YASKAWA

VIPA SPEED7 Library

OPL_SP7-LIB | SW90AS0MA V10.007 | Manual

HB00 | OPL_SP7-LIB | SW90AS0MA V10.007 | en | 19-08

Block library - Modbus Communication

www.vipa.com/en/service-support/manuals

VIPA CONTROLS

VIPA GmbH
Ohmstr. 4
91074 Herzogenaurach
Telephone: +49 9132 744-0
Fax: +49 9132 744-1864
Email: info@vipa.com
Internet: www.vipa.com

SW90AS0MA_000_OPL_SP7-LIB_ModbusCommunication,6,EN -   © 2019

Table of contents

1 General

1.1 Copyright © VIPA GmbH

All Rights Reserved

For permission to reproduce or distribute, please contact: VIPA, Gesellschaft für Visualisierung und Prozessautomatisierung mbH Ohmstraße 4, D-91074 Herzogenaurach, Germany

Tel.: +49 9132 744 -0

Fax.: +49 9132 744-1864

EMail: info@vipa.de

http://www.vipa.com

Every effort has been made to ensure that the information contained in this document was complete and accurate at the time of publishing. Nevertheless, the authors retain the right to modify the information.

This customer document describes all the hardware units and functions known at the present time. Descriptions may be included for units which are not present at the customer site. The exact scope of delivery is described in the respective purchase contract.

EC Conformity Declaration

Hereby, VIPA GmbH declares that the products and systems are in compliance with the essential requirements and other relevant provisions. Conformity is indicated by the CE marking affixed to the product.

Conformity Information

For more information regarding CE marking and Declaration of Conformity (DoC), please contact your local VIPA customer service organization.

Trademarks

VIPA, SLIO, System 100V, System 200V, System 300V, System 300S, System 400V, System 500S and Commander Compact are registered trademarks of VIPA Gesellschaft für Visualisierung und Prozessautomatisierung mbH.

SPEED7 is a registered trademark of profichip GmbH.

SIMATIC, STEP, SINEC, TIA Portal, S7-300, S7-400 and S7-1500 are registered trademarks of Siemens AG.

Microsoft and Windows are registered trademarks of Microsoft Inc., USA.

Portable Document Format (PDF) and Postscript are registered trademarks of Adobe Systems, Inc.

All other trademarks, logos and service or product marks specified herein are owned by their respective companies.

Information product support

Contact your local VIPA Customer Service Organization representative if you wish to report errors or questions regarding the contents of this document. If you are unable to locate a customer service centre, contact VIPA as follows:

VIPA GmbH, Ohmstraße 4, 91074 Herzogenaurach, Germany

Telefax: +49 9132 744-1204

EMail: documentation@vipa.de

Technical support

Contact your local VIPA Customer Service Organization representative if you encounter problems with the product or have questions regarding the product. If you are unable to locate a customer service centre, contact VIPA as follows:

VIPA GmbH, Ohmstraße 4, 91074 Herzogenaurach, Germany

Tel.: +49 9132 744-1150 (Hotline)

EMail: support@vipa.de

1.2 About this manual

Objective and contents

The manual describes the block library *'Modbus Communication'* from VIPA:

- It contains a description of the structure, project implementation and usage in several programming systems.
- The manual is targeted at users who have a background in automation technology.
- The manual is available in electronic form as PDF file. This requires Adobe Acrobat Reader.
- The manual consists of chapters. Every chapter provides a self-contained description of a specific topic.
- The following guides are available in the manual:
  - An overall table of contents at the beginning of the manual
  - References with pages numbers

Icons Headings

Important passages in the text are highlighted by following icons and headings:

⚠ **DANGER!**
Immediate or likely danger. Personal injury is possible.

⚠ **CAUTION!**
Damages to property is likely if these warnings are not heeded.

ℹ *Supplementary information and useful tips.*

2 Important notes

2.1 General

In the following, you will find important notes, which must always be observed when using the blocks.

2.2 Internally used blocks

CAUTION!
The following blocks are used internally and must not be overwritten! The direct call of an internal block leads to errors in the corresponding instance DB! Please always use the corresponding function for the call.

FC/SFC	Designation	Description
FC/SFC 131	TSEND_	is used internally for FB 63
FC/SFC 132	TRECV_	is used internally for FB 64
FC/SFC 133	TCON_	is used internally for FB 65
FC/SFC 134	TDISCON_	is used internally for FB 66
FC/SFC 135	TUSEND_	is used internally for FB 67
FC/SFC 136	TURECV_	is used internally for FB 68
FC/SFC 192	CP_S_R	is used internally for FB 7 and FB 8
FC/SFC 196	AG_CNTRL	is used internally for FC 10
FC/SFC 198	USEND_	is used internally for FB 8
FC/SFC 198	URCV_	is used internally for FB 9
FC/SFC 200	AG_GET	is used internally for FB/SFB 14
FC/SFC 201	AG_PUT	is used internally for FB/SFB 15
FC/SFC 202	AG_BSEND	is used internally for FB/SFB 12
FC/SFC 203	AG_BRCV	is used internally for FB/SFB 13
FC/SFC 204	IP_CONF	is used internally for FB 55 IP_CONF
FC/SFC 205	AG_SEND	is used internally for FC 5 AG_SEND
FC/SFC 206	AG_RECV	is used internally for FC 6 AG_RECV
FC/SFC 253	IBS_ACCESS	is used internally for SPEED bus INTERBUS masters
SFB 238	EC_RWOD	is used internally for EtherCAT Communication
SFB 239	FUNC	is used internally for FB 240, FB 241

3 Include library

Block library *'Modbus Communication'*

The block library can be found for download in the *'Service/Support'* area of www.vipa.com at *'Downloads ➔ VIPA Lib'* as *'Block library Modbus Communication - SW90AS0MA'*. The library is available as packed zip file. As soon as you want to use these blocks you have to import them into your project.

> *Please always use the manual associated with your library. As long as there are no description-relevant changes, the version information in the manual can differ from those of the library and its files.*

The following block libraries are available

File	Description
Modbus_S7_V0006.zip	■ Block library for Siemens SIMATIC Manager. ■ For use in CPUs from VIPA.
Modbus_TIA_V0009.zip	■ Block library for Siemens TIA Portal V14 and 15. ■ For use in CPUs from VIPA.

3.1 Integration into Siemens SIMATIC Manager

Overview

The integration into the Siemens SIMATIC Manager requires the following steps:

1. Load ZIP file
2. "Retrieve" the library
3. Open library and transfer blocks into the project

Load ZIP file

- Navigate on the web page to the desired ZIP file, load and store it in your work directory.

Retrieve library

1. Start the Siemens SIMATIC Manager with your project.
2. Open the dialog window for ZIP file selection via *'File ➔ Retrieve'*.
3. Select the according ZIP file and click at [Open].
4. Select a destination folder where the blocks are to be stored.
5. Start the extraction with [OK].

Open library and transfer blocks into the project

1. Open the library after the extraction.
2. Open your project and copy the necessary blocks from the library into the directory "blocks" of your project.

   ⇨ Now you have access to the VIPA specific blocks via your user application.

> *Are FCs used instead of SFCs, so they are supported by the VIPA CPUs starting from firmware 3.6.0.*

3.2 Integration into Siemens TIA Portal

Overview

The integration into the Siemens TIA Portal requires the following steps:

1. Load ZIP file
2. Unzip the Zip file
3. "Retrieve" the library
4. Open library and transfer blocks into the project

Load ZIP file

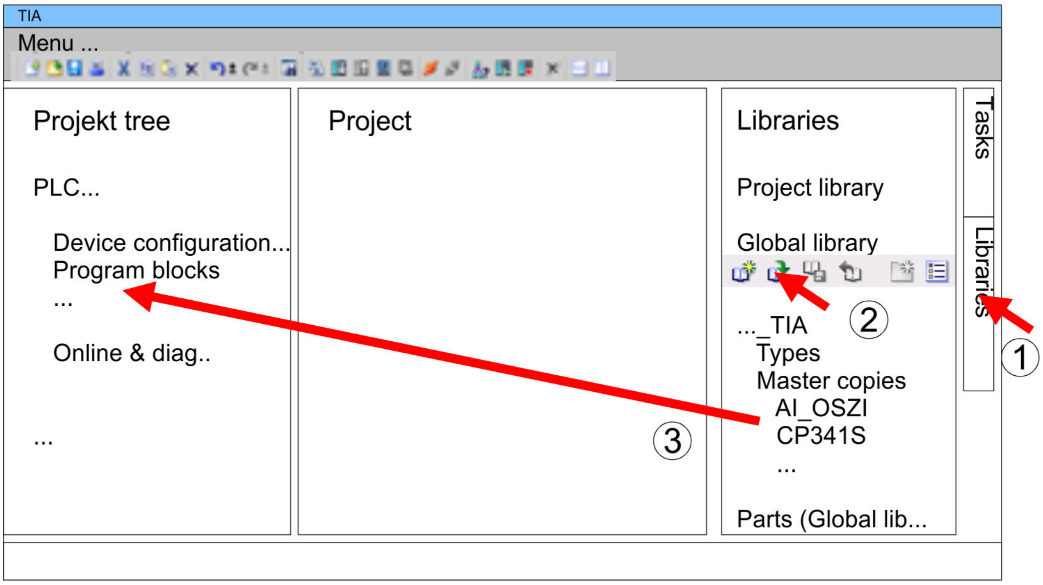
1. Navigate on the web page to the ZIP file, that matches your version of the program.
2. Load and store it in your work directory.

Unzip the Zip file

- Unzip the zip file to a work directory of the Siemens TIA Portal with your unzip application.

Open library and transfer blocks into the project

1. Start the Siemens TIA Portal with your project.
2. Switch to the *Project view*.
3. Choose "Libraries" from the task cards on the right side.
4. Click at "Global libraries".
5. Click at "Open global libraries".
6. Navigate to your work directory and load the file ..._TIA.al1x.



7. Copy the necessary blocks from the library into the "Program blocks" of the *Project tree* of your project. Now you have access to the VIPA specific blocks via your user application.

4 Modbus Communication

4.1 TCP

4.1.1 FB 70 - TCP_MB_CLIENT - Modbus/TCP client

4.1.1.1 Description

This function allows the operation of an Ethernet interface as Modbus/TCP client.

Call parameter

Name	Declaration	Type	Description
REQ	IN	BOOL	Start job with edge 0-1.
ID	IN	WORD	ID from TCON.
MB_FUNCTION	IN	BYTE	Modbus: *Function code*.
MB_DATA_ADDR	IN	WORD	Modbus: Start address or *sub function code*.
MB_DATA_LEN	IN	INT	Modbus: Number of register/bits.
MB_DATA_PTR	IN	ANY	Modbus: Data buffer (only flag area or data block of data type byte allowed) for access with *function code 03h*, *06h* and *10h*.
DONE *	OUT	BOOL	Job finished without error.
BUSY	OUT	BOOL	Job is running.
ERROR *	OUT	BOOL	Job is ready with error - parameter STATUS has error information.
STATUS *	OUT	WORD	Extended status and error information.
*) Parameter is available until the next call of the FB.			

Parameter in instance DB

Name	Declaration	Type	Description
PROTOCOL_TIMEOUT	STAT	INT	Blocking time before an active job can be cancelled by the user. Default: 3s
RCV_TIMEOUT	STAT	INT	Monitoring time for a job. Default: 2s
MB_TRANS_ID	STAT	WORD	Modbus: Start value for the transaction identifier. Default: 1
MB_UNIT_ID	STAT	BYTE	Modbus: Device identification. Default: 255

The following must be observed:

- The *call parameters* must be specified with the block call. Besides the *call parameters* all parameters are located in the instance DB.
- The communication link must be previously initialized via FB 65 (TCON).
- FB 63 (TSEND) and FB 64 (TRCV) are required for the use of the block.
- During a job processing the instance DB is blocked for other clients.

- During job processing changes to the input parameters are not evaluated.
- With the following conditions a job processing is completed or cancelled:
  - DONE = 1 job without error
  - ERROR = 1 job with error
  - Expiration of RCV_TIMEOUT
  - REQ = FALSE after expiration of PROTOCOL_TIMEOUT
- REQ is reset before DONE or ERROR is set or PROTOCOL_TIMEOUT has expired, STATUS 8200h is reported. Here the current job is still processed.

Status and error indication

The function block reports via STATUS the following status and error information.

STATUS	DONE	BUSY	ERROR	Description
0000h	1	0	0	Operation executed without error.
7000h	0	0	0	No connection established or communication error (TCON).
7004h	0	0	0	Connection established and monitored. No job active.
7005h	0	1	0	Data are sent.
7006h	0	1	0	Data are received.
8210h	0	0	1	The hardware is incompatible with the block library Modbus RTU/TCP.
8380h	0	0	1	Received Modbus frame does not have the correct format or has an invalid length.
8381h	0	0	1	Server returns *Exception Code 01h*. ↳ *24*
8382h	0	0	1	Server returns *Exception Code 03h* or wrong start address. ↳ *24*
8383h	0	0	1	Server returns *Exception Code 02h*. ↳ *24*
8384h	0	0	1	Server returns *Exception Code 04h*. ↳ *24*
8386h	0	0	1	Server returns wrong *Function code*.
8387h	0	0	1	Connection ID (TCON) does not match the instance or server returns wrong protocol ID.
8388h	0	0	1	Server returns wrong value or wrong quantity.
80C8h	0	0	1	No answer of the server during the duration (RCV_TIMEOUT).
8188h	0	0	1	MB_FUNCTION not valid.
8189h	0	0	1	MB_DATA_ADDR not valid.
818Ah	0	0	1	MB_DATA_LEN not valid.
818Bh	0	0	1	MB_DATA_PTR not valid.
818Ch	0	0	1	BLOCKED_PROC_TIMEOUT or RCV_TIMEOUT not valid.
818Dh	0	0	1	Server returns wrong transaction ID.
8200h	0	0	1	Another Modbus request is processed at the time via the port (PROTOCOL_TIMEOUT).

4.1.1.2 Example

Task

With *Function code 03h*, starting from address 2000, 100 register are to be read from a Modbus/TCP server and stored in flag area starting from MB200. Errors are to be stored.

OB1

```
 CALL  FB    65 , DB65
       REQ    :=M100.0
       ID     :=W#16#1
       DONE   :=M100.1
       BUSY   :=
       ERROR  :=M100.2
       STATUS :=MW102
       CONNECT:=P#DB255.DBX 0.0 BYTE 64

      UN    M    100.2
      SPB   ERR1
      L     MW   102
      T     MW   104
ERR1: NOP   0
      U     M    100.1
      R     M    100.0

CALL  FB    70 , DB70
       REQ         :=M101.0
       ID          :=W#16#1
       MB_FUNCTION :=B#16#3
       MB_DATA_ADDR:=W#16#7D0
       MB_DATA_LEN :=100
       MB_DATA_PTR :=P#M 200.0 BYTE 200
       DONE        :=M101.1
       BUSY        :=
       ERROR       :=M101.2
       STATUS      :=MW106

      UN    M    101.2
      SPB   ERR2
      L     MW   106
      T     MW   108
ERR2: NOP   0
      U     M    101.1
      R     M    101.0
```

OB1 - Description

1. Calling of FB 65 (TCON) to establish a communication connection with the partner station.
2. Calling the handling block of the Modbus/TCP client with the correct parameters.
3. There is no connection to the partner station and MW102 returns 7000h.
4. Set M100.0 in the CPU to TRUE.

   ⇨ If M100.0 is automatically reset, the connection to the partner station is established and MW108 returns 7004h.
5. Set M101.0 in the CPU to TRUE.

   ⇨ The Modbus request is sent and it is waited for a response.

   If M101.0 is automatically reset, the job was finished without errors and the read data are stored in the CPU starting from bit memory byte 200. MW108 returns 7004h and indicates waiting for a new job.

   If M101.0 is not automatically reset and MW108 returns non-zero, an error has occurred. The cause of error can be read by the code of MW108 (e.g. MW108 = 8382h when the start address 2000 in the server is not available). MW108 returns 7004h and indicates waiting for a new job.

4.1.2 FB 71 - TCP_MB_SERVER - Modbus/TCP server

4.1.2.1 Description

This function allows the operation of an Ethernet interface as Modbus/TCP server.

Call parameter

Name	Declaration	Type	Description
ENABLE	IN	BOOL	Activation/Deactivation Modbus server.
MB_DATA_PTR	IN	ANY	Modbus: Data buffer (only flag area or data block of type Byte allowed) for access with *function code 03h*, *06h* and *10h*.
ID	IN	WORD	ID from TCON.
NDR*	OUT	BOOL	New data were written by the Modbus client.
DR*	OUT	BOOL	Data were read by the Modbus client.
ERROR*	OUT	BOOL	Job is ready with error - parameter STATUS has error information.
STATUS*	OUT	WORD	Extended status and error information.
*) Parameter is available until the next call of the FB.			

Parameter in instance DB

Name	Declaration	Type	Description
REQUEST_COUNT	STAT	WORD	Counter for each received frame.
MESSAGE_COUNT	STAT	WORD	Counter for each valid Modbus request.
XMT_RCV_COUNT	STAT	WORD	Counter for each received frame, which contains no valid Modbus request.
EXCEPTION_COUNT	STAT	WORD	Counter for each negatively acknowledged Modbus request.
SUCCESS_COUNT	STAT	WORD	Counter for each positively acknowledged Modbus request.
FC1_ADDR_OUTPUT_START	STAT	WORD	Modbus *Function code 01h* start register for Q0.0 Default: 0
FC1_ADDR_OUTPUT_END	STAT	WORD	Modbus *Function code 01h* end register for Qx.y Default: 19999
FC1_ADDR_MEMORY_START	STAT	WORD	Modbus *Function code 01h* start register for M0.0 Default: 20000
FC1_ADDR_MEMORY_END	STAT	WORD	Modbus *Function code 01h* end register for Mx.y Default: 39999
FC2_ADDR_INPUT_START	STAT	WORD	Modbus *Function code 02h* start register for I0.0 Default: 0

Name	Declara-tion	Type	Description
FC2_ADDR_INPUT_END	STAT	WORD	Modbus *Function code 02h* end register for Ix.y Default: 19999
FC2_ADDR_MEMORY_START	STAT	WORD	Modbus *Function code 02h* start register for M0.0 Default: 20000
FC2_ADDR_MEMORY_END	STAT	WORD	Modbus *Function code 02h* end register for Mx.y Default: 39999
FC4_ADDR_INPUT_START	STAT	WORD	Modbus *Function code 04h* start register for IW0 Default: 0
FC4_ADDR_INPUT_END	STAT	WORD	Modbus *Function code 04h* end register for IWx Default: 19999
FC4_ADDR_MEMORY_START	STAT	WORD	Modbus *Function code 04h* start register for MW0 Default: 20000
FC4_ADDR_MEMORY_END	STAT	WORD	Modbus *Function code 04h* end register for MWx Default: 39999
FC5_ADDR_OUTPUT_START	STAT	WORD	Modbus *Function code 05h* start register for Q0.0 Default: 0
FC5_ADDR_OUTPUT_END	STAT	WORD	Modbus *Function code 05h* end register for Qx.y Default: 19999
FC5_ADDR_MEMORY_START	STAT	WORD	Modbus *Function code 05h* start register for M0.0 Default: 20000
FC5_ADDR_MEMORY_END	STAT	WORD	Modbus *Function code 05h* end register for Mx.y Default: 39999
FC15_ADDR_OUTPUT_START	STAT	WORD	Modbus *Function code 0Fh* start register for Q0.0 Default: 0
FC15_ADDR_OUTPUT_END	STAT	WORD	Modbus *Function code 0Fh* end register for Qx.y Default: 19999
FC15_ADDR_MEMORY_START	STAT	WORD	Modbus *Function code 0Fh* start register for Q0.0 Default: 20000
FC15_ADDR_MEMORY_END	STAT	WORD	Modbus *Function code 0Fh* end register for Qx.y Default: 39999

The following must be observed:

- The *call parameters* must be specified with the block call. Besides the *call parameters* all parameters are located in the instance DB.
- The communication link must be previously initialized via FB 65 (TCON).
- FB 63 (TSEND) and FB 64 (TRCV) are required for the use of the block.
- The INPUT/OUTPUT Modbus addresses of a *Function code* must be located in front of the MEMORY Modbus address and thus always be lower.

- Within a *Function code* no Modbus address may be defined multiple times - also not 0!
- The server can only process one job simultaneously. New Modbus requests during job processing are ignored and not answered.

Status and error indication

The function block reports via *STATUS* the following status and error information.

STATUS	NDR	DR	ERROR	Description
0000h	0 or 1*		0	Operation executed without error.
7000h	0	0	0	No connection established or communication error (TCON).
7005h	0	0	0	Data are sent.
7006h	0	0	0	Data are received.
8210h	0	0	1	The hardware is incompatible with the block library Modbus RTU/TCP.
8380h	0	0	1	Received Modbus frame does not have the correct format or bytes are missing.
8381h	0	0	1	*Exception Code 01h*, *Function code* is not supported. ☞ *24*
8382h	0	0	1	*Exception Code 03h*, data length or data value are not valid. ☞ *24*
8383h	0	0	1	*Exception Code 02h*, invalid start address or address range ☞ *24*
8384h	0	0	1	*Exception Code 04h*, area length error when accessing inputs, outputs or bit memories. ☞ *24*
8387h	0	0	1	Connection ID (TCON) does not match the instance or client returns wrong protocol ID.
8187h	0	0	1	MB_DATA_PTR not valid.

*) Error free Modbus job with *Function code 05h, 06h, 0Fh* or *10h* returns NDR=1 and DR=0 resp. error free Modbus job with *Function code 01h, 02h, 03h, 04h* return DR=1 and NDR=0.

4.1.2.2 Example

Task

The CPU provides 100 byte data in the flag area starting from MB200 for a Modbus client via the Modbus register 0...49. Data can be read from the Modbus client via *Function code 03h* and written with *Function code 06h, 10h*. The CPU output Q1.0 is to be controlled by a Modbus client via *Function code 05h* and the start address 5008. Errors are to be stored.

OB1

```
CALL  FB    65 , DB65
       REQ    :=M100.0
       ID     :=W#16#1
       DONE   :=M100.1
       BUSY   :=
       ERROR  :=M100.2
       STATUS :=MW102
       CONNECT:=P#DB255.DBX 0.0 BYTE 64

      UN    M    100.2
      SPB   ERR1

      L     MW   102
      T     MW   104

ERR1: NOP   0
      U     M    100.1
      R     M    100.0

      L     5000
      T     DB71.DBW   52

      CALL  FB    71 , DB71
       ENABLE     :=M101.0
       MB_DATA_PTR:=P#M 200.0 BYTE 100
       ID         :=W#16#1
       NDR        :=M101.1
       DR         :=M101.2
       ERROR      :=M101.3
       STATUS     :=MW106

      UN    M    101.3
      SPB   ERR2
      L     MW   106
      T     MW   108
ERR2: NOP   0
```

OB1 - Description

1. Call of FB 65 (TCON) to establish a communication connection with the partner station.
2. Calling the handling block of the Modbus/TCP server with the correct parameters.
3. There is no connection to the partner station and MW102 returns 7000h.
4. Set M100.0 in the CPU to TRUE.
   ⇨ If M100.0 is automatically reset, the connection to the partner station is established and MW108 returns 7006h.
5. The Modbus start register in the process image, which can be reached by *Function code 05h*, may be changed in the example by the parameter FC5_ADDR_OUTPUT_START (word 52 in the instance data block).
6. Set M101.0 in the CPU to TRUE.
   ⇨ The Modbus server now works.
7. The client sends a Modbus request with *Function code 03h* start address 10 and quantity 30.

⇨ The server responds with 60 byte starting from MB220. DR is set for one CPU cycle and thus M101.2 is set to "1".

8. The client sends a Modbus request with *Function code 05h* start address 5008 and the value FF00h.

⇨ The server acknowledges the request and writes "1" to the output Q1.0. NDR is set for one CPU cycle and thus M101.1 is set to "1".

9. The client sends a Modbus request with *Function code 03h* start address 50 (does not exist) and quantity 1.

⇨ The server responds with *Exception Code 02h* an sets ERROR/STATUS for one CPU cycle. MW108 returns 8383h.

4.2 RTU

4.2.1 FB 72 - RTU_MB_MASTER - Modbus RTU master

4.2.1.1 Description

This function block allows the operation of the internal serial RS485 interface of a SPEED7 CPU from VIPA or a System SLIO CP 040 as Modbus RTU master.

Call parameter

Name	Declaration	Type	Description
REQ	IN	BOOL	Start job with edge 0-1.
HARDWARE	IN	BYTE	1 = System SLIO CP 040 / 2 = VIPA SPEED7 CPU
LADDR	IN	INT	Logical address of the System SLIO CP 040 (parameter is ignored with the VIPA SPEED7 CPU).
MB_UNIT_ID	IN	BYTE	Modbus: Device identification = Address of the slave (0 ... 247).
MB_FUNCTION	IN	BYTE	Modbus: *Function code*. Please note that the *Function code* 16h is not supported!
MB_DATA_ADDR	IN	WORD	Modbus: Start address or *Sub function code*.
MB_DATA_LEN	IN	INT	Modbus: Number of register/bits.
MB_DATA_PTR	IN	ANY	Modbus: Data buffer (only flag area or data block of data type byte allowed) for access with *function code 03h*, *06h* and *10h*.
DONE*	OUT	BOOL	Job finished without error.
BUSY	OUT	BOOL	Job is running.
ERROR*	OUT	BOOL	Job is ready with error - parameter *STATUS* has error information.
STATUS*	OUT	WORD	Extended status and error information.
*) Parameter is available until the next call of the FB.			

Parameter in instance DB

Name	Declaration	Type	Description
INIT	STAT	BOOL	With an edge 0-1 an synchronous reset is established at the System SLIO CP 040. After a successful reset the bit automatically reset.

The following must be observed:

- The *call parameters* must be specified with the block call. Besides the *call parameters* all parameters are located in the instance DB.
- The interface to be used must be configured before:
  - VIPA System SLIO CP 040: Configuration as "Modbus master RTU" with 60 byte IO-Size in the hardware configuration.
  - Internal serial RS485 interface of a VIPA CPU:
    Configuration via SFC 216 (SER_CFG) with protocol "Modbus master RTU".
- FB 60 SEND and FB 61 RECEIVE (or FB 65 SEND_RECV) are required for the use of the block, even if the internal serial RS485 interface of a CPU from VIPA is used.
- During job processing changes to the input parameters are not evaluated.
- Broadcast request via MB_UNIT_ID = 0 are only accepted for writing functions.
- With the following conditions a job processing is completed or cancelled:
  - *DONE* = 1 job without error
  - *ERROR* = 1 job with error
  - Expiration of time-out (parameterization at the interface)
- If *REQ* is reset before *DONE* or *ERROR* is set, STATUS 8200h is reported. Here the current job is still processed.

Status and error indication

The function block reports via STATUS the following status and error information.

STATUS	DONE	BUSY	ERROR	Description
0000h	1	0	0	Operation executed without error.
7000h	0	0	0	No connection established or communication error.
7004h	0	0	0	Connection established and monitored. No job active.
7005h	0	1	0	Data are sent.
7006h	0	1	0	Data are received.
8210h	0	0	1	The hardware is incompatible with the block library Modbus RTU/TCP.
8381h	0	0	1	Server returns *Exception Code 01h*. ➜ *24*
8382h	0	0	1	Server returns *Exception Code 03h* or wrong start address. ➜ *24*
8383h	0	0	1	Server returns *Exception Code 02h*. ➜ *24*
8384h	0	0	1	Server returns *Exception Code 04h*. ➜ *24*
8386h	0	0	1	Server returns wrong *Function code*.
8388h	0	0	1	Server returns wrong value or quantity.
80C8h	0	0	1	No answer of the server during the defined duration (time-out parameterizable via interface).
8188h	0	0	1	MB_FUNCTION not valid.
8189h	0	0	1	MB_DATA_ADDR not valid.
818Ah	0	0	1	MB_DATA_LEN not valid.
818Bh	0	0	1	MB_DATA_PTR not valid.
8201h	0	0	1	HARDWARE not valid.
8202h	0	0	1	MB_UNIT_ID not valid.
8200h	0	0	1	Another Modbus request is processed at the time via the port.

4.2.1.2 Example

Task

With *Function code 03h*, starting from address 2000, 100 register are to be read from a Modbus RTU slave with address 99 and stored in flag area starting from MB200. Errors are to be stored. The Modbus RTU master is realized via the internal serial RS485 interface of a VIPA CPU.

OB100

```
 CALL  SFC  216
       Protocol :=B#16#5
       Parameter :=DB10
       Baudrate:=B#16#9
       CharLen:=B#16#3
       Parity:=B#16#2
       StopBits:=B#16#1
       FlowControl:=B#16#1
       RetVal:=MW100
```

OB100 - Description

1. Calling of the SFC 216 (SER_CFG) to configure the internal serial interface of the CPU from VIPA.
2. Protocol: "Modbus Master RTU", 9600 baud, 8 data bit, 1 stop bit, even parity, no flow control.
3. DB10 has a variable of type WORD with a Modbus time-out (value in ms).

OB1

```
CALL  FB    72 , DB72
       REQ         :=M101.0
       HARDWARE    :=B#16#2
       LADDR       :=
       MB_UNIT_ID  :=B#16#63
       MB_FUNCTION :=B#16#3
       MB_DATA_ADDR:=W#16#7D0
       MB_DATA_LEN :=100
       MB_DATA_PTR :=P#M 200.0 BYTE 200
       DONE        :=M101.1
       BUSY        :=
       ERROR       :=M101.2
       STATUS      :=MW102

      UN    M    101.2
      SPB   ERR1
      L     MW   102
      T     MW   104
ERR1: NOP   0
      U     M    101.1
      R     M    101.0
```

OB1 - Description

1. Calling the handling block of the Modbus RTU master with the correct parameters.
2. If the interface was correctly initialized in the OB 100, the master can be used and MW102 returns 7004h.

3. Set M101.0 in the CPU to TRUE.

⇨ The Modbus request is sent and it is waited for a response.

If M101.0 is automatically reset, the job was finished without errors and the read data are stored in the CPU starting from bit memory byte 200. MW104 returns 7004h and indicates waiting for a new job.

If M101.0 is not automatically reset and MW104 returns non-zero, an error has occurred. The cause of error can be read by the code of MW104 (e.g. MW104 = 8382h when the start address 2000 in the server is not available). MW102 returns 7004h and indicates waiting for a new job.

4.2.2 FB 73 - RTU_MB_SLAVE - Modbus RTU slave

4.2.2.1 Description

This function block allows the operation of the internal serial RS485 interface of a SPEED7 CPU from VIPA or a System SLIO CP 040 as Modbus RTU slave.

Call parameter

Name	Declaration	Type	Description
ENABLE	IN	BOOL	Activation/Deactivation Modbus server.
HARDWARE	IN	BYTE	1 = System SLIO CP 040 / 2 = VIPA SPEED7 CPU
LADDR	IN	INT	Logical address of the System SLIO CP 040 (parameter is ignored with the VIPA SPEED7 CPU).
MB_UNIT_ID	IN	BYTE	Modbus: Device identification = own address (1 ... 247).
MB_DATA_PTR	IN	ANY	Modbus: Data buffer (only flag area or data block of data type byte allowed) for access with *function code 03h*, *06h* and *10h*.
NDR*	OUT	BOOL	New data were written by the Modbus client.
DR*	OUT	BOOL	Data were read by the Modbus client.
ERROR*	OUT	BOOL	Job is ready with error - parameter *STATUS* has error information.
STATUS*	OUT	WORD	Extended status and error information.
*) Parameter is available until the next call of the FB			

Parameter in instance DB

Name	Declaration	Type	Description
INIT	STAT	BOOL	With an edge 0-1 an synchronous reset is established at the System SLIO CP 040.
REQUEST_COUNT	STAT	WORD	Counter for each received frame.
MESSAGE_COUNT	STAT	WORD	Counter for each valid Modbus request.
BROADCAST_COUNT	STAT	WORD	Counter for each valid Modbus broadcast request.

Name	Declaration	Type	Description
EXCEPTION_COUNT	STAT	WORD	Counter for each negatively acknowledged Modbus request.
SUCCESS_COUNT	STAT	WORD	Counter for each positively acknowledged Modbus request.
BAD_CRC_COUNT	STAT	WORD	Counter for each valid Modbus request with CRC error.
FC1_ADDR_OUTPUT_START	STAT	WORD	Modbus *Function code 01h* start register for Q0.0 Default: 0
FC1_ADDR_OUTPUT_END	STAT	WORD	Modbus *Function code 01h* end register for Qx.y Default: 19999
FC1_ADDR_MEMORY_START	STAT	WORD	Modbus *Function code 01h* start register for M0.0 Default: 20000
FC1_ADDR_MEMORY_END	STAT	WORD	Modbus *Function code 01h* end register for Mx.y Default: 39999
FC2_ADDR_INPUT_START	STAT	WORD	Modbus *Function code 02h* start register for I0.0 Default: 0
FC2_ADDR_INPUT_END	STAT	WORD	Modbus *Function code 02h* end register for Ix.y Default: 19999
FC2_ADDR_MEMORY_START	STAT	WORD	Modbus *Function code 02h* start register for M0.0 Default: 20000
FC2_ADDR_MEMORY_END	STAT	WORD	Modbus *Function code 02h* end register for Mx.y Default: 39999
FC4_ADDR_INPUT_START	STAT	WORD	Modbus *Function code 04h* start register for IW0 Default: 0
FC4_ADDR_INPUT_END	STAT	WORD	Modbus *Function code 04h* end register for IWx Default: 19999
FC4_ADDR_MEMORY_START	STAT	WORD	Modbus *Function code 04h* start register for MW0 Default: 20000
FC4_ADDR_MEMORY_END	STAT	WORD	Modbus *Function code 04 h* end register for MW0 Default: 39999
FC5_ADDR_OUTPUT_START	STAT	WORD	Modbus *Function code 05h* start register for Q0.0 Default: 0
FC5_ADDR_OUTPUT_END	STAT	WORD	Modbus *Function code 05h* end register for Qx.y Default: 19999
FC5_ADDR_MEMORY_START	STAT	WORD	Modbus *Function code 05h* start register for M0.0 Default: 20000
FC5_ADDR_MEMORY_END	STAT	WORD	Modbus *Function code 05h* end register for Mx.y Default: 39999

Name	Declaration	Type	Description
FC15_ADDR_OUTPUT_START	STAT	WORD	Modbus *Function code 0Fh* start register for Q0.0 Default: 0
FC15_ADDR_OUTPUT_END	STAT	WORD	Modbus *Function code 0Fh* end register for Qx.y Default: 19999
FC15_ADDR_MEMORY_START	STAT	WORD	Modbus *Function code 0Fh* start register for M0.0 Default: 20000
FC15_ADDR_MEMORY_END	STAT	WORD	Modbus *Function code 0Fh* end register for Mx.y Default: 39999

The following must be observed:

- The *call parameters* must be specified with the block call. Besides the *call parameters* all parameters are located in the instance DB.
- The interface to be used must be configured before:
  - VIPA System SLIO CP 040: Configuration as ASCII module with 60 byte IO-Size in the hardware configuration.
  - Internal serial RS485 interface of a VIPA CPU:
    Configuration via SFC 216 (SER_CFG) with protocol "ASCII".
- FB 60 SEND and FB 61 RECEIVE (or FB 65 SEND_RECV) are required for the use of the block, even if the internal serial RS485 interface of a CPU from VIPA is used.
- Broadcast request via MB_UNIT_ID = 0 are only accepted for writing functions.
- The INPUT/OUTPUT Modbus addresses of a *Function code* must be located in front of the MEMORY Modbus address and thus always be lower.
- Within a *Function code* no Modbus address may be defined multiple times - also not 0!
- The slave can only process one job simultaneously. New Modbus requests during job processing are ignored and not answered.

Status and error indication

The function block reports via STATUS the following status and error information.

STATUS	NDR	DR	ERROR	Description
0000h	0 or 1*		0	Operation executed without error.
7000h	0	0	0	No connection established or communication error.
7005h	0	0	0	Data are sent.
7006h	0	0	0	Data are received.
8210h	0	0	1	The hardware is incompatible with the block library Modbus RTU/TCP.
8380h	0	0	1	CRC error
8381h	0	0	1	*Exception Code 01h*, *Function code* is not supported. ☞ *24*
8382h	0	0	1	*Exception Code 03h*, data length or data value are not valid. ☞ *24*
8383h	0	0	1	*Exception Code 02h*, invalid start address or address range. ☞ *24*
8384h	0	0	1	*Exception Code 04h*, area length error when accessing inputs, outputs or bit memories. ☞ *24*
8187h	0	0	1	MB_DATA_PTR not valid.

STATUS	NDR	DR	ERROR	Description
8201h	0	0	1	HARDWARE not valid.
8202h	0	0	1	MB_UNIT_ID not valid.
8203 h	0	0	1	
*) Error free Modbus job with *Function code 05h, 06h, 0Fh* or *10h* returns NDR=1 and DR=0 resp. error free Modbus job with *Function code 01h, 02h, 03h, 04h* return DR=1 and NDR=0.				

4.2.2.2 Example

Task

The CPU provides 100 byte data in the flag area starting from MB200 for a Modbus master via the Modbus register 0 ... 49. Data can be read by the Modbus master via *Function code 03h* and written with *Function code 06h, 10h*. The CPU output Q1.0 is to be controlled by a Modbus master via *Function code 05h* and the start address 5008. Errors are to be stored. The Modbus RTU slave with the address 99 is realized via the internal serial RS485 interface of a VIPA CPU.

OB100

```
 CALL  SFC  216
       Protocol :=B#16#1
       Parameter :=DB10
       Baudrate:=B#16#9
       CharLen:=B#16#3
       Parity:=B#16#2
       StopBits:=B#16#1
       FlowControl:=B#16#1
       RetVal:=MW100
```

OB100 - Description

1. Calling of the SFC 216 (SER_CFG) to configure the internal serial interface of the CPU from VIPA.
2. Protocol: "ASCII", 9600 baud, 8 data bit, 1 stop bit, even parity, no flow control.
3. DB10 has a variable of type WORD and must be passed as "Dummy".

OB1

```
      L      5000
      T      DB73.DBW   58

      CALL   FB     73 , DB73
       ENABLE      :=M101.0
       HARDWARE    :=B#16#2
       LADDR       :=
       MB_UNIT_ID  :=B#16#63
       MB_DATA_PTR:=P#M 200.0 BYTE 100
       NDR         :=M101.1
       DR          :=M101.2
       ERROR       :=M101.3
       STATUS      :=MW102

      UN     M    101.3
      SPB    ERR1
      L      MW   102
      T      MW   104
ERR1: NOP    0
```

OB1 - Description

1. Calling the handling block of the Modbus/TCP server with the correct parameters.
2. If the interface was correctly initialized in the OB100, the slave can be used and MW102 returns 7006h.
3. The Modbus start register in the process image, which can be reached by *Function code 05h*, may be changed in the example by the parameter FC5_ADDR_OUTPUT_START (word 58 in the instance data block).
4. Set M101.0 in the CPU to TRUE.

   ⇨ The Modbus slave now works.
5. The master sends a Modbus request with *Function code 03h* start address 10 and quantity 30.

   ⇨ The slave responds with 60byte starting from MB200. DR is set for one CPU cycle and thus M101.2 is set to "1".
6. The master sends a Modbus request with *Function code 05h* start address 5008 and the value FF00h.

   ⇨ The salve acknowledges the request and writes "1" to the output Q1.0. NDR is set for one CPU cycle and thus M101.1 is set to "1".
7. The master sends a Modbus request with *Function code 03h* start address 50 (does not exist!) and quantity 1.

   ⇨ The server responds with *Exception Code 02h* and sets ERROR/STATUS for one CPU cycle. MW104 returns 8383h.

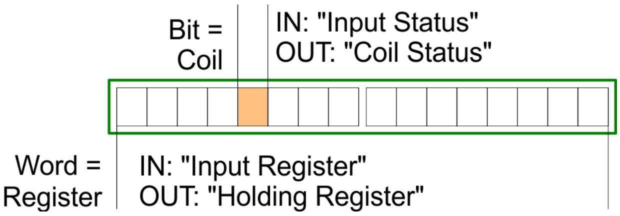
4.3 Modbus *Exception Codes*

Code	Name	Description
01	ILLEGAL FUNCTION	The function code received in the query is not an allowable action for the server (or slave). This may be because the function code is only applicable to newer devices and was not implemented in the unit selected. It could also indicate that the server (or slave) is in the wrong state to process a request of this type, for example because it is unconfigured and is being asked to return register values.
02	ILLEGAL DATA ADDRESS	The data address received in the query is not an allowable address for the server (or slave). More specifically, the combination of reference number and transfer length is invalid. For a controller with 100 registers, the PDU (**P**rotocol **D**ata **U**nit) addresses the first register as 0 and the last one as 99. If a request is submitted with a starting register address of 96 and a quantity of registers of 4, then this request will successfully operate (address-wise at least) on registers 96, 97, 98, 99. If a request is submitted with a starting register address of 96 and a quantity of registers of 5, then this request will fail with *Exception Code* 0x02 "ILLEGAL DATA ADDRESS" since it attempts to operate on registers 96, 97, 98, 99 and 100 and there is no register with address 100.
03	ILLEGAL DATA VALUE	A value contained in the query data field is not an allowable value for server (or slave). This indicates a fault in the structure of the remainder of a complex request, such as that the implied length is incorrect. It specifically does NOT mean that a data item submitted for storage in a register has a value outside the expectation of the application program, since the Modbus protocol is unaware of the significance of any particular value of any particular register.
04	SLAVE DEVICE FAILURE	An unrecoverable error occurred while the server (or slave) was attempting to perform the requested action.
05	ACKNOWLEDGE	Specialized use in conjunction with programming commands. The server (or slave) has accepted the request and is processing it, but a long duration of time will be required to do so. This response is returned to prevent a timeout error from occurring in the client (or master). The client (or master) can next issue a Poll Program Complete message to determine if processing is completed.
06	SLAVE DEVICE BUSY	Specialized use in conjunction with programming commands. The server (or slave) is engaged in processing a long-duration program command. The client (or master) should retransmit the message later when the server (or slave) is free.
08	MEMORY PARITY ERROR	Specialized use in conjunction with function codes 20 and 21 and reference type 6, to indicate that the extended file area failed to pass a consistency check.
0A	GATEWAY PATH UNAVAILABLE	Specialized use in conjunction with gateways, indicates that the gateway was unable to allocate an internal communication path from the input port to the output port for processing the request. Usually means that the gateway is misconfigured or overloaded.
0B	GATEWAY TARGET DEVICE FAILED TO RESPOND	Specialized use in conjunction with gateways, indicates that no response was obtained from the target device. Usually means that the device is not present on the network.

4.4 Modbus *FKT Codes*

Naming convention

Modbus has some naming conventions:



- Modbus differentiates between bit and word access; Bits = "Coils" and Words = "Register".
- Bit inputs are referred to as "Input-Status" and bit outputs as "Coil-Status".
- Word inputs are referred to as "Input-Register" and word outputs as "Holding-Register".

Range definitions

Normally the access with Modbus happens by means of the ranges 0x, 1x, 3x and 4x.

0x and 1x gives you access to *digital* bit areas and 3x and 4x to *analog* word areas.

For the Ethernet coupler from VIPA is not differentiating digital and analog data, the following assignment is valid:

0x - Bit area for master output

Access via function code 01h, 05h, 0Fh

1x - Bit area for master input

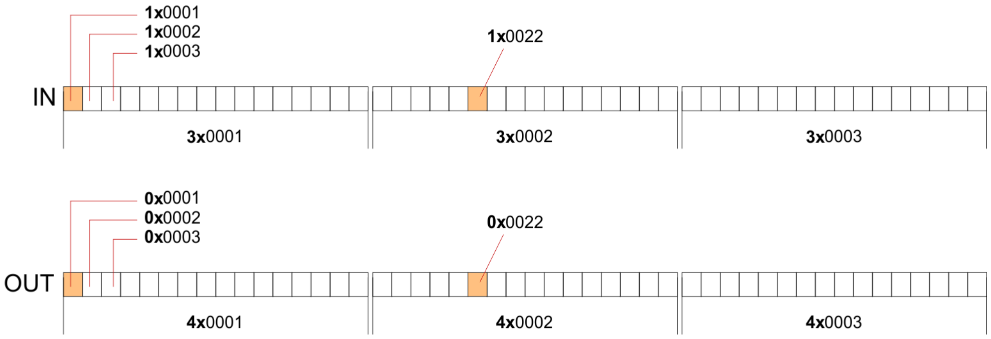
Access via function code 02h

3x - Word area for master input

Access via function code 04h

4x - Word area for master output

Access via function code 03h, 06h, 10h, 16h



Overview

With the following Modbus function codes a Modbus master can access a Modbus slave. The description always takes place from the point of view of the master:

Code	Command	Description
01h	Read n Bits	Read n bits of master output area 0x
02h	Read n Bits	Read n bits of master input area 1x
03h	Read n Words	Read n words of master output area 4x

Code	Command	Description
04h	Read n Words	Read n words master input area 3x
05h	Write 1 Bit	Write 1 bit to master output area 0x
06h	Write 1 Word	Write 1 word to master output area 4x
0Fh	Write n Bits	Write n bits to master area 0x
10h	Write n Words	Write n words to master area 4x
16h	Mask 1 Word	Mask 1 word in master output area 4x
17h	Write n Words and Read m Words	Write n words into master output area 4x and the respond contains m read words of the master input area 3x

Byte sequence in a word

1 word	
High byte	Low byte

Respond of the coupler

If the slave announces an error, the function code is sent back with a "OR" and 80h. Without an error, the function code is sent back.

Coupler answer:	Function code OR 80h	→ Error & error number
	Function code	→ OK

If the slave announces an error, the function code is sent back with a "OR" and 80h. Without an error, the function code is sent back.

01h: Function number is not supported

02h: Addressing errors

03h: Data errors

04h: System SLIO bus is not initialized

07h: General error

Read n Bits 01h, 02h

Code 01h: Read n bits of master output area 0x.

Code 02h: Read n bits of master input area 1x.

Command telegram

Modbus/TCP-Header						Slave address	Function code	Address1. bit	Number of bits
x	x	0	0	0	6				
6byte						1byte	1byte	1word	1word

Respond telegram

Modbus/TCP-Header						Slave address	Function code	Number of read bytes	Data 1. byte	Data 2. byte	...
x	x	0	0	0							
6byte						1byte	1byte	1byte	1byte	1byte	
									max. 252byte		

Read n words 03h, 04h

03h: Read n words of master output area 4x.

04h: Read n words master input area 3x.

Command telegram

Modbus/TCP-Header						Slave address	Function code	Address word	Number of words
x	x	0	0	0	6				
6byte						1byte	1byte	1word	1word

Respond telegram

Modbus/TCP-Header						Slave address	Function code	Number of read bytes	Data 1. word	Data 2. word	...
x	x	0	0	0							
6byte						1byte	1byte	1byte	1word	1word	
									max. 126words		

Write 1 bit 05h

Code 05h: Write 1 bit to master output area 0x.

A status change is via "Status bit" with following values:

"Status bit" = 0000h → Bit = 0

"Status bit" = FF00h → Bit = 1

Command telegram

Modbus/TCP-Header						Slave address	Function code	Address bit	Status bit
x	x	0	0	0	6				
6byte						1byte	1byte	1word	1word

Respond telegram

Modbus/TCP-Header						Slave address	Function code	Address bit	Status bit
x	x	0	0	0	6				
6byte						1byte	1byte	1word	1word

Write 1 word 06h

Code 06h: Write 1 word to master output area 4x.

Command telegram

Modbus/TCP-Header						Slave address	Function code	Address word	Value word
x	x	0	0	0	6				
6byte						1byte	1byte	1word	1word

Respond telegram

Modbus/TCP-Header						Slave address	Function code	Address word	Value word
x	x	0	0	0	6				
6byte						1byte	1byte	1word	1word

Write n bits 0Fh

Code 0Fh: Write n bits to master output area 0x.

Please regard that the number of bits are additionally to be set in byte.

Command telegram

Modbus/TCP-Header						Slave address	Function code	Address1 . bit	Number of bits	Number of bytes	Data 1. byte	Data 2. byte	...
x	x	0	0	0									
6byte						1byte	1byte	1word	1word	1byte	1byte	1byte	1byte
											max. 248byte		

Respond telegram

Modbus/TCP-Header						Slave address	Function code	Address 1. bit	Number of bits
x	x	0	0	0	6				
6byte						1byte	1byte	1word	1word

Write n words 10h

Code 10h: Write n words to master output area 4x.

Command telegram

Modbus/TCP-Header						Slave address	Function code	Address1 . word	Number of words	Number of bytes	Data 1. word	Data 2. word	...
x	x	0	0	0									
6byte						1byte	1byte	1word	1word	1word	1word	1word	1word
											max. 124byte		

Respond telegram

Modbus/TCP-Header						Slave address	Function code	Address 1. word	Number of words
x	x	0	0	0	6				
6byte						1byte	1byte	1word	1word

Mask a word 16h

Code 16h: This function allows to mask a word in the master output area 4x.

Command telegram

Modbus/TCP-Header						Slave address	Function code	Address word	AND Mask	OR Mask
x	x	0	0	0	8					
6byte						1byte	1byte	1word	1word	1word

Respond telegram

Modbus/TCP-Header						Slave address	Function code	Address word	AND Mask	OR Mask
x	x	0	0	0	8					
6byte						1byte	1byte	1word	1word	1word