

|  |
| --- |
| Lu Chunqiu |
| Software RTU of RIP |
| Operation and configuration instructions  Technical description |
|  |

|  |
| --- |
| Information Technology |
| 2013 |

**CONTENTS**

[1 General 5](#_Toc350038472)

[1.1 Software RTU features 5](#_Toc350038473)

[1.2 User interface 5](#_Toc350038474)

[2 operation of the Software RTU 6](#_Toc350038475)

[2.1 Software configuration 6](#_Toc350038476)

[2.1.1 Address map configuration 6](#_Toc350038477)

[2.1.2 Port configuration 7](#_Toc350038478)

[2.1.3 Other configuration 8](#_Toc350038479)

[2.2 Operation after start 8](#_Toc350038480)

# General

This first part (Operation and configuration) of this manual contains general descriptions of the functions and operation instructions. Also, it contains instructions for configuration of the RTU and instructions for changing settings as well as the operation after start.

## Software RTU features

This Software RTU runs on Linux Ubuntu 10.04 LTS whose programming IDE is Qt XXX. It can convert the IEC-60870-101(hereafter referred to as IEC101) frame from controlling station to Modbus RTU (hereafter referred to as Modbus) frame and convert the returning Modbus frame back to IEC101 frame. The following table lists the IEC101 and Modbus frame it supports.

|  |  |  |
| --- | --- | --- |
| **Protocol Name** | **Function(code)** | **Remark** |
| IEC101 | Data link reset(0x0) |  |
| Data link status request(0x9) |  |
| Class 1 data request(0xA) | When it is general data request, it returns 3 sequential addresses value and two separate addresses value. namely 2,3,4 the 3 current value, 0x1001 the breaker status and 0x10 the local/remote status address |
| Class 2 data request(0xB) | The same to Class1 |
| Clock synchronization(0x67) |  |
| Read data(0x66) |  |
| Single command(0x2D) |  |
| Double command(0x2E) |  |
| Interrogation(0x64) |  |
| Modbus | Read holding register(0x3) |  |
| Write single register(0x6) |  |

Table 1. IEC101 & Modbus frame that the RTU supports

## User interface

The entire configuration can be done at the user interface as the following. It can

* Choose the port name
* Configure the port parameter
* Start/Close the port
* Count the receiving frame number
* Illustrate the receiving and send frame in hexadecimal format

# operation of the Software RTU

## Software configuration

Figure 1 shows the user interface of the RTU. The following part will introduce the configuration and operation in detail.

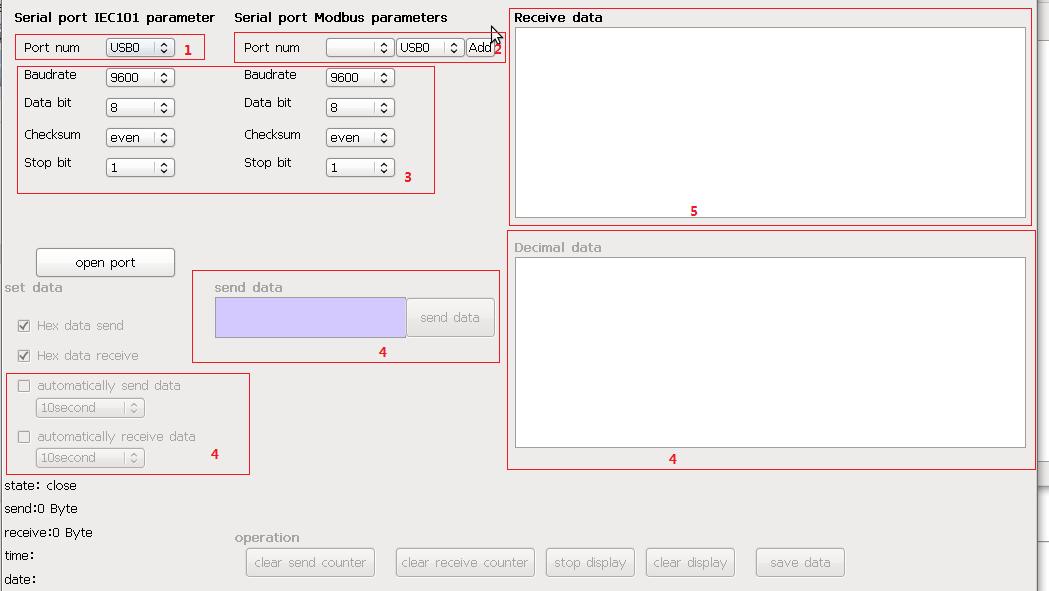


Figure 1. User interface of RTU

### Address map configuration

Figure 2 shows the address map menu by clicking the Param->Port config and File->Open mySerialPortTools/project.xbel. The root of each unit is the name of the relays in the subnet. The first column of each unit is the address name. The second column is the IEC101 address in decimal format. The third address is the Modbus address in decimal format.

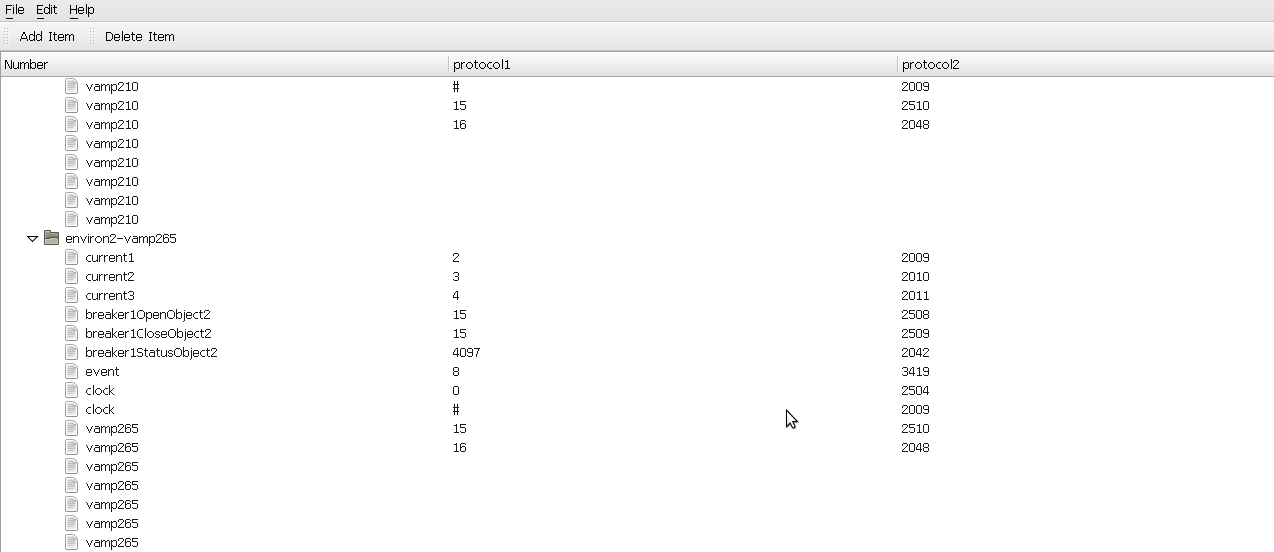


Figure 2. Address map menu

The IEC101 and Modbus address can be changed at the menu. If the address name needs changed, it can only be done at the source mySerialPortTools/project.xbel file. When changing the second or third column which is the address, click the corresponding cell and changes the value. After that click File->save in the top menu and replace the original project.xbel file.

### Port configuration

As the software is installed on PC, the user is recommended to use the USB port.

To find the right port, first plug the controlling station cable to one USB port and enter *dmeg | grep tty* in the console. In the result port list, if one port is connecting now, it is the controlling station port, shown in Figure 3. Then choose the corresponding port name in the *Port num* drop-down box of *Serial port IEC101 parameters*, shown in highlight 1 of Figure 1. For the Modbus (relay) port, it is quite similar and the corresponding port is the second port which shows connecting in the result port list after plugging the relay cable, shown in Figure 4. When choosing the Modbus port name in the *Serial port Modbus parameters*, choose the corresponding name in the right box and press *add*, then the left box will show the chosen port, shown in highlight 2 of Figure 1. Click the *add* button, when the current unit in the right box is now the existing unit in the left box, will remove this unit out of the left box.

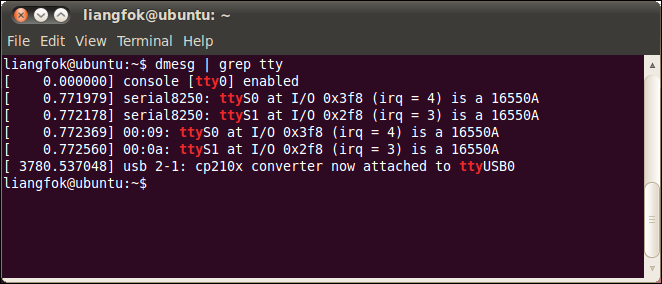


Figure 3. Command result

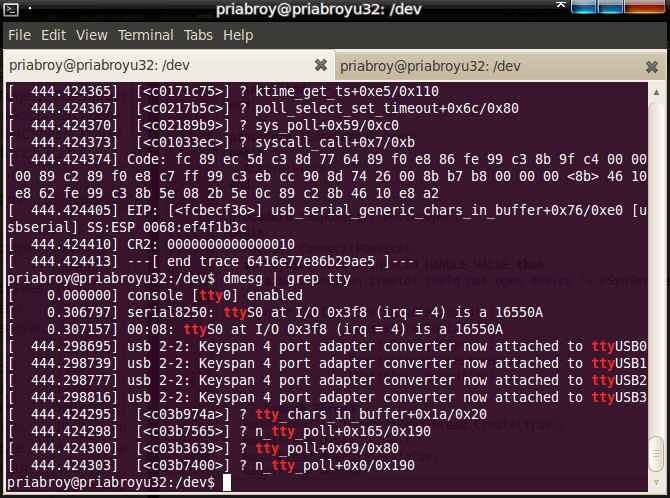


Figure 4. Command result

To configure the serial parameter, choose the proper parameter in the highlight 3 of Figure 1.

### Other configuration

For the sake of stability, do not change or press the other configuration in the user interface. Leave hex data receive clicked, it will convert the receiving frame to hexadecimal format and also will show the frame in the right receive data. Also, button *clear display* can clean all the shown frames in the display screen.

The highlight 4 of Figure 1 only use for debug, so it is not necessary to use it.

## Operation after start

When pressing the *open port* button, the RTU starts work. It will show the receiving and send frame in hexadecimal format as well as the corresponding port name in the right receive data part in highlight 5 of Figure 1.

Before closing the program, please first close the port.

Content

[1 Environment establish 10](#_Toc350038524)

[1.1 Essential equipment 10](#_Toc350038525)

[1.2 Environment configuration 11](#_Toc350038526)

[2 Archetecture of the software 11](#_Toc350038527)

[2.1 Software architecture 11](#_Toc350038528)

[2.2 Software programming description 15](#_Toc350038532)

[3 IEC101& Modbus introduction 16](#_Toc350038524)

[3.1 IEC101 16](#_Toc350038525)

[3.2 Modbus……..……………………………………………………………21](#_Toc350038526)

[4 Possible problem and solution 24](#_Toc350038524)

[5 Future improvement 25](#_Toc350038524)

[6 RTU NO-UI VERSION AND EMBEDDED VERSION 25](#_Toc350038524)

# 1 ENVIRONMENT ESTABLISH

This part of the user manual describes more technical detail of the software RTU. It contains the environment establish process, software architecture, protocol introduction and possible problem as well as the future improvement.

Figure 5 shows the application diagram of this project. RTU acts as the connection between the remote control computer and distribution substation. It converts the remote control protocol to local control protocol.

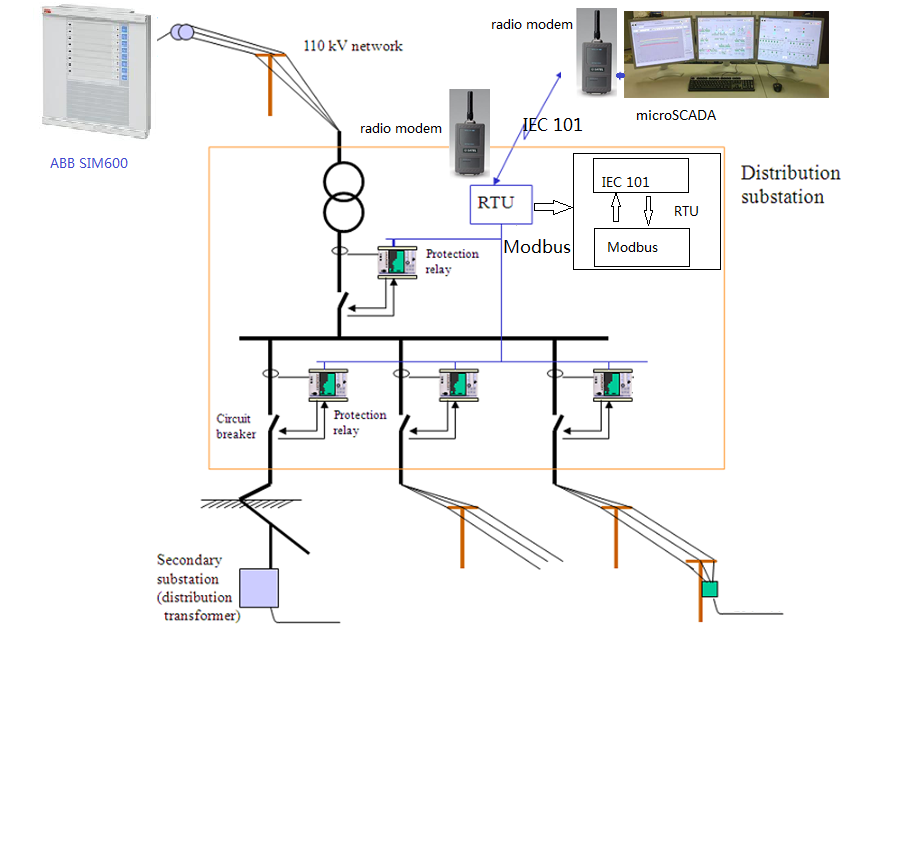
****

Figure 5. Application diagram

## 1.1 Essential equipment

To establish the environment, the following table lists the essential equipment

|  |  |
| --- | --- |
| **Item Name** | **Remark** |
| Ubuntu 10.04 |  |
| Qt 4.6.3 | Qt creater, designer |
| Computer with at list 2 USB port |  |

Table 2. Essential equipment

## 1.2 Environment configuration

To configure the whole system, first install the Ubuntu on PC and then install the Qt on the Ubuntu. After that, put the whole project file mySerialPortTools to home folder. Click the mySerialPortTools.pro to open the whole project and compile the project. Then the compile result will show the user interface normally. Figure 6 shows the environment establishment process and table 3 shows the summary of setting.

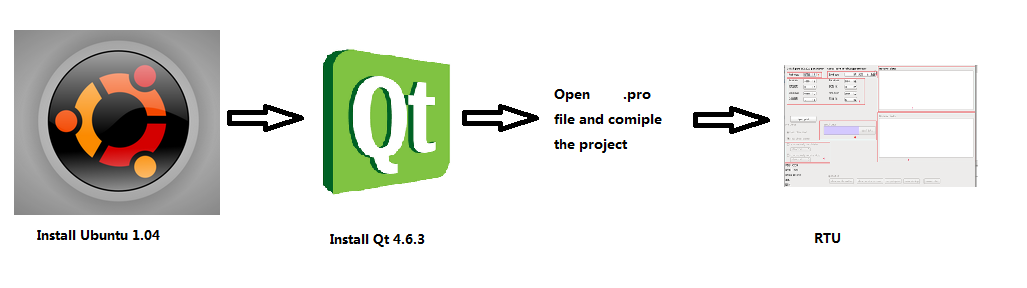


Figure 6. Environment establishment process

|  |  |
| --- | --- |
| **Item name** | **Remark** |
| Address map | Param->port config->open project.xbel |
| Port Name | dmeg | grep tty |
| Serial parameter | data rate, data length, check bit, stop bit |

Table 3. Summary of setting

# 2 ARCHETiCTURE OF THE SOFTWARE

## 2.1 Software architecture

The software is a tool that can convert the IEC101 frame from the controlling station port to Modbus frame and send to relay port. Also, it can convert the reply Modbus frame back to IEC101 frame. Therefore, a good data process design and IEC101 & Modbus converting design is the key point of this software. Figure 7 demonstrates the general design of RTU.

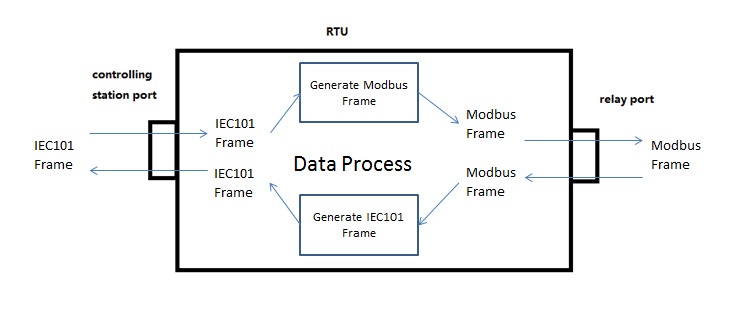


Figure 7. General design of RTU

The following figures demonstrate the detail architecture of this software. Figure 8 shows the data process diagram of the software.

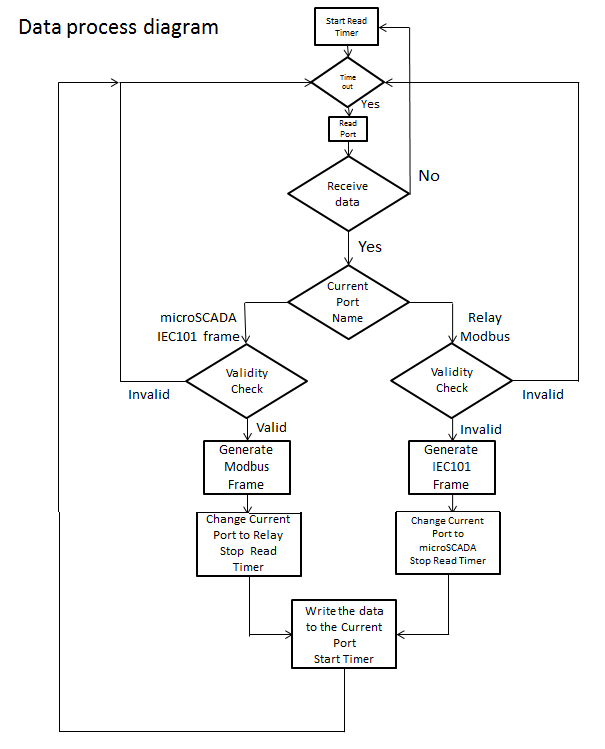


Figure 8. Data process diagram

Figure 9 demonstrate the Modbus frame generating process when receiving the IEC101 frame from the controlling station port.

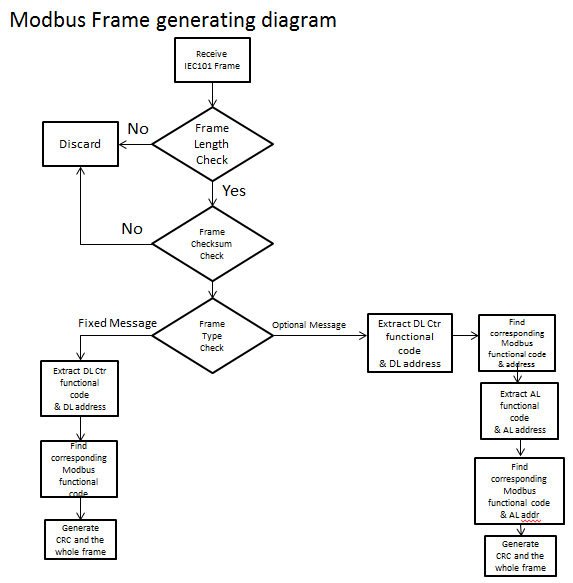


Figure 9. Modbus frame generating process diagram

Figure 10 demonstrate the IEC101 frame generating process diagram when receiving the Modbus frame from the relay port.

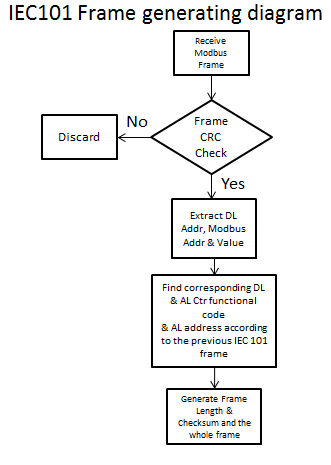


Figure 10. IEC 101 frame generating process diagram

## 2.2 Software programming description

This part introduces the detail programming of the software.

In the project file, contains several header files and corresponding cpp files as well as the other file. The following table shows the function of each file.

|  |  |  |
| --- | --- | --- |
| **Module** | **File name** | **Function** |
| Main UI | frmMain.ui | UI design file |
| frmMain.h/frmMain.cpp | UI function implement file as well as data process implement |
| Menu UI | dialog.ui | Menu(Address Map) UI design file |
| dialog.h/dialog.coo | Menu UI implement file |
| mainwindow.h/mainwindow.cpp | Menu call implement file |
| xbeltree.h/xbeltree.cpp | xbel analysis implement file |
| IEC101 analysis | iec101analyser.h/.cpp | IEC101 frame analysis implement file |
| Modbus analysis | modbusanalyser.h/.cpp | Modbus analysis implement file |
| Port receive and send | qextserialenumerator.h/.cpp | basic serial port implement |
| qextserialbase.h/.cpp | basic serial port implement |
| posix\_qextserialport.h/.cpp | serial port implement |

Table 4. software programming description

For more detail description, please refer to the comment.

# IEC101& MODBUS INTRODUCTION

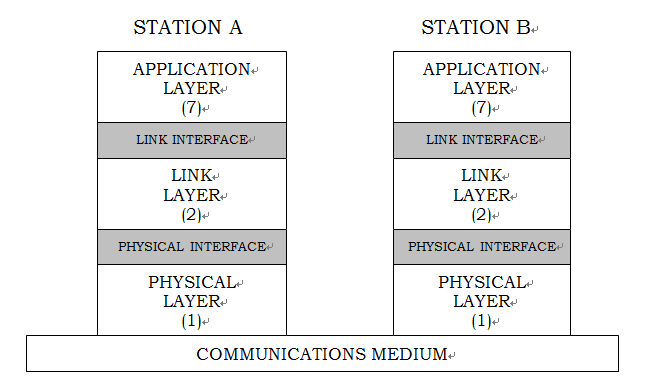
## IEC101

IEC 60870-5-101(also IEC 101) is the remote control and monitor protocol standard for the design of electrical substation automation prepared by the International Electrotechnical Commission’s(IEC). The main goal of this standard is to provide interoperability between IEDs from different suppliers.

The following are some of the important components and characteristics of IEC 101.

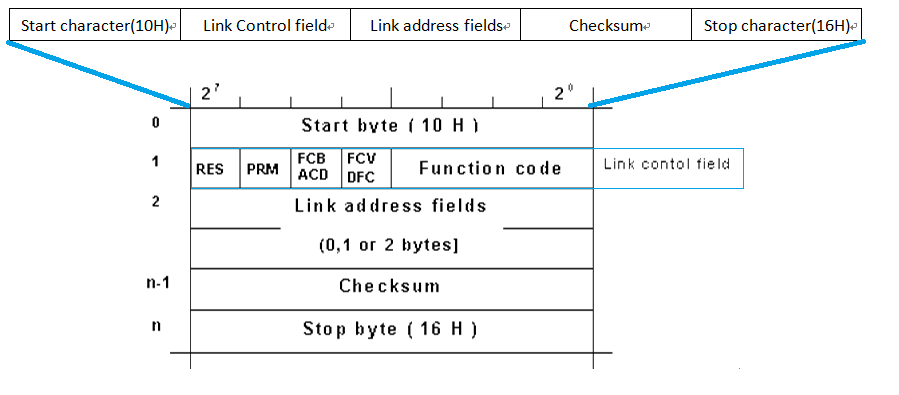
**Data Modeling**

IEC 101 communicates on three layers based on the OSI model, Application Layer, Link Layer and Physical Layer.

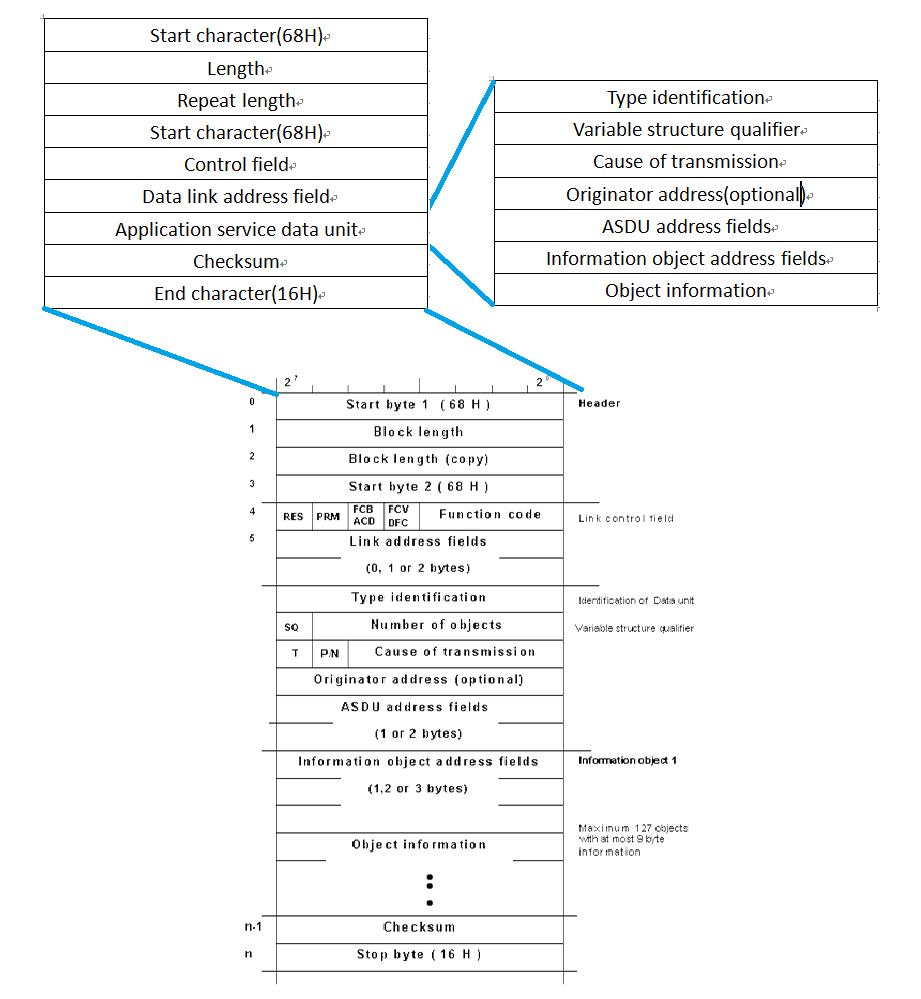


**Figure 11. IEC 101 model**

For the data structure, the IEC 101 has two kinds of frame, fixed length frame and optional length frame. Figure 12 and 13 demonstrates the data structure of the IEC 101 frame.

****

**Figure 12. Fixed length frame**

****

**Figure 13. Optional length frame**

**Communication service**

* **Link layer**

There are two kinds of transmission method, unbalanced and balanced. When using unbalanced transmission, the controlling station controls the data traffic by polling the controlled outstations sequentially. In this case, the controlling station initiates all the message transfers while the controlled outstations can transmit only in response to the message from the controlling station. In other words, in unbalanced transmission, only the controlling station can start the diagram to controlled station and controlled station can response only after receiving the request frame from the controlling station.

When using balanced transmission, each station can initiate message transfer. The stations may act simultaneously as controlling stations and controlled outstations. Therefore, they are called combined stations.

What should be mentioned here is that, VAMP relay can only support unbalanced transmission.

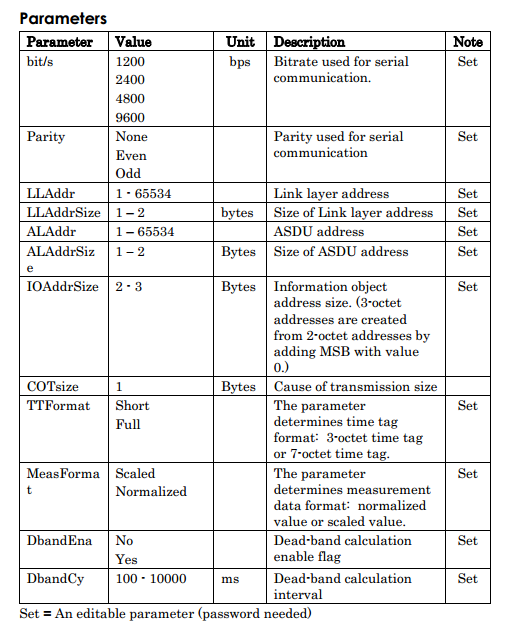
* **Application layer**

The following listed some service in application layer.

* Station Initialisation
* Data acquisition by polling
* Cyclic data transmission
* Acquisition of events
* General interrogation
* Clock synchronisation
* Command transmission
* Transmission of Integrated Totals
* Parameter loading
* Test procedure
* File transfer (for simple files)
* Acquisition of transmission time delay

**Communication Parameter**

According to the data structure above, there are some communication parameters in the protocol. Also, as the transmission media is serial port, serial port parameters also need to be considered in communication. Figure 14 shows the parameters in the protocol.



**Figure 14. Transmission parameter**

**RTU support parameter**

The following table shows the IEC101 function that this RTU supports.

|  |  |
| --- | --- |
| **Function(code)** | **Remark** |
| Data link reset(0x0) |  |
| Data link status request(0x9) |  |
| Class 1 data request(0xA) | When it is general data request, it returns 3 sequential addresses value and two separate addresses value. namely 2,3,4 the 3 current value, 0x1001 the breaker status and 0x10 the local/remote status address |
| Class 2 data request(0xB) | The same to Class1 |
| Clock synchronization(0x67) |  |
| Read data(0x66) |  |
| Single command(0x2D) |  |
| Double command(0x2E) |  |
| Interrogation(0x64) |  |

Table 5. Supporting IEC101 function

The following table demonstrates the transmission parameter that this RTU supports.

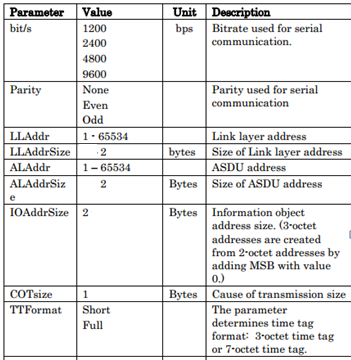


Table 6. Supporting transmission parameter

## Modbus\*

Modbus is a serial communications protocol originally published by Modicon (now Schneider Electric) in 1979 for use with its programmable logic controllers (PLCs). Simple and robust, it has since become a de facto standard communication protocol, and it is now a commonly available means of connecting industrial electronic devices. The main reasons for the use of Modbus in the industrial environment are:

* It has been developed with industrial applications in mind
* It is openly published and royalty-free
* It is easy to deploy and maintain
* It moves raw bits or words without placing many restrictions on vendors

**Protocol versions**

Versions of the Modbus protocol exist for serial port and for Ethernet and other networks that support the Internet protocol suite. There are many variants of Modbus protocols:

* Modbus RTU — This is used in serial communication & makes use of a compact, binary representation of the data for protocol communication. The RTU format follows the commands/data with a cyclic redundancy check checksum as an error check mechanism to ensure the reliability of data. Modbus RTU is the most common implementation available for Modbus. A Modbus RTU message must be transmitted continuously without inter-character hesitations. Modbus messages are framed (separated) by idle (silent) periods.
* Modbus ASCII — This is used in serial communication & makes use of ASCII characters for protocol communication. The ASCII format uses a longitudinal redundancy check checksum. Modbus ASCII messages are framed by leading colon (':') and trailing newline (CR/LF).
* Modbus TCP/IP or Modbus TCP — This is a Modbus variant used for communications over TCP/IP networks, connecting over port 502. It does not require a checksum calculation as lower layers already provide checksum protection.
* Modbus over TCP/IP or Modbus over TCP or Modbus RTU/IP — This is a Modbus variant that differs from Modbus TCP in that a checksum is included in the payload as with Modbus RTU.
* Modbus over UDP — Some have experimented with using Modbus over UDP on IP networks, which removes the overheads required for TCP
* Modbus Plus (Modbus+, MB+ or MBP) Modbus over Fieldbus (Modbus+ or MB+), also exists, but remains proprietary to Schneider Electric. requires a dedicated co-processor to handle fast HDLC-like token rotation. It uses twisted pair at 1 Mbit/s and includes transformer isolation at each node, which makes it transition/edge triggered instead of voltage/level triggered. Special interfaces are required to connect Modbus Plus to a computer, typically a card made for the ISA (SA85), PCI or PCMCIA bus.
* Modbus PEMEX- This variant is an extension of standard Modbus with support for historical and flow data. It was designed for process control and never gained widespread adoption
* Enron Modbus- This variant is an extension of standard Modbus with support for 32 bit Integer and Floating Point variables, and historical and flow data. Data types are mapped using standard addresses. The historical data is used to meet an American Petroleum Institute (API) industry standard for how data should be stored

**Frame format**

Figure 15 shows the Modbus frame format.

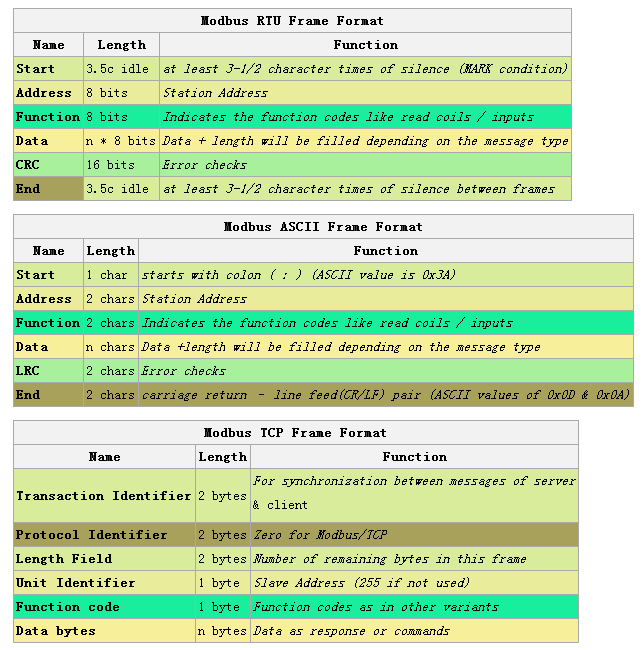


Figure 15. Modbus frame format

**RTU support parameter**

The following table shows the function that this RTU supports.

|  |  |
| --- | --- |
| **Function name** | **Function code** |
| Read holding register | 0x3 |
| Write single register | 0x6 |

Table 7. Function that this RTU supports

\* From http://en.wikipedia.org/wiki/Modbus

# POSSIBLE PROBLEM AND SOLUTION

|  |  |  |
| --- | --- | --- |
| **Problem** | **Possible reason** | **Possible solution** |
| Nothing receive | Check if the port name is chosen right. | Refer to operation and configuration manual 2.1.2. |
| Check if the RTU starts | Start the RTU |
| Check if now display is stopped | Enable display |
| Check if the serial transmission parameter is correct | Keep the RTU and connecting device parameter the same |
| Frame process accidently stops | Check if the configuration of corresponding relay is right. | Configure the relay so that both the transmission parameter is the same |
| Check if the map address of the receiving frame is right | Refer to operation and configuration manual 2.1.1 |
| Other problem | RTU bug | Restart the RTU |

# FUTURE IMPROVEMENT

|  |  |
| --- | --- |
| **Current Defect** | **Solution** |
| RTU can only work when it receives the response frame from each relays. | Add a timer counting for the relay response time. When timeout happens, it responses to controlling station that the link has no response. |
| RTU cannot process the Modbus error message from relay. | Improve the function of Modbus analyzer file. |
| RTU can only support the IEC 101 function in this project. | Improve the function of IEC101 analyzer file. |
| RTU open only one port each time, so it will lose some frame of the closing port. | Implement the software in multithread. So it can process the two ports simultaneously. |
| The whole software programming structure is not clear and difficult for a new programmer to check and improve. | Improve the overall structure. |

# RTU NO-UI VERSION AND EMBEDDED VERSION

In this project, there are another two RTU versions, no-UI version and embedded version. For the no-UI version, it just removes all the UI part and the function and manual is completely the same with the UI version. For the embedded version, it is quite similar to the no-UI version. As this version uses on NGW100 board, the only difference is a few functions which is replaced by other functions so it can work on NGW board. Therefore, no-UI version and embedded version is nearly the same.