight
Else                     Else <current mode not display net>
```

```
Flag on, 1                          turn on flag 1
Display Memory1                     set to display Memory1 (total)
End if                          End if
End                             End of function
```

If not(condition) / Else / End if 242 [nnn] / 251 / 240

Purpose: to control program flow using a condition code.
Remarks: sometimes it is more meaningful or more convenient to test for a false condition rather than a true condition. In these situations, use the If not instruction.

Example:

```
If not, Centerz                 If the scale is not at center of zero
Gosub Zero                          Zero the scale
End if                          End if
```

Inc [r] 204 [r]

Purpose: add 1 to a register. Set condition codes Positive, Negative, and Zero to reflect results.
Remarks: the decimal point is ignored. If Memory1 = 0.05, then Inc Memory1 = 0.06 This instruction is useful for counting the number of times something occurs.

Example:

```
Set, Memory1, 25            For loop count  = 25
Loop1                         Do
        :
<Do loop stuff here>
        :
Dec, Memory1                              loop count = loop count - 1
If, Positive                  While loop count > 0
Next1                         Next loop
End if                      End of For/Next loop
```

Index id [r] 217 [r]

Purpose: to access Id memory as an array of records. The register [r] opens records by their location in memory.

Remarks: the Get id, Make id, Write id, Erase id instructions access Id memory with randomly assigned Id numbers (the Id key). The Index id instruction accesses an Id record by it's location in memory. With this instruction you view Id memory as (approx.) 490 records: | ID record 1 | ID record 2 | ID record 3 | ……. | ID record 490 |. The condition code Zero is set true if the index number > maximum or register [r] does not exist.
NOTE: In EZ Link, the fields in an ID record are accessed via Id1, Id2, …. Do not confuse the ID record with the Id field access symbols.

Example: access ID record 3, set Id register 1 to 0, save the change to ID record 3.

```
Set Memory1, 3              Memory1 = 3
Index id, Memory1           Open ID record 3 (the 3d record in memory)
Set Id1, 0                  Set ID record 3, Id register1 to 0
Write id                    Write the most recently accessed ID to memory.
End                         End of function
```

Keybd off / Keybd on 227 / 226

Purpose: to prohibit access to the keyboard when running a critical function. The Clear key will turn the keyboard on if it has been turned off.

Remarks: the Keyboard off instruction is used to prevent accidental operator interference during a critical function. The Keyboard on instruction restores access to the keyboard functions. The Clear key is a fail-safe, in case the keyboard is disabled and accidentally not re-enabled.

Example: setpoint 1 is set to trip continuously (upper register = lower register) and activate function 1. Function 1 turns off the keyboard and reactivates setpoint 1. This effectively disables the Clear key from re-enabling the keyboard because the keyboard is turned off 60 times a second.

```
Keybd off                   disable the keyboard
Set pt on, 1                re-arm setpoint monitor 1
End                         End of  function
```

Loop1 / Next1 Loop2 / Next2 245 / 246 : 247 / 248

Purpose: the loop instruction is used to make Do/Until, Do/While, and For/Next loops.
Remarks: to have a loop within a loop, use loop1…. loop2…..next2….next1

Example: The following is an example of a Do/Until loop.

```
Loop1                       Do
Gosub Update                        Update weight display and scan events
If, Motion
Next1                        Until Scale is NOT in motion
End if
End                         End of function
```

The following is an example of a Do/While:

```
Loop1                       Do
If, Motion                    While scale is in motion
Gosub Update                        Update weight display and scan events
Next1
End if                      End Do While
End                         End of Function
```

The following is an example of a For/Next loop:

```
Set Memory1, 20
Loop1                     For Memory1 = 20 down-to 0
Index id, Memory1
Set Id1, 0                     ID[Memroy1].register 1 = 0
Dec Memory1
If Positive
Next1
End if                    End For …
End                       End of function
```

Make id [r] 235 [r]

Purpose: to open an ID record. If the ID number in register [r] in not found, open a new ID record and assign ID[r] to it.

Remarks: condition code Positive is set true if the Id is made or found, condition code Zero is set true if the Id could not be made (memory full).

Example: Get an ID number from the keyboard, open/make the ID, get tare from keyboard, save the data.

```
Get data, Memory1         Get an ID number from the keyboard
If not, Clear key         If NOT Clear key
Make id, Memory1               Open/Make ID
Get data, Id2                  put keyboard tare into ID register 2
Write id                       Save ID data
End if                    End if
End                       End of function
```

Mul [r] = [a] * [b] 202 [r] [a] [b]

Purpose: multiply register [a] by register [b], put results into register [r].

Remarks: sets condition codes Positive, Negative, and Zero to reflect results. The decimal position of the result = decimal position of register a + decimal position of register b. NOTE: it is preferable to use a multiply in place of a divide (see Div instruction example)

Example: convert net weight pounds to tons. Scale is calibrated x10 lb.

```
Set Memory1, 5
Dp adjust, Memory1, 4          Memory1 = 0.0005
Mul, Memory2, Net, Memory1     Memory2 = Net * 0.0005
Dp adjust, Memory2, 2          set result to 2 decimal positions
Display, Memory2               display tons
End                            End of function
```

Next1 (see Loop1 / Loop2 instruction) 246

Next2 (see Loop1 / Loop2 instruction) 248

Nop 254

Purpose: no-operation, this instruction does nothing.
Remarks: rarely used in scale basic.

Example:
```
Nop                         No-operation, do nothing
```

Prompt (nnn) (nnn) (nnn)… 232 [nnn] [nnn]…[0]

Purpose: display a message on the numeric display.
Remarks: this instruction is used to display status or to prompt the user for data input. If the Get data instruction immediately follows the Prompt instruction, then the prompt is used for data input. NOTE: user all capital letters. Most cap's will display legibly, lower case characters will not display legibly. The following upper case characters will not display: K, M, Q, V, W, X.

Example: prompt the user for target weight using the prompt "SETPT".
```
Prompt, "SETPT"             display "SEtPt" on the display
Get data, Memory1           get setpoint data.
```
An alternative method to prompt for data uses a time delay to display the prompt, then the Get data instruction uses it's default "ENTER" prompt. Assume Timer 1 is configured for 1 second delay:
```
Prompt, SETPT                       display "SEtPt on the display
Timer on, 1                 turn on timer 1 (1 seconds delay).
Loop1                       While timer 1 on
If, Timer1                          <keep looping>
Next1
End if                      End while
Get data, Memory1           get setpoint. Prompt is replaced by "EntEr".
End                         End of function
```

Relay off / Relay on (relay no.) 223 [n] / 222 [n]

Purpose: to turn off / on relay outputs.
Remarks: the TTL output signals are labeled Relay1, Relay2, …. The Relay off instruction sets the TTL output signal to TTL high, the Relay on instruction sets the TTL output signal to TTL low. When the TTL output is connected to a solid state relay, the TTL low level turns ON the relay.
NOTE1: Relay off, 0 turns off all relays.
NOTE2: Relay on/off, 10 gets relay number from register 10. For example
```
Set Memory10, 4             set memory10 = 4
```

```
        Relay on, 10               turn relay number contained in Memory10 on
```

Example: turn off all relays. If scale is at center of zero, turn on relay 3.

```
        Relay off, 0               turn off all relays
        If, Centerz                If scale is at center of zero
        Relay on, 3                        turn on relay 3
        End if                     End if
        End                        End of function
```

Resume (see Suspend instruction) 250

Set [r] (nnn) 209 [r] [nnn]

Purpose: set a register to a value between 0 and 255.
Remarks: values greater than 255 are replaced by the modulo of 255.

Example:

```
        Set, Memory1, 0
        Set, Tare, 100
        Set, Id1, 299              ERROR, the maximum number is 255
```

Set pt off / Set pt on (nn) 221 [nn] / 220 [nn]

Purpose: to deactivate / activate a setpoint monitor.
Remarks: the active setpoint monitors are scanned by the event monitor, the inactive setpoint monitors are not scanned. When the setpoint condition is reached in an active setpoint monitor, it activates a scale basic function, and de-activates itself. The Set pt on instruction must be executed to reactivate a setpoint monitor.
Set pt off 0 turns off all setpoint monitors.

Example: a filling operation uses relay output 1 to open a fill valve. Setpoint monitor 1 monitors for overweight conditions, Setpoint monitor 2 is set to activate function 2 when the net weight on the scale > Memory1. Function key 1 activates function 1 which begins the fill operation.

```
    Fn1 Get data, Memory1          Get the fill amount
        If not, Clear key          If Clear key NOT pressed
        Gosub Tare                         Tare the scale
        Relay on, 1                        Turn on fill valve
        Set pt on, 1                       Turn on over-weight monitor
        Set pt on, 2                       Turn on setpoint monitor 2
        End if                     End if
        End                        End of function

    Fn2 Relay off, 1               Turn off relay1
        Set pt off,  0             Turn off all Setpoint monitors
        End                        End of function
```

Sign [r] 207 [r]

Purpose: to set condition codes based on the value in register [r]. Positive is set if the register is greater than 0, Negative is set if the register is less than 0, Zero is set if the register is zero.
Remarks:

Example: Check for scale below zero.

```
Sign, Gross          set condition codes using the Gross weight register
If, Negative         If Gross weight < 0
End                       Exit
End if               End if
```

Sub [r] = [a] - [b] 201 [r] [a] [b]

Purpose: subtract register[b] from register [a], put the result into register [r].
Remarks: sets condition codes Positive, Negative, and Zero to reflect results.

Example:

```
Sub, Memory1, Memory2, Memory3          Memory1 = Memory2 - Memory3
```

Suspend / Resume 249 / 250

Purpose: Suspend stops a function from executing until a Resume is executed.
Remarks: the Suspend instruction is executed in a scale basic function, the Resume instruction is executed in an event monitor.

Example: A filling operation stops, waits 4 seconds, then stores the gross weight in Memory1. Timer 1 is configured for 4 second time-out and execute function Resume.

Timer1:	Time:	**40**
	Function:	**Resume**.

```
Fn. 1: Relay off, 1            turn off fill relay
       Timer on, 1             turn on settle timer
       Suspend                 wait for scale to settle. Timer1 executes Resume
       Copy, Memory1, Gross    Memory1 = Gross weight
```

The above example could use motion detect to wait for scale stable condition.

Set pt 1:	Upper register	**False**
	Lower register	**Motion**
	Execute function	**Resume**

```
Fn. 1: Relay off, 1            turn off fill relay
       Set pt on, 1            turn on motion detect monitor
       Suspend                 Setpoint monitor 1 executes Resume
       Copy, Memory1, Gross    Memory1 = Gross weight
```

Timer off / Timer on (n) 225 [n] / 224 [n]

Purpose: to activate / de-activate a timer.

Remarks: the active timers are scanned by the event monitor. When an active timer counts down to 0 (times out) it activates a scale basic function and de-activates itself. Timer off 0 turns off all timers. NOTE: timer 5 is a special timer. It is an 'auto-reload' timer. When it times out, it automatically restarts itself, and then executes a scale basic function. The Timer off instruction must be executed to turn off timer 5.

Example: keyboard key F1 activates function 1. Function 1 starts a relay output cycle where relay 1 turns on in 5 second intervals. Relay 1 turns off based on the number of tenths of seconds entered into Memory1.

Timer1:	Time:	**1**	time interval = 0.1 seconds
	Function	**User3**	execute function 3: turn off relay 1
Timer2:	Time:	**20**	time interval = 2.0 seconds
	Function:	**Resume**	execute Resume (used for Prompt pause)
Timer5:	Time:	**50**	time interval = 5.0 seconds
	Function:	**User2**	turn on relay1, turn on timer 1

```
Fn. 1:  All off                  Begin with all setpoints, timers, and relays off.
        Prompt "ONTIME"          prompt for on time data entry.
        Timer On, 2              turn on prompt pause timer
        Suspend                  suspend for 2 seconds
        Dp adjust, Memory1, 1    set memory1 for 1 decimal position
        Get data, Memory1        get ON time
        If not Clear key         if NOT Clear key
        Timer on 5                   turn on timer 5 (auto-reload timer)
        Goto User2                   GOTO user function 2
        End if                   end if
        End                      End of function

Fn. 2:  Relay on, 1              Turn on relay 1
        Timer on, 1              Turn on timer 1
        Copy Memory2, Memory1    copy ON time to Memory2
        End                      End of function: note timer 5 restarts itself.

Fn. 3:  Dec Memory2              subtract 1 from ON time interval
        If Positive              If ON time > 0
        Timer on, 1                  re-activate timer 1
        Else                     Else
        Relay off, 1                 Turn off relay 1
        End if                   End if
        End                      End of function
```

```
Fn. 4:  All off                turn off all timers, relays, etc.
        End                    End of function.
```

Txcom1 / Txcom2 (n) 237 [nnn] / 238 [nnn]

Purpose: to transmit a message from serial communications port 1 / port 2.

Remarks: the Txcom instructions transmit formatted page output with n = 1, 2, 3, or 4 (for pages 1, 2, 3 & 4). The Txcom instructions transmit labels if n = 32, 33, 34, 84, 85, 86, 87 where 32, … = print labels. The print labels can be fixed messages or can be embedded with register data.

Embedded Code	EZ Link Entry	Keyboard Entry	Comments
any ASCII code	<nnn>	nnn	nnn = any number 0 - 255
register, min. size	<reg. name>	253, rr	rr = register number
register, 7 char. field size	[reg. name]	254, rr	7 char field, right justified

NOTE1: The indicator waits for COM port 1 to complete its transmission i.e. nothing else is processed. If COM port 1 is unable to transmit, press the Clear key to exit.

NOTE2: COM port 2 transmits each message using interrupts (the indicator continues operation while the message is sent in the background). If a second message is sent, COM port 2 waits up to 2 seconds for the first message to complete. Message 2 aborts after waiting for 2 seconds. Use 'Tx2ready' status check before sending if all messages must be sent or if sending only if not busy.

Example: all messages sent:

```
Loop1
If not, Tx2ready
Next1
End if
Txcom2, Label90
```

Example: send only if not busy:

```
If, Tx2ready
Txcom2, Lable32
End if
```

Example: use Print label 34 to transmit ASCII -STX, Net weight, ASCII-CR. Print Net in 7 character field, right justified. Configure Print label 34: <02>[Net]<13>

keyboard entry = 02, 66, 13, 0

```
F1:   Txcom1, 34        transmit print label 34 out serial communications port 1.
      End               end of function.
```

Valid wt 219

Purpose: wait for valid, printable (hb44) weight. Reads scale and updates display.

Remarks: this instruction takes a minimum of 3 A/D conversion cycles. Press the Clear key to abort **Valid wt** command; also aborts the scale basic function using this instruction, and all calling functions (unwinds subroutine stack).

Example:
```
Valid wt        wait for valid weight.  Abort all if Clear key pressed.
Gosub Print1    print format 1
```

Write id 228

Purpose: write ID data back to ID memory.

Remarks: ID data is read into the ID registers by the Make id, Get id, and Index id instructions or by the Open id, Open new, Read first, and Read next functions. If any ID data is modified, it must be written back to ID memory to make the change permanent. The ID registers are cleared after a Write id instruction.

Example: Get an ID number from the keyboard, open/make the ID, get tare from keyboard, save the data.

```
Get data, Memory1       Get an ID number from the keyboard
If not, Clear key           If NOT Clear key
Make id, Memory1            Open/Make ID
Get data, Id2               put keyboard tare into ID register 2
Write id                        Save ID data
End if                  End if
End                     End of function
```

Condition Codes

The condition codes are used in the IF instruction to determine if the instructions following the IF are to be executed. The arithmetic condition codes are set every time a calculation instruction is performed. The Setpoint monitor condition codes are true if the monitor corresponding to the condition code is active. The following table lists the condition codes.

Altunits 109

Purpose: true if display is in alt-units mode.

Barcode 111

Purpose: true if bar-code message has been received on Com Port 1

Centerz 104

Purpose: true if scale is at center of zero

Clear key 106

Purpose: true if last key pressed was the Clear key
Remarks: the Clear key is used to abort an operation. Test for this key after data entry.
Example:

```
Get data, Memory1         Get the fill amount
If not, Clear key                 If Clear key NOT pressed
Gosub Tare                        Tare the scale
```

Enter key 105

Purpose: true if last key pressed was the Enter key.
Remarks: the Enter key is used to complete data entry or to continue to the next operation.

Flag 1…9 51…59

Purpose: true if FlagX is ON.
Remarks: the flags are used to remember a state or condition.
Example: Use the F1 key to toggle the display between Net display mode and Total (Memory1) display mode. Flag 1 is set ON to display Total, Off to display Gross.

```
Fn. 1:  If, Flag1               if Flag1 is on (ON = display total)
        Display, Gross                  set display mode to gross weight
        Else                    else (Flag is off = display gross)
        Display, Memory1        set display mode to total weight
        End if                  End if
```

Flags 10 -16

Purpose: Enables control of status LEDs on display panel (Gross, Net, Motion. Zero, Lb, Kg)
Remarks: Flag 10 is used to allow turning on and off the status LED displays, flags 11-16.
Example: Use the F1 & F2 keys to toggle on and off the all the displays. Individual LEDs could be used to show an over under or between status, high or low status as a bar graph or to emulate a masters display in a master/ slave network.

Set CFG 57 to 1. F1 key will execute user Fn 1, turn on all LEDs.
Set CFG 58 to 2. F2 key will execute user Fn 2, turn off all LEDs.

Fn. 1:	Flag on, 10	Enable LED control flag 10
	Flag on, 11	Flag on, 11 = Gross LED on
	Flag on, 12	Flag on, 12 = Net LED on
	Flag on, 13	Flag on, 13 = Motion LED on
	Flag on, 14	Flag on, 14 = Zero LED on
	Flag on, 15	Flag on, 15 = Lb LED on
	Flag on, 16	Flag on, 16 = Kg LED on
	Display, Gross	Display the gross weight
Fn. 2:	All off	Turns all flags to false and off.

NOTE: The all off instruction turns off all flags, setpoints & timers, one may use the (flag off, 10 thru 16 for individual control of each LED).

Input 1..6 61…66

Purpose: true if TTL input is idle (TTL high)
Remarks: the TTL inputs are normally high. They are often used to detect a switch closure which connects the TTL input to Ground (TTL low). **The TTL inputs use negative logic!**

Example:	If, Input5	test for input 5 idle
	If not, Input5	test for input 5 ON

Ktare / Wtare 112 / 113

Purpose: true if the tare register data was entered from the keyboard (Ktare) or read from the scale (Wtare).

Minus 102

Purpose: true if previous calculation result was negative.

Motion 107

Purpose: true if the scale is in motion.

Netmode 103

Purpose: true if display is in Net mode.

Overload 115

Purpose: true if scale status = Overload

Positive 101

Purpose: true if previous calculation result was Positive.
Remarks: Also set by Get id and Make id to signal that the instruction was successful.

Printable 108

Purpose: true if the weight registers (Gross/Net) contain printable (handbook 44) weight.

Qstatus 110

Purpose: true if communications port 2 receives an ‘I’ status from a previous status request or if communications port 2 receives a ‘V’ (valid weight) from a previously sent RQ message.
Remarks: Qstatus is used when multiple indicators are connected together in a network. The networks are normally setup in a Master/Slave configuration.
The Master unit sends out status inquires [S] to the slave units. The slave units send back an [I] if they are idle (Qstatus is turned ON) or a [B] if they are busy (Qstatus is turned OFF).
The Master units sends out an RT request to automatically read a slave units data registers (including Gross and Net registers). The Qstatus flag is set if the slave units weight registers contain printable (handbook44) weight.

Setpoint 1…15 1…15

Purpose: true if SetpointX is active.

Timer 1…5 41…45

Purpose: true if TimerX is active.

Tx2ready 114

Purpose: true if Com port 2 transmit is ready (not busy sending a previous message).

Zero 100

Purpose: true if previous calculation result was Zero.
Remarks: Also set by Get id and Make id to signal that the instruction was un-successful.

Built in Functions

Close id 142

Purpose: use after Open id or Open new to save ID data to memory.
Remarks: the Close id function and Write id instruction perform identical functions. ID data is read into the ID registers by the Make id, Get id, and Index id instructions or by the Open id, Open new, Read first, and Read next functions. If any ID data is modified, it must be written back to ID memory to make the change permanent. The ID registers are cleared after a Close id instruction.

Example:
```
Gosub, Open id          Open an existing id
Set, Id1, 0          Set Id field 1 to zero
Gosub,  Close id        save change to ID record
End
```

Gross/net 128

Purpose: toggle between Gross display and Net display modes.
Remarks: this is the default function for the Gross/Net key.

Open id 140

Purpose: get an ID number from the keyboard and then search memory for the ID record.
Remarks: If the ID is not found then "Err 21" is displayed until the operator presses a key. The function that invoked Open id, and all calling functions are automatically terminated (unwinds subroutine stack).

Example:
```
Gosub, Open id          Open an existing id
Set, Id1, 0          Set Id field 1 to zero
Gosub,  Close id        save change to ID record
```

Open new 141

Purpose: get an ID number from the keyboard and then search memory for the ID record. Open the ID if it is found, make a new ID if it is not found.

Peak clear 143

Purpose: set the Peak weight register to 0.
Remarks: the Peak weight is obtained from raw, unfiltered A/D conversions and is stored in binary format. The Peak Clear function is required to access the peak weight register.

Peak gross 156

Purpose: to set the display to Peak Gross weight mode.

Peak net 157

Purpose: to set the display to Peak Net weight mode.

Print1, Print2, Print3, Print4 133, 134, 135, 146

Purpose: to send formatted data to the printer.
Remarks: there are 4 pages of print formats that can be configured. Page 1 defaults to print the Gross weight when the display is in Gross mode, Page 2 defaults to print Gross, Tare, and Net weights when the display is in the Net mode. Print1 prints page1, Print2 prints page 2, etc.

Print mode 136

Purpose: to send formatted print pages to the printer. Send page 1 if the display is in gross mode, send page 2 if the display is in net mode.
Remarks: this is the default function for the Print key.

Pulse clear 149

Purpose: set the pulse count register to 0.
Remarks: pulses received on TTL input 6 are counted using the Count register[57]. The register can be cleared using the **Set Pulse, 0** instruction if no counts are being received instruction is executed. The **Pulse clear** function is used to guarantee that the register is set to 0 even if pulses are being received.

Read first 147

Purpose: read the first ID record in memory. Condition code **Positive** is true if successful.
Remarks: Open new and Make ID open new ID's using an ID number (key) to arrange the ID records in numerical order. Read first reads the ID record with the lowest ID number (key).

Read next 148

Purpose: read the next ID record in memory. Condition code **Positive** is true if successful.
Remarks: the Read first / Read next functions are used to scan ID memory from lowest ID to highest ID. Begin scanning using Read first, then Read next while Positive status is true.

Example: register Id1 is used for totals. Page format 4 is setup to print ID totals.

```
Read first          read the first ID in memory
Loop1               While -
If Positive           Read successful
Gosub Print4                Print total
Read next                   Read next ID in memory
Next 1              End -
End if                While
End                 End of function
```

Resume 250

Purpose: used by Setpoints and Timers to resume processing a function that has been suspended.

Remarks: the Suspend instruction stops a Scale Basic function from executing, thus allowing the Event Monitor to scan for events. The Resume function reactivates the function that suspended, at the instruction following the Suspend instruction.

Example: Timer1 is set for 4 second delay, then it activates the Resume function.

```
Timer on, 1                 Turn on timer 1
Suspend                             Suspend until Resume executed
Relay off, 6                Turn off relay 6
```

Set gross 131

Purpose: to set the display to the Gross mode.
Remarks: this function is equivalent to the instruction **Display Gross**

Set net 132

Purpose: to set the display to Net mode.
Remarks: this function is equivalent to the instruction **Display Net**

Tare 129

Purpose: put the gross weight into the tare register. Change to Net display mode.
Remarks: Condition code **Positive** is true if successful. The scale must be stable (no motion) and above zero for the Tare function to succeed.

Tx data 144

Purpose: transmit formatted data (see parameter 28) out communications port 2.
Remarks: configuration parameter 28 (see operation parameters) selects a format (AND, Condec, etc.) for transmission via communications port 2. If configuration parameter 27 is set to 0, then the format selected by parameter 28 is continuously transmitted. Do not use function Tx data if parameter 28 is set to 0.

Units 127

Purpose: toggle display mode between primary weight units and alternate weight units.
Remarks: this is the default function for the Units key.

Update 137

Purpose: update weight registers, update display, and scan for events.

Remarks: Scale Basic functions should execute quickly and terminate to allow the event scanner to process all events that occur. If you write a function that does not terminate quickly, or depends on scale status to continue, use the Update function if you need updated weight data or if a critical event might be pending.

Example: a function waits for stable weight (motion = false) before issuing a print command.

```
Loop1                   While -
If Motion                 Motion on scale
Update                          Update scale readings
Next1                   End -
End if                    While Motion
Gosub Print1            Print page 1
```

Update alt 145

Purpose: update alternate weight units registers.
Remarks: the indicator is designed provide weights in 2 units of measure: primary and alternate units. The all weight calculations are performed in the primary weight units. The alternate unit weights are only calculated as needed. If you use alternate units in Scale Basic functions, the Gosub Update alt before using the alternate weight registers.

User1…User15 1…15

Purpose: provide user programmable functions.

Zero 130

Purpose: to zero the weight on the scale.
Remarks: condition code **Positive** is true if successful. The gross weight must be within the configured zero range and the scale must be stable for this function to succeed. This is the default function for the Zero key.

Events

Keyboard Events

Purpose: to execute a Scale Basic function when a key is pressed. A key-press event occurs when a function key (PRINT, UNITS, GROSS/NET, TARE, F1, F2) is pressed.

Setpoint Monitors

Purpose 1: to monitor 2 registers and activate a scale basic function when the lower register is greater than or equal to the upper register (trigger when: lower register >= upper register).
Purpose 2: to monitor a condition code and activate a scale basic function when the condition is true or false. Set the upper register to True or False then the lower register to a condition code.
Remarks: Setpoint monitors are activated using the Scale Basic instruction: Setpoint On [x]. There are 16 setpoint monitor records that contain the following data:

Upper register [P0]	the upper value register, setpoint triggers when lower >= upper
Lower register [P1]	the lower value register
Scale Basic Function [P2}	the scale basic function to execute when lower >= upper

Use the EZ Link button ‘Setpoint Events’ or the indicator configuration function 71 to enter setpoint parameters.

Example Purpose1:

trigger user function 1 when gross weight is greater than Memory register 12 (weight above setpoint).

Upper register	Memory12
Lower register	Gross
Execute Function	User1

trigger user function 2 when Memory register 12 is greater than Gross weight (weight below setpoint).

Upper register	Gross
Lower register	Memory12
Execute Function	User2

Example Purpose2:

trigger user function 6 when scale is stable (motion false)

Upper register	False
Lower register	Motion
Execute Function	User6

trigger user function 9 when TTL input 3 is high (true, inactive)

Upper register	True
Lower register	Input3
Execute Function	User9

Timer Events

Purpose 1: to trigger a scale basic function after a set time interval.
Purpose 2: to wait an interval of time inside a scale basic function.

Remarks: Timers are activated using the Scale Basic instruction: Timer on [t]. When a timer is activated, the time interval is set into the timers countdown register. The countdown register decrements by 1 every 0.1 seconds. When the countdown reaches 0 the Scale Basic function is executed. There are 5 timer records that contain the following data:

Time interval [P0]	time interval in tenths of a second (x0.1sec). Max = 6553.0 sec.
Execute Function[P1]	the scale basic function to execute when time-out occurs.

Timers 1-4 must be reactivated with a Timer on instruction to begin again. Timer 5 is an auto-reload timer.

Timer5: automatically reloads after it counts down to 0. This provides more accurate timing of repetitive events (such as pulse outputs, speed calculations, interval timing, etc.). Timer5 continues cycling until the Timer off, 5 instruction is executed.

Example Purpose 1: trigger user function 7 after a 1 minute time-out. Configure timer 1:

Time interval	600	1 minute = 60.0 seconds
Execute Function	User7	execute user function 6

Example Purpose 2: display a message for 2 seconds then display Gross weight. Configure timer 1:

Time interval	20	set for 2.0 seconds
Execute Function	User1	execute user function 1

```
Fn. 1 Prompt,  ERROR                       Display ‘ERROR
      Timer on, 1                    turn on timer 1
      Loop1                          While
      If Timer1                            Timer 1 is on
      Next1                                   Wait
      End if                         End While
      Display, Gross                 Display Gross weight
```

Example Timer5: Pulse relay output 1 at 0.2 second intervals.
Configure F1 key to execute user function 10.
Configure Timer5 to execute function 10 after a 0.2 second time-out.

Time interval	2	set for 0.2 seconds
Execute Function	User10	execute user function 10

```
Fn10  If, Flag1                            if Flag 1 is on
      Flag off, 1                          turn off flag 1
      Relay off, 1                         turn off relay 1
      Else                           Else
```

```
        Flag on, 1                          turn on flag 1
        Relay on, 1                         turn on relay 1
        End if                      End if
```

This example will cause relay output 1 to continuously turn on and off in 0.2 second intervals.

TTL Input Events

Purpose: execute a scale basic function when a TTL input is activated (shorted to ground).

Remarks: TTL inputs 1 through 4 execute scale basic functions 1 through 4. The TTL inputs trigger when the input signal goes from TTL high to TTL low. The TTL input will not re-trigger until the TTL input returns to TTL high. NOTE: TTL inputs must be enabled by setting parameter 39 = 1.
The automatic scan of TTL inputs 1 thru 4 is useful for events that must be constantly monitored such as a STOP switch used in bulk weigh or an overload limit switch. For conditions where the TTL input is monitored on a temporary basis (such as a GO switch) use the ability of the Setpoint Monitor to trigger on a condition code.

TTL input 6: Pulse Count Input - TTL input 6 is scanned for signal pulses. The maximum pulse rate is 450 pulses / second, each pulse must be low a minimum 1.1 ms, and high a minimum of 1.1 ms. The pulses are accumulated in the **Count** register [57]. Each second, the number of pulses that occurred in that second is stored in the **Rate** register [55] (pulses/second).

Events - Serial Communications Input: Port 1 & Port 2

Serial Communications Port 1 Input

Purpose: receive data input.

Remarks: port 1 receives data into a buffer until an ACCII carriage return (13) is received. The condition code, **Barcode,** is set true when a message is received. The data is copied to a message buffer where it is held for the next data input command. Data input commands include any command where the operator would key in data and then press the Enter key. This includes ID number entry and Get data [r] instructions.

Example1: a bar code reader is connected to Port 1. The operator presses the F1 key which activates function 1.

```
Fn. 1     Gosub, Open new          open an ID or make new if not found
          Copy Id2, Gross          Id2 (the ID's tare register) = weight on the scale
          Gosub Close id           Save ID data to memory
          End                      End of function
```

Using this program, the operator presses the F1 key. The indicator prompts “Id”. The operator scans a bar code label. The bar code data is used for the ID number.

Example2: a bar code reader is used to read the tare weight for the scale. Setpoint 1 is configured to trigger when a bar-code message is received. Function 1 reads the bar-code data into the Tare register.

Setpoint1: Upper register = True, Lower register = Barcode, Function = User1.

Fn. 1:	Get data, Tare	read bar-code data into the tare register.
	End	End of function

Serial Communications Port 2 Input

Purpose: execute commands received by communications port 2.
Remarks: Communications port 2 operates by it's own event scanner. It operates in the background (unnoticed by the operator). The following commands are recognized by communications port 2:
NOTE: xxx = 3 digit number(1..255), nnnn = up to 10 digit number, <cr> = ASCII carriage return (13)

CR, CU, CV, CW

Purpose: Read and write configuration data. These commands are used by EZ Link only.

Fxxx<cr>

Purpose: execute scale basic function xxx.
Remarks: this instruction provides remote control of any indicator function by way of scale basic functions.
Example: the indicator receives F130<cr>. Function 130 is executed (Zero the scale).
The indicator receives F2<cr>. User function 2 is executed.

Nxxx<cr>

Purpose: select station number xxx.
Remarks: Port 2 can be assigned a station ID (EZ Link: I/O Ports / Station ID. Configuration parameter 27). If the station ID is between 1 and 254 then port 2 powers up in network mode. In network mode, the port 2 transmitter is turned off and no commands are accepted until a station select is received.
Example: Station ID is 125. The following commands are received:

```
N120<cr>F2<cr>
```

The indicator does nothing because its station ID has not been selected.

```
N125<cr>F5<cr>N110<cr>F10<cr>
```

The indicator is selected, executes function 5 and is then deselected when it receives the N110<cr> command.

RRxxx<cr>

Purpose: transmit contents of register xxx.

Remarks: xxx can be any valid register number. The data is transmitted in the form: nnnnnnnnn<cr> where nnnnnnnnn is the contents of register xxx, in a 9 character field, zero blanked.

Example: the net weight is 1945 lb. The indicator receives R66<cr> then it transmits the contents of the Net weight register in the form <sp><sp><sp><sp><sp>1945<cr>.

RTxxx<cr> / RQ<status>nnnn<cr>

Purpose: automatic transfer of register data from the recipient of RT command to the sender of the RT command.

Remarks: network systems are normally setup in a master/slave configuration. The master units sends an RT command to the slave unit. The slave unit returns an RQ reply (RQ<scale status><contents of register xxx><cr>) to the master unit. The RQ command sets the Qflag true if scale weight is printable (handbook 44), stores the nnnn data into register 15, , and then executes Scale Basic function 15.

Example: transfer the Net weight from station 25 to the master unit. In the Master unit configure the following:

Set print label 32 = N<M10><13>RT66<13>

Note: this uses Memory10 for station ID. If Memory10 contains 25 then the following would be sent: N25<cr>RT66<13> which translates into, select station 25, Request Transmit of register 66 (Net weight).

```
Fn. 1:    Set Memory10, 25          Memory10 = station ID 25
          Txcom2, Label32           Transmit RT request to station 25
          End                       End of function
```

The receipt of an RQ reply from the slave unit causes the slave's Net weight to be stored in register Memory15 and function 15 is automatically invoked.

```
Fn. 15:   If, Qstatus               If received data is printable weight
          Copy Memory1, Memory15        Memory1 = Unit25's Net weight
          Else                      Else
          Set Memory1, 0                Memory1 = 0
          End if                    End if
          End                       End of function
```

NOTE: On the slave station, only parameter 27 (station ID) needs to be configured. All other functions in the slave operate automatically and in the background.

RWxxx<cr>nnnn<cr>

Purpose: write nnnn data into register xxx.

Remarks: xxx can be any valid register number.

Example: the indicator receives: RW5<cr>150.00<cr>. The result is that register Memory5 now contains 150.00.

S / I

Purpose: Status / Idle pairs are used to verify that a remote station is available to accept commands.

Remarks: network systems are normally setup in a master/slave configuration. The master units sends 'S' command to the slave unit. The slave replies with an 'I' status if it is on line and available. The Master receives the 'I' as a command to execute user function 14.

Example: scan all network addresses for available stations.

1. Memory10 is used as the current station ID register.
2. Print label 32 = N<M10><13>S<13> : Select station (Register 10) then send 'S' command.
3. Timer1 is used to determine that a station is not responding. If it times out it activates function 2.
4. Function 2 decrements Memory10 and checks for end of scan.
5. F1 activates function 1 which starts the station scan.

```
Timer1:   2, 2                     Set for 0.2 sec time-out, activate function 2

Fn. 1:    Set Memory10, 0=255      Set current station ID = 255
          Goto User2               Goto user function 2
          End                      End of function

Fn. 2:    Dec Memory10             current station ID = current station ID - 1
          If positive              If not end of scan
          Timer on, 1                  reset timer 1
          Txcom1, Label32              transmit next station request
          Else                     Else
          Timer off, 1                 turn off timer 1
          Txcom1, Label32              deselect all stations by selecting station 0
          End if                   End if
          End                      End of function

Fn. 14:   Make id, Memory10        Make a new ID with station ID as the ID number
          Goto User2               Goto user function 2
          End                      End of function
```

NOTE: On the slave stations, only parameter 27 (station ID) needs to be configured. All other functions in the slave operate automatically and in the background.

X0 / X1 / X2

Purpose: to turn off continuous transmit, transmit TX2 data 1 time, or turn on continuous transmit of TX2 data.

Remarks: EZ Link, I/O Ports, TX Format (Parameter 28) selects a data format that can be sent out communications port 2. Parameter 27=0 or serial input command X0 turns off continuous transmit of TX2 data. Parameter 27 = 255 or serial input command X2 turns on continuous transmit of TX2 data. Serial input command X1 transmits TX2 data 1 time.

Registers

There are 5 types of registers: memory registers, fixed registers, scale registers, Id registers and special purpose registers.

Memory Registers

Use memory registers for calculated setpoints and for operator entry setpoints. There are 16 memory registers. To access the registers, enter the memory register number (1 to 16) and press the ENTER key. The display prompts "rEg xx" for 1 second and then displays the contents of the register. Press the CLEAR key to exit or enter a new value and press the ENTER key. The following registers have special purpose uses:

Memory register 10: used for indirect relay control. The number in register 10 can be used to turn on/off a relay.

example:	Set Memory10, 7	Set memory register 10 = 7
	Relay on, 10	Turn on Relay in register 10 = Turn on relay 7

Memory register 15: used for networking (serial input RQ command). When communications port 2 receives an RQ command, the data received is automatically put into register 15.

Reg. Number	Register Name
1	Memory Register 1
2	Memory Register 2
:	:
16	Memory Register 16

Fixed Registers

Use the fixed registers for values that will not be changed. The fixed registers are entered using configuration parameters 43-50. They are stored in EA-ROM.

Reg. Number	Register Name
43	Fixed Register 1
:	:
50	Fixed Register 8

Scale Registers

The scale registers are the gross, tare, and net weights on the scale and the alternate units registers. The alternate units are not calculated until needed. If the Alt units registers are used in scale basic, use the Update alt instruction to calculate the current value of the alternate weights.

Reg. Number	Register Name

61	Alt units - Gross weight
62	Alt units - Tare weight
63	Alt units - Net weight
64	Gross Weight
65	Tare Weight
66	Net Weight

ID Registers

Each ID record in memory has 6 registers. The register data for an ID is available when it is "opened" (instructions Get id, Make id, functions Open id and Open new). The register data is written to an ID record when it is "closed" (instruction Write id and function Close id). The Id Number register contains the ID of the currently open ID record. This register is read only, it should not be written to.

Reg. Number	Register Name
67	Id Register 1
68	Id Register 2
69	Id Register 3
70	Id Register 4
71	Id Register 5
72	Id Register 6
73	Id Number (read only)

Other Registers

The Pulse register contains the number of pulses received on TTL input 6. The Rate register contains the number of pulses received on TTL input 6 in 1 second (pulses / second).

The True and False registers are used by the Setpoint Monitors to enable triggering on condition codes. If a setpoints upper register is set to true, and the lower register is set to Motion, then the setpoint will trigger when the scale is in motion.

Reg. Number	Register Name
55	**Rate** - calculated from TTL input 6
57	**Pulse -** from TTL input 6
115	**True** - used in Setpoint Monitors
116	**False** - used in Setpoint Monitors
118	**Time** – used to copy time to reg x
119	**Date -** used to copy time to reg x
120	**Stime** – used to copy stored time to reg x
121	**Sdate** – used to copy stored date to reg x

Time / Date Logging Modification

118 Time Time Register

The Time register has the following uses:

Copy, <register>, Time
The copy instruction only works when copying from Time to a decimal register. The current time **and** date is read from the clock and then copied in BCD format to the decimal register.

Display Time

The current time is read from the clock and then copied to the display.

119 Date Date Register

The Date register has the following uses:

Display Date

The current date is read from the clock and then copied to the display.

120 Stime Stored Time register

The Stime register has the following uses:

Copy Stime, <register>

The copy instruction only works when copying from a decimal register to the Stime register. The contents of <register> is copied to the Stime **and** Sdate registers.

Display Stime

The Stime register is copied to the display.

Print the Stime register using the page formatter

Item #13 prints the contents of the Stime register.

121 Sdate Stored Date register

The Sdate register has the following uses:

Display Sdate

The Sdate register is copied to the display.

Print the Sdate register using the page formatter

Item #14 prints the contents of the Sdate register.

Error Codes

Err 0 Power on acquire zero error. Occurs when parameter 20 is set to acquire zero on power-up. If the scale is out of zero range or in motion then Err 0 occurs. Remove weight or wait for stable scale, then press Clear key.

Err 1 Keyboard error. Occurs when a key is pressed while power-on test is in progress.

Err 2 Restart Trap. The microprocessor accessed a non-existent memory location. Usual cause is electrical noise from the A/C power supply.

Err 3 Watchdog Time-out. The weight display has not been updated within the watchdog time-out period. Usual cause is A/C power glitch or static electric discharge

Err 4 Battery Error. (Optional) battery voltage level was below 2 V when the indicator was powered off.

Err 5 EAROM memory error. A checksum error has occurred when reading the configuration EAROM memory. Check calibration parameters and rewrite the EAROM memory. If error continues, replace EAROM (U15)

Err 5.1 EAROM Time-out, unable to write to EAROM memory.

Err 6 Ram memory error. Replace U22 if error continues.

Err 7 The A/D converter is "locked up". Check the load-cell wiring.

Err 8 Negative Deadload error.

Err 9 Count-by error. The entered value is not consistent with the configured count-by.

Err 11 A/D converter is not converting.

Err 12 Negative dead-load. Check loadcell wiring.

Err 13 Printer busy error. The busy signal (TB4 pin 14) is active (TTL high).

Err 14 Page Format line length error. A print line exceeded the maximum line length (135 char).

Err 15 The function selected is locked. Switch 2 on the mother board unlocks the parameters.

Err 16 Scale Basic Stack Overflow. Too many nested subroutine calls.

Err 18 Scale Basic Error. A scale basic function attempted to execute an instruction that does not exits or a Goto/Gosub instruction referenced a non-existent function.

Err 20 Double weigh-in. A weight-in was attempted on an ID that has been weighed-in but not weighed-out.

Err 21 ID not found.

Err 22 Scale Basic Prompt error. The message length is greater than the display maximum.

Err 23 ID memory full.

Error OL Calibration error or scale is over-loaded.

undEr Calibration error or scale is under-loaded (below readable weight).

Appendix A: Design Template

Application Description. This is a general description as given by the customer.

Outputs

list the outputs that will be produced by this application.

Inputs

list the inputs required by this application.

Sequence of operation

describe the sequence of operation of this application.

Event Function Comments

list the events used (keyboard keys, setpoint monitors, timers, communications ports)

Function Instructions Comments

write the scale basic functions needed by this application.

Parameter Used for

list resources used by this program (include memory registers and configuration parameters).

Operators Manual

Application Description. Begin with design description.

Installation

list parameters that are required to be setup
list cable requirements (TTL inputs, TTL outputs, Com ports)

Setup Parameters

list parameters that the supervisor is required to setup.

Operators Functions

describe how to use this application.