CyFlex ® GModBus Setup

Version 7

February 14, 2024

Developed by Transportation Laboratories

Version History

Version	Date	Revision Description
1	10/1/2015	Initial publication
2	8/23/2018	Format with SGS brand
3	4/3/2020	Refreshed example specifications in *Section 4.1 Identifying Information in the File* Retrofit to new template
4	12/2/2012	Removed `gmodbus` usage content from *Section 5 Device Driver Commands* on page 11 and added a hypertext link to its usage on cyflex.com.
5	4/21/2022	Revised hypertext linked cross-reference to `gmodbus` usage help on cyflex.com in *Section 5* Device Driver Commands on page 11 to align with its updated category.
6	6/2/2022	Updated hypertext linked cross-reference to `gmodbus` usage help on cyflex.com in *Section 5* Device Driver Commands on page 11
7	2/14/2024	Rebrand to TRP Laboratories

Document Conventions

This document uses the following typographic and syntax conventions.

- Commands, command options, file names or any user-entered input appear in Courier type. Variables appear in Courier italic type.
  Example: Select the `cmdapp-`*`relVersion-buildVersion`*`.zip` file….
- User interface elements, such as field names, button names, menus, menu commands, and items in clickable dropdown lists, appear in Arial bold type.
  Example: **Type**: Click **Select Type** to display drop-down menu options.
- Cross-references are designated in Arial italics.
  Example: Refer to *Figure 1*…
- Click intra-document cross-references and page references to display the stated destination.
  Example: Refer to *Section 1 Overview* on page 1.
  The clickable cross-references in the preceding example are *1*, *Overview*, and on page 1.

CyFlex Documentation

CyFlex documentation is available at https://cyflex.com/. View **Help & Docs** topics or use the **Search** facility to find topics of interest.

Table of Contents

LIST OF TABLES

1 Overview

This document explains how to configure a test cell computer running CyFlex software for connecting to and controlling devices that support the Modbus protocol. This includes Intelligent Electronic Devices (IED) such as Programmable Logic Controllers (PLC). CyFlex communicates with the devices over a network, automating electro-mechanical processes from the test cell.

Important:

Determine whether a particular device will operate with the Modbus protocol supported by CyFlex by testing the device using the polling utility available at:
http://www.focus-sw.com/fieldtalk/modpoll.html

1.1 Modbus Background

1.1.1 Modbus TCP

Modbus evolved from previous protocols. It utilizes the Modbus Transmission Control Protocol (TCP) communication standard, which is basically the Modbus RTU protocol with a TCP interface that runs on the Ethernet.

The original Modbus was a serial communications protocol designed several years ago for use with PLCs. A standard for connecting industrial electronic devices, it supports communication between multiple devices on a network.

The more recent Modbus TCP protocol is based on a client-server structure. It operates over an Ethernet network, and is often made up of several clients and servers. Several Modbus networks can be implemented on the same platform. With the Modbus TCP protocol, only the client can prompt communication (with the server), by sending a request and waiting for an answer. The TCP layer controls the validity of the message exchange.

The GModBus driver works with the TCP layer and supports adding a computer into the network. The computer can act as a client or server.

1.1.2 Modbus RTU

Modbus Remote Terminal Unit (RTU) utilizes a RS-485 serial interface, which specifies the electrical characteristics of the generator and the receiver. RS-485 is not a communications protocol.

RS-485 uses differential signaling, commonly over a “twisted pair” of wires, for transmitting data. Using differential signals eliminates noise and allows for long cables.

A RS-485 network can be configured in two ways: "2-wire" or "4-wire." In a "2-wire" network, the transmitter and receiver of each device are connected using a twisted pair. In a "4-wire" network, the master’s transmitter is connected to each of the slave receivers on one twisted pair. The slaves’ transmitters are all connected to the master’s receiver on a second twisted pair. In either configuration, devices can be addressed and each node communicated with independently. RS-485 is usually a 2-wire system, although some manufacturers may specify 4-wire RS-485, which is less common.

2 Installation Requirements

The following are required for installation:

- Test cell computer running CyFlex
- Network connection (Ethernet or RS-485)
- GModBus driver application
- Device specification file(s)
- Go-script

2.1 Physical Connection

The GModBus device driver application supports connecting the test cell to a TCP network via the computer's Ethernet port, or to an RTU network through a RS-485 serial interface.

2.2 Communication Parameters

Modbus RTU requires the following information:

- Baud rate
- Character format (for example, 8 bits no parity, etc.)
- Slave ID (slave index, address, unit number, or unit ID)

Modbus TCP requires the IP address of the server (slave or host)

2.3 Driver and Specification Files

Table 1 lists the files needed for setting up GModBus on the test cell computer. The driver and specification files are copied to the test cell during the Scientific Linux (SL) 6.3 installation. Once CyFlex is installed and the cell is up and running, users typically create or copy a Go-script from another cell; customized for the tasks the cell performs. This avoids manually modifying the file. The same may be true for a device spec file depending on the variables the device measures. If a suitable spec file is not available for copying, TRP Laboratories can provide a "template" file as a starting point.

In testing scenarios that employ multiple PLCs, for example, users can create more than one instance of the `gmodbus_specs` file. This means one spec file for each copy of the driver program being run, to configure communication with PLCs that perform different functions and consequently use different sets of variables. This can help simplify managing several devices.

Table 1: GModBus Setup Files

File Type	Filename	Directory
GModBus driver	`gmodbus`	`/cyflex/bin/`

Device specification	`gmodbus_specs.<tc_number>`
	Example:
	`gmodbus_specs.tc1`
	`<tc_number>` is the test cell number
		`/specs/`

Go-script	`go.scp`	`/cell/`

3 Modbus Protocol

Modbus is a master/slave protocol, which means a “master” device polls a “slave” device(s). In this case, the slave device cannot volunteer information; it must wait to be asked for it. The master writes data to the slave’s registers and reads data from the slave’s registers. A register address is always in the context of the slave’s registers.

In Modbus TCP, Ethernet allows peer to peer communication. In Ethernet based networking, “client” and “server” are more commonly recognized than master and slave. In this context, the master becomes the client and the slave becomes the server.

There can be more than one client obtaining data from a server (depending on connections available on the server). In “Modbus terms,” this means there can be multiple masters as well as multiple slaves. With Modbus TCP, instead of defining master and slave by a device-by-device basis, the system designer creates logical associations between master and slave functionality.

3.1 Modbus Register Types

Table 2 lists the types of registers used in Modbus-supported devices. Whether a particular device includes all of these register types is up to the device manufacturer. It is common to find all I/O mapped to holding registers only.

Table 2: Modbus Register Types

Register Type	Bit	Read / Write	Description
Coil (Discrete Output)	1-bit	Read or write	Used to control discrete outputs
Discrete Input	1-bit	Read only	Used as inputs
Input Register	16-bit	Read only	Used as inputs
Holding Register	16-bit	Read or write	Used for inputs, outputs, configuration data, or any other requirement for "holding" data

3.2 Device Register Map

Modbus protocol does not provide a means for registers to automatically identify themselves. To identify registers for a particular device such as a PLC, consult the equipment manufacturer’s documentation. It should include a register map for the device.

3.3 Modbus Function Codes

Modbus protocol defines several function (command) codes for accessing Modbus registers. Modbus also defines four different data blocks, and the addresses or register numbers in the data blocks “overlap”. For this reason, defining where to find a piece of data requires both the address (register number) and the function code (register type).

Table 3 on page 4 lists the function codes most commonly recognized by Modbus devices. Each function code identifies a register type in the message.

Table 3: Modbus Function Codes

Function Code	Register Type
1	Read Coil
2	Read Discrete Input
3	Read Holding Registers
4	Read Input Registers
5	Write Single Coil
6	Write Single Holding Register
15	Write Multiple Coils
16	Write Multiple Holding Registers

4 Device Specification File

The example device specification file described in this section is for a PLC. The spec file must be configured for the particular test setup and can be modified to work with most PLCs. PLC-specific information is required for each PLC.

To find out whether a particular device will work with the Modbus protocol supported by CyFlex, test the device using the utility mentioned in *Section 1 Overview* on page 1

If connecting to a device other than a PLC, TRP Laboratories can provide a “template” spec file as a starting point if needed.

4.1 Identifying Information in the File

Table 4 describes the information in the `gmodbus_specs` file. Refer to the example spec file on page 7.

Table 4: gmod_specs Content

Name	Description
CyFlex variable	Basic unit of information, often changing in value
CyFlex unit	Unit of measure used by CyFlex
Modbus unit	Unit of measure in the Modbus device
Input/Output (I/O)	PLC I/O – read from hold address
Update rate	Frequency the variable value is updated (in ms)
Slave Index	PLC index (slave address) for data sorting
Mbus address	Modbus address of a device. Modbus can connect to multiple devices, and send messages tagged for certain devices. Note: This is usually “0” for Modbus/TCP.
Register	Register number for a variable in the Modbus device
Type	PLC I/O ports (register formats): `b` = bit (coil or discrete input) `d` = double precision `s` = short (16-bit signed) `l`= long (32-bit signed) `f`= float (32-bit IEEE float) `us` = unsigned short (16-bit unsigned) `ul` = unsigned long (32-bit unsigned)
Scale slope	Raw values are sometimes scaled to represent the actual floating-point values, saving space in data messages.

Name	Description
Scale offset	This is usually provided by the manufacturer. Example: An engine sensor measures values in a range from -25.0 to 62.5, and in 0.25 increments. Scale offset = -25.0 Scale slope = 0.25
Server (slave) IP	For Ethernet/TCP connections, IP address of the PLC
Other Information	
Endian	Big endian and little endian describe the order in which a sequence of bytes are stored in computer memory. Big endian is an order in which the "big end" (most significant value in the sequence) is stored first (at the lowest storage address). Little endian is an order in which the "little end" (least significant value in the sequence) is stored first.

Each time the `gmodbus` driver program is started, a copy runs on the system. Each copy is an instance. All instances are based on the same code but are configured according to the associated specification file. The spec file can be modified to contain instructions for one instance of the driver or several instances. In order for a spec file to configure the driver, the registered name of the driver instance must be listed in the spec file.

In the example specification file on page 7 (for RS-485), the registered name of the GModBus instance is `GM_SEEC`.

Example specification file for RS-485 serial connection;

```
@REG_NAME
GM_SEEC

#          nth slave             device /dev/ser1[,9600,8,N,1endian motorola=big
# 0        /dev/rckt5,38400,8,1,N big

# Some devices get both the inputs and outputs from what is normally considered the “output area” of a Modbus device.
# To support this, place an “i0” in the Input/Output column in the spec file. Normally, only an “i” or an “o” would be placed in this column.
#
# Reference manufacturer’s documentation for the device:
#    i  -> Read from Input Register table 3:0000
#    o  -> Write to Hold Register table 4:0000
#    i0 -> Read from Hold Register table 4:0000
#
# The exceptions are discretes (type b); inputs (i) are -> table 1:0000
# and outputs (o) -> table 0:0000 and read output coils (i0) -> table 0:0000

$
# CyFlex Variable   CyFlex Units    Modbus Units   Input/Output   Update Rate  (ms) Slave Index    Mbus address   Register   Type     Scale Slope     Scale Offset
# Name              PAM Unit        PAM Unit       i              2-1000             0-n             1-255          1          b,s,l,f  10.             0.0
# my_fuel_consumed  lb              lb             o              1000               0               12             1          us       100.0           0.0
  fl_flw_sup        lb              lb             o              1000               0               12             1          us       100.0           0.0
  fuel_density      lb/ft3          lb/gal         o              1000               0               12             2          us       1000            0.0
  ctl_spd           rpm             rpm            o              1000               0               12             3          us       1.0             0.0
  AbateTorq         lb_ft           lb_ft          o              1000               0               12             4          us       1.0             0.0
  tur_ot_T          deg_F           deg_F          o              1000               0               12             5          us       1.0             0.0
  AbateReq          None            None           o              1000               0               12             6          us       1.0             0.0
  Dspl              in3             liter          o              1000               0               12             7          us       1.0             0.0
  EngineMaxPower    hp              hp             o              1000               0               12             8          us       1.0             0.0
  FuelFlowRate      lb/hr           lb/hr          o              1000               0               12             9          us       10.             0.0
  fuel_typ          None            None           o              1000               0               12             10         us       1.0             0.0
  InAirFlow         lb/hr           lb/hr          o              1000               0               12             11         us       1.0             0.0
  RollTransCnt      None            None           o              1000               0               12             12         us       1.0             0.0
  ECMFuelRate       lb/hr           lb/hr          o              1000               0               12             13         us       10.             0.0
  ECMChargeFlow     lb/hr           lb/hr          o              1000               0               12             14         us       1.0             0.0
  AbateIntMnfP      in_hg           psi            o              1000               0               12             15         us       100.            0.0
  int_mnf_T         deg_F           deg_F          o              1000               0               12             16         us       1.0             0.0
  AbateLogi         none            none           o              1000               0               12             17         us       1.0             0.0
  Spare_2           none            none           o              1000               0               12             18         us       1.0             0.0
  Spare_3           none            none           o              1000               0               12             19         us       1.0             0.0
  Spare_4           none            none           o              1000               0               12             20         us       1.0             0.0

  AbateErrCode      None            None           i              1000               0               12             1          s        1.0             0.0
  AbateSts          None            None           i              1000               0               12             2          s        1.0             0.0
  T2                deg_F           deg_F          i              1000               0               12             3          s        1.0             0.0
  T3                deg_F           deg_F          i              1000               0               12             4          s        1.0             0.0
  T4                deg_F           deg_F          i              1000               0               12             5          s        1.0             0.0
  AbateNOxEff       %               %              i              1000               0               12             6          s        0.01            0.0
  AbateShtdwnTm     sec             sec            i              1000               0               12             7          s        1.0             0.0
  AbateNumErrs      none            none           I              1000               0               12             8          s        1.0             0.0
  Spare_30          none            none           i              1000               0               12             9          s        1.0             0.0
  Spare_40          none            none           i              1000               0               12             10         s        1.0             0.0
```

Example specification file – Ethernet connection:

```
# index       Modbus slave interface  endian
# hostIP[:port]                       default port is 502 intel=little
#             device /dev/ser1[,9600,8,1,E] motorola=big
#
0 192.168.222.200:502      big
#   1            /dev/ser1     little
# optional arguments:    [RS485] [timeout] [polldelay]
#   2            /dev/ser2     little    RS485 100 100

$
#
# Some devices get both the inputs and outputs from what is normally considered the “output area” of a Modbus device.
# To support this, place an “i0” in the Input/Output column in the spec file. Normally, only an “i” or an “o” would be placed in this column.
#
# Reference manufacturer’s documentation for the device:
#    i  -> Read from Input Register table 3:0000
#    o  -> Write to Hold Register table 4:0000
#    i0 -> Read from Hold Register table 4:0000
#
# The exceptions are discretes (type b); inputs (i) are -> table 1:0000
# and outputs (o) -> table 0:0000 and read output coils (i0) -> table 0:0000
```

# CyFlex Variable Name	CyFlex Units PAM Unit	Modbus Units PAM Unit	Input/Output i,i0,	Update Rate (ms) 2-1000	Slave Index 0-n	Mbus Address 1-255	Register	Type b,s,l,f	Scale Slope	Scale Offset
spd1	rpm	rpm	I	100	0	1	1	l	0.1	0.0
spd2	rpm	rpm	I	100	0	1	2	l	0.000001	0.0
spd3	rpm	rpm	I	100	0	1	3	l	0.1	0.0
spd4	rpm	rpm	I	100	0	1	4	l	0.1	0.0

4.2 Modifying the Specification File

4.2.1 General Steps

1. At the test cell computer, open the device specification file with a text editor Refer to *Section 2.3 Driver and Specification Files* on page 2 for the file location.
2. Determine which information in the file needs updating.

Note:

Users typically choose to copy the specification file from another computer performing a similar role to save time editing the file manually. Otherwise, this information can be determined as described in *Table 5*.

Table 5: Specification File Content Descriptions

Spec File Element	Information Source Method
Server (slave) IP	The manufacturer's documentation should explain how to identify the device's IP address. Normally, the technician or engineer who set up the test cell equipment has this information.
Command port	The command port should be specified in the device manufacturer's software documentation. The default is port 502.

Local IP	The local IP is not needed if the test cell can ping the server (slave) IP.
	From the test cell/Linux computer:
	1. Open a terminal window.
	2. Enter:
	`hostname`
	This displays the hostname of the test cell computer.
	3. Enter:
	`nslookup <hostname>`
	Example: `nslookup CTC-TC1`
	This displays the IP address for the test cell (in this case, test cell 1).

CyFlex variables	Consult the engineer running the test for the variables to monitor.
Mbus address	Typically, the technician or engineer who set up the test cell equipment can determine these.
Input/Output	
Update rate	
Slave Index	
Type	
Scale slope	
Scale offset	

3. Update the specification file as needed.
4. Save and exit.

4.2.2 Mapping to a Specification File

Mapping to the registers of a particular device depends on the manufacturer's register map for that device as previously mentioned.

Example:

For this example, reference the `Input/Output` and `Register` columns of the example specification file shown on page 7. The register column identifies a specific register of the device. The I/O column identifies the signal as follows:

Table 6: I/O Column Signal Identification

Input/Output	Description
`i`	Read from input register
`o`	Write holding register
`io`	Read holding register

Enter the following to have the specification file read device input register 30009 for instance:

- `i` for the I/O
- ” for the register

The actual Modbus messages would be similar to the following:

```
Master request to read input register:
11 04 0008 0001 B298
11:   Slave Address (17 = 11 hex)
04:   Function Code (read analog input registers)
0008: Data Address of the first register requested (30009-30001 = 8)
0001: Total number of registers requested (read 1 register)
B298: CRC (cyclic redundancy check) for error checking
Note:  In this example, the slave device has address 17.
Reply from device:
11 04 02 000A F8F4
11:   Slave Address (17 = 11 hex)
04:   Function Code (read Analog Input Registers)
02:   Number of data bytes to follow (1 registers x 2 bytes each = 2
bytes)
000A: Contents of register 30009
F8F4: CRC (cyclic redundancy check)
```

5 Device Driver Commands

The device driver is general purpose and supports the Modbus TCP protocol and RS-485.

Although the driver file cannot be modified, it can be manipulated with the `gmodbus` command. Refer to cyflex.com usage help for `gmodbus`.

6 Installing the Files

The GModBus device driver and specification files are located in the directories mentioned in *Section 2.3 Driver and Specification Files* on page 2.

The driver and specification files can be added to a Go-script, which launches these files when the computer starts. CyFlex test cells already in service typically have a Go-script, saved in the `/cyflex/cell/` directory.

If building a CyFlex system to include support for Modbus, consider copying the Go-script from another test cell performing identical functions, or contact TRP Laboratories to check the configuration or provide a default one.

Execute the following steps to run the GModBus driver and specification file from the Go-script:

1. At the test cell computer, open the Go-script with a text editor.
2. Add the following line:

   `gmodbus priority /specs/gmodbus_specs.`<tc_number>

   This adds both the driver and the spec file.

   Notes:

   The GModBus application uses CyFlex variables listed in its specification file. However, if the variables were included in the gen_labels spec file and the gen_labels application is listed in the Go-script file ahead of GModBus, the gen_labels spec file determines those variables. If a variable is not defined in a spec file, Modbus automatically creates it.

   Replace <tc_number> with your test cell number.

3. Save the `go.scp` file and exit.

7 Performance Tuning

Modbus (RTU or TCP) is a master/slave communications protocol, with the master sending a command and the slave replying with a response. There should be no un-requested communications from the slave to the master.

GModbus is a Modbus master, sending a request to the slave and waiting (with a timeout) for a response from the slave. GModbus starts a “child” task (gmodbus_child) for each interface (serial device or IP address). This keeps slower-responding devices from affecting the performance of faster-responding devices on the connection.

However, if there are multiple devices on a multi-drop serial connection, the slower devices affect the communication speed of all devices on the connection.

7.1 Checking Performance

The factor most often slowing a slave’s communication is its speed in responding to requests from the master. If a slave is suspected of not keeping up with the requested transfer rate, try using a utility to:

- Examine logs for a “not keeping up” error
- View timestamps between requests and responses

Recommended utilities:

- For Modbus TCP:
  - `Tcpdump`
  - `Wireshark`
- For Modbus RTU: `strace`

7.1.1 Serial Connection

For a serial connection, the Linux strace command can display communications between the master (GModBus driver) and slave devices.

Execute the following steps to view whether a slave is keeping up with the requested transfer rate (master):

1. Determine the process ID of the specification file for the slave, by entering at the command line:

   ```
   $ sin -P gmodbus
   ```

   The output displays processes for the GModBus driver, including the slaves’ process IDs.
2. Once the process ID of the slave’s specification file is known, enter the command below and watch for that ID in the output. The `–tt` argument shows timestamps followed by microseconds.

   ```
   $ strace –tt –p <process ID of spec file for the slave>
   ```

   Note:

   The command `strace -tt` shows timestamps for all of the system calls.

7.1.2 TCP Connection

For a TCP connection, use either of the following packet analyzer utilities:

1. Go to the URL:
   Tcpdump: http://www.tcpdump.org/
   or
   Wireshark: https://www.wireshark.org/
2. Refer to the online documentation to use the respective utility.

7.2 Optimizing Transfer Performance

Suggestions for optimizing the performance of transfers:

- Move devices to separate interfaces:
  If there are multiple devices on a multi-drop serial connection, moving devices to separate interfaces can improve performance. This generally only applies to Modbus/RTU configurations, since there is usually more bandwidth available in a Modbus/TCP setup.
- Combine variables into a single transfer:
  Because combining variables into a single transfer usually improves performance, try adding placeholders (“dummy variables”) between registers that otherwise are not adjacent. GModbus attempts to combine variables into a single request when possible. Note however that “double precision” variables are not combined and each must be sent as a separate request.
  For variables to be combined, they must be:
  - For the same device
  - All input or all output
  - The same type (`b`, `d`, `f`, `l`, `s`, `ul`, `us`)

  And have:
  - Adjacent addresses
  - The same update event/interval

  Refer to *Section 4.1 Identifying Information in the File* on page 5.
- Reduce the update rate in the specification file. Refer to the example specification file on page 7.