## Type 1: Toledo

W = 57 <STX> = 02 <CR> = 0D ? = 3F

| Register | Scale | Action |
|---|---|---|
| W-------------> | | The register requests the weight from the scale. |
| <---- <STX> $X_1X_2X_3X_4X_5$ <CR> | | ***If (Weight ≠ 0) && (Weight > 0) && (Weight <= Capacity + 9 divisions) && (Weight is Stable) then:*** The scale transmits the weight in 7 bytes where <STX> is the first byte, $X_1$ is the 2nd byte and represents the MSD (Most Significant Digit) of the weight value, etc, $X_5$ is the 6th byte and represents the LSD (Least Significant Digit) of the weight, and <CR> is the 7th and final byte. In this protocol the register decides the decimal place and units (lb or kg.) Must use leading zeros whenever necessary. |
| <---- <STX> ? $Y_1$ <CR> | | ***Else:*** The scale transmits a status response in 4 bytes where <STX> is the first byte, 3F is the 2nd byte, $Y_1$ is the 3rd byte and represents the status byte, and <CR> is the 4th and final byte. ***End if*** |

## Status Byte Definition:

| Definition | Hex | ASCII Text | Binary |
|---|---|---|---|
| **Motion** | $Y_1$ = 61 | a | 01100001 |
| **Scale at ZERO** | $Y_1$ = 70 | p | 01110000 |
| **Weight < 0** | $Y_1$ = 64 | d | 01100100 |
| **Weight > Capacity** | $Y_1$ = 62 | b | 01100010 |
| **Weight < 0 & Motion** | $Y_1$ = 65 | e | 01100101 |
| **Weight > Capacity & Motion** | $Y_1$ = 63 | c | 01100011 |

| Status Byte Formation | | |
|---|---|---|
| **Bit** | **Description** | |
| Bit 7 (MSB) | Parity Bit. | |
| Bit 6 | (Not used.) | **Always = 1**. |
| Bit 5 | Net Weight Bit. | **Gross = 0**, Net = 1. |
| Bit 4 | Zero Bit. | Zero = 1, Not Zero = 0. |
| Bit 3 | Outside Zero Range Bit. | **Within = 0**, Outside = 1. |
| Bit 2 | Negative Weight Bit. | Negative = 1, Non-Negative = 0. |
| Bit 1 | Overload Bit. | Overload = 1, Non-Overload = 0. |
| Bit 0 (LSB) | Motion Bit. | Motion = 1, Stable = 0. |

**Example 1:**

If weight is 21.30 lb and the scale is stable, it will transmit the following 7 bytes:

| 02 | 30 | 32 | 31 | 33 | 30 | 0D |
|---|---|---|---|---|---|---|
| <STX> | 0 | 2 | 1 | 3 | 0 | <CR> |

**Example 2:**

If weight is unstable, it will transmit the following 4 bytes:

| 02 | 3F | 61 | 0D |
|---|---|---|---|
| <STX> | ? | a | <CR> |

**Exceptions:** If Weight = 12345.6 then $X_1X_2X_3X_4X_5X_6$ = 123456

**<u>Type 2: NCI-ECR</u>**

W = 57 <CR> = 0D <LF> = 0A L = 4C B = 42 K = 4B G = 47 S = 53 <ETX> = 03 **.** = 2E

| Register | Scale | Action |
|---|---|---|
| W-------------><br><CR>----------> | | The register requests the weight from the scale by sending a W followed by a <CR>. |
| | <--<LF> $X_1X_2X_3X_4X_5X_6U_1U_2$ <CR> | <LF> $S_1S_2S_3$ <CR> <ETX><br>**The scale always responds to a valid request.**<br>The scale transmits the weight in 16 bytes where: |

**<LF>** is the first byte.

$\mathbf{X_1}$ to $\mathbf{X_6}$ are bytes 2 thru 7 and represent the weight value **including the decimal point.** Therefore, a weight of 3.02 will be 003.02 = $\mathbf{X_1}$ to $\mathbf{X_6}$, or a weight of 3.002 will be 03.002 = $\mathbf{X_1}$ to $\mathbf{X_6}$.

$\mathbf{U_1}$ to $\mathbf{U_2}$ are bytes 8 & 9 and represent the unit. For pounds $\mathbf{U_1}$ = 4C = "L" and $\mathbf{U_2}$ = 42 = "B"; for kilos $\mathbf{U_1}$ = 4B = "K" and $\mathbf{U_2}$ = 47 = "G".

**<CR>** is the $10^{th}$ byte.

**<LF>** is the $11^{th}$ byte.

$\mathbf{S_1}$ is the $12^{th}$ byte and is always $\mathbf{S_1}$ = 53 = "S".

$\mathbf{S_2}$ to $\mathbf{S_3}$ are the $13^{th}$ & $14^{th}$ bytes. They form the status word.

**<CR>** is the $15^{th}$ byte.

**<ETX>** is the $16^{th}$ byte.

## Status Word Definition:

| Definition | Hex | ASCII String $S_2S_3$ |
|---|---|---|
| **OK (Stable)** | $S_2$ = 30, $S_3$ = 30 | "00" |
| **Motion** | $S_2$ = 31, $S_3$ = 30 | "10" |
| **Scale at ZERO** | $S_2$ = 32, $S_3$ = 30 | "20" |
| **Weight < 0** | $S_2$ = 30, $S_3$ = 31 | "01" |
| **Weight > Capacity** | $S_2$ = 30, $S_3$ = 32 | "02" |
| **Motion & Weight < 0*** | $S_2$ = 31, $S_3$ = 30 | "11" |
| **Motion & Weight > Capacity*** | $S_2$ = 31, $S_3$ = 30 | "12" |

Note*: Whenever the Weight > **Max Capacity + 9 divisions** then the scale must transmit a "zero" weight value. This is 0.00 or 0.000 or whatever the "zero" weight needs to be for that scale's capacity/resolution/unit setting.

| Bit | $S_2$ Bit Description | $S_3$ Bit Description |
|---|---|---|
| 7 (MSB) | Parity Bit. | Parity Bit. |
| 6 | (Not used) **Always = 0**. | (Not used) **Always = 0**. |
| 5 | (Not used) **Always = 1**. | (Not used) **Always = 1**. |
| 4 | (Not used) **Always = 1**. | (Not used) **Always = 1**. |
| 3 | (Not used) **Always = 0**. | (Not used) **Always = 0**. |
| 2 | (Not used) **Always = 0**. | (Not used) **Always = 0**. |
| 1 | Zero Bit: Zero = 1, Non-Zero = 0. | Overload Bit: Overload = 1, Non-Overload = 0. |
| 0 (LSB) | Motion Bit: Motion = 1, Stable = 0. | Negative Weight Bit: Negative = 1, Non-Negative = 0. |

**<u>Example:</u>**

If weight is 21.30 lb and the scale is **Stable**, it will transmit the following 16 bytes:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0A | 30 | 32 | 31 | 2E | 33 | 30 | 4C | 42 | 0D | 0A | 53 | 30 | 30 | 0D | 03 |
| <LF> | 0 | 2 | 1 | . | 3 | 0 | L | B | <CR> | <LF> | S | 0 | 0 | <CR> | <ETX> |

**Type 3: NCI-General**

W = 57 <CR> = 0D <LF> = 0A L = 4C B = 42 K = 4B G = 47 <ETX> = 03 **.** = 2E

| Register | Scale | Action |
|---|---|---|
| W-------------><br><CR>----------> | | The register requests the weight from the scale by sending a W followed by a <CR>. |
| <--<LF> $X_1X_2X_3X_4X_5X_6U_1U_2$ <CR> | | <LF> $S_1S_2$ <CR> <ETX><br>**The scale always responds to a valid request.**<br>The scale transmits the weight in 15 bytes where: |

**<LF>** is the first byte.

$\mathbf{X_1}$ to $\mathbf{X_6}$ are bytes 2 thru 7 and represent the weight value **including the decimal point.** Therefore, a weight of 3.02 will be 003.02 = $\mathbf{X_1}$ to $\mathbf{X_6}$, or a weight of 3.002 will be 03.002 = $\mathbf{X_1}$ to $\mathbf{X_6}$.

$\mathbf{U_1}$ to $\mathbf{U_2}$ are bytes 8 & 9 and represent the unit. For pounds $\mathbf{U_1}$ = 4C = “L” and $\mathbf{U_2}$ = 42 = “B”; for kilos $\mathbf{U_1}$ = 4B = “K” and $\mathbf{U_2}$ = 47 = “G”.

**<CR>** is the 10th byte.

**<LF>** is the 11th byte.

$\mathbf{S_1}$ to $\mathbf{S_2}$ are the 12th & 13th bytes. They form the status word.

**<CR>** is the 14th byte.

**<ETX>** is the 15th byte.

## Status Word Definition:

| Definition | Hex | ASCII String $S_2S_3$ |
|---|---|---|
| **OK (Stable)** | $S_2$ = 30, $S_3$ = 30 | “00” |
| **Motion** | $S_2$ = 31, $S_3$ = 30 | “10” |
| **Scale at ZERO** | $S_2$ = 32, $S_3$ = 30 | “20” |
| **Weight < 0** | $S_2$ = 30, $S_3$ = 31 | “01” |
| **Weight > Capacity*** | $S_2$ = 30, $S_3$ = 32 | “02” |
| **Motion & Weight < 0** | $S_2$ = 31, $S_3$ = 30 | “11” |
| **Motion & Weight > Capacity*** | $S_2$ = 31, $S_3$ = 30 | “12” |

Note*: Whenever the Weight > **Max Capacity + 9 divisions** then the scale must transmit a “zero” weight value. This is 0.00 or 0.000 or whatever the “zero” weight needs to be for that scale’s capacity/resolution/unit setting.

| Bit | $S_1$ Bit Description | $S_2$ Bit Description |
|---|---|---|
| 7 (MSB) | Parity Bit. | Parity Bit. |
| 6 | (Not used) **Always = 0**. | (Not used) **Always = 0**. |
| 5 | (Not used) **Always = 1**. | (Not used) **Always = 1**. |
| 4 | (Not used) **Always = 1**. | (Not used) **Always = 1**. |
| 3 | (Not used) **Always = 0**. | (Not used) **Always = 0**. |
| 2 | (Not used) **Always = 0**. | (Not used) **Always = 0**. |
| 1 | Zero Bit: Zero = 1, Non-Zero = 0. | Overload Bit: Overload = 1, Non-Overload = 0. |
| 0 (LSB) | Motion Bit: Motion = 1, Stable = 0. | Negative Weight Bit: Negative = 1, Non-Negative = 0. |

**Example:**

If weight is 11.300 kg and the scale is **Stable**, it will transmit the following 15 bytes:

| 0A | 31 | 31 | 2E | 33 | 30 | 30 | 4B | 47 | 0D | 0A | 30 | 30 | 0D | 03 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| <LF> | 1 | 1 | . | 3 | 0 | 0 | K | G | <CR> | <LF> | 0 | 0 | <CR> | <ETX> |

## Type 4: TEC

<ENQ> = 05, <ACK> = 06, <NAK> = 15, <BEL> = 07, <DC1> = 11, <DC2> = 12, <STX> = 02, <ETX> = 03, <DEL> = 7F, <NUL> = 00

| Register | Scale | Action |
|---|---|---|
| <ENQ>-------------> | | 1) The register establishes communications by sending <ENQ>. |
| | | 2) **If (Weight is stable) then** |
| | <-------------<ACK> | The scale will transmit <ACK>. |
| | | **Else** |
| | <-------------<BEL> | The scale will transmit <BEL>. |
| | | Go back to step 1. |
| | | **End If** |
| <DC2>-------------> | | 3) The register requests the weight by sending <DC2>. |
| | | 4) Scale will transmit the following 9 bytes: |
| | <-------------<STX> | Start of text. |
| | <-------------<ID> | Identification byte defined bellow. |
| | <-------------<$W_5$> | MSD (Most significant digit) of weight data. |
| | <-------------<$W_4$> | 2nd MSD (Most significant digit) of weight data. |
| | <-------------<$W_3$> | 3rd MSD (Most significant digit) of weight data. |
| | <-------------<$W_2$> | 4th MSD (Most significant digit) of weight data. |
| | <-------------<$W_1$> | LSD (Least significant digit) of weight data. |
| | <-------------<BCC> | Block check character defined below. |
| | <-------------<ETX> | End of text. |
| | | 5) **If (Register verified data correctly) then** |
| <ACK>-------------> | | The register will transmit <ACK>. |
| | | Go back to step 1. |
| | | **Else** |
| | | Go back to step 1. |
| | | **End If** |

### Identifier Byte Definition

| <ID> | ASCII Text | Description |
|---|---|---|
| **7F** | <DEL> | If (Weight < 0) OR (Weight > Max Capacity + 9 divisions) then<br><ID> = 7F<br>$W_5 = W_4 = W_3 = W_2 = W_1 = 30$; however, $W_1$ or $W_5$ can be <NUL> sometimes (see below.)<br>Else<br>Follow the <ID> codes below.<br>End If |
| **41** | A | **Not Used** |
| **42** | B | **Not Used** |
| **43** | C | **Not Used** |
| **44** | D | **Not Used** |
| **45** | E | For 120 lb or 300 lb scales with 2 decimal places: Format **0.00 lb** |
| **46** | F | **Not Used** |
| **47** | G | For 600 lb or 120 kg or 300 kg or 60 kg scales. |

### <BCC> Definition

The BCC character is formed by performing an XOR operation on the following 6 bytes: <ID>, $W_5$, $W_4$, $W_3$, $W_2$, and $W_1$

**<BCC> = <ID>** XOR $\mathbf{W_5}$ XOR $\mathbf{W_4}$ XOR $\mathbf{W_3}$ XOR $\mathbf{W_2}$ XOR $\mathbf{W_1}$

See the examples below for the formation of the BCC character.

**Example 1:**

If weight is 250.05 lb, you are using a 300 x 0.05 lb scale, the scale is **Stable**, and F16 = Even, it will transmit the following 9 bytes:

| 02 | 45 | 32 | 35 | 30 | 30 | 35 | 77 | 03 |
|---|---|---|---|---|---|---|---|---|
| <STX> | <ID> | 2 | 5 | 0 | 0 | 5 | <BCC> | <ETX> |

**Example 2:**

If weight is 39.55 lb, you are using a 300 x 0.01 lb scale, the scale is **Stable**, and F16 = Even, it will transmit the following 9 bytes:

| 02 | 45 | 0 | 33 | 39 | 35 | 35 | 4F | 03 |
|---|---|---|---|---|---|---|---|---|
| <STX> | <ID> | <NUL> | 3 | 9 | 5 | 5 | <BCC> | <ETX> |

**Example 3:**

If weight is -5.01 lb, you are using a 300 x 0.05 lb scale, the scale is **Stable**, and F16 = Even, it will transmit the following 9 bytes:

| 02 | 7F | 30 | 30 | 30 | 30 | 30 | 4F | 03 |
|---|---|---|---|---|---|---|---|---|
| <STX> | <ID> | 0 | 0 | 0 | 0 | 0 | <BCC> | <ETX> |

## Type 5: Easy Weigh Protocol

R = 52 <STX> = 02 <CR> = 0D R = 52 L = 4C B = 42 K = 4B G = 47 **.** = 2E F = 46 W = 57 Z = 5A
<DC1> = 11 <DC2> = 12 <DC3> = 13 <DC4> = 14 S = 53 <EOT> = 04

| Register | Scale | Action |
|---|---|---|
| `R------------->` | | The register requests the Raw A/D Counts from the scale. |
| `<----` <STX> $X_1X_2X_3X_4X_5X_6$ <CR> | | The scale transmits the Raw A/D Counts in 8 bytes where <STX> is the first byte, $X_1$ is the 2nd byte and represents the MSD (Most Significant Digit) of the data, etc, $X_6$ is the 7th byte and represents the LSD (Least Significant Digit) of the data, and <CR> is the 8th and final byte. **The Raw A/D Counts are like the counts displayed in F9 CAL mode.** |

**Example:**

If the Raw A/D counts are 22,130 counts then it will transmit the following 8 bytes:

| 02 | 30 | 32 | 32 | 31 | 33 | 30 | 0D |
|---|---|---|---|---|---|---|---|
| <STX> | 0 | 2 | 2 | 1 | 3 | 0 | <CR> |

---

`F------------->` The register requests All Displays Data from the scale.

`<----` <STX> $W_1W_2W_3W_4W_5W_6W_7$ $Z_1Z_2$ $U_1U_2U_3U_4U_5U_6U_7$ $X_1X_2X_3X_4X_5X_6X_7$ $T_1T_2T_3T_4T_5T_6T_7$ $P_1P_2P_3P_4P_5P_6$ <CR>

**The scale transmits the data in 38 bytes where:**
<STX> is the 1st byte
$W_1$ to $W_7$ are the 2nd to 8th bytes and represent the Weight value
$Z_1$ to $Z_2$ are the 9th and 10th bytes and are either "LB" or "KG"
$U_1$ to $U_7$ are the 11th to 17th bytes and represent the Unit Price
$X_1$ to $X_7$ are the 18th to 24th bytes and represent the Total Price
$T_1$ to $T_7$ are the 25th to 31st bytes and represent the Tare
$P_1$ to $P_6$ are the 32nd to 37th bytes and represent the PLU Number
<CR> is the 38th byte

**Example:**

If the Weight = 22.005 lb, Unit Price = $1.99, Total Price = $43.79, Tare = 0.010, and the PLU Number = 4, then it will transmit the following 38 bytes:

| 02 | 30 | 32 | 32 | 2E | 30 | 30 | 35 | 4C | 42 | 30 | 30 | 30 | 31 | 2E | 39 | 39 | 30 | 30 | 34 | 33 | 2E | 37 | 39 | 30 | 30 | 30 | 2E | 30 | 31 | 30 | 30 | 30 | 30 | 30 | 30 | 34 | 0D |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| <STX> | 0 | 2 | 2 | . | 0 | 0 | 5 | L | B | 0 | 0 | 0 | 1 | . | 9 | 9 | 0 | 0 | 4 | 3 | . | 7 | 9 | 0 | 0 | 0 | . | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | <CR> |

**Note:** If you are not using a PLU or Speed Key on the scale, then the PLU Number = 0.

---

`<DC1>------------->` The register requests the CZP data from the scale. This is the Raw A/D counts with no load on the scale.

`<----` <STX> $X_1X_2X_3X_4X_5X_6$ <CR> The scale transmits the CZP in 8 bytes where <STX> is the first byte, $X_1$ is the 2nd byte and represents the MSD (Most Significant Digit) of the data, etc, $X_6$ is the 7th byte and represents the LSD (Least Significant Digit) of the data, and <CR> is the 8th and final byte.

**Example:**

If the CZP is 2,542 counts then it will transmit the following 8 bytes:

| 02 | 30 | 30 | 32 | 35 | 34 | 32 | 0D |
|---|---|---|---|---|---|---|---|
| <STX> | 0 | 0 | 2 | 5 | 4 | 2 | <CR> |

| | |
|---|---|
| <DC2>-------------> | The register requests the CSP data from the scale. This is the Calibrated Span Point of the scale and is expressed as the Raw A/D counts with the full capacity load on the scale. The CSP should not be Zero Adjusted. |
| <---- <STX> $X_1X_2X_3X_4X_5X_6$ <CR> | The scale transmits the CSP in 8 bytes where <STX> is the first byte, $X_1$ is the $2^{nd}$ byte and represents the MSD (Most Significant Digit) of the data, etc, $X_6$ is the $7^{th}$ byte and represents the LSD (Least Significant Digit) of the data, and <CR> is the $8^{th}$ and final byte. |

**Example:**

If the CSP is 202,542 counts then it will transmit the following 8 bytes:

| 02 | 32 | 30 | 32 | 35 | 34 | 32 | 0D |
|---|---|---|---|---|---|---|---|
| <STX> | 2 | 0 | 2 | 5 | 4 | 2 | <CR> |

---

| | |
|---|---|
| <DC3>-------------> | The register requests to set the CZP data from the scale. |
| <------------------<ACK> | The scale transmits the <ACK>. |
| <STX> $X_1X_2X_3X_4X_5X_6$ <CR>-------> | The register transmits the CZP in 8 bytes where <STX> is the first byte, $X_1$ is the $2^{nd}$ byte and represents the MSD (Most Significant Digit) of the data, etc, $X_6$ is the $7^{th}$ byte and represents the LSD (Least Significant Digit) of the data, and <CR> is the $8^{th}$ and final byte. |
| <------------------<ACK> | The scale transmits the <ACK> if the CZP was stored and accepted by the scale; otherwise, it will transmit a <NAK>. |

---

| | |
|---|---|
| <DC4>-------------> | The register requests to set the CSP data from the scale. |
| <------------------<ACK> | The scale transmits the <ACK>. |
| <STX> $X_1X_2X_3X_4X_5X_6$ <CR>-------> | The register transmits the CSP in 8 bytes where <STX> is the first byte, $X_1$ is the $2^{nd}$ byte and represents the MSD (Most Significant Digit) of the data, etc, $X_6$ is the $7^{th}$ byte and represents the LSD (Least Significant Digit) of the data, and <CR> is the $8^{th}$ and final byte. |
| <------------------<ACK> | The scale transmits the <ACK> if the CSP was stored and accepted by the scale; otherwise, it will transmit a <NAK>. |

---

| | |
|---|---|
| Z-------------> | The register requests the ZP data from the scale. This is the current Zero Point of the scale. |
| <---- <STX> $X_1X_2X_3X_4X_5X_6$ <CR> | The scale transmits the ZP in 8 bytes where <STX> is the first byte, $X_1$ is the $2^{nd}$ byte and represents the MSD (Most Significant Digit) of the data, etc, $X_6$ is the $7^{th}$ byte and represents the LSD (Least Significant Digit) of the data, and <CR> is the $8^{th}$ and final byte. |

---

| | |
|---|---|
| S-------------> | The register requests the scale to Zero itself. This is like pressing the Zero key on the scale. Doing this will change the current Zero Point of the scale. |
| <------------------<ACK> | The scale transmits the <ACK> if the Re-Zero was accepted by the scale; otherwise, it will transmit a <NAK>. |

---

| | |
|---|---|
| <EOT>-------------> | The register sends the command to quit Continuous data transfer. |
| <------------------<ACK> | The scale transmits the <ACK>. |

`W------------->` The register requests Continuous Weight data from the scale.

`<---- <STX>` $P_1$ $W_1W_2W_3W_4W_5W_6$ $Z_1Z_2$ $S_1$`<CR>`

**The scale transmits the data in 12 bytes once and then will continue to transmit data whenever the Weight display changes. It will do this until the scale receives a <EOT> which is the command to quit continuous data transfer.**

<STX> is the 1st byte
$P_1$ is the 2nd byte, is the sign of the Weight value, & can be 2B or 2D
$W_1$ to $W_6$ are the 3rd to 8th bytes & is the displayed Weight with decimal point
$Z_1$ to $Z_2$ are the 9th and 10th bytes and are either "LB" or "KG"
$S_1$ is the 11th byte, is the Status byte of the scale (see below)
<CR> is the 12th byte

| Definition | Hex | ASCII Text | Binary |
|---|---|---|---|
| **Motion** | $Y_1$ = 61 | a | 01100001 |
| **Scale at ZERO** | $Y_1$ = 70 | p | 01110000 |
| **Weight < 0** | $Y_1$ = 64 | d | 01100100 |
| **Weight > Capacity** | $Y_1$ = 62 | b | 01100010 |
| **Weight < 0 & Motion** | $Y_1$ = 65 | e | 01100101 |
| **Weight > Capacity & Motion** | $Y_1$ = 63 | c | 01100011 |

| Status Byte Formation | | |
|---|---|---|
| **Bit** | **Description** | |
| Bit 7 (MSB) | Parity Bit. | |
| Bit 6 | (Not used.) | **Always = 1**. |
| Bit 5 | Net Weight Bit. | **Gross = 0**, Net = 1. |
| Bit 4 | Zero Bit. | Zero = 1, Not Zero = 0. |
| Bit 3 | Outside Zero Range Bit. | Outside = 1, **Within = 0**. |
| Bit 2 | Negative Weight Bit. | Negative = 1, Non-Negative = 0. |
| Bit 1 | Overload Bit. | Overload = 1, Non-Overload = 0. |
| Bit 0 (LSB) | Motion Bit. | Motion = 1, Stable = 0. |

**Type 6: Printer:** Whenever F12 = Printer, the scale's RS-232 protocol will be determined by the F13 & F18 Settings.

<STX> = 02  P = 50  <CR> = 0D  R = 52  L = 4C  B = 42  K = 4B  G = 47  . = 2E  <ETX> = 03  0 = 30  1 = 31
S = 53

Register  Scale  Action

**PLU Programming:** The PC sends 63 Bytes of programming data.

<STX> P $N_1N_2$ 0 $R_1R_2R_3R_4R_5R_6$ $P_1P_2P_3P_4P_5$ $U_1U_2U_3U_4U_5U_6U_7$ $C_1C_2C_3C_4C_5C_6$ $F_1F_2$ $S_1S_2S_3$ $T_1$~$T_{28}$ <ETX> ------>
**OR**
<STX> P $N_1N_2$ 1 $Q_1Q_2Q_3Q_4Q_5Q_6$ $W_1W_2W_3W_4W_5$ $U_1U_2U_3U_4U_5U_6U_7$ $C_1C_2C_3C_4C_5C_6$ $F_1F_2$ $S_1S_2S_3$ $T_1$~$T_{28}$ <ETX> ------>

P is the character "P", ASCII hex = 50
$N_1N_2$ is the PLU number in ASCII text, from 1 to 34
0 is the number 0, ASCII hex = 30, and represents a By-Weight PLU
1 is the number 1, ASCII hex = 31, and represents a By-Count PLU
$R_1$~$R_6$ is the Tare in ASCII text including the decimal point
$Q_1$~$Q_6$ is the Quantity in ASCII text
$P_1$~$P_6$ is the %Tare in ASCII text including the decimal point
$W_1$~$W_6$ is the Net Weight Statement in ASCII text
$U_1$~$U_7$ is the Unit Price or Price in ASCII text including the decimal point
$C_1$~$C_6$ is the UPC Code in ASCII text
$F_1$~$F_6$ is the UPC Format in ASCII text, 00 to 15
$S_1$~$S_3$ is the Sell By Date in ASCII text, 000 to 255
$T_1$~$T_{28}$ is the PLU Text in ASCII text, always 28 bytes
**Note:** All numeric values must always use leading zeros and all text should use trailing <NUL> characters.

**The Scale Responds with:**
<---- <ACK>  The scale transmits <ACK> if it received the data and stored it
**OR**
<---- <NAK>  The scale transmits <NAK> if there was a problem with the data.

---

**Store Name Programming:** The PC sends 60 Bytes of programming data.

<STX> S $N_1$ $T_1$~$T_{56}$ <ETX> ------>

S is the character "S", ASCII hex = 53
$N_1$ is the Store Name Line number in ASCII text, from 1 to 3
$T_1$~$T_{56}$ is the Store Name Text in ASCII text, always 56 bytes
**Note:** All numeric values must always use leading zeros and all text should use trailing <NUL> characters.

**The Scale Responds with:**
<---- <ACK>  The scale transmits <ACK> if it received the data and stored it
**OR**
<---- <NAK>  The scale transmits <NAK> if there was a problem with the data.

---

**Date Programming:** The PC sends 10 Bytes of programming data.

```
<STX> D N1N2N3N4N5N6N7N8 <ETX> ------>
```

D is the character “D”, ASCII hex = 44

$N_1$~$N_8$ is the Date in ASCII text format MM DD YYYY,

**Note:** All numeric values must always use leading zeros and all text should use trailing <NUL> characters.

**The Scale Responds with:**

```
<---- <ACK>
```

The scale transmits <ACK> if it received the data and stored it

**OR**

```
<---- <NAK>
```

The scale transmits <NAK> if there was a problem with the data.

---