PDS/tDS-700 Series

Linux User Manual

Warranty

All products manufactured by ICP DAS are warranted against defective materials for a period of one year from the date of delivery to the original purchaser.

Warning

ICP DAS assume no liability for damages consequent to the use of this product. ICP DAS reserves the right to change this manual at any time without notice. The information furnished by ICP DAS is believed to be accurate and reliable. However, no responsibility is assumed by ICP DAS for its use, nor for any infringements of patents or other rights of third parties resulting from its use.

Copyright

Trademark

The names used for identification only may be registered trademarks of their respective companies.

Tables of Contents

1. PDS/tDS-700 Static Library Function Description

Before user use the library, user should configure and check the network configuration of PDS/tDS-700 module with demo “eSearch” first. The static library is the collection of function calls of the PDS/tDS-700 series for linux kernel 2.6.x system. The application structure is presented as below figure “Figure 1-1”. The user application program developed by C (C++) language can call library “libpds700.a” for x86 or “libpds700_64.a” for x64 or “libpds700_arm.a” for arm linux PC. And then static library will call the module command to access the hardware system.

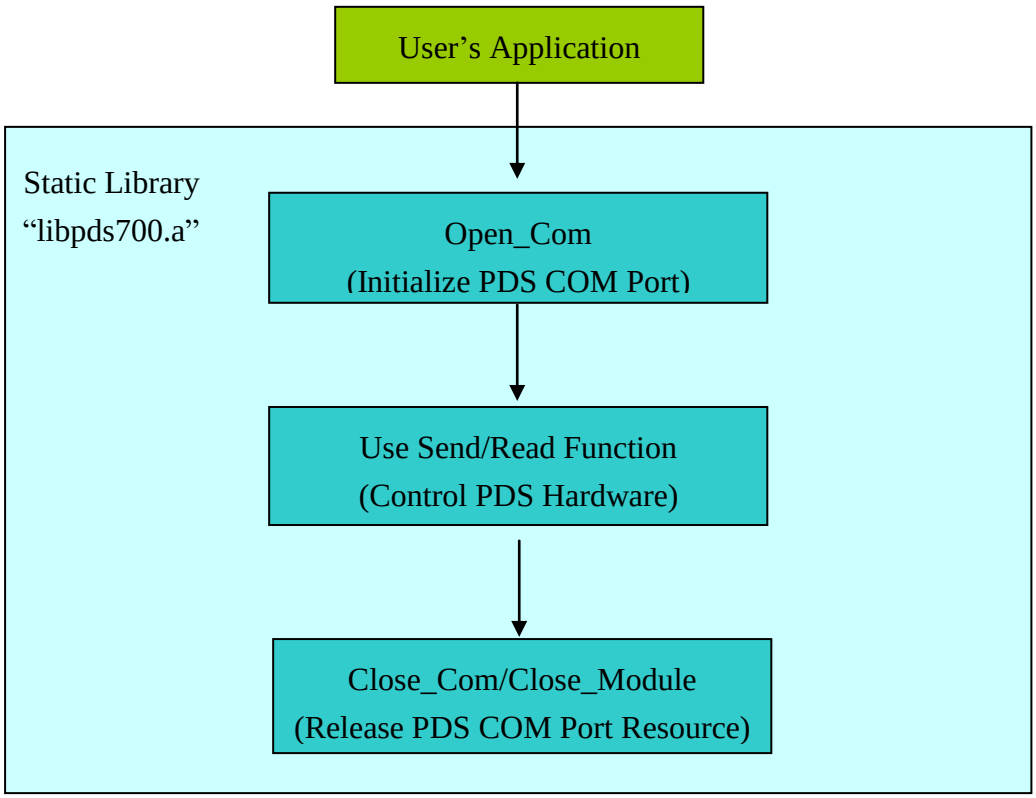


Figure 1-1

1.1 Table of ErrorCode and ErrorString

Table 1.1

Error Code	Error ID	Error String
0	PDS700_NOERROR	OK (No error!)
1	PDS700_OPENCOMERROR	Initializing PDS COM error
2	PDS700_CMDPORTERROR	The PDS/tDS-700 command port over range
3	PDS700_COMNUMBERERROR	The PDS/tDS-700 COM port over range
4	PDS700_OPENSOCKETERROR	Creating socket error
5	PDS700_CONNECTSOCKETERROR	Connecting socket error
6	PDS700_GETHOSTNAMEERROR	Getting PDS/tDS-700 host name error
7	PDS700_SETCOMBAUDERROR	Setting the baud rate of COM port error
8	PDS700_SETCOMDATAFORMATERROR	Setting the data format of COM port error
9	PDS700_SETCMDRETURNMODEERROR	Setting the response mode of PDS/tDS-700 error
10	PDS700_CLOSECOMSOCKETERROR	Closing COM socket error
11	PDS700_CLOSECMDSOCKETERROR	Closing command socket error
12	PDS700_SENDSTRINGERROR	Sending string from the PDS/tDS-700 COM port error
13	PDS700_READSTRINGERROR	Reading string from the PDS/tDS-700 COM port error
14	PDS700_SETSERVERIPERROR	Setting the IP address of PDS/tDS-700 error
15	PDS700_SETSERVERGATEWAYERROR	Setting the gateway of PDS/tDS-700 error
16	PDS700_SETSERVERSUBMASKER	Setting the sub-network

	ROR	mask of PDS/tDS-700 error
17	PDS700_CLEARBUFFERERROR	Clear COM port buffer error
18	PDS700_NORESPONSE	The PDS/tDS-700 no response

1.2 Function Descriptions

Table 1.2

Library Function
WORD Open_Com(pds_mod_t *PDSMod, char *ServerIP, WORD CommandPort, WORD ServerCOM, DWORD Baudrate, BYTE Data, BYTE Parity, BYTE Stop);
WORD Close_Com(pds_mod_t *PDSMod, WORD ServerCOM);
WORD Open_CAN(pds_mod_t *PDSMod, char *ServerIP);
WORD Close_CAN(pds_mod_t *PDSMod);
WORD Close_Module(pds_mod_t *PDSMod);
char * Get_PDS_Library_Version(void);
WORD Get_PDS_MiniOS_Version(pds_mod_t *PDSMod, char *MiniOSVer);
WORD Get_PDS_Vxcomm_Version(pds_mod_t *PDSMod, char *VxcommVer);
WORD Get_PDS_Module_Name(pds_mod_t *PDSMod, char *ModuleName);
WORD Get_PDS_COM_Info(pds_mod_t *PDSMod, WORD ServerCOM);
WORD Get_PDS_Server_Network_Info(pds_mod_t *PDSMod);
WORD Reset_PDS_Module(pds_mod_t *PDSMod);
WORD Set_PDS_Server_IP(pds_mod_t *PDSMod, char *NewServerIP);
WORD Set_PDS_Server_Gateway(pds_mod_t *PDSMod, char *NewServerGateway);
WORD Set_PDS_Server_SubMask(pds_mod_t *PDSMod, char *NewServerSubMask);
WORD Clear_PDS_Buffer(pds_mod_t *PDSMod, int Port);
WORD Send_Read_String(pds_mod_t *PDSMod, WORD ServerCOM, char Out_String[], char In_String[]);

WORD Send_String(pds_mod_t *PDSMod, WORD ServerCOM, char Out_String[]);
WORD Read_String(pds_mod_t *PDSMod, WORD ServerCOM, char In_String[]);
WORD Send_Modbus(pds_mod_t *PDSMod, WORD ServerCOM, char Out_String[], char Out_len);
WORD Read_Modbus(pds_mod_t *PDSMod, WORD ServerCOM, char In_String[], char In_len);
WORD Send_Binary(pds_mod_t *PDSMod, WORD ServerCOM, char Out_String[], int len);
WORD Read_Binary(pds_mod_t *PDSMod, WORD ServerCOM, char In_String[], int len);
WORD Send_CAN_Msg(pds_mod_t *PDSMod, char* can_msg);
WORD Read_CAN_Msg(pds_mod_t *PDSMod, char *In_String);

1.3 Library FUNCTIONS

1.3.1 Open_Com

- **Description:**
  This function is used to configure and open a PDS/tDS-700 Server COM port for service by Ethernet. The COM port must be called up before users can begin sending/receiving commands through it.
- **Syntax:**
  WORD Open_Com(pds_mod_t *PDSMod, char *ServerIP, WORD CommandPort, WORD ServerCOM, DWORD Baudrate, BYTE Data, BYTE Parity, BYTE Stop);
- **Parameter:**
  PDSMod: [output] A structure pointer.
  ServerIP: [input] The IP address of PDS/tDS-700 server.
  CommandPort: [input] PDS/tDS-700 command port is 10000.
  ServerCOM: [input] The number of PDS/tDS-700 COM port.
  Baudrate: [input] The baud of PDS/tDS-700 COM port.
  Data: [input] 5/6/7/8 data bit
  Parity: [input] 0=None Parity, 1=Odd Parity, 2=Even Parity
  Stop: [input] 1=1-stop, 2=2-stop
- **Return:**
  Return 0 means success, others means failure (Please refer to "Section 1.1 Error Code").

1.3.2 Close_Com

- **Description:**
  This function is used to close and release COM port resources. It must be called before exiting the application program.
- **Syntax:**
  WORD Close_Com(pds_mod_t *PDSMod, WORD ServerCOM);
- **Parameter:**
  PDSMod: [input] A structure pointer.
  ServerCOM: [input] The number of PDS/tDS-700 COM port.
- **Return:**
  Return 0 means success, others means failure (Please refer to "Section

1.1 Error Code").

1.3.3 Open_CAN

- **Description:**
  Open CAN device for service by Ethernet (support I-7054D for now).
  You should call up before send & read can messages.
- **Syntax:**
  WORD Open_CAN(pds_mod_t *PDSMod, char *ServerIP)
- **Parameter:**
  PDSMod: [input] A structure pointer.
  ServerIP: [input] The IP address of PDS/tDS-700 server
- **Return:**
  Return 0 means success, others means failure (Please refer to "Section 1.1 Error Code").

1.3.4 Close_CAN

- **Description:**
  This function is used to close and release CAN port resources.
  It must be called before exiting the application program.
- **Syntax:**
  WORD Close_CAN(pds_mod_t *PDSMod)
- **Parameter:**
  PDSMod: [input] A structure pointer.
- **Return:**
  Return 0 means success, others means failure (Please refer to "Section 1.1 Error Code").

1.3.5 Close_Module

- **Description:**
  To release PDS/tDS-700 module and all COM ports resources.
- **Syntax:**
  WORD Close_Module(pds_mod_t *PDSMod);
- **Parameter:**
  PDSMod: [input] A structure pointer.
- **Return:**
  Return 0 means success, others means failure (Please refer to "Section 1.1 Error Code").

1.3.6 Get_PDS_Library_Version

- **Description :**
  To get the version of PDS/tDS-700 linux library.
- **Syntax :**
  char * Get_PDS_Library_Version(void);
- **Parameter :**
  None
- **Return:**
  Return the PDS/tDS-700 library version.

1.3.7 Get_PDS_MiniOS_Version

- **Description :**
  To read the MiniOS7 version of PDS/tDS-700 server.
- **Syntax :**
  WORD Get_PDS_MiniOS_Version(pds_mod_t *PDSMod, char *MiniOSVer);
- **Parameter :**
  PDSMod: [input] A structure pointer.
  MiniOSVer: [output] The MiniOS7 version of PDS/tDS-700 server.
- **Return:**
  Return 0 means success, others means failure (Please refer to "Section 1.1 Error Code").

1.3.8 Get_PDS_Vxcomm_Version

- **Description :**
  To read the VxComm.exe version of PDS/tDS-700 server.
- **Syntax :**
  WORD Get_PDS_Vxcomm_Version(pds_mod_t *PDSMod, char *VxcommVer);
- **Parameter :**
  PDSMod: [input] A structure pointer.
  VxcommVer: [output] The VxComm.exe version of PDS/tDS-700 server.
- **Return:**
  Return 0 means success, others means failure (Please refer to "Section 1.1 Error Code").

1.3.9 Get_PDS_Module_Name

- **Description :**
  To read the PDS/tDS-700 module name.
- **Syntax :**
  WORD Get_PDS_Module_Name(pds_mod_t *PDSMod, char *ModuleName);
- **Parameter :**
  PDSMod: [input] A structure pointer.
  ModuleName: [output] The name of PDS/tDS-700 module.
- **Return:**
  Return 0 means success, others means failure (Please refer to "Section 1.1 Error Code").

1.3.10 Get_PDS_COM_Info

- **Description :**
  To read the COM port Configuration of PDS/tDS-700 server.
- **Syntax :**
  WORD Get_PDS_COM_Info(pds_mod_t *PDSMod, WORD ServerCOM);
- **Parameter :**
  PDSMod: [input] A structure pointer.
  ServerCOM: [input] The number of PDS/tDS-700 COM port.
- **Return:**
  Return 0 means success, others means failure (Please refer to "Section 1.1 Error Code").

1.3.11 Get_PDS_Server_Nerwork_Info

- **Description :**
  To read the network configuration of PDS/tDS-700 server.
- **Syntax :**
  WORD Get_PDS_Server_Network_Info(pds_mod_t *PDSMod);
- **Parameter :**
  PDSMod: [input] A structure pointer.None.
- **Return:**
  Return 0 means success, others means failure (Please refer to "Section

1.1 Error Code").

1.3.12 Reset_PDS_Module

- **Description :**
  To restart the PDS/tDS-700 server.
- **Syntax :**
  WORD Reset_PDS_Module(pds_mod_t *PDSMod);
- **Parameter :**
  PDSMod: [input] A structure pointer.None.
- **Return:**
  Return 0 means success, others means failure (Please refer to "Section 1.1 Error Code").

1.3.13 Set_PDS_Server_IP

- **Description :**
  To configure the IP address of PDS/tDS-700 server.
- **Syntax :**
  WORD Set_PDS_Server_IP(pds_mod_t *PDSMod, char *NewServerIP);
- **Parameter :**
  PDSMod: [input] A structure pointer.
  NewServerIP: [input] The new IP address of PDS/tDS-700 server.
- **Return:**
  Return 0 means success, others means failure (Please refer to "Section 1.1 Error Code").

1.3.14 Set_PDS_Server_Gateway

- **Description :**
  To configure the new gateway of PDS/tDS-700 server.
- **Syntax :**
  WORD Set_PDS_Server_Gateway(pds_mod_t *PDSMod, char *NewServerGateway);
- **Parameter :**
  PDSMod: [input] A structure pointer.
  NewServerGateway: [input] The new gateway of PDS/tDS-700 server.
- **Return:**
  Return 0 means success, others means failure (Please refer to "Section

1.1 Error Code").

1.3.15 Set_PDS_Server_SubMask

- **Description :**
  To configure the new sub-network mask of PDS/tDS-700 server.
- **Syntax :**
  WORD Set_PDS_Server_SubMask(pds_mod_t *PDSMod, char *NewServerSubMask);
- **Parameter :**
  PDSMod: [input] A structure pointer.
  NewServerSubMask: [input] The new submask of PDS/tDS-700 server.
- **Return:**
  Return 0 means success, others means failure (Please refer to "Section 1.1 Error Code").

1.3.16 Clear_PDS_Buffer

- **Description :**
  Clear comport buffer of PDS/tDS-700 server.
- **Syntax :**
  WORD Clear_PDS_Buffer(pds_mod_t *PDSMod, int Port);
- **Parameter :**
  PDSMod: [input] A structure pointer.
  Port: [input] Comport number of PDS/tDS-700 server.
- **Return:**
  Return 0 means success, others means failure (Please refer to "Section 1.1 Error Code").

1.3.17 Send_Read_String

- **Description :**
  To send and receive string from PDS/tDS-700 server.
- **Syntax :**
  WORD Send_Read_String(pds_mod_t *PDSMod, WORD ServerCOM, char Out_String[], char In_String[]);
- **Parameter :**
  PDSMod: [input] A structure pointer.
  ServerCOM: [input] The COM port number of PDS/tDS-700 server.
  Out_String: [input] The sending string.

In_String: [Output] Receiving the response string from the modules.

- **Return:**
  Return 0 means success, others means failure (Please refer to "Section 1.1 Error Code").

1.3.18 Send_String

- **Description :**
  To send string from the COM port of PDS/tDS-700 server.
- **Syntax :**
  WORD Send_String(pds_mod_t *PDSMod, WORD ServerCOM, char Out_String[]);
- **Parameter :**
  PDSMod: [input] A structure pointer.
  ServerCOM: [input] The COM port number of PDS/tDS-700 server.
  Out_String: [input] The sending string.
- **Return:**
  Return 0 means success, others means failure (Please refer to "Section 1.1 Error Code").

1.3.19 Read_String

- **Description :**
  To receive from the COM port of PDS/tDS-700 server.
- **Syntax :**
  WORD Read_String(pds_mod_t *PDSMod, WORD ServerCOM, char In_String[]);
- **Parameter :**
  PDSMod: [input] A structure pointer.
  ServerCOM: [input] The COM port number of PDS/tDS-700 server.
  In_String: [output] The receiving string from the PDS/tDS-700 server.
- **Return:**
  Return 0 means success, others means failure (Please refer to "Section 1.1 Error Code").

1.3.20 Send_Modbus

- **Description :**
  To send Modbus protocol from the COM port of PSD-700 server.
- **Syntax :**
  WORD Send_Modbus(pds_mod_t *PDSMod, WORD ServerCOM, char Out_String[], char Out_len);
- **Parameter :**
  PDSMod: [input] A structure pointer.
  ServerCOM: [input] The COM port number of PDS/tDS-700 server.
  Out_String: [input] The sending string.
  Out_len:[input] The “Out_String “ string length.
- **Return:**
  Return 0 means success, others means failure (Please refer to "Section 1.1 Error Code").

1.3.21 Read_Modbus

- **Description :**
  To receive Modbus protocol from the COM port of PDS/tDS-700 server.
- **Syntax :**
  WORD Read_Modbus(pds_mod_t *PDSMod, WORD ServerCOM, char In_String[], char In_len);
- **Parameter :**
  PDSMod: [input] A structure pointer.
  ServerCOM: [input] The COM port number of PDS/tDS-700 server.
  In_String: [output] The receiving string from the PDS/tDS-700 server.
  In_len:[input] The “In_String” string length.
- **Return:**
  Return 0 means success, others means failure (Please refer to "Section 1.1 Error Code").

1.3.22 Send_Binary

- **Description :**
  To send raw data to the COM port of PDS/tDS-700 server.
- **Syntax :**
  WORD Send_Binary(pds_mod_t *PDSMod, WORD ServerCOM, char Out_String[], int len);

- **Parameter :**

  PDSMod: [input] A structure pointer.

  ServerCOM: [input] The COM port number of PDS/tDS-700 server.

  Out_String: [output] The sending string.

  In_len:[input] The “Out_String” string length.

- **Return:**

  Return 0 means success, others means failure (Please refer to "Section 1.1 Error Code").

1.3.23 Read_Binary

- **Description :**

  To receive raw data from the COM port of PDS/tDS-700 server.

- **Syntax :**

  WORD Read_Binary(pds_mod_t *PDSMod, WORD ServerCOM, char In_String[], int len);

- **Parameter :**

  PDSMod: [input] A structure pointer.

  ServerCOM: [input] The COM port number of PDS/tDS-700 server.

  In_String: [output] The receiving string from the PDS/tDS-700 server.

  In_len:[input] The “In_String” string length.

- **Return:**

  Return 0 means success, others means failure (Please refer to "Section 1.1 Error Code").

1.3.24 Send_CAN_Msg

- **Description :**

  To send CAN message from the CAN port of remote server.

- **Syntax :**

  WORD Send_CAN_Msg(pds_mod_t *PDSMod, char* can_msg)

- **Parameter :**

  PDSMod: [input] A structure pointer.

  can_msg: [output] Send CAN message to remote server.

- **Return:**

  Return 0 means success, others means failure (Please refer to "Section 1.1 Error Code").

1.3.25 Read_CAN_Msg

- **Description :**

  To receive raw data from the COM port of PDS/tDS-700 server.

- **Syntax :**

  WORD Read_CAN_Msg(pds_mod_t *PDSMod, char *In_String)

- **Parameter :**

  PDSMod: [input] A structure pointer.

  In_String: [output] The receiving string from the remote server.

- **Return:**

  Return 0 means success, others means failure (Please refer to "Section 1.1 Error Code").

2. PDS/tDS-700 Library Demo For Linux

The PDS/tDS-700 library package provided the demo code in the package directory “examples” (Please refer to Table 2.1).

Table 2.1

Directory Path	File Name	Description
include	pds700.h	The header of PDS/tDS-700 library.
lib	libpds700.a	PDS/tDS-700 library for x86 Linux PC.
	libpds700_64.a	PDS/tDS-700 library for x64 Linux PC.
	libpds700_arm.a	PDS/tDS-700 library for arm x86 Linux PC.
doc	PDS700-Linux-Manual.pdf	The linux manual for PDS/tDS-700 library.
examples	pds_info.c	Show PDS/tDS-700 server information.
	pds_send_read.c pds_send_read2.c	Sending and Receiving string from PDS/tDS-700 server.
	pds_send_read_modbus.c	Sending and Receiving Modbus protocol from PDS/tDS-700 server.
	pds_send_read_binary.c	Sending and Receiving raw data from PDS/tDS-700 server.
	pds_set_net	Reset the network configuration of PDS/tDS-700 server.
	pds_send_can	Send CAN message to remote server.
	pds_read_can	Receive CAN message from remote server.
	eSearch	Get current subnet pds/tds modules information (IP, Submask, Gateway, MAC address)
	pds_clear_buffer	Clear a specific comport buffer

2.1 Demo code “pds_info”

This demo program would show the information of PDS/tDS-700 server. Please refer to figure 2-1

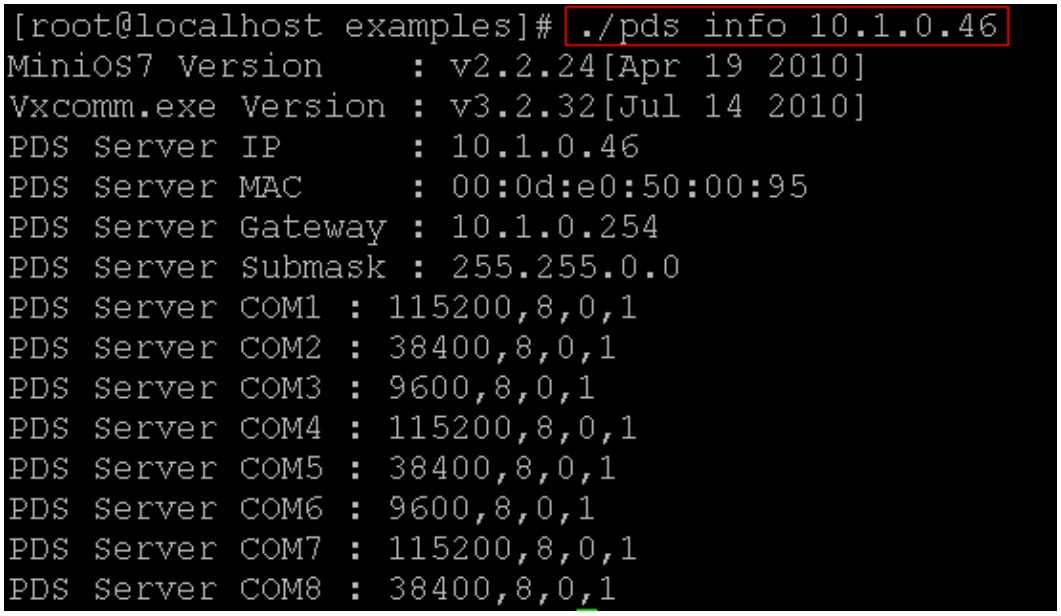


Figure 2-1

2.2 Demo code “pds_send_read”

This “pds_send_read” and “pds_send_read2” demo program would send and receive string from the COM port 1 of PDS/tDS-700 server. Please refer to figure 2-2.

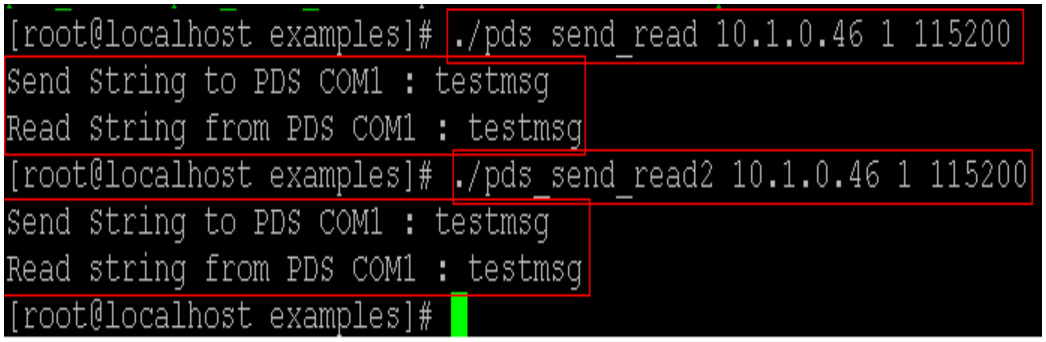


Figure 2-2

2.3 Demo code “pds_send_read_modbus”

This demo program would send and receive modbus protocol from the COM port 1 of PDS/tDS-700 server. Please refer to figure 2-3.

```
[root@localhost examples]# ./pds_send_read_modbus 10.1.0.17 1 115200
Send modbus to PDS COM1 :
send_cmd[0] 0x1    NetID
send_cmd[1] 0x3    Function code
send_cmd[2] 0x1    Modbus address high byte
send_cmd[3] 0xe2   Modbus address low byte
send_cmd[4] 0x0    Channel number high byte
send_cmd[5] 0x2    Channel number low byte
send_cmd[6] 0x65   CRC
send_cmd[7] 0xc1   CRC
Read modbus from PDS COM1 :
receive_cmd[0] 0x1     NetID
receive_cmd[1] 0x3     Function code
receive_cmd[2] 0x4     Byte conunt
receive_cmd[3] 0x17    Data Hi (address 482)
receive_cmd[4] 0x0     Data Low (address 482)
receive_cmd[5] 0x0     Data Hi (address 483)
receive_cmd[6] 0x70    Data Low (address 483)
receive_cmd[7] 0xfe    CRC
receive_cmd[8] 0x63    CRC
```

Figure 2-3

2.4 Demo code “pds_send_read_binary”

This demo program would send and receive raw data from the COM port 1 of PDS-tDS-700 server. Please refer to figure 2-4.

```
[root@localhost examples]# ./pds_send_read_binary 10.1.0.58 1
Send Binary to PDS COM1 :   0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
Read Binary from PDS COM1 : 0 1 2 3 4 5 6 7 8 9
Read Binary from PDS COM1 : 10 11 12 13 14 15 16 17 18 19
```

Figure 2-4

2.5 Demo code “pds_set_net”

This demo program would reset the network configuration of PDS/tDS-700 server. Please refer to figure 2-5

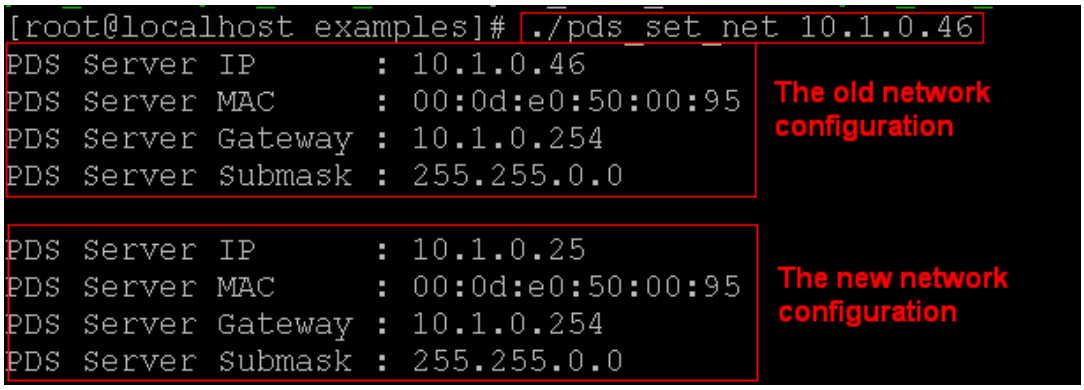
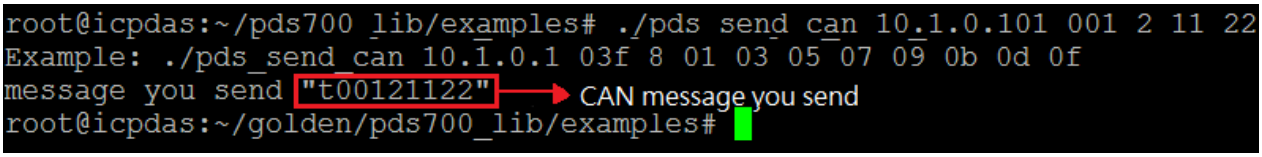


Figure 2-5

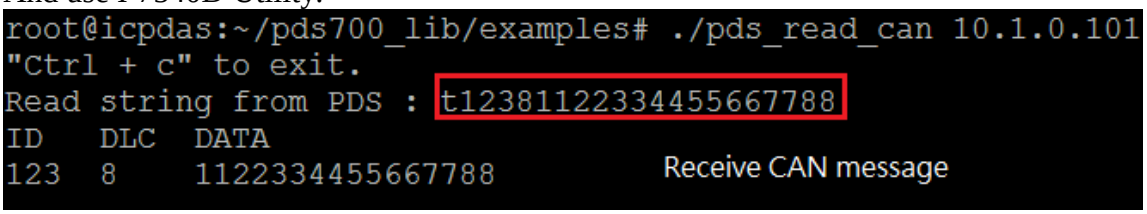
2.6 Demo code “pds_send_can”

This demo program have to set IP & baud rate first, you can set above two parameter in Windows, refer to https://www.icpdas.com/web/product/download/industrial_communication/can_bus/converter/ethernet/i-7540d/document/manual/I-7540D_UserManual_en.pdf
And use I-7540D Utility.



2.7 Demo code “pds_read_can”

This demo program have to set IP & baud rate first, you can set above two parameter in Windows, refer to https://www.icpdas.com/web/product/download/industrial_communication/can_bus/converter/ethernet/i-7540d/document/manual/I-7540D_UserManual_en.pdf
And use I-7540D Utility.



2.8 Demo code “eSearch”

This demo program shows modules information (IP, submask, gateway) in your sub-netmask

```
root@winson:~/pds700_lib/examples# ./eSearch
List modules in submask
NAME=tDS-715_RevB
IP=10.1.0.9         MASK=255.255.0.0    GATEWAY=10.1.0.254  MAC=00:0d:e0:12:34:56
NAME=PDS-782
IP=10.1.102.235     MASK=255.255.0.0    GATEWAY=10.1.0.254  MAC=00:0d:e0:50:02:9e
NAME=PDS-782
IP=10.1.102.234     MASK=255.255.0.0    GATEWAY=10.1.0.254  MAC=00:0d:e0:50:02:ba
```

2.9 Demo code “pds_clear_buffer”

This demo program shows clear comport buffer status.

```
root@winson-G41M-ES2L:~/pds700_lib/examples# ./pds_clear_buffer 192.168.2.234 1
Clear COM1 buffer OK.
```