SDLC-BOARD

Datasheet

Email : yacer@yacer.cn
Website : www.yacer.cn

yacer 亚册
Building Blocks of Communication

Table of Contents

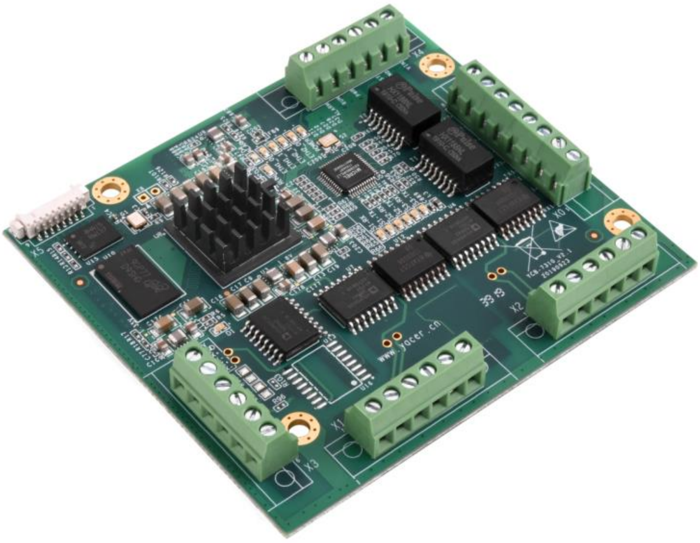
1 Overview

1.1 Introduction

The Yacer SDLC-BOARD Embedded development board provides HDLC synchronous full feature serial ports, asynchronous UART, CAN bus and Ethernet interface to achieve the protocol conversion between different interfaces.

It adopts small size and terminal interface design, which is especially convenient for users to realize system integration.

The on-board application CPU is used by users for secondary development to realize seamless integration of business software and communication software.



1.2 Applications

- Conversion between synchronous and asynchronous serial ports
- Protocol conversion between the serial port and Ethernet
- Conversion between CAN bus and serial port
- Conversion between CAN bus and Ethernet
- Train Control and Monitoring System (TCMS), Train Communication Network (TCN)
- Telemetry, measurement and control data acquisition and transmission
- Satellite, radio data transmission
- Air Traffic Control (ATC), Air Traffic Management (ATM)
- Embedded application and development

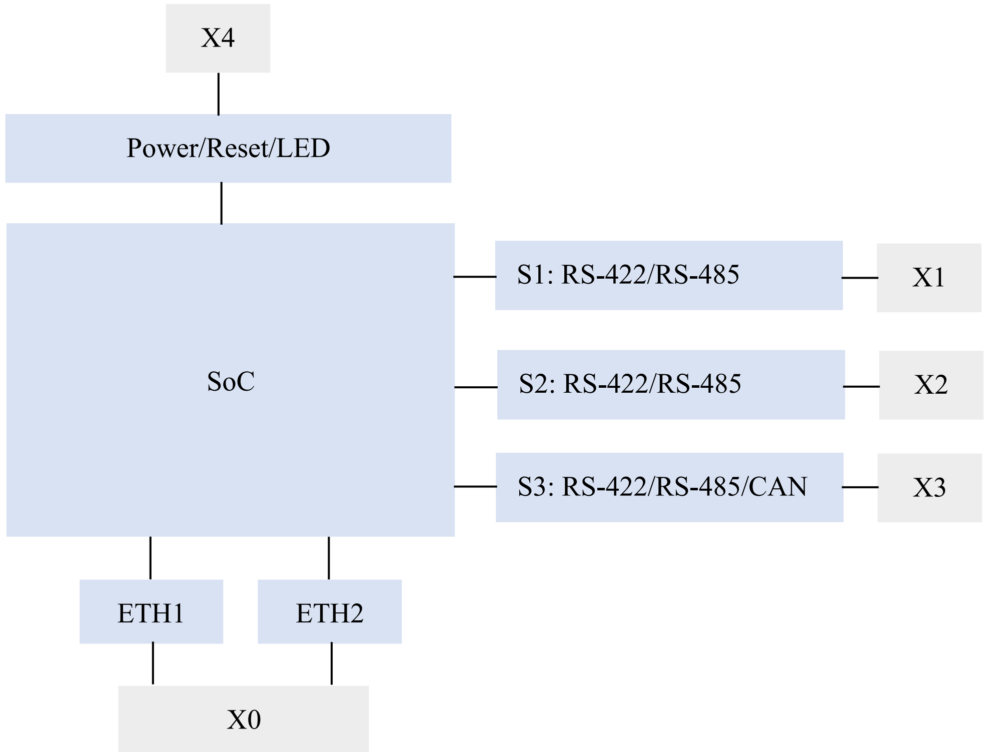
1.3 Features

- Dual 10/100M Ethernet, support Ethernet switching and dual IP function
- Dual synchronous/asynchronous serial port with isolation, RS-422 full-duplex and RS-485 half-duplex optional
- One extension interface with isolation, optional RS-422, RS-485 serial port or CAN bus interface
- All serial ports support synchronous HDLC or asynchronous UART working mode
- The coding formats support NRZ, NRZI, DBPL, Manchester and Diff-Man
- On-board application CPU provides secondary development for users
- Complete isolation protection, industrial wide temperature
- Easy integration: small size, terminal interface, 5 ~ 17V wide voltage power supply

1.4 Functions

Basic functions are as follows:

- X0: Dual 100M Ethernet interface ETH1 and ETH2
- X1: Synchronous/asynchronous serial port S1, RS-422 or RS-485 optional
- X2: Synchronous/asynchronous serial port S2, RS-422 or RS-485 optional
- X3: Extension serial port S3, RS-422, RS-485 or CAN bus optional
- X4: Power, Reset and LED indicators



1.5 Technical Specifications

Synchronous/asynchronous Serial Port S1,S2	
Connector	3.81mm terminal
Optional Interface (1 of 2 options)	RS-422 Full-duplex serial port with isolation RS-485 Half-duplex serial port with isolation
Working mode	Synchronous HDLC, Asynchronous UART, Bit stream
Encoding format	NRZ, NRZI, DBPL (Differential Bi-Phase Level), Manchester, Diff-Man (Differential Manchester)
Baud rate	Synchronous NRZ ≤ 12 Mbps; Other synchronous ≤ 6 Mbps Asynchronous ≤ 3 Mbps
Isolation protection	2.5 kVrms
Extension serial Port S3	
Connector	3.81mm terminal
Optional Interface (One output of three)	CAN Bus with isolation (CAN2.0A, CAN2.0B, ISO 11898) RS-422 Full-duplex serial port with isolation RS-485 Half-duplex serial port with isolation
Working mode	Synchronous HDLC, Asynchronous UART, Bit stream
Encoding format	NRZI, DBPL, Manchester, Diff-Man
Baud rate	Serial port : Synchronous ≤ 6 Mbps; Asynchronous ≤ 3 Mbps CAN: 50 Kbps ~ 1 Mbps
Isolation protection	2.5 kVrms
Ethernet interface ETH1, ETH2	
Connector	3.81mm terminal
Function	Ethernet switching, Dual IP function
Speed	10/100 Mbps, Auto MDI/MDI-X
Network protocol	TCP/IP
Programming interface	UDP Server, UDP Client Unicast/Multicast/Broadcast
Isolation protection	1.5 kVrms
Secondary development support	
CPU	ARM Cortex-A9 processor, main frequency 250 MHz
Memory	DDR3, 128MB
FLASH	Version space 6MB, Configuration space 1MB

Interface	Data interaction based on shared memory with CPU
Configuration Management	
Configuration tool	yacer-DMS configuration management software
Configuration interface	Ethernet Interface DMS-UART interface (rely on yacer's DMS-UART-8P configure cable)
Power Requirements	
Input voltage	4.9 ~ 17 VDC
Power consumption	< 3 W
Power interface	3.81mm terminal
Mechanical Characteristics	
Dimensions	H x W: 90 mm x 80 mm
Weight	100 g
Operating Environment	
Operating temperature	-40 ~ +75℃
Storage temperature	-40 ~ +85℃
Operating humidity	5 ~ 95% RH (no condensation)

1.6 Order Information

SDLC-BOARD	-	5	5	0	-	LG
X1interface definition:						
● RS-422 Full-duplex		4				
● RS-485 Half-duplex		5				
X2 interface definition:						
● RS-422 Full-duplex			4			
● RS-485 Half-duplex			5			
X3 interface definition:						
● None				0		
● RS-422 Full-duplex				4		
● RS-485 Half-duplex				5		
● CAN Bus				6		
Connection mode:						
● Terminal direct wiring						LG
● Horizontal connection						RM
● Vertical connection						VM

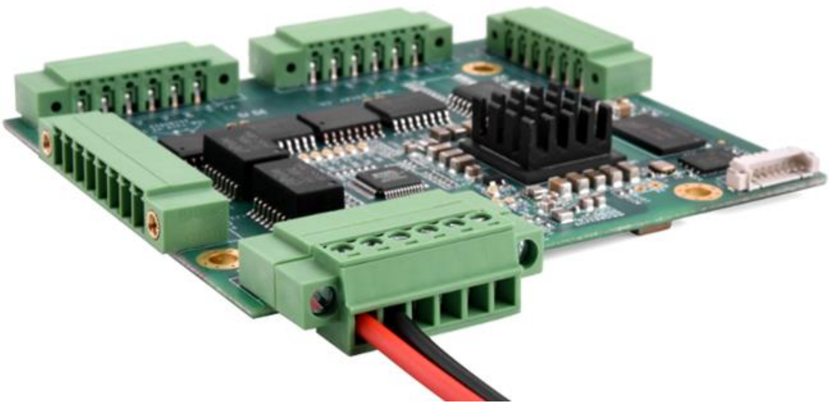
2 Hardware and Physical Interface

2.1 Connection mode

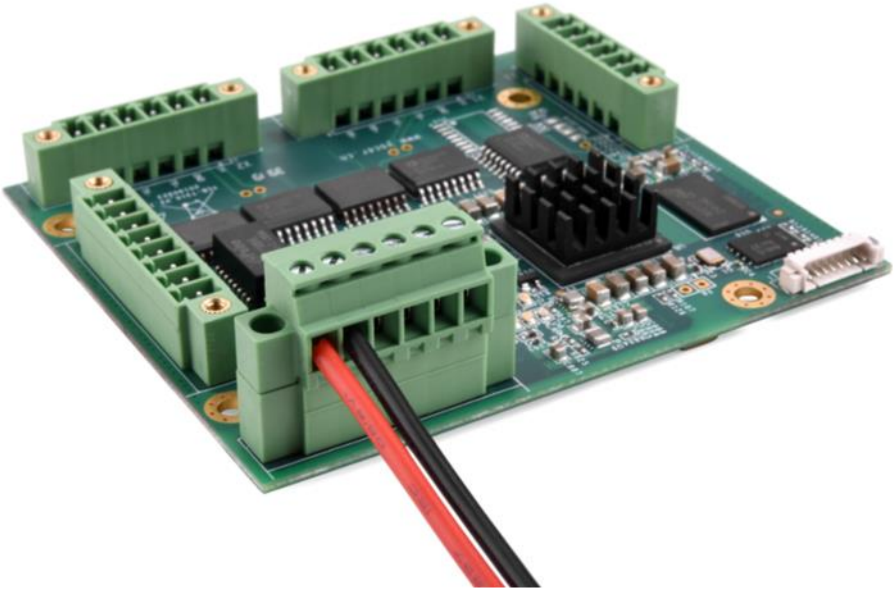
In order to adapt to different application scenarios, SDLC-BOARD provides three different wiring modes.

2.1.1 LG: Terminal direct wiring

2.1.2 RM: Horizontal connection



2.1.3 VM: Vertical connection



2.2 LED Indicators

LED	Description
POWER	Power indicator, constantly on after powered
RUN	Running indicator, flashing during normal operation
ALARM	Alarm indicator, flashing during initialization Light on during operation indicates the device failure
ETH0_0	LINK/ACT indicator of Ethernet EHT1
ETH0_1	Speed indicator of Ethernet EHT1, constantly on when speed up to 100M/s
ETH1_0	LINK/ACT indicator of Ethernet EHT2
ETH2_1	Speed indicator of Ethernet EHT2, constantly on when speed up to 100M/s
LINK/ACT	LINK/ACT indicator of 2 Ethernet interface
S1-RX	Rx indicator of S1, flashing during received a frame of data
S1-TX	Tx indicator of S1, flashing during transmitted a frame of data
S2-RX	Rx indicator of S2, flashing during received a frame of data
S2-TX	Tx indicator of S2, flashing during transmitted a frame of data
S3-RX	Rx indicator of S3, flashing during received a frame of data
S3-TX	Tx indicator of S3, flashing during transmitted a frame of data

2.3 Ethernet interface ETH1, ETH2: X0

2.3.1 Function description

ETH1, ETH2 are dual 10/100M Ethernet interfaces. There are two working modes of dual Ethernet interface:

- Ethernet switching mode: Enable built-in Ethernet switching function;
- Dual IP function mode: Enable each Ethernet interface has an independent IP address.

2.3.2 X0 Pin Definition

X0 pin is defined as follows :

X0 Pin	Signal
1	ETH1_RX-
2	ETH1_RX+
3	ETH1_TX-
4	ETH1_TX+
5	ETH2_RX-
6	ETH2_RX+
7	ETH2_TX-
8	ETH2_TX+

2.4 Synchronous/asynchronous serial port S1: X1

2.4.1 Function description

Serial port S1 supports synchronous SDLC/HDLC protocol and asynchronous UART mode, encoding format supports NRZ, NRZI, Manchester, differential Manchester, DBPL, etc.

Users can choose one of the following types when ordering:

- RS-422 Full-duplex with isolation
- RS-485 Half-duplex with isolation

2.4.2 No clock mode X1 Pin definition

While Serial port S1 operates in the following modes, it is in no clock mode:

- HDLC-NRZI
- HDLC-MAN
- HDLC-DiffMAN
- DBPL
- UART
- UART-HDLC

No clock mode only has data signal, X1 pin is defined as follows:

X1 Pin	RS-422 Full-duplex	RS-485 Half-duplex
1	5V_OUT	5V_OUT
2	GND	GND
3	TxD +	Data +
4	TxD -	Data -
5	RxD +	Term +
6	RxD -	Term -

Notice: 5V_OUT is +5VDC output, can use for Bus pull-up only, can NOT use for extension power supply. Hang up when not in use.

2.4.3 Clock mode X1+X2 pin definition

While Serial port S1 operates in the following modes, it is in clock mode:

- HDLC-NRZ
- Bit stream

X1 and X2 work together to build a complete working signal with clock, include data signal and clock signal, and the pin is defined as follows:

Pin	RS-422 Full-duplex	RS-485 Half-duplex
X1-1	5V_OUT_1	5V_OUT_1
X1-2	GND_1	GND_1
X1-3	TxD +	Data +
X1-4	TxD -	Data -
X1-5	RxD +	Data_Term +
X1-6	RxD -	Data_Term -
X2-1	5V_OUT_2	5V_OUT_2
X2-2	GND_2	GND_2
X2-3	TxC +	Clock +
X2-4	TxC -	Clock -
X2-5	RxC +	Clock_Term +
X2-6	RxC -	Clock_Term -

Notice:

- While X1 and X2 work together, S2 serial port number is invalid;
- Users need to short-circuit GND-1 and GND-2

2.4.4 RS-485 Terminal matching

In RS-485 mode, if terminal matching is necessary, the matching resistor should be straddled to the Term+ and Term- pins.

2.5 Synchronous/asynchronous serial port S2: X2

2.5.1 Function description

Serial port S2 supports synchronous SDLC/HDLC protocol and asynchronous UART mode, encoding format supports NRZI, Manchester, differential Manchester, DBPL, etc.

Users can choose one of the following types when ordering:

- RS-422 Full-duplex with isolation
- RS-485 Half-duplex with isolation

2.5.2 X2 pin definition

X2 pin is defined as follows：

X2 Pin	RS-422 Full-duplex	RS-485 Half-duplex
1	5V_OUT	5V_OUT
2	GND	GND
3	TxD +	Data +
4	TxD -	Data -
5	RxD +	Term +
6	RxD -	Term -

Notice: 5V_OUT is +5VDC output, can use for Bus pull-up only, can NOT use for extension power supply. Hang up when not in use.

2.5.3 RS-485 Terminal matching

In RS-485 mode, if terminal matching is necessary, the matching resistor should be straddled to the Term+ and Term- pins.

2.6 Extension Interface S3: X3

2.6.1 Function description

Serial port S3 is extension interface, and users can choose serial port or CAN Bus interface at the factory. When configured as a serial port, one of the following configurations can be selected at the factory:

- RS-422 Full-duplex with isolation
- RS-485 Half-duplex with isolation

Serial port supports synchronous SDLC/HDLC protocol and asynchronous UART mode, encoding format supports NRZI, Manchester, differential Manchester, DBPL, etc.

2.6.2 X3 pin definition

X3 pin is defined as follows：

X3 Pin	CAN	RS-422 Full-duplex	RS-485 Half-duplex
1	5V_OUT	5V_OUT	5V_OUT
2	GND	GND	GND
3	CAN_H	TxD +	Data +
4	CAN_L	TxD -	Data -
5	Term +	RxD +	Term +
6	Term -	RxD -	Term -

Notice: 5V_OUT is +5VDC output, can use for Bus pull-up only, can NOT use for extension power supply. Hang up when not in use.

2.6.3 CAN Bus terminal matching

In CAN Bus mode, if terminal matching is necessary, the matching resistor should be straddled to the Term+ and Term- pins

2.6.4 RS-485 Terminal matching

In RS-485 mode, if terminal matching is necessary, the matching resistor should be straddled to the Term+ and Term- pins

2.7 X4: Power/Reset/LED interface

X4 pin definitions are as follows:

X4 Pin	Signal	Description
1	POWER_IN	Power input, +4.9 ~ 17VDC
2	GND	Ground
3	Reset	Module reset, it can reset the board, active low.
4	LED_POWER	Power supply working indicator, low-level enable
5	LED_RUN	System running indicator, low-level enable
6	LED_ALARM	System alarm indicator, low-level enable

2.8 X5: DMS-UART Configuration management interface

This interface is used to connect the yacer's DMS-UART-8P configuration cable, which can be used for configuration management through the USB interface of PC.

3 Building Configuration Environment

3.1 How to get configuration management software yacer-DMS

Users can obtain the configuration management software package, yacer-DMS.zip, by two ways as follow:

- "Tools" directiory of the USB-Disk attached with SDLC-BOARD
- The software channel of the official website http://en.yacer.cn/softwares/det/yacer-dms

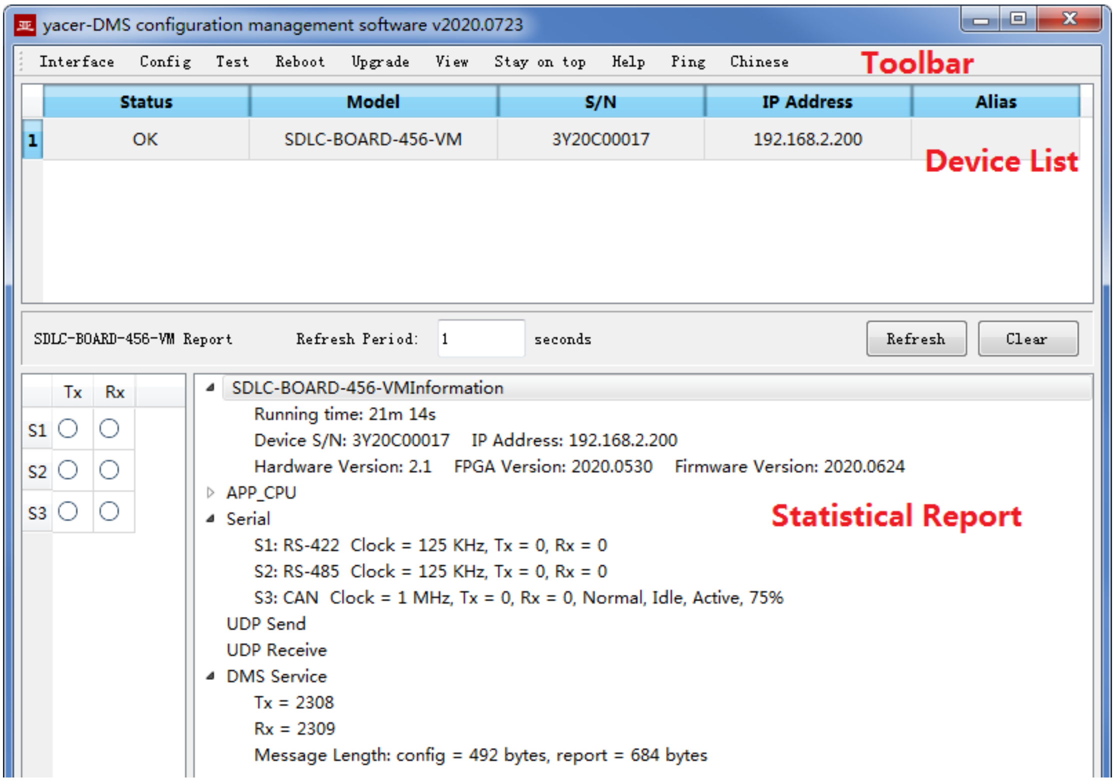
3.2 Run software yacer-DMS

The yacer-DMS is a portable software, to unzip the yacer-DMS.zip and then enter the directory yacer-DMS and then run the execute file yacer-DMS.exe.

3.3 Main Window of yacer-DMS Software

Below is the main window of the configuration and management software yacer-DMS, including three parts:

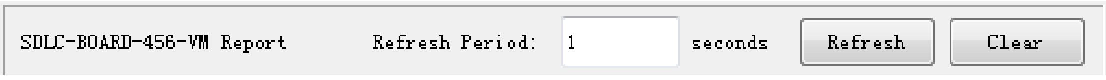
- Toolbar: Function operation buttons;
- Device List: Displaying the basic information and running status of the on-line device;
- Statistical Report: Displaying the receive/transmit indication & statistics, and device details of the specified device.



3.4 Statistical Report

The statistical report has three panels: control panel, receive/transmit indication panel, information panel.

3.4.1 Control Panel



Control Widget	Function
Refresh Period: 1 seconds	Set the refresh period of report
Refresh	Manually refresh the statistical report
Clear	Clear the statistical report

3.4.2 Receive/transmit indication

- Transmit: Each time the corresponding serial port transmits one frame data, the transmit indicator flashes once.
- Receive: Each time the corresponding serial port receives one frame data, the receive indicator flashes once.

3.4.3 Information Panel

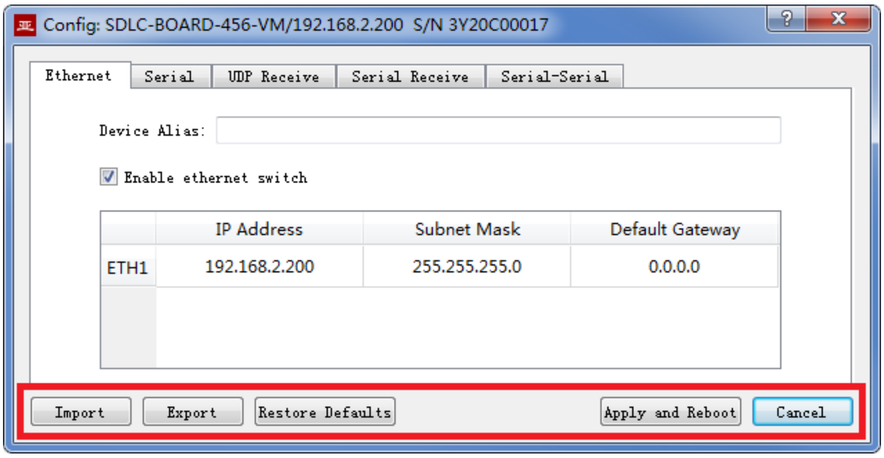
The information panel is located on the right side of the statistical report, showing the following contents:

- Device Information: Running time, serial number, IP address, version number
- Serial Port: Working clock, data and receive/transmit statistics of all serial ports (including CAN Bus)
- UDP Transmit: Displaying the relevant transmitted packets of the UDP Client for each enabled serial port to UDP entry
- UDP Receive: Displaying the relevant received packets of the UDP Server for each enabled UDP to serial port entry
- DMS Service: Displaying the information receive/transmit statistics of the configuration management between the SDLC-BOARD and the configured management computer

3.5 Configure Device

Click on the `Config` button on the toolbar or double-click on the specified device in the device list; yacer-DMS pops up the configuration dialog.

According to the interface and function, the dialog divides the configuration item into several configuration pages.

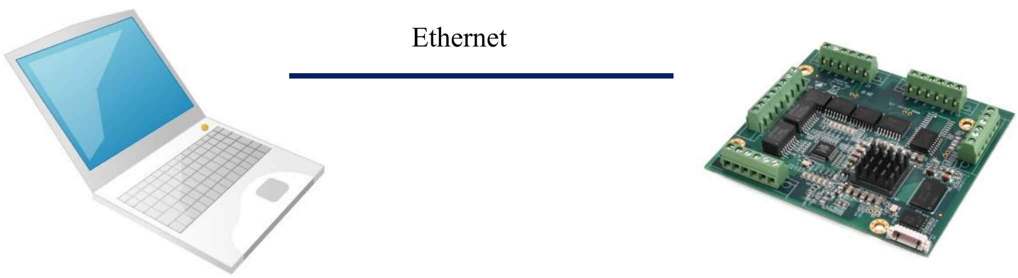


The following operation buttons are located at the bottom of the dialog:

Buttons	Functions
Import	Open the configuration file and read the refreshed configuration dialog content
Export	Save the configuration content of the current dialog to the file
Restore Defaults	Refresh the dialog content with the device's default factory configuration
Apply and Reboot	Write the configuration content of the dialog into the device and restart the device to bring the configuration into effect
Cancel	Cancel the current configuration operation

3.6 Configuration through Ethernet port

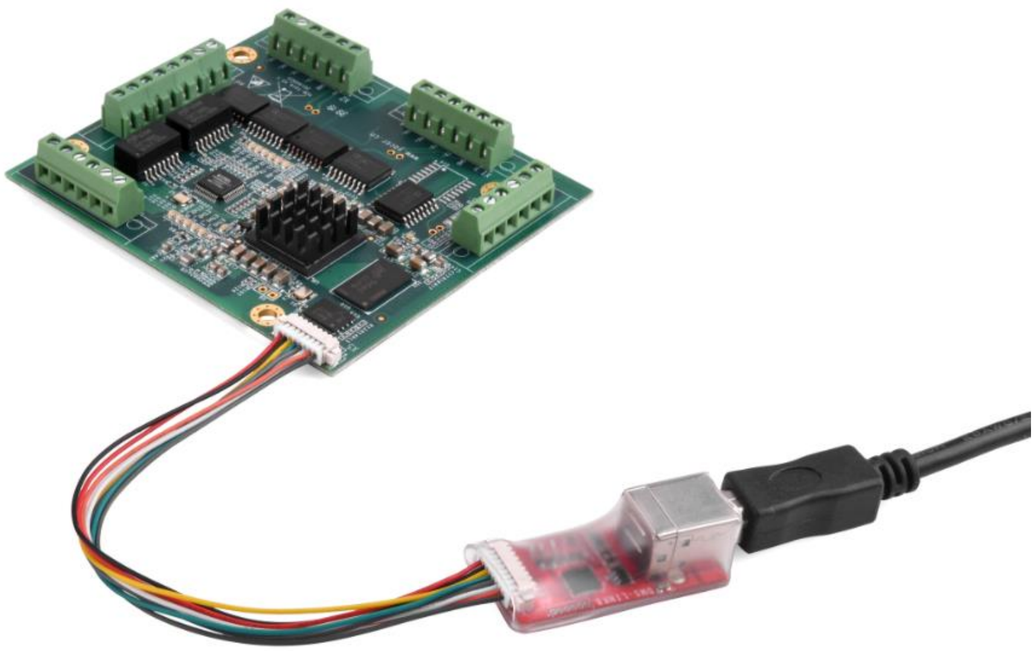
Connect the Ethernet port of SDLC-BOARD to the management computer over the network cable, running yacer-DMS configuration management software on the computer can monitor the running state and configure the parameters of SDLC-BOARD.



3.7 Configuration through the DMS-UART interface

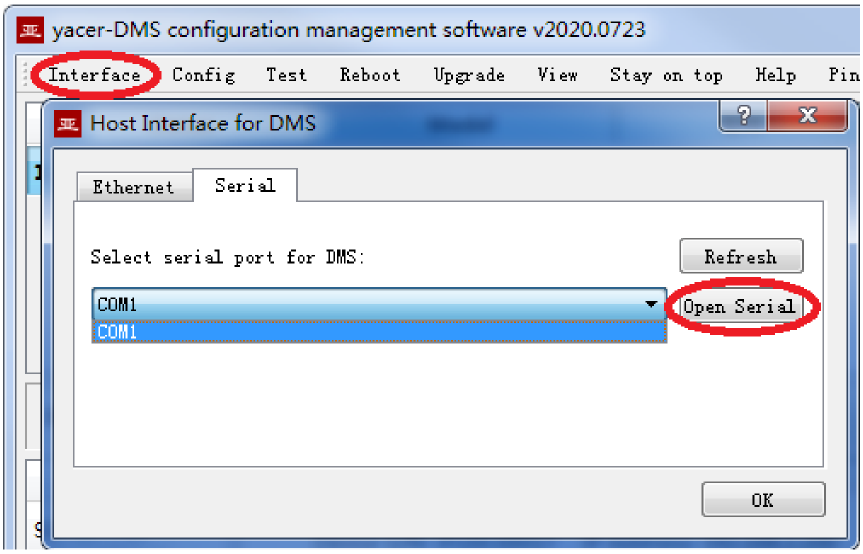
If the Ethernet interface of SDLC-BOARD is occupied, users can configure it through DMS-UART interface (X5).

DMS-UART-8P configuration cable is special configuration tool provided by Yacer, it can convert the DMS-UART interface to USB interface.

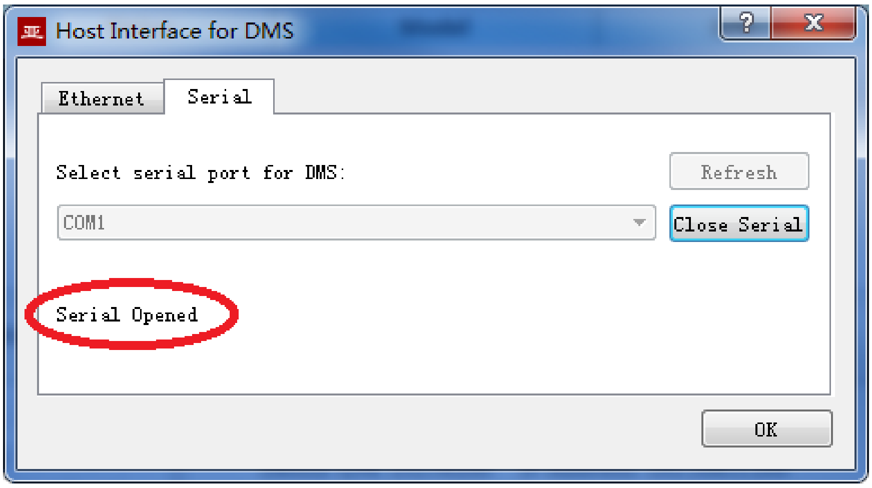


3.8 Configuration of the DMS interface

DMS-UART-8P configuration cable is connected to the USB interface of management PC, and a USB simulation serial port will be added to the computer.



Select the simulation serial port and click "open serial port". If the serial port is opened successfully, the status is as follows:



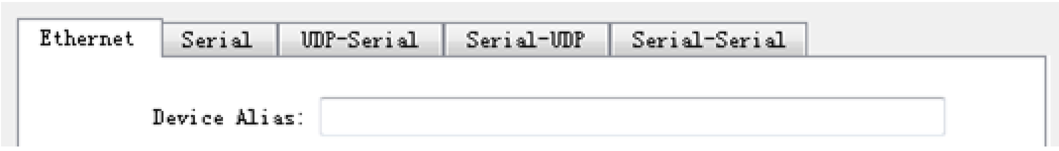
For details of DMS-UART-8P configuration cable, please refer to “User Manual of DMS-UART Interface and Configuration Cable”.

4 Function and Configuration

4.1 Ethernet Interface (ETH1, ETH2)

4.1.1 Device alias

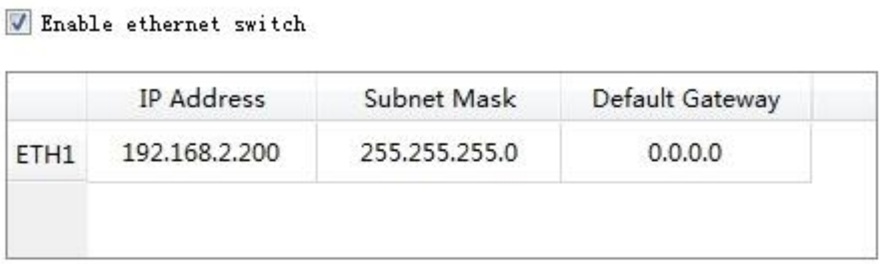
It allows users to set an alias for the SDLC-BOARD, thus adding description to the device or helping to remember the identification.



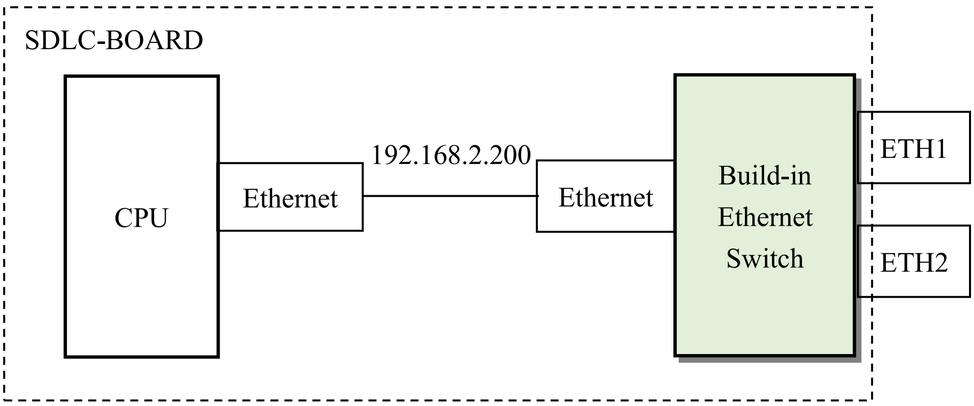
4.1.2 IP address configuration

4.1.2.1 Ethernet switching enable

By default, check the ☑ Enable ethernet switch checkbox to enable the built-in Ethernet switch, providing the Ethernet switching function between ETH1 and ETH2.



While the Ethernet switching function has been enabled, SDLC-BOARD only has one IP address. The network function figure is as follows:



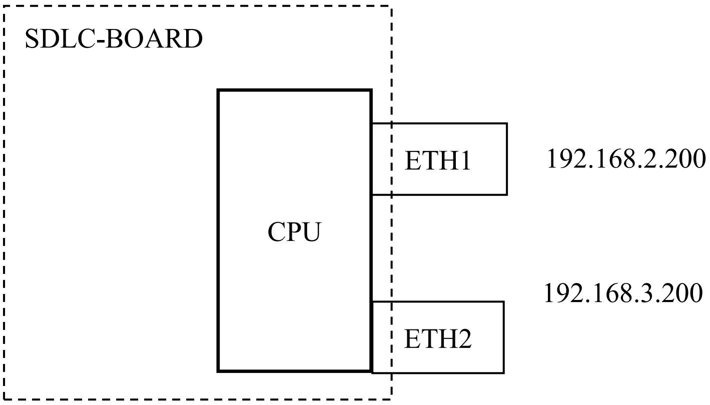
4.1.2.2 Dual IP configuration

When the ☐ Enable ethernet switch checkbox is unchecked, ensure ETH1 and ETH2 are not on the same subnet for configuration as they have an independent IP address.

☐ Enable ethernet switch

	IP Address	Subnet Mask	Default Gateway
ETH1	192.168.2.200	255.255.255.0	0.0.0.0
ETH2	0.0.0.0	0.0.0.0	0.0.0.0

With the dual-IP function figure as follows, the SDLC-BOARD is equivalent to a PC equipped with two network cards.

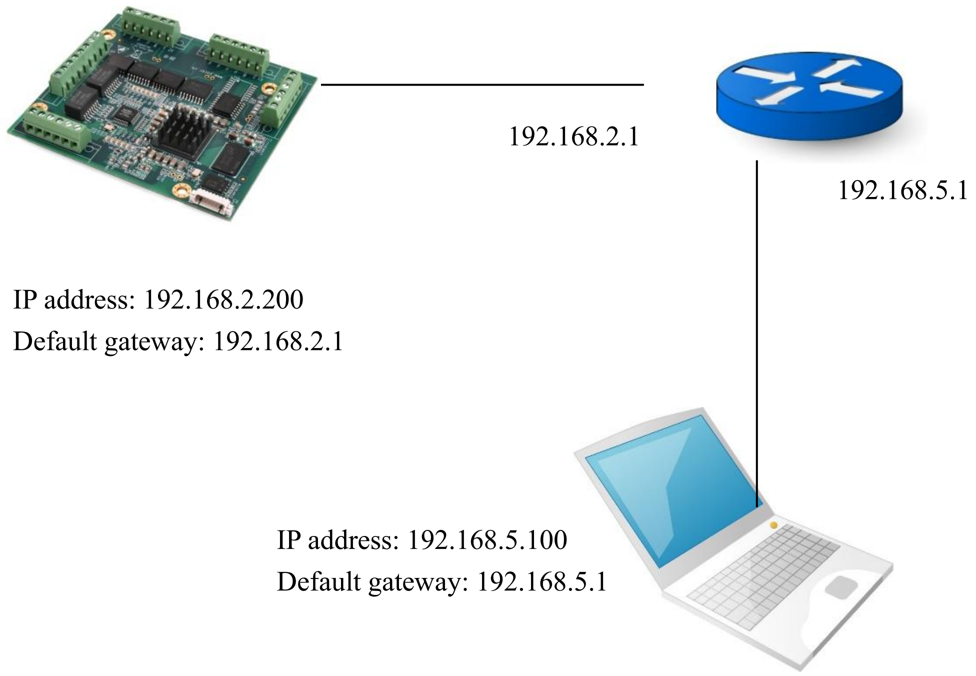


4.1.3 Default Gateway

By default, the default gateway is 0.0.0.0, representing that there is no gateway configuration.

If SDLC-BOARD needs to communicate with the host on other subnet, it must rely on an external router. At this time, the SDLC-BOARD's IP address must be on the same subnet with the IP address of the connected router port. Meanwhile, the IP address of router is set to the default gateway.

As shown below, the IP address of SDLC-BOARD and remote PC is 192.168.2.200 and 192.168.5.100 respectively. As they do not belong to the same subnet, they must rely on the router for communication. SDLC- BOARD and PC need to set the IP address of the connected router port to the default gateway of this device.



4.2 Serial Port

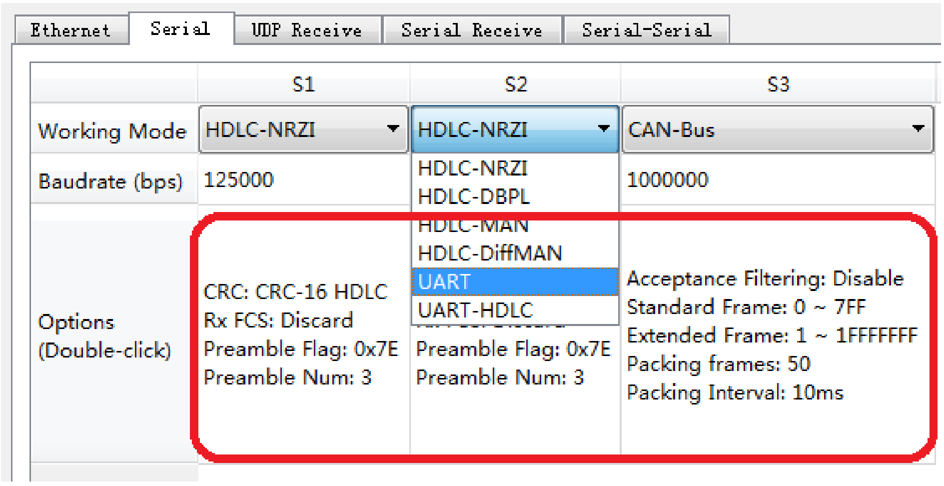
4.2.1 Working mode of No Clock mode

S1, S2 and S3 is configured as RS-422 full duplex or RS-485 half duplex at the factory, they support the synchronous and asynchronous working modes described in the following table.

Working Mode		Description
Synchronous	HDLC-NRZI	Synchronous HDLC protocol based on NRZI coding
	HDLC-DBPL	Synchronous HDLC protocol based on Differential Bi-Phase Level coding
	HDLC-MAN	Synchronous HDLC protocol based on Manchester coding
	HDLC-Diff-MAN	Synchronous HDLC protocol based on Differential Manchester coding
Asynchronous	Asynchronous UART	Common asynchronous serial, equivalent to the serial port on the common computer
	Asynchronous HDLC	UART-based similar HDLC communication protocol

Users can select the desired working mode from the "working mode" combo box. Due to different parameter configuration of each working mode, contents of the "Options" cell will be adjusted automatically according to the determined working mode.

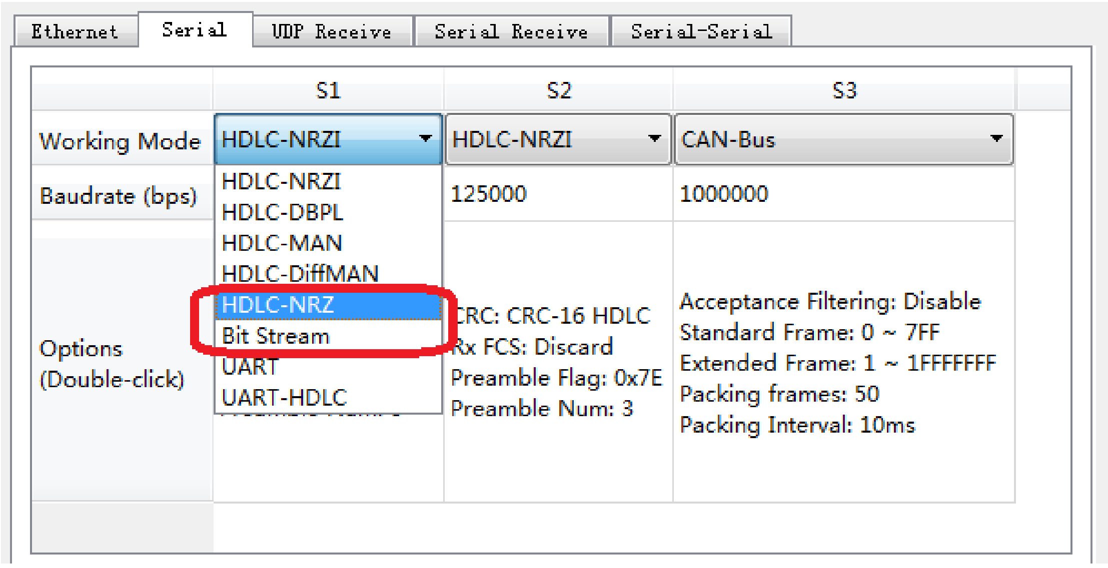
If further configuration of working parameters of the selected working mode is required, mouse double-click on the "Options" cell to pop up the parameter configuration dialog.



4.2.2 Working mode of Clock mode

X1, X2 work together to build a synchronous serial port with clock, the following working modes can be additionally supported:

Working Mode		Description
Synchronous	HDLC-NRZ	Synchronous HDLC protocol based on NRZ coding
	Bit stream	Transmit or sampling serial bit data based on receiving clock

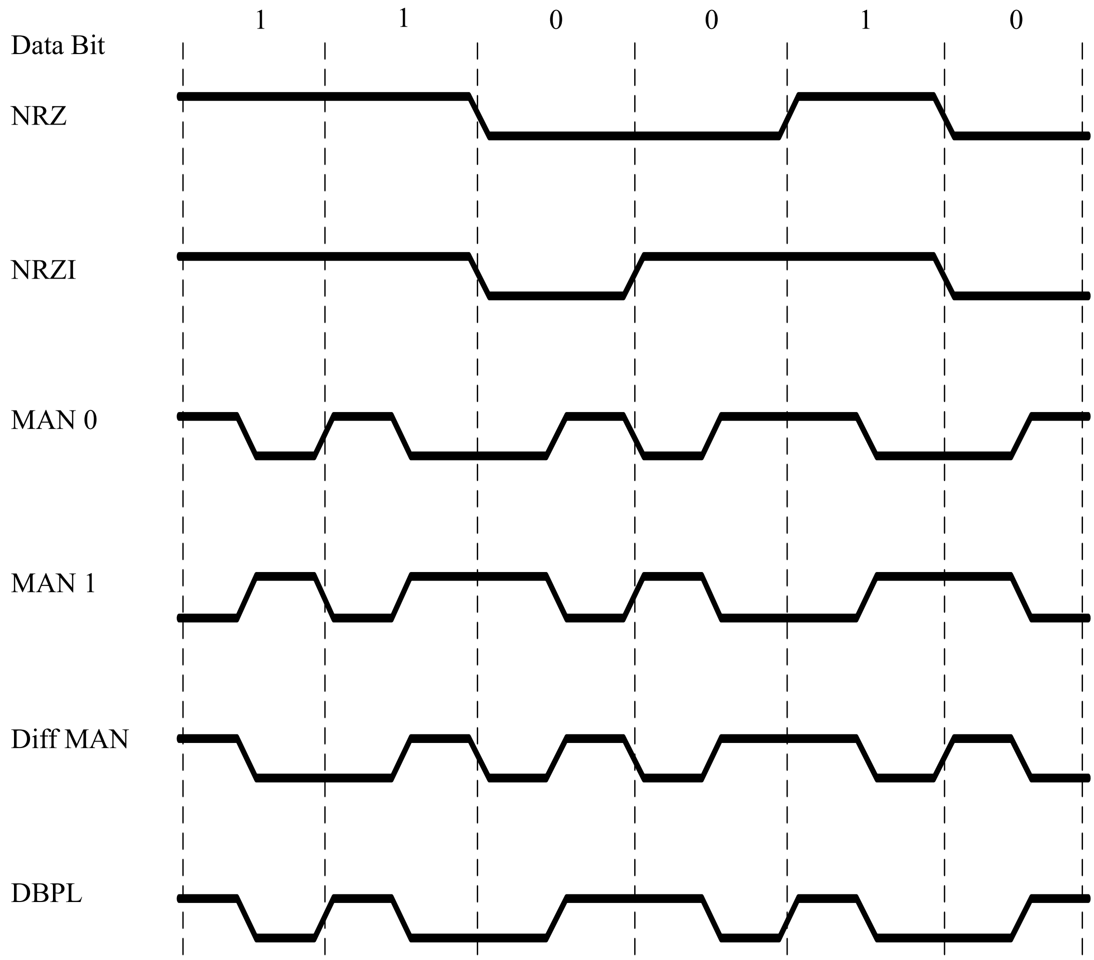


4.2.3 Baud rate

The "baud rate" configures the communication rate of the serial port. For the synchronous working modes such as HDLC-NRZI, HDLC-DBPL, HDLC-MAN, HDLC-DiffMAN and all asynchronous working modes, the baud rate of both sides of the communication must be the same in order to ensure the correct transmission of data.

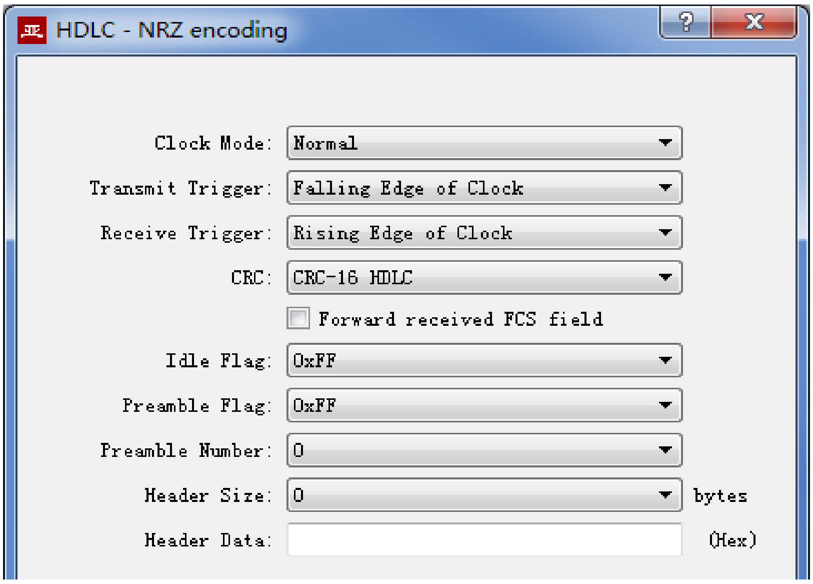
4.2.4 Encoding format of the synchronous serial port

For HDLC-NRZ, HDLC-NRZI, HDLC-DBPL, HDLC-MAN, HDLC-DiffMAN and other synchronous working modes, the link layer adopts the HDLC protocol with the encoding format difference as follows:

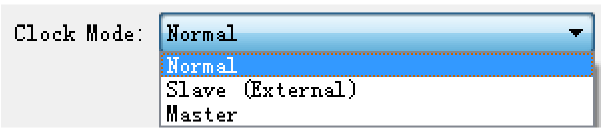


4.2.5 HDLC-NRZ parameter configuration

HDLC-NRZ is the common synchronous working mode, which is relies on receiving and receiving clock signals to achieve data bit synchronization, so the configuration of clock parameters is particularly important.



4.2.5.1 Clock mode



There are 3 clock modes for the synchronous serial port, normal, slave clock & master clock.

Clock Mode	Transmit Clock	Receive Clock
Normal	Generation from the local device, output through pin TxC.	Generation from the remote terminal device, input through pin RxC.
Slave Clock (External)	Generation from the peer device, input through pin RxC. TxC output synchronizes with RxC automatically.	Generation from the remote terminal device, Input through pin RxC.
Master Clock	Generation from the local device, output through pin TxC.	Generation from the local device, ignoring the clock of pin RxC.

The slave clock mode is also called as the external clock working mode. When the remote terminal device is the DCE (Data Communication Equipment), SDLC-BOARD is often configured as the slave clock mode and transmits data with the clock provided by the DCE, ensuring the data transmission across the whole network based a clock and avoiding packet loss concerns caused due to different clock sources.

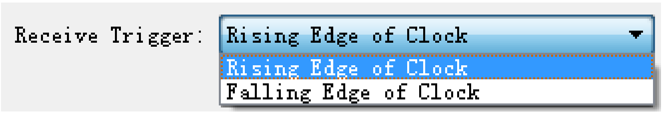
4.2.5.2 Transmit trigger

Transmit trigger defines the generation clock edge of the new data bit:

- Falling edge of clock: A new data bit is generated on the falling edge of clock
- Rising edge of clock: A new data bit is generated on the rising edge of clock

For communications that follow the HDLC protocol specification, the clock drop edge should be selected to trigger new data transmission. There are also some special applications where users use non-standard communication and use rising edge to trigger new data transmission.

4.2.5.3 Receive trigger



Receive trigger defines the sampling clock edge of the serial port receive data:

- Rising edge of clock: Data on the RxD line is read on the rising edge of the RxC signal
- Falling edge of clock: Data on the RxD line is read on the falling edge of the RxC signal

In accordance with HDLC protocol specification for communication, since the falling edge is used to trigger new data, considering the stable time of new data, in order to ensure the correct reading of data, the receiving trigger must be configured as the clock rising edge.

The local receive trigger configuration is determined according to the transmit trigger of the remote terminal device:

Remote Transmit Trigger	Local Receive Trigger
Falling edge of clock	Rising edge of clock
Rising edge of clock	Falling edge of clock

4.2.5.4 CRC

In order to verify the correctness of data communication, CRC function should be enabled.

By default, configure the protocol CRC check type with CRC-16-HDLC as the most commonly used type for the HDLC protocol communication.

CRC	Description
Disable	CRC disable: ● No CRC calculation for data transmission or FCS field for HDLC frame ● No CRC check for data receiving
CRC-16 HDLC	Adopt the 16-bit IBM HDLC CRC check method
CRC-16 SDLC	Adopt the 16-bit IBM SDLC CRC check method

4.2.5.5 Forward received FCS field

This configuration is only effective with CRC enable.

The HDLC frame structure is shown in the following table, where FCS is the frame check sequence field.

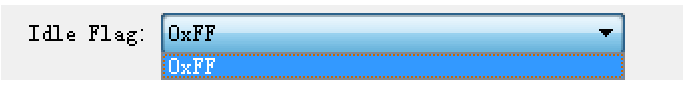
Opening Flag	Address Field	Control Field	Information Field	FCS Field	Closing Flag
0x7E	1 byte	1 byte	Variable length	CRC 2/4 bytes	0x7E
0x7E	User data			CRC 2/4 bytes	0x7E

If this option is checked, then forward the user data and FCS field.

If this option is not checked, SDLC-BOARD will discard the 2/4-byte FCS field at the end of data and only forward the user data after the receive HDLC frame check is passed.

4.2.5.6 Idle Flag

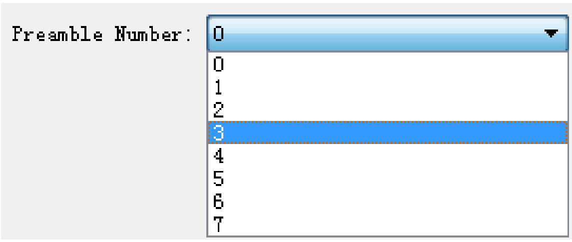
The definition of HDLC inter frame filling content, the default should be 0xFF.



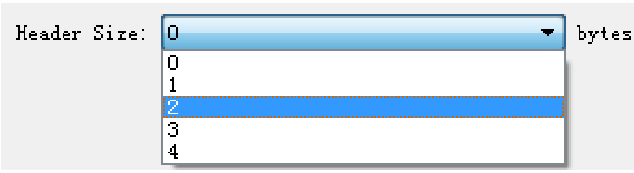
4.2.5.7 Preamble flag and number

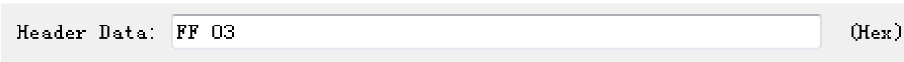
During the half-duplex communication, a preamble flag is often required in front of the frame for receiving party synchronization, and the most commonly used method is to add 2~5 0x7E.

For full duplex applications, the Preamble number is often unrequired, set it to 0(no preamble).



4.2.5.8 Header size and data





As shown above, the header size is defined as 2, and the header data is defined as FF 03.

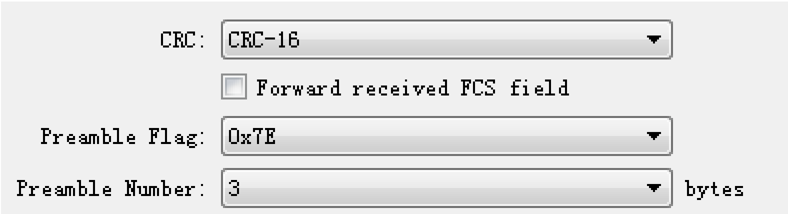
- While transmitting HDLC, the FF 03 is added before the user data, and HDLC frame data is composed with user data.
- When receiving HDLC, SDLC-BOARD discards the first 2 bytes of HDLC frame data as the frame header, and only forwards the subsequent data to the user.

Opening Flag	Frame Header	User Data	FCS Field	Closing Flag
0x7E	0xFF 0x03	Variable length	CRC 2/4byte	0x7E

4.2.6 HDLC-NRZI parameter configuration

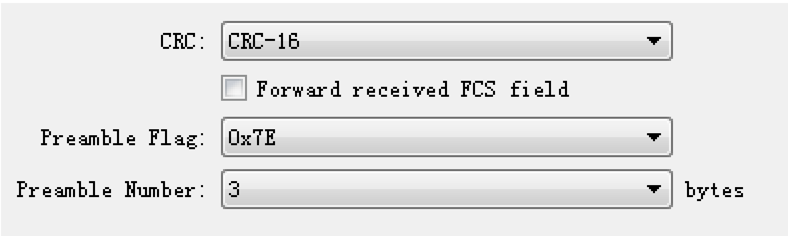
Unlike the NRZ encoding format, the NRZI encoding format data contains clock information, which only requires that the baud rate of the both communication sides should be the same, instead of the clock mode, transmit trigger, receive trigger and other parameters.

The option dialog of the HDLC-NRZI working modes is shown as follows:



4.2.7 HDLC-DBPL parameter configuration

HDLC-DBPL adopts Differential Bi-Phase Level encoding format, its advanced options are as follows:

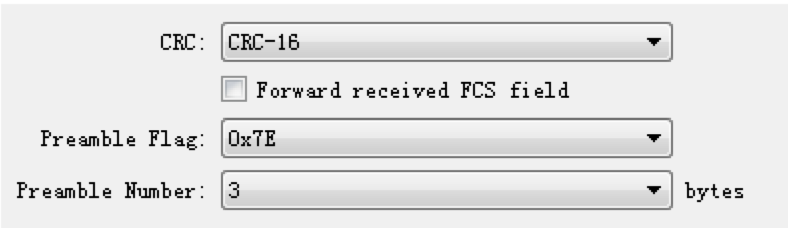


Parameters of HDLC-DBPL are the same of HDLC-NRZ.

It should be noted that many claims that the encoding implementation of DBPL is the same of the differential Manchester, so users need to carefully refer to the definition of encoding format in Chapter 4.2.4 and choose the right working mode.

4.2.8 HDLC-DiffMAN (Differential Manchester) parameter configuration

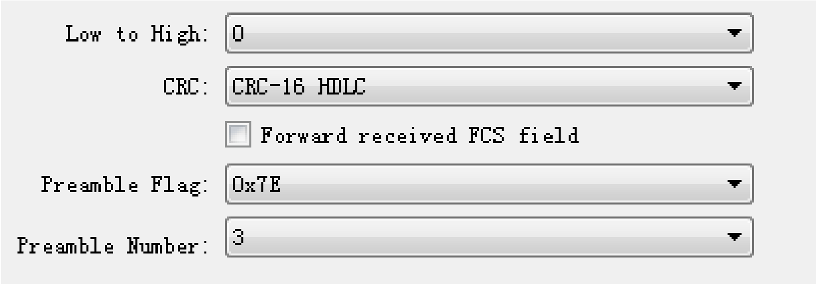
The advanced options of Differential Manchester encoding format are as follows:



Parameters of HDLC-DiffMAN are the same of HDLC-NRZ.

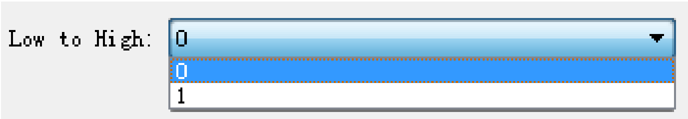
4.2.9 HDLC-MAN (Manchester) parameter configuration

The advanced options of the Manchester encoding format are as follows:



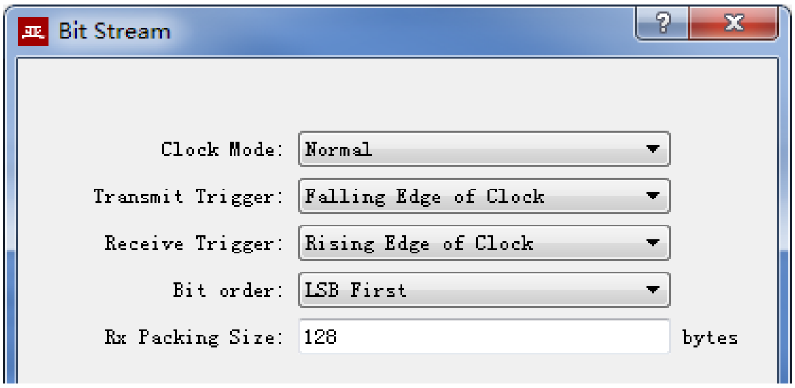
Except for the configuration parameters same as NRZ, parameters with the data line low-to-high transition definition are included for the Manchester encoding format:

- 0: Low-to-high transition indicates the logic 0;
- 1: Low-to-high transition indicates the logic 1;



4.2.10 Bit Stream parameter configuration

The rising or falling edge of each clock cycle samples the 1bit data on the data line, which forms a UDP message and transmits to the destination IP after receiving a byte with the packet length by forming a byte with each 8bit.



Refer to the HDLC-NRZ parameter configuration for configuration of clock mode, transmit trigger and receive trigger.

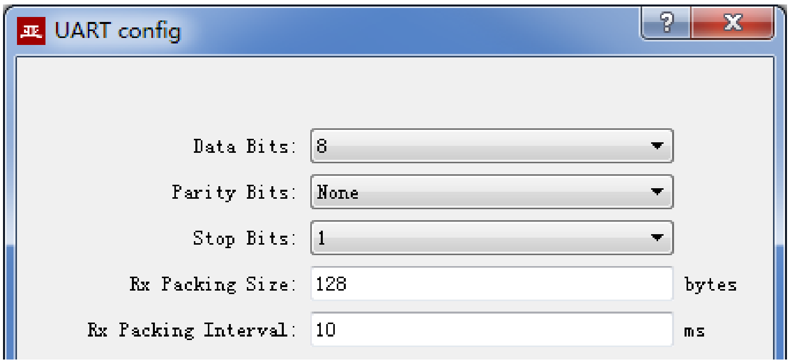
The online bit stream is stored in the computer or system memory in the form of byte. The receive/transmit sequence determines the conversion mode of byte and bit.

Receive/Transmit Sequence	Transmit Operation	Receive Operation
MSB first	First transmit the high-bit byte	Data received first is placed on the byte high bit
LSB first	First transmit the low-bit byte	Data received first is placed on the byte low bit

4.2.11 UART parameter configuration

UART is a means of character stream communication; data bits, parity bits and stop bits define the basic working parameters of the asynchronous serial port, which must be configured identically to the remote terminal device.

Generally, data bits are defined as 8, i.e. one byte, and UART corresponds to the byte stream communication.



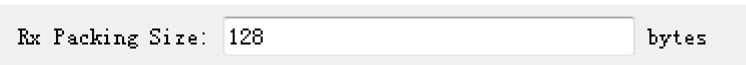
When the UART byte stream is converted into a UDP message or HDLC frame, it is too costly and inefficient if each byte is converted to a UDP message for transmission.

To improve efficiency, SDLC-BOARD buffers the received byte stream and forms a number of buffered bytes into a UDP message to transmit, of which this process is called as packet.

Packets are controlled with two parameters, namely the packet length and the packet interval.

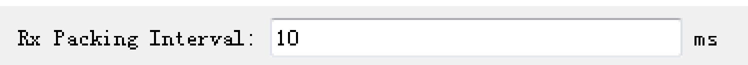
4.2.11.1 Rx packeting length

For example, if the packet length is set to 128 bytes, then it will form a UDP message to transmit after UART receives the full 128 bytes.



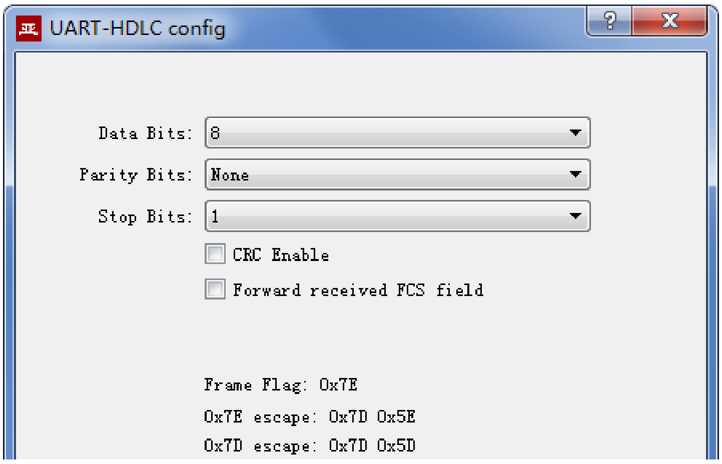
4.2.11.2 Rx Packet interval

As shown in the example above, the packet interval 10ms is set, and if no new byte data is received over 10ms, it will form the buffer data as a packet to forward no matter whether it has received the full 128 bytes.



4.2.12 UART-HDLC parameter configuration

The UART-HDLC working mode is a custom protocol by Yacer which form the asynchronous HDLC frame on the basis of the normal UART communication by packaging the byte stream. Therefore, the asynchronous serial port can perform the packet-based communication with the UDP message and synchronous HDLC frame.



The UART-HDLC frame format adds 0x7E before and after the packet as the opening flag and closing flag with the frame structure is as follows:

opening Flag	Information Field	FCS Field	closing Flag
0x7E	2~1470 bytes of data	2-byte CRC data	0x7E

As the information field and FCS field may appear 0x7E, perform the character escape on such fields before transmission with the escape rules are as follows:

- 0x7E: Escaped to two characters, 0x7D 0x5E
- 0x7D: Escaped to two characters, 0x7D 0x5D
- Other characters: No escape

The escape operation of data transmit is as follows:

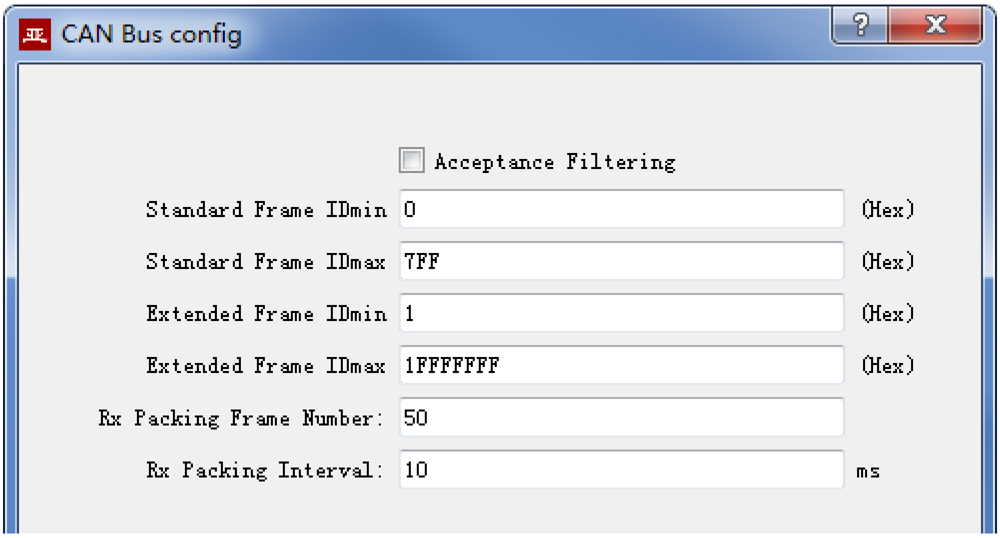
Original Data	Actual Transmit Data
0x7E	0x7D 0x5E
0x7D	0x7D 0x5D
Others	No change

The escape operation of data receive is as follows:

Actual Receive Data	Data
0x7D 0x5E	0x7E
0x7D 0x5D	0x7D
Others	No change

4.2.13 CAN Bus configuration

Users can select the S3 as a CAN Bus interface at the factory, configuration is as follows:



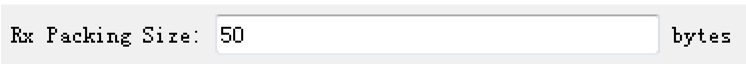
As CAN frame is short, when the CAN frame is converted into a UDP message or HDLC frame, it is too costly and inefficient if each byte is converted to a UDP message for transmission.

To improve efficiency, SDLC-BOARD buffers the received CAN frames and forms a number of buffered CAN frames into a data packet to transmit, of which this process is called as packet.

Packets are controlled with two parameters, namely the packet length and the packet interval.

4.2.13.1 Rx Packing Size

The maximum Packet length is 50.

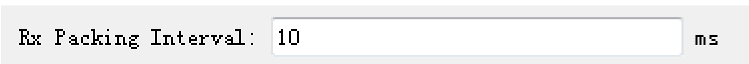


For example, if the packet length is set to 50 frames, then it will form a data packet to transmit after CAN Bus receives the full 50 frames.

As shown above the Rx Packing Size has been set to 50, the SDLC-BOARD pack the data into a packet to transmit out when it received 50 frames data from CAN Bus.

4.2.13.2 Rx Packet Interval

As shown in the example above, the packet interval 10ms is set, and if no new CAN frame is received over 10ms, it will form the buffer data as a packet to forward no matter whether it has received the full 50 frames.



4.2.13.3 Acceptance filtering

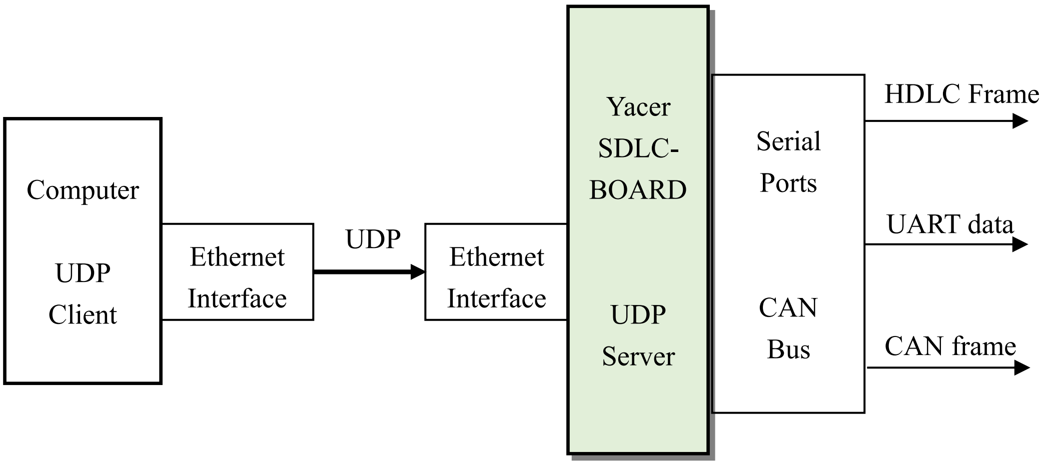
Acceptance filter allows user configure the acceptance of frame ID range, and frames outside the range are automatically discarded.

4.3 UDP to Serial Conversion

4.3.1 Function description

With SDLC-BOARD, PC or server can realize the transmit function of the synchronous HDLC, asynchronous UART or CAN Bus.

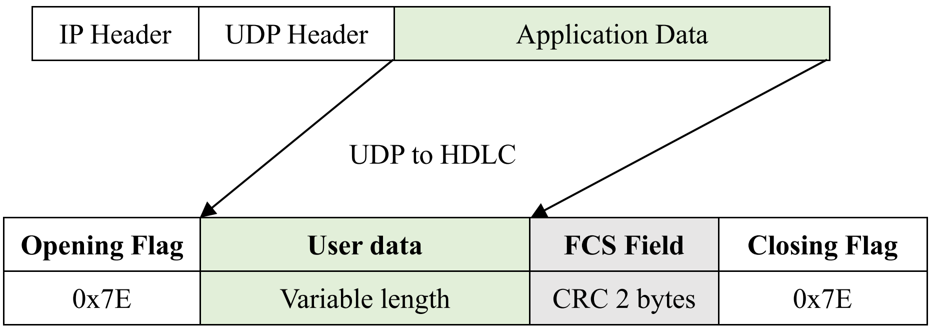
The typical application is shown as follows. PC transmits a UDP message over the Ethernet interface as the UDP Client, and SDLC-BOARD transmits it out from the CAN Bus after converting the received UDP message into the HDLC frame, UART data stream or CAN frame.



4.3.2 Protocol conversion

The most typical UDP-to-HDLC application is shown below. SDLC-BOARD loads the UDP application data into the user data area of the HDLC frame, and then calculates CRC and populates the FCS field to form a complete HDLC frame to transmit.

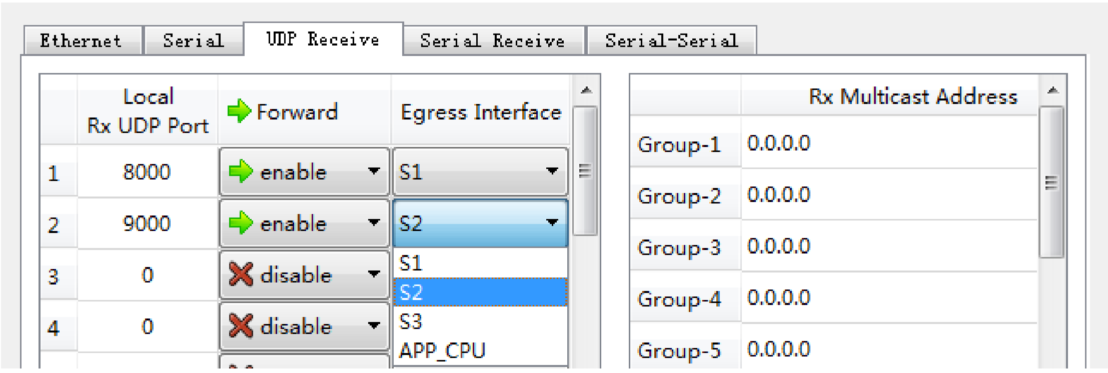
In order to reduce the calculation load of PC and the user programming complexity, normally, the FCS field of HDLC is not included in the UDP message, which is populated through SDLC-BOARD calculation.



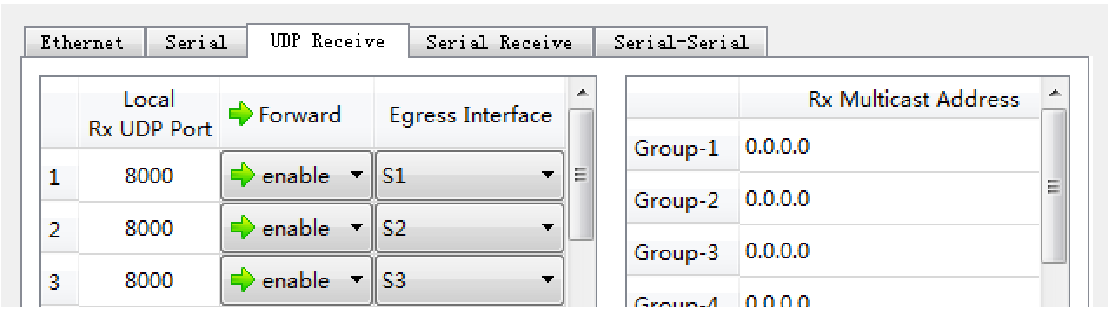
4.3.3 Forward Configuration

Set the UDP to serial port. Each row represents the forwarding entry from a UDP port to the serial port while "enable" is selected with three forwarding strategies to be achieved:

- Forwarding: Data received by the specified UDP port can be forwarded to the specified serial port.
- Multiplexer: Data received by several different UDP ports can be forwarded to the same serial port.
- Demultiplexer: Data received from the same UDP port can be forwarded to the different serial ports.



The following configuration realizes the application, where data received from 1 UDP and distributed to 3 serial ports:



4.3.4 Receive multicast

If users need to receive the multicast UDP message, add the required multicast address from the right "Rx Multicast Address" list.

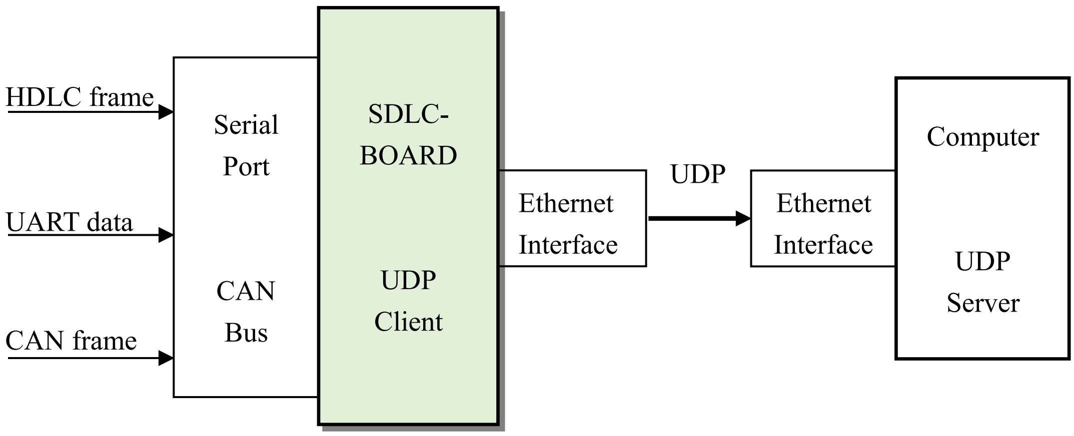
Range of the multicast address is 224.0.0.0 ~ 239.255.255.255, 224.8.8.8 is the configuration management address of the SDLC-BOARD and users can't use this address.

The multicast address configured as 0.0.0.0 indicates that the entry is not in effect.

4.4 Serial to UDP Conversion

4.4.1 Function description

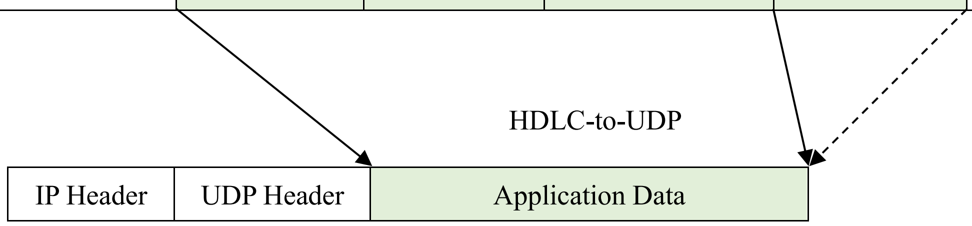
The SDLC-BOARD receives HDLC frames through synchronous serial port, receives UART strings through asynchronous serial port and unpack them, receives CAN frames through CAN port and groups the packets, converts the above data frames or packets into UDP packets, and then transmits the packets to the computer or server through Ethernet port according to the configuration.



4.4.2 Protocol Conversion

To ensure the completeness of user data, SDLC-BOARD places the complete HDLC frame in the UDP application data, and forwards to the UDP Server.

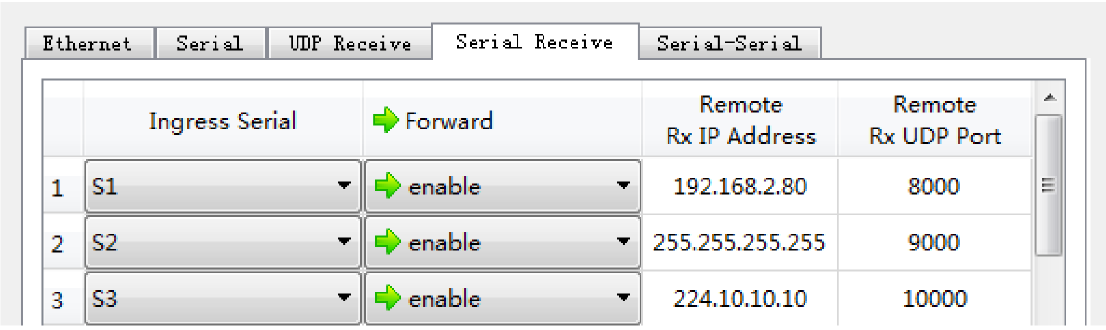
Opening Flag	Address Field	Control Field	Information Field	FCS Field	Closing Flag
0x7E	1 byte	1 byte	Variable length, N-byte	CRC 2 bytes	0x7E



4.4.3 Forward Configuration

Set the serial port to UDP. Each row represents the forwarding entry from a serial port to the destination UDP port while "enable" is selected with three forwarding strategies to be achieved:

- Data received from the specified serial port can be forwarded to the specified UDP port and destination IP.
- Multiplexer: Data received from several different serial ports can be forwarded to the same UDP port and the same destination IP.
- Demultiplexer: Data received from the same serial port can be forwarded to the different UDP ports or the different destination IP.



As shown above, three serial port to UDP entries are configured for achieving the following items:

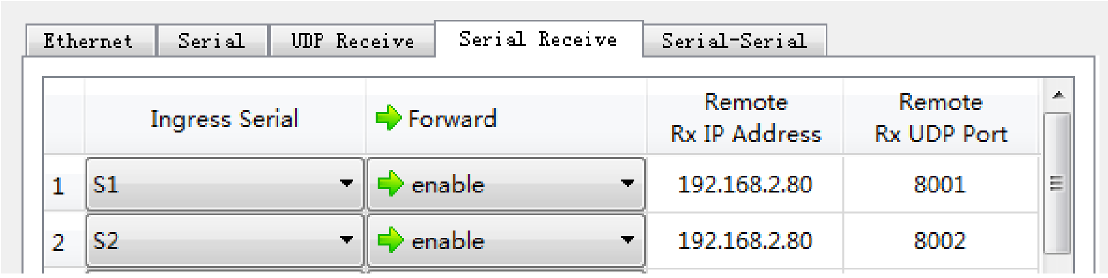
- Serial port S1 to UDP unicast, with the destination IP address as 192.168.2.80 and destination UDP port as 8000
- Serial port S2 to UDP broadcast, all hosts on the subnet can receive data from S2 at the 9000 port
- Serial port S3 to UDP multicast, only the PC joining Group 224.10.10.10 on the network can receive data from S3.

4.4.4 How does the UDP server identify the source serial

In the multiplexer application mode, HDLC frames originating from several different serial ports need to be forwarded to a server or computer for unified processing. In this case, a strategy is required to enable the computer to know which serial port the UDP packets are received from.

4.4.4.1 Distinguish the source serial port according to the source UDP port

As shown below, set different forwarding destination UDP ports for each serial port. The UDP Server PC receives data at the different UDP ports. Message received at port 8001 is from the serial port S1 while message received at port 8002 is from the serial port S2.



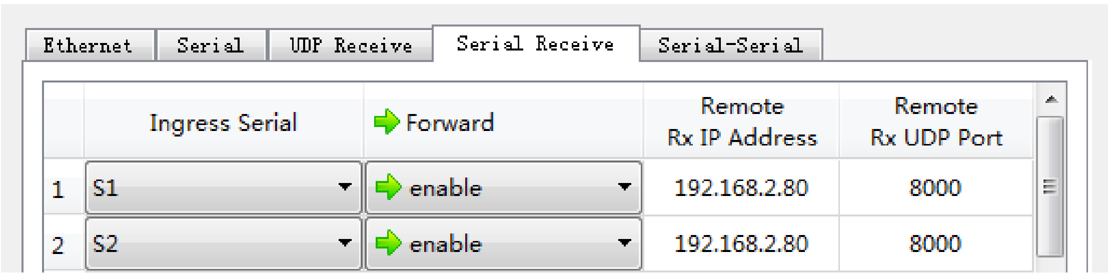
	Ingress Serial	Forward	Remote Rx IP Address	Remote Rx UDP Port
1	S1	enable	192.168.2.80	8001
2	S2	enable	192.168.2.80	8002

4.4.4.2 Distinguish the source serial port according to the source UDP port

When the source serial port is identified with the destination UDP port, UDP Server needs to listen and receive data on a plurality of UDP ports. In case there are many serial ports, not only the UDP Server port occupies too many resources, the configuration and programming complexity also increases significantly.

In order to simplify implementation of the UDP Server side, we can use the configuration example below, forwarding each conversion to the same port of the UDP Server. *During yacer's SDLC-BOARD forwarding, it will automatically adjust the source port number of the UDP message according to the source serial port. The source ports of the UDP message forwarded by the serial ports S1 and S2 are 8001 and 8002 respectively; the following is gradually increasing.*

Thus, UDP Server only needs to listen and receive data at a port (8000 in the example below) and distinguishes the source serial port according to the source UDP port. If several SDLC-BOARDs are provided, UDP Server can distinguish the source device via the source IP.



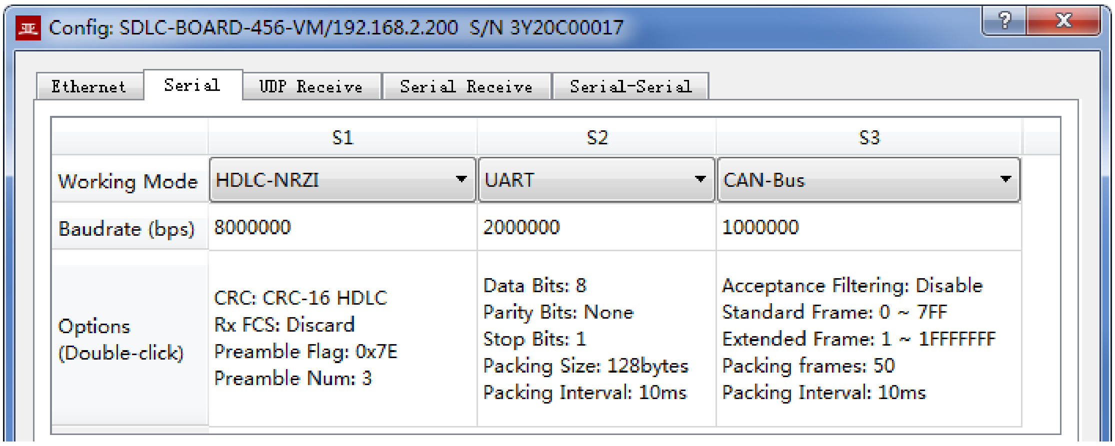
	Ingress Serial	Forward	Remote Rx IP Address	Remote Rx UDP Port
1	S1	enable	192.168.2.80	8000
2	S2	enable	192.168.2.80	8000

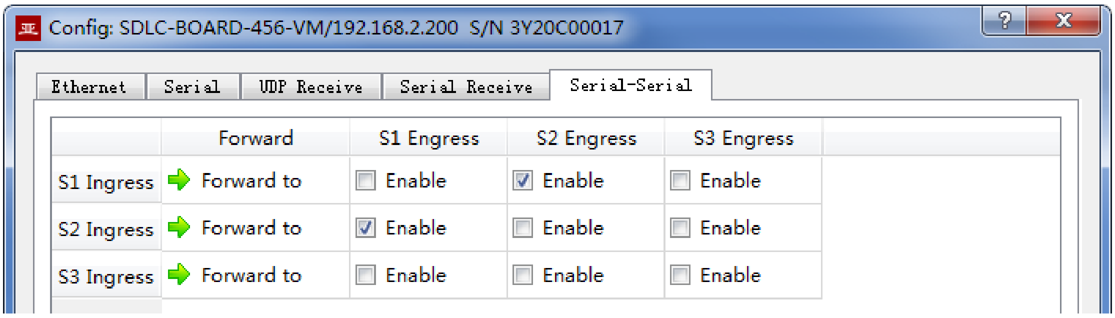
4.5 Serial Port to Serial Port

Serial-serial can forward the input data of the specified serial port to other serial port outputs, which is mainly used for:

- Conversion between synchronous and asynchronous serial ports
- Conversion between serial ports and CAN Bus.

As configured below, serial Port S1 works in synchronous HDLC mode, while S2 works in asynchronous UART mode. The serial port to serial port is configured as the mutual forwarding between S1 and S2, so the data conversion between the same and asynchronous serial ports can be realized.



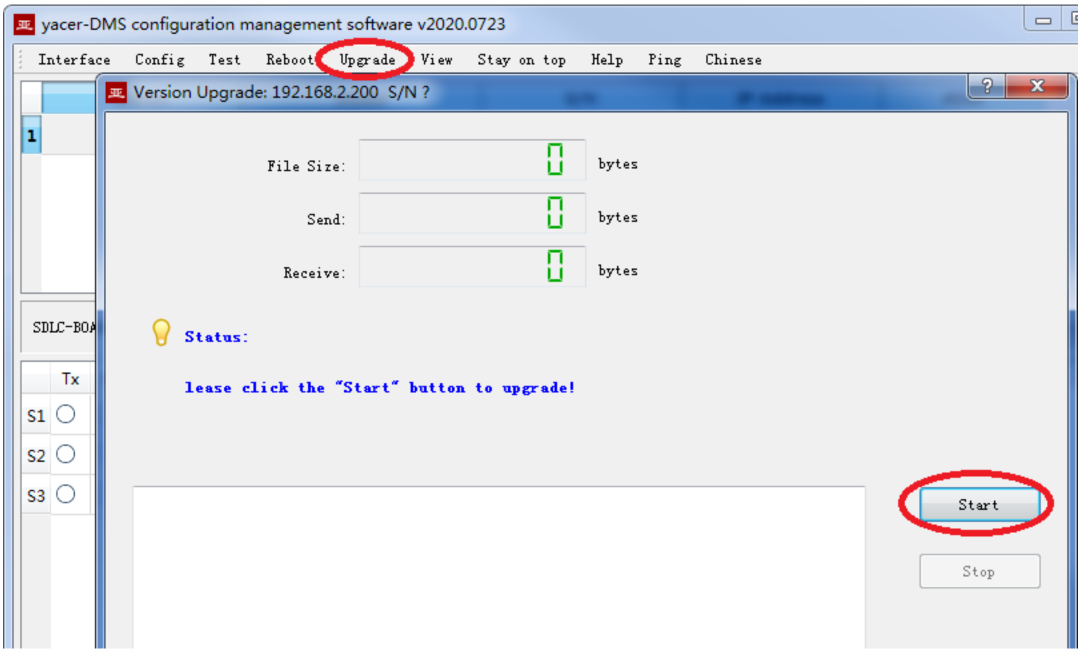


5 System Maintenance

5.1 Firmware Version Upgrade

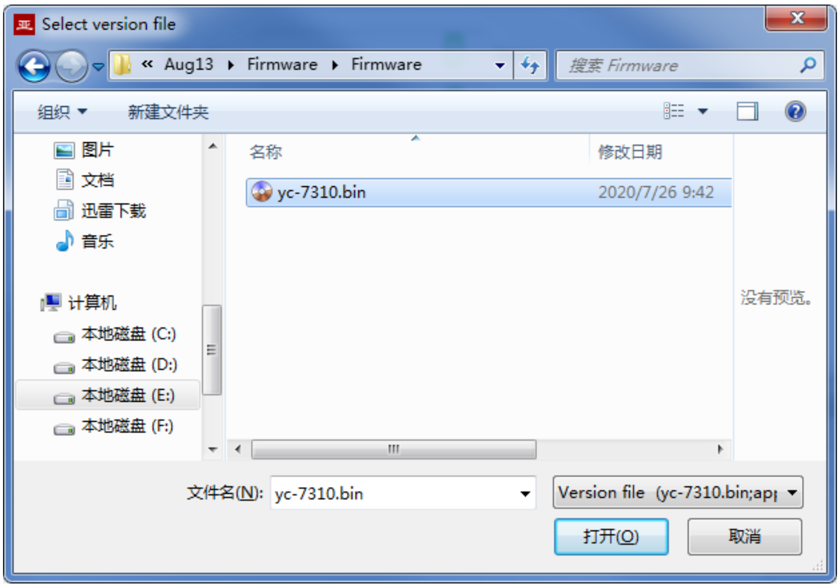
5.1.1 Start updating

Click on the Upgrade button on the toolbar to pop up the version upgrade dialog, and then click on the Start button.



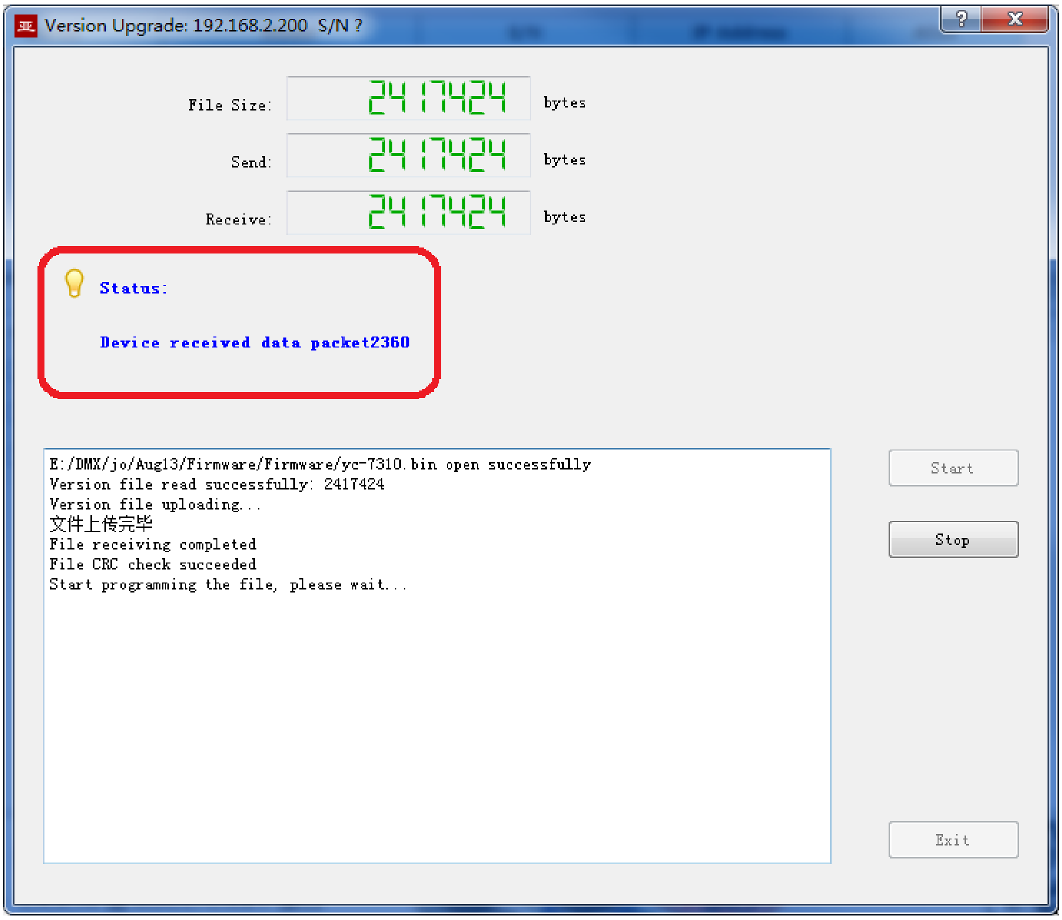
5.1.2 Locate Firmware Version

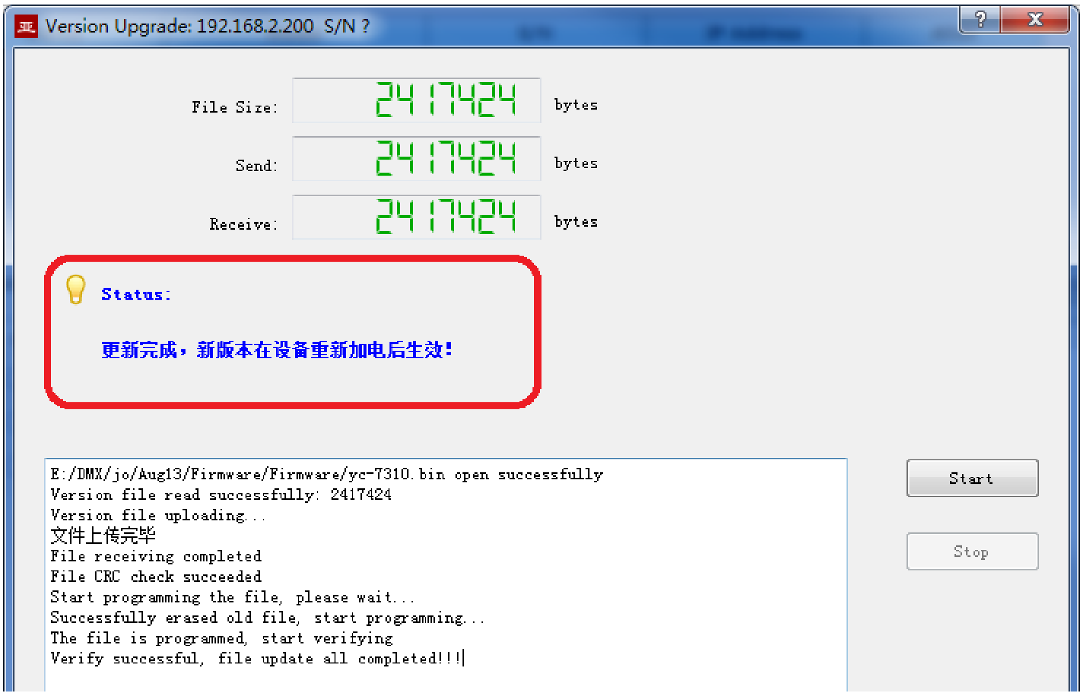
The “Selection Version File” dialog pops up. Locate the folder for storing the latest firmware version, select and click “Open” to start updating.



5.1.3 Upgrade Completed

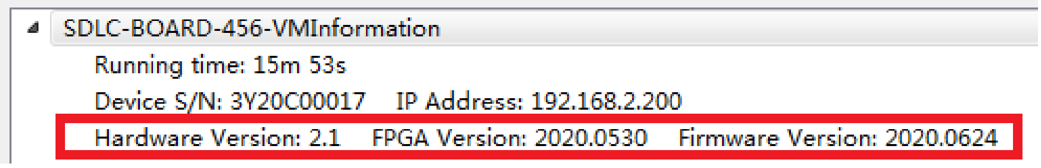
After completion of upgrade, “Version Update Completed” displaying on the page indicates that the version update is completed.





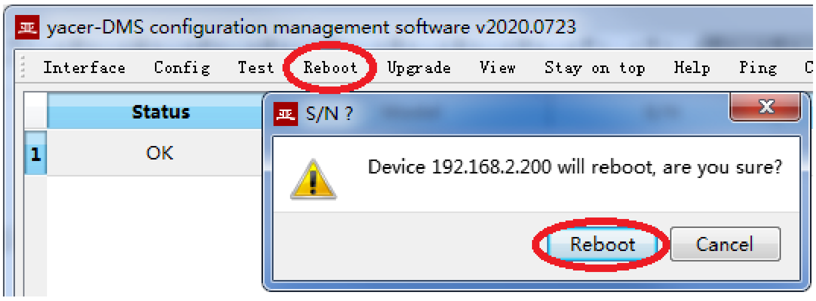
5.1.4 Upgrade Confirmation

After completion of update, re power-on the device, observe the version information in the statistical report and determine whether the new version is updated successfully via the version date.



5.2 Device Reboot

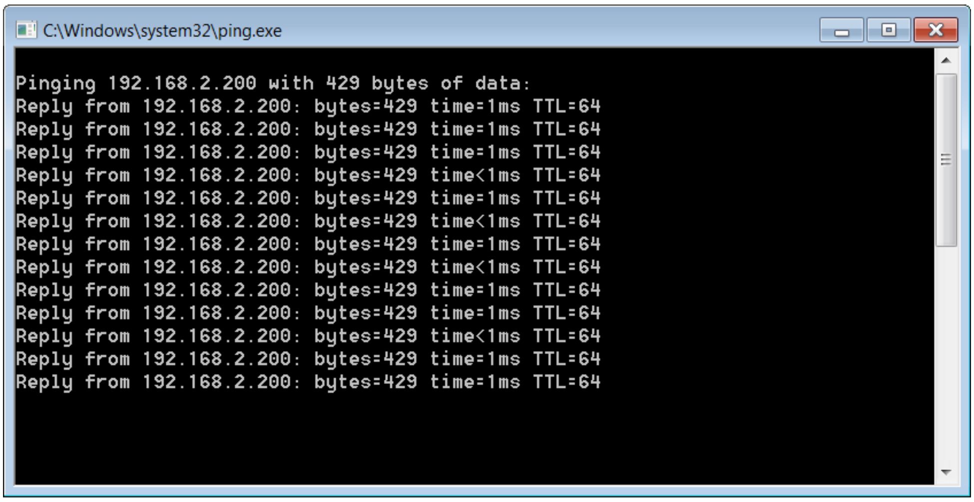
Click on the Reboot button on the toolbar to pop up the device reboot dialog, and then click on the Reboot button to reboot the device.



5.3 Ping

By clicking on the Ping button on the toolbar, DMS will start the ping command automatically for the selected device so as to check whether the network connection between the configuration management computer and SDLC-BOARD is normal.

Before performing the Ping command, first ensure that the IP address of PC and SDLC-BOARD is on the same subnet.



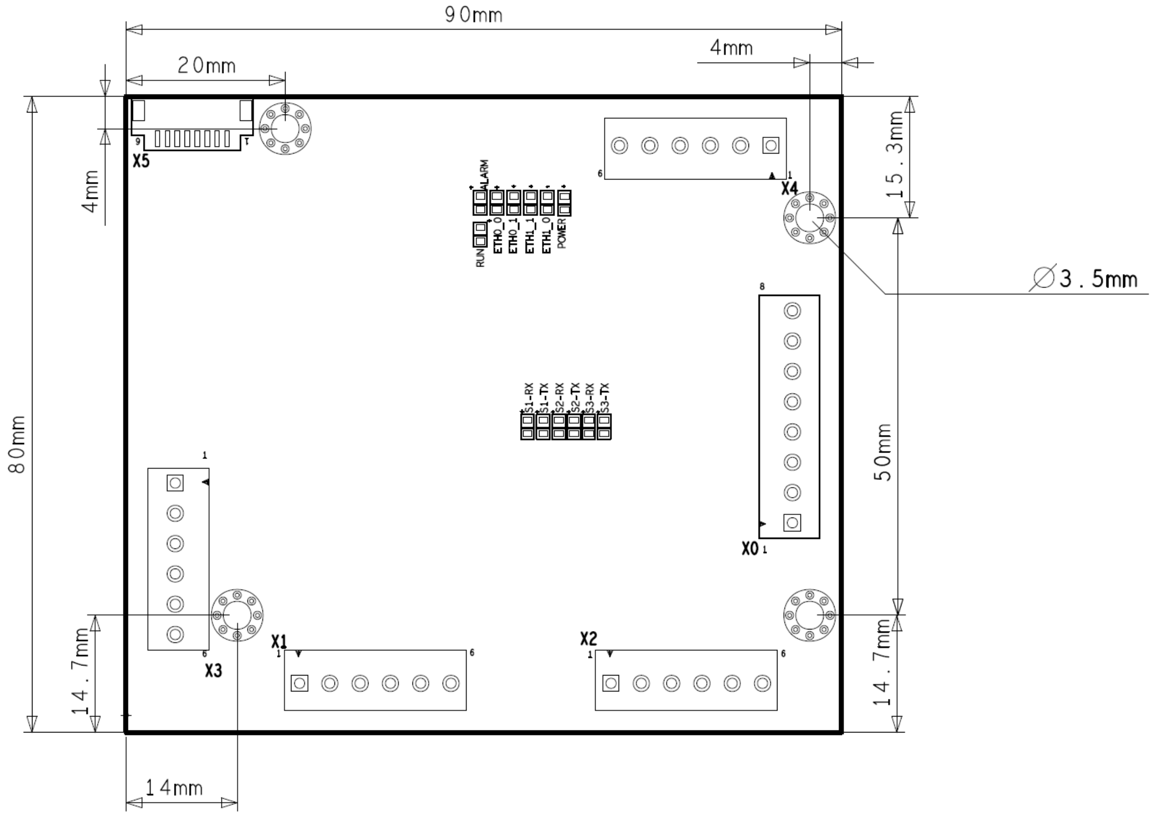
6 Mechanical characteristics and installation

6.1 Installation

Fixed with 4 m3 screws, mounting hole diameter = 3.5mm.

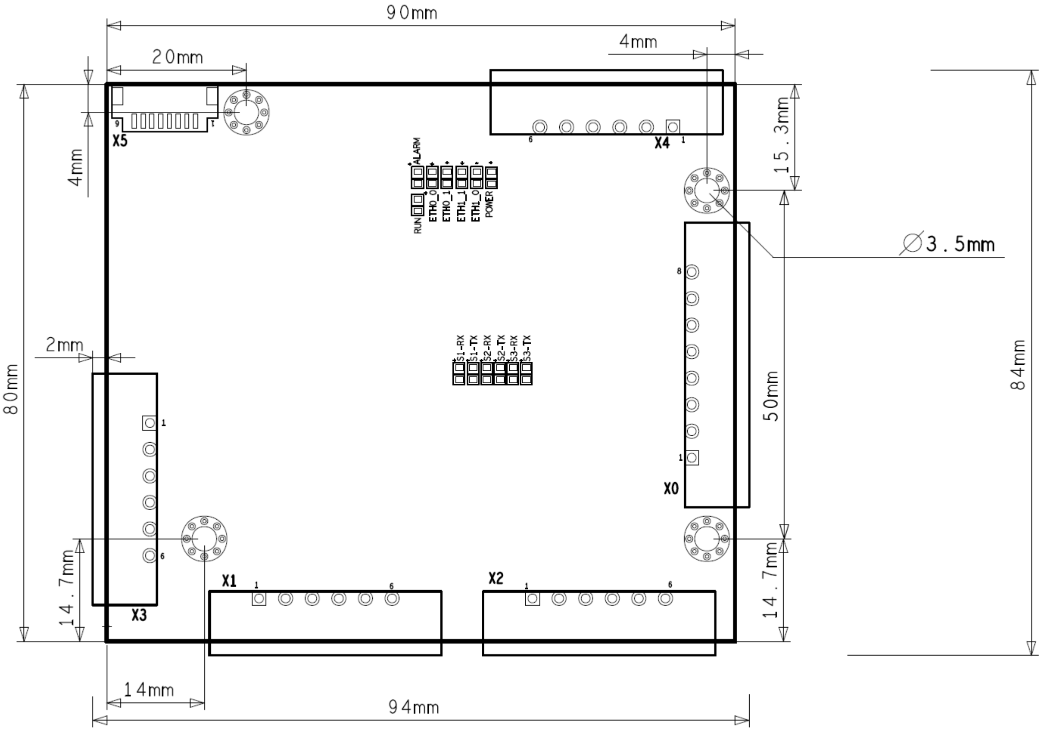
6.2 LG Dimension

Height occupied 130mm.



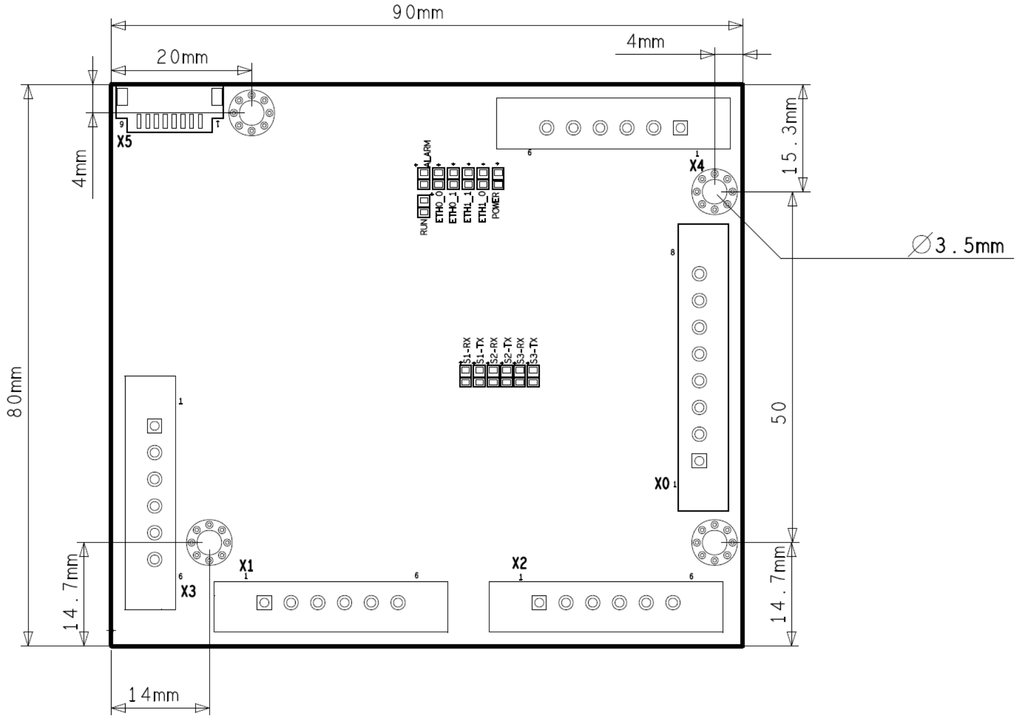
6.3 RM Dimension

Height occupied 130mm with plug.



6.4 VM Dimension

Height occupied 230mm with plug.



7 Protocol Conversion Application

7.1 Serial port data conversion

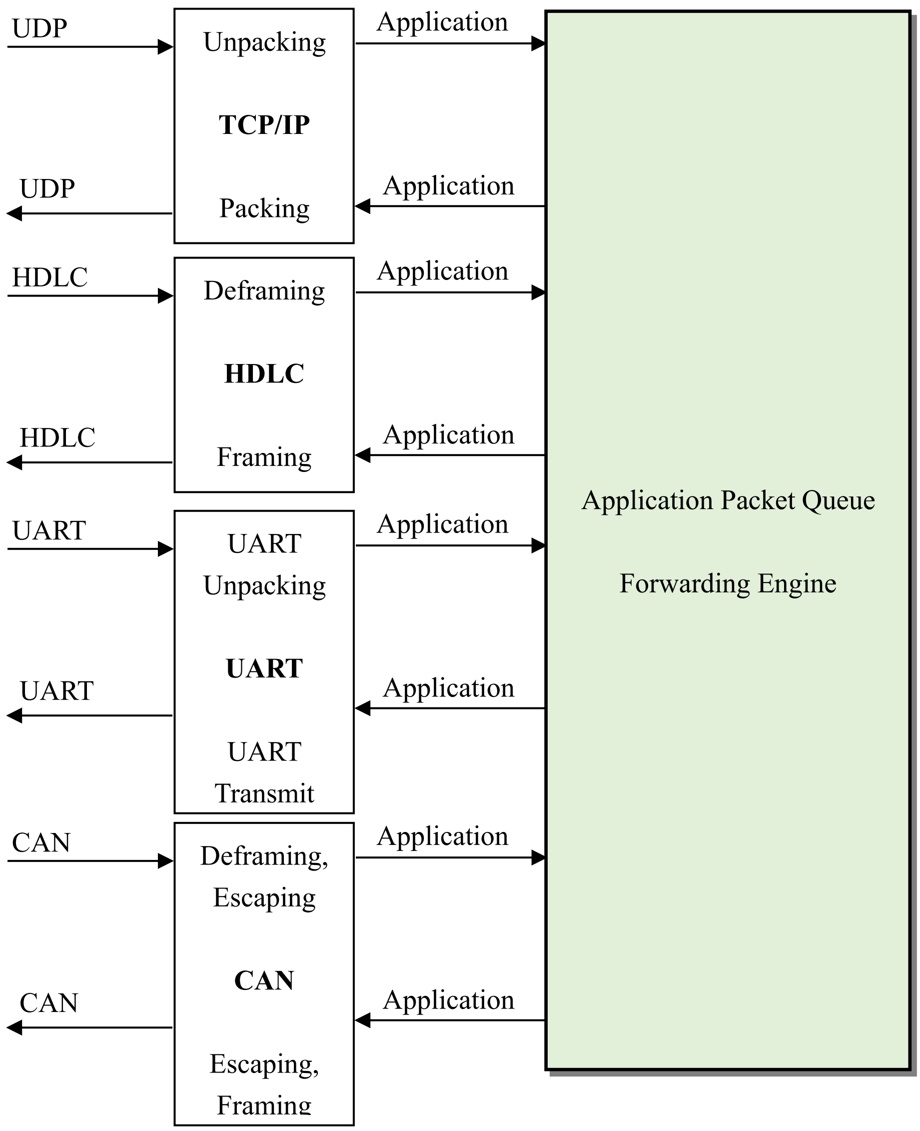
7.1.1 Application packet and conversion model

Serial port data conversion includes:

- Protocol conversion between the serial port and UDP
- Data conversion between synchronous and asynchronous serial ports

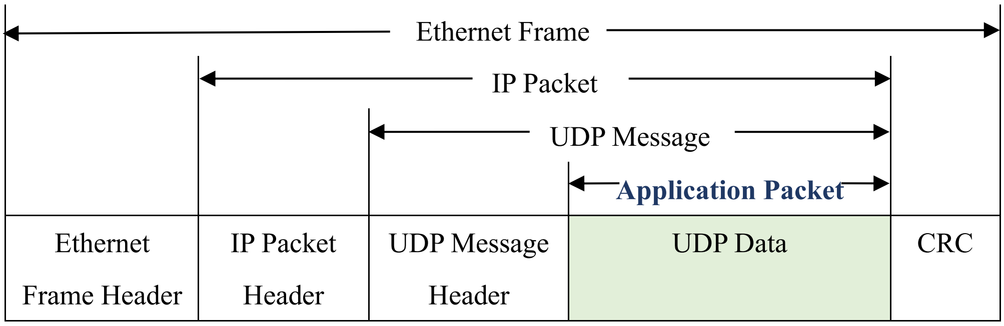
Upon receiving, the receiving and processing module of different types of interfaces unpacks or deframes the data, extracts the application packet, and transmits it to the system queue.

SDLC-BOARD's forwarding engine will read the application packet and transmit it to the transmission module of each interface according to the forwarding configuration. It transmits modules for framing or packing operation on application packets to generate different types of protocol packets or data frames, which will be transmit out through the physical interface.



7.1.2 UDP message format

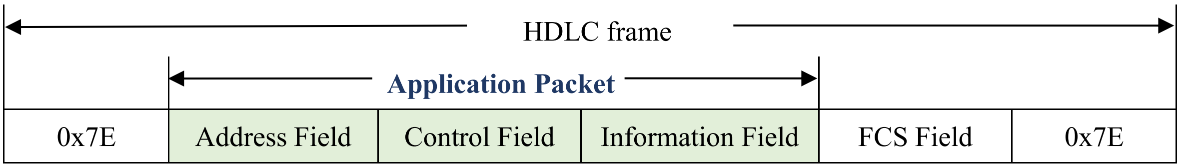
In the UDP protocol, the application packet is packaged in the data area of the UDP message. Each UDP packet contains a complete application packet.



7.1.3 HDLC frame format

A complete HDLC frame consists of several fields between the leading flag and the closing flag, including address field, control field, information field and FCS field for CRC check.

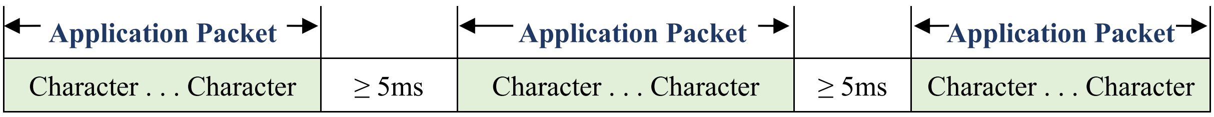
For SDLC-BOARD, instead of distinguishing between address field, control field, and information field, they are uniformly presented as application packets to the upper application to fill in and process the UART packet format.



7.1.4 UART data packet

When the serial port is working in the asynchronous UART mode, there is a character stream without head or tail received from the serial port, where there is no information used to perform unpacking or deframing.

SDLC-BOARD adopts the time information for unpacking, allowing users to define the packet interval of UART. For example, if the packet interval is 5ms, when no new characters are received over 5ms, then the packet receiving is considered to be complete.



In the actual application, data transmission is not allowed during the packet interval; otherwise, it may result in a waste of communication bandwidth, and the higher the baud rate is, the more serious the waste is.

7.1.5 UART-HDLC frame format

The UART-HDLC working mode adopts another strategy to provide the unpacking capacity for UART. As shown in the following figure, the data sender calculates the application packet's CRC and adds the 0x7e to the head and tail as the leading and closing flags to form an UART-HDLC frame.

This strategy does not require increasing the additional packet interval and can make full use of the communication bandwidth, but increases the processing complexity of both communication sides.

UART-HDLC frame			
	Application Packet		
opening Flag 0x7E	Character . . . Character	frame check sequence FCS Field	closing Flag 0x7E

As the application packet and FCS field may appear 0x7E, the sender and receiver shall perform the character escape on the application packet and FCS field with the escape rules as follows:

- 0x7E: Escaped to two characters, 0x7D 0x5E
- 0x7D: Escaped to two characters, 0x7D 0x5D
- Other characters: No escape

The escape operation of data transmit is as follows:

Original Data	**Actual Transmit Data**
0x7E	0x7D 0x5E
0x7D	0x7D 0x5D
Others	No change

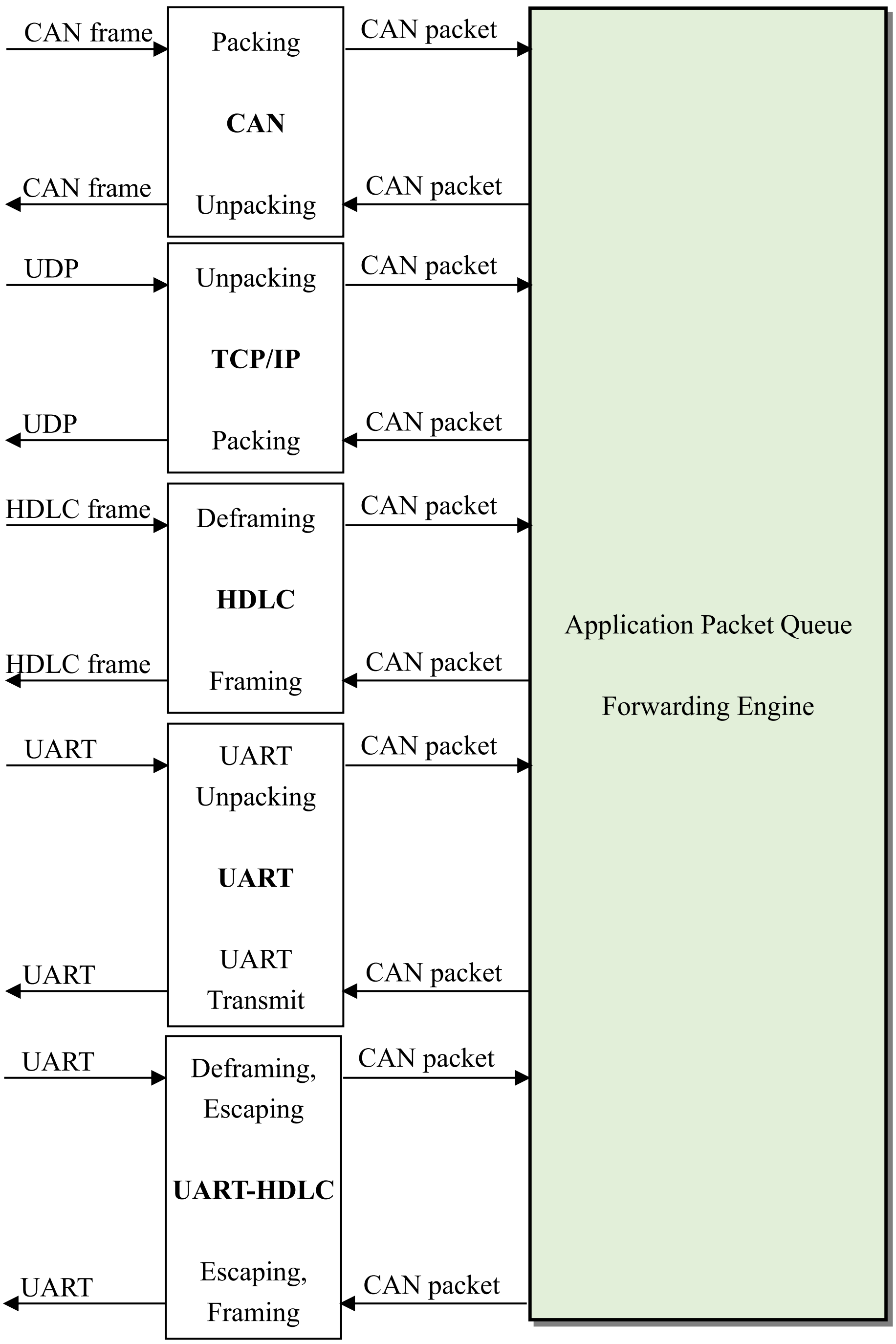
The escape operation of data transmit is as follows:

Actual Receive Data	**Data**
0x7D 0x5E	0x7E
0x7D 0x5D	0x7D
Others	No change

7.2 CAN frame data conversion

7.2.1 CAN data packets and conversion model

Because CAN frames are very short, in order to improve forwarding efficiency, multiple CAN data frames are formed into a CAN packet, and each CAN packet corresponds to an application packet of other interfaces.



7.2.2 CAN packet format

7.2.2.1 Packet format

CAN packet consists of 1 ~ 50 CAN frames with the fixed length of each CAN frame as 13 bytes.

CAN packet (1 ~ 50 CAN frames)

CAN frame	CAN frame	CAN frame	CAN frame	…	CAN frame

	ID0	ID1	ID2	ID3	D0	D1	D2	D3	D4	D5	D6	D7
Frame information	frame ID (4 bytes)				frame (8 bytes)							

CAN frame, 13 bytes fixed length

7.2.2.2 Frame information

The frame information is 1-byte long, with the format defined as follows:

- FF: Identification of the standard frame and the extended frame, 1 for the extended frame, 0 for the standard frame;
- RTR: Identification of the remote frame and the data frame, 1 for the remote frame, 0 for the data frame;
- DLC: Length of the CAN actual data.

Frame Message	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	FF	RTR	Reserved	Reserved	DLC.3	DLC.2	DLC.1	DLC.0

7.2.2.3 Frame ID

The frame ID occupies 4 bytes, but the ID bit number of the standard and extended frames differs.

Standard frame ID: 11 bits for the standard frame ID, with the value range of 0x000 ~ 0x7FF, valid fill range of ID.10 ~ ID.0.

Standard frame ID 4 bytes		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	ID0								
	ID1								
	ID2						ID.10	ID.9	ID.8
	ID3	ID.7	ID.6	ID.5	ID.4	ID.3	ID.2	ID.1	ID.0

Extended frame ID: 29 bits for the Extended frame ID, with the value range of 0x00000000 ~ 0x1FFFFFFF, valid fill range of ID.28 ~ ID.0.

Standard frame ID 4 bytes		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	ID0				ID.28	ID.27	ID.26	ID.25	ID.24
	ID1	ID.23	ID.22	ID.21	ID.20	ID.19	ID.18	ID.17	ID.16
	ID2	ID.15	ID.14	ID.13	ID.12	ID.11	ID.10	ID.9	ID.8
	ID3	ID.7	ID.6	ID.5	ID.4	ID.3	ID.2	ID.1	ID.0

7.2.2.4 Frame data

The frame data occupies 8-byte space with the effective data length of 0 ~ 8 bytes; the first byte is the starting byte of the valid data, and the effective length is determined by the DLC value in the frame information.

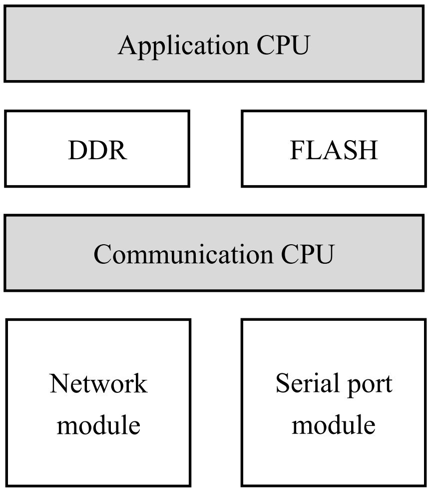
Frame data 8 bytes	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
	Data 0	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7

8 Secondary Development

8.1 Double CPU architecture

The SDLC-BOARD core contains two separate CPUs that interact with data through Shared memory:

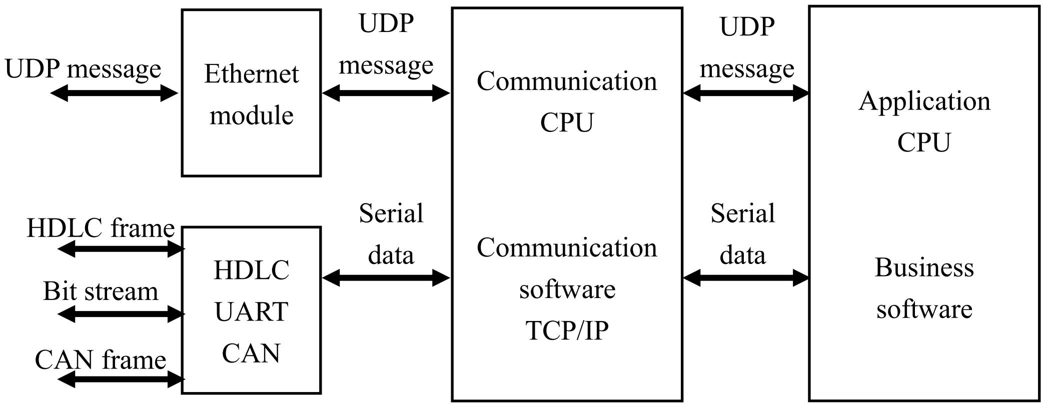
- Communication CPU: Provide network, serial communication support, provide configuration management support
- Application CPU: Run the business software of user secondary development, to process the data from the Communication CPU



8.2 Data interaction model

The system data flow is shown below, including:

- UDP receiving process: The TCP/IP protocol stack of Communication CPU receives UDP messages, which are converted into UDP messages and sent to Application CPU through the Shared memory;
- UDP sending process: Application CPU sends UDP messages to Communication CPU through the Shared memory, which is processed by the TCP/IP protocol stack of Communication CPU and converted into UDP messages which are sent through the Ethernet module;
- Serial port receiving process: Communication CPU receives data through serial port module, and gives it to application CPU to read and process through Shared memory;
- Serial port sending process: Application CPU sends the serial port data to the Communication CPU through the Shared memory, and then sends out the grouped frames through the serial port module group frame.

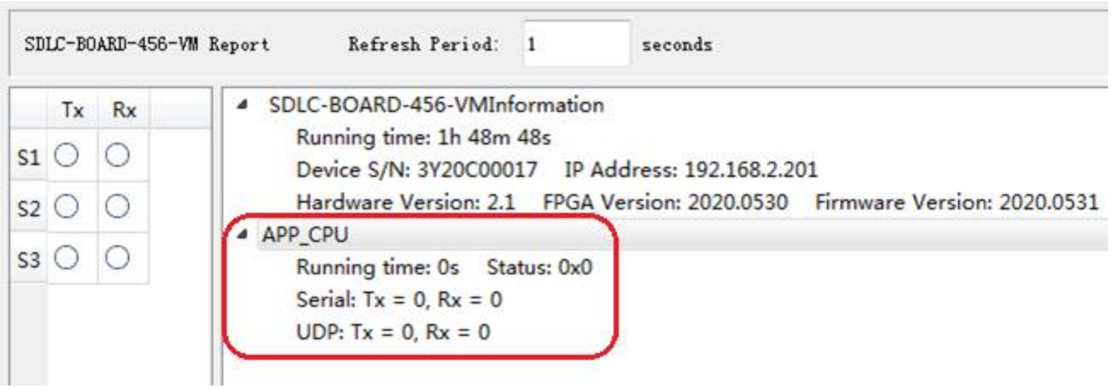


8.3 DMS support

8.3.1 Display the running information of Application CPU

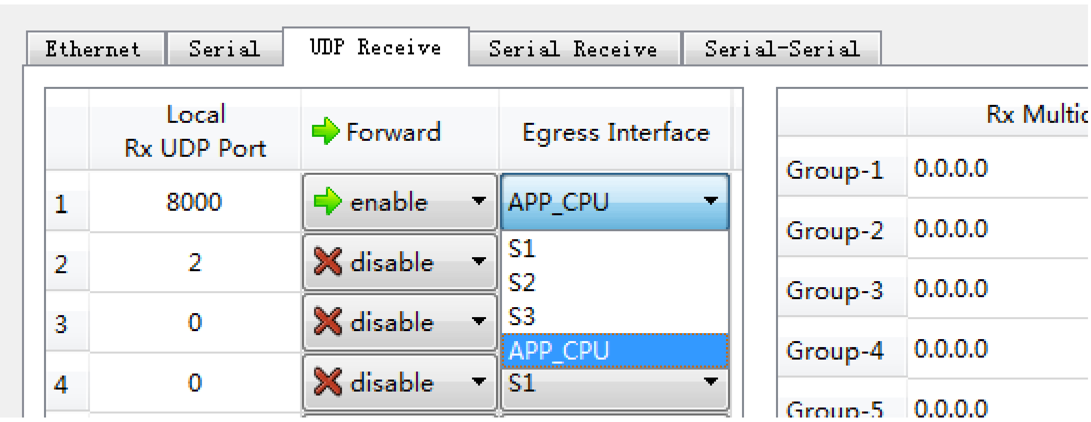
DMS statistical report can display the running information of Application CPU (APP_CPU):

- Running time: Elapsed time since the Application CPU was started;
- Status: A status code of Application CPU, the specific meaning is defined by the user;
- Serial port transmit/receive statistics: The count of data packet transmitted or received by Application CPU from serial port;
- UDP transmit/receive statistics: The count of data packet transmitted or received by Application CPU from UDP;
- APP_CPU: A string of running status.



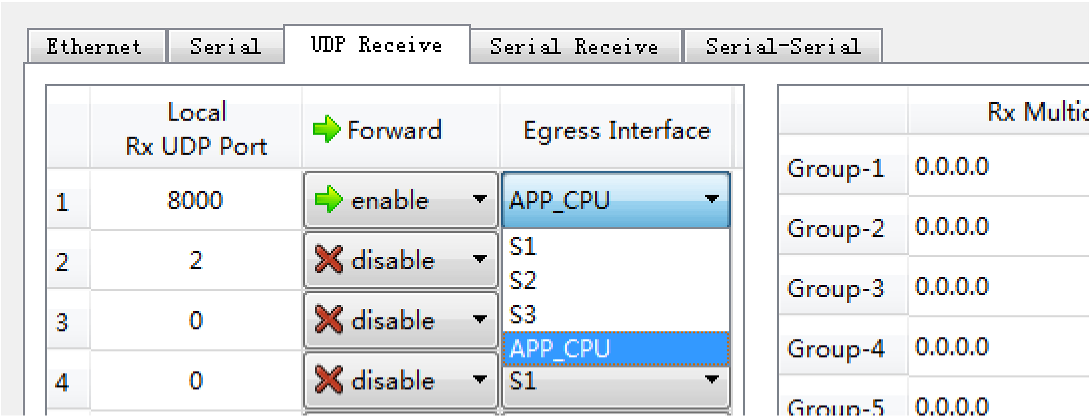
8.3.2 Application CPU receiving configuration of UDP message

As shown in the figure below, when the 'output interface' of UDP receive selects APP_CPU, the UDP message received from port 8000 is forwarded to the application CPU for processing.



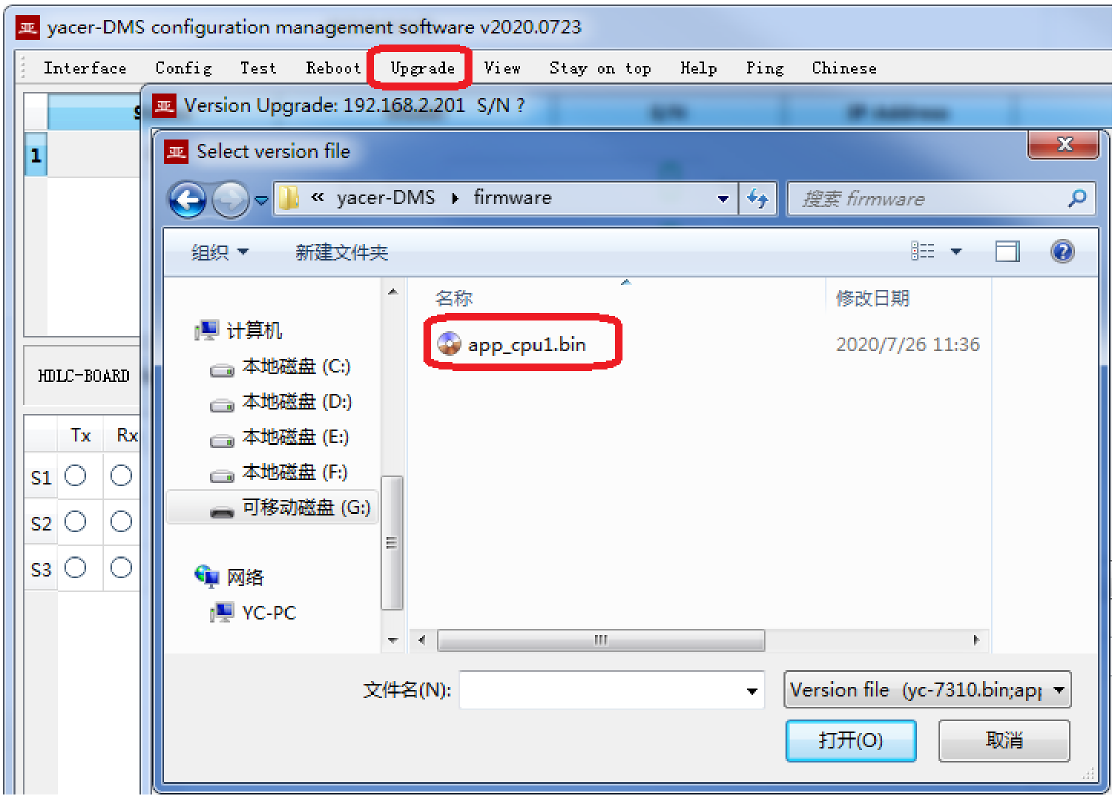
8.3.3 Application CPU receiving configuration of Serial data

As shown in the figure below, the 'Forward to' of serial port S1 is enabled, if the remote IP address or remote UDP port is 0, the serial port data received from S1 will be forwarded to the application CPU for processing.



8.3.4 Load Application CPU software version

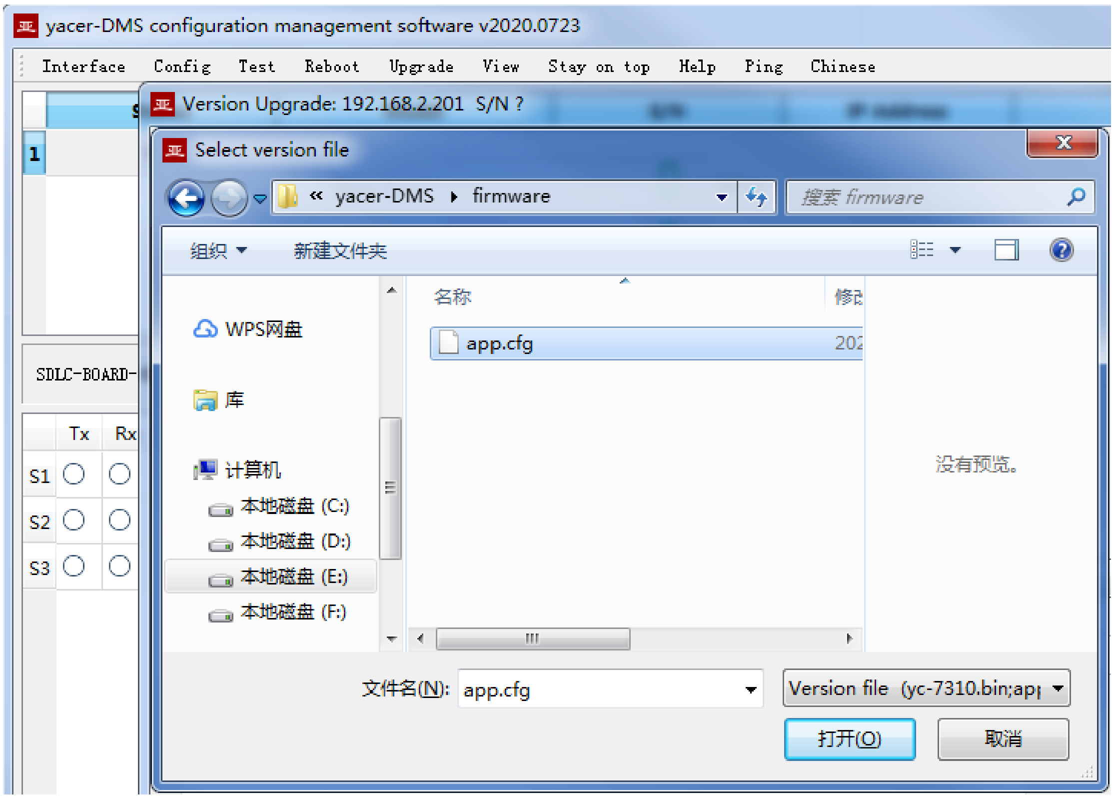
The software version developed by the user generates a version file named APP_cpu1.bin, which is uploaded and burned with the help of DMS version update function. After burning, the device can be reset to load and run the latest Application CPU software version.



8.3.5 Load Application CPU configuration

Configuration file of Application CPU must be configured as app.cfg, its content and format defined by the user, and its length must not exceed 1 M byte.

DMS can also be used to upload and burn configuration files to FLASH, when the system reset, the new configuration can be enabled.



8.4 Software development and compilation

Please contact the manufacturer for technical support.