Polling ASCII devices as Modbus slaves

February 2018

90000648

Contents

1 Introduction

1.1 Abstract

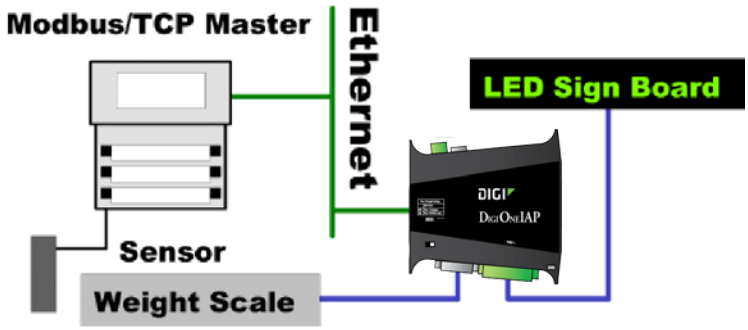
This application note describes how to use a Modbus, and Modicon MSTR blocks, to remotely read or write ASCII data using the Digi One IAP and the Release E firmware. The Digi One IAP makes the ASCII device (such as a barcode scanner, weight scale, or LED marquee sign) appear as a Modbus slave device.

1.2 Sample application

The sample application shown next includes two ASCII devices in two distinct modes and uses both serial ports.

A Momentum PLC uses digital input from a position sensor to sense the presence of a product on a moving conveyor, and uses an ASCII command to poll the weight from a weight scale under the conveyor. So the PLC sends a fixed, ASCII-poll command and receives a variable ASCII response.

For the second device, the Momentum formats and sends ASCII strings to be displayed on a marquee or LED signboard—the Momentum expects no response.



1.3 Overview of ASCII Import function

A Modbus/TCP master such as a Modicon PLC, supports only Modbus protocols on its Ethernet port. Unless you purchase special hardware modules, you cannot open a raw socket to access a remote ASCII device by TCP/IP tunneling.

The Digi One IAP ASCII Import function solves this problem by making the ASCII serial device appear as a Modbus slave. ASCII Import accepts Modbus reads and writes, and responds with the expected Modbus response. In this way, a Modicon PLC can use MSTR blocks to read and write the Digi One IAP. In addition, ASCII data is packed into the data registers that are read or written.

1.3.1 ASCII Import as receive-only

Some ASCII devices—such as the serial output printer output from a vibration monitor or a barcode scanner—only send out messages. The Digi One IAP considers such devices as **receive-only**. Your PLC does a Modbus read of the Digi One IAP, and the response is one of the following:

- All zero values (meaning no waiting data)
- The oldest ASCII message waiting in queue packed as holding registers

Setting the slave timeout to zero (0) causes old messages to be saved indefinitely, while setting slave timeout from 1 to 65,535 ms (about one minute) causes old messages to be discarded after this timeout if they are not read.

1.3.2 ASCII Import as transmit-only

Some ASCII devices, such as a serial printer or an LED marquee display, only receive messages and never send a response. The Digi One IAP considers such devices as **transmit-only**. Your PLC creates an ASCII string packed in holding registers and does a Modbus write to the Digi One IAP. The Digi One IAP unpacks the ASCII data and sends it out the serial port.

1.3.3 ASCII Import as half-duplex (poll/response)

Some ASCII devices—such as a weight scale that can be queried for gross, net, and tare weights—must be polled for a response. The Digi One IAP considers such devices as half-duplex. Your PLC creates an ASCII string that represents the poll and does a Modbus write to the Digi One IAP. The Digi One IAP unpacks the ASCII data and remembers it in one of 10 buffers. The data is not sent out at this time. Instead, when your PLC does a Modbus read of the Digi One IAP, it sends out the remembered poll and returns the ASCII response packed into the Modbus read response. The Digi One IAP maintains ten separate buffer pairs per serial port. These buffer pairs can be used by 10 remote masters to avoid conflicts, or by a single master to remember 10 distinct polls. For example, you could use buffers 1, 2, and 3 to hold the requests to read gross, net, and tare weights respectively, and use buffer 4 to hold the request to reset or zero the tare weight.

2 Setting up the Digi One IAP

The instructions in this application guide are based on the assumption that you understand the basics of setting up and accessing your Digi product by Ethernet and TCP/IP. If you need help with these procedures, see the related guides at Digi's support site.

2.1 Overview

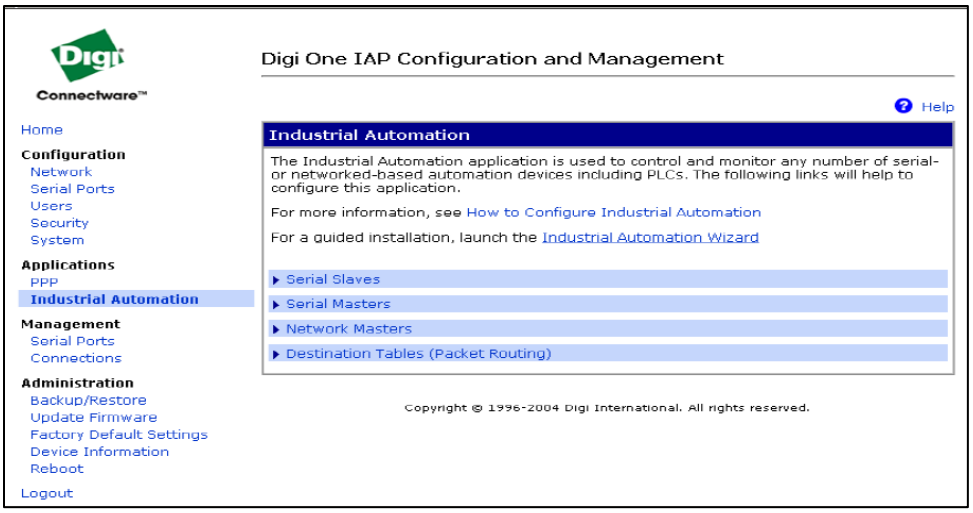
Here is a summary of the steps for configuring the Digi One IAP.

- Configure incoming Modbus/TCP masters (a message source).
- Configure ASCII device on port 1 for half-duplex (weight scale).
- Configure ASCII device on port 2 for transmit-only (LED marquee display).
- Configure Modbus messages to slave 1 to go to port 1.
- Configure Modbus messages to slave 2 to go to port 2.

2.2 By web wizard, release E

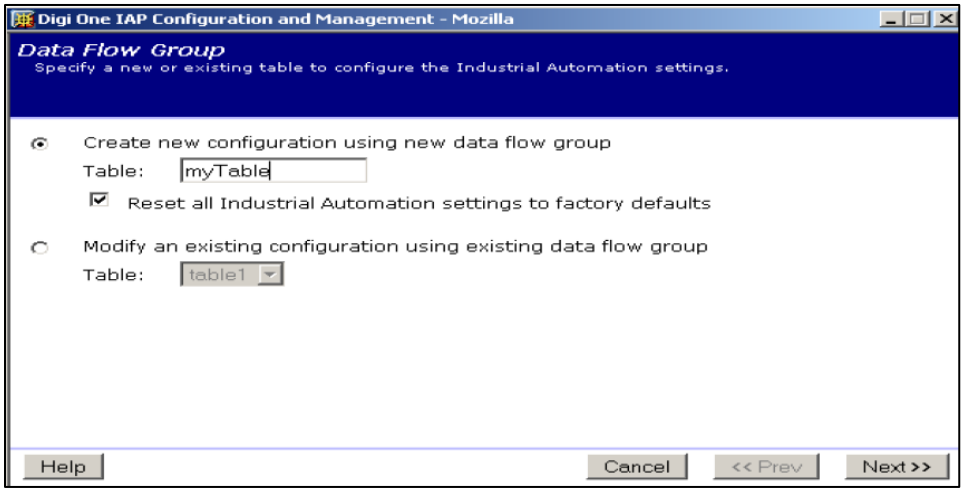
2.2.1 Select the Industrial Automation Wizard

1. In the left column, under the Applications section, click **Industrial Automation**.
2. Click **Launch the Industrial Automation Wizard**. The wizard walks you through the process of defining where messages come from and where they go.



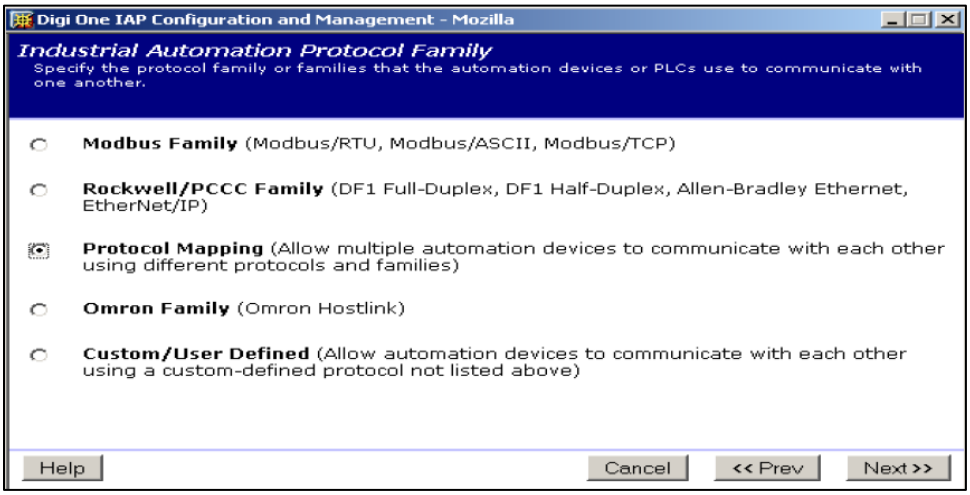
2.2.2 Create a new group and reset all IA settings

1. Click Create new configuration using new data flow group, and check Reset all Industrial Automation settings to factory defaults.
2. Enter a name in the Table field. Use any name you like; for example, table1 or myTable.
3. Click Next.



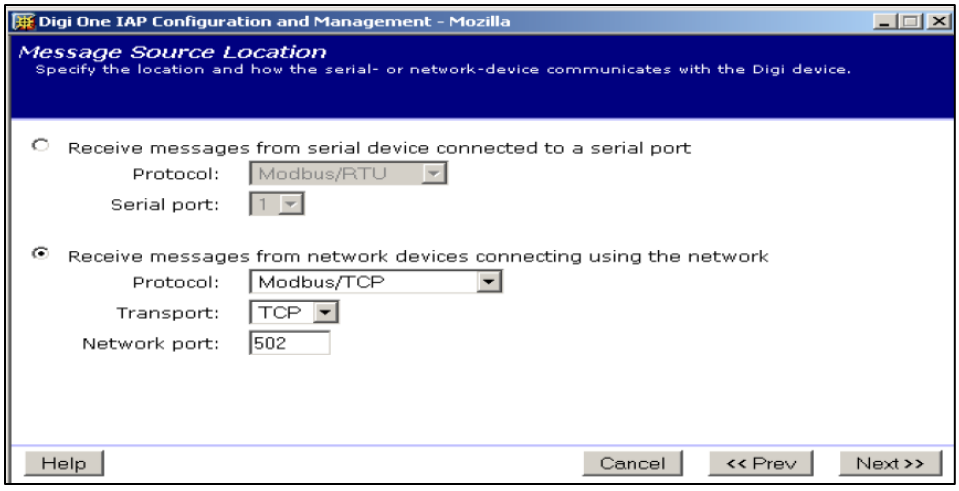
2.2.3 Limit this group to any protocol that supports mapping

1. To map Modbus to ASCII, click **Protocol Mapping**. This setting lets you use most protocols within your configuration, including mixing Modbus and Rockwell protocols.
2. Click **Next**.



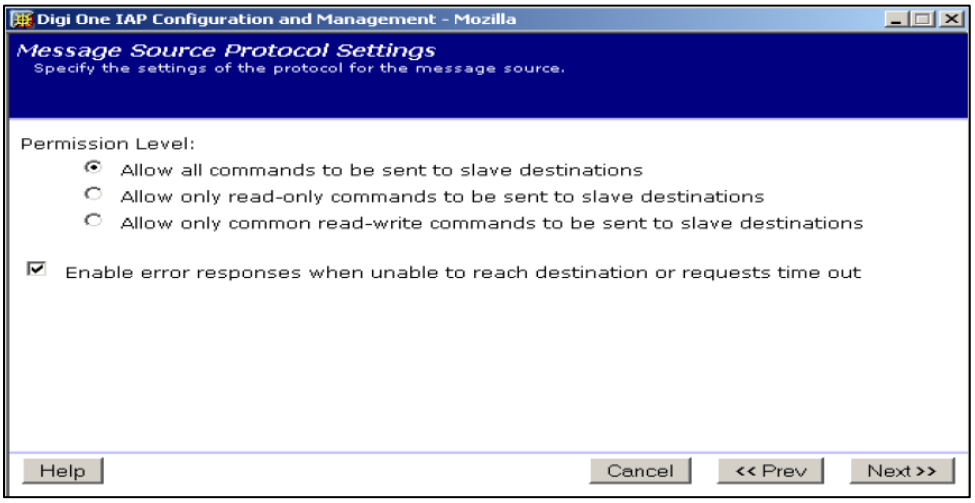
2.2.4 Define Modbus Ethernet/IP as the master or message source

1. When you see the information dialog box about Message Sources (not shown in this document), click **Next**.
2. Click **Receive messages from network devices connecting using the network**.
3. From the pulldown menu, select the Modbus/TCP **Protocol**. The **Network port** automatically sets to 502.
4. Click **Next**.



2.2.5 Message source protocol settings

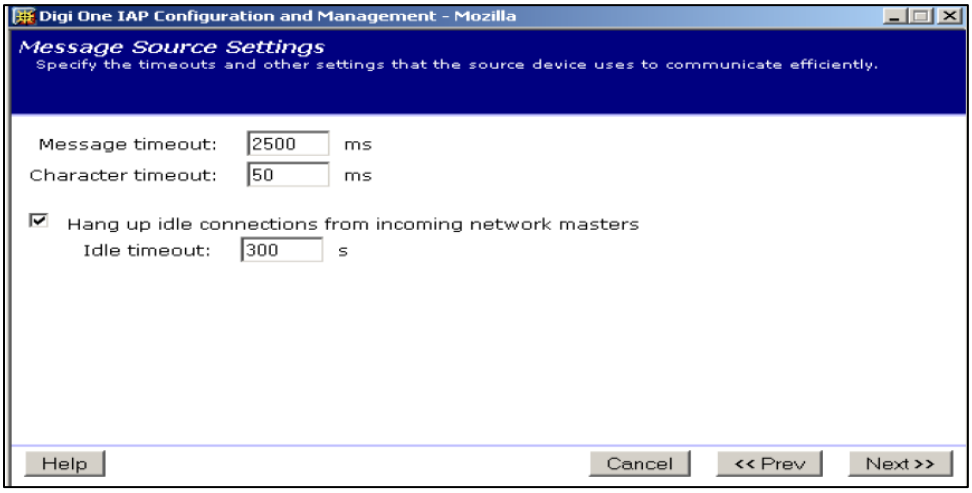
You don't need to make any changes in this window; the default settings are fine as is. Note that the **Permission Level is** set to allow all Modbus messages, and error responses (Modbus exception 0x0A and 0x0B) are enabled. Click **Next** >>.



2.2.6 Define timeouts for the incoming Modbus/TCP connection

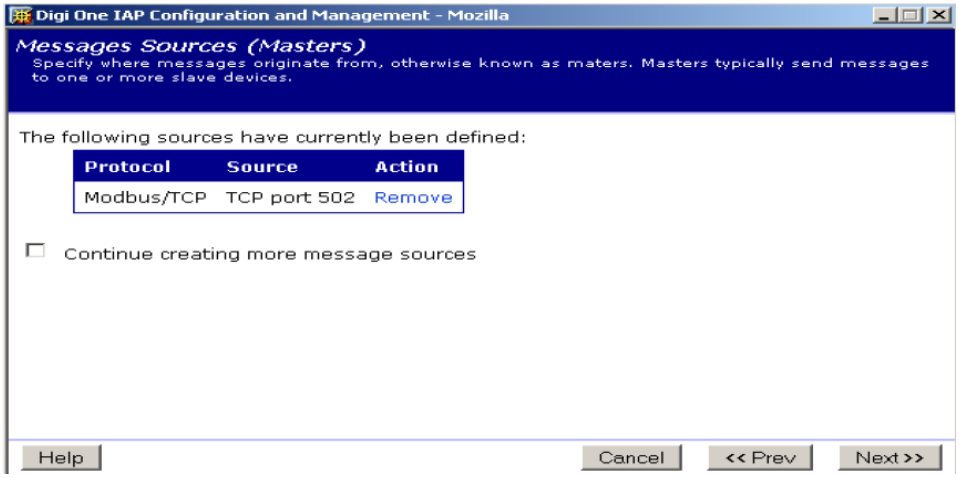
You don't need to make any changes in this window; the default settings are fine as is.

Click **Next** >>.



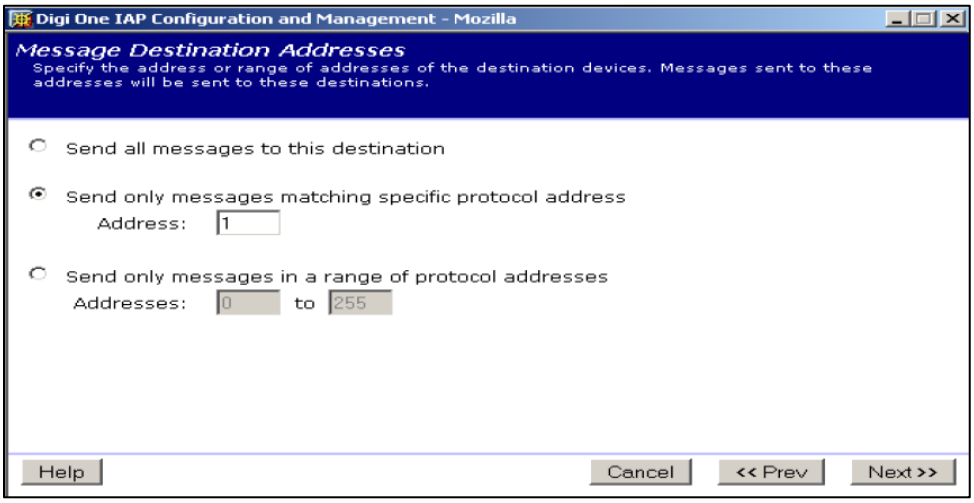
2.2.7 Master summary

1. (Note that this step is optional.) To define more masters, check **Continue creating more message sources**.
   In this example, no additional masters are required, but you could enable some Rockwell protocols. If you have only one ASCII device, you can also enable a serial master on the other Digi One IAP's serial port.
2. Click **Next**.



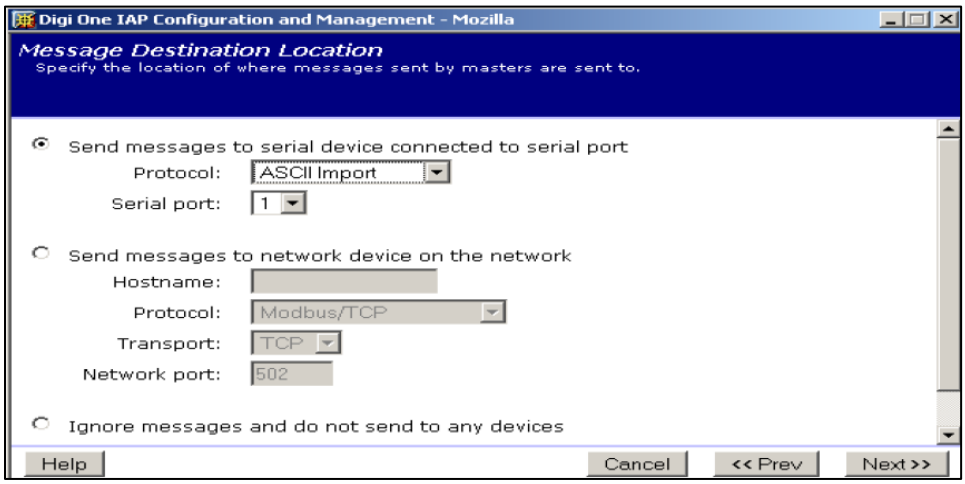
2.2.8 Define first message destination (serial port #1)

1. Click **Next** to pass an information dialog about **Message Destinations** and go to the window shown next.
2. Click **Send only messages matching specific protocol address**.
3. Enter **1** for the **Address**. This address is the slave address or unit ID from the Modbus requests. The first message destination does not have to be 1; it could be 9, 23, or any valid Modbus address.
4. Click **Next**.



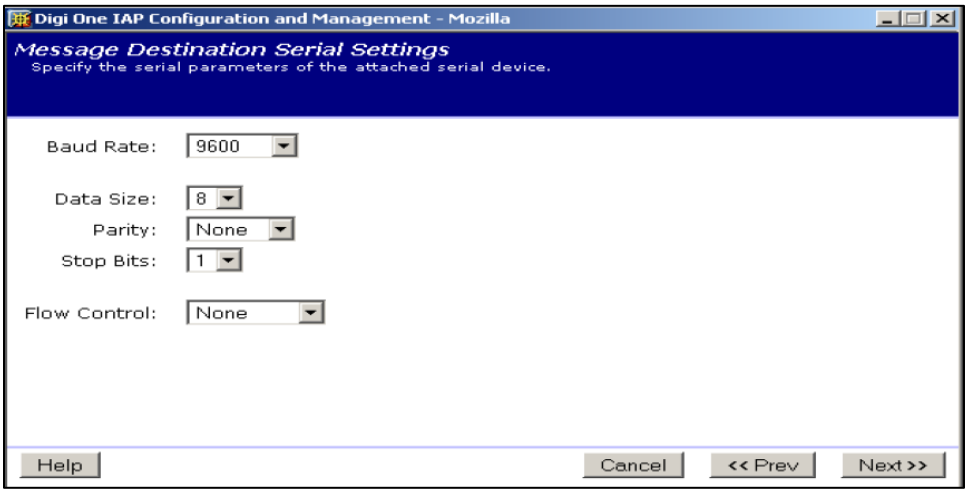
2.2.9 Set the protocol and port number

1. Click Send messages to serial device connected to serial port.
2. Select Serial Port 1 from the pulldown menu. Note that slave address of 1 and serial port 1 are independent. Setting slave address of 1 to serial port 2 is just as valid.
3. Click Next.



2.2.10 Set the serial port settings

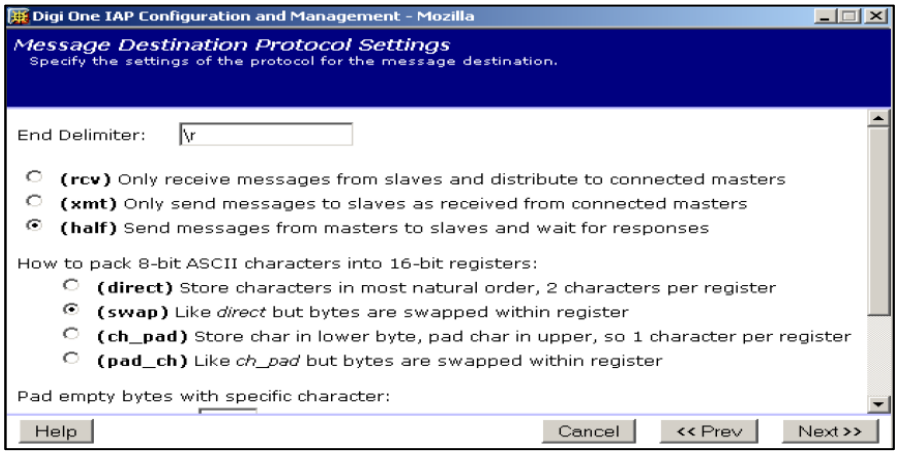
1. Select the serial port settings from the pulldown menus.
2. For most IA protocols, you must set **Flow Control** to **None**.
3. Click **Next**.

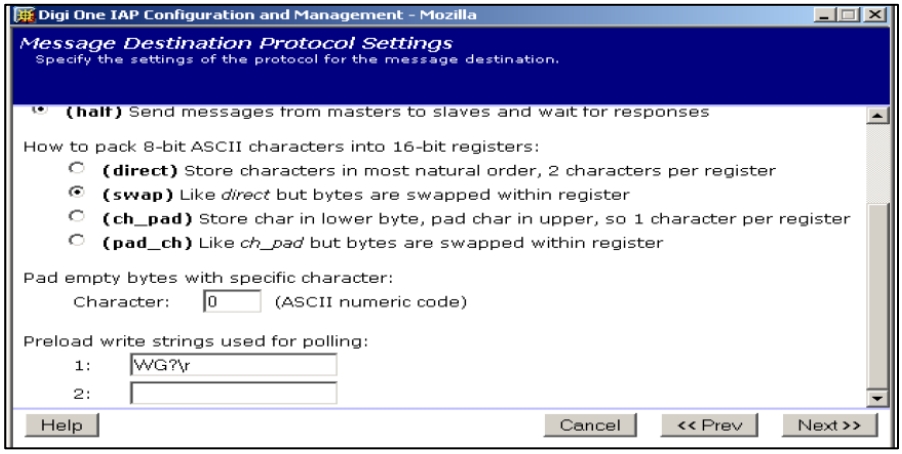


2.2.11 Set the ASCII Import-specific protocol settings

You will need to scroll through this window. For detailed help, including the format for characters such as carriage return, click **Help**.

1. In the **End Delimiter** input box, specify the delimiter (a carriage return) for this device entering \r.
2. In the example, click **half** and **swap**, leave **Pad empty bytes**… set to 0.
3. In the **Preload write strings used for polling** input box, predefine the poll string for the gross weight by entering WG?\r.



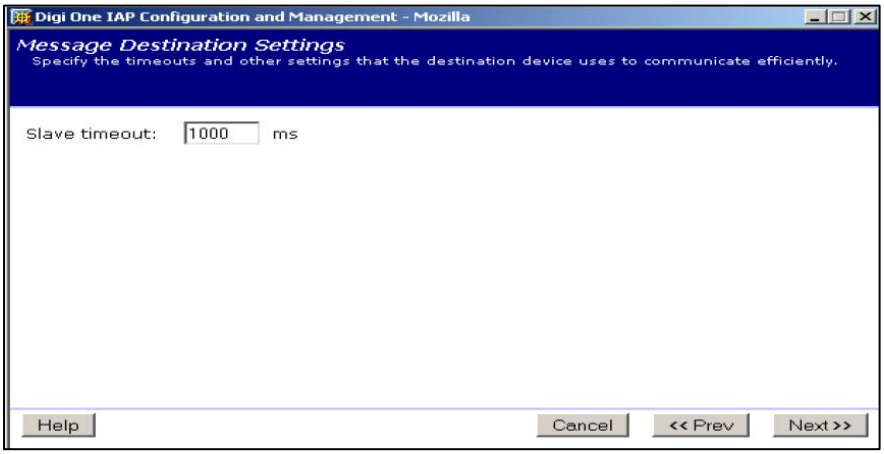


2.2.12 Set the serial port settings

Slave timeout applies only to half-duplex or receive-only ASCII Import modes. The table describes how slave timeout is used in the modes:

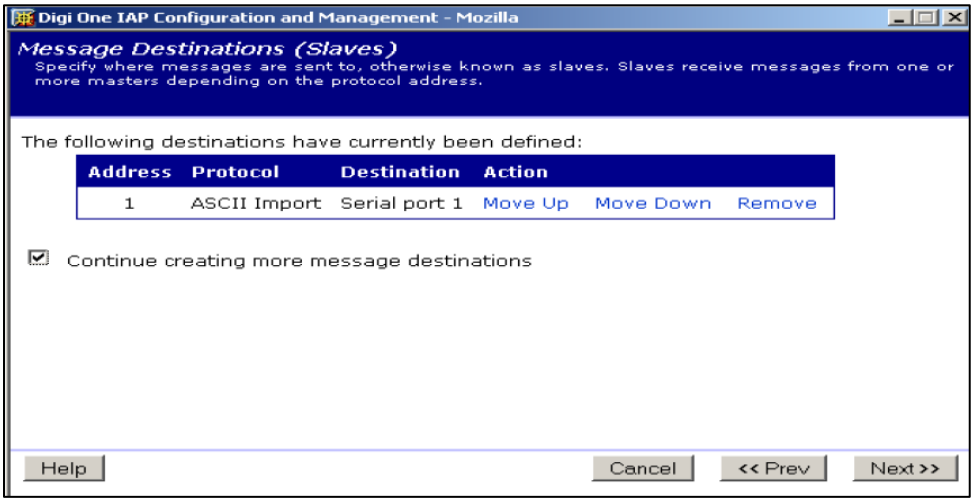
In this mode	Slave timeout is
Transmit-only	Ignored.
Half-duplex	The maximum time the Digi One IAP waits for the start of a slave process/.
Receive-only	Used as an age limit for received ASCII messages. If it is set to **0**, there is no age limit. Otherwise, it is the age in milliseconds that the received ASCII message can wait in queue to be read.

1. Enter a value in the **Slave timeout** input box.
2. Click **Next**.



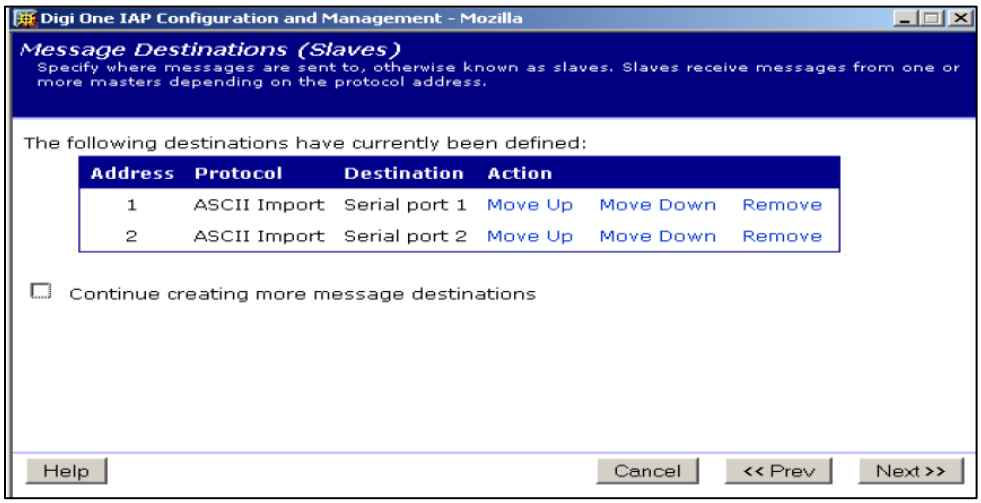
2.2.13 Repeat for second destination (serial port #2)

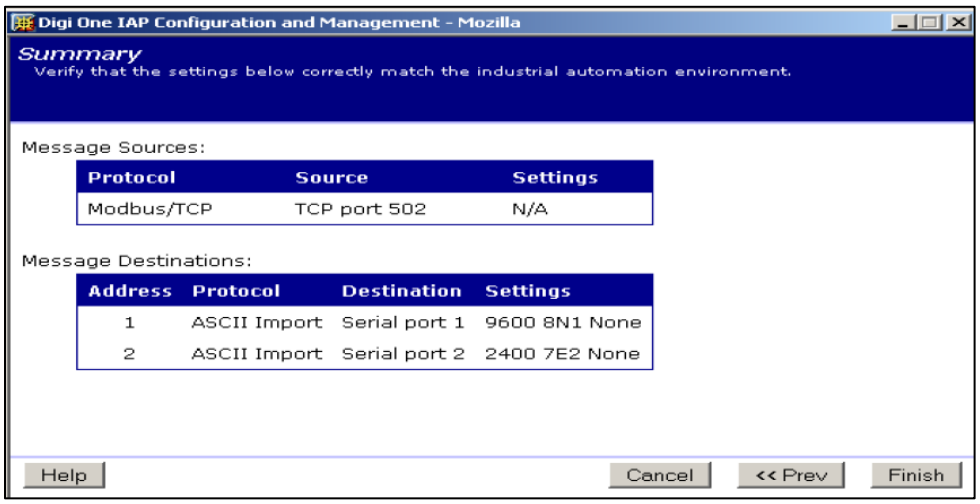
1. Click the Continue creating more message destinations checkbox.
2. Click Next.
3. Repeat the steps from 2.2.8 to 2.2.12 to the second serial port for ASCII Import and Modbus slave address 2. The LED sign uses transmit-only because you send it only strings to display.



2.2.14 Review the summary

1. After you set up port 2, uncheck the **Continue creating more message destinations** checkbox.
2. Click **Next**. A summary of the master and destinations setup appears.
3. Click **Next**.





2.3 Reboot the Digi One IAP

You can make minor changes to the Digi One IAP configuration without rebooting. However, when you change the number or type of master (message sources), or the number or type of slave (message destinations), it is safest to reboot. These changes, which affect the tasks running in the Digi One IAP RTOS, occasionally fail to take effect unless you reboot.

2.4 By telnet, release E PN

1. Use either HyperTerminal or telnet to log into your Digi One IAP.
2. Enter the IP address of your DS and the well-known telnet port of 23.
   You can use the text script shown next.
3. Cut and paste the script into a text editor like WordPad.
4. Edit it as required.
5. Cut and paste again into HyperTerminal using **Edit** > **Paste to Host menu**.

```
# clear all IA config revert ia=factory

# setup port 1 (screw term) as ascii_import (baud = 9600,8,N,1) set port ra=1 dev=ia
set line ra=1 baud=9600 csize=8 parity=N stopb=1
set ia serial=1 protocol=ascii_import type=slave table=1 set ia serial=1 slavetimeout=5sec
duplex=half packing=swap set ia serial=1 end="\r" prewrite1="WG?\r"

# setup port 2 (db9) as ascii_import (baud = 2400,7,E,2) set port ra=2 dev=ia
set line ra=2 baud=2400 csize=7 parity=E stopb=2
set ia serial=2 protocol=ascii_import type=slave table=1 set ia serial=2 duplex=xmt
packing=swap

# setup network for Modbus/TCP incoming
set ia master=1 active=on protocol=mbtcp transport=tcp ipport=502 table=1

# setup destination table set ia table=1 name=table1
set ia table=1 addroute=1 active=on protocol=ascii_import set ia table=1 route=1 protaddr=1
type=serial port=1

set ia table=1 addroute=2 active=on protocol=ascii_import set ia table=1 route=2 protaddr=2
type=serial port=2

# reboot the Digi One IAP boot action=reset
```

You can add the third and fourth routes to your table to enable use of Modbus slave address 3 or 4 to remote Digi One SP or Digi One IA. The ASCII handling is done within the local Digi One IAP, so the remote units can be a non-IAP device server. Note that the IP address is defined; you'll need to change it. The Digi One IAP supports up to 64 incoming or outgoing TCP sockets, so you should be able to connect to approximately 60 remote device servers.

```
# setup a remote Digi One SP or Digi One IA for receive-only ascii_import
# via MSG block Destination Node = 3 & 4

set ia table=1 addroute=3 active=on protocol=ascii_import

set ia table=1 route=3 protaddr=3 type=ip ipaddress=192.168.1.60

set ia table=1 route=3 ipport=2101 connect=active duplex=rcv slvtout=0ms set ia table=1 addroute=4 active=on
protocol=ascii_import

set ia table=1 route=4 protaddr=4 type=ip ipaddress=192.168.1.61 set ia table=1 route=4 ipport=2101 connect=active
duplex=xmt
```

3 Programming the Momentum MSTR block

The instructions in this application guide are based on the assumption that you understand the basics of programming and accessing your PLC. The examples in the next sections are taken from Concept v2.6 and a 984 ladder program for a 171CCC96020 (“E1M”) Momentum processor.

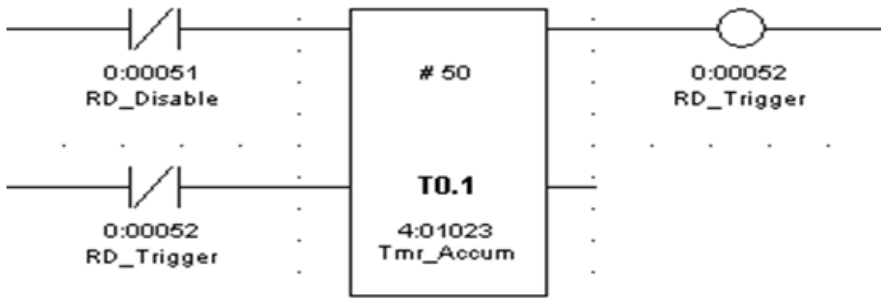
3.1 MSTR read using Modbus/TCP from an ASCII device

The ladder networks described in this section enable reading from the Digi One IAP. You can use the ladder networks for either receive-only or half-duplex.

3.1.1 Create the MSTR read block trigger

A timer is the most common form of trigger. Because the Modicon PLC can manage only a small number of MSTR blocks running concurrently, it may be necessary to chain your blocks to run in series instead of in parallel.

The ladder code below creates a cyclic (repeating) timer that triggers every five seconds (50 x 0.1 sec). Coil RD_Trigger is used to trigger the MSTR read, and setting coil **RD_Disable** to **1** disables the MSTR read function.



3.1.2 Suggested MSTR read network including exception response abort

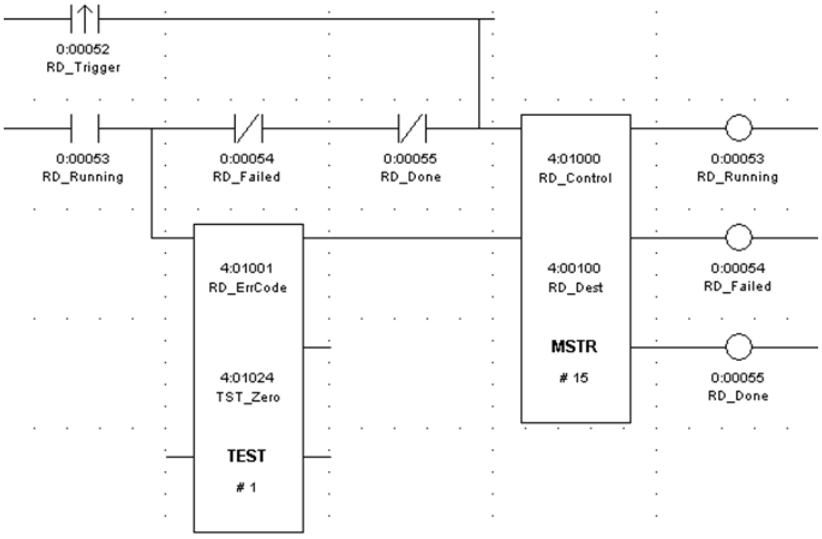
Ladder logic is a combination of art and science, and you can code the same functionality in many different ways. Because some design assumptions in the Modicon PLC differ from design assumptions within the Digi One IAP, Digi suggests that you follow this network design.

- Rung 1: Positive-transition contact RD_Trigger starts the MSTR block. Operation of the MSTR block is independent of the width of the trigger signal.
- Rung 2: This rung self-latches the MSTR block as long as:
    - RD_Running is true
    - Both RD_Failed and RD_Done are false

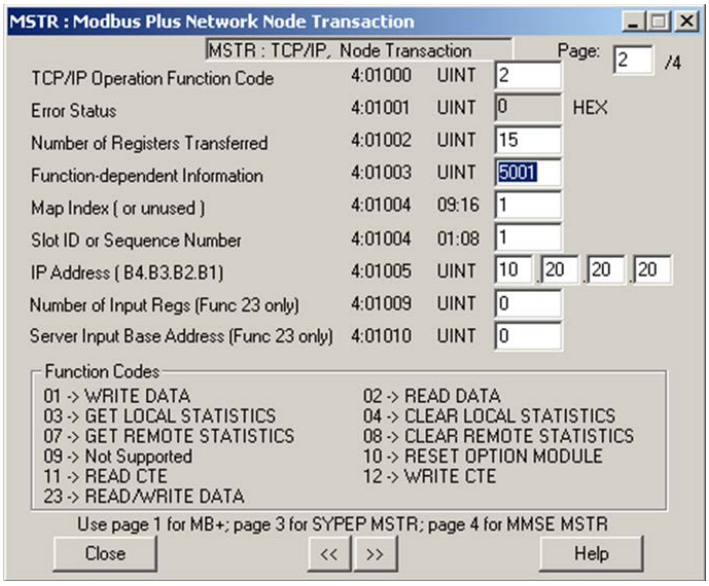
      This design causes a single Modbus/TCP read per trigger, and exception responses do not cause immediate retries. This is important for robust behavior with the Digi One IAP. RD_Running is true as long as the MSTR block is active.

- Rung 3: A TEST block aborts the MSTR block if an error code other than zero (0) is generated. Note that RD_ErrCode is the second register in the MSTR blocks control register space. TST_Zero holds the initialized constant zero (0).
- Rung 4: Coil RD_Done combined with RD_Failed can be used as a trigger for another MSTR block.

RD_Dest is defined as a ByteArr36 or 36-character byte array.



Here is the DX Zoom of the MSTR block. Page 2 of 4 is for Modbus/TCP reads using Ethernet. For descriptions of the fields, see the list after this screen.



Field descriptions:

- **TCP/IP Operation Function Code** is **2**.
- **Error Status** is read-only. Codes are in the form hex 300X, where X is 2, A, or B, indicates that an exception response was received. For example:

  - 3002 indicates that either the read is invalid or, for half-duplex, the poll buffer is empty.
  - 300A indicates that the slave address is not in the Digi One IAP destination table.
  - 300B indicates that the ASCII device is not responding.

  For descriptions of other values, see the Modicon documentation.

- **Number of Registers Transferred** is the number of registers to read.
- **Function-dependent Information** is the offset. The value of 5001 means read starting at holding register 4x05001 (offset 5000 on the wire). The Digi One IAP uses the register offset to select which buffer to use. (For more information, see section 4.)
- **Map Index** is the unit ID sent or slave address. Set this as needed according to the Digi One IAP's destination table you created—probably either **1** or **2**.
- **Slot ID** must be **1**, to indicate the first Ethernet port on Momentum.
- **IP Address** is a standard IP address, such as 192.168.1.23 or 10.20.20.20.
- The Digi One IAP does not support function 23, so leave the last two entries set at zero.

3.2 MSTR write using Modbus/TCP to an ASCII device

The ladder networks to write are the same as to read except as noted next.

3.2.1 Create the MSTR write block trigger

A timer also can be used to write. Because the Modicon PLC can manage only a small number of MSTR blocks running concurrently, you can chain the blocks to run in series and not in parallel.

3.2.2 Suggested MSTR write network including exception response abort

This network is the same as that shown in section 3.1.2. The only differences in the MSTR "DX Zoom" settings are:

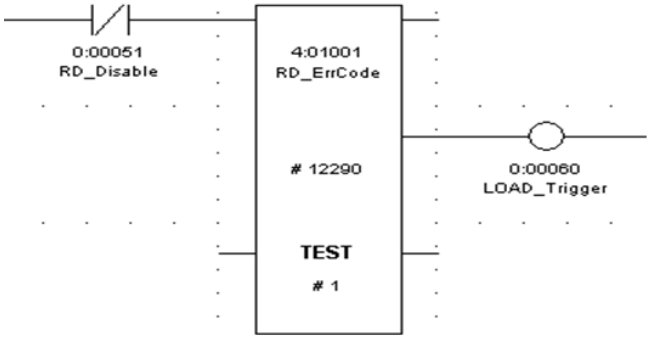
- **TCP/IP Operation Function Code** is **1.**
- The middle node is the destination holding registers, not the source registers.

3.3 MSTR write using Modbus/TCP to load the half-duplex poll

The design of the Digi One IAP in half-duplex mode requires that the poll request be written to the Digi One IAP before a Modbus read triggers the half-duplex poll-response cycle. The first two (of 10) buffers allow a non-volatile load of 12 bytes each. For larger poll strings, or for the remaining eight buffers, the Modbus master must write the strings to the Digi One IAP. The next section shows two networks that sense and trigger such a reload if the Digi One IAP returns an exception code 2 in response to a read in half-duplex mode.

3.3.1 Create the MSTR Reload block trigger

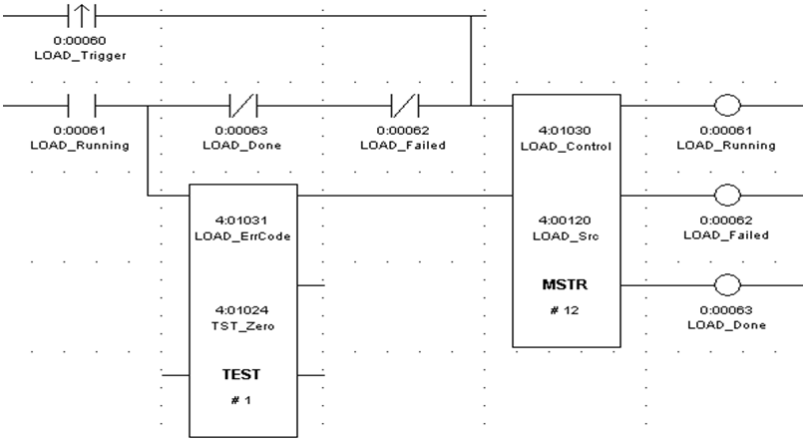
This TEST block watches for an error code of hex 3002 (decimal 12290) in the associated MSTR read block. If an error is seen and RD_Disable is 0, the coil LOAD_Trigger is set to **true** to trigger the Modbus write to reload the Digi One IAP.



3.3.2 Suggested MSTR write network including exception response abort

This network is the same as the one shown in section 3.1.2. The only differences in the MSTR "DX Zoom" settings are:

- **TCP/IP Operation Function Code** is 1.
- The middle node is the destination holding registers, not the source.



4 Application examples: Putting it all together

4.1 Using Modbus/TCP for receive-only from an ASCII device

You can configure an ASCII device such as a barcode scanner to send only ASCII strings. Modbus reads are used to fetch the strings.

4.1.1 Digi One IAP

Configure the serial port for ASCII Import and receive-only. In this mode, the Digi One IAP receives and queues at least 5000 ASCII strings.

If you have configured an **end pattern** (such as carriage return or "\r"), it is used to determine the break between strings. Otherwise, the **character timeout** is used to detect an idle gap between strings. The default character timeout is 50 milliseconds.

The **slave timeout** has special meaning in this mode. When this value is set to zero (0), the strings are queued until either they are read or the Digi One IAP is powered off. Therefore, a Modbus master reading the Digi One IAP might see strings that are days or even months old. Setting the slave timeout anywhere from 1 to 65,535 defines an age in milliseconds after which the string is discarded. Setting the slave timeout to 1000 milliseconds specifies that a Modbus master sees only strings that have been received within the last second.

Q: *What if no strings are waiting?* If no strings are waiting, the response is fully padded. The default pad byte is zero or null, but you can change this to any 8-bit value.

Q: *Can I use an external hardware trigger?* Many barcode scanners allow a digital input to trigger a scan. You will find this problematic because of both the time it takes the Modbus read request to be processed and the time it takes to visually detect the barcode is variable. The result is a race condition, and you could miss the string at times. The most robust solution is a half-duplex poll/response, if your ASCII device supports this.

Q: *Can the Digi One IAP parse the string into a number?* No. The Digi One IAP always returns the ASCII string packed as registers. You need to process the string within your PLC program.

Q: *Can the Digi One IAP push or write the string into the PLC?* No. The Digi One IAP is always the slave here. The PLC must read the Digi One IAP.

Q: *Can multiple masters share an ASCII device?* Not really. Because only one queue exists, concurrent masters reading see an unpredictable subset of the strings.

4.1.2 Modicon MSTR block

Follow the ladder example in section 3.1.1 to create a MSTR read block. You are not required to load or write a string to the Digi One IAP.

If you are reading the ASCII string into a structured byte array, use either of these steps:

- Set the Digi One IAP to pack the bytes as swapped, not direct.
- Select to pack as ch_pad or pad_ch to put one ASCII character per register.

4.1.3 Modbus register offsets to use

You can read from 1 to 125 registers (2 to 250 characters) using Modbus function 3. The next string in queue is returned as the read response. The string is truncated if it is longer than the

read and padded if it is shorter. The concept of 10 buffers is ignored, so only the offset 4x05001 is used when you are sending pre-formatted strings exactly as the ASCII device requires. Multiple masters can safely share a single ASCII device in this mode.

Duplex Setting	Elements	Buffer	Read	Write
rcv/receive-only	Max 250 bytes	n/a	4x05001	Error

4.2 Using Modbus/TCP for transmit-only to an ASCII device

You can configure an ASCII device such as an LED sign or serial printer to receive only ASCII strings. Modbus writes are used to send the strings. Any responses are discarded.

4.2.1 Digi One IAP

Configure the serial port for ASCII Import and Transmit Only. Most other settings are ignored. By default, the ASCII string is assumed to end at the first **pad character**, which usually is zero or null. Sending a short string like "HELLO" as any number of registers (for example, 3, 30 or 100) results in the same six characters being sent.

Q: *Can multiple masters share an ASCII device as transmit only?* Yes. The Digi One IAP properly handles many concurrent masters writing to the ASCII device. You cannot predict the exact order in which the strings will be seen by the ASCII device. However, every Modbus write is sent as one continuous ASCII message.

4.2.2 Modicon MSTR block

Follow the ladder example in section 3.1.2 to create a MSTR write block. You are not required to read any data from the Digi One IAP because it discards any response from the ASCII device in this mode.

If you are writing the ASCII string from a structured byte array, use either of these steps:

- Set the Digi One IAP to pack the bytes as swapped, and not direct.
- Select to pack as ch_pad or pad_ch to send one ASCII character per register.

4.2.3 Modbus register offsets to use (preformed strings)

You can write from 1 to 122 registers using Modbus function 16. Trailing pad bytes (by default zero or NUL) are truncated and not sent. The concept of 10 buffers is ignored, so only the offset 4x00001 is used when you are sending pre-formatted strings exactly as the ASCII device requires. Multiple masters can safely share a single ASCII device in this mode.

Duplex Setting	Elements	Buffer	Read	Write
xmt/transmit-only	Max 244 bytes	n/a	Error	4x00001

4.3 Using Modbus/TCP for half-duplex to an ASCII device

Many ASCII devices—such as barcode scanners or weight scales—require a poll before they send a response. A Modbus write loads the poll into the Digi One IAP. Subsequent Modbus reads cause this poll to be repeated and the response returned as the Modbus read response.

4.3.1 Digi One IAP

Configure the serial port for ASCII Import and half-duplex. Set the other parameters as needed. For details, see the web user interface help.

The first two buffers include non-volatile initial values up to 12 characters (the prewrite1 and prewrite2). Poll strings for buffers three to 10, or longer than 12 characters, must be dynamically written as needed. The poll buffer is limited to 256 bytes.

Q: *Can multiple masters share an ASCII device as half-duplex?* Yes, but all masters share the 10 buffers per serial port. Because a time gap exists between the Modbus write to load the poll and the Modbus read to send the poll, multiple masters must either:

- All share the same poll
- Use distinct buffers to hold their distinct polls

4.3.2 Modicon MSTR block

You will probably require two MSTR blocks per slave poll:

- An MSTR write to load the poll
- An MSTR read to use the poll and obtain a response
- For more information, see sections 3.3 and 3.1.

You must reload the poll string if an exception response 2 is returned to the read.

4.3.3 Modbus register offsets to use (preformed poll strings)

You can write from one to 122 registers (two to 244 bytes) using Modbus function 16.

You can read from one to 125 registers (two to 250 bytes) using Modbus function 3.

Function	Elements	Buffe	Read	Write
Load/read poll string	Max 244 bytes	1	4x00001	4x00001
Load/read poll string	Max 244 bytes	2	4x00101	4x00101
Load/read poll string	Max 244 bytes	3	4x00201	4x00201
Load/read poll string	Max 244 bytes	4	4x00301	4x00301
Load/read poll string	Max 244 bytes	5	4x00401	4x00401
Load/read poll string	Max 244 bytes	6	4x00501	4x00501
Load/read poll string	Max 244 bytes	7	4x00601	4x00601
Load/read poll string	Max 244 bytes	8	4x00701	4x00701
Load/read poll string	Max 244 bytes	9	4x00801	4x00801

Function	Elements	Buffe	Read	Write
Load/read poll string	Max 244 bytes	10	4x00901	4x00901
Issue poll/return response	Max 250 bytes	1	4x05001	error
Issue poll/return response	Max 250 bytes	2	4x05101	error
Issue poll/return response	Max 250 bytes	3	4x05201	error
Issue poll/return response	Max 250 bytes	4	4x05301	error
Issue poll/return response	Max 250 bytes	5	4x05401	error
Issue poll/return response	Max 250 bytes	6	4x05501	error
Issue poll/return response	Max 250 bytes	7	4x05601	error
Issue poll/return response	Max 250 bytes	8	4x05701	error
Issue poll/return response	Max 250 bytes	9	4x05801	error
Issue poll/return response	Max 250 bytes	10	4x05901	error

4.3.4 Modbus register offsets to use (for Digi One IAP forming of strings)

Sending fixed or constant strings from a PLC is easy. However, formatting strings such as “AMT002345q\r” to instruct a motion controller to move where the number changes frequently can be difficult in PLC ladder logic.

The Digi One IAP provides special support for formatting. It supports common “C” language printf-style formatting commands. So the poll in the previous paragraph is loaded as “AMT%06dq\r”. After loading the poll, use a Modbus write to send integers to be formatted. From one to four values are supported per string, so format string “1:%d 2:%02x 3:%04d 4:%c” would expect four integers to be written each time it is to be formatted.

command	Description	example
%d	Signed decimal. Width is as required.	1 or -99
%04d	Signed decimal with fixed-width zero-padded.	0001 or 0175
%u	Unsigned decimal. Width is as required.	0 or 17556
%02u	Unsigned decimal with fixed-width zero-padded.	01 or 75
%x	Hexadecimal 0-9,a-f. Width is as required.	0 or 1abf6

%04x	Hexadecimal 0-9,a-f. Fixed-width zero-padded.	0001 or 7fff
%X	Hexadecimal 0-9,A-F. Width is as required.	0 or 1ABF6
%04X	Hexadecimal 0-9,A-F. Fixed-width zero-padded.	0001 or 7FFF
%c	Send a single ASCII character.	A or ~
%%	Send a single '%' character.	0001 or 7FFF

Use a Modbus write to 4x00n01—where 'n' selects the buffer—to load a poll string. The string is not sent out the serial port.

Function	Elements	Buffer	Read	Write
Load/read poll string	Max 244 bytes	1	4x00001	4x00001
Load/read poll string	Max 244 bytes	2	4x00101	4x00101
Load/read poll string	Max 244 bytes	3	4x00201	4x00201
Load/read poll string	Max 244 bytes	4	4x00301	4x00301
Load/read poll string	Max 244 bytes	5	4x00401	4x00401
Load/read poll string	Max 244 bytes	6	4x00501	4x00501
Load/read poll string	Max 244 bytes	7	4x00601	4x00601
Load/read poll string	Max 244 bytes	8	4x00701	4x00701
Load/read poll string	Max 244 bytes	9	4x00801	4x00801
Load/read poll string	Max 244 bytes	10	4x00901	4x00901

Use a Modbus write to 4x02n01—where 'n' selects the buffer—to send from one to four integers to be formatted.

Note that:

Sending too many integers for your format string causes the extra integers to be ignored.

Sending too few integers for your format string causes unpredictable values to be used for the missing integers, and the formatted string is not sent out the serial port.

Function	Elements	Buffer	Read	Write
Have integers formatted	1 to 4 words	1	error	4x02001
Have integers formatted	1 to 4 words	2	error	4x02101
Have integers formatted	1 to 4 words	3	error	4x02201
Have integers formatted	1 to 4 words	4	error	4x02301
Have integers formatted	1 to 4 words	5	error	4x02401
Have integers formatted	1 to 4 words	6	error	4x02501
Have integers formatted	1 to 4 words	7	error	4x02601
Have integers formatted	1 to 4 words	8	error	4x02701

Function	Elements	Buffer	Read	Write
Have integers formatted	1 to 4 words	9	error	4x02801
Have integers formatted	1 to 4 words	10	error	4x02901

Use a Modbus read to 4x05n01—where 'n' selects the buffer—to send the last formatted string as a poll and return the response.

Function	Elements	Buffer	Read	Write
Issue poll/return response	Max 250 bytes	1	4x0500	error
Issue poll/return response	Max 250 bytes	2	4x0510	error
Issue poll/return response	Max 250 bytes	3	4x0520	error
Issue poll/return response	Max 250 bytes	4	4x0530	error
Issue poll/return response	Max 250 bytes	5	4x0540	error
Issue poll/return response	Max 250 bytes	6	4x0550	error
Issue poll/return response	Max 250 bytes	7	4x0560	error
Issue poll/return response	Max 250 bytes	8	4x0570	error
Issue poll/return response	Max 250 bytes	9	4x0580	error
Issue poll/return response	Max 250 bytes	10	4x0590	error

5 Troubleshooting and FAQ

5.1 My ASCII is showing up funny; "Stop the Press" becomes "tSpoT ehP erss."

This is controlled by the packing parameter ("packing=direct" in CLI example). The settings are referred to as **direct**, **swap**, **ch_pad**, and **pad_ch**. Because of the nature of big versus little-endian designs—and because some protocols or applications treat text differently—there is no absolute way to define this packing.

So if you have "packing=direct" and you see this problem, set "packing=swap" or vice-versa. These two settings are the byte-swap of each other.

If you have "packing=ch_pad" and you see this problem, set "packing=pad_ch" or vice-versa. These two are the byte-swap of each other.

5.2 Can I use the Modbus function 23 to read/write in a single command?

No. The Digi One IAP currently does not support this function.