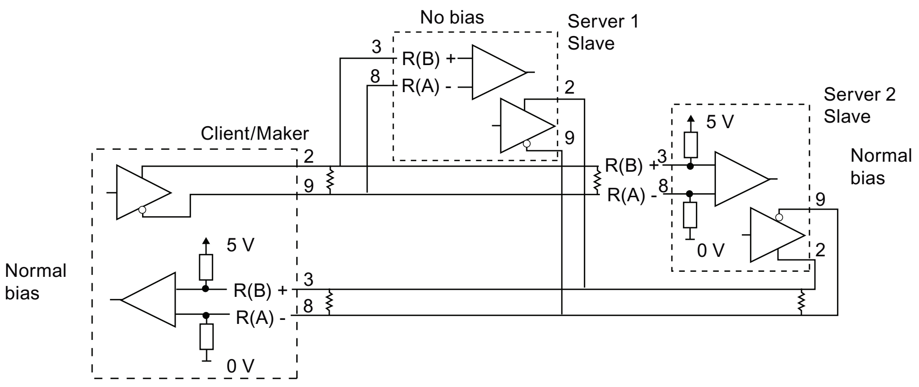
## Case 3: RS422: No cable break detection, no bias

- Mode of operation: RS422
- Receive line initial state: no bias
- Cable break: No cable break detection (transmitter enabled only while transmitting)

Bias and termination are added by the user at the end nodes of the network.



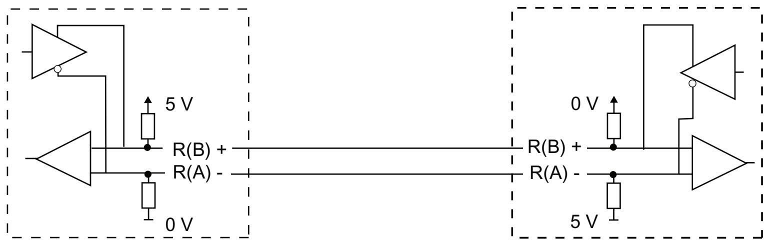
## Configuring the RS485

For RS485 mode, there is only one operating mode. The different selections for Receive line initial state reference the cases shown below for more details.

- Half duplex (RS485) two wire mode. In the Receive line initial state:
  - Select none when you supply the bias and termination (Case 5).
  - Select forward bias to use internal bias and termination (Case 4).

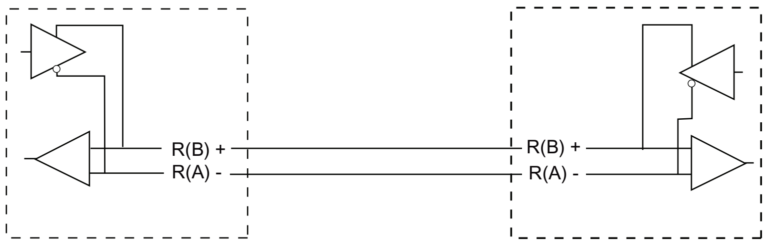
## Case 4: RS485: Forward bias

- Mode of operation: RS485
- Receive line initial state: Forward bias (biased with R(B) > R(A) > 0V)



### Case 5: RS485: No bias (external bias)

- Mode of operation: RS485
- Receive line initial state: No bias (external bias required)

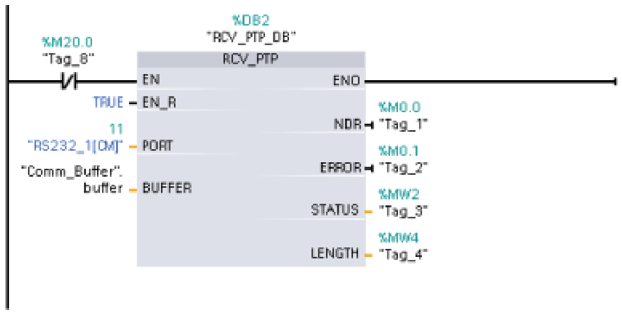


## 12.3.5.3 Programming the STEP 7 program

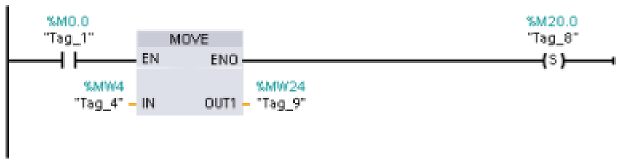
The example program uses a global data block for the communication buffer, a RCV_PTP instruction (Page 578) to receive data from the terminal emulator, and a SEND_PTP instruction (Page 575) to echo the buffer back to the terminal emulator. To program the example, add the data block configuration and program OB1 as described below.

**Global data block "Comm_Buffer":** Create a global data block (DB) and name it "Comm_Buffer". Create one value in the data block called "buffer" with a data type of "array [0 .. 99] of byte".

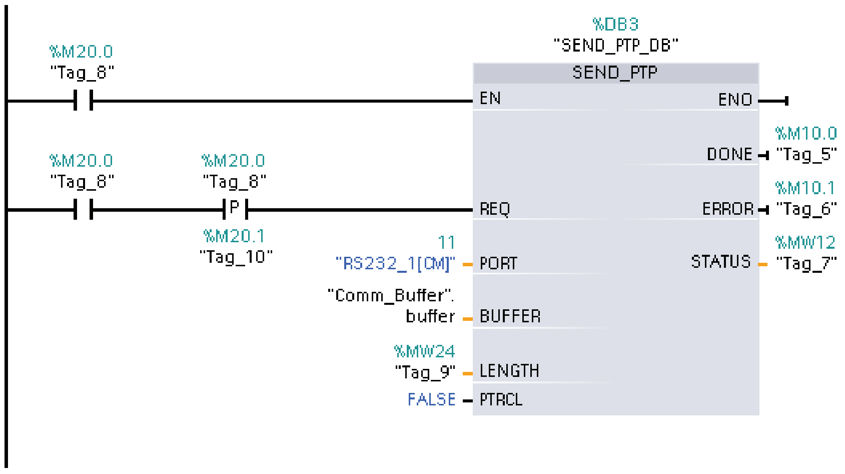
**Network 1:** Enable the RCV_PTP instruction whenever SEND_PTP is not active. Tag_8 at MW20.0 indicates when sending is complete in Network 4, and when the communication module is thus ready to receive a message.



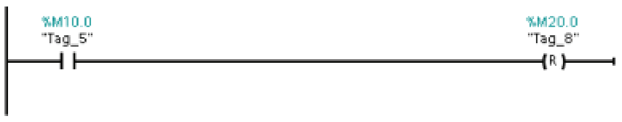
**Network 2:** Use the NDR value (Tag_1 at M0.0) set by the RCV_PTP instruction to make a copy of the number of bytes received and to set a flag (Tag_8 at M20.0) to trigger the SEND_PTP instruction.



**Network 3:** Enable the SEND_PTP instruction when the M20.0 flag is set. Also use this flag to set the REQ input to TRUE for one scan. The REQ input tells the SEND_PTP instruction that a new request is to be transmitted. The REQ input must only be set to TRUE for one execution of SEND_PTP. The SEND_PTP instruction is executed every scan until the transmit completes. The transmit is complete when the last byte of the message has been transmitted from the CM 1241. When the transmit is complete, the DONE output (Tag_5 at M10.0) is set TRUE for one execution of SEND_PTP.



**Network 4:** monitor the DONE output of SEND_PTP and reset the transmit flag (Tag_8 at M20.0) when the transmit operation is complete. When the transmit flag is reset, the RCV_PTP instruction in Network 1 is enabled to receive the next message.



### 12.3.5.4 Configuring the terminal emulator

You must set up the terminal emulator to support the example program. You can use most any terminal emulator on your PC, such as HyperTerminal. Make sure that the terminal emulator is in the disconnected mode before editing the settings as follows:

1. Set the terminal emulator to use the RS232 port on the PC (normally COM1).
2. Configure the port for 9600 baud, 8 data bits, no parity (none), 1 stop bit and no flow control.
3. Change the settings of the terminal emulator to emulate an ANSI terminal.
4. Configure the terminal emulator ASCII setup to send a line feed after every line (after the user presses the Enter key).
5. Echo the characters locally so that the terminal emulator displays what is typed.

### 12.3.5.5 Running the example program

To exercise the example program, follow these steps:

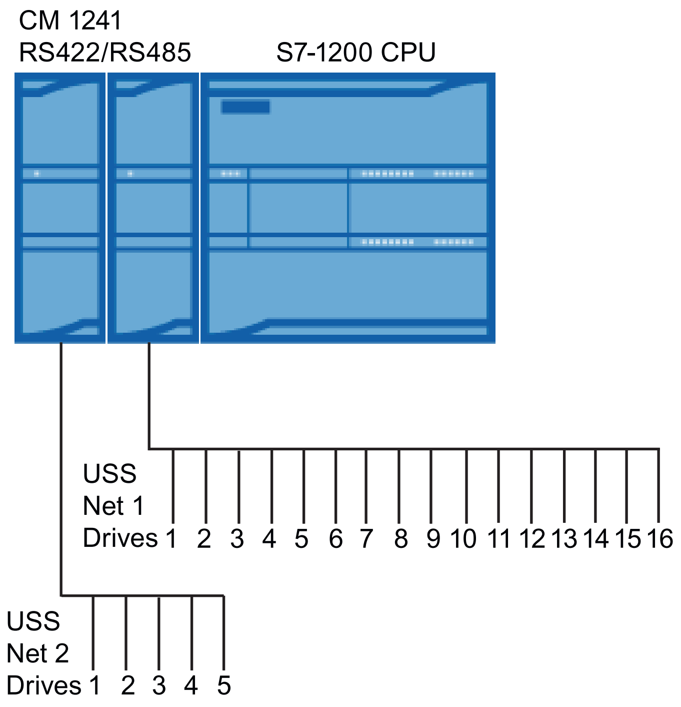
1. Download the STEP 7 program to the CPU and ensure that it is in RUN mode.
2. Click the "connect" button on the terminal emulator to apply the configuration changes and open a terminal session to the CM 1241.
3. Type characters at the PC and press Enter.

The terminal emulator sends the characters to the CM 1241 and to the CPU. The CPU program then echoes the characters back to the terminal emulator.

## 12.4 Universal serial interface (USS) communication

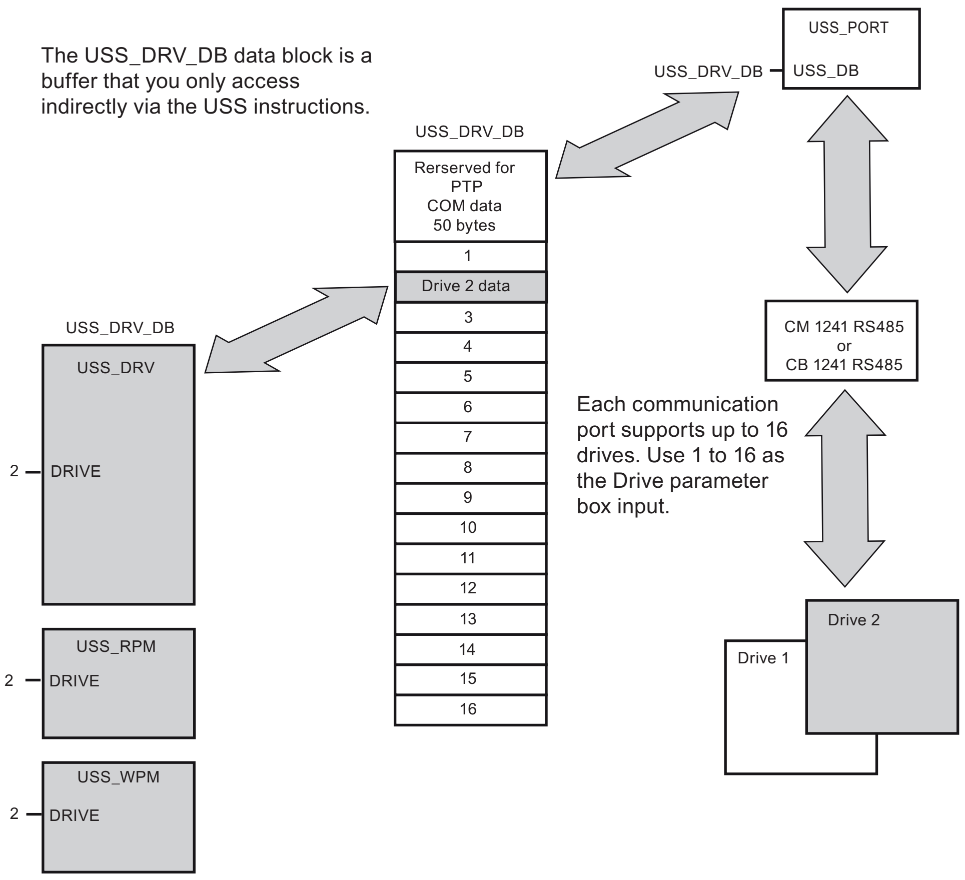
The USS instructions control the operation of motor drives which support the universal serial interface (USS) protocol. You can use the USS instructions to communicate with multiple drives through RS485 connections to CM 1241 RS485 communication modules or a CB 1241 RS485 communication board. Up to three CM 1241 RS422/RS485 modules and one CB 1241 RS485 board can be installed in a S7-1200 CPU. Each RS485 port can operate up to sixteen drives.

The USS protocol uses a master-slave network for communications over a serial bus. The master uses an address parameter to send a message to a selected slave. A slave itself can never transmit without first receiving a request to do so. Direct message transfer between the individual slaves is not possible. USS communication operates in half-duplex mode. The following USS illustration shows a network diagram for an example drive application.



### 12.4.1 Requirements for using the USS protocol

The four USS instructions use 1 FB and 3 FCs to support the USS protocol. One USS_PORT instance data block (DB) is used for each USS network. The USS_PORT instance data block contains temporary storage and buffers for all drives on that USS network. The USS instructions share the information in this data block.


The USS_DRV_DB data block is a buffer that you only access indirectly via the USS instructions.
USS_DRV_DB
USS_PORT
USS_DB
USS_DRV_DB
Rerserved for PTP COM data 50 bytes
1
Drive 2 data
3
4
5
6
7
8
9
10
11
12
13
14
15
16
USS_DRV_DB
USS_DRV
2
DRIVE
USS_RPM
2
DRIVE
USS_WPM
2
DRIVE
CM 1241 RS485 or CB 1241 RS485
Each communication port supports up to 16 drives. Use 1 to 16 as the Drive parameter box input.
Drive 2
Drive 1

All drives (up to 16) connected to a single RS485 port are part of the same USS network. All drives connected to a different RS485 port are part of a different USS network. Each USS network is managed using a unique data block. All instructions associated with a single USS network must share this data block. This includes all USS_DRV, USS_PORT, USS_RPM, and USS_WPM instructions used to control all drives on a single USS network.

The USS_DRV instruction is a Function Block (FB). When you place the USS_DRV instruction into the program editor, you will be prompted by the "Call options" dialog to assign a DB for this FB. If this is the first USS_DRV instruction in this program for this USS network, then you can accept the default DB assignment (or change the name if you wish) and the new DB is created for you. If however this is not the first USS_DRV instruction for this channel, then you must use the drop-down list in the "Call options" dialog to select the DB name that was previously assigned for this USS network.

Instructions USS_PORT, USS_RPM, and USS_WPM are all Functions (FCs). No DB is assigned when you place these FCs in the editor. Instead, you must assign the appropriate DB reference to the "USS_DB" input of these instructions. Double-click on the parameter field and then click on the parameter helper icon to see the available DB names).

The USS_PORT function handles the actual communication between the CPU and the drives via the Point-to-Point (PtP) RS485 communication port. Each call to this function handles one communication with one drive. Your program must call this function fast enough to prevent a communication timeout by the drives. You may call this function in a main program cycle OB or any interrupt OB.

Typically, you should call the USS_PORT function from a cyclic interrupt OB. The cycle time of the cyclic interrupt OB should be set to about half of the minimum call interval (As an example, 1200 baud communication should use a cyclic time of 350 ms or less).

The USS_DRV function block provides your program access to a specified drive on the USS network. Its inputs and outputs are the status and controls for the drive. If there are 16 drives on the network, your program must have at least 16 USS_DRV calls, one for each drive. These blocks should be called at the rate that is required to control the operation of the drive.

You may only call the USS_DRV function block from a main program cycle OB.

⚠ CAUTION

Only call USS_DRV, USS_RPM, and USS_WPM from a main program cycle OB. The USS_PORT function can be called from any OB, usually from a cyclic interrupt OB.

Do not use instructions USS_DRV, USS_RPM, or USS_WPM in a higher priority OB than the corresponding USS_PORT instruction. For example, do not place the USS_PORT in the main and a USS_RPM in a cyclic interrupt OB. Failure to prevent interruption of USS_PORT execution may produce unexpected errors.

The USS_RPM and USS_WPM functions read and write the remote drive operating parameters. These parameters control the internal operation of the drive. See the drive manual for the definition of these parameters. Your program can contain as many of these functions as necessary, but only one read or write request can be active per drive, at any given time. You may only call the USS_RPM and USS_WPM functions from a main program cycle OB.

## Calculating the time required for communicating with the drive

Communications with the drive are asynchronous to the S7-1200 scan cycle. The S7-1200 typically completes several scans before one drive communications transaction is completed.

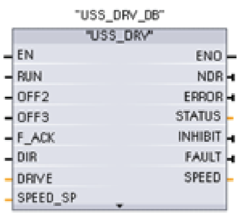
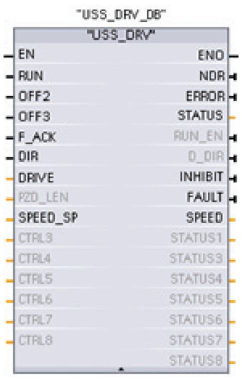
The USS_PORT interval is the time required for one drive transaction. The table below shows the minimum USS_PORT interval for each communication baud rate. Calling the USS_PORT function more frequently than the USS_PORT interval will not increase the number of transactions. The drive timeout interval is the amount of time that might be taken for a transaction, if communications errors caused 3 tries to complete the transaction. By default, the USS protocol library automatically does up to 2 retries on each transaction.

Table 12- 34 Calculating the time requirements

| Baud rate | Calculated minimum USS_PORT call Interval ( milliseconds ) | Drive message interval timeout per drive ( milliseconds ) |
|---|---|---|
| 1200 | 790 | 2370 |
| 2400 | 405 | 1215 |
| 4800 | 212.5 | 638 |
| 9600 | 116.3 | 349 |
| 19200 | 68.2 | 205 |
| 38400 | 44.1 | 133 |
| 57600 | 36.1 | 109 |
| 115200 | 28.1 | 85 |

## 12.4.2 USS_DRV instruction

Table 12- 35 USS_DRV instruction

| LAD / FBD | SCL | Description |
|---|---|---|
| Default view<br> "USS_DRV_DB"<br>"USS_DRV"<br>EN<br>RUN<br>OFF2<br>OFF3<br>F_ACK<br>DIR<br>DRIVE<br>SPEED_SP<br>ENO<br>NDR<br>ERROR<br>STATUS<br>INHIBIT<br>FAULT<br>SPEED<br>Expanded view<br> "USS_DRV_DB"<br>"USS_DRV"<br>EN<br>RUN<br>OFF2<br>OFF3<br>F_ACK<br>DIR<br>DRIVE<br>PZD_LEN<br>SPEED_SP<br>CTRL3<br>CTRL4<br>CTRL5<br>CTRL6<br>CTRL7<br>CTRL8<br>ENO<br>NDR<br>ERROR<br>STATUS<br>RUN_EN<br>D_DIR<br>INHIBIT<br>FAULT<br>SPEED<br>STATUS1<br>STATUS3<br>STATUS4<br>STATUS5<br>STATUS6<br>STATUS7<br>STATUS8 | `"USS_DRV_DB"(`<br>`    RUN:=_bool_in_,`<br>`    OFF2:=_bool_in_,`<br>`    OFF3:=_bool_in_,`<br>`    F_ACK:=_bool_in_,`<br>`    DIR:=_bool_in_,`<br>`    DRIVE:=_usint_in_,`<br>`    PZD_LEN:=_usint_in_,`<br>`    SPEED_SP:=_real_in_,`<br>`    CTRL3:=_word_in_,`<br>`    CTRL4:=_word_in_,`<br>`    CTRL5:=_word_in_,`<br>`    CTRL6:=_word_in_,`<br>`    CTRL7:=_word_in_,`<br>`    CTRL8:=_word_in_,`<br>`    NDR=>_bool_out_,`<br>`    ERROR=>_bool_out_,`<br>`    STATUS=>_word_out_,`<br>`    RUN_EN=>_bool_out_,`<br>`    D_DIR=>_bool_out_,`<br>`    INHIBIT=>_bool_out_,`<br>`    FAULT=>_bool_out_,`<br>`    SPEED=>_real_out_,`<br>`    STATUS1=>_word_out_,`<br>`    STATUS3=>_word_out_,`<br>`    STATUS4=>_word_out_,`<br>`    STATUS5=>_word_out_,`<br>`    STATUS6=>_word_out_,`<br>`    STATUS7=>_word_out_,`<br>`    STATUS8=>_word_out_);` | The USS_DRV instruction exchanges data with a drive by creating request messages and interpreting the drive response messages. A separate function block should be used for each drive, but all USS functions associated with one USS network and PtP communication port must use the same instance data block. You must create the DB name when you place the first USS_DRV instruction and then reference the DB that was created by the initial instruction usage.<br>STEP 7 automatically creates the DB when you insert the instruction. |

1 LAD and FBD: Expand the box to reveal all the parameters by clicking the bottom of the box. The parameter pins that are grayed are optional and parameter assignment is not required.

Table 12- 36 Data types for the parameters

| Parameter and type | | Data type | Description |
|---|---|---|---|
| RUN | IN | Bool | Drive start bit: When true, this input enables the drive to run at the preset speed. When RUN goes to false while a drive is running, the motor will be ramped down to a stop. This behavior differs from the dropping power (OFF2) or braking the motor (OFF3). |
| OFF2 | IN | Bool | Electrical stop bit: When false, this bit cause the drive to coast to a stop with no braking. |
| OFF3 | IN | Bool | Fast stop bit: When false, this bit causes a fast stop by braking the drive rather than just allowing the drive to coast to a stop. |

| Parameter and type | | Data type | Description |
|---|---|---|---|
| F_ACK | IN | Bool | Fault acknowledge bit: This bit is set to reset the fault bit on a drive. The bit is set after the fault is cleared to indicate to the drive it no longer needs to indicate the previous fault. |
| DIR | IN | Bool | Drive direction control: This bit is set to indicate that the direction is forward (for positive SPEED_SP). |
| DRIVE | IN | USInt | Drive address: This input is the address of the USS drive. The valid range is drive 1 to drive 16. |
| PZD_LEN | IN | USInt | Word length: This is the number of words of PZD data. The valid values are 2, 4, 6, or 8 words. The default value is 2. |
| SPEED_SP | IN | Real | Speed set point: This is the speed of the drive as a percentage of configured frequency. A positive value specifies forward direction (when DIR is true). Valid range is 200.00 to -200.00. |
| CTRL3 | IN | Word | Control word 3: A value written to a user-configurable parameter on the drive. You must configure this on the drive. (optional parameter) |
| CTRL4 | IN | Word | Control word 4: A value written to a user-configurable parameter on the drive. You must configure this on the drive. (optional parameter) |
| CTRL5 | IN | Word | Control word 5: A value written to a user-configurable parameter on the drive. You must configure this on the drive. (optional parameter) |
| CTRL6 | IN | Word | Control word 6: A value written to a user-configurable parameter on the drive. You must configure this on the drive. (optional parameter) |
| CTRL7 | IN | Word | Control word 7: A value written to a user-configurable parameter on the drive. You must configure this on the drive. (optional parameter) |
| CTRL8 | IN | Word | Control word 8: A value written to a user-configurable parameter on the drive. You must configure this on the drive. (optional parameter) |
| NDR | OUT | Bool | New data ready: When true, the bit indicates that the outputs contain data from a new communication request. |
| ERROR | OUT | Bool | Error occurred: When true, this indicates that an error has occurred and the STATUS output is valid. All other outputs are set to zero on an error. Communication errors are only reported on the USS_PORT instruction ERROR and STATUS outputs. |
| STATUS | OUT | Word | The status value of the request indicates the result of the scan. This is not a status word returned from the drive. |
| RUN_EN | OUT | Bool | Run enabled: This bit indicates whether the drive is running. |
| D_DIR | OUT | Bool | Drive direction: This bit indicates whether the drive is running forward. |
| INHIBIT | OUT | Bool | Drive inhibited: This bit indicates the state of the inhibit bit on the drive. |
| FAULT | OUT | Bool | Drive fault: This bit indicates that the drive has registered a fault. You must fix the problem and then set the F_ACK bit to clear this bit when set. |
| SPEED | OUT | Real | Drive Current Speed (scaled value of drive status word 2): The value of the speed of the drive as a percentage of configured speed. |
| STATUS1 | OUT | Word | Drive Status Word 1: This value contains fixed status bits of a drive. |
| STATUS3 | OUT | Word | Drive Status Word 3: This value contains a user-configurable status word on the drive. |
| STATUS4 | OUT | Word | Drive Status Word 4: This value contains a user-configurable status word on the drive. |
| STATUS5 | OUT | Word | Drive Status Word 5: This value contains a user-configurable status word on the drive. |

| Parameter and type | | Data type | Description |
|---|---|---|---|
| STATUS6 | OUT | Word | Drive Status Word 6: This value contains a user-configurable status word on the drive. |
| STATUS7 | OUT | Word | Drive Status Word 7: This value contains a user-configurable status word on the drive. |
| STATUS8 | OUT | Word | Drive Status Word 8: This value contains a user-configurable status word on the drive. |

When the initial USS_DRV execution occurs, the drive indicated by the USS address (parameter DRIVE) is initialized in the Instance DB. After this initialization, subsequent executions of USS_PORT can begin communication to the drive at this drive number.

Changing the drive number requires a CPU STOP-to-RUN mode transition that initializes the instance DB. Input parameters are configured into the USS TX message buffer and outputs are read from a "previous" valid response buffer if any exists. There is no data transmission during USS_DRV execution. Drives communicate when USS_PORT is executed. USS_DRV only configures the messages to be sent and interprets data that might have been received from a previous request.

You can control the drive direction of rotation using either the DIR input (Bool) or using the sign (positive or negative) with the SPEED_SP input (Real). The following table indicates how these inputs work together to determine the drive direction, assuming the motor is wired for forward rotation.

Table 12- 37 Interaction of the SPEED_SP and DIR parameters

| SPEED_SP | DIR | Drive rotation direction |
|---|---|---|
| Value > 0 | 0 | Reverse |
| Value > 0 | 1 | Forward |
| Value < 0 | 0 | Forward |
| Value < 0 | 1 | Reverse |

### 12.4.3 USS_PORT instruction

Table 12- 38 USS_PORT instruction

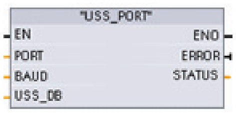
| LAD / FBD | SCL | Description |
|---|---|---|
|  "USS_PORT"<br>EN<br>ENO<br>PORT<br>ERROR<br>BAUD<br>STATUS<br>USS_DB | `USS_PORT(`<br>`    PORT:=_uint_in_,`<br>`    BAUD:=_dint_in_,`<br>`    ERROR=>_bool_out_,`<br>`    STATUS=>_word_out_,`<br>`    USS_DB:=_fbtref_inout_);` | The USS_PORT instruction handles communication over a USS network. |

Table 12- 39 Data types for the parameters

| Parameter and type | | Data type | Description |
|---|---|---|---|
| PORT | IN | Port | After you install and configure a CM or CB communication device, the port identifier appears in the parameter helper drop-list available at the PORT box connection. The assigned CM or CB port value is the device configuration property "hardware identifier". The port symbolic name is assigned in the "System constants" tab of the PLC tag table. |
| BAUD | IN | DInt | The baud rate used for USS communication. |
| USS_DB | INOUT | USS_BASE | The name of the instance DB that is created and initialized when a USS_DRV instruction is placed in your program. |
| ERROR | OUT | Bool | When true, this output indicates that an error has occurred and the STATUS output is valid. |
| STATUS | OUT | Word | The status value of the request indicates the result of the scan or initialization. Additional information is available in the "USS_Extended_Error" variable for some status codes. |

Typically, there is only one USS_PORT instruction per PtP communication port in the program, and each call of this function handles a transmission to or from a single drive. All USS functions associated with one USS network and PtP communication port must use the same instance DB.

Your program must execute the USS_PORT instruction often enough to prevent drive timeouts. USS_PORT is usually called from a cyclic interrupt OB to prevent drive timeouts and keep the most recent USS data updates available for USS_DRV calls.

## 12.4.4 USS_RPM instruction

Table 12- 40 USS_RPM instruction

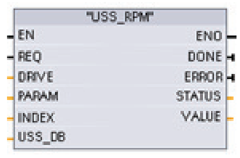
| LAD / FBD | SCL | Description |
|---|---|---|
|  "USS_RPM"<br>EN<br>ENO<br>REQ<br>DONE<br>DRIVE<br>ERROR<br>PARAM<br>STATUS<br>INDEX<br>VALUE<br>USS_DB | `USS_RPM(REQ:=_bool_in_,`<br>`DRIVE:=_usint_in_,`<br>`PARAM:=_uint_in_,`<br>`INDEX:=_uint_in_,`<br>`DONE=>_bool_out_,`<br>`ERROR=>_bool_out_,`<br>`STATUS=>_word_out_,`<br>`VALUE=>_variant_out_,`<br>`USS_DB:=_fbtref_inout_);` | The USS_RPM instruction reads a parameter from a drive. All USS functions associated with one USS network and PtP communication port must use the same data block. USS_RPM must be called from a main program cycle OB. |

Table 12- 41 Data types for the parameters

| Parameter type | | Data type | Description |
|---|---|---|---|
| REQ | IN | Bool | Send request: When true, REQ indicates that a new read request is desired. This is ignored if the request for this parameter is already pending. |
| DRIVE | IN | USInt | Drive address: DRIVE is the address of the USS drive. The valid range is drive 1 to drive 16. |

| Parameter type | | Data type | Description |
|---|---|---|---|
| PARAM | IN | UInt | Parameter number: PARAM designates which drive parameter is written. The range of this parameter is 0 to 2047. On some drives, the most significant byte can access PARAM values greater than 2047. See your drive manual for details on how to access an extended range. |
| INDEX | IN | UInt | Parameter index: INDEX designates which Drive Parameter index is to be written. A 16-bit value where the Least Significant Byte is the actual index value with a range of (0 to 255). The Most Significant Byte may also be used by the drive and is drive-specific. See your drive manual for details. |
| USS_DB | INOUT | USS_BASE | Then name of the instance DB that is created and initialized when a USS_DRV instruction is placed in your program. |
| VALUE | IN | Word, Int, UInt, DWord, DInt, UDInt, Real | This is the value of the parameter that was read and is valid only when the DONE bit is true. |
| DONE[1] | OUT | Bool | When true, indicates that the VALUE output holds the previously requested read parameter value. This bit is set when USS_DRV sees the read response data from the drive. This bit is reset when either: you request the response data via another USS_RPM poll, or on the second of the next two calls to USS_DRV |
| ERROR | OUT | Bool | Error occurred: When true, ERROR indicates that an error has occurred and the STATUS output is valid. All other outputs are set to zero on an error. Communication errors are only reported on the USS_PORT instruction ERROR and STATUS outputs. |
| STATUS | OUT | Word | STATUS indicates the result of the read request. Additional information is available in the "USS_Extended_Error" variable for some status codes. |

[1] The DONE bit indicates that valid data has been read from the referenced motor drive and delivered to the CPU. It does not indicate that the USS library is capable of immediately reading another parameter. A blank PKW request must be sent to the motor drive and must also be acknowledged by the instruction before the parameter channel for the specific drive becomes available for use. Immediately calling a USS_RPM or USS_WPM FC for the specified motor drive will result in a 0x818A error.

### 12.4.5 USS_WPM instruction

**Note**

**EEPROM write operations (for the EEPROM inside a USS drive)**

Do not overuse the EEPROM permanent write operation. Minimize the number of EEPROM write operations to extend the EEPROM life.

Table 12- 42 USS_WPM instruction

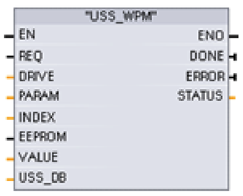
| LAD / FBD | SCL | Description |
|---|---|---|
|   "USS_WPM"<br>EN<br>REQ<br>DRIVE<br>PARAM<br>INDEX<br>EEPROM<br>VALUE<br>USS_DB<br>ENO<br>DONE<br>ERROR<br>STATUS | `USS_WPM(REQ:=_bool_in_,`<br>`DRIVE:=_usint_in_,`<br>`PARAM:=_uint_in_,`<br>`INDEX:=_uint_in_,`<br>`EEPROM:=_bool_in_,`<br>`VALUE:=_variant_in_,`<br>`DONE=>_bool_out_,`<br>`ERROR=>_bool_out_,`<br>`STATUS=>_word_out_,`<br>`USS_DB:=_fbtref_inout_);` | The USS_WPM instruction modifies a parameter in the drive. All USS functions associated with one USS network and PtP communication port must use the same data block.<br>USS_WPM must be called from a main program cycle OB. |

Table 12- 43 Data types for the parameters

| Parameter and type | | Data type | Description |
|---|---|---|---|
| REQ | IN | Bool | Send request: When true, REQ indicates that a new write request is desired. This is ignored if the request for this parameter is already pending. |
| DRIVE | IN | USInt | Drive address: DRIVE is the address of the USS drive. The valid range is drive 1 to drive 16. |
| PARAM | IN | UInt | Parameter number: PARAM designates which drive parameter is written. The range of this parameter is 0 to 2047. On some drives, the most significant byte can access PARAM values greater than 2047. See your drive manual for details on how to access an extended range. |
| INDEX | IN | UInt | Parameter index: INDEX designates which Drive Parameter index is to be written. A 16-bit value where the least significant byte is the actual index value with a range of (0 to 255). The most significant byte may also be used by the drive and is drive-specific. See your drive manual for details. |
| EEPROM | IN | Bool | Store To Drive EEPROM: When true, a write drive parameter transaction will be stored in the drive EEPROM. If false, the write is temporary and will not be retained if the drive is power cycled. |
| VALUE | IN | Word, Int, UInt, DWord, DInt, UDInt, Real | The value of the parameter that is to be written. It must be valid on the transition of REQ. |
| USS_DB | INOUT | USS_BASE | The name of the instance DB that is created and initialized when a USS_DRV instruction is placed in your program. |
| DONE[1] | OUT | Bool | When true, DONE indicates that the input VALUE has been written to the drive. This bit is set when USS_DRV sees the write response data from the drive. This bit is reset when either you request the response data via another USS_WPM poll, or on the second of the next two calls to USS_DRV |

| Parameter and type | | Data type | Description |
|---|---|---|---|
| ERROR | OUT | Bool | When true, ERROR indicates that an error has occurred and the STATUS output is valid. All other outputs are set to zero on an error. Communication errors are only reported on the USS_PORT instruction ERROR and STATUS outputs. |
| STATUS | OUT | Word | STATUS indicates the result of the write request. Additional information is available in the "USS_Extended_Error" variable for some status codes. |

[1] The DONE bit indicates that valid data has been read from the referenced motor drive and delivered to the CPU. It does not indicate that the USS library is capable of immediately reading another parameter. A blank PKW request must be sent to the motor drive and must also be acknowledged by the instruction before the parameter channel for the specific drive becomes available for use. Immediately calling a USS_RPM or USS_WPM FC for the specified motor drive will result in a 0x818A error.

## 12.4.6 USS status codes

USS instruction status codes are returned at the STATUS output of the USS functions.

Table 12- 44 STATUS codes [1]

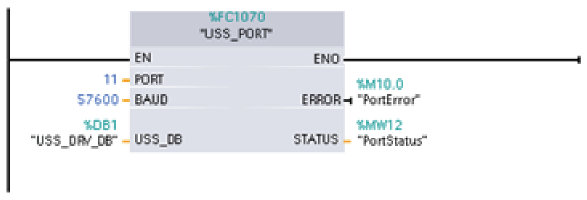
| STATUS (W#16#....) | Description |
|---|---|
| 0000 | No error |
| 8180 | The length of the drive response did not match the characters received from the drive. The drive number where the error occurred is returned in the "USS_Extended_Error" variable. See the extended error description below this table. |
| 8181 | VALUE parameter was not a Word, Real or DWord data type. |
| 8182 | The user supplied a Word for a parameter value and received a DWord or Real from the drive in the response. |
| 8183 | The user supplied a DWord or Real for a parameter value and received a Word from the drive in the response. |
| 8184 | The response telegram from drive had a bad checksum. The drive number where the error occurred is returned in the "USS_Extended_Error" variable. See the extended error description below this table. |
| 8185 | Illegal drive address (valid drive address range: 1 to16) |
| 8186 | The speed set point is out of the valid range (valid speed SP range: -200% to 200%). |
| 8187 | The wrong drive number responded to the request sent. The drive number where the error occurred is returned in the "USS_Extended_Error" variable. See the extended error description below this table. |
| 8188 | Illegal PZD word length specified (valid range = 2, 4, 6 or 8 words) |
| 8189 | Illegal Baud Rate was specified. |
| 818A | The parameter request channel is in use by another request for this drive. |
| 818B | The drive has not responded to requests and retries. The drive number where the error occurred is returned in the "USS_Extended_Error" variable. See the extended error description below this table. |
| 818C | The drive returned an extended error on a parameter request operation. See the extended error description below this table. |
| 818D | The drive returned an illegal access error on a parameter request operation. See your drive manual for information of why parameter access may be limited. |

| STATUS (W#16#....) | Description |
|---|---|
| 818E | The drive has not been initialized. This error code is returned to USS_RPM or USS_WPM when USS_DRV, for that drive, has not been called at least once. This keeps the initialization on first scan of USS_DRV from overwriting a pending parameter read or write request, since it initializes the drive as a new entry. To fix this error, call USS_DRV for this drive number. |
| 80Ax-80Fx | Specific errors returned from PtP communication FBs called by the USS Library - These error code values are not modified by the USS library and are defined in the PtP instruction descriptions. |

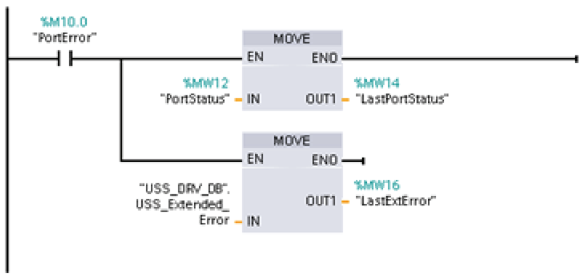
[1] In addition to the USS instruction errors listed above, errors can be returned from the underlying PtP communication instructions (Page 566).

For several STATUS codes, additional information is provided in the "USS_Extended_Error" variable of the USS_DRV Instance DB. For STATUS codes hexadecimal 8180, 8184, 8187, and 818B, USS_Extended_Error contains the drive number where the communication error occurred. For STATUS code hexadecimal 818C, USS_Extended_Error contains a drive error code returned from the drive when using a USS_RPM or USS_WPM instruction.

Communication errors (STATUS = 16#818B) are only reported on the USS_PORT instruction and not on the USS_DRV instruction. For example, if the network is not properly terminated then it is possible for a drive to go to RUN but the USS_DRV instruction will show all 0's for the output parameters. In this case, you can only detect the communication error on the USS_PORT instruction. Since this error is only visible for one scan, you will need to add some capture logic as illustrated in the following example. In this example, when the error bit of the USS_PORT instruction is TRUE, then the STATUS and the USS_Extended_Error values are saved into M memory. The drive number is placed in USS_Extended_Error variable when the STATUS code value is hexadecimal 8180, 8184, 8187, or 818B.



**Network 1** "PortStatus" port status and "USS_DRV_DB".USS_Extended_Error extended error code values are only valid for one program scan. The values must be captured for later processing.



**Network 2** The "PortError" contact triggers the storage of the "PortStatus" value in "LastPortStatus" and the "USS_DRV_DB".USS_Extended_Error value in "LastExtError".

USS drives support read and write access to a drive's internal parameters. This feature allows remote control and configuration of the drive. Drive parameter access operations can fail due to errors such as values out of range or illegal requests for a drive's current mode. The drive generates an error code value that is returned in the "USS_Extended_Error" variable. This error code value is only valid for the last execution of a USS_RPM or USS_WPM instruction. The drive error code is put into USS_Extended_Error variable when the STATUS code value is hexadecimal 818C. The error code value of "USS_Extended_Error" depends on the drive model. See the drive's manual for a description of the extended error codes for read and write parameter operations.

## 12.4.7 General drive setup information

### General drive setup requirements

- The drives must be set to use 4 PKW words.
- The drives can be configured for 2, 4, 6, or 8 PZD words.
- The number of PZD word's in the drive must match PZD_LEN input on the USS_DRV instruction for that drive.
- The baud rate in all the drives must match the BAUD input on the USS_PORT instruction.
- The drive must be set for remote control.
- The drive must be set for frequency set-point to USS on COM Link.
- The drive address must be set to 1 to 16 and match the DRIVE input on the USS_DRV block for that drive.
- The drive direction control must be set to use the polarity of the drive set-point.
- The RS485 network must be terminated properly.

### Connecting a MicroMaster drive

This information about SIEMENS MicroMaster drives is provided as an example. For other drives, refer to the drive's manual for setup instructions.

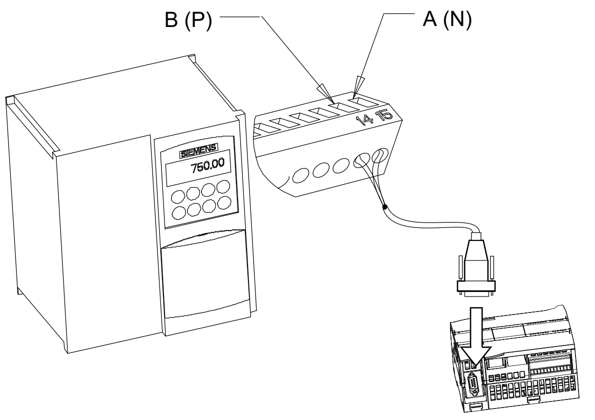
To make the connection to a MicroMaster Series 4 (MM4) drive, insert the ends of the RS485 cable into the two caged-clamp, screw-less terminals provided for USS operation. Standard PROFIBUS cable and connectors can be used to connect the S7-1200.

**CAUTION**

**Interconnecting equipment with different reference potentials can cause unwanted currents to flow through the interconnecting cable**

These unwanted currents can cause communications errors or damage equipment. Be sure all equipment that you are about to connect with a communications cable either shares a common circuit reference or is isolated to prevent unwanted current flows. The shield must be tied to chassis ground or pin 1 on the 9-pin connector. It is recommended that you tie wiring terminal 2--0V on the MicroMaster drive to chassis ground.

The two wires at the opposite end of the RS485 cable must be inserted into the MM4 drive terminal blocks. To make the cable connection on a MM4 drive, remove the drive cover(s) to access the terminal blocks. See the MM4 user manual for details about how to remove the covers(s) of your specific drive.

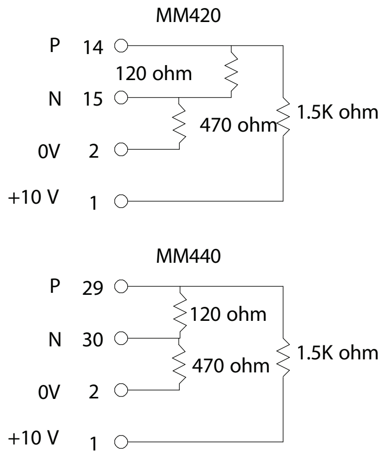


The terminal block connections are labeled numerically. Using a PROFIBUS connector on the S7-1200 side, connect the A terminal of the cable to the drive terminal 15 (for an MM420) or terminal 30 (MM440). Connect the B terminal of B (P) A (N) the cable connector to terminal 14 (MM420) or terminal 29 (MM440).

If the S7-1200 is a terminating node in the network, or if the connection is point-to-point, it is necessary to use terminals A1 and B1 (not A2 and B2) of the connector since they allow the termination settings to be set (for example, with DP connector type 6ES7 972--0BA40--0X40).

| CAUTION |
|---|
| Make sure the drive covers are replaced properly before supplying power to the unit. |

If the drive is configured as the terminating node in the network, then termination and bias resistors must also be wired to the appropriate terminal connections. This diagram shows examples of the MM4 drive connections necessary for termination and bias.



## Setting up the MM4 drive

Before you connect a drive to the S7-1200, you must ensure that the drive has the following system parameters. Use the keypad on the drive to set the parameters:

| | |
|---|---|
| 1. Reset the drive to factory settings (optional). | P0010=30<br>P0970=1 |
| If you skip step 1, then ensure that these parameters are set to the indicated values. | USS PZD length = P2012 Index 0=(2, 4, 6, or 8)<br>USS PKW length = P2013 Index 0=4 |
| 2. Enable the read/write access to all parameters (Expert mode). | P0003=3 |
| 3. Check the motor settings for your drive. The settings will vary according to the motor(s) being used.<br>To set the parameters P304, P305, P307, P310, and P311, you must first set parameter P010 to 1 (quick commissioning mode). When you are finished setting the parameters, set parameter P010 to 0. Parameters P304, P305, P307, P310, and P311 can only be changed in the quick commissioning mode. | P0304 = Rated motor voltage (V)<br>P0305 = Rated motor current (A)<br>P0307 = Rated motor power (W)<br>P03010 = Rated motor frequency (Hz)<br>P0311 = Rated motor speed |
| 4. Set the local/remote control mode. | P0700 Index 0=5 |
| 5. Set selection of frequency set-point to USS on COM link. | P1000 Index 0=5 |
| 6. Ramp up time (optional)<br>This is the time in seconds that it takes the motor to accelerate to maximum frequency. | P1120=(0 to 650.00) |
| 7. Ramp down time (optional)<br>This the time in seconds that it takes the motor to decelerate to a complete stop. | P1121=(0 to 650.00) |
| 8. Set the serial link reference frequency: | P2000=(1 to 650 Hz) |
| 9. Set the USS normalization: | P2009 Index 0=0 |
| 10. Set the baud rate of the RS485 serial interface: | P2010 Index 0= 4 (2400 baud)<br>5 (4800 baud)<br>6 (9600 baud)<br>7 (19200 baud<br>8 (38400 baud)<br>9 (57600 baud)<br>12 (115200 baud) |
| 11. Enter the Slave address.<br>Each drive (a maximum of 31) can be operated over the bus. | P2011 Index 0=(0 to 31) |
| 12. Set the serial link timeout.<br>This is the maximum permissible period between two incoming data telegrams. This feature is used to turn off the inverter in the event of a communications failure. Timing starts after a valid data telegram has been received. If a further data telegram is not received within the specified time period, the inverter will trip and display fault code F0070. Setting the value to zero switches off the control. | P2014 Index 0=(0 to 65,535 ms)<br>0=timeout disabled |
| 13. Transfer the data from RAM to EEPROM: | P0971=1 (Start transfer) Save the changes to the parameter settings to EEPROM |

## 12.5 Modbus communication

### 12.5.1 Overview of Modbus RTU and TCP communication

#### Modbus function codes

- A CPU operating as a Modbus RTU master (or Modbus TCP client) can read/write both data and I/O states in a remote Modbus RTU slave (or Modbus TCP server). Remote data can be read and processed in the user program.
- A CPU operating as a Modbus RTU slave (or Modbus TCP server) allows a supervisory device to read/write both data and I/O states in a remote CPU. The supervisor device can write new values in remote CPU memory that can be processed in the user program.

Table 12- 45 Read data functions: Read remote I/O and program data

| Modbus function code | Read slave (server) functions - standard addressing |
|---|---|
| 01 | Read output bits: 1 to 2000 bits per request |
| 02 | Read input bits: 1 to 2000 bits per request |
| 03 | Read Holding registers: 1 to 125 words per request |
| 04 | Read input words: 1 to 125 words per request |

Table 12- 46 Write data functions: Write remote I/O and modify program data

| Modbus function code | Write slave (server) functions - standard addressing |
|---|---|
| 05 | Write one output bit: 1 bit per request |
| 06 | Write one holding register: 1 word per request |
| 15 | Write one or more output bits: 1 to 1968 bits per request |
| 16 | Write one or more holding registers: 1 to 123 words per request |

- Modbus function codes 08 and 11 provide slave device communication diagnostic information.
- Modbus function code 0 broadcasts a message to all slaves (with no slave response). The broadcast function is not available for Modbus TCP, because communication is connection based.

Table 12- 47 Modbus network station addresses

| Station | | Address |
|---|---|---|
| RTU station | Standard station address | 1 to 247 |
| | Extended station address | 1 to 65535 |
| TCP station | Station address | IP address and port number |

## Modbus memory addresses

The actual number of Modbus memory addresses available depends on the CPU model, how much work memory exists, and how much CPU memory is used by other program data. The table below gives the nominal value of the address range.

Table 12- 48 Modbus memory addresses

| Station | | Address range |
|---|---|---|
| RTU station | Standard memory address | 10K |
| | Extended memory address | 64K |
| TCP station | Standard memory address | 10K |

## Modbus RTU communication

Modbus RTU (Remote Terminal Unit) is a standard network communication protocol that uses the RS232 or RS485 electrical connection for serial data transfer between Modbus network devices. You can add PtP (Point to Point) network ports to a CPU with a RS232 or RS485 CM or a RS485 CB.

Modbus RTU uses a master/slave network where all communications are initiated by a single Master device and slaves can only respond to a master's request. The master sends a request to one slave address and only that slave address responds to the command.

## Modbus TCP communication

Modbus TCP (Transmission Control Protocol) is a standard network communication protocol that uses the PROFINET connector on the CPU for TCP/IP communication. No additional communication hardware module is required.

Modbus TCP uses Open User Communications (OUC) connections as a Modbus communication path. Multiple client-server connections may exist, in addition to the connection between STEP 7 and the CPU. Mixed client and server connections are supported up to the maximum number of connections allowed by the CPU model (Page 424).

Each MB_SERVER connection must use a unique instance DB and IP port number. Only 1 connection per IP port is supported. Each MB_SERVER (with its unique instance DB and IP port) must be executed individually for each connection.

**Note**

Modbus TCP will only operate correctly with CPU firmware release V1.02 or later. An attempt to execute the Modbus instructions on an earlier firmware version will result in an error.

A Modbus TCP client (master) must control the client-server connection with the DISCONNECT parameter. The basic Modbus client actions are shown below.

1. Initiate a connection to a particular server (slave) IP address and IP port number
2. Initiate client transmission of a Modbus messages and receive the server responses
3. When desired, initiate the disconnection of client and server to enable connection with a different server.

## Modbus RTU instructions in your program

- MB_COMM_LOAD: One execution of MB_COMM_LOAD is used to set up PtP port parameters like baud rate, parity, and flow control. After a CPU port is configured for the Modbus RTU protocol, it can only be used by either the MB_MASTER or MB_SLAVE instructions.
- MB_MASTER: The Modbus master instruction enables the CPU to act as a Modbus RTU master device and communicate with one or more Modbus slave devices.
- MB_SLAVE: The Modbus slave instruction enables the CPU to act as a Modbus RTU slave device and communicate with a Modbus master device.

## Modbus TCP instructions in your program

- MB_CLIENT: Make client-server TCP connection, send command message, receive response, and control the disconnection from the server
- MB_SERVER: Connect to a Modbus TCP client upon request, receive Modbus message, and send response

## 12.5.2 Modbus TCP

### 12.5.2.1 MB_CLIENT (Modbus TCP)

Table 12- 49 MB_CLIENT instruction

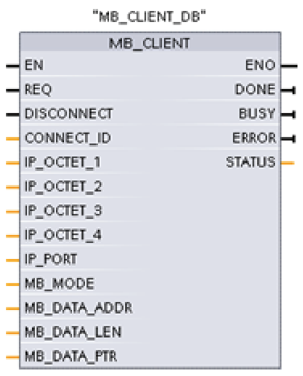
| LAD / FBD | SCL | Description |
|---|---|---|
|  "MB_CLIENT_DB"<br>MB_CLIENT<br>EN<br>REQ<br>DISCONNECT<br>CONNECT_ID<br>IP_OCTET_1<br>IP_OCTET_2<br>IP_OCTET_3<br>IP_OCTET_4<br>IP_PORT<br>MB_MODE<br>MB_DATA_ADDR<br>MB_DATA_LEN<br>MB_DATA_PTR<br>ENO<br>DONE<br>BUSY<br>ERROR<br>STATUS | `"MB_CLIENT_DB"(`<br>`    REQ:=_bool_in_,`<br>`    DISCONNECT:=_bool_in_,`<br>`    CONNECT_ID=_uint_in_,`<br>`    IP_OCTET_1:=_byte_in_,`<br>`    IP_OCTET_2:=_byte_in_,`<br>`    IP_OCTET_3:=_byte_in_,`<br>`    IP_OCTET_4:=_byte_in_,`<br>`    IP_PORT:=_uint_in_,`<br>`    MB_MODE:=_usint_in_,`<br>`    MB_DATA_ADDR:=_udint_in_,`<br>`    MB_DATA_LEN:=_uint_in_,`<br>`    DONE=>_bool_out_,`<br>`    BUSY=>_bool_out_,`<br>`    ERROR=>_bool_out_,`<br>`    STATUS=>_word_out_,`<br>`    MB_DATA_PTR:=_variant_inout_);` | MB_CLIENT communicates as a Modbus TCP client through the PROFINET connector on the S7-1200 CPU. No additional communication hardware module is required.<br>MB_CLIENT can make a client-server connection, send a Modbus function request, receive a response, and control the disconnection from a Modbus TCP server. |

Table 12- 50 Data types for the parameters

| Parameter and type | | Data type | Description |
|---|---|---|---|
| REQ | In | Bool | FALSE = No Modbus communication request<br>TRUE = Request to communicate with a Modbus TCP server |
| DISCONNECT | IN | Bool | The DISCONNECT parameter allows your program to control connection and disconnection with a Modbus server device.<br>If DISCONNECT = 0 and a connection does not exist, then MB_CLIENT attempts to make a connection to the assigned IP address and port number.<br>If DISCONNECT = 1 and a connection exists, then a disconnect operation is attempted. Whenever this input is enabled, no other operation will be attempted. |
| CONNECT_ID | IN | UInt | The CONNECT_ID parameter must uniquely identify each connection within the PLC. Each unique instance of the MB_CLIENT or MB_SERVER instruction must contain a unique CONNECT_ID parameter. |
| IP_OCTET_1 | IN | USInt | Modbus TCP server IP address: Octet 1<br>8-bit part of the 32-bit IPv4 IP address of the Modbus TCPserver to which the client will connect and communicate using the Modbus TCP protocol. |
| IP_OCTET_2 | IN | USInt | Modbus TCP server IP address: Octet 2 |
| IP_OCTET_3 | IN | USInt | Modbus TCP server IP address: Octet 3 |
| IP_OCTET_4 | IN | USInt | Modbus TCP server IP address: Octet 4 |
| IP_PORT | IN | UInt | Default value = 502: The IP port number of the server to which the client will attempt to connect and ultimately communicate using the TCP/IP protocol. |

| Parameter and type | | Data type | Description |
|---|---|---|---|
| MB_MODE | IN | USInt | Mode Selection: Assigns the type of request (read, write, or diagnostic). See the Modbus functions table below for details. |
| MB_DATA_ADDR | IN | UDInt | Modbus starting Address: Assigns the starting address of the data to be accessed by MB_CLIENT. See the Modbus functions table below for valid addresses. |
| MB_DATA_LEN | IN | UInt | Modbus data Length: Assigns the number of bits or words to be accessed in this request. See the Modbus functions table below for valid lengths |
| MB_DATA_PTR | IN_OUT | Variant | Pointer to the Modbus data register: The register buffers data going to or coming from a Modbus server. The pointer must assign a standard global DB or a M memory address. |
| DONE | OUT | Bool | The DONE bit is TRUE for one scan, after the last request was completed with no error. |
| BUSY | OUT | Bool | • 0 - No MB_CLIENT operation in progress<br>• 1 - MB_CLIENT operation in progress |
| ERROR | OUT | Bool | The ERROR bit is TRUE for one scan, after the MB_CLIENT execution was terminated with an error. The error code value at the STATUS parameter is valid only during the single cycle where ERROR = TRUE. |
| STATUS | OUT | Word | Execution condition code |

## REQ parameter

FALSE = No Modbus communication request
TRUE = Request to communicate with a Modbus TCP server

If no instance of MB_CLIENT is active and parameter DISCONNECT=0, when REQ=1 a new Modbus request will start. If the connection is not already established then a new connection will be made.

If the same instance of MB_CLIENT is executed again with DISCONNECT=0 and REQ=1, before the completion of the current request, then no subsequent Modbus transmission will be made. However, as soon as the current request is completed, a new request can be processed if MB_CLIENT is executed with REQ=1.

When the current MB_CLIENT communication request is complete, the DONE bit is TRUE for one cycle. The DONE bit can be used as a time gate to sequence multiple MB_CLIENT requests.

Note

**Input data consistency during MB_CLIENT processing**

Once a Modbus client initiates a Modbus operation, all the input states are saved internally and are then compared on each successive call. The comparison is used to determine if this particular call was the originator of the active client request. More than one MB_CLIENT call can be performed using a common instance DB.

As a result, it is important that the inputs are not changed during the period of time that a MB_CLIENT operation is actively being processed. If this rule is not followed, then a MB_CLIENT cannot determine that it is the active instance.

## MB_MODE and MB_DATA_ADDR parameters select the Modbus communication function

MB_DATA_ADDR assigns the starting Modbus address of the data to be accessed. The MB_CLIENT instruction uses a MB_MODE input rather than a function code input.

The combination of MB_MODE and MB_DATA_ADDR values determine the function code that is used in the actual Modbus message. The following table shows the correspondence between parameter MB_MODE, Modbus function, and Modbus address range.

Table 12- 51 Modbus functions

| MB_MODE | Modbus function | Data length | Operation and data | MB_DATA_ADDR |
|---|---|---|---|---|
| 0 | 01 | 1 to 2000 | Read output bits:<br>1 to 2000 bits per request | 1 to 9999 |
| 0 | 02 | 1 to 2000 | Read input bits:<br>1 to 2000 bits per request | 10001 to 19999 |
| 0 | 03 | 1 to 125 | Read Holding registers:<br>1 to 125 words per request | 40001 to 49999 or<br>400001 to 465535 |
| 0 | 04 | 1 to 125 | Read input words:<br>1 to 125 words per request | 30001 to 39999 |
| 1 | 05 | 1 | Write one output bit:<br>One bit per request | 1 to 9999 |
| 1 | 06 | 1 | Write one holding register:<br>1 word per request | 40001 to 49999 or<br>400001 to 465535 |
| 1 | 15 | 2 to 1968 | Write multiple output bits:<br>2 to 1968 bits per request | 1 to 9999 |
| 1 | 16 | 2 to 123 | Write multiple holding registers:<br>2 to 123 words per request | 40001 to 49999 or<br>400001 to 465535 |
| 2 | 15 | 1 to 1968 | Write one or more output bits:<br>1 to 1968 bits per request | 1 to 9999 |
| 2 | 16 | 1 to 123 | Write one or more holding registers:<br>1 to 123 words per request | 40001 to 49999 or<br>400001 to 465535 |
| 11 | 11 | 0 | Read the server communication status word and event counter. The status word indicates busy (0 – not busy, 0xFFFF - busy). The event counter is incremented for each successful completion of a message.<br>Both the MB_DATA_ADDR and MB_DATA_LEN parameters of MB_CLIENT are ignored for this function. | |
| 80 | 08 | 1 | Check server status using data diagnostic code 0x0000 (Loopback test – server echoes the request)<br>1 word per request | |

| MB_MODE | Modbus function | Data length | Operation and data | MB_DATA_ADDR |
|---|---|---|---|---|
| 81 | 08 | 1 | Reset server event counter using data diagnostic code 0x000A<br>1 word per request | |
| 3 to 10,<br>12 to 79,<br>82 to 255 | | | Reserved | |

**Note**

**MB_DATA_PTR assigns a buffer to store data read/written to/from a Modbus TCP server**

The data buffer can be in a standard global DB or M memory address.

For a buffer in M memory, use the standard Any Pointer format. This is in the format P#"Bit Address" "Data Type" "Length", an example would be P#M1000.0 WORD 500.

## MB_DATA_PTR assigns a communication buffer

- MB_CLIENT communication functions:
  - Read and write 1-bit data from Modbus server addresses (00001 to 09999)
  - Read 1-bit data from Modbus server addresses (10001 to 19999)
  - Read 16-bit word data from Modbus server addresses (30001 to 39999) and (40001 to 49999)
  - Write 16-bit word data to Modbus server addresses (40001 to 49999)
- Word or bit sized data is transferred to/from the DB or M memory buffer assigned by MB_DATA_PTR.
- If a DB is assigned as the buffer by MB_DATA_PTR, then you must assign data types to all DB data elements.
  - The 1-bit Bool data type represents one Modbus bit address
  - 16-bit single word data types like WORD, UInt, and Int represent one Modbus word address
  - 32-bit double word data types like DWORD, DInt, and Real represent two Modbus word addresses

- Complex DB elements can be assigned by MB_DATA_PTR, such as
  - Standard arrays
  - Named structures where each element is unique.
  - Named complex structures where each element has a unique name and a 16 or 32 bit data type.
- There is no requirement that the MB_DATA_PTR data areas be in the same global data block (or M memory area). You can assign one data block for Modbus reads, another data block for Modbus writes, or one data block for each MB_CLIENT station.

## Multiple client connections

A Modbus TCP client can support concurrent connections up to the maximum number of Open User Communications connections allowed by the PLC. The total number of connections for a PLC, including Modbus TCP Clients and Servers, must not exceed the maximum number of supported Open User Communications connections (Page 424). The Modbus TCP connections may be shared between Client and/or Server type connections.

Individual client connections must follow these rules:

- Each MB_CLIENT connection must use a distinct instance DB
- Each MB_CLIENT connection must specify a unique server IP address
- Each MB_CLIENT connection must specify a unique connection ID
- Unique IP port numbers may or may not be required depending upon the server configuration

The Connection ID must be unique for each individual connection. This means a single, unique Connection ID must only be used with each individual instance DB. In summary, the instance DB and the Connection ID are paired together and must be unique for every connection.

Table 12- 52 MB_CLIENT instance data block user accessible static variables

| Variable | Data type | Default | description |
|---|---|---|---|
| Blocked_Proc_Timeout | Real | 3.0 | Amount of time (in seconds) to wait upon a blocked Modbus client instance before removing this instance as being ACTIVE. This can occur, for example, when a client request has been issued and then application stops executing the client function before it has completely finished the request. The maximum S7-1200 limit is 55 seconds. |
| MB_Unit_ID | Word | 255 | Modbus unit identifier:<br>A Modbus TCP server is addressed using its IP address. As a result, the MB_UNIT_ID parameter is not used for Modbus TCP addressing.<br>The MB_UNIT_ID parameter corresponds to the slave address in the Modbus RTU protocol. If a Modbus TCP server is used for a gateway to a Modbus RTU protocol, the MB_UNIT_ID can be used to identify the slave device connected on the serial network. The MB_UNIT_ID would be used to forward the request to the correct Modbus RTU slave address.<br>Please note that some Modbus TCP devices may require the MB_UNIT_ID parameter to be initialized within a restricted range of values. |

| Variable | Data type | Default | description |
|---|---|---|---|
| RCV_TIMEOUT | Real | 2.0 | Time in seconds that the MB_CLIENT waits for a server to respond to a request. |
| Connected | Bool | 0 | Indicates whether the connection to the assigned server is connected or disconnected: 1=connected, 0=disconnected |

Table 12- 53 MB_CLIENT protocol errors

| STATUS (W#16#) | Response code to Modbus client (B#16#) | Modbus protocol errors |
|---|---|---|
| 8381 | 01 | Function code not supported |
| 8382 | 03 | Data length error |
| 8383 | 02 | Data address error or access outside the bounds of the MB_HOLD_REG address area |
| 8384 | 03 | Data value error |
| 8385 | 03 | Data diagnostic code value not supported (function code 08) |

Table 12- 54 MB_CLIENT execution condition codes [1]

| STATUS (W#16#) | MB_CLIENT parameter errors |
|---|---|
| 7001 | MB_CLIENT is waiting for a Modbus server response to a connect or disconnect request, on the assigned TCP port. This is only reported for the first execution of a connect or disconnect operation. |
| 7002 | MB_CLIENT is waiting for a Modbus server response to a connect or disconnect request, for the assigned TCP port. This will be reported for any subsequent executions, while waiting for completion of a connect or disconnect operation. |
| 7003 | A disconnect operation has successfully completed (Only valid for one PLC scan). |
| 80C8 | The server did not respond in the assigned time. MB_CLIENT must receive a response using the transaction ID that was originally transmitted within the assigned time or this error is returned. Check the connection to the Modbus server device.<br>This error is only reported after any configured retries (if applicable) have been attempted. |
| 8188 | Invalid mode value |
| 8189 | Invalid data address value |
| 818A | Invalid data length value |
| 818B | Invalid pointer to the DATA_PTR area. This can be the combination of MB_DATA_ADDRESS + MB_DATA_LEN. |
| 818C | Pointer to a optimized DATA_PTR area (must be a standard DB area or M memory area) |
| 8200 | The port is busy processing an existing Modbus request. |
| 8380 | Received Modbus frame is malformed or too few bytes have been received. |

| STATUS (W#16#) | MB_CLIENT parameter errors |
|---|---|
| 8387 | The assigned Connection ID parameter is different from the ID used for previous requests. There can only be a single Connection ID used within each MB_CLIENT instance DB.<br>This is also used as an internal error if the Modbus TCP protocol ID received from a server is not 0. |
| 8388 | A Modbus server returned a quantity of data that is different than what was requested. This applies to Modbus functions 15 or 16 only. |

[1] In addition to the MB_CLIENT errors listed above, errors can be returned from the underlying T block communication instructions (TCON, TDISCON, TSEND, and TRCV).

## See also

TCON, TDISCON, TSEND, AND TRCV (Page 439)

## 12.5.2.2 MB_SERVER (Modbus TCP)

Table 12- 55 MB_SERVER instruction

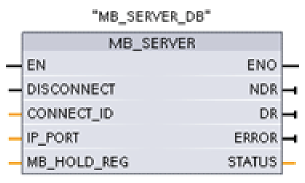
| LAD / FBD | SCL | Description |
|---|---|---|
|  "MB_SERVER_DB"<br>MB_SERVER<br>EN<br>DISCONNECT<br>CONNECT_ID<br>IP_PORT<br>MB_HOLD_REG<br>ENO<br>NDR<br>DR<br>ERROR<br>STATUS | `"MB_SERVER_DB"(`<br>`DISCONNECT:=_bool_in_,`<br>`CONNECT_ID:=_uint_in_,`<br>`IP_PORT:=_uint_in_,`<br>`NDR=>_bool_out_,`<br>`DR=>_bool_out_,`<br>`ERROR=>_bool_out_,`<br>`STATUS=>_word_out_,`<br>`MB_HOLD_REG:=_variant_inout_);` | MB_SERVER communicates as a Modbus TCP server through the PROFINET connector on the S7-1200 CPU. No additional communication hardware module is required.<br>MB_SERVER can accept a request to connect with Modbus TCP client , receive a Modbus function request, and send a response message. |

Table 12- 56 Data types for the parameters

| Parameter and type | | Data type | Description |
|---|---|---|---|
| DISCONNECT | IN | Bool | MB_SERVER attempts to make a "passive" connection with a partner device. This means that the server is passively listening for a TCP connection request from any requesting IP address.<br>If DISCONNECT = 0 and a connection does not exist, then a passive connection can be initiated.<br>If DISCONNECT = 1 and a connection exists, then a disconnect operation is initiated. This allows your program to control when a connection is accepted. Whenever this input is enabled, no other operation will be attempted. |
| CONNECT_ID | IN | UInt | CONNECT_ID uniquely identifies each connection within the PLC. Each unique instance of the MB_CLIENT or MB_SERVER instruction must contain a unique CONNECT_ID parameter. |

| Parameter and type | | Data type | Description |
|---|---|---|---|
| IP_PORT | IN | UInt | Default value = 502: The IP port number that identifies the IP port that will be monitored for a connection request from a Modbus client.<br>These TCP port numbers are not allowed for a MB_SERVER passive connection: 20, 21, 25, 80, 102, 123, 5001, 34962, 34963, and 34964. |
| MB_HOLD_REG | IN_OUT | Variant | Pointer to the MB_SERVER Modbus holding register: The holding register must either be a standard global DB or a M memory address. This memory area is used to hold the values a Modbus client is allowed to access using Modbus register functions 3 (read), 6 (write), and 16 (write). |
| NDR | OUT | Bool | New Data Ready: 0 = No new data, 1 = Indicates that new data has been written by a Modbus client |
| DR | OUT | Bool | Data Read: 0 = No data read, 1 = Indicates that data has been read by a Modbus client. |
| ERROR | OUT | Bool | The ERROR bit is TRUE for one scan, after MB_SERVER execution was terminated with an error. The error code value at the STATUS parameter is valid only during the single cycle where ERROR = TRUE. |
| STATUS | OUT | Word | Execution condition code |

MB_SERVER allows incoming Modbus function codes (1, 2, 4, 5, and 15) to read or write bits and words directly in the input process image and output process image of the S7-1200 CPU. For data transfer function codes (3, 6, and 16), the MB_HOLD_REG parameter must be defined as a data type larger than a byte. The following table shows the mapping of Modbus addresses to the process image in the CPU.

Table 12- 57 Mapping of Modbus addresses to the process image

| Modbus functions | | | | | | S7-1200 | |
|---|---|---|---|---|---|---|---|
| Codes | Function | Data area | Address range | | | Data area | CPU address |
| 01 | Read bits | Output | 1 | To | 8192 | Output Process Image | Q0.0 to Q1023.7 |
| 02 | Read bits | Input | 10001 | To | 18192 | Input Process Image | I0.0 to I1023.7 |
| 04 | Read words | Input | 30001 | To | 30512 | Input Process Image | IW0 to IW1022 |
| 05 | Write bit | Output | 1 | To | 8192 | Output Process Image | Q0.0 to Q1023.7 |
| 15 | Write bits | Output | 1 | To | 8192 | Output Process Image | Q0.0 to Q1023.7 |

Incoming Modbus message function codes function codes (3, 6, and 16) read or write words in a Modbus holding register which can be an M memory address range or a data block. The type of holding register is specified by the MB_HOLD_REG parameter.

**Note**

**MB_HOLD_REG parameter assignment**

The Modbus Holding Register can be in a standard global DB or a M memory address.

For A Modbus holding register in M memory, use the standard Any Pointer format. This is in the format P#"Bit Address" "Data Type" "Length". An example would be P#M1000.0 WORD 500

The following table shows examples of Modbus address to holding register mapping used for Modbus function codes 03 (read words), 06 (write word), and 16 (write words). The actual upper limit of DB addresses is determined by the maximum work memory limit and M memory limit, for each CPU model.

Table 12- 58 Mapping examples of Modbus address to CPU memory address

| Modbus Address | MB_HOLD_REG parameter examples | | |
|---|---|---|---|
| | P#M100.0 Word 5 | P#DB10.DBx0.0 Word 5 | "Recipe".ingredient |
| 40001 | MW100 | DB10.DBW0 | "Recipe".ingredient[1] |
| 40002 | MW102 | DB10.DBW2 | "Recipe".ingredient[2] |
| 40003 | MW104 | DB10.DBW4 | "Recipe".ingredient[3] |
| 40004 | MW106 | DB10.DBW6 | "Recipe".ingredient[4] |
| 40005 | MW108 | DB10.DBW8 | "Recipe".ingredient[5] |

## Multiple server connections

Multiple server connections may be created. This permits a single PLC to establish concurrent connections to multiple Modbus TCP clients.

A Modbus TCP server can support concurrent connections up to the maximum number of Open User Communications connections allowed by the PLC. The total number of connections for a PLC, including Modbus TCP Clients and Servers, must not exceed the maximum number of supported Open User Communications connections (Page 424). The Modbus TCP connections may be shared between Client and/or Server type connections.

Individual server connection must follow these rules:

- Each MB_SERVER connection must use a distinct instance DB.
- Each MB_SERVER connection must be established with a unique IP port number. Only 1 connection per port is supported.
- Each MB_SERVER connection must use a unique connection ID.
- The MB_SERVER must be called individually for each connection (with its respective instance DB).

The Connection ID must be unique for each individual connection. This means a single, unique Connection ID must only be used with each individual instance DB. In summary, the instance DB and the Connection ID are paired together and must be unique for every connection.

Table 12- 59 Modbus diagnostic function codes

| MB_SERVER Modbus diagnostic functions | | |
|---|---|---|
| Codes | Sub-function | Description |
| 08 | 0x0000 | Return query data echo test: The MB_SERVER will echo back to a Modbus client a word of data that is received. |
| 08 | 0x000A | Clear communication event counter: The MB_SEVER will clear out the communication event counter that is used for Modbus function 11. |
| 11 | | Get communication event counter: The MB_SERVER uses an internal communication event counter for recording the number of successful Modbus read and write requests that are sent to the Modbus server. The counter does not increment on any Function 8 or Function 11 requests. It is also not incremented on any requests that result in a communication error.<br>The broadcast function is not available for Modbus TCP, because only one client-server connection exists at any one time. |

## MB_SERVER variables

This table shows the public static variables stored in the MB_SERVER instance data block that can be used in your program

Table 12- 60 MB_SERVER public static variables

| Variable | Data type | Default value | Description |
|---|---|---|---|
| HR_Start_Offset | Word | 0 | Assigns the starting address of the Modbus Holding register |
| Request_Count | Word | 0 | The number of all requests received by this server. |
| Server_Message_Count | Word | 0 | The number of requests received for this specific server. |
| Xmt_Rcv_Count | Word | 0 | The number of transmissions or receptions that have encountered an error. Also, incremented if a message is received that is an invalid Modbus message. |
| Exception_Count | Word | 0 | Modbus specific errors that require a returned exception |
| Success_Count | Word | 0 | The number of requests received for this specific server that have no protocol errors. |
| Connected | Bool | 0 | Indicates whether the connection to the assigned client is connected or disconnected: 1=connected, 0=disconnected |

Your program can write values to the HR_Start_Offset and control Modbus server operations. The other variables can be read to monitor Modbus status.

## HR_Start_Offset

Modbus holding register addresses begin at 40001 . These addresses correspond to the beginning PLC memory address of the holding register. However, you can configure the "HR_Start_Offset" variable to start the beginning Modbus holding register address at another value instead of 40001.

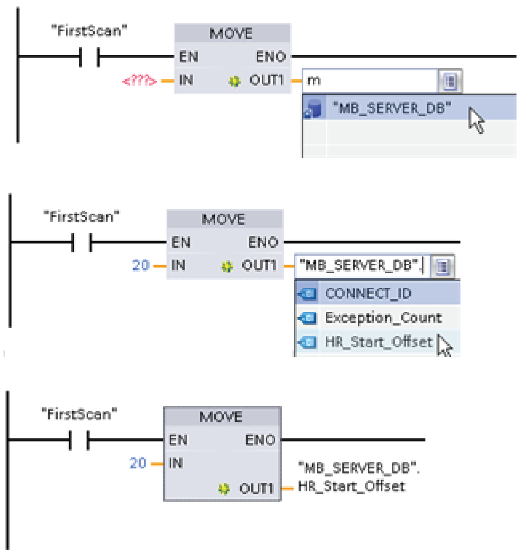
For example, if the holding register is configured to start at MW100 and is 100 words long. An offset of 20 specifies a beginning holding register address of 40021 instead of 40001. Any address below 40021 and above 40119 will result in an addressing error.

Table 12- 61 Example of Modbus holding register addressing

| HR_Start_Offset | Address | Minimum | Maximum |
|---|---|---|---|
| 0 | Modbus address (Word) | 40001 | 40099 |
| | S7-1200 address | MW100 | MW298 |
| 20 | Modbus address (Word) | 40021 | 40119 |
| | S7-1200 address | MW100 | MW298 |

HR_Start_Offset is a word value that specifies the starting address of the Modbus holding register and is stored in the MB_SERVER instance data block. You can set this public static variable value by using the parameter helper drop-list, after MB_SERVER is placed in your program.

For example, after MB_SERVER is placed in a LAD network, you can go to a previous network and assign the HR_Start_Offset value. The value must be assigned prior to execution of MB_SERVER.



Entering a Modbus server variable using the default DB name:

1. Set the cursor in the parameter field and type an m character.
2. Select "MB_SERVER_DB" from the drop-list of DB names.
3. Select "MB_SERVER_DB.HR_Start_Offset" from the drop-list of DB variables.

Table 12- 62 MB_SERVER execution condition codes [1]

| STATUS (W#16#) | Response code to Modbus server (B#16#) | Modbus protocol errors |
|---|---|---|
| 7001 | | MB_SERVER is waiting for a Modbus client to connect to the assigned TCP port. This code is reported on the first execution of a connect or disconnect operation. |
| 7002 | | MB_SERVER is waiting for a Modbus client to connect to the assigned TCP port. This code is reported for any subsequent executions, while waiting for completion of a connect or disconnect operation. |
| 7003 | | A disconnect operation has successfully completed (Only valid for one PLC scan). |
| 8187 | | Invalid pointer to MB_HOLD_REG: area is too small |

| STATUS (W#16#) | Response code to Modbus server (B#16#) | Modbus protocol errors |
|---|---|---|
| 818C | | Pointer to a optimized MB_HOLD_REG area (must be a standard DB area or M memory area) or Blocked process timeout exceeds the limit of 55 seconds. (S7-1200 specific) |
| 8381 | 01 | Function code not supported |
| 8382 | 03 | Data length error |
| 8383 | 02 | Data address error or access outside the bounds of the MB_HOLD_REG address area |
| 8384 | 03 | Data value error |
| 8385 | 03 | Data diagnostic code value not supported (function code 08) |

[1] In addition to the MB_SERVER errors listed above, errors can be returned from the underlying T block communication instructions (TCON, TDISCON, TSEND, and TRCV).

## See also

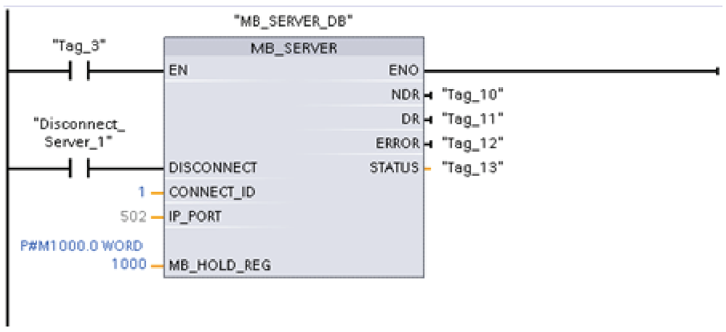
TCON, TDISCON, TSEND, AND TRCV (Page 439)

### 12.5.2.3 MB_SERVER example: Multiple TCP connections

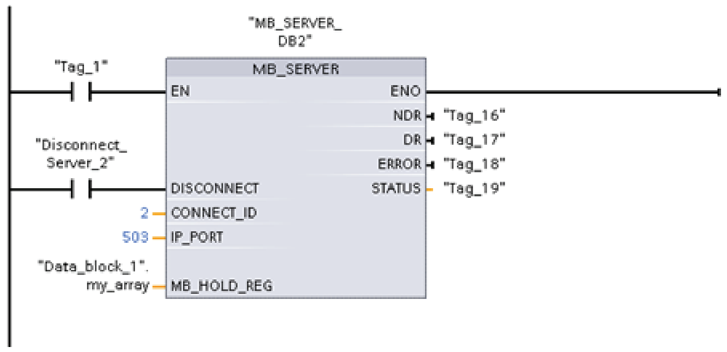
You can have multiple Modbus TCP server connections. To accomplish this, MB_SERVER must be independently executed for each connection. Each connection must use an independent instance DB, connection ID, and IP port. The S7-1200 allows only one connection per IP port.

For best performance, MB_SERVER should be executed every cycle for each connection.

**Network 1:** Connection #1 with independent IP_PORT, connection ID, and instance DB



**Network 2:** Connection #2 with independent IP_PORT, connection ID, and instance DB



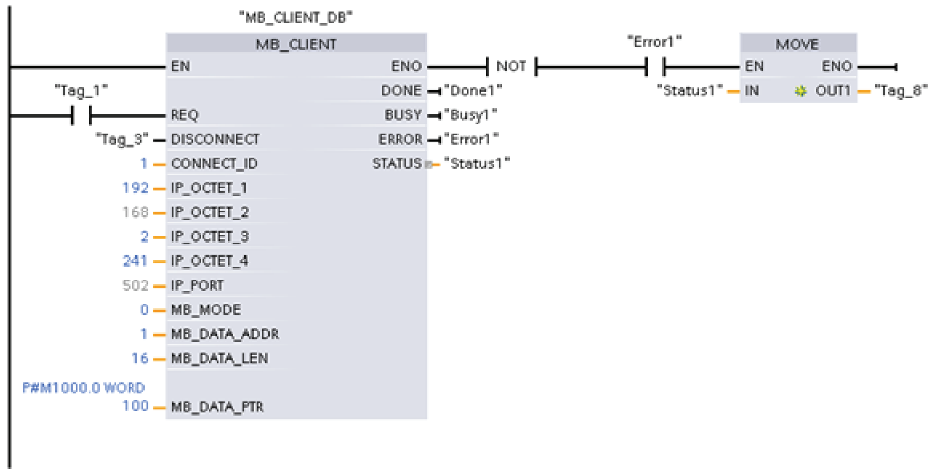
### 12.5.2.4 MB_CLIENT example 1: Multiple requests with common TCP connection

Multiple Modbus client requests can be sent over the same connection. To accomplish this, use the same instance DB, connection ID, and port number.

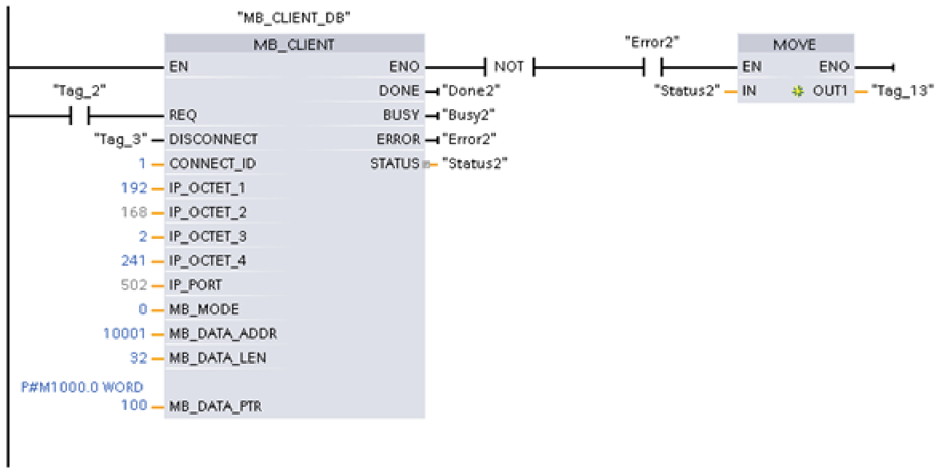
Only 1 client can be active at any given time. Once a client completes its execution, the next client begins execution. Your program is responsible for the order of execution.

The example shows both clients writing to the same memory area. Also, a returned error is captured which is optional.

**Network 1:** Modbus function 1 - Read 16 output image bits



**Network 2:** Modbus function 2 - Read 32 input image bits



## 12.5.2.5 MB_CLIENT example 2: Multiple requests with different TCP connections

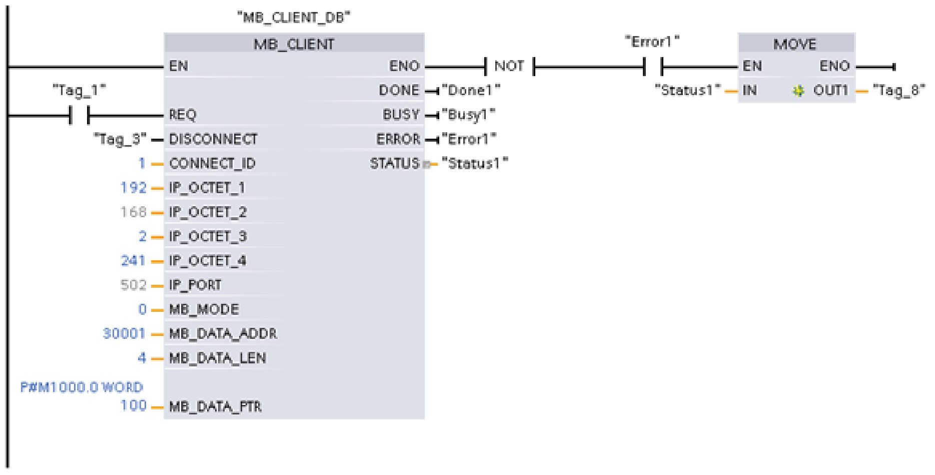
Modbus client requests can be sent over different connections. To accomplish this, different instance DBs, IP addresses, and connection IDs must be used.

The port number must be different if the connections are established to the same Modbus server. If the connections are on different servers, there is no port number restriction.

The example shows both clients writing to the same memory area. Also, a returned error is captured which is optional.

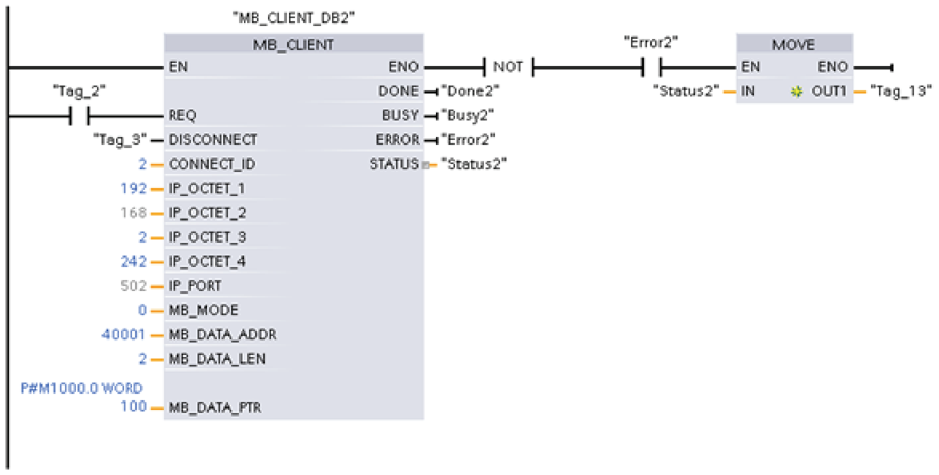
**Network 1:**

Modbus function 4 - Read input words (in S7-1200 memory)



**Network 2:**

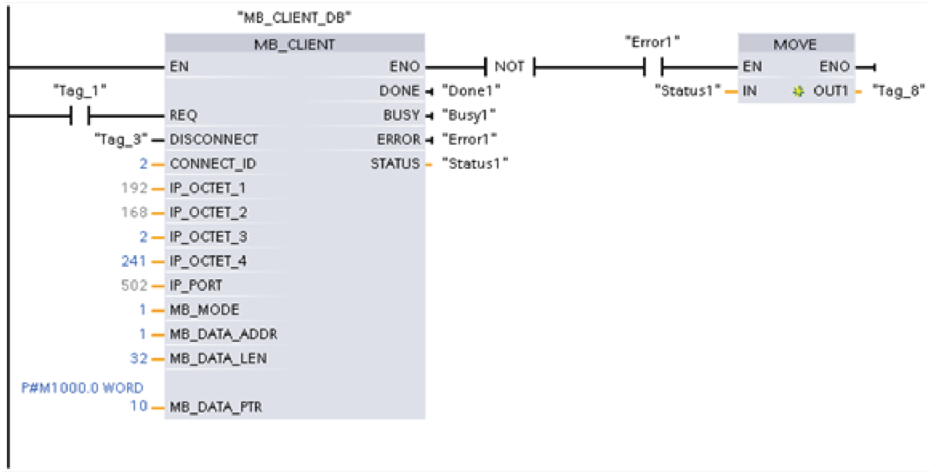
Modbus function 3 - Read holding register words (in S7-1200 memory)



## 12.5.2.6 MB_CLIENT example 3: Output image write request

This example shows a Modbus client request to write the S7-1200 output image.

**Network 1:** Modbus function 15 - Write S7-1200 output image bits

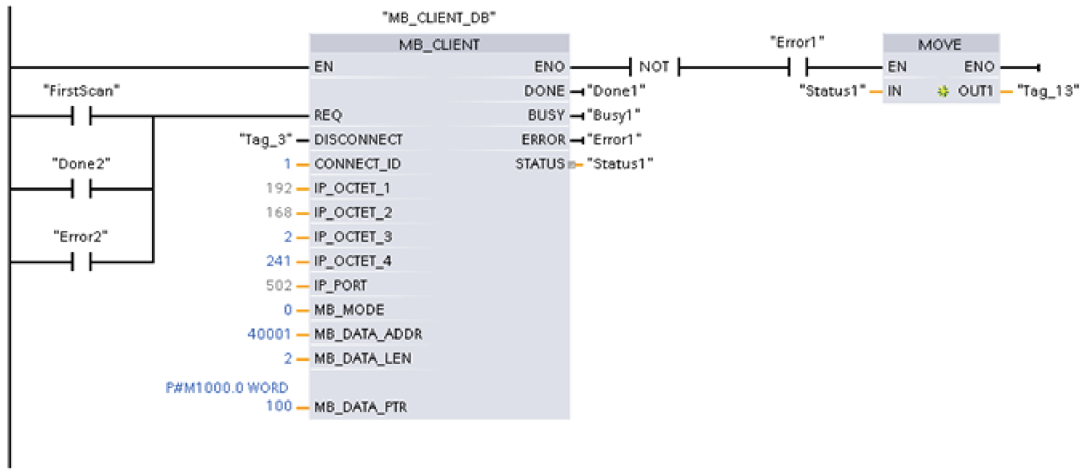


## 12.5.2.7 MB_CLIENT example 4: Coordinating multiple requests

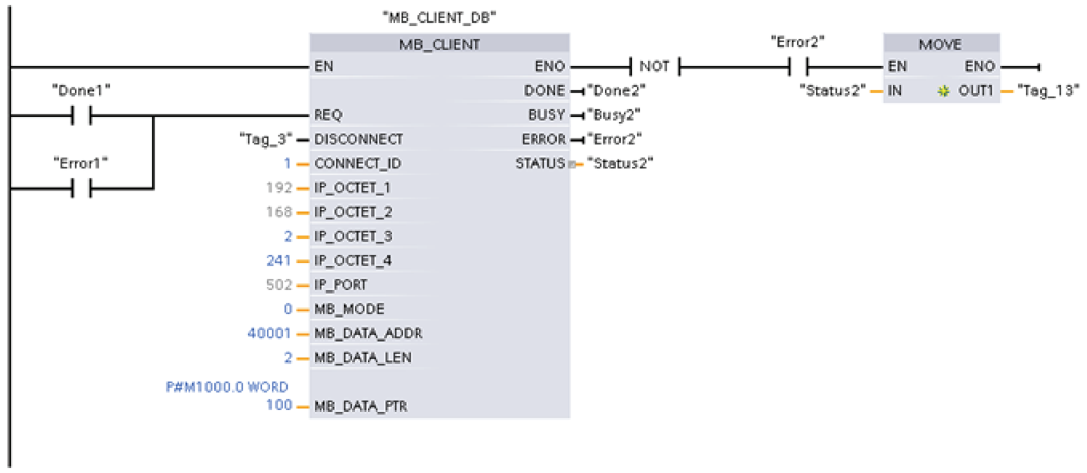
You must ensure that each individual Modbus TCP request finishes execution. This coordination must be provided by your program. The example below shows how the outputs of the first and second client requests can be used to coordinate execution.

The example shows both clients writing to the same memory area. Also, a returned error is captured which is optional.

**Network 1:** Modbus function 3 - Read holding register words



**Network 2:** Modbus function 3 - Read holding register words



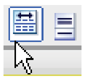
## 12.5.3 Modbus RTU

There are two versions of the Modbus RTU instructions available in STEP 7:

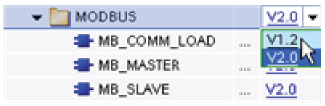
- Version 1 was initially available in STEP 7 Basic V10.5.
- Version 2 is available in STEP 7 Basic/Professional V11. The version 2 design adds REQ and DONE parameters to MB_COMM_LOAD. Also, the MB_ADDR parameter for MB_MASTER and MB_SLAVE now allows a UInt value for extended addressing.

For compatibility and ease of migration, you can choose which instruction version to insert into your user program.

Do not use both 1.x and 2.y instruction versions in the same CPU program. Your program's Modbus instructions must have the same major version number (**1**.x, **2**.y, or **V**.z). The individual instructions within a major version group may have different minor versions (1.**x**).



Click the icon on the instruction tree task card to enable the headers and columns of the instruction tree.



To change the version of the Modbus instructions, select the version from the drop-down list. You can select the group or individual instructions.

When you use the instruction tree to place a Modbus instruction in your program, a new FB instance is created in the project tree. You can see new FB instance in the project tree under PLC_x > Program blocks > System blocks > Program resources.

To verify the version of a Modbus instruction in a program, you must inspect project tree properties and not the properties of a box displayed in the program editor. Select a project tree Modbus FB instance, right-click, select "Properties", and select the "Information" page to see the Modbus instruction version number.

### 12.5.3.1 MB_COMM_LOAD

Table 12- 63 MB_COMM_LOAD instruction

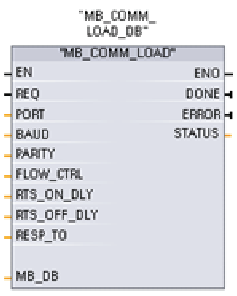
| LAD / FBD | SCL | Description |
|---|---|---|
|   "MB_COMM_ LOAD_DB"<br>"MB_COMM_LOAD"<br>EN<br>ENO<br>REQ<br>DONE<br>PORT<br>ERROR<br>BAUD<br>STATUS<br>PARITY<br>FLOW_CTRL<br>RTS_ON_DLY<br>RTS_OFF_DLY<br>RESP_TO<br>MB_DB | `"MB_COMM_LOAD_DB"(`<br>`    REQ:=_bool_in,`<br>`    PORT:=_uint_in_,`<br>`    BAUD:=_udint_in_,`<br>`    PARITY:=_uint_in_,`<br>`    FLOW_CTRL:=_uint_in_,`<br>`    RTS_ON_DLY:=_uint_in_,`<br>`    RTS_OFF_DLY:=_uint_in_,`<br>`    RESP_TO:=_uint_in_,`<br>`    DONE=>_bool_out,`<br>`    ERROR=>_bool_out_,`<br>`    STATUS=>_word_out_,`<br>`    MB_DB:=_fbtref_inout_);` | The MB_COMM_LOAD instruction configures a PtP port for Modbus RTU protocol communications. Modbus port hardware options: Install up to three CMs (RS485 or RS232), plus one CB (R4845). An instance data block is assigned automatically when you place the MB_COMM_LOAD instruction in your program. |

Table 12- 64 Data types for the parameters

| Parameter and type | | Data type | Description |
|---|---|---|---|
| REQ | IN | Bool | A low to high (positive edge) signal starts the operation.<br>(Version 2.0 only) |
| PORT | IN | Port | After you install and configure a CM or CB communication device, the port identifier appears in the parameter helper drop-list available at the PORT box connection. The assigned CM or CB port value is the device configuration property "hardware identifier". The port symbolic name is assigned in the "System constants" tab of the PLC tag table. |
| BAUD | IN | UDInt | Baud rate selection:<br>300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 76800, 115200, all other values are invalid |
| PARITY | IN | UInt | Parity selection:<br>• 0 – None<br>• 1 – Odd<br>• 2 – Even |
| FLOW_CTRL | IN | UInt | Flow control selection:<br>• 0 – (default) no flow control<br>• 1 – Hardware flow control with RTS always ON (does not apply to RS485 ports)<br>• 2 – Hardware flow control with RTS switched |
| RTS_ON_DLY | IN | UInt | RTS ON delay selection:<br>• 0 – (default) No delay from RTS active until the first character of the message is transmitted<br>• 1 to 65535 – Delay in milliseconds from RTS active until the first character of the message is transmitted (does not apply to RS485 ports). RTS delays shall be applied independent of the FLOW_CTRL selection. |
| RTS_OFF_DLY | IN | UInt | RTS OFF delay selection:<br>• 0 – (default) No delay from the last character transmitted until RTS goes inactive<br>• 1 to 65535 – Delay in milliseconds from the last character transmitted until RTS goes inactive (does not apply to RS485 ports). RTS delays shall be applied independent of the FLOW_CTRL selection. |
| RESP_TO | IN | UInt | Response timeout:<br>Time in milliseconds allowed by MB_MASTER for the slave to respond. If the slave does not respond in this time period, MB_MASTER will retry the request or terminate the request with an error when the specified number of retries has been sent.<br>5 ms to 65535 ms (default value = 1000 ms). |
| MB_DB | IN | Variant | A reference to the instance data block used by the MB_MASTER or MB_SLAVE instructions. After MB_SLAVE or MB_MASTER is placed in your program, the DB identifier appears in the parameter helper drop-list available at the MB_DB box connection. |
| DONE | OUT | Bool | The DONE bit is TRUE for one scan, after the last request was completed with no error. (Version 2.0 only) |

| Parameter and type | | Data type | Description |
|---|---|---|---|
| ERROR | OUT | Bool | The ERROR bit is TRUE for one scan, after the last request was terminated with an error. The error code value at the STATUS parameter is valid only during the single scan where ERROR = TRUE. |
| STATUS | OUT | Word | Execution condition code |

MB_COMM_LOAD is executed to configure a port for the Modbus RTU protocol. Once a port is configured for the Modbus RTU protocol, it can only be used by either the MB_MASTER or MB_SLAVE instructions.

One execution of MB_COMM_LOAD must be used to configure each communication port that is used for Modbus communication. Assign a unique MB_COMM_LOAD instance DB for each port that you use. You can install up to three communication modules (RS232 or RS485) and one communication board (RS485) in the CPU. Call MB_COMM_LOAD from a startup OB and execute it one time or use the first scan system flag (Page 84) to initiate the call to execute it one time. Only execute MB_COMM_LOAD again if communication parameters like baud rate or parity must change.

An instance data block is assigned for MB_MASTER or MB_SLAVE when you place these instructions in your program. This instance data block is referenced when you specify the MB_DB parameter for the MB_COMM_LOAD instruction.

## MB_COMM_LOAD data block variables

The following table shows the public static variables stored in the instance DB for the MB_COMM_LOAD that can be used in your program.

Table 12- 65 Static variables in the instance DB

| Variable | Data type | Description |
|---|---|---|
| ICHAR_GAP | Word | Delay for Inter-character gap between characters. This parameter is specified in milliseconds and is used to increase the expected amount of time between received characters. The corresponding number of bit times for this parameter is added to the Modbus default of 35 bit times (3.5 character times). |
| RETRIES | Word | Number of retries that the master will attempt before returning the no response error code 0x80C8. |

Table 12- 66 MB_COMM_LOAD execution condition codes [1]

| STATUS (W#16#) | Description |
|---|---|
| 0000 | No error |
| 8180 | Invalid port ID value (wrong port/hardware identifier for communication module) |
| 8181 | Invalid baud rate value |
| 8182 | Invalid parity value |
| 8183 | Invalid flow control value |

| STATUS (W#16#) | Description |
|---|---|
| 8184 | Invalid response timeout value (response timeout less than the 5 ms minimum) |
| 8185 | MB_DB parameter is not an instance data block of a MB_MASTER or MB_SLAVE instruction. |

[1] In addition to the MB_COMM_LOAD errors listed above, errors can be returned from the underlying PtP communication instructions.

## See also

Point-to-Point instructions (Page 566)

### 12.5.3.2 MB_MASTER

Table 12- 67 MB_MASTER instruction

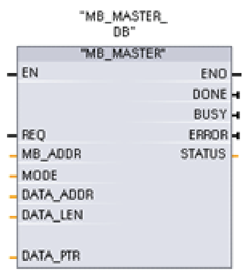
| LAD / FBD | SCL | Description |
|---|---|---|
|  "MB_MASTER_DB"<br>"MB_MASTER"<br>EN<br>ENO<br>DONE<br>BUSY<br>REQ<br>ERROR<br>MB_ADDR<br>STATUS<br>MODE<br>DATA_ADDR<br>DATA_LEN<br>DATA_PTR | `"MB_MASTER_DB"(`<br>`    REQ:=_bool_in_,`<br>`    MB_ADDR:=_uint_in_,`<br>`    MODE:=_usint_in_,`<br>`    DATA_ADDR:=_udint_in_,`<br>`    DATA_LEN:=_uint_in_,`<br>`    DONE=>_bool_out_,`<br>`    BUSY=>_bool_out_,`<br>`    ERROR=>_bool_out_,`<br>`    STATUS=>_word_out_,`<br>`    DATA_PTR:=_variant_inout_);` | The MB_MASTER instruction communicates as a Modbus master using a port that was configured by a previous execution of the MB_COMM_LOAD instruction. An instance data block is assigned automatically when you place the MB_MASTER instruction in your program. This MB_MASTER instance data block is used when you specify the MB_DB parameter for the MB_COMM_LOAD instruction. |

Table 12- 68 Data types for the parameters

| Parameter and type | | Data type | Description |
|---|---|---|---|
| REQ | IN | Bool | 0=No request<br>1= Request to transmit data to Modbus slave |
| MB_ADDR | IN | V1.0: USInt<br>V2.0: UInt | Modbus RTU station address:<br>Standard addressing range (1 to 247)<br>Extended addressing range (1 to 65535)<br>The value of 0 is reserved for broadcasting a message to all Modbus slaves. Modbus function codes 05, 06, 15 and 16 are the only function codes supported for broadcast. |
| MODE | IN | USInt | Mode Selection: Specifies the type of request (read, write, or diagnostic). See the Modbus functions table below for details. |
| DATA_ADDR | IN | UDInt | Starting Address in the slave: Specifies the starting address of the data to be accessed in the Modbus slave. See the Modbus functions table below for valid addresses. |
| DATA_LEN | IN | UInt | Data Length: Specifies the number of bits or words to be accessed in this request. See the Modbus functions table below for valid lengths. |

| Parameter and type | | Data type | Description |
|---|---|---|---|
| DATA_PTR | IN | Variant | Data Pointer: Points to the M or DB address (Standard DB type) for the data being written or read. |
| DONE | OUT | Bool | The DONE bit is TRUE for one scan, after the last request was completed with no error. |
| BUSY | OUT | Bool | • 0 – No MB_MASTER operation in progress<br>• 1 – MB_MASTER operation in progress |
| ERROR | OUT | Bool | The ERROR bit is TRUE for one scan, after the last request was terminated with an error. The error code value at the STATUS parameter is valid only during the single scan where ERROR = TRUE. |
| STATUS | OUT | Word | Execution condition code |

## Modbus master communication rules

- MB_COMM_LOAD must be executed to configure a port before a MB_MASTER instruction can communicate with that port.
- If a port is to be used to initiate Modbus master requests, that port should not be used by MB_SLAVE . One or more instances of MB_MASTER execution can be used with that port, but all MB_MASTER execution must use the same MB_MASTER instance DB for that port.
- The Modbus instructions do not use communication interrupt events to control the communication process. Your program must poll the MB_MASTER instruction for transmit and receive complete conditions.
- It is recommended that you call all MB_MASTER execution for a given port from a program cycle OB. Modbus master instructions may execute in only one of the program cycle or cyclic/time delay execution levels. They must not execute in both execution priority levels. Pre-emption of a Modbus Master instruction by another Modbus master instruction in a higher priority execution priority level will result in improper operation. Modbus master instructions must not execute in the startup, diagnostic or time error execution priority levels.
- Once a master instruction initiates a transmission, this instance must be continually executed with the EN input enabled until a DONE=1 state or ERROR=1 state is returned. A particular MB_MASTER instance is considered active until one of these two events occurs. While the original instance is active, any call to any other instance with the REQ input enabled will result in an error. If the continuous execution of the original instance stops, the request state remains active for a period of time specified by the static variable Blocked_Proc_Timeout. Once this period of time expires, the next master instruction called with an enabled REQ input will become the active instance. This prevents a single Modbus master instance from monopolizing or locking access to a port. If the original active instance is not enabled within the period of time specified by the static variable "Blocked_Proc_Timeout", then the next execution by this instance (with REQ not set) will clear the active state. If (REQ is set), then this execution initiates a new master request as if no other instance was active.

## REQ parameter

0 = No request; 1 = Request to transmit data to Modbus Slave

You may control this input either through the use of a level or edge triggered contact. Whenever this input is enabled, a state machine is started to ensure that no other MB_MASTER using the same instance DB is allowed to issue a request, until the current request is completed. All other input states are captured and held internally for the current request, until the response is received or an error detected.

If the same instance of MB_MASTER is executed again with REQ input = 1 before the completion of the current request, then no subsequent transmissions are made. However, when the request is completed, a new request is issued whenever MB_MASTER is executed again with REQ input = 1.

## DATA_ADDR and MODE parameters select the Modbus function type

DATA_ADDR (starting Modbus address in the slave): Specifies the starting address of the data to be accessed in the Modbus slave.

The MB_MASTER instruction uses a MODE input rather than a Function Code input. The combination of MODE and Modbus address determine the Function Code that is used in the actual Modbus message. The following table shows the correspondence between parameter MODE, Modbus function code, and Modbus address range.

Table 12- 69 Modbus functions

| MODE | Modbus Function | Data length | Operation and data | Modbus Address |
|---|---|---|---|---|
| 0 | 01 | 1 to 2000<br>1 to 1992 [1] | Read output bits:<br>1 to (1992 or 2000) bits per request | 1 to 9999 |
| 0 | 02 | 1 to 2000<br>1 to 1992 [1] | Read input bits:<br>1 to (1992 or 2000) bits per request | 10001 to 19999 |
| 0 | 03 | 1 to 125<br>1 to 124 [1] | Read Holding registers:<br>1 to (124 or 125) words per request | 40001 to 49999 or<br>400001 to 465535 |
| 0 | 04 | 1 to 125<br>1 to 124 [1] | Read input words:<br>1 to (124 or 125) words per request | 30001 to 39999 |
| 1 | 05 | 1 | Write one output bit:<br>One bit per request | 1 to 9999 |
| 1 | 06 | 1 | Write one holding register:<br>1 word per request | 40001 to 49999 or<br>400001 to 465535 |
| 1 | 15 | 2 to 1968<br>2 to 1960 [1] | Write multiple output bits:<br>2 to (1960 or 1968) bits per request | 1 to 9999 |
| 1 | 16 | 2 to 123<br>2 to 122 [1] | Write multiple holding registers:<br>2 to (122 or 123) words per request | 40001 to 49999 or<br>400001 to 465535 |
| 2 | 15 | 1 to 1968<br>2 to 1960 [1] | Write one or more output bits:<br>1 to (1960 or 1968) bits per request | 1 to 9999 |
| 2 | 16 | 1 to 123<br>1 to 122 [1] | Write one or more holding registers:<br>1 to (122 or 123) words per request | 40001 to 49999 or<br>400001 to 465535 |

| MODE | Modbus Function | Data length | Operation and data | Modbus Address |
|---|---|---|---|---|
| 11 | 11 | 0 | Read the slave communication status word and event counter. The status word indicates busy (0 – not busy, 0xFFFF - busy). The event counter is incremented for each successful completion of a message.<br>Both the DATA_ADDR and DATA_LEN operands of MB_MASTER are ignored for this function. | |
| 80 | 08 | 1 | Check slave status using data diagnostic code 0x0000 (Loopback test – slave echoes the request)<br>1 word per request | |
| 81 | 08 | 1 | Reset slave event counter using data diagnostic code 0x000A<br>1 word per request | |
| 3 to 10,<br>12 to 79,<br>82 to 255 | | | Reserved | |

[1] For "Extended Addressing" mode the maximum data lengths are reduced by 1 byte or 1 word depending upon the data type used by the function.

## DATA_PTR parameter

The DATA_PTR parameter points to the DB or M address that is written to or read from. If you use a data block, then you must create a global data block that provides data storage for reads and writes to Modbus slaves.

**Note**

**The DATA_PTR data block type must allow direct addressing**

The data block must allow both direct (absolute) and symbolic addressing. When you create the data block the "Standard" access attribute must be selected.

## Data block structures for the DATA_PTR parameter

- These data types are valid for **word reads** of Modbus addresses 30001 to 39999, 40001 to 49999, and 400001 to 465536 and also for **word writes** to Modbus addresses 40001 to 49999 and 400001 to 465536.
  - Standard array of WORD, UINT, or INT data types
  - Named WORD, UINT, or INT structure where each element has a unique name and 16 bit data type.
  - Named complex structure where each element has a unique name and a 16 or 32 bit data type.
- For **bit reads** and writes of Modbus addresses 00001 to 09999 and bit reads of 10001 to 19999.
  - Standard array of Boolean data types.
  - Named Boolean structure of uniquely named Boolean variables..
- Although not required, it is recommended that each MB_MASTER instruction have its own separate memory area. The reason for this recommendation is that there is a greater possibility of data corruption if multiple MB_MASTER instructions are reading and writing to the same memory area.
- There is no requirement that the DATA_PTR data areas be in the same global data block. You can create one data block with multiple areas for Modbus reads, one data block for Modbus writes, or one data block for each slave station.

## Modbus master data block variables

The following table shows the public static variables stored in the instance DB for MB_MASTER that can be used in your program.

Table 12- 70 Static variables in the instance DB

| Variable | Data type | Initial value | Description |
|---|---|---|---|
| Blocked_Proc_Timeout | Real | 3.0 | Amount of time (in seconds) to wait for a blocked Modbus Master instance before removing this instance as being ACTIVE. This can occur, for example, when a Master request has been issued and then the program stops calling the Master function before it has completely finished the request. The time value must be greater than 0 and less than 55 seconds, or an error occurs. The default value is .5 seconds. |
| Extended_Addressing | Bool | False | Configures single or double-byte slave addressing. The default value = 0.<br>(0=single byte address, 1=double-byte address) |

Your program can write values to the Blocked_Proc_Timeout and Extended_Addressing variables to control Modbus master operations. See the MB_SLAVE topic description of HR_Start_Offset and Extended_Addressing for an example of how to use these variables in the program editor and details about Modbus extended addressing (Page 647).

## Condition codes

Table 12- 71 MB_MASTER execution condition codes (communication and configuration errors) [1]

| STATUS (W#16#) | Description |
|---|---|
| 0000 | No error |
| 80C8 | Slave timeout. Check baud rate, parity, and wiring of slave. |
| 80D1 | The receiver issued a flow control request to suspend an active transmission and never re-enabled the transmission during the specified wait time.<br>This error is also generated during hardware flow control when the receiver does not assert CTS within the specified wait time. |
| 80D2 | The transmit request was aborted because no DSR signal is received from the DCE. |
| 80E0 | The message was terminated because the receive buffer is full. |
| 80E1 | The message was terminated as a result of a parity error. |
| 80E2 | The message was terminated as a result of a framing error. |
| 80E3 | The message was terminated as a result of an overrun error. |
| 80E4 | The message was terminated as a result of the specified length exceeding the total buffer size. |
| 8180 | Invalid port ID value or error with MB_COMM_LOAD instruction |
| 8186 | Invalid Modbus station address |
| 8188 | Invalid Mode specified for broadcast request |
| 8189 | Invalid Data Address value |
| 818A | Invalid Data Length value |
| 818B | Invalid pointer to the local data source/destination: Size not correct |
| 818C | Invalid pointer for DATA_PTR or invalid Blocked_Proc_Timeout: The data area must be a DB (that allows both symbolic and direct access) or a M memory. |
| 8200 | Port is busy processing a transmit request. |

Table 12- 72 MB_MASTER execution condition codes (Modbus protocol errors) [1]

| STATUS (W#16#) | Response code from slave | Modbus protocol errors |
|---|---|---|
| 8380 | - | CRC error |
| 8381 | 01 | Function code not supported |
| 8382 | 03 | Data length error |
| 8383 | 02 | Data address error or address outside the valid range of the DATA_PTR area |
| 8384 | Greater than 03 | Data value error |
| 8385 | 03 | Data diagnostic code value not supported (function code 08) |
| 8386 | - | Function code in the response does not match the code in the request. |
| 8387 | - | Wrong slave responded |
| 8388 | - | The slave response to a write request is incorrect. The write request returned by the slave does not match what the master actually sent. |

[1] In addition to the MB_MASTER errors listed above, errors can be returned from the underlying PtP communication instructions.

## See also

Point-to-Point instructions (Page 566)

### 12.5.3.3 MB_SLAVE

Table 12- 73 MB_SLAVE instruction

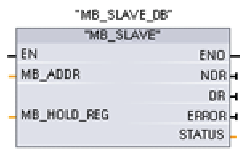
| LAD / FBD | SCL | Description |
|---|---|---|
|  "MB_SLAVE_DB"<br>"MB_SLAVE"<br>EN<br>ENO<br>MB_ADDR<br>NDR<br>DR<br>MB_HOLD_REG<br>ERROR<br>STATUS | `"MB_SLAVE_DB"(`<br>`MB_ADDR:=_uint_in_,`<br>`NDR=>_bool_out_,`<br>`DR=>_bool_out_,`<br>`ERROR=>_bool_out_,`<br>`STATUS=>_word_out_,`<br>`MB_HOLD_REG:=_variant_inout_);` | The MB_SLAVE instruction allows your program to communicate as a Modbus slave through a PtP port on the CM (RS485 or RS232) and CB (RS485). When a remote Modbus RTU master issues a request, your user program responds to the request by MB_SLAVE execution. STEP 7 automatically creates an instance DB when you insert the instruction. Use this MB_SLAVE_DB name when you specify the MB_DB parameter for the MB_COMM_LOAD instruction. |

Table 12- 74 Data types for the parameters

| Parameter and type | | Data type | Description |
|---|---|---|---|
| MB_ADDR | IN | V1.0: USInt<br>V2.0: UInt | The station address of the Modbus slave:<br>Standard addressing range (1 to 247)<br>Extended addressing range (0 to 65535) |
| MB_HOLD_REG | IN | Variant | Pointer to the Modbus Holding Register DB: The Modbus holding register can be M memory or a data block. |
| NDR | OUT | Bool | New Data Ready:<br>• 0 – No new data<br>• 1 – Indicates that new data has been written by the Modbus master |
| DR | OUT | Bool | Data Read:<br>• 0 – No data read<br>• 1 – Indicates that data has been read by the Modbus master |
| ERROR | OUT | Bool | The ERROR bit is TRUE for one scan, after the last request was terminated with an error. If execution is terminated with an error, then the error code value at the STATUS parameter is valid only during the single scan where ERROR = TRUE. |
| STATUS | OUT | Word | Execution error code |

Modbus communication function codes (1, 2, 4, 5, and 15) can read and write bits and words directly in the input process image and output process image of the CPU. For these function codes, the MB_HOLD_REG parameter must be defined as a data type larger than a byte. The following table shows the example mapping of Modbus addresses to the process image in the CPU.

Table 12- 75 Mapping of Modbus addresses to the process image

| Modbus functions | | | | | | S7-1200 | |
|---|---|---|---|---|---|---|---|
| Codes | Function | Data area | Address range | | | Data area | CPU address |
| 01 | Read bits | Output | 1 | to | 8192 | Output Process Image | Q0.0 to Q1023.7 |
| 02 | Read bits | Input | 10001 | to | 18192 | Input Process Image | I0.0 to I1023.7 |
| 04 | Read words | Input | 30001 | to | 30512 | Input Process Image | IW0 to IW1022 |
| 05 | Write bit | Output | 1 | to | 8192 | Output Process Image | Q0.0 to Q1023.7 |
| 15 | Write bits | Output | 1 | to | 8192 | Output Process Image | Q0.0 to Q1023.7 |

Modbus communication function codes (3, 6, 16) use a Modbus holding register which can be a M memory address range or a data block. The type of holding register is specified by the MB_HOLD_REG parameter on the MB_SLAVE instruction.

**Note**

**MB_HOLD_REG data block type**

A Modbus holding register data block must allow both direct (absolute) and symbolic addressing. When you create the data block the "Standard" access attribute must be selected.

The following table shows examples of Modbus address to holding register mapping that is used for Modbus function codes 03 (read words), 06 (write word), and 16 (write words). The actual upper limit of DB addresses is determined by the maximum work memory limit and M memory limit, for each CPU model.

Table 12- 76 Mapping of Modbus addresses to CPU memory

| **Modbus Master Address** | **MB_HOLD_REG parameter examples** | | | | |
|---|---|---|---|---|---|
| | **MW100** | **DB10.DBw0** | **MW120** | **DB10.DBW50** | **"Recipe".ingredient** |
| 40001 | MW100 | DB10.DBW0 | MW120 | DB10.DBW50 | "Recipe".ingredient[1] |
| 40002 | MW102 | DB10.DBW2 | MW122 | DB10.DBW52 | "Recipe".ingredient[2] |
| 40003 | MW104 | DB10.DBW4 | MW124 | DB10.DBW54 | "Recipe".ingredient[3] |
| 40004 | MW106 | DB10.DBW6 | MW126 | DB10.DBW56 | "Recipe".ingredient[4] |
| 40005 | MW108 | DB10.DBW8 | MW128 | DB10.DBW58 | "Recipe".ingredient[5] |

Table 12- 77 Diagnostic functions

| **S7-1200 MB_SLAVE Modbus diagnostic functions** | | |
|---|---|---|
| Codes | Sub-function | Description |
| 08 | 0000H | Return query data echo test: The MB_SLAVE will echo back to a Modbus master a word of data that is received. |

| S7-1200 MB_SLAVE Modbus diagnostic functions | | |
|---|---|---|
| 08 | 000AH | Clear communication event counter: The MB_SLAVE will clear out the communication event counter that is used for Modbus function 11. |
| 11 | | Get communication event counter: The MB_SLAVE uses an internal communication event counter for recording the number of successful Modbus read and write requests that are sent to the Modbus slave. The counter does not increment on any Function 8, Function 11, or broadcast requests. It is also not incremented on any requests that result in a communication error (for example, parity or CRC errors). |

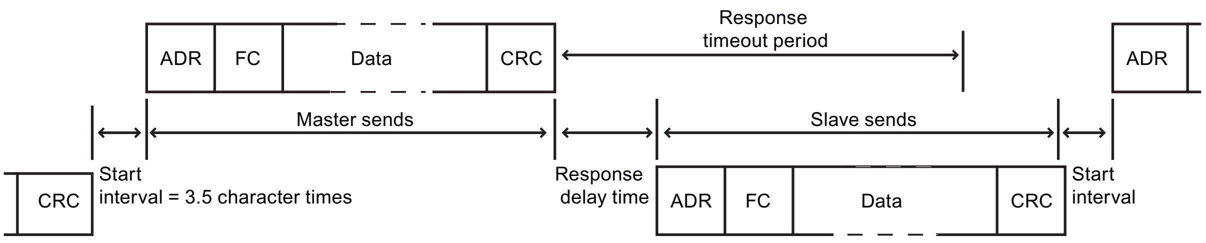
The MB_SLAVE instruction supports broadcast write requests from any Modbus master as long as the request is for accessing valid addresses. MB_SLAVE will produce error code 0x8188 for function codes not supported in broadcast.

## Modbus slave communication rules

- MB_COMM_LOAD must be executed to configure a port, before a MB_SLAVE instruction can communicate through that port.
- If a port is to respond as a slave to a Modbus master, then do not program that port with the MB_MASTER instruction.
- Only one instance of MB_SLAVE can be used with a given port, otherwise erratic behavior may occur.
- The Modbus instructions do not use communication interrupt events to control the communication process. Your program must control the communication process by polling the MB_SLAVE instruction for transmit and receive complete conditions.
- The MB_SLAVE instruction must execute periodically at a rate that allows it to make a timely response to incoming requests from a Modbus master. It is recommended that you execute MB_SLAVE every scan from a program cycle OB. Executing MB_SLAVE from a cyclic interrupt OB is possible, but is not recommended because of the potential for excessive time delays in the interrupt routine to temporarily block the execution of other interrupt routines.

## Modbus signal timing

MB_SLAVE must be executed periodically to receive each request from the Modbus master and then respond as required. The frequency of execution for MB_SLAVE is dependent upon the response timeout period of the Modbus master. This is illustrated in the following diagram.



The response timeout period RESP_TO is the amount of time a Modbus master waits for the start of a response from a Modbus slave. This time period is not defined by the Modbus protocol, but is a parameter of each Modbus master. The frequency of execution (the time between one execution and the next execution) of MB_SLAVE must be based on the particular parameters of your Modbus master. At a minimum, you should execute MB_SLAVE twice within the response timeout period of the Modbus master.

## Modbus slave variables

This table shows the public static variables stored in the MB_SLAVE instance data block that can be used in your program

Table 12- 78 Modbus slave variables

| Variable | Data type | Description |
|---|---|---|
| HR_Start_Offset | Word | Specifies the starting address of the Modbus Holding register (default = 0) |
| Extended_Addressing | Bool | Configures single or double-byte slave addressing<br>(0=single byte address, 1=double-byte address, default = 0) |
| Request_Count | Word | The number of all requests received by this slave |
| Slave_Message_Count | Word | The number of requests received for this specific slave |
| Bad_CRC_Count | Word | The number of requests received that have a CRC error |
| Broadcast_Count | Word | The number of broadcast requests received |
| Exception_Count | Word | Modbus specific errors that require a returned exception |
| Success_Count | Word | The number of requests received for this specific slave that have no protocol errors |

Your program can write values to the HR_Start_Offset and Extended_Addressing variables and control Modbus slave operations. The other variables can be read to monitor Modbus status.

## HR_Start_Offset

Modbus holding register addresses begin at 40001 or 400001. These addresses correspond to the beginning PLC memory address of the holding register. However, you can configure the "HR_Start_Offset" variable to start the beginning Modbus holding register address at another value instead of 40001 or 400001.

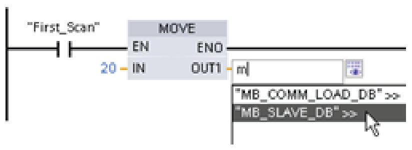
For example, if the holding register is configured to start at MW100 and is 100 words long. An offset of 20 specifies a beginning holding register address of 40021 instead of 40001. Any address below 40021 and above 400119 will result in an addressing error.

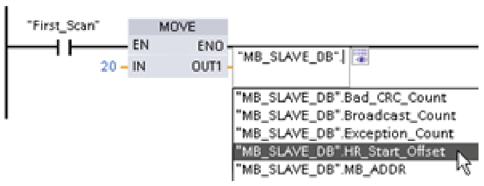
Table 12- 79 Example of Modbus holding register addressing

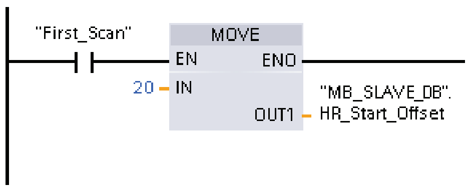
| HR_Start_Offset | Address | Minimum | Maximum |
|---|---|---|---|
| 0 | Modbus address (Word) | 40001 | 40099 |
| | S7-1200 address | MW100 | MW298 |
| 20 | Modbus address (Word) | 40021 | 40119 |
| | S7-1200 address | MW100 | MW298 |

HR_Start_Offset is a word value that specifies the starting address of the Modbus holding register and is stored in the MB_SLAVE instance data block. You can set this public static variable value by using the parameter helper drop-list, after MB_SLAVE is placed in your program.

For example, after MB_SLAVE is placed in a LAD network, you can go to a previous network and assign the HR_Start_Offset value. The value must be assigned prior to execution of MB_SLAVE.




"First_Scan"
MOVE
EN
ENO
20
IN
OUT1
"MB_SLAVE_DB".
"MB_SLAVE_DB".Bad_CRC_Count
"MB_SLAVE_DB".Broadcast_Count
"MB_SLAVE_DB".Exception_Count
"MB_SLAVE_DB".HR_Start_Offset
"MB_SLAVE_DB".MB_ADDR



Entering a Modbus slave variable using the default DB name:

1. Set the cursor in the parameter field and type an m character.
2. Select "MB_SLAVE_DB" from the drop-list.
3. Set the cursor at the right side of the DB name (after the quote character) and enter a period character.
4. Select "MB_SLAVE_DB.HR_Start_Offset" from the drop list.

## Extended_Addressing

The Extended_Addressing variable is accessed in a similar way as the HR_Start_Offset reference discussed above except that the Extended_Addressing variable is a boolean value. The boolean value must be written by an output coil and not a move box.

Modbus slave addressing can be configured to be either a single byte (which is the Modbus standard) or double byte. Extended addressing is used to address more than 247 devices within a single network. Selecting extended addressing allows you to address a maximum of 64000 addresses. A Modbus function 1 frame is shown below as an example.

Table 12- 80 Single-byte slave address (byte 0)

| Function 1 | Byte 0 | Byte 1 | Byte 2 | Byte 3 | Byte 4 | Byte 5 | |
|---|---|---|---|---|---|---|---|
| Request | Slave addr. | F code | Start address | | Length of coils | | |
| Valid Response | Slave addr. | F code | Length | Coil data | | | |
| Error response | Slave addr. | 0x81 | E code | | | | |

Table 12- 81 Double-byte slave address (byte 0 and byte 1)

| | Byte 0 | Byte 1 | Byte 2 | Byte 3 | Byte 4 | Byte 5 | Byte 6 |
|---|---|---|---|---|---|---|---|
| Request | Slave address | | F code | Start address | | Length of coils | |
| Valid Response | Slave address | | F code | Length | Coil data | | |
| Error response | Slave address | | 0x81 | E code | | | |

## Condition codes

Table 12- 82 MB_SLAVE execution condition codes (communication and configuration errors) [1]

| STATUS (W#16#) | Description |
|---|---|
| 80D1 | The receiver issued a flow control request to suspend an active transmission and never re-enabled the transmission during the specified wait time.<br>This error is also generated during hardware flow control when the receiver does not assert CTS within the specified wait time. |
| 80D2 | The transmit request was aborted because no DSR signal is received from the DCE. |
| 80E0 | The message was terminated because the receive buffer is full. |
| 80E1 | The message was terminated as a result of a parity error. |
| 80E2 | The message was terminated as a result of a framing error. |
| 80E3 | The message was terminated as a result of an overrun error. |
| 80E4 | The message was terminated as a result of the specified length exceeding the total buffer size. |
| 8180 | Invalid port ID value or error with MB_COMM_LOAD instruction |
| 8186 | Invalid Modbus station address |
| 8187 | Invalid pointer to MB_HOLD_REG DB: Area is too small |
| 818C | Invalid MB_HOLD_REG pointer to M memory or DB (DB area must allow both symbolic and direct address) |

Table 12- 83 MB_SLAVE execution condition codes (Modbus protocol errors) [1]

| STATUS (W#16#) | Response code from slave | Modbus protocol errors |
|---|---|---|
| 8380 | No response | CRC error |
| 8381 | 01 | Function code not supported or not supported within broadcasts |
| 8382 | 03 | Data length error |
| 8383 | 02 | Data address error or address outside the valid range of the DATA_PTR area |
| 8384 | 03 | Data value error |
| 8385 | 03 | Data diagnostic code value not supported (function code 08) |

[1] In addition to the MB_SLAVE errors listed above, errors can be returned from the underlying PtP communication instructions.

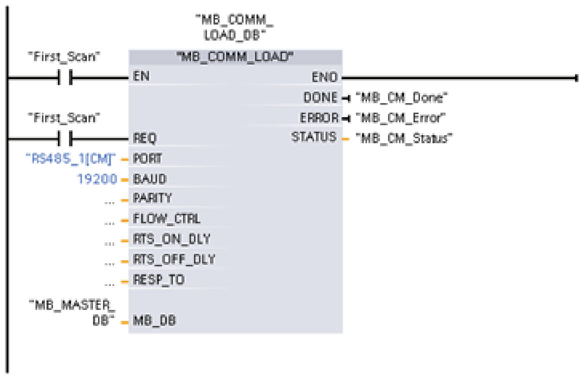
## See also

Point-to-Point instructions (Page 566)

### 12.5.3.4 Modbus RTU master example program

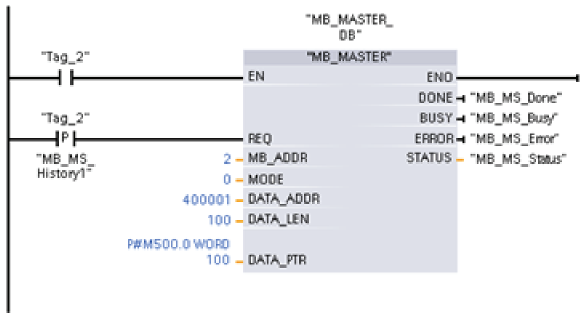
MB_COMM_LOAD is initialized during start-up by using the first scan flag. Execution of MB_COMM_LOAD in this manner should only be done when the serial port configuration will not change at runtime.

**Network 1** Initialize the RS485 module parameters only once during the first scan.

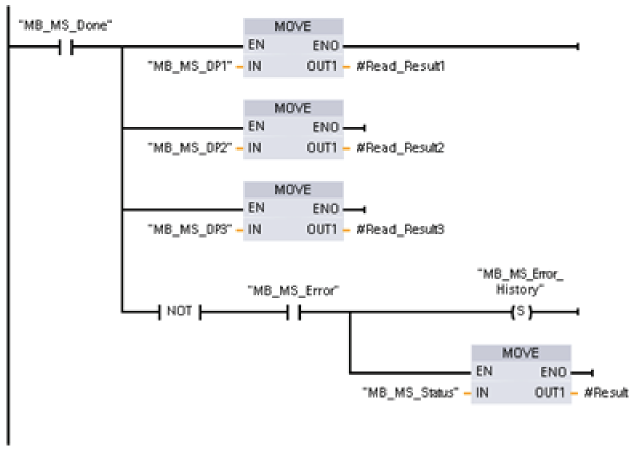


One MB_MASTER instruction is used in the program cycle OB to communicate with a single slave. Additional MB_MASTER instructions can be used in the program cycle OB to communicate with other slaves, or one MB_MASTER FB could be re-used to communicate with additional slaves.

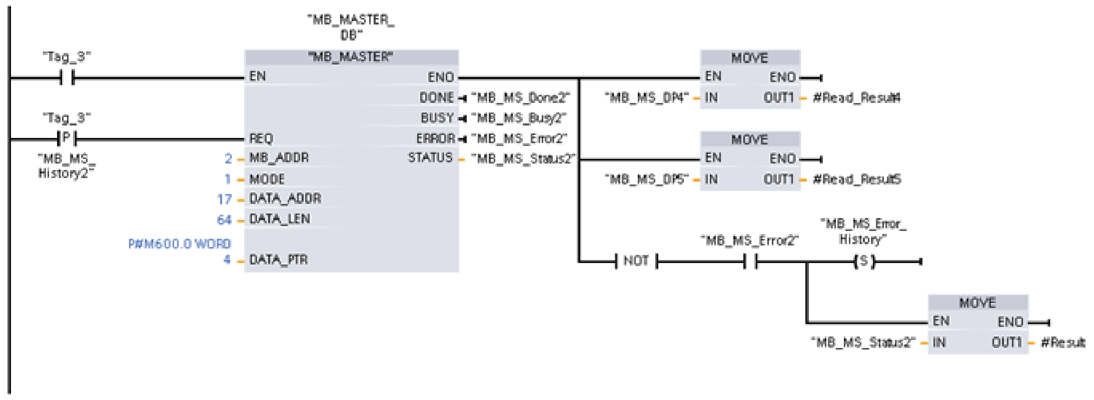
**Network 2** Read 100 words from the slave holding register.



**Network 3** This is an optional network that just shows the values of the first 3 words once the read operation is done.


"MB_MS_Done"
MOVE
EN ENO
"MB_MS_DP1" IN OUT1 #Read_Result1
MOVE
EN ENO
"MB_MS_DP2" IN OUT1 #Read_Result2
MOVE
EN ENO
"MB_MS_DP3" IN OUT1 #Read_Result3
"MB_MS_Error_History"
"MB_MS_Error"
NOT
S
MOVE
EN ENO
"MB_MS_Status" IN OUT1 #Result

**Network 4** Write 64 bits to the output image register starting at slave address Q2.0.

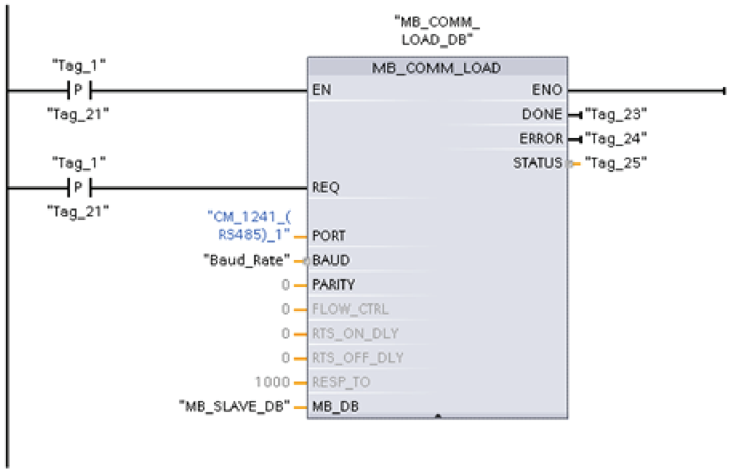


### 12.5.3.5 Modbus RTU slave example program

MB_COMM_LOAD shown below is initialized each time "Tag_1" is enabled.

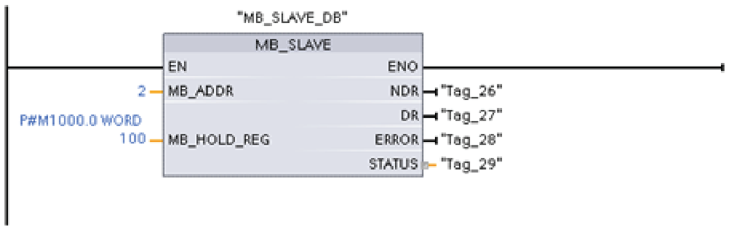
Execution of MB_COMM_LOAD in this manner should only be done when the serial port configuration will change at runtime, as a result of HMI configuration.

Network 1 Initialize the RS485 module parameters each time they are changed by an HMI device.



MB_SLAVE shown below is placed in a cyclic OB that is executed every 10ms. While this does not give the absolute fastest response by the slave, it does provide good performance at 9600 baud for short messages (20 bytes or less in the request).

Network 2 Check for Modbus master requests during each scan. The Modbus holding register is configured for 100 words starting at MW1000.



## 12.6 Telecontrol and TeleService with the CP 1242-7

### 12.6.1 Connection to a GSM network

#### IP-based WAN communication via GPRS

Using the CP 1242-7 communications processor, the S7-1200 can be connected to GSM networks. The CP 1242-7 allows WAN communication from remote stations with a control center and inter-station communication.

Inter-station communication is possible only via a GSM network. For communication between a remote station and a control room, the control center must have a PC with Internet access.

The CP 1242-7 supports the following services for communication via the GSM network:

- GPRS (General Packet Radio Service)

  The packet-oriented service for data transmission "GPRS" is handled via the GSM network.
- SMS (Short Message Service)

  The CP 1242-7 can receive and send SMS messages. The communications partner can be a mobile phone or an S7-1200.

The CP 1242-7 is suitable for use in industry worldwide and supports the following frequency bands:

- 850 MHz
- 900 MHz
- 1 800 MHz
- 1 900 MHz

## Requirements

The equipment used in the stations or the control center depends on the particular application.

- For communication with or via a central control room, the control center requires a PC with Internet access.
- Apart from the station equipment, a remote S7-1200 station with a CP 1242-7 must meet the following requirements to be able to communicate via the GSM network:
  - A contract with a suitable GSM network provider

    If GPRS is used, the contract must allow the use of the GPRS service.

    If there is to be direct communication between stations only via the GSM network, the GSM network provider must assign a fixed IP address to the CPs. In this case, communication between stations is not via the control center.
  - The SIM card belonging to the contract

    The SIM card is inserted in the CP 1242-7.
  - Local availability of a GSM network in the range of the station

## 12.6.2 Applications of the CP 1242-7

The CP 1242-7 can be used for the following applications:

### Telecontrol applications

- Sending messages by SMS

  Via the CP 1242-7, the CPU of a remote S7-1200 station can receive SMS messages from the GSM network or send messages by SMS to a configured mobile phone or an S7-1200.

- Communication with a control center

  Remote S7-1200 stations communicate via the GSM network and the Internet with a telecontrol server in the master station. For data transfer using GPRS, the "TELECONTROL SERVER BASIC" application is installed on the telecontrol server in the master station. The telecontrol server communicates with a higher-level central control system using the integrated OPC server function.

- Communication between S7-1200 stations via a GSM network

  Communication between remote stations with a CP 1242-7 can be handled in two different ways:

  – Inter-station communication via a master station

    In this configuration, a permanent secure connection between S7-1200 stations that communicate with each other and the telecontrol server is established in the master station. Communication between the stations is via the telecontrol server. The CP 1242-7 operates in "Telecontrol" mode.

  – Direct communication between the stations

    For direct communication between stations without the detour via the master station, SIM cards with a fixed IP address are used that allow the stations to address each other directly. The possible communications services and security functions (for example VPN) depend on what is offered by the network provider. The CP 1242-7 operates in "GPRS direct" mode.

### TeleService via GPRS

A TeleService connection can be established between an engineering station with STEP 7 and a remote S7-1200 station with a CP 1242-7 via the GSM network and the Internet. The connection runs from the engineering station via a telecontrol server or a TeleService gateway that acts as an intermediary forwarding frames and establishing the authorization. These PCs use the functions of the "TELECONTROL SERVER BASIC" application.

You can use the TeleService connection for the following purposes:

- Downloading configuration or program data from the STEP 7 project to the station
- Querying diagnostics data on the station

### 12.6.3 Other properties of the CP

#### Other services and functions of the CP 1242-7

- Time-of-day synchronization of the CP via the Internet

  You can set the time on the CP as follows:

  – In "Telecontrol" mode, the time of day is transferred by the telecontrol server. The CP uses this to set its time.

  – In "GPRS direct" mode, the CP can request the time using SNTP.

  To synchronize the CPU time, you can read out the current time from the CP using a block.
- Interim buffering of messages to be sent if there are connection problems
- Increased availability thanks to the option of connecting to a substitute telecontrol server
- Optimized data volume (temporary connection)

  As an alternative to a permanent connection to the telecontrol server, the CP can be configured in STEP 7 with a temporary connection to the telecontrol server. In this case, a connection to the telecontrol server is established only when required.
- Logging the volume of data

  The volumes of data transferred are logged and can be evaluated for specific purposes.

#### Configuration and module replacement

To configure the module, the following configuration tool is required:

STEP 7 version V11.0 SP1 or higher

For STEP 7 V11.0 SP1, you also require support package "CP 1242-7" (HSP0003001).

For process data transfer using GPRS, use the telecontrol communications instructions in the user program of the station.

The configuration data of the CP 1242-7 is stored on the local CPU. This allows simple replacement of the CP when necessary.

You can insert up to three modules of the CP 1242-7 type per S7-1200. This, for example, allows redundant communications paths to be established.

#### Electrical connections

- Power supply of the CP 1242-7

  The CP has a separate connection for the external 24 VDC power supply.
- Wireless interface for the GSM network

  An extra antenna is required for GSM communication. This is connected via the SMA socket of the CP.

### Further information

The CP 1242-7 manual contains detailed information. You will find this on the Internet on the pages of Siemens Industrial Automation Customer Support under the following entry ID:

42330276 (http://support.automation.siemens.com/WW/view/en/42330276)

## 12.6.4 Accessories

### The ANT794-4MR GSM/GPRS antenna

The following antennas are available for use in GSM/GPRS networks and can be installed both indoors and outdoors:

- Quadband antenna ANT794-4MR

Figure 12-1 ANT794-4MR GSM/GPRS antenna

| Short name | Order no. | Explanation |
|---|---|---|
| ANT794-4MR | 6NH9 860-1AA00 | Quadband antenna (900, 1800/1900 MHz, UMTS); weatherproof for indoor and outdoor areas; 5 m connecting cable connected permanently to the antenna; SMA connector, including installation bracket, screws, wall plugs |

- Flat antenna ANT794-3M

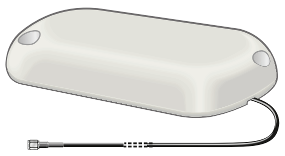


Figure 12-2 Flat antenna ANT794-3M

| Short name | Order no. | Explanation |
|---|---|---|
| ANT794-3M | 6NH9 870-1AA00 | Flat antenna (900, 1800/1900 MHz); weatherproof for indoor and outdoor areas; 1.2 m connecting cable connected permanently to the antenna; SMA connector, including adhesive pad, screws mounting possible |

The antennas must be ordered separately.

### Further information

You will find detailed information in the device manual. You will find this on the Internet on the pages of Siemens Industrial Automation Customer Support under the following entry ID:

23119005 (http://support.automation.siemens.com/WW/view/en/23119005)

## 12.6.5 Configuration examples for telecontrol

Below, you will find several configuration examples for stations with a CP 1242-7.

### Sending messages by SMS

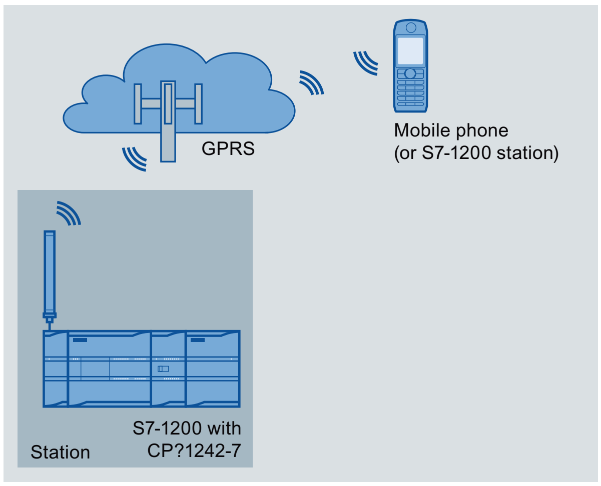


Figure 12-3 Sending messages by SMS from an S7-1200 station

A SIMATIC S7-1200 with a CP 1242-7 can send messages by SMS to a mobile phone or a configured S7-1200 station.

## Telecontrol by a control center

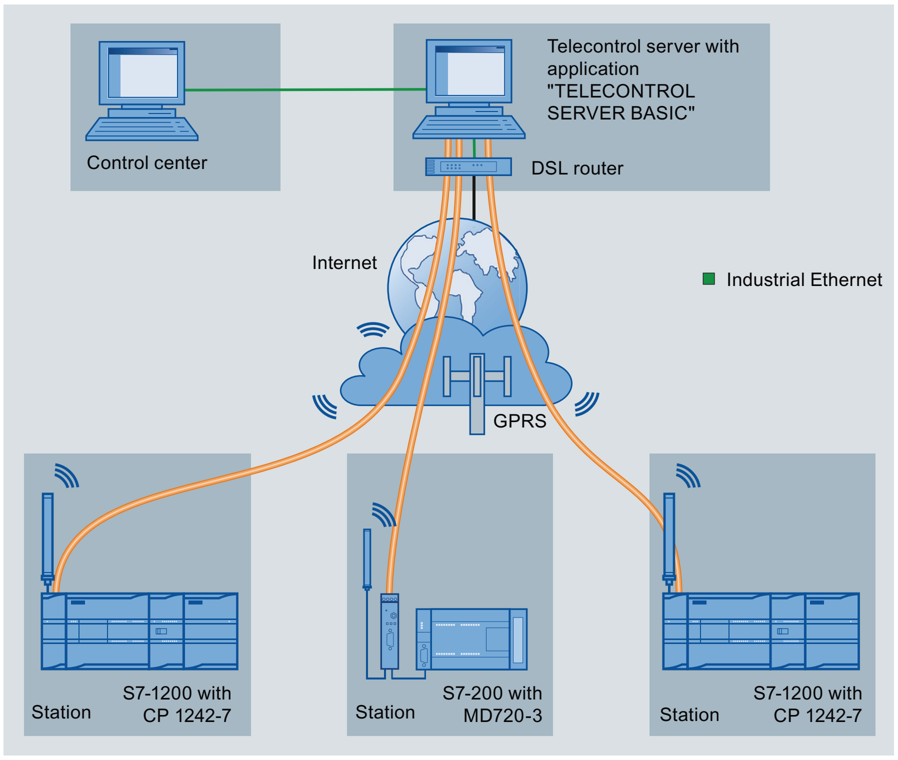


Figure 12-4 Communication between S7-1200 stations and a control center

In telecontrol applications, SIMATIC S7-1200 stations with a CP 1242-7 communicate with a control center via the GSM network and the Internet. The "TELECONTROL SERVER BASIC" (TCSB) application is installed on the telecontrol server in the master station. This results in the following use cases:

- Telecontrol communication between station and control center

  In this use case, data from the field is sent by the stations to the telecontrol server in the master station via the GSM network and Internet. The telecontrol server is used to monitor remote stations.

- Communication between a station and a control room with OPC client

  As in the first case, the stations communicate with the telecontrol server. Using its integrated OPC server, the telecontrol server exchanges data with the OPC client of the control room.

  The OPC client and telecontrol server can be located on a single computer, for example when TCSB is installed on a control center computer with WinCC.

- Inter-station communication via a control center

  Inter-station communication is possible with S7 stations equipped with a CP 1242-7.

  To allow inter-station communication, the telecontrol server forwards the messages of the sending station to the receiving station.

## Direct communication between stations

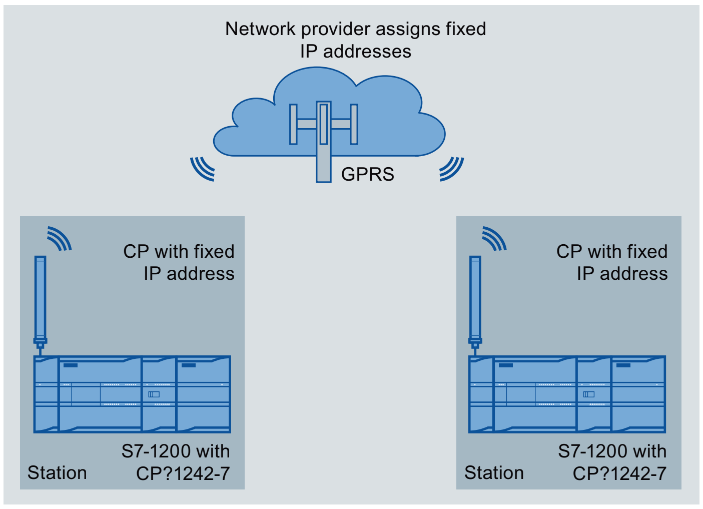


Figure 12-5 Direct communication between two S7-1200 stations

In this configuration, two SIMATIC S7-1200 stations communicate directly with each other using the CP 1242-7 via the GSM network. Each CP 1242-7 has a fixed IP address. The relevant service of the GSM network provider must allow this.

## TeleService via GPRS

In TeleService via GPRS, an engineering station on which STEP 7 is installed communicates via the GSM network and the Internet with the CP 1242-7 in the S7-1200.

Since a firewall is normally closed for connection requests from the outside, a switching station between the remote station and the engineering station is required. This switching station can be a telecontrol server or, if there is no telecontrol server in the configuration, a TeleService gateway.

## TeleService with telecontrol server

The connection runs via the telecontrol server.

- The engineering station and telecontrol server are connected via the Intranet (LAN) or Internet.
- The telecontrol server and remote station are connected via the Internet and via the GSM network.

The engineering station and telecontrol server can also be the same computer; in other words, STEP 7 and TCSB are installed on the same computer.

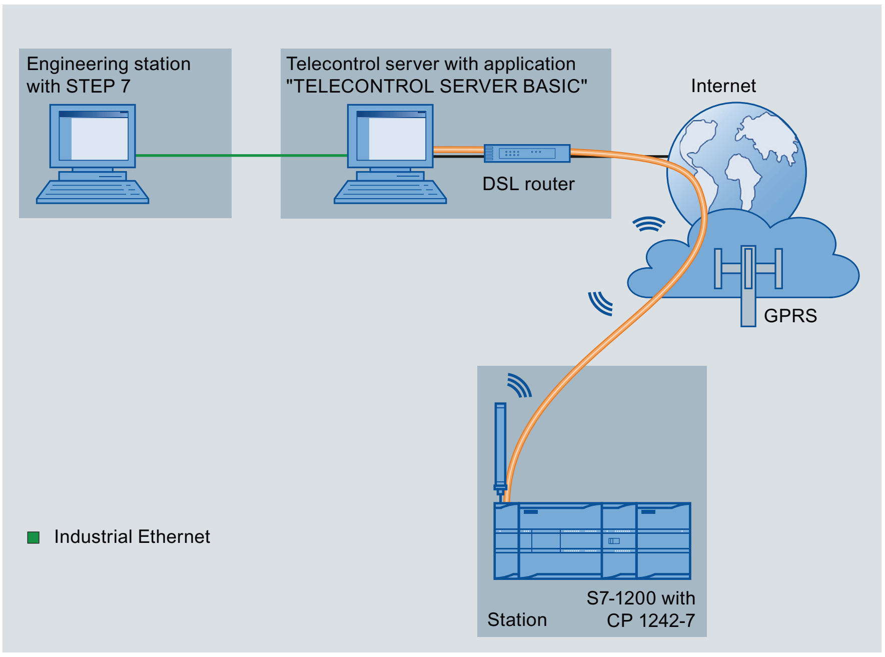

Engineering station with STEP 7
Telecontrol server with application "TELECONTROL SERVER BASIC"
Internet
DSL router
GPRS
🟩 Industrial Ethernet
Station
S7-1200 with CP 1242-7

Figure 12-6 TeleService via GPRS in a configuration with telecontrol server

## TeleService without a telecontrol server

The connection runs via the TeleService gateway.

The connection between the engineering station and the TeleService gateway can be local via a LAN or via the Internet.

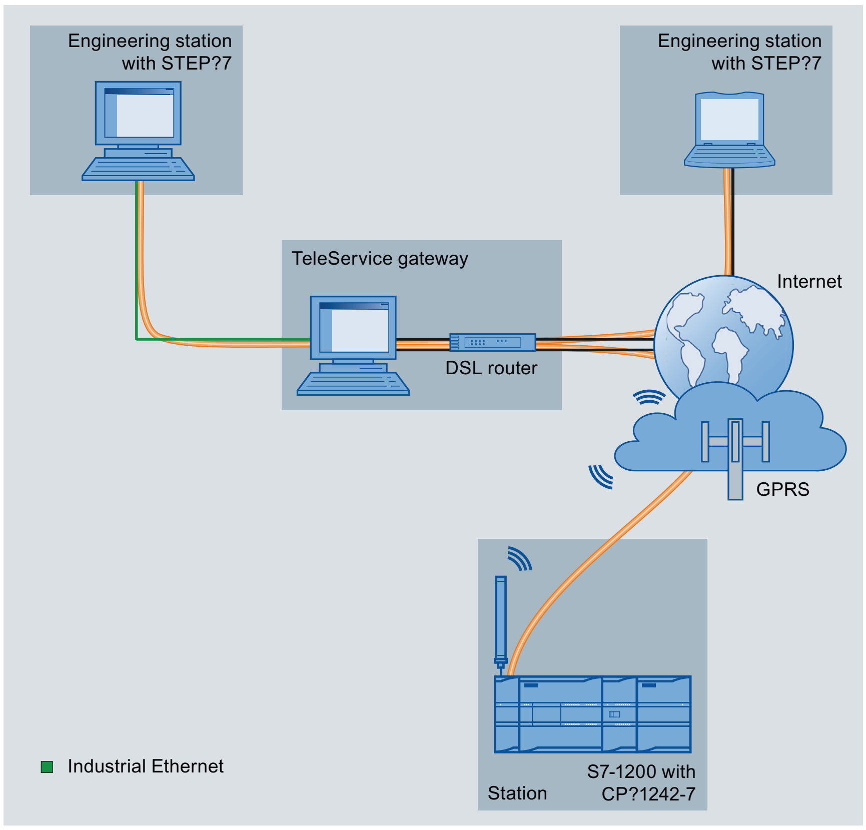
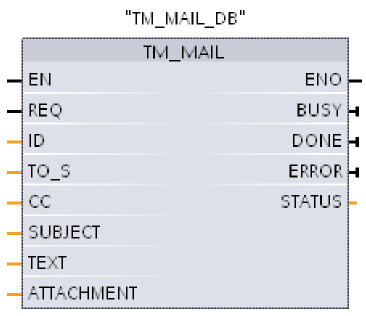


Figure 12-7 TeleService via GPRS in a configuration with TeleService gateway

# Teleservice communication (SMTP email) 13

## 13.1 TM_Mail transfer email instruction

Table 13- 1 TM_MAIL instruction

| LAD / FBD | SCL | Description |
|---|---|---|
|  "TM_MAIL_DB"<br>TM_MAIL<br>EN<br>ENO<br>REQ<br>BUSY<br>ID<br>DONE<br>TO_S<br>ERROR<br>CC<br>STATUS<br>SUBJECT<br>TEXT<br>ATTACHMENT | `"TM_MAIL_DB"(`<br>`REQ:=_bool_in_,`<br>`ID:=_int_in_,`<br>`TO_S:=_string_in_,`<br>`CC:=_string_in_,`<br>`SUBJECT:=_string_in_,`<br>`TEXT:= _string_in_,`<br>`ATTACHMENT:=_variant_in_,`<br>`BUSY=>_bool_out_,`<br>`DONE=>_bool_out_,`<br>`ERROR=>_bool_out_,`<br>`STATUS=>_word_out_);` | The TM_MAIL instruction sends an email message using the SMTP (Simple Mail Transfer Protocol) over TCP/IP via the CPU Industrial Ethernet connection. Where Ethernet-based Internet connectivity is not available, an optional Teleservice adapter can be used for connection with telephone land lines. TM_MAIL executes asynchronously and the job extends over multiple TM_MAIL calls. When you call TM_MAIL, you must assign an instance DB. **The instance DB retentive attribute must not be set**. This ensures that the instance DB is initialized in the transition of the CPU from STOP to RUN and that a new TM_MAIL operation can be triggered. |

1 STEP 7 automatically creates the instance DB when you insert the instruction.

You start sending an email with a positive edge change from 0 to 1, at input parameter REQ. The following table shows the relationship between BUSY, DONE and ERROR. You can monitor the progress of TM_MAIL execution and detect completion, by evaluating these parameters in successive calls.

The output parameters DONE, ERROR, STATUS, and SFC_STATUS are valid for only one cycle, when the state of the output parameter BUSY changes from 1 to 0. Your program logic must save temporary output state values, so you can detect state changes in subsequent program execution cycles.

Table 13- 2 Interaction of the Done, Busy and Error parameters

| DONE | BUSY | ERROR | Description |
|---|---|---|---|
| Irrelevant | 1 | Irrelevant | Job is in progress. |
| 1 | 0 | 0 | The job was completed successfully. |
| 0 | 0 | 1 | The job was terminated with an error. For the cause of the error, refer to the STATUS parameter. |
| 0 | 0 | 0 | No job in progress |

If the CPU is changed to STOP mode while TM_MAIL is active, then the communication connection to the email server is terminated. The communication connection to the email server is also lost if problems occur in CPU communication on the Industrial Ethernet bus. In these cases, the send process is suspended and the email does not reach the recipient.

CAUTION

**Modifiying user programs**

Only change the parts of your user program that directly affect the TM_MAIL calls in the following cases:

- The CPU in the STOP mode
- No email is sent (REQ and BUSY = 0)

This refers, in particular, to the deletion and replacement of program blocks, the calls to TM_MAIL, or calls to the instance DBs of TM_MAIL.

If you fail to maintain linked program blocks, then the TPC/IP communication functions can enter an undefined state. After transferring a modified program block, you must perform a CPU restart (warm) or cold start.

## Data consistency

The input parameter ADDR_MAIL_SERVER is read when the operation is started. A new value does not take effect until the current operation is complete and a new TM_MAIL operation is initiated.

In contrast, the parameters WATCH_DOG_TIME, TO_S, CC, FROM, SUBJECT, TEXT, ATTACHMENT, USERNAME and PASSWORD are read during the execution of TM_MAIL and may be changed only when the job is finished (BUSY = 0)

## Dial-up connection: Configuring the TS adapter IE parameters

You must configure the Teleservice adapter IE parameters for outgoing calls to connect with the dial-up server of your Internet Service Provider. If you set the call "on demand" attribute, then the connection is established only when an e-mail will be sent. For an analog modem connection, more time is required for the connection process (approx. a minute longer). You must include the extra time, in the WATCH_DOG_TIME value.

Table 13- 3 Data types for the parameters

| Parameter and type | | Data types | Description |
|---|---|---|---|
| REQ | IN | Bool | A low to high (positive edge) signal starts the operation. |
| ID | IN | Int | Connection identifier: See the ID parameter of the instructions TCON, TDISCON, TSEND and TRCV.<br>A number that is not used for any additional instances of this instruction in the user program must be used. |
| TO_S | IN | String | Recipient addresses: STRING data with a maximum length of 240 characters |

| Parameter and type | | Data types | Description |
|---|---|---|---|
| CC | IN | String | CC copy to recipient addresses (optional): STRING data with a maximum length of 240 characters |
| SUBJECT | IN | String | Subject name of the email: STRING data with a maximum length 240 characters. |
| TEXT | IN | String | Text message of the email (optional): STRING data with a maximum length of 240 characters.<br>If this parameter is an empty string, then the email will be sent without message text. |
| ATTACHMENT | IN | Variant | Pointer to email attachment data: Byte, word, or double word data with a maximum length of 65534 bytes.<br>If no value is assigned, then the email sent without an attachment. |
| DONE | OUT | Bool | • 0 - Job not yet started or still executing.<br>• 1 - Job was executed error-free. |
| BUSY | OUT | Bool | • 0 - No operation in progress<br>• 1- Operation in progress |
| ERROR | OUT | Bool | The ERROR bit =1 for one scan, after the last request was terminated with an error. The error code value at the STATUS output is valid only during the single scan where ERROR = 1. |
| STATUS | OUT | Word | Return value or error information of the TM_MAIL instruction. |
| ADDR_MAIL_SERVER | $^1$ Static | DWord | IP address of the mail server: You must assign each IP address fragment as an octet of two 4-bit hexadecimal characters. If the IP address fragment = decimal value 10 which equals hexadecimal value A, then you must enter "0A" for that octet.<br>For example: IP address = 192.168.0.10<br>ADDR_MAIL_SERVER = DW#16#C0A8000A, where:<br>• 192 = 16#C0,<br>• 168 =16#A8<br>• 0 = 16#00<br>• 10 = 16#0A |
| WATCH_DOG_TIME | $^1$ Static | Time | The maximum time allowed for TM_MAIL to establish a server connection. If this time is exceeded, then TM_MAIL execution ends with an error.<br>The actual time delay until TM_MAIL ends and the error is issued may exceed the WATCH_DOG_TIME, because of the additional time required for the disconnect operation.<br>At first you should set a time of 2 minutes. This time can be much smaller for an ISDN phone connection. |
| USERNAME | $^1$ Static | String | Mail account user name: STRING data with a maximum length 180 characters. |
| PASSWORD | $^1$ Static | String | Mail server password: STRING data with a maximum length 180 characters. |

| Parameter and type | | Data types | Description |
|---|---|---|---|
| FROM | [1] Static | String | Sender address: STRING with a maximum length of 240 characters |
| SFC_STATUS | [1] Static | Word | Execution condition code of the called communication blocks |

[1] The values of these parameters are not modified at every call of TM_MAIL. The values are assigned in the TM_MAIL instance data block and are only referenced once, on the first call of TM_MAIL.

## SMTP authentication

TM_MAIL supports the SMTP AUTH LOGIN authentication method. For information on this authentication method, please refer to the manual of the mail server or the website of your internet service provider.

The AUTH LOGIN authentication method uses the TM_MAIL USERNAME and PASSWORD parameters to connect with the mail server. The user name and password must be previously set up on an email account at an email server.

If no value is assigned for the USERNAME parameter, then the AUTH LOGIN authentication method is not used and the email is sent without authentication.

## TO_S:, CC:, and FROM: parameters

The parameters TO_S:, CC: and FROM: are strings, as shown in the following examples:

TO_S: <wenna@mydomain.com>, <ruby@mydomain.com>,

CC: <admin@mydomain.com>, <judy@mydomain.com>,

FROM: <admin@mydomain.com>

The following rules must be used when entering these character strings:

- The characters "TO_S:", "CC:" and "FROM:" must be entered, including the colon character.
- A space character and an opening angle bracket "<" must precede each address. For example, there must be a space character between "TO_S:" and <email address>.
- A closing angle bracket ">" must be entered after each address.
- A comma character "," must be entered after each email address for the TO_S: and CC: addresses. For example, the comma after the single email address is required in "TO_S: <email address>,".
- Only one email address may be used for the FROM: entry, with no comma at the end.

Because of run-time mode and memory usage, a syntax check is not performed on the TM_MAIL TO_S:, CC: and FROM: data. If the format rules above are not followed exactly. The SMTP email server transaction will fail.

## STATUS and SFC_STATUS parameters

The execution condition codes returned by TM_MAIL can be classified as follows:

- W#16#0000: Operation of TM_MAIL was completed successfully
- W#16#7xxx: Status of TM_MAIL operation
- W#16#8xxx: An error in an internal call to a communication device or the mail server

The following table shows the execution condition codes of TM_MAIL with the exception of the error codes from internally called communication modules.

Note

**Email server requirements**

TM_MAIL can only communicate with an email server using SMTP via port 25. The assigned port number cannot be changed.

Most IT departments and external email servers now block port 25 to prevent a PC infected with a virus from becoming a rogue email generator.

You can connect to an internal mail server via SMTP and let the internal server manage the current security enhancements that are required to relay email through the Internet to an external mail server.

## Internal email server configuration example

If you use Microsoft Exchange as an internal mail server, then you can configure the server to allow SMTP access from the IP address assigned the S7-1200 PLC. Configure the Exchange management console: Server configuration > Hub transport > Receive connectors > IP relay. On the Network tab, there is a box named "Receive mail from remote servers that have these IP addresses". This is where you put the IP address of the PLC that is executing the TM_MAIL instruction. No authentication is required for this type of connection with an internal Microsoft Exchange server.

## Email server configuration

TM_MAIL can only use an email server that allows port 25 communication, SMTP, and AUTH LOGIN authentication (optional).

Configure a compatible email server account to accept remote SMTP log in. Then edit the instance DB for TM_MAIL to put in the TM_MAIL USERNAME and PASSWORD character strings that are used to authenticate the connection with your email account.

Table 13- 4 Condition codes

| STATUS (W#16#...): | SFC_STATUS (W#16#...): | Description |
|---|---|---|
| 0000 | - | The TM_MAIL operation completed without error. This zero STATUS code does not guarantee that an email was actually sent (See the first item in the note following this table). |
| 7001 | - | TM_MAIL is active (BUSY = 1). |
| 7002 | 7002 | TM_MAIL is active (BUSY = 1). |
| 8xxx | xxxx | The TM_MAIL operation was completed with an error in the internal communication instruction calls. For more information about the SFC_STATUS parameter, see the descriptions of the STATUS parameter of the underlying PROFINET open user communication instructions. |
| 8010 | xxxx | Failed to connect: For more information about the SFC_STATUS parameter, see the STATUS parameter of the TCON instruction. |
| 8011 | xxxx | Error sending data: For more information about SFC_STATUS parameter ,see the STATUS parameter of the TSEND instruction. |
| 8012 | xxxx | Error while receiving data: For more information about the SFC_STATUS parameter, see the STATUS parameter descriptions of the TRCV instruction. |
| 8013 | xxxx | Failed to connect: For more information for evaluating the SFC_STATUS parameter, see the STATUS parameter descriptions of the TCON and TDISCON instructions. |
| 8014 | - | Failed to connect: You may have entered an incorrect mail server IP address (ADDR_MAIL_SERVER) or too little time (WATCH_DOG_TIME) for the connection. It is also possible that the CPU has no connection to the network or the CPU configuration is incorrect. |
| 82xx, 84xx, 85xx | - | The error message comes from the mail server and corresponds to error number "8" of the SMTP protocol. See the second item in the note following this table. |
| 8450 | - | Operation does not run: Mailbox is not available; try again later. |
| 8451 | - | Operation aborted: Local error in processing, .try again later |
| 8500 | - | Command syntax error: The cause may be that the email server does not support the LOGIN authentication process. Check the parameters of TM_MAIL. Try to send an email without authentication. Try replacing the parameter USERNAME with an empty string. |
| 8501 | - | Syntax error: Incorrect parameter or argument; you may have typed an incorrect address in the TO_S or CC parameters. |
| 8502 | - | Command is unknown or not implemented: Check your entries, especially the parameter FROM. Perhaps this is incomplete and you have omitted the "@" or "." characters. |
| 8535 | - | SMTP authentication is incomplete. You may have entered an incorrect username or password. |
| 8550 | - | The mail server cannot be reached, or you have no access rights. You may have entered an incorrect username or password or of your mail server does not support log in access. Another cause of this error could be an erroneous entry of the domain name after the "@" character in the TO_S or CC parameters. |
| 8552 | - | Operation aborted: Exceeded the allocated memory size; try again later. |
| 8554 | - | Transmission failed: Try again later. |

**Note**

**Possible unreported email transmission errors**

- Incorrect entry of a recipient address does not generate a STATUS error for TM_MAIL. In this case, there is no guarantee that additional recipients (with correct email addresses), will receive the email.
- More information on SMTP error codes can be found on the internet or in the error documentation for the mail server. You can also read the last error message from the mail server. The error message in stored in buffer1parameter of the instance DB for TM_MAIL.

# Online and diagnostic tools 14

## 14.1 Status LEDs

The CPU and the I/O modules use LEDs to provide information about either the operational status of the module or the I/O.

### Status LEDs on a CPU

The CPU provides the following status indicators:

- STOP/RUN
  - Solid yellow indicates STOP mode
  - Solid green indicates RUN mode
  - Flashing (alternating green and yellow) indicates that the CPU is in STARTUP mode
- ERROR
  - Flashing red indicates an error, such as an internal error in the CPU, a error with the memory card, or a configuration error (mismatched modules)
  - Solid red indicates defective hardware
- MAINT (Maintenance) flashes whenever you insert a memory card. The CPU then changes to STOP mode. After the CPU has changed to STOP mode, perform one of the following functions to initiate the evaluation of the memory card:
  - Change the CPU to RUN mode
  - Perform a memory reset (MRES)
  - Power-cycle the CPU

You can also use the LED instruction (Page 298) to determine the status of the LEDs.

Table 14- 1 Status LEDs for a CPU

| Description | STOP/RUN<br>Yellow / Green | ERROR<br>Red | MAINT<br>Yellow |
|---|---|---|---|
| Power is off | Off | Off | Off |
| Startup, self-test, or firmware update | Flashing<br>(alternating yellow and green) | - | Off |
| Stop mode | On (yellow) | - | - |
| Run mode | On (green) | - | - |
| Remove the memory card | On (yellow) | - | Flashing |
| Error | On (either yellow or green) | Flashing | - |
| Maintenance requested | On (either yellow or green) | - | On |

| Description | STOP/RUN<br>Yellow / Green | ERROR<br>Red | MAINT<br>Yellow |
|---|---|---|---|
| Defective hardware | On (yellow) | On | Off |
| LED test or defective CPU firmware | Flashing<br>(alternating yellow and green) | Flashing | Flashing |

The CPU also provides two LEDs that indicate the status of the PROFINET communications. Open the bottom terminal block cover to view the PROFINET LEDs.

- Link (green) turns on to indicate a successful connection
- Rx/Tx (yellow) turns on to indicate transmission activity

The CPU and each digital signal module (SM) provide an I/O Channel LED for each of the digital inputs and outputs. The I/O Channel (green) turns on or off to indicate the state of the individual input or output.

## Status LEDs on an SM

In addition, each digital SM provides a DIAG LED that indicates the status of the module:

- Green indicates that the module is operational
- Red indicates that the module is defective or non-operational

Each analog SM provides an I/O Channel LED for each of the analog inputs and outputs.

- Green indicates that the channel has been configured and is active
- Red indicates an error condition of the individual analog input or output

In addition, each analog SM provides a DIAG LED that indicates the status of the module:

- Green indicates that the module is operational
- Red indicates that the module is defective or non-operational

The SM detects the presence or absence of power to the module (field-side power, if required).

Table 14- 2 Status LEDs for a signal module (SM)

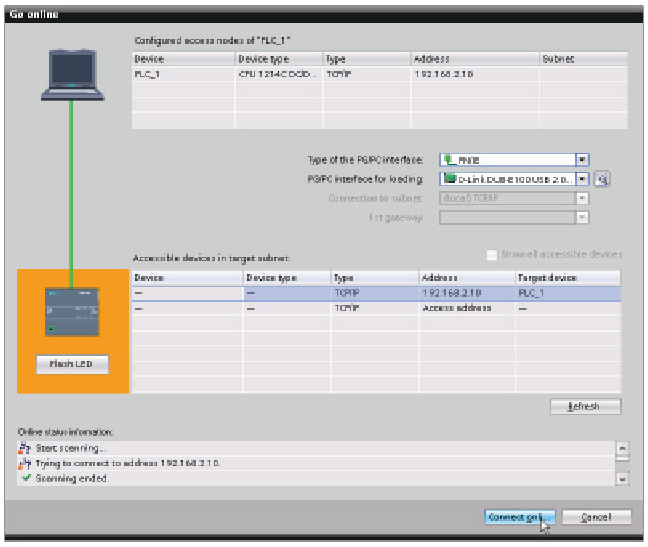
| Description | DIAG<br>(Red / Green) | I/O Channel<br>(Red / Green) |
|---|---|---|
| Field-side power is off | Flashing red | Flashing red |
| Not configured or update in progress | Flashing green | Off |
| Module configured with no errors | On (green) | On (green) |
| Error condition | Flashing red | - |
| I/O error (with diagnostics enabled) | - | Flashing red |
| I/O error (with diagnostics disabled) | - | On (green) |

## 14.2 Going online and connecting to a CPU

An online connection between the programming device and CPU is required for loading programs and project engineering data as well as for activities such as the following:

- Testing user programs
- Displaying and changing the operating mode of the CPU (Page 679)
- Displaying and setting the date and time of day of the CPU (Page 678)
- Displaying the module information
- Comparing and synchronizing (Page 681) offline to online program blocks
- Uploading and downloading program blocks
- Displaying diagnostics and the diagnostics buffer (Page 680)
- Using a watch table (Page 685) to test the user program by monitoring and modifying values
- Using a force table to force values in the CPU (Page 688)

To establish an online connection to a configured CPU, click the CPU from the Project Navigation tree and click the "Go online" button from the Project View:



Configured access nodes of "PLC_1"

| Device | Device type | Type | Address | Subnet |
|---|---|---|---|---|
| PLC_1 | CPU 1214C DC/D... | TCP/IP | 192.168.2.10 | |

Type of the PG/PC interface: PN/IE
PG/PC interface for loading: D-Link DUB-E100 USB 2.0...
Connection to subnet: (local) TCP/IP
1st gateway:

Accessible devices in target subnet:

Show all accessible devices

| Device | Device type | Type | Address | Target device |
|---|---|---|---|---|
| — | — | TCP/IP | 192.168.2.10 | PLC_1 |
| — | — | TCP/IP | Access address | — |

Online status information:
Start scanning...
Trying to connect to address 192.168.2.10.
Scanning ended.

If this is the first time to go online with this CPU, you must select the type of PG/PC interface and the specific PG/PC interface from the Go Online dialog before establishing an online connection to a CPU found on that interface.

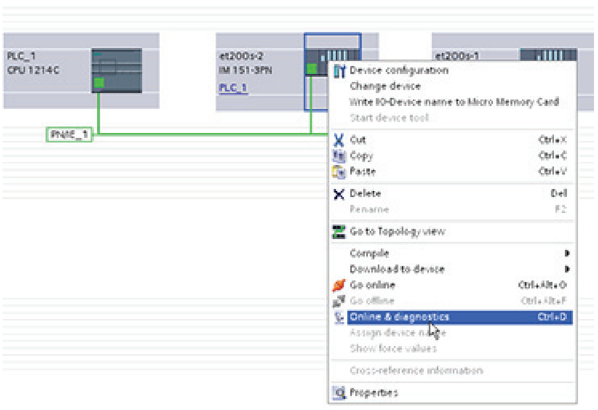
Your programming device is now connected to the CPU. The orange color frames indicate an online connection. You can now use the Online & diagnostics tools from the Project tree and the Online tools task card.

## 14.3 Assigning a name to a PROFINET IO device online

The devices on your PROFINET network must have an assigned name before you can connect with the CPU. Use the "Devices & networks" editor to assign names to your PROFINET devices if the devices have not already been assigned a name or if the name of the device is to be changed.

For each PROFINET IO device, you must assign the same name to that device in both the STEP 7 project and, using the "Online & diagnostics" tool, to the PROFINET IO device configuration memory (for example, an ET200 S interface module configuration memory). If a name is missing or does not match in either location, the PROFINET IO data exchange mode will not run.

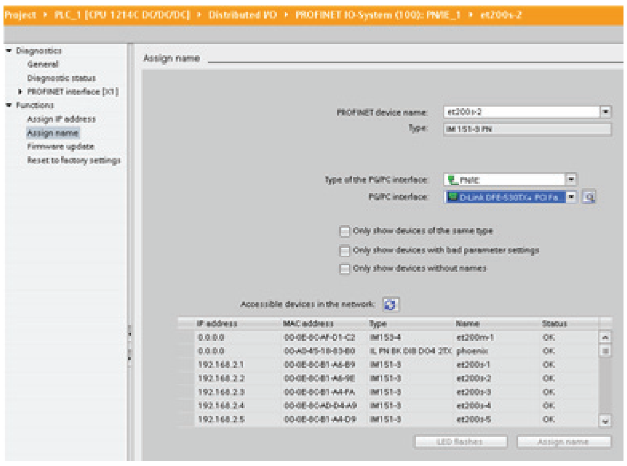
1. In the "Devices & networks" editor, right-click on the required PROFINET IO device, and select "Online & diagnostics".



2. In the "Online & diagnostics" dialog, make the following menu selections:

- "Functions"
- "Assign name"

Click the "Accessible devices in the network" icon to display all of the PROFINET IO devices on the network.



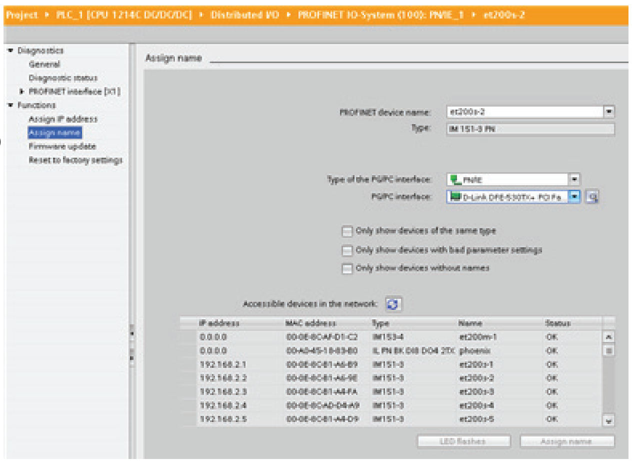
PROFINET device name: et200s-2
Type: IM 151-3 PN
Type of the PG/PC interface: PN/IE
PG/PC interface: D-Link DFE-530TX+ PCI Fa

☐ Only show devices of the same type
☐ Only show devices with bad parameter settings
☐ Only show devices without names

Accessible devices in the network:

| IP address | MAC address | Type | Name | Status |
|---|---|---|---|---|
| 0.0.0.0 | 00-0E-8C-AF-D1-C2 | IM153-4 | et200m-1 | OK |
| 0.0.0.0 | 00-A0-45-18-83-B0 | IL PN BK DI8 DO4 2TX | phoenix | OK |
| 192.168.2.1 | 00-0E-8C-B1-A6-B9 | IM151-3 | et200s-1 | OK |
| 192.168.2.2 | 00-0E-8C-B1-A6-9E | IM151-3 | et200s-2 | OK |
| 192.168.2.3 | 00-0E-8C-B1-A4-FA | IM151-3 | et200s-3 | OK |
| 192.168.2.4 | 00-0E-8C-AD-D4-A9 | IM151-3 | et200s-4 | OK |
| 192.168.2.5 | 00-0E-8C-B1-A4-D9 | IM151-3 | et200s-5 | OK |

3. In the list that is displayed, click the required PROFINET IO device, and click the "Assign name" button to write the name to the PROFINET IO device configuration memory.



PROFINET device name: et200s-2
Type: IM 151-3 PN
Type of the PG/PC interface: PN/IE
PG/PC interface: D-Link DFE-530TX+ PCI Fa

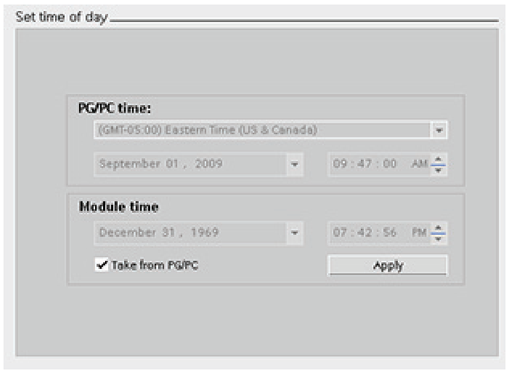
☐ Only show devices of the same type
☐ Only show devices with bad parameter settings
☐ Only show devices without names

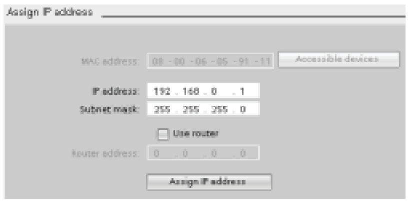
Accessible devices in the network:

| IP address | MAC address | Type | Name | Status |
|---|---|---|---|---|
| 0.0.0.0 | 00-0E-8C-AF-D1-C2 | IM153-4 | et200m-1 | OK |
| 0.0.0.0 | 00-A0-45-18-83-B0 | IL PN BK DI8 DO4 2TX | phoenix | OK |
| 192.168.2.1 | 00-0E-8C-B1-A6-B9 | IM151-3 | et200s-1 | OK |
| 192.168.2.2 | 00-0E-8C-B1-A6-9E | IM151-3 | et200s-2 | OK |
| 192.168.2.3 | 00-0E-8C-B1-A4-FA | IM151-3 | et200s-3 | OK |
| 192.168.2.4 | 00-0E-8C-AD-D4-A9 | IM151-3 | et200s-4 | OK |
| 192.168.2.5 | 00-0E-8C-B1-A4-D9 | IM151-3 | et200s-5 | OK |

## 14.4 Setting the IP address and time of day

You can set the IP address (Page 134) and time of day in the online CPU. After accessing "Online & diagnostics" from the Project tree for an online CPU, you can display or change the IP address. You can also display or set the time and date parameters of the online CPU.


Set time of day
PG/PC time:
(GMT-05:00) Eastern Time (US & Canada)
September 01 , 2009 09 : 47 : 00 AM
Module time
December 31 , 1969 07 : 42 : 56 PM
✓ Take from PG/PC


Assign IP address
MAC address: 08 - 00 - 06 - 05 - 91 - 11
IP address: 192 . 168 . 0 . 1
Subnet mask: 255 . 255 . 255 . 0
☐ Use router
Router address: 0 . 0 . 0 . 0

---

**Note**

This feature is available only for a CPU that either has only a MAC address (has not yet been assigned an IP address) or has been reset to factory settings.

---

## 14.5 Resetting to factory settings

You can reset an S7-1200 to its original factory settings under the following conditions:

- No memory card is inserted in the CPU.
- The CPU has an online connection.
- The CPU is in STOP mode.

---

**Note**

If the CPU is in RUN mode and you start the reset operation, you can place it in STOP mode after acknowledging a confirmation prompt.

---

### Procedure

To reset a CPU to its factory settings, follow these steps:

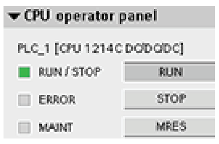
1. Open the Online and Diagnostics view of the CPU.
2. Select "Reset to factory settings" from the "Functions" folder.
3. Select the "Keep IP address" check box if you want to retain the IP address or the "Reset IP address" check box if you want to delete the IP address.
4. Click the "Reset" button.
5. Acknowledge the confirmation prompt with "OK".

### Result

The module is switched to STOP mode if necessary, and it is reset to the factory settings:

- The work memory and internal load memory and all operand areas are cleared.
- All parameters are reset to their defaults.
- The diagnostics buffer is cleared.
- The time of day is reset.
- The IP address is retained or deleted based on the setting you made. (The MAC address is fixed and is never changed.)

## 14.6 CPU operator panel for the online CPU



The "CPU operator panel" displays the operating mode (STOP or RUN) of the online CPU. The panel also shows whether the CPU has an error or if values are being forced.

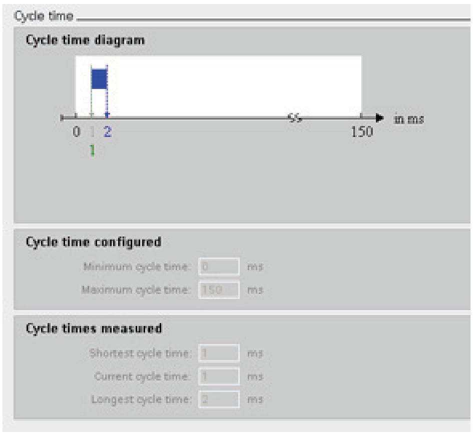
Use the CPU operating panel of the Online Tools task card to change the operating mode of an online CPU. The Online Tools task card is accessible whenever the CPU is online.

## 14.7 Monitoring the cycle time and memory usage

You can monitor the cycle time and memory usage of an online CPU.

After connecting to the online CPU, open the Online tools task card to view the following measurements:

- Cycle time
- Memory usage

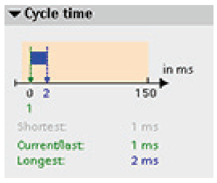


Cycle time configured

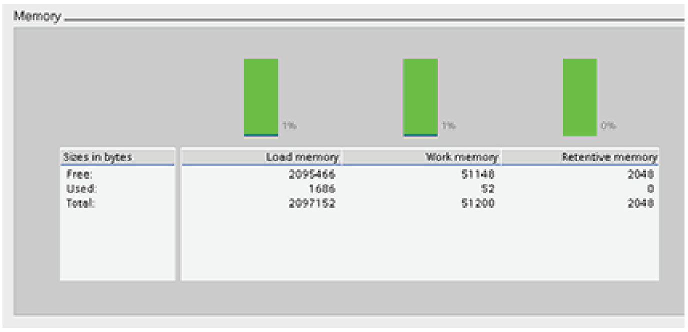
| | | |
|---|---|---|
| Minimum cycle time: | 0 | ms |
| Maximum cycle time: | 150 | ms |

Cycle times measured

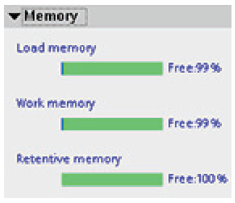
| | | |
|---|---|---|
| Shortest cycle time: | 1 | ms |
| Current cycle time: | 1 | ms |
| Longest cycle time: | 2 | ms |



| | |
|---|---|
| Shortest: | 1 ms |
| Current/last: | 1 ms |
| Longest: | 2 ms |



| Sizes in bytes | Load memory | Work memory | Retentive memory |
|---|---|---|---|
| Free: | 2095466 | 51148 | 2048 |
| Used: | 1686 | 52 | 0 |
| Total: | 2097152 | 51200 | 2048 |

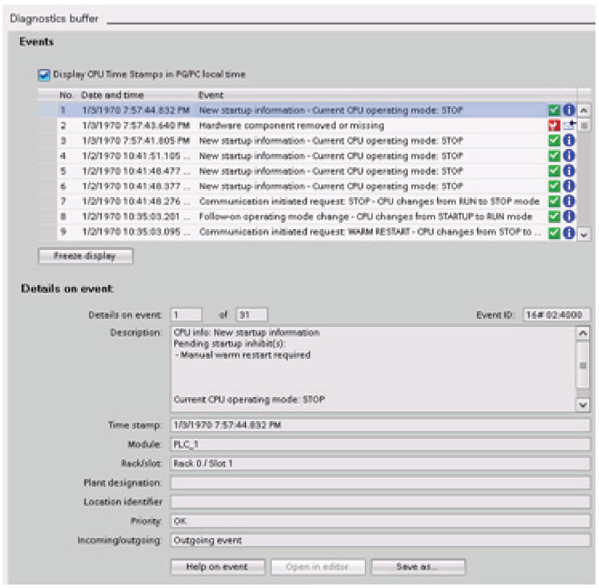


| | |
|---|---|
| Load memory | Free:99 % |
| Work memory | Free:99 % |
| Retentive memory | Free:100 % |

## 14.8 Displaying diagnostic events in the CPU

Use the diagnostics buffer to review the recent activity in the CPU. The diagnostics buffer is accessible from "Online & Diagnostics" for an online CPU in the Project tree. It contains the following entries:

- Diagnostic events
- Changes in the CPU operating mode (transitions to STOP or RUN mode)


Diagnostics buffer

**Events**

☑ Display CPU Time Stamps in PG/PC local time

| No. | Date and time | Event |
|---|---|---|
| 1 | 1/3/1970 7:57:44.832 PM | New startup information - Current CPU operating mode: STOP |
| 2 | 1/3/1970 7:57:43.640 PM | Hardware component removed or missing |
| 3 | 1/3/1970 7:57:41.805 PM | New startup information - Current CPU operating mode: STOP |
| 4 | 1/2/1970 10:41:51.105 ... | New startup information - Current CPU operating mode: STOP |
| 5 | 1/2/1970 10:41:48.477 ... | New startup information - Current CPU operating mode: STOP |
| 6 | 1/2/1970 10:41:48.377 ... | New startup information - Current CPU operating mode: STOP |
| 7 | 1/2/1970 10:41:48.276 ... | Communication initiated request: STOP - CPU changes from RUN to STOP mode |
| 8 | 1/2/1970 10:35:03.201 ... | Follow-on operating mode change - CPU changes from STARTUP to RUN mode |
| 9 | 1/2/1970 10:35:03.095 ... | Communication initiated request: WARM RESTART - CPU changes from STOP to ... |

Freeze display

**Details on event**

Details on event: 1 of 31 Event ID: 16# 02:4000

Description: CPU info: New startup information
Pending startup inhibit(s):
- Manual warm restart required

Current CPU operating mode: STOP

Time stamp: 1/3/1970 7:57:44.832 PM
Module: PLC_1
Rack/slot: Rack 0 / Slot 1
Plant designation:
Location identifier
Priority: OK
Incoming/outgoing: Outgoing event

Help on event | Open in editor | Save as...

The first entry contains the latest event. Each entry in the diagnostic buffer contains the date and time the event was logged, and a description.

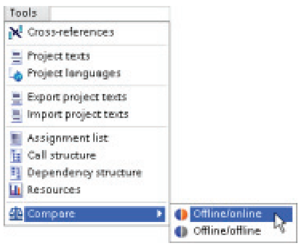
The maximum number of entries is dependent on the CPU. A maximum of 50 entries is supported.

Only the 10 most recent events in the diagnostic buffer are stored permanently. Resetting the CPU to the factory settings resets the diagnostic buffer by deleting the entries.

You can also use the GET_DIAG instruction (Page 302) to collect the diagnostic information.

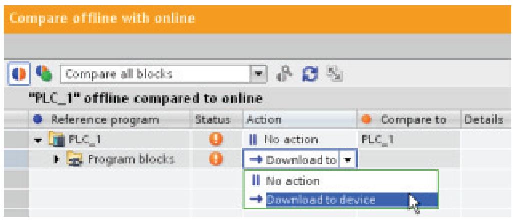
## 14.9 Comparing offline and online CPUs

You can compare the code blocks in an online CPU with the code blocks in your project. If the code blocks of your project do not match the code blocks of the online CPU, the "Compare" editor allows you to synchronize your project with the online CPU by downloading the code blocks of your project to the CPU, or by deleting blocks from the project that do not exist in the online CPU.



Select the CPU in your project.

Use the "Compare Offline/online" command to open the "Compare" editor. (Access the command either from the "Tools" menu or by right-clicking the CPU in your project.)


Compare offline with online

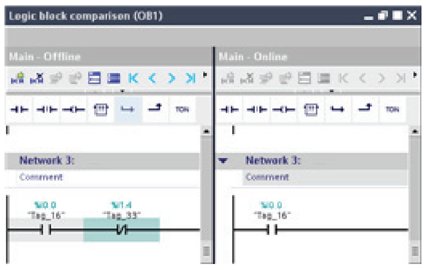
Compare all blocks

"PLC_1" offline compared to online

| Reference program | Status | Action | Compare to | Details |
|---|---|---|---|---|
| PLC_1 | ! | No action | PLC_1 | |
| Program blocks | ! | Download to | | |

Click in the "Action" column for an object to select whether to delete the object, take no action, or download the object to the device.

Click the "Synchronize" button to load the code blocks.



Right-click an object in the "Compare to" column and select "Start detailed comparison" button to show the code blocks side-by-side.

The detailed comparison highlights the differences between the code blocks of online CPU and the code blocks of the CPU in your project.

## 14.10 Monitoring and modifying values in the CPU

STEP 7 provides online tools for monitoring the CPU:

- You can display or monitor the current values of the tags. The monitoring function does not change the program sequence. It presents you with information about the program sequence and the data of the program in the CPU.
- You can also use other functions to control the sequence and the data of the user program:
  - You can modify the value for the tags in the online CPU to see how the user program responds.
  - You can force a peripheral output (such as Q0.1:P or "Start":P) to a specific value.
  - You can enable outputs in STOP mode.

**Note**

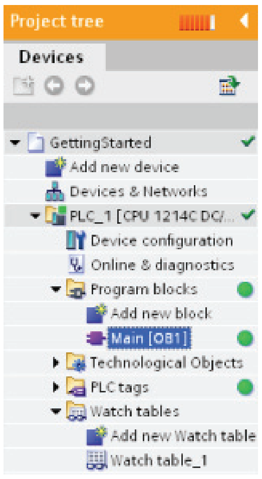
Always exercise caution when using control functions. These functions can seriously influence the execution of the user/system program.

Table 14- 3 Online capabilities of the STEP 7 editors

| Editor | Monitor | Modify | Force |
|---|---|---|---|
| Watch table | Yes | Yes | No |
| Force table | Yes | No | Yes |
| Program editor | Yes | Yes | No |
| Tag table | Yes | No | No |
| DB editor | Yes | No | No |

### 14.10.1 Going online to monitor the values in the CPU

To monitor the tags, you must have an online connection to the CPU. Simply click the "Go online" button in the toolbar.



Project tree

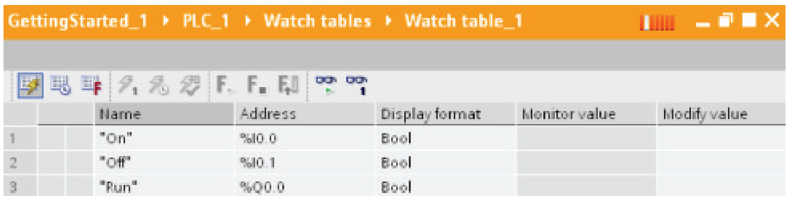
Devices

- GettingStarted ✓
  - Add new device
  - Devices & Networks
  - PLC_1 [CPU 1214C DC/... ✓
    - Device configuration
    - Online & diagnostics
    - Program blocks ●
      - Add new block
      - Main [OB1] ●
    - Technological Objects
    - PLC tags ●
    - Watch tables
      - Add new Watch table
      - Watch table_1

When you have connected to the CPU, STEP 7 turns the headers of the work areas orange.

The project tree displays a comparison of the offline project and the online CPU. A green circle means that the CPU and the project are synchronized, meaning that both have the same configuration and user program.

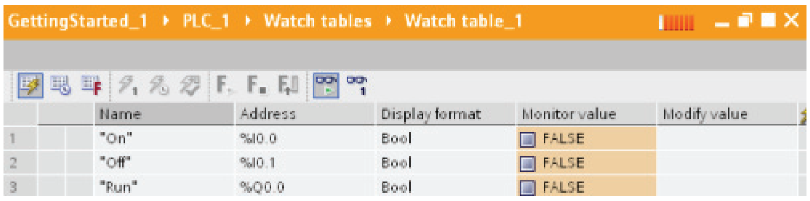
Tag tables show the tags. Watch tables can also show the tags, as well as direct addresses.



GettingStarted_1 ▸ PLC_1 ▸ Watch tables ▸ Watch table_1

| | Name | Address | Display format | Monitor value | Modify value |
|---|---|---|---|---|---|
| 1 | "On" | %I0.0 | Bool | | |
| 2 | "Off" | %I0.1 | Bool | | |
| 3 | "Run" | %Q0.0 | Bool | | |

To monitor the execution of the user program and to display the values of the tags, click the "Monitor all" button in the toolbar.



GettingStarted_1 ▸ PLC_1 ▸ Watch tables ▸ Watch table_1

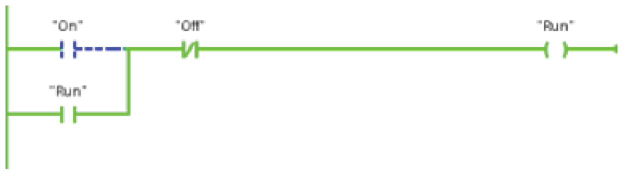
| | Name | Address | Display format | Monitor value | Modify value |
|---|---|---|---|---|---|
| 1 | "On" | %I0.0 | Bool | FALSE | |
| 2 | "Off" | %I0.1 | Bool | FALSE | |
| 3 | "Run" | %Q0.0 | Bool | FALSE | |

The "Monitor value" field shows the value for each tag.

## 14.10.2 Displaying status in the program editor

You can monitor the status of the tags in the LAD and FBD program editors. Use the editor bar to display the LAD editor. The editor bar allows you to change the view between the open editors without having to open or close the editors.

In the toolbar of the program editor, click the "Monitoring on/off" button to display the status of your user program.



The network in the program editor displays power flow in green.

You can also right-click on the instruction or parameter to modify the value for the instruction.

## 14.10.3 Capturing the online values of a DB to reset the start values

You can capture the current values being monitored in an online CPU to become the start values for a global DB.

- You must have an online connection to the CPU.
- The CPU must be in RUN mode.
- You must have opened the DB in STEP 7.

Use the "Show a snapshot of the monitored values" button to capture the current values of the selected tags in the DB. You can then copy these values into the "Start value" column of the DB.

1. In the DB editor, click the "Monitor all tags" button. The "Monitor value" column displays the current data values.
2. Click the "Show a snapshot of the monitored values" button to display the current values in the "Snapshot" column.
3. Click the "Monitor all" button to stop monitoring the data in the CPU.
4. Copy a value in the "Snapshot" column for a tag.
   - Select a value to be copied.
   - Right-click the selected value to display the context menu.
   - Select the "Copy" command.
5. Paste the copied value into the corresponding "Start value" column for the tag. (Right-click the cell and select "Paste" from the context menu.)

6. Save the project to configure the copied values as the new start values for the DB.
7. Compile and download the DB to the CPU. The DB uses the new start values after the CPU goes to RUN mode.

**Note**

The values that are shown in the "Monitor value" column are always copied from the CPU. STEP 7 does not check whether all values come from the same scan cycle of the CPU.

### 14.10.4 Using a watch table to monitor and modify values in the CPU

A watch table allows you to perform monitoring and control functions on data points as the CPU executes your program. These data points can be process image (I or Q), M, DB or physical inputs (I_:P), depending on the monitor or control function. You cannot accurately monitor the physical outputs (Q_:P) because the monitor function can only display the last value written from Q memory and does not read the actual value from the physical outputs.

The monitoring function does not change the program sequence. It presents you with information about the program sequence and the data of the program in the CPU.

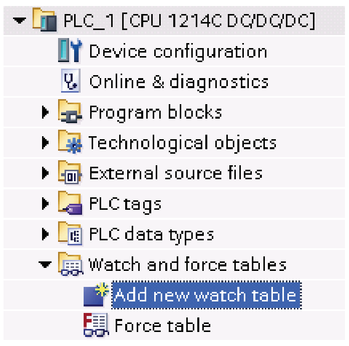
Control functions enable the user to control the sequence and the data of the program. Caution must be exercised when using control functions. These functions can seriously influence the execution of the user/system program. The three control functions are Modify, Force and Enable Outputs in STOP.

With the watch table, you can perform the following online functions:

- Monitoring the status of the tags
- Modifying values for the individual tags

You select when to monitor or modify the tag:

- Beginning of scan cycle: Reads or writes the value at the beginning of the scan cycle
- End of scan cycle: Reads or writes the value at the end of the scan cycle
- Switch to stop



To create a watch table:

1. Double-click "Add new watch table" to open a new watch table.
2. Enter the tag name to add a tag to the watch table.

The following options are available for monitoring tags:

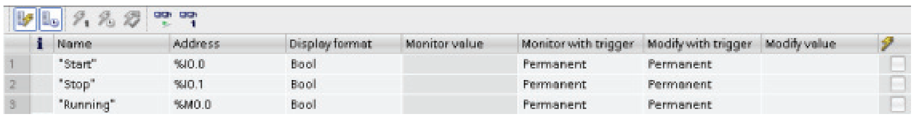
- Monitor all: This command starts the monitoring of the visible tags in the active watch table.
- Monitor now: This command starts the monitoring of the visible tags in the active watch table. The watch table monitors the tags immediately and once only.

The following options are available for modifying tags:

- "Modify to 0" sets the value of a selected address to "0".
- "Modify to 1" sets the value of a selected address to "1".
- "Modify now" immediately changes the value for the selected addresses for one scan cycle.
- "Modify with trigger" changes the values for the selected addresses.

  This function does not provide feedback to indicate that the selected addresses were actually modified. If feedback of the change is required, use the "Modify now" function.
- "Enable peripheral outputs" disables the command output disable and is available only when the CPU is in STOP mode.

To monitor the tags, you must have an online connection to the CPU.



| | i | Name | Address | Display format | Monitor value | Monitor with trigger | Modify with trigger | Modify value | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | | "Start" | %I0.0 | Bool | | Permanent | Permanent | | |
| 2 | | "Stop" | %I0.1 | Bool | | Permanent | Permanent | | |
| 3 | | "Running" | %M0.0 | Bool | | Permanent | Permanent | | |

The various functions can be selected using the buttons at the top of the watch table.

Enter the tag name to monitor and select a display format from the dropdown selection. With an online connection to the CPU, clicking the "Monitor" button displays the actual value of the data point in the "Monitor value" field.

### 14.10.4.1 Using a trigger when monitoring or modifying PLC tags

Triggering determines at what point in the scan cycle the selected address will be monitored or modified.

Table 14- 4 Types of triggers

| Trigger | Description |
|---|---|
| Permanent | Continuously collects the data |
| At scan cycle start | Permanent: Continuously collects the data at the start of the scan cycle, after the CPU reads the inputs |
| | Once: Collects the data at the start of the scan cycle, after the CPU reads the inputs |
| At scan cycle end | Permanent: Continuously collects the data at the end of the scan cycle, before the CPU writes the outputs |
| | Once: Collects the data once at the end of the scan cycle, before the CPU writes the outputs |
| At transition to STOP | Permanent: Continuously collects data when the CPU transitions to STOP |
| | Once: Collects the data once after the CPU transitions to STOP |

For modifying a PLC tag at a given trigger, select either the start or the end of cycle.

- Modifying an output: The best trigger event for modifying an output is at the end of the scan cycle, immediately before the CPU writes the outputs.

  Monitor the value of the outputs at the beginning of the scan cycle to determine what value is written to the physical outputs. Also, monitor the outputs before the CPU writes the values to the physical outputs in order to check program logic and to compare to the actual I/O behavior.

- Modifying an input: The best trigger event for modifying an input is at the start of the cycle, immediately after the CPU reads the inputs and before the user program uses the input values.

  If you are modifying inputs the start of the scan cycle, you should also monitor the value of the inputs at the end of the scan cycle to ensure that the value of the input at the end the scan cycle has not changed from the start of the scan cycle. If there is a difference in the values, your user program may be writing to an input in error.

To diagnose why the CPU might have gone to STOP, use the "Transition to STOP" trigger to capture the last process values.

### 14.10.4.2 Enabling outputs in STOP mode

The watch table allows you to write to the outputs when the CPU is in STOP mode. This functionality allows you to check the wiring of the outputs and verify that the wire connected to an output pin initiates a high or low signal to the terminal of the process device to which it is connected.

| ⚠ WARNING |
|---|
| Even though the CPU is in STOP mode, enabling a physical output can activate the process point to which it is connected. |

You can change the state of the outputs in STOP mode when the outputs are enabled. If the outputs are disabled, you cannot modify the outputs in STOP mode.

- To enable the modification of the outputs in STOP, select the "Enable peripheral outputs" option of the "Modify" command of the "Online" menu, or by right-clicking the row of the Watch table.

  You cannot enable outputs in STOP mode if you have configured distributed I/O,. An error is returned when you try to do this.

- Setting the CPU to RUN mode disables "Enable peripheral outputs" option.
- If any inputs or outputs are forced, the CPU is not allowed to enable outputs while in STOP mode. The force function must first be cancelled.

## 14.10.5 Forcing values in the CPU

### 14.10.5.1 Using the force table

A force table provides a "force" function that overwrites the value for an input or output point to a specified value for the peripheral input or peripheral output address. The CPU applies this forced value to the input process image prior to the execution of the user program and to the output process image before the outputs are written to the modules.

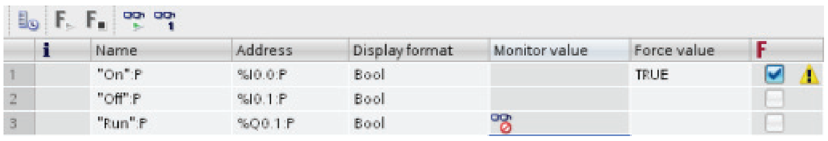
---

#### Note

The force values are stored in the CPU and not in the force table.

You cannot force an input (or "I" address) or an output (or "Q" address). However, you can force a peripheral input or peripheral output. The force table automatically appends a ":P" to the address (for example: "On":P or "Run":P).

---



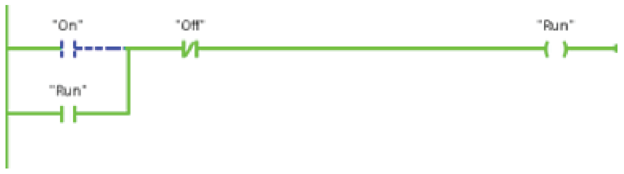
| | i | Name | Address | Display format | Monitor value | Force value | F | |
|---|---|---|---|---|---|---|---|---|
| 1 | | "On":P | %I0.0:P | Bool | | TRUE | ☑ | ⚠ |
| 2 | | "Off":P | %I0.1:P | Bool | | | ☐ | |
| 3 | | "Run":P | %Q0.1:P | Bool | | | ☐ | |

In the "Force value" cell, enter the value for the input or output to be forced. You can then use the check box in the "Force" column to enable forcing of the input or output.

Use the "Start or replace forcing" button to force the value of the tags in the force table. Click the "Stop forcing" button to reset the value of the tags.

In the force table, you can monitor the status of the forced value for an input. However, you cannot monitor the forced value of an output.

You can also view the status of the forced value in the program editor.



| NOTICE |
|---|
| When an input or output is forced in a force table, the force actions become part of the project configuration. If you close STEP 7, the forced elements remain active in the CPU program until they are cleared. To clear these forced elements, you must use STEP 7 to connect with the online CPU and then use the force table to turn off or stop the force function for those elements. |

#### 14.10.5.2 Operation of the Force function

The CPU allows you to force input and output point(s) by specifying the physical input or output address (I_:P or Q_:P) in the force table and then starting the force function.

In the program, reads of physical inputs are overwritten by the forced value. The program uses the forced value in processing. When the program writes a physical output, the output value is overwritten by the force value. The forced value appears at the physical output and is used by the process.

When an input or output is forced in the force table, the force actions become part of the user program. Even though the programming software has been closed, the force selections remain active in the operating CPU program until they are cleared by going online with the programming software and stopping the force function. Programs with forced points loaded on another CPU from a memory card will continue to force the points selected in the program.

If the CPU is executing the user program from a write-protected memory card, you cannot initiate or change the forcing of I/O from a watch table because you cannot override the values in the write-protected user program. Any attempt to force the write-protected values generates an error. If you use a memory card to transfer a user program, any forced elements on that memory card will be transferred to the CPU.

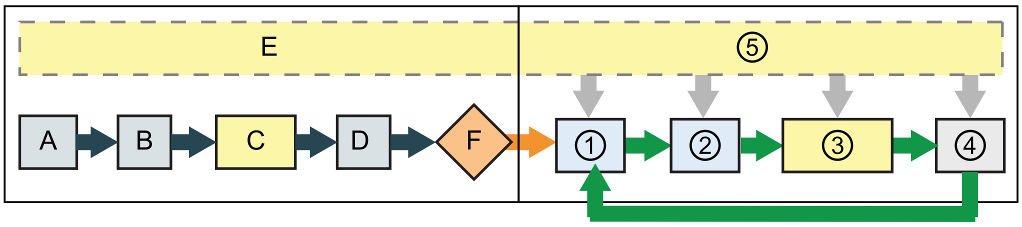
---

Note

**Digital I/O points assigned to HSC, PWM, and PTO cannot be forced**

The digital I/O points used by the high-speed counter (HSC), pulse-width modulation (PWM), and pulse-train output (PTO) devices are assigned during device configuration. When digital I/O point addresses are assigned to these devices, the values of the assigned I/O point addresses cannot be modified by the force function of the watch table.

---



| Startup | | RUN | |
|---|---|---|---|
| A | The clearing of the I memory area is not affected by the Force function. | ① | While writing Q memory to the physical outputs, the CPU applies the force value as the outputs are updated. |
| B | The initialization of the outputs values is not affected by the Force function. | ② | When reading the physical inputs, the CPU applies the force values just prior to copying the inputs into I memory. |
| C | During the execution of the startup OBs, the CPU applies the force value when the user program accesses the physical input. | ③ | During the execution of the user program (program cycle OBs), the CPU applies the force value when the user program accesses the physical input or writes the physical output. |
| D | The storing of interrupt events into the queue is not affected. | ④ | Handling of communication requests and self-test diagnostics are not affected by the Force function. |
| E | The enabling of the writing to the outputs is not affected. | ⑤ | The processing of interrupts during any part of the scan cycle is not affected. |

## 14.11 Downloading in RUN mode

The CPU supports "Download in RUN mode". This capability is intended to allow you to make small changes to a user program with minimal disturbance to the process being controlled by the program. However, implementing this capability also allows massive program changes that could be disruptive or even dangerous.

**WARNING**

When you download changes to the CPU in RUN mode, the changes immediately affect process operation. Changing the program in RUN mode can result in unexpected system operation, which could cause death or serious injury to personnel, and/or damage to equipment.

Only authorized personnel who understand the effects of RUN mode changes on system operation should perform a download in RUN mode.

The "Download in RUN mode" feature allows you to make changes to a program and download them to your CPU without switching to STOP mode:

- You can make minor changes to your current process without having to shut down (for example, change a parameter value).
- You can debug a program more quickly with this feature (for example, invert the logic for a normally open or normally closed switch).

You can make the following program block and tag changes and download them in RUN mode:

- Create, overwrite, and delete Functions (FC), Function Blocks (FB), and Tag tables.
- Create and delete Data Blocks (DB); however, DB structure changes cannot be overwritten. Initial DB values can be overwritten. You cannot download a web server DB (control or fragment) in RUN mode.
- Overwrite Organization Blocks (OB); however, you cannot create or delete OBs.

A maximum number of ten blocks can be downloaded in RUN mode at one time. If more than ten blocks are downloaded, the CPU must be placed into STOP mode.

If you download changes to a real process (as opposed to a simulated process, which you might do in the course of debugging a program), it is vital to think through the possible safety consequences to machines and machine operators before you download.

**Note**

If the CPU is in RUN mode and program changes have been made, STEP 7 always try to download in RUN first. If you do not want this to happen, you must put the CPU into STOP.

If the changes made are not supported in "Download in RUN", STEP 7 prompts the user that the CPU must go to STOP.

## 14.11.1 Prerequisites for "Download in RUN mode"

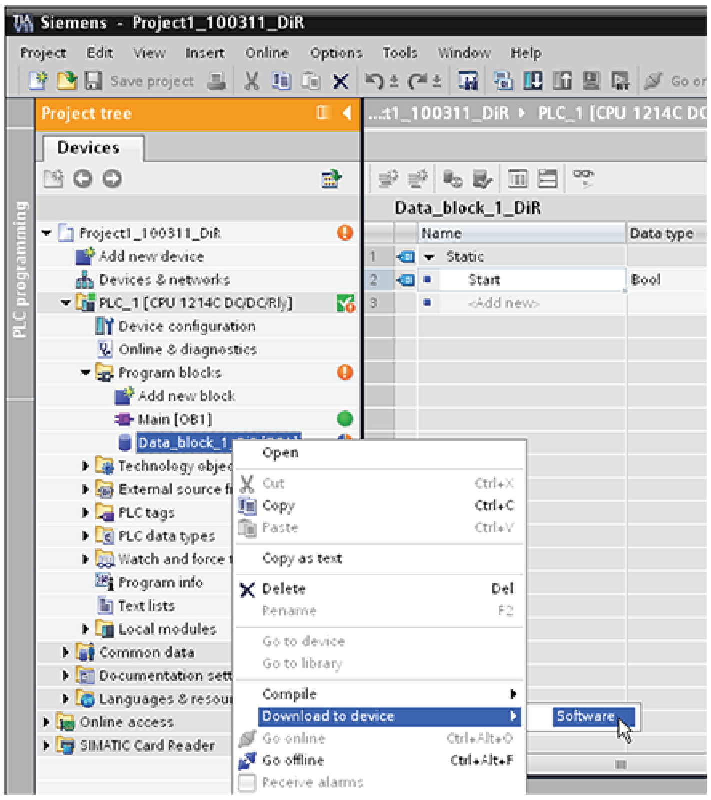
You cannot download your program changes to a CPU that is in RUN mode unless you have met these prerequisites:

- Your program must compile successfully.
- You must have successfully established communication between the programming device where you are running STEP 7 and the CPU.
- In V3.0 or later, the firmware of the target CPU must support the "Download in RUN mode" feature.

### 14.11.2 Changing your program in RUN mode

To change the program in RUN mode, your must first ensure that the CPU supports "Download in RUN mode" and that the CPU is in RUN mode:

1. To download your program in RUN mode, select one of the following methods:
   - "Download to device" command from the "Online" menu
   - "Download to device" button in the toolbar
   - In the "Project tree", right-click "Program blocks" and select the "Download to device > Software" command.

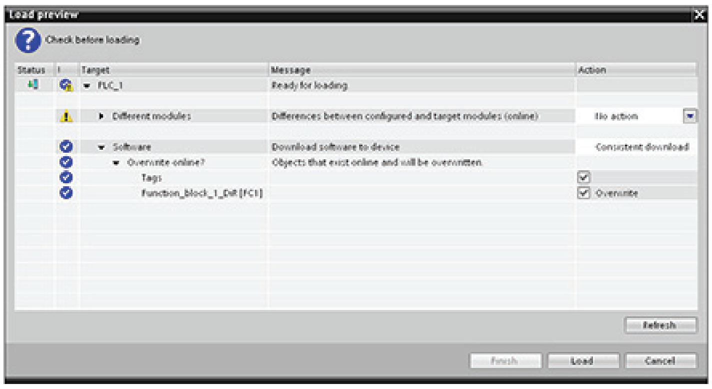


2. If the program compiles successfully, STEP 7 downloads the program to the CPU.
3. STEP 7 prompts you to load your program or cancel the operation.
4. If you click "Load", STEP 7 downloads the program to the CPU.

## 14.11.3 Downloading selected blocks

The focus is on the Program blocks folder, selection of blocks, or one single block.

1. If the user selects a single block for downloading from within the block editor, then the only option in the "Action" column is "Consistent download".
The user can expand the category line to be sure what blocks are to be loaded. In this example, a small change was made to the offline block, and no other blocks need to be loaded.

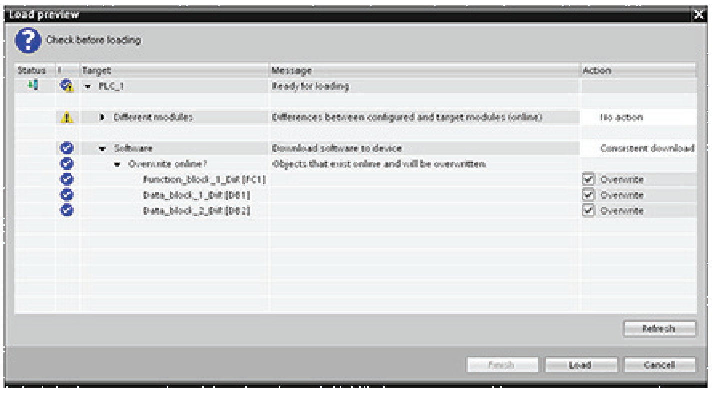


Load preview

Check before loading

| Status | ! | Target | Message | Action |
|---|---|---|---|---|
| | | FLC_1 | Ready for loading | |
| | | Different modules | Differences between configured and target modules (online) | No action |
| | | Software | Download software to device | Consistent download |
| | | Overwrite online? | Objects that exist online and will be overwritten. | |
| | | Tags | | ✓ |
| | | Function_block_1_Diff [FC1] | | ✓ Overwrite |

2. In this example, more than one block is needed for downloading.



Load preview

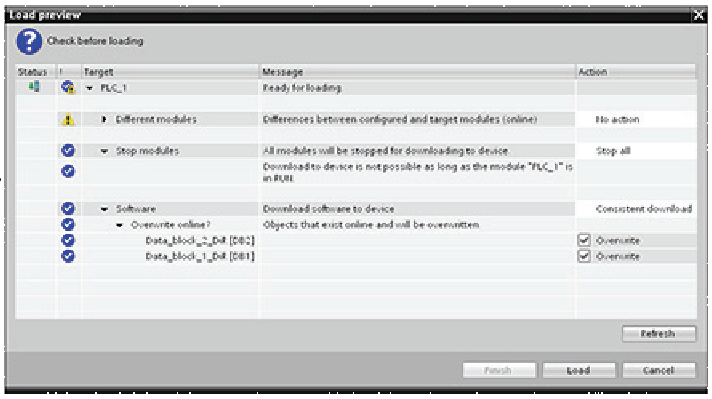
Check before loading

| Status | ! | Target | Message | Action |
|---|---|---|---|---|
| | | PLC_1 | Ready for loading | |
| | | Different modules | Differences between configured and target modules (online) | No action |
| | | Software | Download software to device | Consistent download |
| | | Overwrite online? | Objects that exist online and will be overwritten. | |
| | | Function_block_1_Diff [FC1] | | ✓ Overwrite |
| | | Data_block_1_Diff [DB1] | | ✓ Overwrite |
| | | Data_block_2_Diff [DB2] | | ✓ Overwrite |

**Note**

A maximum number of ten blocks can be downloaded in RUN mode at one time. If more than ten blocks are downloaded, the CPU must be placed into STOP mode.

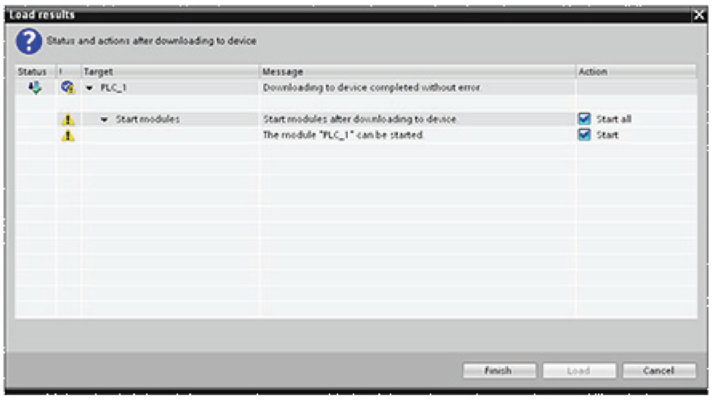
3. If the user attempts to download in RUN, but the system detects that this is not possible prior to the actual download, then the Stop modules category line appears in the dialog.


Load preview

Check before loading

| Status | ! | Target | Message | Action |
|---|---|---|---|---|
| | | ▾ PLC_1 | Ready for loading | |
| | ⚠ | ▸ Different modules | Differences between configured and target modules (online) | No action |
| | ✓ | ▾ Stop modules | All modules will be stopped for downloading to device | Stop all |
| | ✓ | | Download to device is not possible as long as the module "PLC_1" is in RUN | |
| | ✓ | ▾ Software | Download software to device | Consistent download |
| | ✓ | ▾ Overwrite online? | Objects that exist online and will be overwritten | |
| | ✓ | Data_block_2_DiF [DB2] | | ☑ Overwrite |
| | ✓ | Data_block_1_DiF [DB1] | | ☑ Overwrite |

4. Click the "Load" button, and the "Load results" dialog appears. Click the "Finish" button to complete the download.

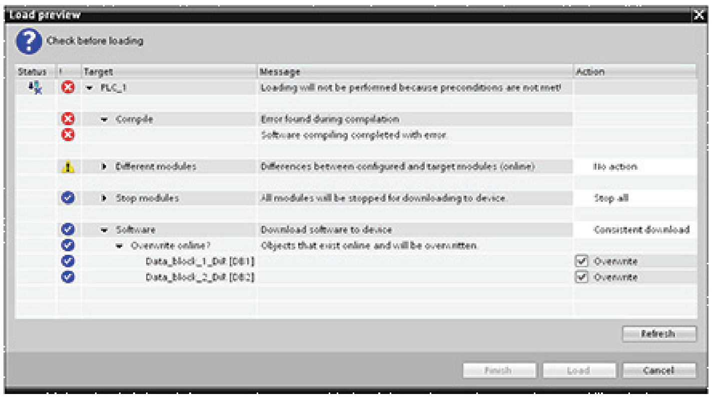

Load results

Status and actions after downloading to device

| Status | ! | Target | Message | Action |
|---|---|---|---|---|
| | | ▾ PLC_1 | Downloading to device completed without error | |
| | ⚠ | ▾ Start modules | Start modules after downloading to device | ☑ Start all |
| | ⚠ | | The module "PLC_1" can be started | ☑ Start |

## 14.11.4 Downloading a single selected block with a compile error in another block

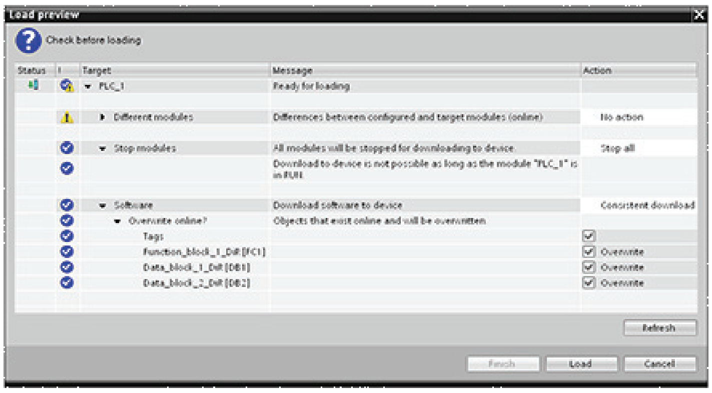
If the user attempts a consistent download with compile error in another block, then the dialog will indicate an error, and the load button will be disabled.


Load preview

Check before loading

| Status | ! | Target | Message | Action |
|---|---|---|---|---|
| | ✗ | ▾ PLC_1 | Loading will not be performed because preconditions are not met! | |
| | ✗ | ▾ Compile | Error found during compilation | |
| | ✗ | | Software compiling completed with error | |
| | ⚠ | ▸ Different modules | Differences between configured and target modules (online) | No action |
| | ✓ | ▸ Stop modules | All modules will be stopped for downloading to device | Stop all |
| | ✓ | ▾ Software | Download software to device | Consistent download |
| | ✓ | ▾ Overwrite online? | Objects that exist online and will be overwritten | |
| | ✓ | Data_block_1_DiF [DB1] | | ☑ Overwrite |
| | ✓ | Data_block_2_DiF [DB2] | | ☑ Overwrite |

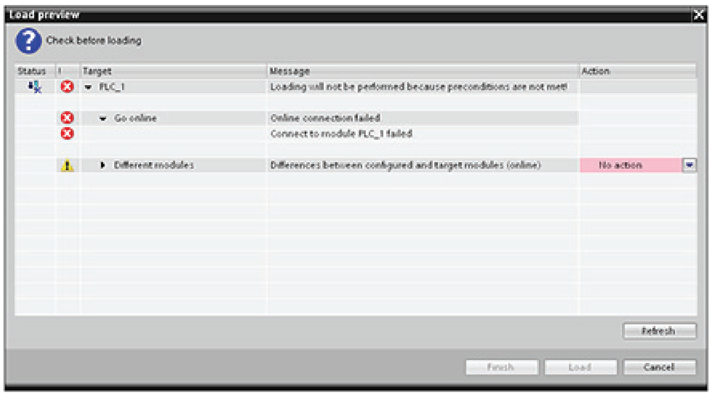
The user must correct the compile error in the other block. Then, the "Load" button becomes active.



| Status | ! | Target | Message | Action |
|---|---|---|---|---|
| | | PLC_1 | Ready for loading | |
| | | Different modules | Differences between configured and target modules (online) | No action |
| | | Stop modules | All modules will be stopped for downloading to device | Stop all |
| | | | Download to device is not possible as long as the module "PLC_1" is in RUN | |
| | | Software | Download software to device | Consistent download |
| | | Overwrite online? | Objects that exist online and will be overwritten | |
| | | Tags | | ✓ |
| | | Function_block_1_Diff [FC1] | | ✓ Overwrite |
| | | Data_block_1_Diff [DB1] | | ✓ Overwrite |
| | | Data_block_2_Diff [DB2] | | ✓ Overwrite |

## 14.11.5 System reaction if the download process fails

During the initial Download in RUN operation, if a network connection failure occurs, a "Load preview" dialog will result as shown in the figure below.



| Status | ! | Target | Message | Action |
|---|---|---|---|---|
| | | PLC_1 | Loading will not be performed because preconditions are not met! | |
| | | Go online | Online connection failed | |
| | | | Connect to module PLC_1 failed | |
| | | Different modules | Differences between configured and target modules (online) | No action |

### 14.11.6 Downloading the program in RUN mode

Before downloading the program in RUN mode, consider the effect of a RUN-mode modification on the operation of the CPU for the following situations:

- If you deleted the control logic for an output, the CPU maintains the last state of the output until the next power cycle or transition to STOP mode.
- If you deleted a high-speed counter or pulse output functions which were running, the high-speed counter or pulse output continues to run until the next power cycle or transition to STOP mode.
- Any logic that is conditional on the state of the first scan bit will not be executed until the next power cycle or transition from STOP to RUN mode. The first scan bit is set only by the transition to RUN mode and is not affected by a download in RUN mode.
- The current values of data blocks (DB) and/or tags will not be overwritten.

---

Note

Before you can download your program in RUN mode, the CPU must support changes in RUN mode, the program must compile with no errors, and the communication between STEP 7, and the CPU must be error-free.

You can make the following changes in program blocks and tags and download them in RUN mode:

- Create, overwrite, and delete Functions (FC), Function Blocks (FB), and Tag tables.
- Create and delete Data Blocks (DB); however, DB structure changes cannot be overwritten. Initial DB values can be overwritten. You cannot download a web server DB (control or fragment) in RUN mode.
- Overwrite Organization Blocks (OB); however, you cannot create or delete OBs.

A maximum number of ten blocks can be downloaded in RUN mode at one time. If more than ten blocks are downloaded, the CPU must be placed into STOP mode.

Once a download is initiated, you cannot perform other tasks in STEP 7 until the download is complete.

---

## Instructions that may fail due to "Download in RUN mode"

The following instructions may experience a temporary error when download in run changes are being activated in the CPU. The error occurs when the instruction is initiated while the CPU is preparing to activate the downloaded changes. During this time, the CPU suspends initiation of user-program access to the Load Memory, while it completes in-progress user-program access to Load Memory. This is done so that downloaded changes can be activated consistently.

| Instruction | Response while Activation is Pending |
|---|---|
| DataLogCreate | STATUS = W#16#80C0, ERROR = TRUE |
| DataLogOpen | STATUS = W#16#80C0, ERROR = TRUE |
| DataLogWrite | STATUS = W#16#80C0, ERROR = TRUE |
| DataLogClose | STATUS = W#16#80C0, ERROR = TRUE |
| DataLogNewFile | STATUS = W#16#80C0, ERROR = TRUE |
| READ_DBL | RET_VAL = W#16#82C0 |
| WRIT_DBL | RET_VAL = W#16#82C0 |
| RTM | RET_VAL = 0x80C0 |

In all cases the RLO output from the instruction will be false when the error occurs. The error is temporary. If it occurs, the instruction should be retried later.

**Note**

You must not retry the operation in the current execution of the OB.

# Technical specifications A

## A.1 General Technical Specifications

### Standards compliance

The S7-1200 automation system design conforms with the following standards and test specifications. The test criteria for the S7-1200 automation system are based on these standards and test specifications.

Note that not all S7-1200 models may be certified to these standards, and certification status may change without notification. It is the user's responsibility to determine applicable certifications by referring to the ratings marked on the product. Consult your local Siemens representative if you need additional information related to the latest listing of exact approvals by part number.

### CE approval

The S7-1200 Automation System satisfies requirements and safety related objectives according to the EC directives listed below, and conforms to the harmonized European standards (EN) for the programmable controllers listed in the Official Journals of the European Community.

- EC Directive 2006/95/EC (Low Voltage Directive) "Electrical Equipment Designed for Use within Certain Voltage Limits"
  - EN 61131-2:2007 Programmable controllers - Equipment requirements and tests
- EC Directive 2004/108/EC (EMC Directive) "Electromagnetic Compatibility"
  - Emission standard
    EN 61000-6-4:2007: Industrial Environment
  - Immunity standard
    EN 61000-6-2:2005: Industrial Environment
- EC Directive 94/9/EC (ATEX) "Equipment and Protective Systems Intended for Use in Potentially Explosive Atmosphere"
  - EN 60079-15:2005: Type of Protection 'n'

The CE Declaration of Conformity is held on file available to competent authorities at:

Siemens AG
IA AS RD ST PLC Amberg
Werner-von-Siemens-Str. 50
D92224 Amberg
Germany

## cULus approval

Underwriters Laboratories Inc. complying with:

- Underwriters Laboratories, Inc.: UL 508 Listed (Industrial Control Equipment)
- Canadian Standards Association: CSA C22.2 Number 142 (Process Control Equipment)

| NOTICE |
|---|
| The SIMATIC S7-1200 series meets the CSA standard.<br>The cULus logo indicates that the S7-1200 has been examined and certified by Underwriters Laboratories (UL) to standards UL 508 and CSA 22.2 No. 142. |

## FM approval

Factory Mutual Research (FM)
Approval Standard Class Number 3600 and 3611
Approved for use in:
Class I, Division 2, Gas Group A, B, C, D, Temperature Class T3C Ta = 60° C
Class I, Zone 2, IIC, Temperature Class T3 Ta = 60° C
Canadian Class I, Zone 2 Installation per CEC 18-150

IMPORTANT EXCEPTION: See Technical Specifications for the number of inputs or outputs allowed on simultaneously. Some models are de-rated for Ta = 60° C.

| WARNING |
|---|
| Substitution of components may impair the suitability for Class I, Division 2 and Zone 2.<br>Repair of units should only be performed by an authorized Siemens Service Center. |

## ATEX approval

ATEX approval applies to DC models only. ATEX approval does not apply to AC and Relay models.

EN 60079-0:2006: Explosive Atmospheres - General Requirements

EN 60079-15:2005: Electrical Apparatus for Potentially Explosive Atmospheres;
Type of protection 'nA'
II 3 G Ex nA II T3

IMPORTANT EXCEPTION: See Technical Specifications for the number of inputs or outputs allowed on simultaneously. Some models are de-rated for Ta = 60° C.

Install modules in a suitable enclosure providing a minimum degree of protection of IP54 according to EN 60529 and take into account the environmental conditions under which the equipment will be used.

When the temperature under rated conditions exceed 70° C at the cable entry point, or 80° C at the branching point of the conductors, the temperature specification of the selected cable should be in compliance with the actual measured temperature.