RSLogix5000 String Handling on the DeviceMaster UP

1 Introduction

The DeviceMaster UP has been designed to provide connectivity for a vast array of devices. Installations involving devices such as weigh scales, barcode scanners, RFID readers, visions systems, torque wrenches, motor drives, robotics, and encoders are common. Each and every installation can have unique requirements and it is often necessary to transfer large data packet sizes between the PLC and end device. To meet this requirement, the DeviceMaster UP interface has been designed to transfer the maximum data payload provided by RSLogix5000 based PLCs.

One of the more common PLC data formats is the STRING format. While the STRING format is ideal for some installations, the maximum length of 82 ASCII characters does not provide the necessary length or data type for many installations. For this reason, the STRING format is not directly supported by the DeviceMaster UP.

The DeviceMaster UP interface has been designed so that a PLC program can be easily modified to transfer and receive string data. The STRING format and the DeviceMaster UP data transfer format are quite similar and it requires just a few PLC instructions to convert the formats.

2 Data Formats

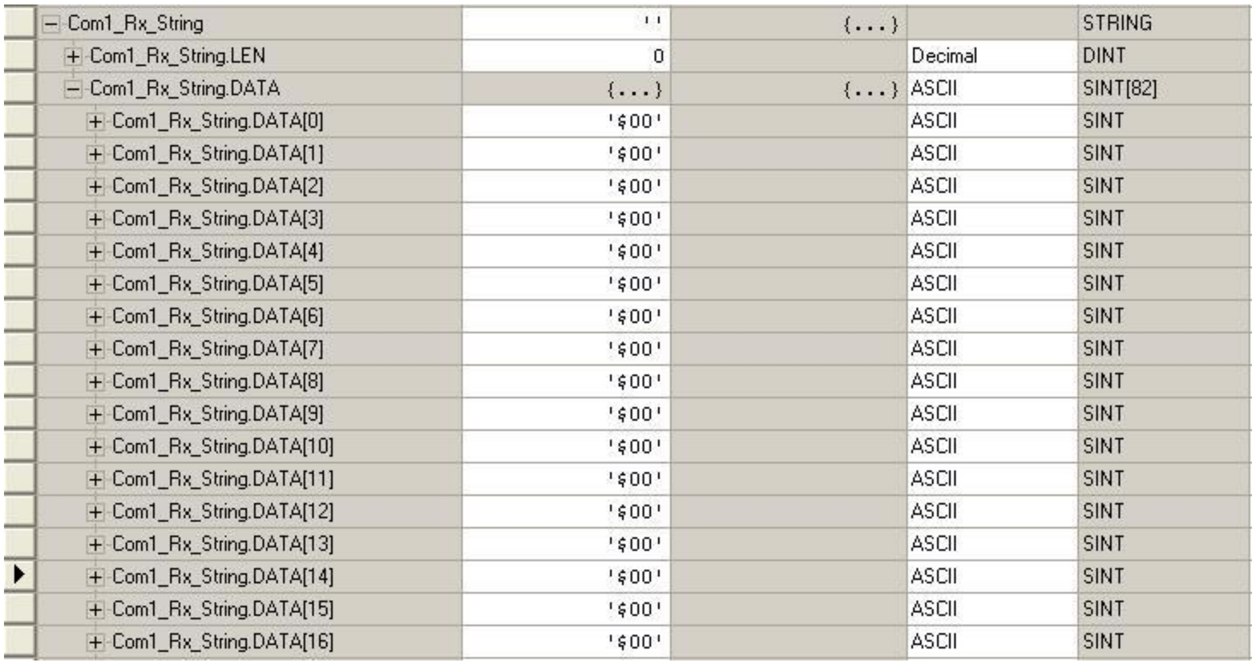
The following formats apply to the PLC program and DeviceMaster UP.

Please note: The screen displayed are for a modified serial port loopback example using the Write-to-Tag receive data format.

2.1 String Format

The string format contains:

- A length field that is 32 bit DINT, (4 bytes), in length.
- Up to 82 characters of data.

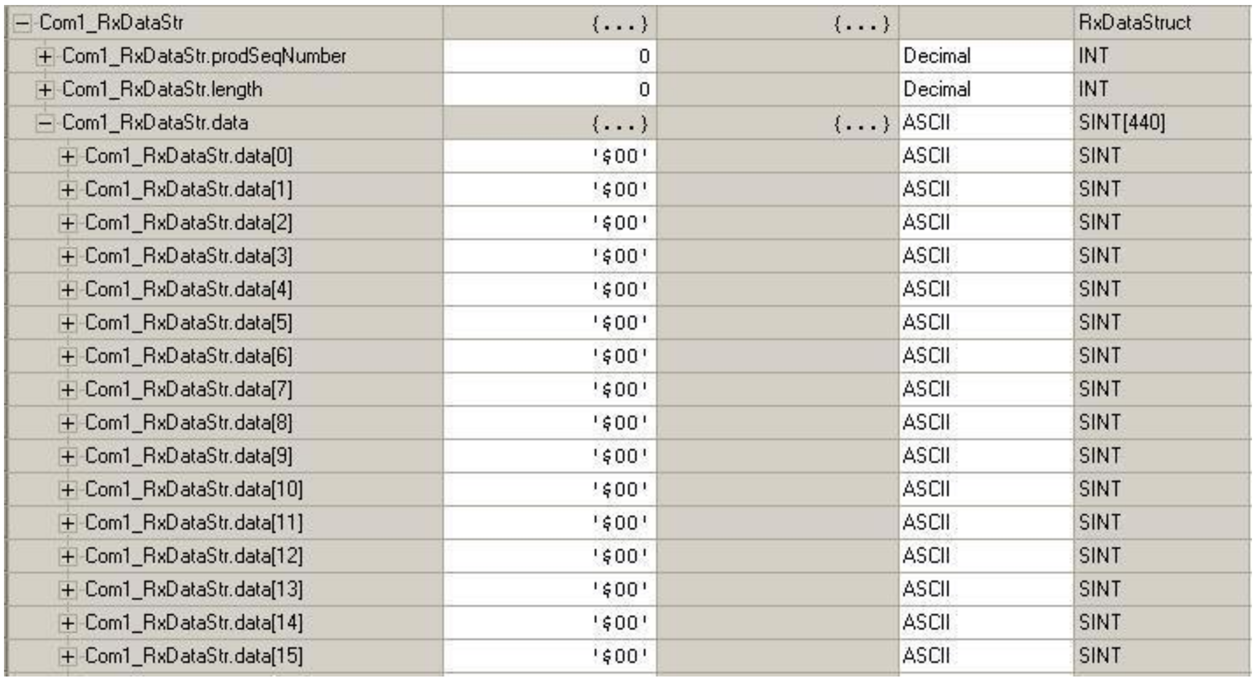


Com1_Rx_String	''	{...}		STRING
Com1_Rx_String.LEN	0		Decimal	DINT
Com1_Rx_String.DATA	{...}	{...}	ASCII	SINT[82]
Com1_Rx_String.DATA[0]	'$00'		ASCII	SINT
Com1_Rx_String.DATA[1]	'$00'		ASCII	SINT
Com1_Rx_String.DATA[2]	'$00'		ASCII	SINT
Com1_Rx_String.DATA[3]	'$00'		ASCII	SINT
Com1_Rx_String.DATA[4]	'$00'		ASCII	SINT
Com1_Rx_String.DATA[5]	'$00'		ASCII	SINT
Com1_Rx_String.DATA[6]	'$00'		ASCII	SINT
Com1_Rx_String.DATA[7]	'$00'		ASCII	SINT
Com1_Rx_String.DATA[8]	'$00'		ASCII	SINT
Com1_Rx_String.DATA[9]	'$00'		ASCII	SINT
Com1_Rx_String.DATA[10]	'$00'		ASCII	SINT
Com1_Rx_String.DATA[11]	'$00'		ASCII	SINT
Com1_Rx_String.DATA[12]	'$00'		ASCII	SINT
Com1_Rx_String.DATA[13]	'$00'		ASCII	SINT
Com1_Rx_String.DATA[14]	'$00'		ASCII	SINT
Com1_Rx_String.DATA[15]	'$00'		ASCII	SINT
Com1_Rx_String.DATA[16]	'$00'		ASCII	SINT

2.2 DeviceMaster UP Data Transfer Format

The DeviceMaster UP data transfer format contains:

- A 16 bit INT, (2 byte), sequence number.
    - The sequence number is incremented every time a new packet is received from the DeviceMaster UP.
    - Incrementing the sequence number for transmit messages is optional.
- A 16 bit INT, (2 bytes), defines the data length
- Up to 440 SINTs, (bytes), of data. This may contain any sort of data including ASCII characters. *Larger receive packets up to 1518 bytes for serial and 2048 bytes for Ethernet devices can be received by using a sequence of 444 SINT tags.*



⊟ Com1_RxDataStr	{...}	{...}		RxDataStruct
⊞ Com1_RxDataStr.prodSeqNumber	0		Decimal	INT
⊞ Com1_RxDataStr.length	0		Decimal	INT
⊟ Com1_RxDataStr.data	{...}	{...}	ASCII	SINT[440]
⊞ Com1_RxDataStr.data[0]	'$00'		ASCII	SINT
⊞ Com1_RxDataStr.data[1]	'$00'		ASCII	SINT
⊞ Com1_RxDataStr.data[2]	'$00'		ASCII	SINT
⊞ Com1_RxDataStr.data[3]	'$00'		ASCII	SINT
⊞ Com1_RxDataStr.data[4]	'$00'		ASCII	SINT
⊞ Com1_RxDataStr.data[5]	'$00'		ASCII	SINT
⊞ Com1_RxDataStr.data[6]	'$00'		ASCII	SINT
⊞ Com1_RxDataStr.data[7]	'$00'		ASCII	SINT
⊞ Com1_RxDataStr.data[8]	'$00'		ASCII	SINT
⊞ Com1_RxDataStr.data[9]	'$00'		ASCII	SINT
⊞ Com1_RxDataStr.data[10]	'$00'		ASCII	SINT
⊞ Com1_RxDataStr.data[11]	'$00'		ASCII	SINT
⊞ Com1_RxDataStr.data[12]	'$00'		ASCII	SINT
⊞ Com1_RxDataStr.data[13]	'$00'		ASCII	SINT
⊞ Com1_RxDataStr.data[14]	'$00'		ASCII	SINT
⊞ Com1_RxDataStr.data[15]	'$00'		ASCII	SINT

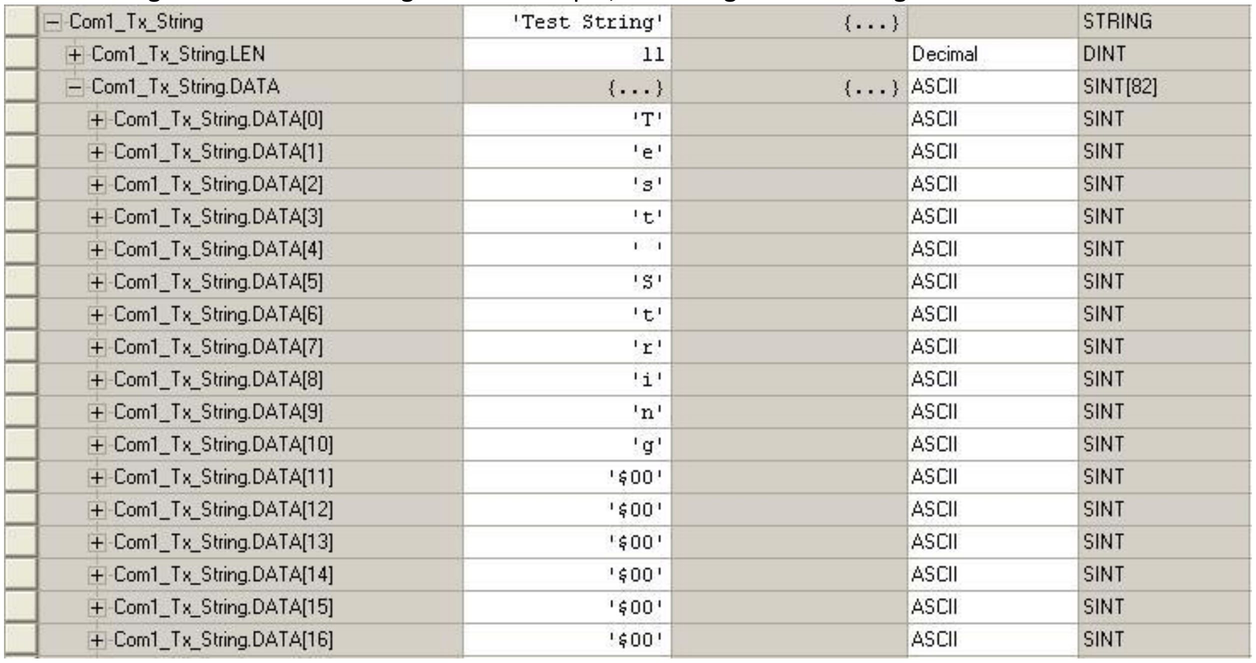
3 PLC Programming Example

The following display example PLC program enhancements that transfer STRING format tag data to and from a DeviceMaster UP.

3.1 Required Tags

3.1.1 Com1_Tx_String

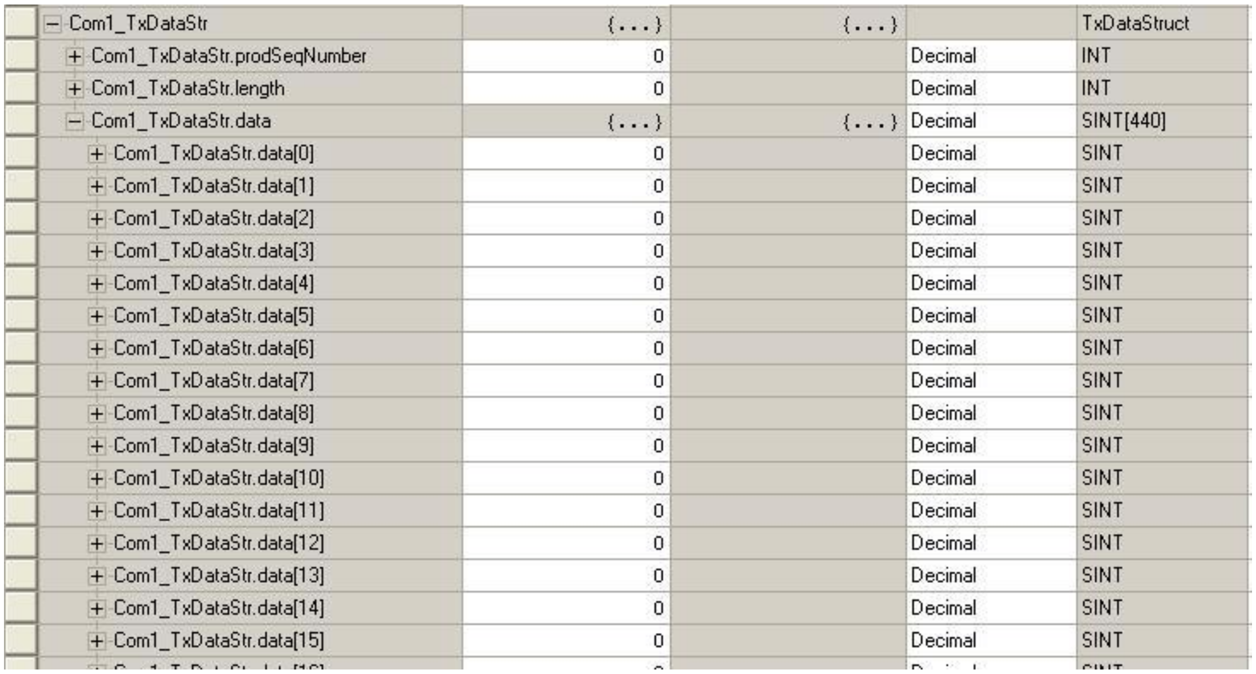
The following is the transmit string. In this example, the string is "Test String".



− Com1_Tx_String	'Test String'	{...}		STRING
+ Com1_Tx_String.LEN	11		Decimal	DINT
− Com1_Tx_String.DATA	{...}	{...}	ASCII	SINT[82]
+ Com1_Tx_String.DATA[0]	'T'		ASCII	SINT
+ Com1_Tx_String.DATA[1]	'e'		ASCII	SINT
+ Com1_Tx_String.DATA[2]	's'		ASCII	SINT
+ Com1_Tx_String.DATA[3]	't'		ASCII	SINT
+ Com1_Tx_String.DATA[4]	' '		ASCII	SINT
+ Com1_Tx_String.DATA[5]	'S'		ASCII	SINT
+ Com1_Tx_String.DATA[6]	't'		ASCII	SINT
+ Com1_Tx_String.DATA[7]	'r'		ASCII	SINT
+ Com1_Tx_String.DATA[8]	'i'		ASCII	SINT
+ Com1_Tx_String.DATA[9]	'n'		ASCII	SINT
+ Com1_Tx_String.DATA[10]	'g'		ASCII	SINT
+ Com1_Tx_String.DATA[11]	'$00'		ASCII	SINT
+ Com1_Tx_String.DATA[12]	'$00'		ASCII	SINT
+ Com1_Tx_String.DATA[13]	'$00'		ASCII	SINT
+ Com1_Tx_String.DATA[14]	'$00'		ASCII	SINT
+ Com1_Tx_String.DATA[15]	'$00'		ASCII	SINT
+ Com1_Tx_String.DATA[16]	'$00'		ASCII	SINT

3.1.2 Com1_TxDataStr

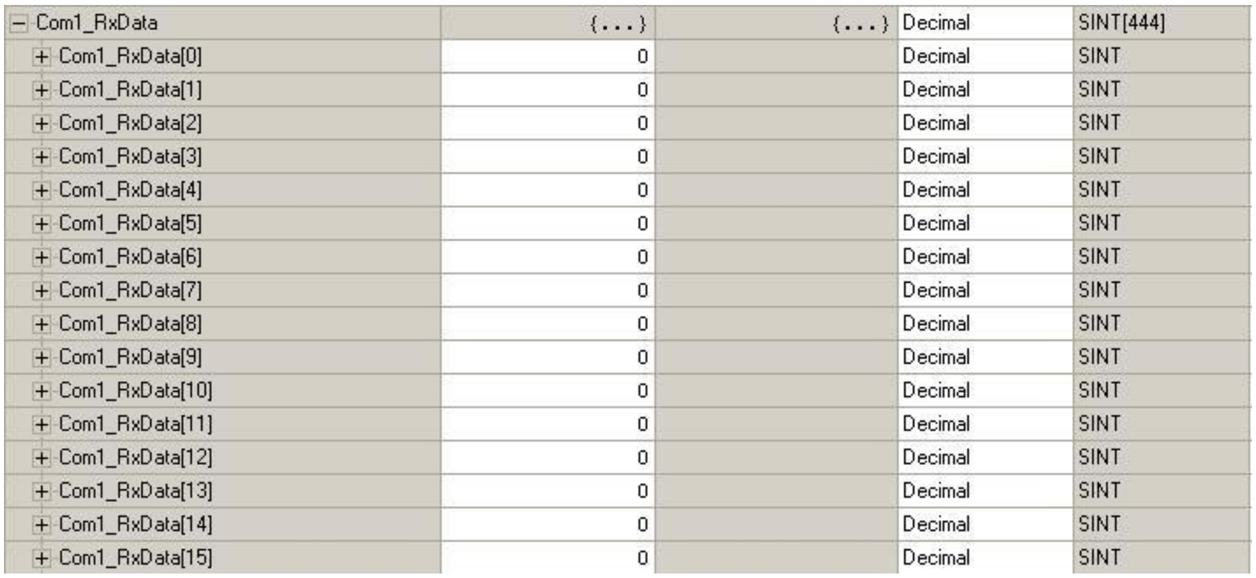
This is the data format required by the DeviceMaster UP:



Com1_TxDataStr	{...}	{...}		TxDataStruct
Com1_TxDataStr.prodSeqNumber	0		Decimal	INT
Com1_TxDataStr.length	0		Decimal	INT
Com1_TxDataStr.data	{...}	{...}	Decimal	SINT[440]
Com1_TxDataStr.data[0]	0		Decimal	SINT
Com1_TxDataStr.data[1]	0		Decimal	SINT
Com1_TxDataStr.data[2]	0		Decimal	SINT
Com1_TxDataStr.data[3]	0		Decimal	SINT
Com1_TxDataStr.data[4]	0		Decimal	SINT
Com1_TxDataStr.data[5]	0		Decimal	SINT
Com1_TxDataStr.data[6]	0		Decimal	SINT
Com1_TxDataStr.data[7]	0		Decimal	SINT
Com1_TxDataStr.data[8]	0		Decimal	SINT
Com1_TxDataStr.data[9]	0		Decimal	SINT
Com1_TxDataStr.data[10]	0		Decimal	SINT
Com1_TxDataStr.data[11]	0		Decimal	SINT
Com1_TxDataStr.data[12]	0		Decimal	SINT
Com1_TxDataStr.data[13]	0		Decimal	SINT
Com1_TxDataStr.data[14]	0		Decimal	SINT
Com1_TxDataStr.data[15]	0		Decimal	SINT

3.1.3 Com1_RxData

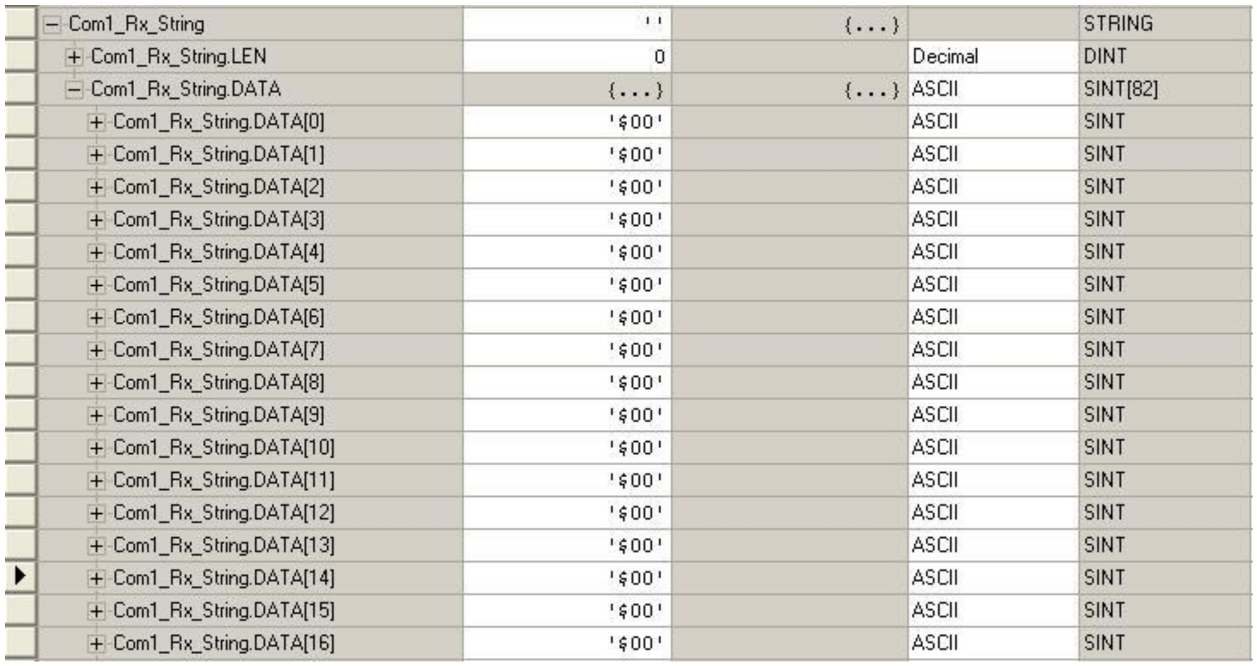
The tag used to receive data from the DeviceMaster UP must be an array of SINTs large enough to the hold both the data and the 4 byte sequence/length header.



Com1_RxData	{...}	{...}	Decimal	SINT[444]
Com1_RxData[0]	0		Decimal	SINT
Com1_RxData[1]	0		Decimal	SINT
Com1_RxData[2]	0		Decimal	SINT
Com1_RxData[3]	0		Decimal	SINT
Com1_RxData[4]	0		Decimal	SINT
Com1_RxData[5]	0		Decimal	SINT
Com1_RxData[6]	0		Decimal	SINT
Com1_RxData[7]	0		Decimal	SINT
Com1_RxData[8]	0		Decimal	SINT
Com1_RxData[9]	0		Decimal	SINT
Com1_RxData[10]	0		Decimal	SINT
Com1_RxData[11]	0		Decimal	SINT
Com1_RxData[12]	0		Decimal	SINT
Com1_RxData[13]	0		Decimal	SINT
Com1_RxData[14]	0		Decimal	SINT
Com1_RxData[15]	0		Decimal	SINT

3.1.4 Com1_RxString

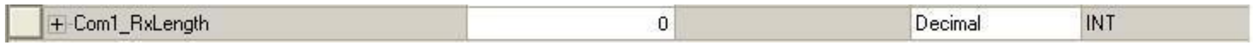
The following is the string that the received data will be transferred into:



− Com1_Rx_String	''	{...}		STRING
+ Com1_Rx_String.LEN	0		Decimal	DINT
− Com1_Rx_String.DATA	{...}	{...}	ASCII	SINT[82]
+ Com1_Rx_String.DATA[0]	'$00'		ASCII	SINT
+ Com1_Rx_String.DATA[1]	'$00'		ASCII	SINT
+ Com1_Rx_String.DATA[2]	'$00'		ASCII	SINT
+ Com1_Rx_String.DATA[3]	'$00'		ASCII	SINT
+ Com1_Rx_String.DATA[4]	'$00'		ASCII	SINT
+ Com1_Rx_String.DATA[5]	'$00'		ASCII	SINT
+ Com1_Rx_String.DATA[6]	'$00'		ASCII	SINT
+ Com1_Rx_String.DATA[7]	'$00'		ASCII	SINT
+ Com1_Rx_String.DATA[8]	'$00'		ASCII	SINT
+ Com1_Rx_String.DATA[9]	'$00'		ASCII	SINT
+ Com1_Rx_String.DATA[10]	'$00'		ASCII	SINT
+ Com1_Rx_String.DATA[11]	'$00'		ASCII	SINT
+ Com1_Rx_String.DATA[12]	'$00'		ASCII	SINT
+ Com1_Rx_String.DATA[13]	'$00'		ASCII	SINT
+ Com1_Rx_String.DATA[14]	'$00'		ASCII	SINT
+ Com1_Rx_String.DATA[15]	'$00'		ASCII	SINT
+ Com1_Rx_String.DATA[16]	'$00'		ASCII	SINT

3.1.5 Com1_RxLength

An intermediate 16 bit INT is required to transfer the received length field from the SINT array into the string length field:

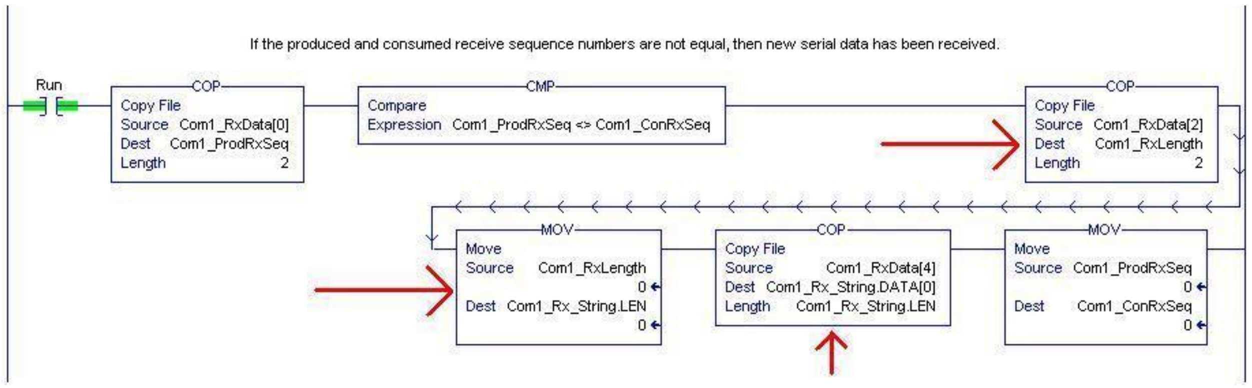


+ Com1_RxLength	0		Decimal	INT

3.2 PLC Ladder Instructions

3.2.1 Receive Data Rung

The following is an example receive rung that copies the received data into a receive string tag.

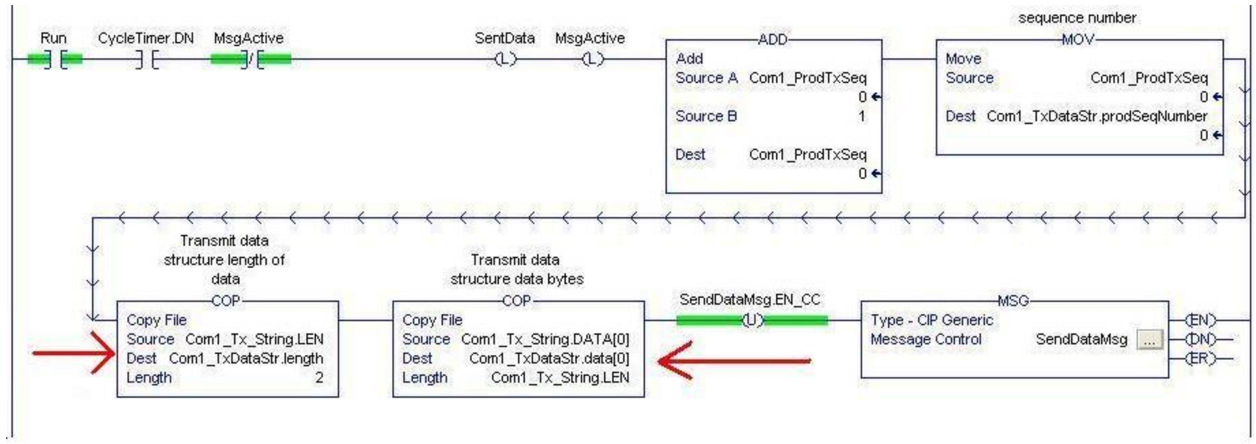

If the produced and consumed receive sequence numbers are not equal, then new serial data has been received.

```
Run   COP                      CMP                                           COP
      Copy File                Compare                                       Copy File
      Source  Com1_RxData[0]   Expression  Com1_ProdRxSeq <> Com1_ConRxSeq   Source  Com1_RxData[2]
      Dest    Com1_ProdRxSeq                                                 Dest    Com1_RxLength
      Length               2                                                 Length               2

      MOV                           COP                               MOV
      Move                          Copy File                         Move
      Source      Com1_RxLength     Source        Com1_RxData[4]      Source  Com1_ProdRxSeq
                              0     Dest  Com1_Rx_String.DATA[0]                           0
      Dest  Com1_Rx_String.LEN      Length    Com1_Rx_String.LEN      Dest      Com1_ConRxSeq
                              0                                                            0
```

Where:

- The first two bytes of the received data, (indexes 0 and 1 of Com1_RxData), indicate the sequence number. New received data is indicated by an incremented sequence number.
- The next two bytes, (indexes 2 and 3 of Com1_RxData), indicate the length of the data.
  - These two bytes are first copied in the 16 bit Com1_RxLength tag.
  - The Com1_RxLength is then moved into the 32 bit Com1_Rx_String.LEN field.
- The received data starts at index 4 (Com1_RxData[4]). The string length, (Com1_Rx_String.LEN), can be used to directly copy the received data into the string data field.

3.2.2 Transmit Data Rung

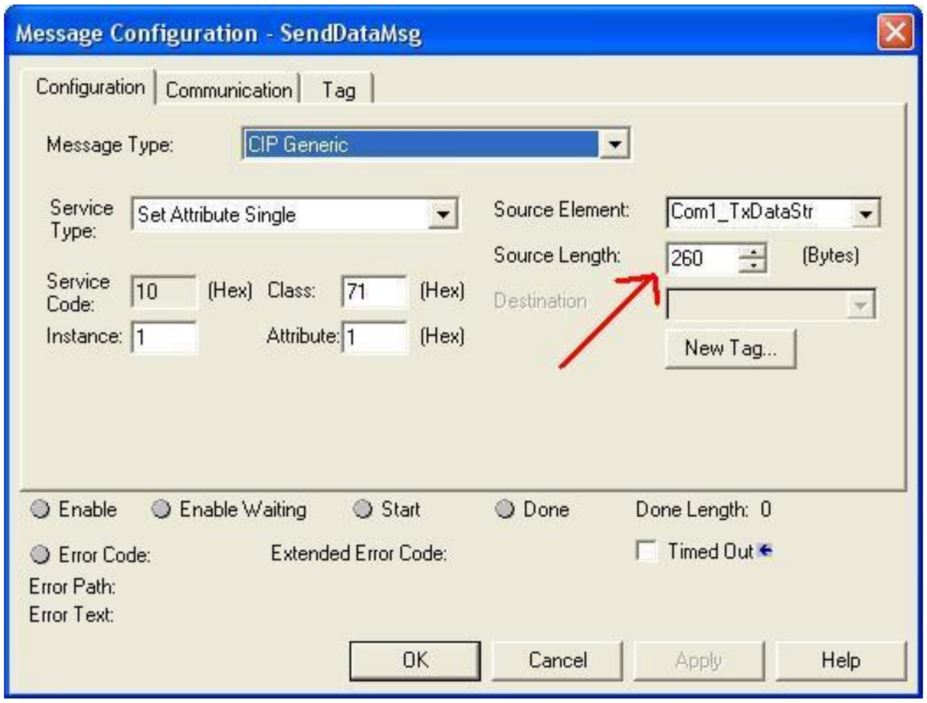


```
ADD
Add
Source A  Com1_ProdTxSeq
                       0
Source B               1
Dest      Com1_ProdTxSeq
                       0

MOV
Move
Source            Com1_ProdTxSeq
                               0
Dest  Com1_TxDataStr.prodSeqNumber
                               0

COP
Copy File
Source  Com1_Tx_String.LEN
Dest  Com1_TxDataStr.length
Length                   2

COP
Copy File
Source  Com1_Tx_String.DATA[0]
Dest     Com1_TxDataStr.data[0]
Length       Com1_Tx_String.LEN

MSG
Type - CIP Generic
Message Control       SendDataMsg
```

Where:

- The Com1_TxDataStr tag is used to transmit the data from the PLC to the DeviceMaster UP.
- Incrementing the 16 bit sequence number, (Com1_TxDataStr.prodSeqNumber), is optional for transmit messages.
- The string length, (Com1_Tx_String.LEN), can be directly copied into the data length field (Com1_TxDataStr.length).
- The string data, (Com1_Tx_String.DATA), can be directly copied into the transmit data field (Com1_TxDataStr.data).

3.2.3 Transmit Data Message Length



Message Type: CIP Generic
Service Type: Set Attribute Single
Service Code: 10 (Hex) Class: 71 (Hex)
Instance: 1 Attribute: 1 (Hex)
Source Element: Com1_TxDataStr
Source Length: 260 (Bytes)
Destination
Done Length: 0
Error Code: Extended Error Code:
Error Path:
Error Text:

The **Source Length** of the MSG instruction must be long enough to transfer the full transmit data message. This includes the 4 byte sequence number/length header and the length of data.

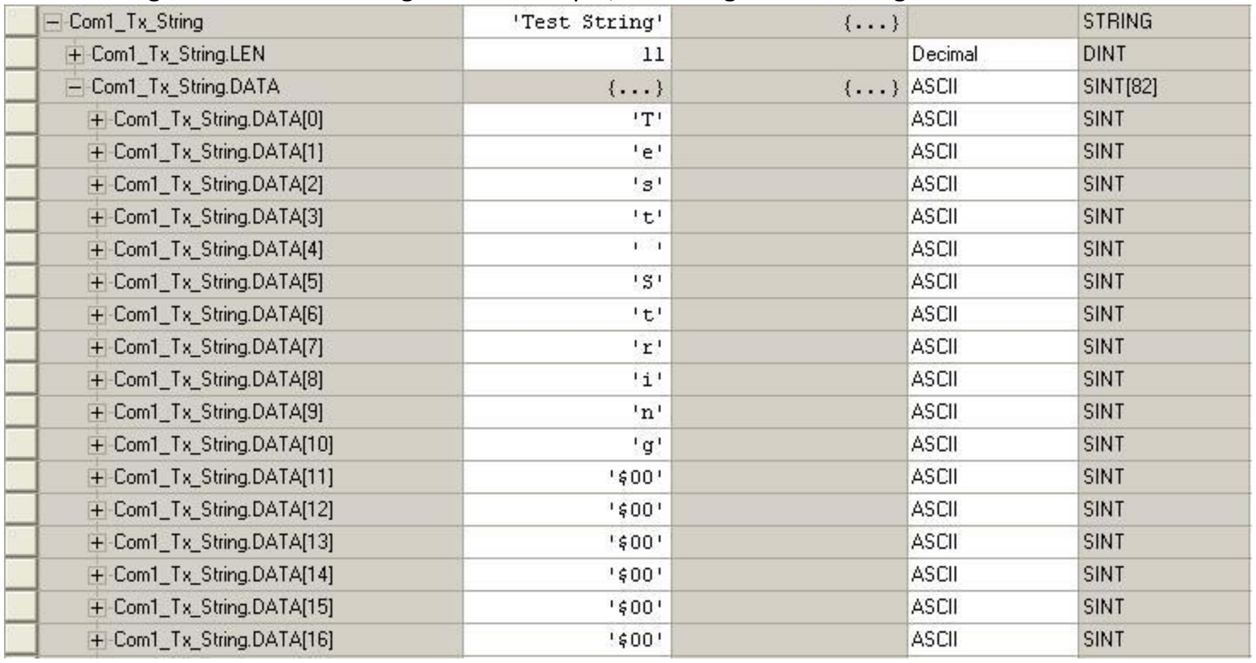
As an example, if the data length was 32 bytes, the **Source Length** would need to be at least 36 bytes to transfer the data. Any extra bytes are ignored by the DeviceMaster UP..

3.3 Tags When the Example Program is Running

The following indicate the tags when the example program is running on a DeviceMaster UP with a loopback plug attached to port 1:

3.3.1 Com1_Tx_String

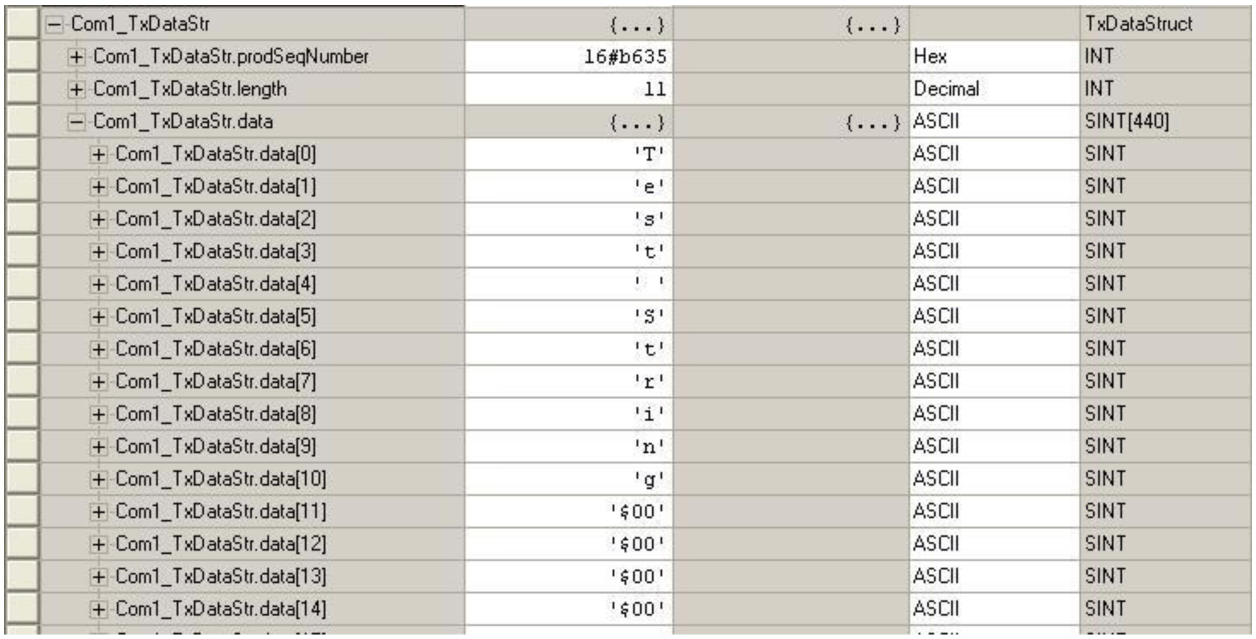
The following is the transmit string. In this example, the string is "Test String".



Com1_Tx_String	'Test String'	{...}		STRING
Com1_Tx_String.LEN	11		Decimal	DINT
Com1_Tx_String.DATA	{...}	{...}	ASCII	SINT[82]
Com1_Tx_String.DATA[0]	'T'		ASCII	SINT
Com1_Tx_String.DATA[1]	'e'		ASCII	SINT
Com1_Tx_String.DATA[2]	's'		ASCII	SINT
Com1_Tx_String.DATA[3]	't'		ASCII	SINT
Com1_Tx_String.DATA[4]	' '		ASCII	SINT
Com1_Tx_String.DATA[5]	'S'		ASCII	SINT
Com1_Tx_String.DATA[6]	't'		ASCII	SINT
Com1_Tx_String.DATA[7]	'r'		ASCII	SINT
Com1_Tx_String.DATA[8]	'i'		ASCII	SINT
Com1_Tx_String.DATA[9]	'n'		ASCII	SINT
Com1_Tx_String.DATA[10]	'g'		ASCII	SINT
Com1_Tx_String.DATA[11]	'$00'		ASCII	SINT
Com1_Tx_String.DATA[12]	'$00'		ASCII	SINT
Com1_Tx_String.DATA[13]	'$00'		ASCII	SINT
Com1_Tx_String.DATA[14]	'$00'		ASCII	SINT
Com1_Tx_String.DATA[15]	'$00'		ASCII	SINT
Com1_Tx_String.DATA[16]	'$00'		ASCII	SINT

3.3.2 Com1_TxDataStr

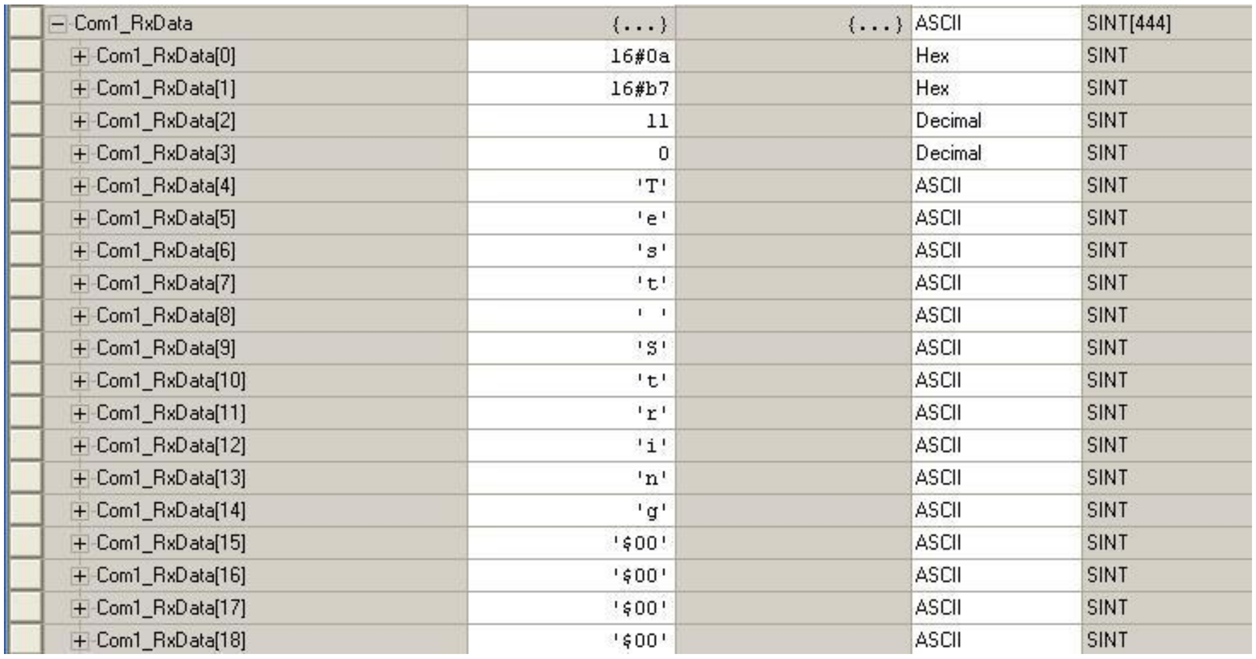
The Com1_TxDataStr is filled in from Com1_Tx_String by the transmit rung of the PLC program. This is the data format required by the DeviceMaster UP.



− Com1_TxDataStr	{...}	{...}		TxDataStruct
+ Com1_TxDataStr.prodSeqNumber	16#b635		Hex	INT
+ Com1_TxDataStr.length	11		Decimal	INT
− Com1_TxDataStr.data	{...}	{...}	ASCII	SINT[440]
+ Com1_TxDataStr.data[0]	'T'		ASCII	SINT
+ Com1_TxDataStr.data[1]	'e'		ASCII	SINT
+ Com1_TxDataStr.data[2]	's'		ASCII	SINT
+ Com1_TxDataStr.data[3]	't'		ASCII	SINT
+ Com1_TxDataStr.data[4]	' '		ASCII	SINT
+ Com1_TxDataStr.data[5]	'S'		ASCII	SINT
+ Com1_TxDataStr.data[6]	't'		ASCII	SINT
+ Com1_TxDataStr.data[7]	'r'		ASCII	SINT
+ Com1_TxDataStr.data[8]	'i'		ASCII	SINT
+ Com1_TxDataStr.data[9]	'n'		ASCII	SINT
+ Com1_TxDataStr.data[10]	'g'		ASCII	SINT
+ Com1_TxDataStr.data[11]	'$00'		ASCII	SINT
+ Com1_TxDataStr.data[12]	'$00'		ASCII	SINT
+ Com1_TxDataStr.data[13]	'$00'		ASCII	SINT
+ Com1_TxDataStr.data[14]	'$00'		ASCII	SINT

3.3.3 Com1_RxData

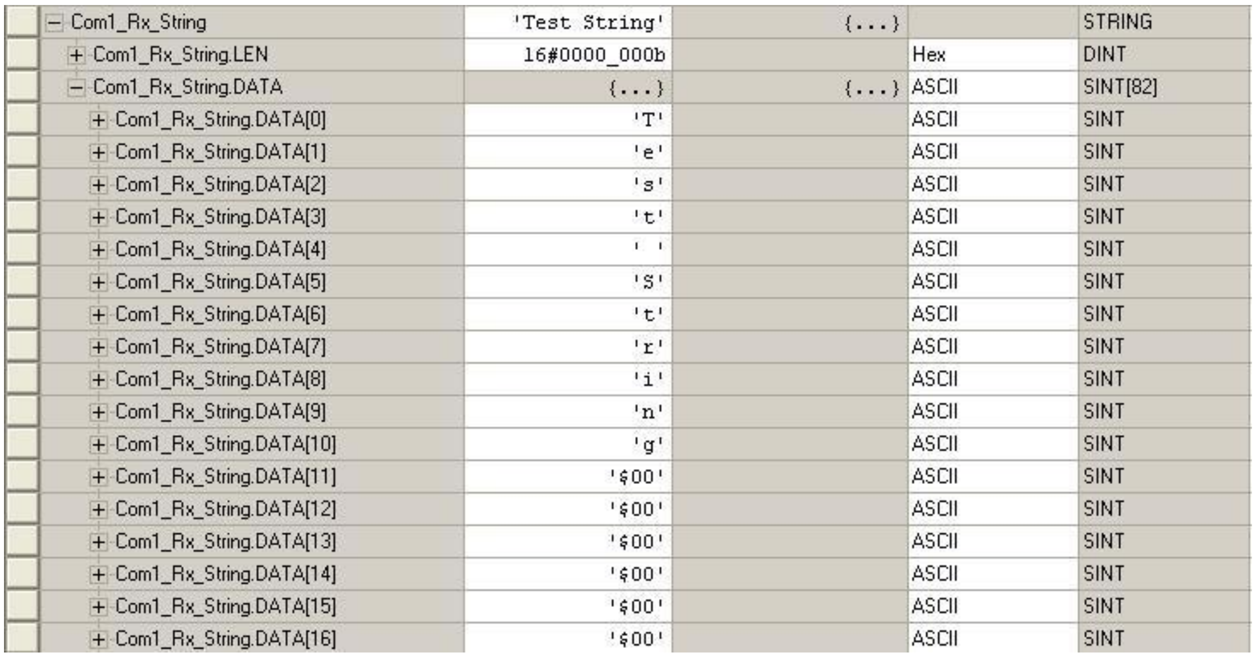
In the loopback program, the data received is the same as the data transmitted.



− Com1_RxData	{...}	{...}	ASCII	SINT[444]
+ Com1_RxData[0]	16#0a		Hex	SINT
+ Com1_RxData[1]	16#b7		Hex	SINT
+ Com1_RxData[2]	11		Decimal	SINT
+ Com1_RxData[3]	0		Decimal	SINT
+ Com1_RxData[4]	'T'		ASCII	SINT
+ Com1_RxData[5]	'e'		ASCII	SINT
+ Com1_RxData[6]	's'		ASCII	SINT
+ Com1_RxData[7]	't'		ASCII	SINT
+ Com1_RxData[8]	' '		ASCII	SINT
+ Com1_RxData[9]	'S'		ASCII	SINT
+ Com1_RxData[10]	't'		ASCII	SINT
+ Com1_RxData[11]	'r'		ASCII	SINT
+ Com1_RxData[12]	'i'		ASCII	SINT
+ Com1_RxData[13]	'n'		ASCII	SINT
+ Com1_RxData[14]	'g'		ASCII	SINT
+ Com1_RxData[15]	'$00'		ASCII	SINT
+ Com1_RxData[16]	'$00'		ASCII	SINT
+ Com1_RxData[17]	'$00'		ASCII	SINT
+ Com1_RxData[18]	'$00'		ASCII	SINT

3.3.4 Com1_RxString

The Com1_RxData length and data are transferred into Com1_RxString by the receive rung of the PLC program.



Com1_Rx_String	'Test String'	{...}		STRING
Com1_Rx_String.LEN	16#0000_000b		Hex	DINT
Com1_Rx_String.DATA	{...}	{...}	ASCII	SINT[82]
Com1_Rx_String.DATA[0]	'T'		ASCII	SINT
Com1_Rx_String.DATA[1]	'e'		ASCII	SINT
Com1_Rx_String.DATA[2]	's'		ASCII	SINT
Com1_Rx_String.DATA[3]	't'		ASCII	SINT
Com1_Rx_String.DATA[4]	' '		ASCII	SINT
Com1_Rx_String.DATA[5]	'S'		ASCII	SINT
Com1_Rx_String.DATA[6]	't'		ASCII	SINT
Com1_Rx_String.DATA[7]	'r'		ASCII	SINT
Com1_Rx_String.DATA[8]	'i'		ASCII	SINT
Com1_Rx_String.DATA[9]	'n'		ASCII	SINT
Com1_Rx_String.DATA[10]	'g'		ASCII	SINT
Com1_Rx_String.DATA[11]	'$00'		ASCII	SINT
Com1_Rx_String.DATA[12]	'$00'		ASCII	SINT
Com1_Rx_String.DATA[13]	'$00'		ASCII	SINT
Com1_Rx_String.DATA[14]	'$00'		ASCII	SINT
Com1_Rx_String.DATA[15]	'$00'		ASCII	SINT
Com1_Rx_String.DATA[16]	'$00'		ASCII	SINT