I7565H1H2 Linux Software Manual

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. i-7565-H1/H2 Linux Driver Installation

The I-7565-H1/H2 can be used in linux. For Linux O.S, the recommended installation and uninstall steps are given in Sec 1.1 ~ 1.2

1.1 Linux Driver Installing Procedure

(1) LinPAC-8x41

- Type below command to install linux driver

  I-7565-H1 module:

  #**cd /lib/modules/2.6.19**
  #**insmod usbserial vendor=0x1b5c product=0x0201**

  I-7565-H2 module:

  #**cd /lib/modules/2.6.19**
  #**insmod usbserial vendor=0x1b5c product=0x0202**

- Type command “dmesg” to check I-7565-H1/H2 device file(please refer to Figure 1-1)

  #**dmesg**

(2) LinPAC-8x81 or **Linux PC(x86)**

- Type below command to install linux driver

  I-7565-H1 module:

  #**modprobe usbserial vendor=0x1b5c product=0x0201**

  I-7565-H2 module:

  #**modprobe usbserial vendor=0x1b5c product=0x0202**

- Type command “dmesg” to check I-7565-H1/H2 device file(please refer to Figure 1-1)

  #**dmesg**

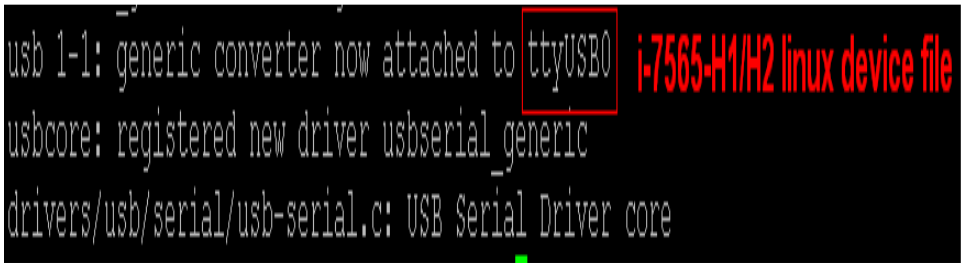


Figure 1-1

1.2 Linux Driver Uninstalling Procedure

(1) LinPAC-8x41

- Type below command to remove i-7565-H1/H2 linux driver.

  #**rmmod usbserial**

(2) LinPAC-8x81 or **Linux PC(x86)**

- Type below command to remove i-7565-H1/H2 linux driver
  #**modprobe –r usbserial**

2. i-7565-H1/H2 Static Library Function Description

The static library is the collection of function calls of the i-7565-H1/H2 for linux kernel 2.6.x system. The application structure is presented as below figure "Figure 2-1". The user application program developed by C (C++) language can call library "libI7565H1H2.a" for LinPAC-8x81(or x86 linux PC) or "libI7565H1H2_arm.a" for LinPAC-8x41 in user mode. And then static library will call the module command to access the hardware system.

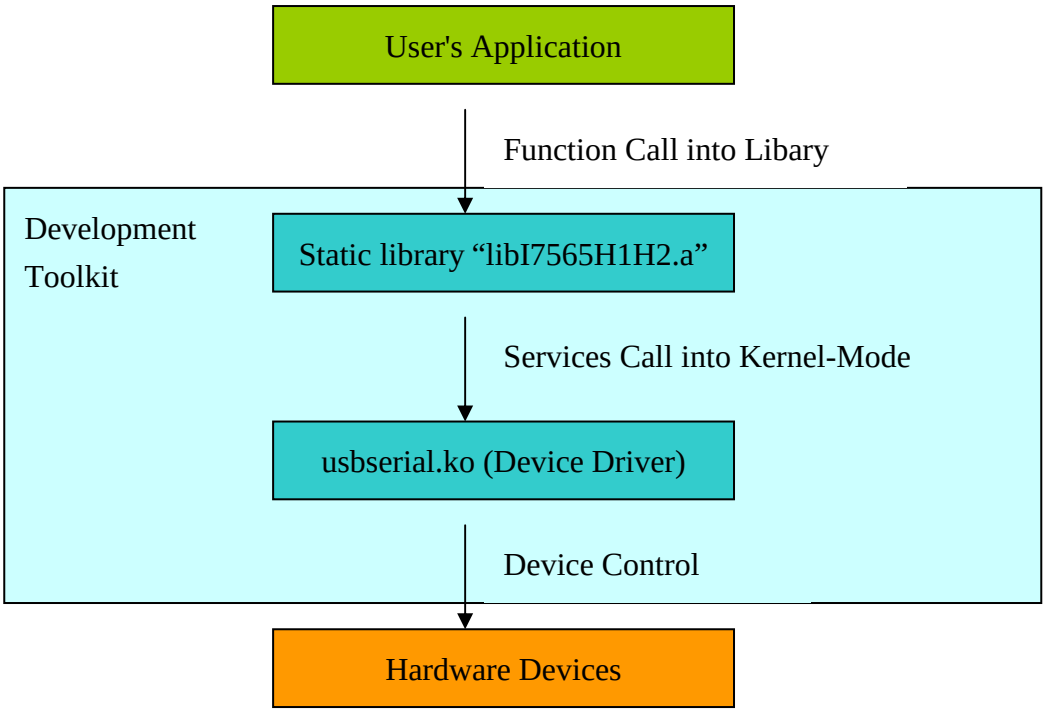


Figure 2-1

2.1 Table of ErrorCode and ErrorString

Table 2.1

Error Code	Error ID	Error String
0	No_Err	OK (No error !)
1	DEV_ModName_Err	The modules error
2	DEV_ModNotExist_Err	The module doesn't exist in this port
3	DEV_PortNotOpen_Err	The port doesn't open
4	CAN_ConFigureFail_Err	CAN hardware configure fail
5	CAN_HARDWARE_Err	CAN hardware Init fail
6	CAN_PortNo_Err	The Device doesn't support this CAN port
7	CAN_FIDLength_Err	The CAN Filter-ID number exceed max number
8	CAN_DevDisconnect_Err	The connection of device is broken
9	CAN_TimeOut_Err	The configure command is timeout
10	CAN_ConFigureCmd_Err	The configure command doesn't support
11	CAN_ConFigureBusy_Err	The configure command is busy
12	CAN_RxBufEmpty	The CAN receive buffer is empty
13	CAN_TxBufFull	The CAN send buffer is full
14	CAN_EnableHWCyclicTxMsg_Err	To enable hardware cyclic fail

2.2 Function Descriptions

Table 2.2

NO	Init Function
1	VCI_OpenCAN
2	VCI_CloseCAN
No	Module Configure Function
1	VCI_Set_CANFID
2	VCI_Get_CANFID
3	VCI_Clr_CANFID

4	VCI_Get_CANStatus
5	VCI_Clr_BufOverflowLED
6	VCI_Get_MODInfo
7	VCI_Rst_MOD
NO	Communication Function
1	VCI_SendCANMsg
2	VCI_RecvCANMsg
3	VCI_EnableHWCyclicTxMsg
4	VCI_DisableHWCyclicTxMsg
NO	Software Buffer Function
1	VCI_Get_RxMsgCnt
2	VCI_Get_RxMsgBufIsFull
3	VCI_Clr_RxMsgBuf
NO	Other Function
1	VCI_Get_DllVer

2.3 Init FUNCTIONS

2.3.1 VCI_OpenCAN

- **Description:**
  To enable the assigned CAN port function of I-7565-H1/H2. After the CAN port function is enabled, users can use “Communication” functions to send / receive CAN messages.

- **Syntax:**

  int VCI_OpenCAN(PVCI_CAN_PARAM pCANPARAM)

- **Parameter:**
  pCANPARAM**:**
  A structure pointer of _VCI_CAN_PARAM is used to set the CAN port communication parameters shown as below.

```
typedef struct _VCI_CAN_PARAM
{
    BYTE DevPort;
    BYTE DevType;
    DWORD CAN1_Baud;
    DWORD CAN2_Baud;
} _VCI_CAN_PARAM, *PVCI_CAN_PARAM;
```

  DevPort**:** The virtual com port number.
  DevType**:** The module type (1**:** I-7565-H1; 2**:** I-7565-H2).

CAN1_Baud**:** CAN1 port baud rate.
(0**:** Disable CAN1 port Others**:** Enable CAN1 port)
CAN2_Baud**:** CAN2 port baud rate.
(0**:** Disable CAN2 port Others**:** Enable CAN2 port)

- **Return:**
  Return 0 means success, others means failure (Please refer to "Section 2.1 Error Code").

2.3.2 VCI_CloseCAN

- **Description:**
  To disable all CAN port function of I-7565-H1/H2. After the CAN port function is disabled, it will not interfere the communication of CAN bus network even if I-7565-H1/H2 is power on.
- **Syntax:**
  int VCI_CloseCAN(BYTE DevPort)
- **Parameter:**
  DevPort**:** The virtual com port number.
- **Return:**
  Return 0 means success, others means failure (Please refer to "Section 2.1 Error Code").

2.4 Module ConFigureure FUNCTIONS

2.4.1 VCI_Set_CANFID

- **Description:**
  To set CAN Filter-ID in the assigned CAN port.
- **Syntax:**
  int VCI_Set_CANFID(BYTE CAN_No, PVCI_CAN_FID pCANFID)
- **Parameter:**
  CAN_No **:** The assigned CAN port number.
  pCANFID**:**
  A structure pointer of _VCI_CAN_FilterID is used to set the CAN Filter-ID data shown as below.

```
typedef struct _VCI_CAN_FilterID
{
    WORD SSFF_Num;
    WORD GSFF_Num;
    WORD SEFF_Num;
    WORD GEFF_Num;
    WORD SSFF_FID[512];
    DWORD GSFF_FID[512];
```

```
    DWORD SEFF_FID[512];
    DWORD GEFF_FID[512];
} _VCI_CAN_FilterID, *PVCI_CAN_FID;
```

SSFF_Num**:** Single 11-bit CAN Filter-ID number
GSFF_Num**:** Group 11-bit CAN Filter-ID number
SEFF_Num**:** Single 29-bit CAN Filter-ID number
GEFF_Num**:** Group 29-bit CAN Filter-ID number
SSFF_FID[512]**:** Single 11-bit CAN Filter-ID data array
GSFF_FID[512]**:** Group 11-bit CAN Filter-ID data array

- **Return:**

  Return 0 means success, others means failure (Please refer to "Section 2.1 Error Code").

2.4.2 VCI_Get_CANFID

- **Description :**
  To get CAN Filter-ID in the assigned CAN port.
- **Syntax :**
  int VCI_Get_CANFID(BYTE CAN_No, PVCI_CAN_FID pCANFID)
- **Parameter :**
  CAN_No**:** The assigned CAN port number.
  pCANFID**:**
  A structure pointer of _VCI_CAN_FilterID is used to receive the CAN Filter-ID data shown as below.

```
typedef struct _VCI_CAN_FilterID
{
    WORD SSFF_Num;
    WORD GSFF_Num;
    WORD SEFF_Num;
    WORD GEFF_Num;
    WORD SSFF_FID[512];
    DWORD GSFF_FID[512];
    DWORD SEFF_FID[512];
    DWORD GEFF_FID[512];
} _VCI_CAN_FilterID, *PVCI_CAN_FID;
```

SSFF_Num**:** Single 11-bit CAN Filter-ID number
GSFF_Num**:** Group 11-bit CAN Filter-ID number
SEFF_Num**:** Single 29-bit CAN Filter-ID number
GEFF_Num**:** Group 29-bit CAN Filter-ID number
SSFF_FID[512]**:** Single 11-bit CAN Filter-ID data array
GSFF_FID[512]**:** Group 11-bit CAN Filter-ID data array
SEFF_FID[512]**:** Single 29-bit CAN Filter-ID data array
GEFF_FID[512]**:** Group 29-bit CAN Filter-ID data array

- **Return:**
  Return 0 means success, others means failure (Please refer to "Section 2.1 Error Code").

2.4.3 VCI_Clr_CANFID

- **Description :**
  To clear CAN Filter-ID in the assigned CAN port.
- **Syntax :**
  int VCI_Clr_CANFID(BYTE CAN_No)
- **Parameter :**
  CAN_No**:** The assigned CAN port number.
- **Return:**
  Return 0 means success, others means failure (Please refer to "Section 2.1 Error Code").

2.4.4 VCI_Get_CANStatus

- **Description :**
  To get the assigned CAN port status
- **Syntax :**
  int VCI_Get_CANStatus(BYTE CAN_No, PVCI_CAN_STATUS pCANStatus)
- **Parameter :**
  CAN_No**:** The assigned CAN port number.
  pCANStatus:
  A structure pointer of _VCI_CAN_STATUS is used to receive the CAN port status shown as below.

```
typedef struct _VCI_CAN_STATUS
{
   DWORD CurCANBaud;
   BYTE CANReg;
   BYTE CANTxErrCnt;
   BYTE CANRxErrCnt;
   BYTE MODState;
   DWORD Reserved;
} _VCI_CAN_STATUS, *PVCI_CAN_STATUS;
```

CurCANBaud**:** Return the assigned CAN port baud rate.
CANReg**:** Return the assigned CAN port register value.

CANTxErrCnt **:** Return the assigned CAN port Tx error count.
CANRxErrCnt **:** Return the assigned CAN port Rx error count.
MODState **:** Return the module state.

- **Return:**

  Return 0 means success, others means failure (Please refer to "Section 2.1 Error Code").

2.4.5 VCI_Clr_BufOverflowLED

- **Description :**
  To clear buffer overflow ERR LED state (flash per second) in the assigned CAN port.
- **Syntax :**

  int VCI_Clr_BufOverflowLED(BYTE CAN_No)
- **Parameter :**

  CAN_No**:** The assigned CAN port number.
- **Return:**

  Return 0 means success, others means failure (Please refer to "Section 2.1 Error Code").

2.4.6 VCI_Get_MODInfo

- **Description :**
  To get the information of module.
- **Syntax :**

  int VCI_Get_MODInfo(PVCI_MOD_INFO pMODInfo)
- **Parameter :**
  pMODInfo:
  A structure pointer of _VCI_MODULE_INFO is used to receive the module information shown as below.

```
typedef struct _VCI_MODULE_INFO
{
    char Mod_ID[12];
    char FW_Ver[12];
} _VCI_MODULE_INFO, *PVCI_MOD_INFO;
```

  Mod_ID[12]**:** Return the module name string.
  FW_Ver[12]**:** Return the module firmware version string.
- **Return:**

  Return 0 means success, others means failure (Please refer to "Section

2.1 Error Code").

2.4.7 VCO_Rst_MOD

- **Description :**
  To reset module.
- **Syntax :**
  int VCI_Rst_MOD(void)
- **Parameter :**
  None.
- **Return:**
  Return 0 means success, others means failure (Please refer to "Section 2.1 Error Code").

2.5 Communication FUNCTIONS

2.5.1 VCI_SendCANMsg

- **Description :**
  To send CAN messages in the assigned CAN port.
- **Syntax :**
  int VCI_SendCANMsg(BYTE CAN_No, PVCI_CAN_MSG pCANMsg)
- **Parameter :**
  CAN_No**:** The assigned CAN port number.
  pCANMsg:
  A structure pointer of _VCI_CAN_MSG is used to set the CAN message parameters shown as below.

```
typedef struct _VCI_CAN_MSG
{
    BYTE Mode;
    BYTE RTR;
    BYTE DLC;
    BYTE Reserved;
    DWORD ID;
    DWORD TimeL;
    DWORD TimeH;
    BYTE Data[8];
} _VCI_CAN_MSG, *PVCI_CAN_MSG;
```

Mode**:** CAN message Mode (0**:** 11-bit**;** 1**:** 29-bit).
RTR**:** CAN message RTR (0**:** No RTR**;** 1**:** RTR).

DLC**:** CAN message Data Length (0~8).
ID**:** CAN message ID.
TimeL**:** CAN message Time-Stamp (Lo-DWORD).
TimeH**:** CAN message Time-Stamp (Hi-DWORD).
Data[8]**:** CAN message Data Array.

- **Return:**

  Return 0 means success, others means failure (Please refer to "Section 2.1 Error Code").

2.5.2 VCI_RecvCANMsg

- **Description :**
  To receive CAN messages that are saved in software buffer in the assigned CAN port.
- **Syntax :**

  int VCI_RecvCANMsg(BYTE CAN_No, PVCI_CAN_MSG pCANMsg)
- **Parameter :**

  CAN_No**:** The assigned CAN port number.
  pCANMsg:
  A structure pointer of _VCI_CAN_MSG is used to receive the CAN message shown as below.

```
typedef struct _VCI_CAN_MSG
{
    BYTE Mode;
    BYTE RTR;
    BYTE DLC;
    BYTE Reserved;
    DWORD ID;
    DWORD TimeL;
    DWORD TimeH;
    BYTE Data[8];
} _VCI_CAN_MSG, *PVCI_CAN_MSG;
```

  Mode**:** CAN message Mode (0**:** 11-bit**;** 1**:** 29-bit).
  RTR**:** CAN message RTR (0**:** No RTR**;** 1**:** RTR).
  DLC**:** CAN message Data Length (0~8).
  ID**:** CAN message ID.
  TimeL**:** CAN message Time-Stamp (Lo-DWORD).
  TimeH**:** CAN message Time-Stamp (Hi-DWORD).
  Data[8]**:** CAN message Data Array.
- **Return:**

  Return 0 means success, others means failure (Please refer to "Section 2.1 Error Code").

2.5.3 VCI_EnableHWCyclicTxMsg

- **Description :**
  To send CAN messages in the assigned CAN port by using module hardware timer and it will be more precise than PC software timer.

- **Syntax :**
  int VCI_EnableHWCyclicTxMsg(BYTE CAN_No, PVCI_CAN_MSG pCANMsg, DWORD TimePeriod, DWORD TransmitTimes)

- **Parameter :**
  CAN_No**:** The assigned CAN port number.
  pCANMsg:
  A structure pointer of _VCI_CAN_MSG is used to set the CAN message parameters shown as below.

```
typedef struct _VCI_CAN_MSG
{
    BYTE Mode;
    BYTE RTR;
    BYTE DLC;
    BYTE Reserved;
    DWORD ID;
    DWORD TimeL;
    DWORD TimeH;
    BYTE Data[8];
} _VCI_CAN_MSG, *PVCI_CAN_MSG;
```

  Mode**:** CAN message Mode (0**:** 11-bit**;** 1**:** 29-bit).
  RTR**:** CAN message RTR (0**:** No RTR**;** 1**:** RTR).
  DLC**:** CAN message Data Length (0~8).
  ID**:** CAN message ID.
  TimeL**:** CAN message Time-Stamp (Lo-DWORD).
  TimeH**:** CAN message Time-Stamp (Hi-DWORD).
  Data[8]**:** CAN message Data Array.

  TimePeriod**:** The time period of module hardware timer for sending CAN message. If the value is zero, this function doesn't work.

  TransmitTimes**:** The count for sending CAN message. If the value is zero, it means that CAN message will be sent periodically and permanently.

- **Return:**
  Return 0 means success, others means failure (Please refer to "Section 2.1 Error Code").

2.5.4 VCI_DisableHWCyclicTxMsg

- **Description :**
  To stop sending CAN messages by module hardware timer.
- **Syntax :**
  int VCI_DisableHWCyclicTxMsg(PVCI_CAN_PARAM pCANPARAM);
- **Parameter :**
  pCANPARAM**:**
  A structure pointer of _VCI_CAN_PARAM is used to set the CAN port communication parameters shown as below.

```
typedef struct _VCI_CAN_PARAM
{
    BYTE DevPort;
    BYTE DevType;
    DWORD CAN1_Baud;
    DWORD CAN2_Baud;
} _VCI_CAN_PARAM, *PVCI_CAN_PARAM;
```

  DevPort**:** The virtual com port number
  DevType**:** The module type (1**:** I-7565-H1; 2**:** I-7565-H2)
  CAN1_Baud**:** CAN1 port baud rate
  (0 **:** Disable CAN1 port Others**:** Enable CAN1 port)
  CAN2_Baud**:** CAN2 port baud rate
  (0 **:** Disable CAN2 port Others**:** Enable CAN2 port)
- **Return:**
  Return 0 means success, others means failure (Please refer to "Section 2.1 Error Code").

2.6 Software Buffer FUNCTIONS

2.6.1 VCI_Get_RxMsgCnt

- **Description :**
  To get the count of these received CAN messages saved in software buffer that are not received by users' program in the assigned CAN port.
- **Syntax :**
  int VCI_Get_RxMsgCnt(BYTE CAN_No, DWORD* RxMsgCnt)
- **Parameter :**
  CAN_No**:** The assigned CAN port number.
  RxMsgCnt**:** The pointer is used to receive the CAN message count saved in software buffer.

- **Return:**

  Return 0 means success, others means failure (Please refer to "Section 2.1 Error Code").

2.6.2 VCI_Get_RxMsgBufIsFull

- **Description :**
  To get the software buffer state whether it is full or not in the assigned CAN port. If the software buffer is full, it means that some CAN messages are lost.
- **Syntax :**

  int VCI_Get_RxMsgBufIsFull(BYTE CAN_No, BYTE* Flag)
- **Parameter :**
  CAN_No**:** The assigned CAN port number.
  Flag**:** The pointer is used to receive the state of software buffer. If the value is zero, the software buffer is not full. If not, it means that the software buffer is full.
- **Return:**

  Return 0 means success, others means failure (Please refer to "Section 2.1 Error Code").

2.6.3 VCI_Clr_RxMsgBuf

- **Description :**
  To clear the software buffer in the assigned CAN port.
- **Syntax :**

  int VCI_Clr_RxMsgBuf(BYTE CAN_No)
- **Parameter :**

  CAN_No**:** The assigned CAN port number.
- **Return:**

  Return 0 means success, others means failure (Please refer to "Section 2.1 Error Code").

2.7 Other FUNCTIONS

2.7.1 VCI_Get_DllVer

- **Description :**

To get the version of VCI_CAN library.

- **Syntax :**

  DWORD VCI_Get_DllVer(void)

- **Parameter :**

  None.

- **Return:**

  Return the VCI_CAN library version. Hi-byte is the major version and lo-byte is the minor version.

  .

3. i-7565-H1/H2 Demo Programs For Linux

All function of demo programs will not work normally if i-7565-H1/H2 linux driver would not be installed correctly.

Table 3.1

Directory Path	File Name	Description
Include	i7565H1H2.h	The header of i-7565-H1/H2 library.
lib	libI7565H1H2.a	The i-7565-H1/H2 library for LinPAC-8x81(or x86 Linux PC).
	libI7565H1H2_arm.a	The i-7565-H1/H2 library for LinPAC-8x41.
doc	i7565-Linux-Manual.pdf	The linux manual for i-7565-H1/H2.
examples	I7565H1H2.c	The i-7565-H1/H2 demo source code.
	i7565H1H2	A execution for LinPAC-8x81(or x86 Linux PC).
	i7565H1H2_arm	A execution for LinPAC-8x41.

3.1 Demo code “i7565H1H2”

This i-7565-H1/H2 demo program had provided below capability. Please follow below step to operate i-7565-H1 module in linux system.

Step 1: To choose the device file of i-7565 module (user could refer to page 3 to check the device file in linux system). Please refer to figure 3-1.

Step 2: To choose the device type of i-7565 module. Please refer to figure 3-1.

Step 3: To choose the baud rate of i-7565 CAN port. Please refer to figure 3-1

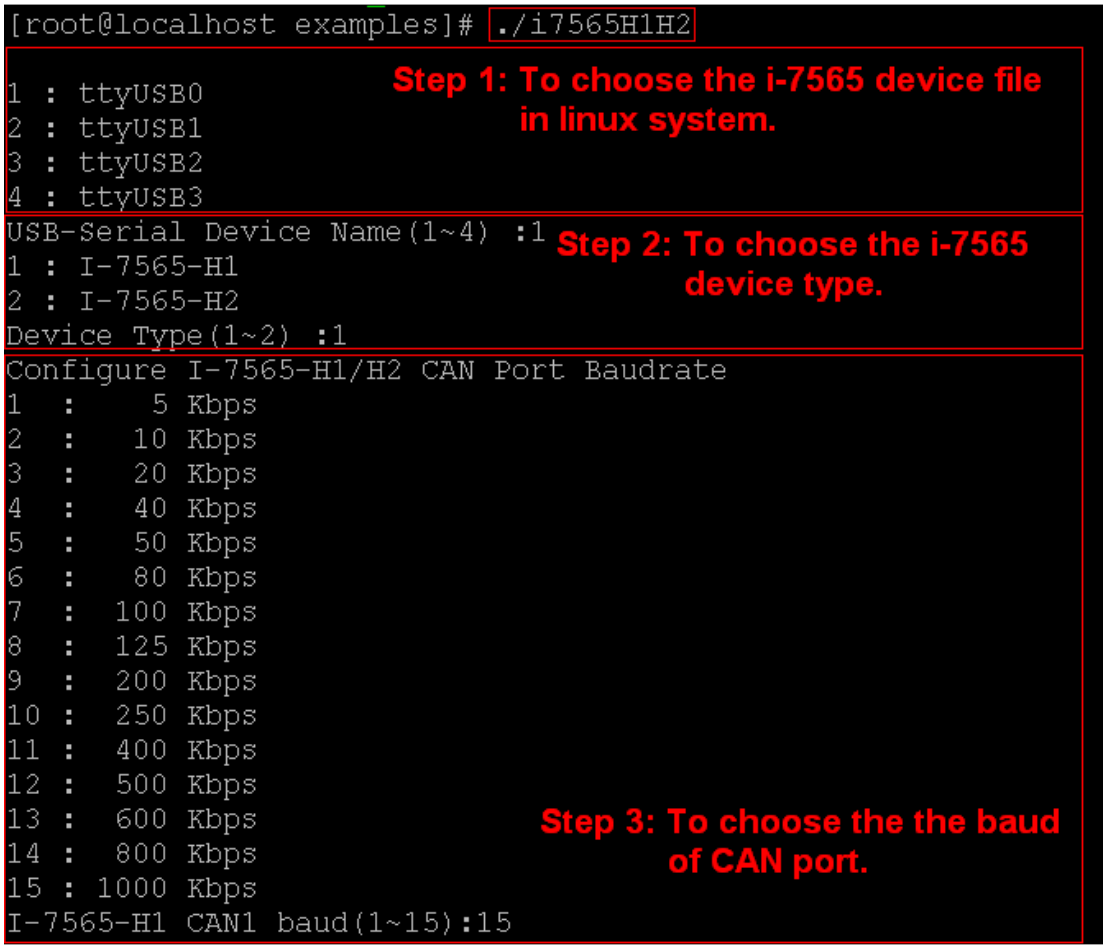


Figure 3-1

Step 4: After user initial i-7565-H1 module well, the demo would show all capability. Please refer to figure 3-2.

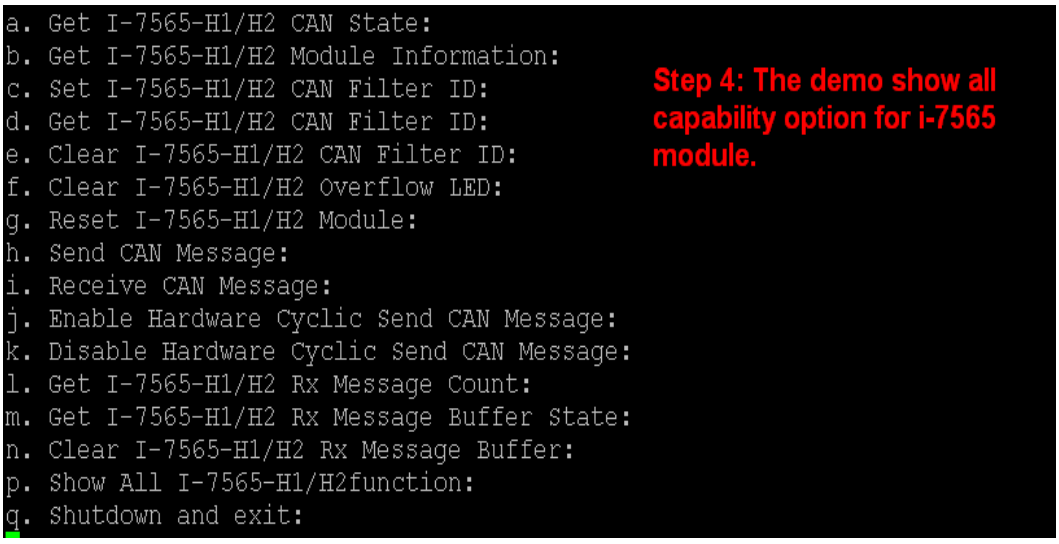


Figure 3-2

Step 5: To choose option ‘a’, ‘b’ to get the information of i-7565-H1 module. Please refer to figure 3-3.

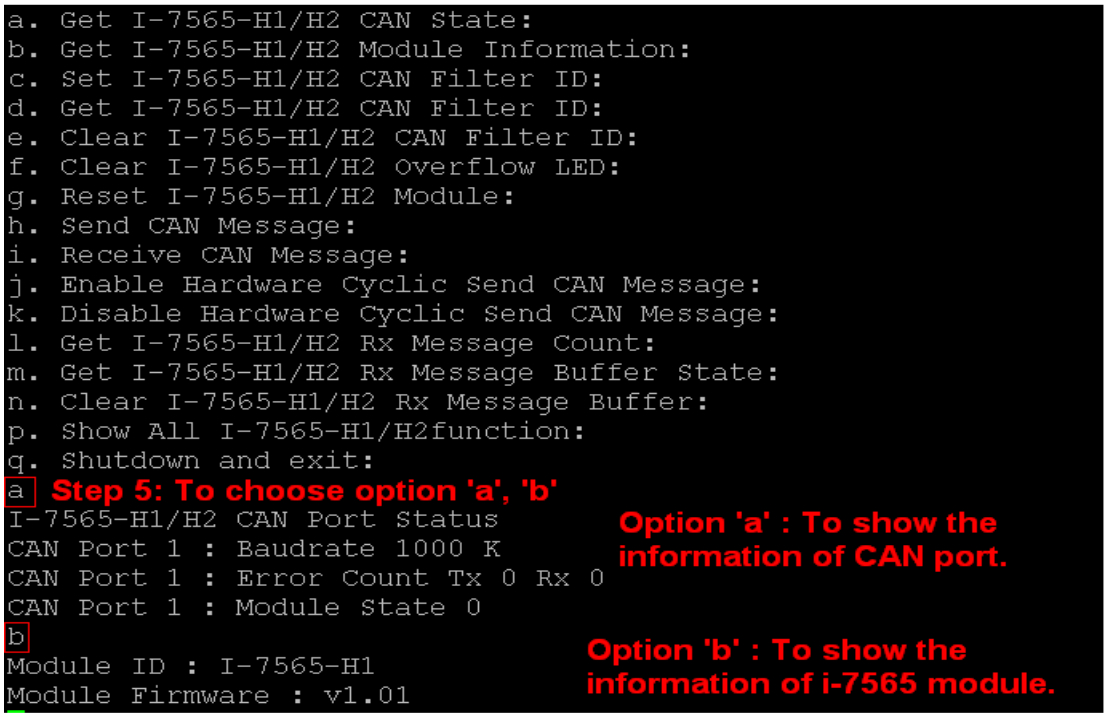


Figure 3-3

Step 6: To choose option ‘c’, ‘d’ and ‘e’ to configure the filter ID of i-7565-H1 module. Please refer to figure 3-4, 3-5.

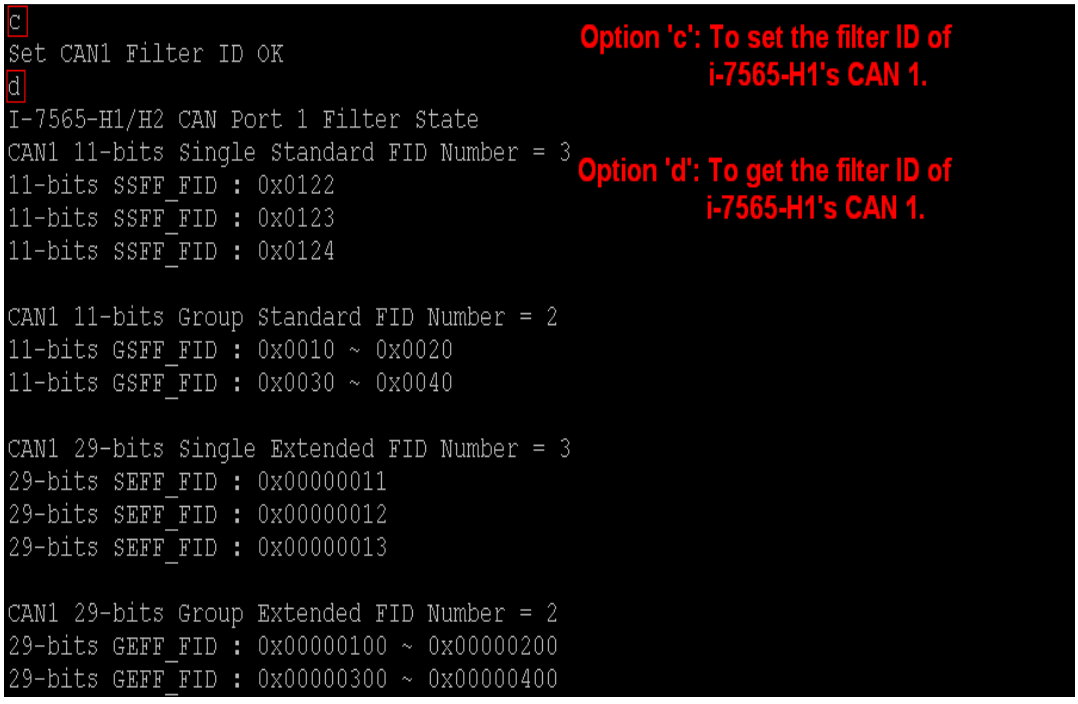


Figure 3-4

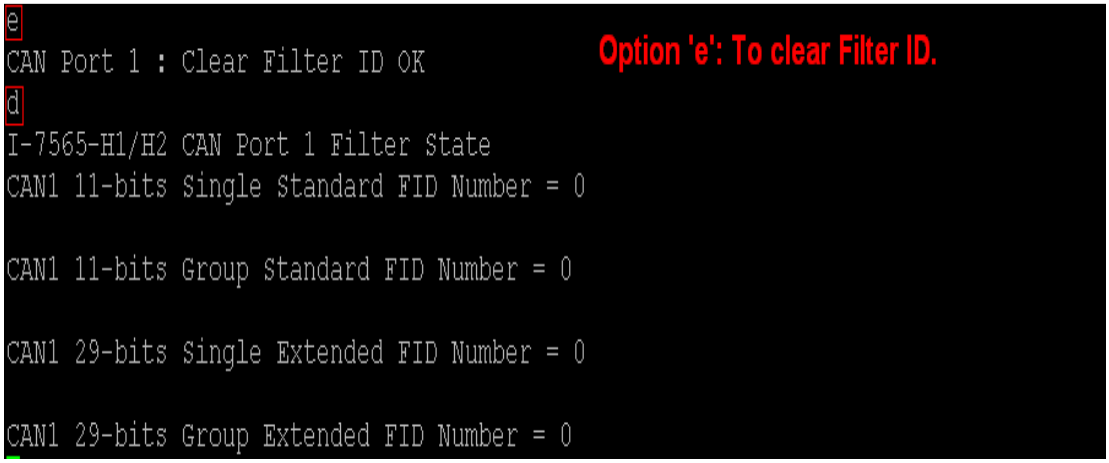
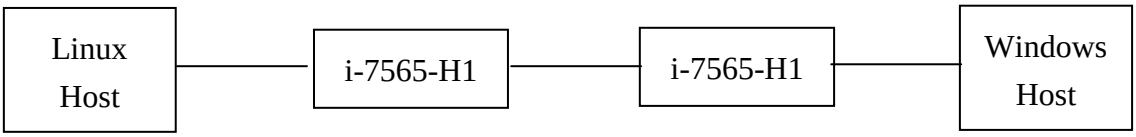


Figure 3-5

Step 7: Before choosing the option, user should build the test environment first. Please refer to below test environment.



To choose option ‘h’ to send a CAN message to another i-7565-H1 module that installing in Windows system. Please refer to figure 3-6, 3-7(Windows host use i-7565 utility to get CAN message).

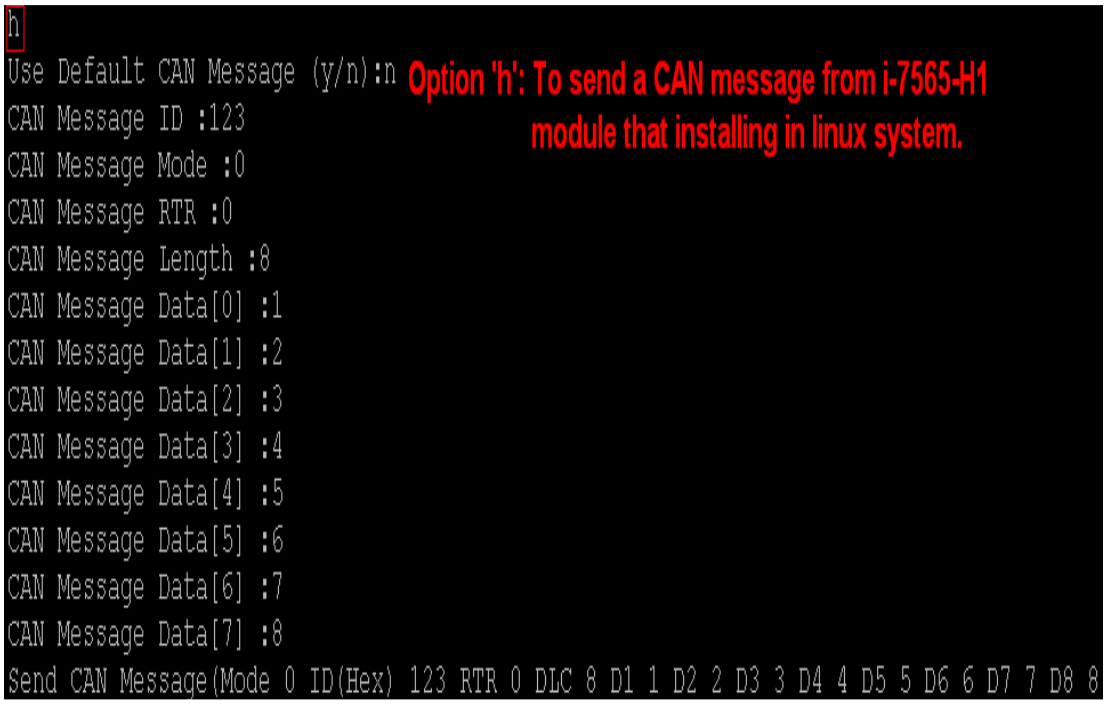


Figure 3-6

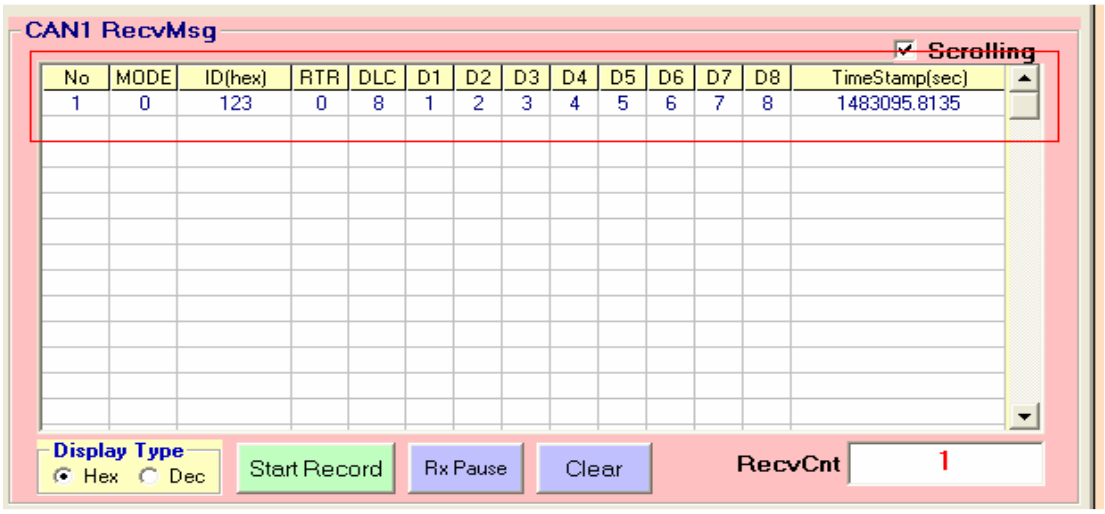
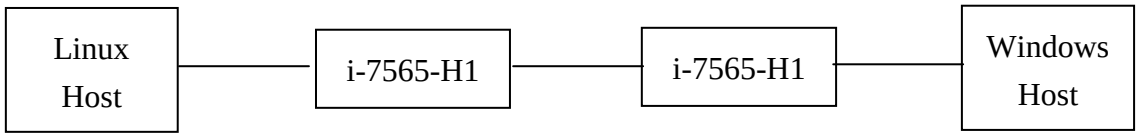


Figure 3-7

Step 8: Before choosing the option, user should build the test environment first. Please refer to below test environment.



To choose option ‘i’ to receive a CAN message from i-7565-H1 module that installing in Windows system. Please refer to figure 3-8(Windows host use i-7565 utility to send CAN message “Mode 0 ID(Hex) 110 RTR 0 DLC 8 D1 1 D2 2 D3 3 D4 4 D5 5 D6 6 D7 7 D8 8”), 3-9

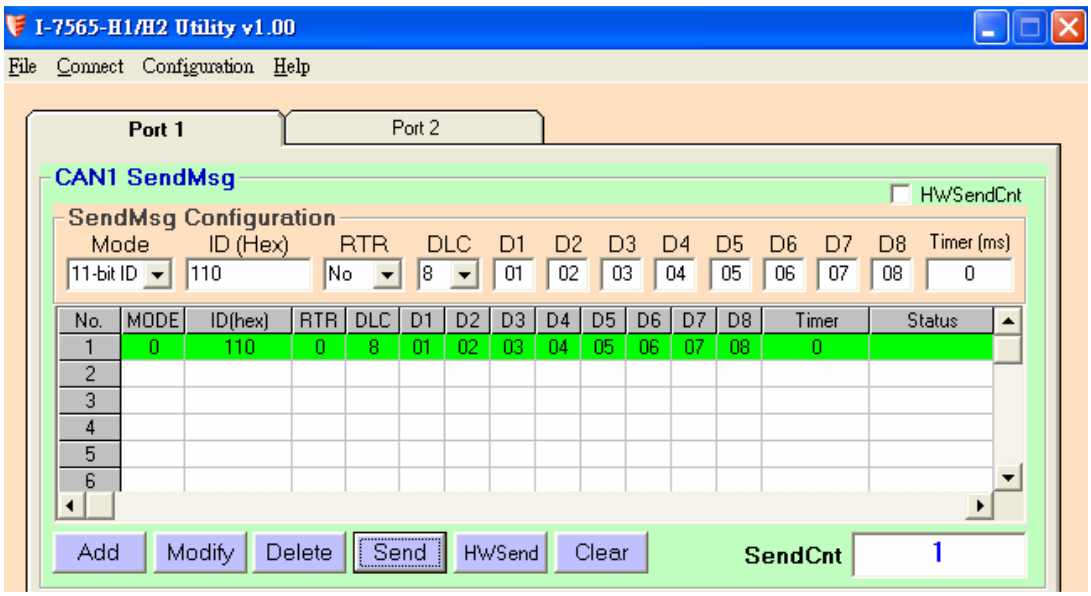
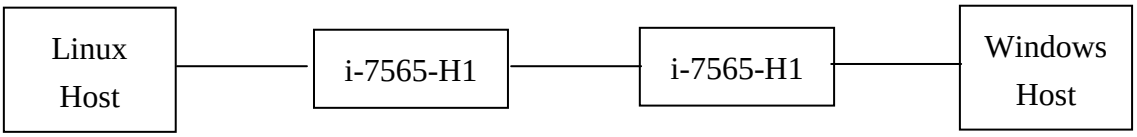


Figure 3-8

Figure 3-9

Step 9: Before choosing the option, user should build the test environment first. Please refer to below test environment.



To choose option ‘j’ to enable hardware timer to send CAN message. Please refer to figure 3-10, 3-11.

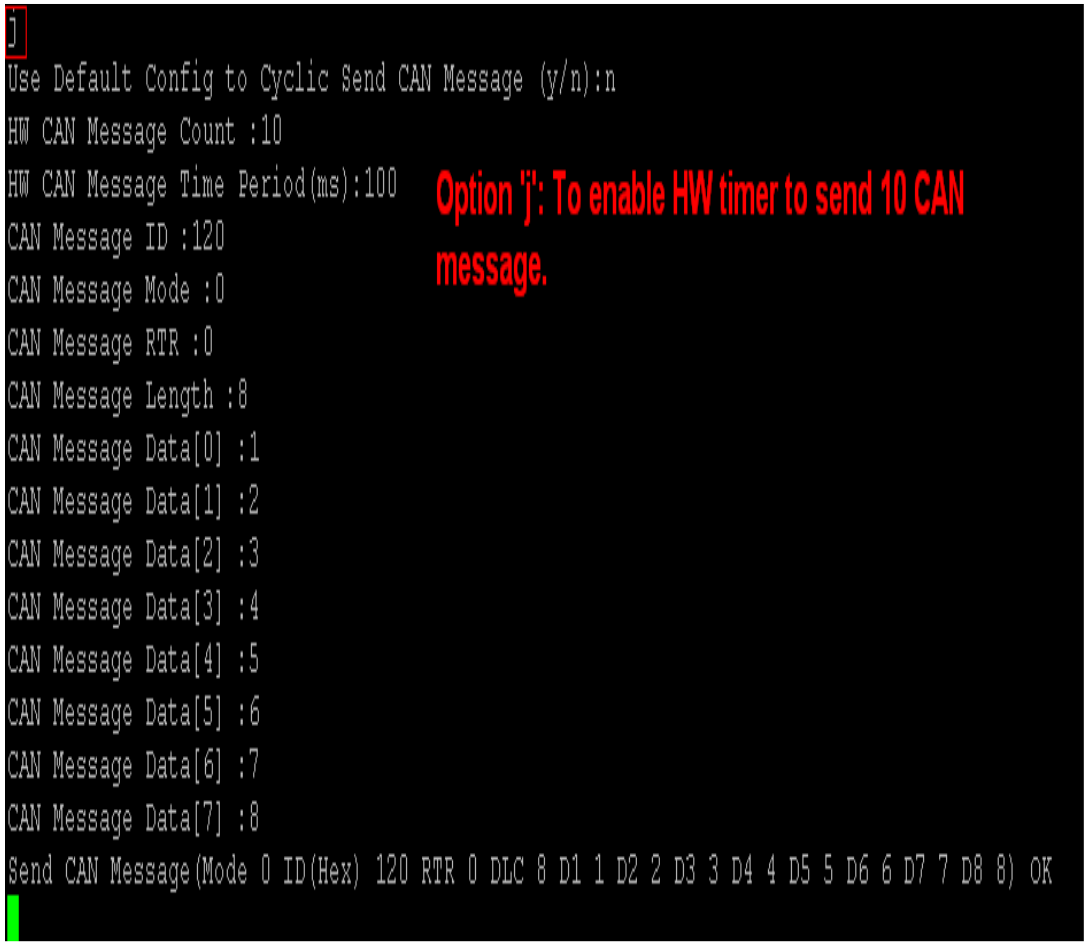


Figure 3-10

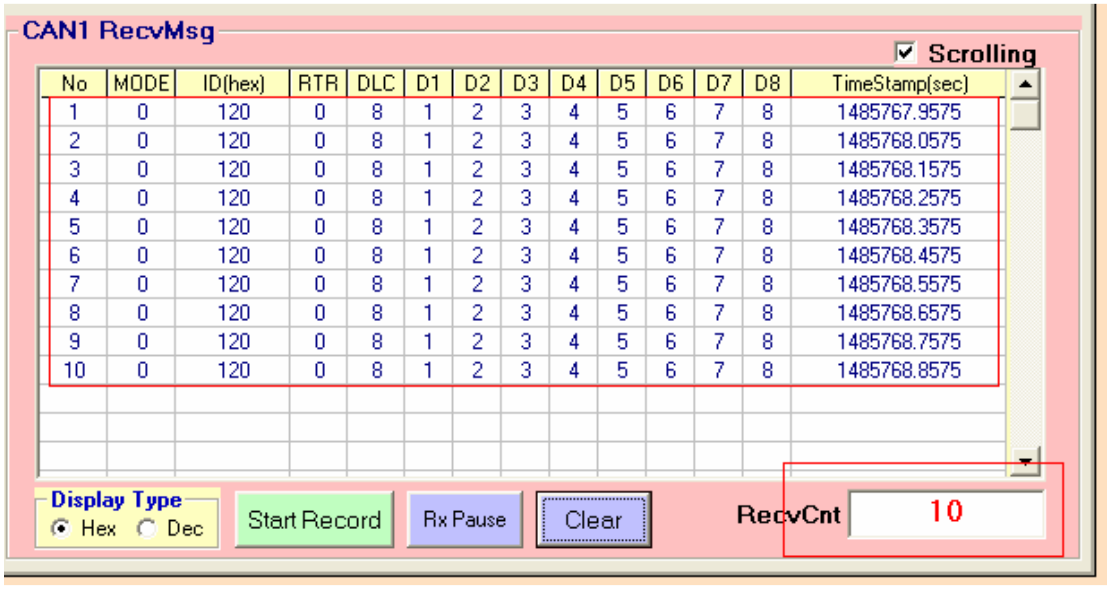


Figure 3-11

Step 10: To choose option ‘q’ to finish demo.