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(arm)

- 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
  ```

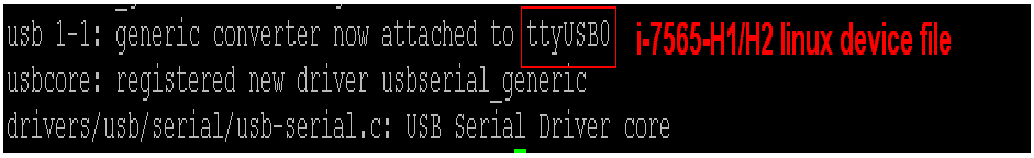
(2) LinPAC-8x81/9x71/9x81 series or Linux PC(x86)

Execute script “I7565H1H2_install” to install I7565-H1/H2 driver.

```
#./I7565H1H2_install
```

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

```
#dmesg
```



```
usb 1-1: generic converter now attached to ttyUSB0
usbcore: registered new driver usbserial_generic
drivers/usb/serial/usb-serial.c: USB Serial Driver core
```

i-7565-H1/H2 linux device file

Figure 1-1

1.2 Linux Driver Uninstalling Procedure

(1) LinPAC-8x41(arm)

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

#rmmod usbserial

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

Execute script “I7565H1H2_install” to install I7565-H1/H2 driver.

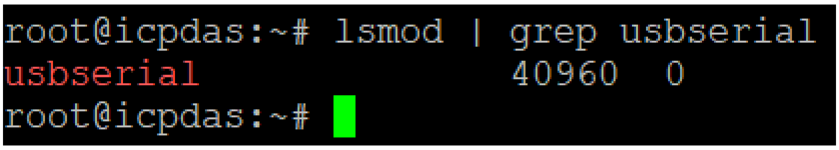
./ I7565H1H2_install remove

1.3 Linux Driver Install At Boot Time

If you want install driver at boot time. You can execute script “I7565H1H2_install”

./ I7565H1H2_install bootinstall

Type command “lsmod | grep usbserial” to check I-7565-H1/H2 driver exists or not when you reboot OS. (Please refer to Figure 1-2)



```
root@icpdas:~# lsmod | grep usbserial
usbserial              40960  0
root@icpdas:~# 
```

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

The static library is the collection of function calls of 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”(32 bit OS) or “libI7565H1H2_64.a”(64 bit OS) 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.

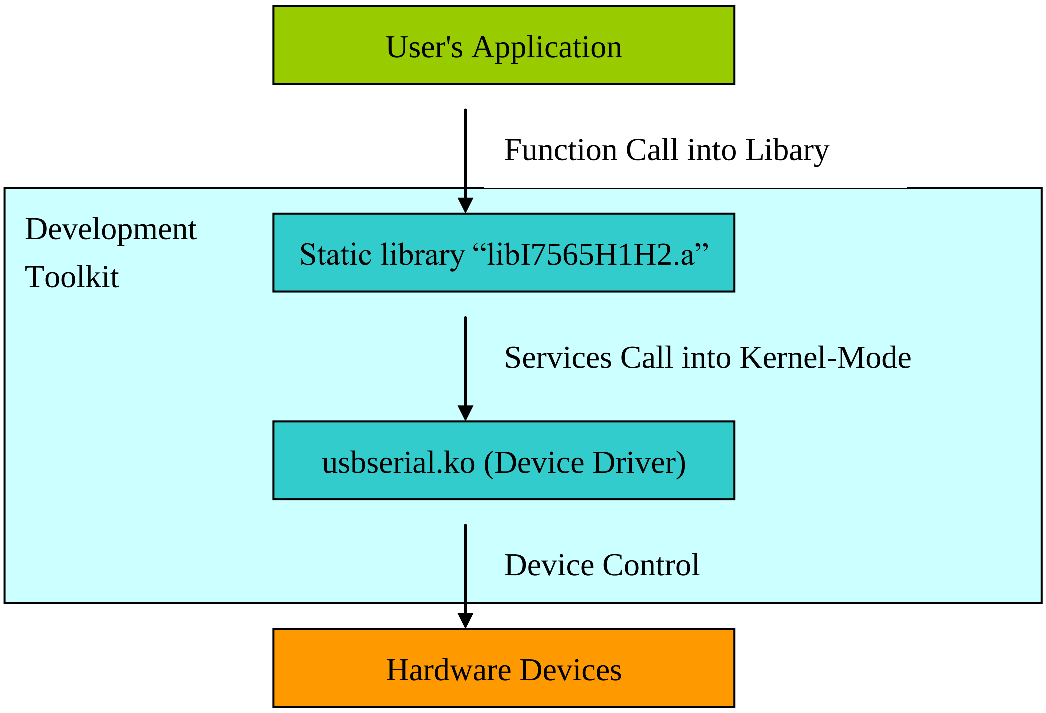

User's Application
Function Call into Libary
Development Toolkit
Static library “libI7565H1H2.a”
Services Call into Kernel-Mode
usbserial.ko (Device Driver)
Device Control
Hardware Devices

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	Getting the modules name error
2	DEV_ModNotExist_Err	The module doesn't exist in this port
3	DEV_PortNotOpen_Err	The port doesn't open
4	DEV_PortClose_Err	Closing i-7565-H1/H2 error
5	DEV_Reset_Err	Resetting i-7565-H1/H2 error
6	CAN_ConfigureFail_Err	CAN hardware configure fail
7	CAN_Hardware_Err	CAN hardware Init fail
8	CAN_PortNo_Err	The CAN port of Device over range
9	CAN_FIDLength_Err	The CAN Filter-ID number exceed max number
10	CAN_DevDisconnect_Err	The connection of device is broken
11	CAN_TimeOut_Err	The configure command is timeout
12	CAN_ConFigureCmd_Err	The configure command doesn't support
13	CAN_ConFigureBusy_Err	The configure command is busy
14	CAN_RxBufEmpty	The CAN receive buffer is empty
15	CAN_TxBufFull	The CAN send buffer is full
16	CAN_EnableHWCyclicTxMsg_Err	To enable hardware cyclic fail
17	CreateRxThread_Err	Creating Rx thread error
18	RestartRxThread_Err	Restarting Rx thread error

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
8	VCI_Set_MOD_Ex
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
4	VCI_Get_TxMsgCnt
5	VCI_Clr_TxMsgBuf
6	VCI_Get_TxSentCnt
7	VCI_Clr_TxSentCnt
NO	Other Function
1	VCI_Get_DllVer
NO	Extended Function
1	VCI_OpenCAN_Ex
2	VCI_Get_CANBaud_BitTime

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 PortOpen;
     BYTE DevPort;
     BYTE DevType;
     DWORD CAN1_Baud;
     DWORD CAN2_Baud;
} _VCI_CAN_PARAM, *PVCI_CAN_PARAM;
```

PortOpen: The assigned port status.
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(PVCI_CAN_PARAM pCANPARAM)

- **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 DevPort, BYTE CAN_No, PVCI_CAN_FID pCANFID)
- **Parameter:**
  DevPort**:** The i-7565-H1/H2 device index.
  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 DevPort, BYTE CAN_No, PVCI_CAN_FID pCANFID)
- **Parameter :**
  DevPort**:** The i-7565-H1/H2 device index.
  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 DevPort, BYTE CAN_No)

- **Parameter :**

  DevPort**:** The i-7565-H1/H2 device index.

  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 DevPort, BYTE CAN_No, PVCI_CAN_STATUS pCANStatus)

- **Parameter :**

  DevPort**:** The i-7565-H1/H2 device index.

  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 DevPort, BYTE CAN_No)
- **Parameter :**

  DevPort**:** The i-7565-H1/H2 device index.

  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(BYTE DevPort, PVCI_MOD_INFO pMODInfo)
- **Parameter :**
  DevPort**:** The i-7565-H1/H2 device index.
  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];
    char HW_SN[16];
} _VCI_MODULE_INFO, *PVCI_MOD_INFO;
```

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

HW_SN[16]**:** Return the module hardware serial number string.

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

2.4.7 VCI_Rst_MOD

- **Description :**
  To reset module.
- **Syntax :**
  int VCI_Rst_MOD(PVCI_CAN_PARAM pCANPARAM)
- **Parameter :**
  None.
- **Return:**

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

2.4.8 VCI_Set_MOD_Ex

- **Description :**
  This extended function is used to set the module parameters of new functions.
- **Syntax :**
  int VCI_Set_MOD_Ex (BYTE CfgData[512]);
- **Parameter :**
  CfgData[512]:
    [in] Module setting parameter array.
      [Byte 0] : CAN1 Listen Only Function (0:Disable, 1:Enable)
      [Byte 1] : CAN2 Listen Only Function (0:Disable, 1:Enable)
- **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 DevPort, BYTE CAN_No, PVCI_CAN_MSG pCANMsg)
- **Parameter :**

  DevPort**:** The i-7565-H1/H2 device index.

  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 DevPort, BYTE CAN_No,
  PVCI_CAN_MSG pCANMsg)

- **Parameter :**

  DevPort**:** The i-7565-H1/H2 device index.

  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 DevPort, BYTE CAN_No,
  PVCI_CAN_MSG pCANMsg, DWORD TimePeriod, DWORD
  TransmitTimes)

- **Parameter :**

DevPort**:** The i-7565-H1/H2 device index.

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 PortOpen;
    BYTE DevPort;
    BYTE DevType;
    DWORD CAN1_Baud;
    DWORD CAN2_Baud;
} _VCI_CAN_PARAM, *PVCI_CAN_PARAM;
```

PortOpen: The assigned port status.
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 DevPort, BYTE CAN_No, DWORD* RxMsgCnt)
- **Parameter :**
  DevPort**:** The i-7565-H1/H2 device index.
  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 DevPort, BYTE CAN_No, BYTE* Flag)
- **Parameter :**
  DevPort**:** The i-7565-H1/H2 device index.
  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 DevPort, BYTE CAN_No)
- **Parameter :**
  DevPort**:** The i-7565-H1/H2 device index.
  CAN_No**:** The assigned CAN port number.
- **Return:**
  Return 0 means success, others means failure (Please refer to "Section 2.1 Error Code").

2.6.4 VCI_Get_TxMsgCnt

- **Description :**
  This function is used to get the count of CAN messages that needed to be sent in software buffer of the assigned CAN port.
- **Syntax :**
  int VCI_Get_TxMsgCnt (BYTE DevPort ,BYTE CAN_No, DWORD*

TxMsgCnt);

- **Parameter :**

  DevPort**:** The i-7565-H1/H2 device index.

  CAN_No: [in] The assigned CAN port number.

  TxMsgCnt: [out] The pointer is used to get the CAN message count that needed to be sent in software buffer.

- **Return:**

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

2.6.5 VCI_Clr_TxMsgBuf

- **Description :**
  This function is used to clear the sending software buffer in the assigned CAN port.

- **Syntax :**

  int VCI_Clr_TxMsgBuf (BYTE CAN_No);

- **Parameter :**

  DevPort**:** The i-7565-H1/H2 device index.

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

- **Return:**

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

2.6.6 VCI_Get_TxSentCnt

- **Description :**
  This function is used to get the total CAN message count that had been sent in the assigned CAN port.

- **Syntax :**

  int VCI_Get_TxSentCnt (BYTE DevPort ,BYTE CAN_No, DWORD* TxSentCnt);

- **Parameter :**

  DevPort**:** The i-7565-H1/H2 device index.

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

  TxSentCnt: [out] The pointer is used to get the total CAN message count that have been sent.

- **Return:**

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

2.6.7 VCI_Clr_TxSentCnt

- **Description :**
  This function is used to clear the total CAN message count that had been sent in the assigned CAN port.
- **Syntax :**
  int VCI_Clr_TxSentCnt (BYTE DevPort ,BYTE CAN_No);
- **Parameter :**
  DevPort**:** The i-7565-H1/H2 device index.
  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_Library_Version

- **Description :**
  To get the version of VCI_CAN library.
- **Syntax :**
  char * VCI_Get_Library_Version (void)
- **Parameter :**
  None.
- **Return:**
  Return the VCI_CAN library version.

2.8 Extended FUNCTIONS

2.8.1 VCI_OpenCAN_Ex

- **Description :**
  This function is the same with the VCI_OpenCAN( ) but it adds the function able to adjust the sample point (Tseg2 value) of bit-timing of

CAN baud. It is useful when CAN bus communication failed in the occasion filled with electromagnetic interference (such as: motor starts causing interference), then users can use the bigger Tseg2 value in the same baudrate for CAN bus communication.

- **Syntax :**

  `int VCI_OpenCAN_Ex ( PVCI_CAN_PARAM_EX pCANPARAMEx );`

- **Parameter :**

  pCANPARAM:

  [in] A structure pointer of _VCI_CAN_PARAM_EX is used to set the CAN port communication parameters and Tseg2 value shown as below.

```
typedef struct _VCI_CAN_PARAM_EX{
  _VCI_CAN_PARAM vc;
  BYTE CAN1_T2Val;
  BYTE CAN2_T2Val;
  BYTE Reserved[32];
  } _VCI_CAN_PARAM_EX, *PVCI_CAN_PARAM_EX;
```

  Vc : The struct of _VCI_CAN_PARAM.
  CAN1_T2Val : The Tesg2 value of CAN1
  CAN2_T2Val : The Tesg2 value of CAN2
  Reserved[32] : Reserved

- **Return:**

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

2.8.2 VCI_Get_CANBaud_BitTime

- **Description :**

  This function is used to get the Tseg1, Tseg2 and SJW values of the CAN baud bit-timing parameters of the assigned CAN port.

  When the CAN communication failed, it can be used to check if the bit-timing parameters of the other CAN devices are the same with I-7565-H1/H2 module.

- **Syntax :**

```
int VCI_Get_CANBaud_BitTime (
        BYTE CAN_No,
        BYTE* T1Val,
        BYTE* T2Val,
        BYTE* SJWVal
        );
```

- **Parameter :**

  CAN_No:

  [in] The assigned CAN port number.

  T1Val:

  [out] The pointer is used to receive the Tseg1 value of the assigned CAN port.

  T2Val:

  [out] The pointer is used to receive the Tseg2 value of the assigned CAN port.

  SJWVal:

  [out] The pointer is used to receive the SJW value of the assigned CAN port.

- **Return:**

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

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	Static library for LinPAC-8x81(or x86 32bits Linux PC).
	libI7565H1H2.so.1.0	Shared library for LinPAC-8x81(or x86 32bits Linux PC).
	libI7565H1H2_64.a	Static library for LinPAC-8x81(or x86 64bits Linux PC).
	libI7565H1H2_64.so.1.0	Shared library for LinPAC-8x81(or x86 64bits 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	An execution for LinPAC-8x81(or x86 Linux PC).
	i7565H1H2_.so	An execution for LinPAC-8x81(or x86 Linux PC).
	i7565H1H2_arm	An execution for LinPAC-8x41.

3.1 Demo name “i7565H1H2_a” & “i7565H1H2_so”

The i7565H1H2_a is link to libI7565H1H2.a, and i7565H1H2_so is link to libI7565H1H2.so.1.0

3.2 Demo code “i7565H1H2_a”

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

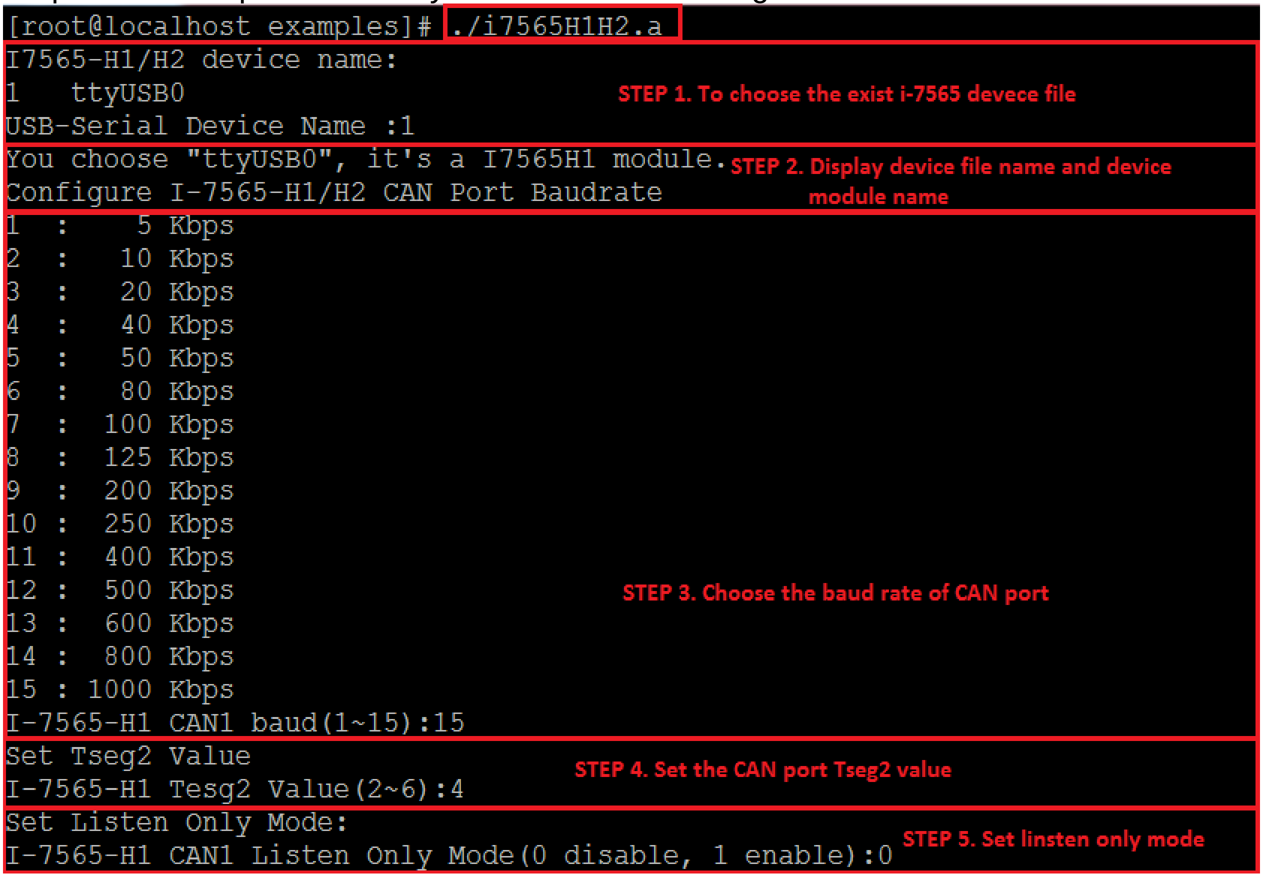
Step 1: To choose exist device file of i-7565 module. Please refer to figure 3-1.

Step 2: Display device file name and device module name you choose. Please refer to figure 3-1.

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

Step 4: Set the CAN port Tesg2 value. Please refer to figure 3-1

Step 5.Set CAN port listen only mode. Please refer figure 3-1



```
[root@localhost examples]# ./i7565H1H2.a
I7565-H1/H2 device name:
1   ttyUSB0
USB-Serial Device Name :1
You choose "ttyUSB0", it's a I7565H1 module.
Configure I-7565-H1/H2 CAN Port Baudrate
1  :    5 Kbps
2  :   10 Kbps
3  :   20 Kbps
4  :   40 Kbps
5  :   50 Kbps
6  :   80 Kbps
7  :  100 Kbps
8  :  125 Kbps
9  :  200 Kbps
10 :  250 Kbps
11 :  400 Kbps
12 :  500 Kbps
13 :  600 Kbps
14 :  800 Kbps
15 : 1000 Kbps
I-7565-H1 CAN1 baud(1~15):15
Set Tseg2 Value
I-7565-H1 Tesg2 Value(2~6):4
Set Listen Only Mode:
I-7565-H1 CAN1 Listen Only Mode(0 disable, 1 enable):0
```

STEP 1. To choose the exist i-7565 devece file

STEP 2. Display device file name and device module name

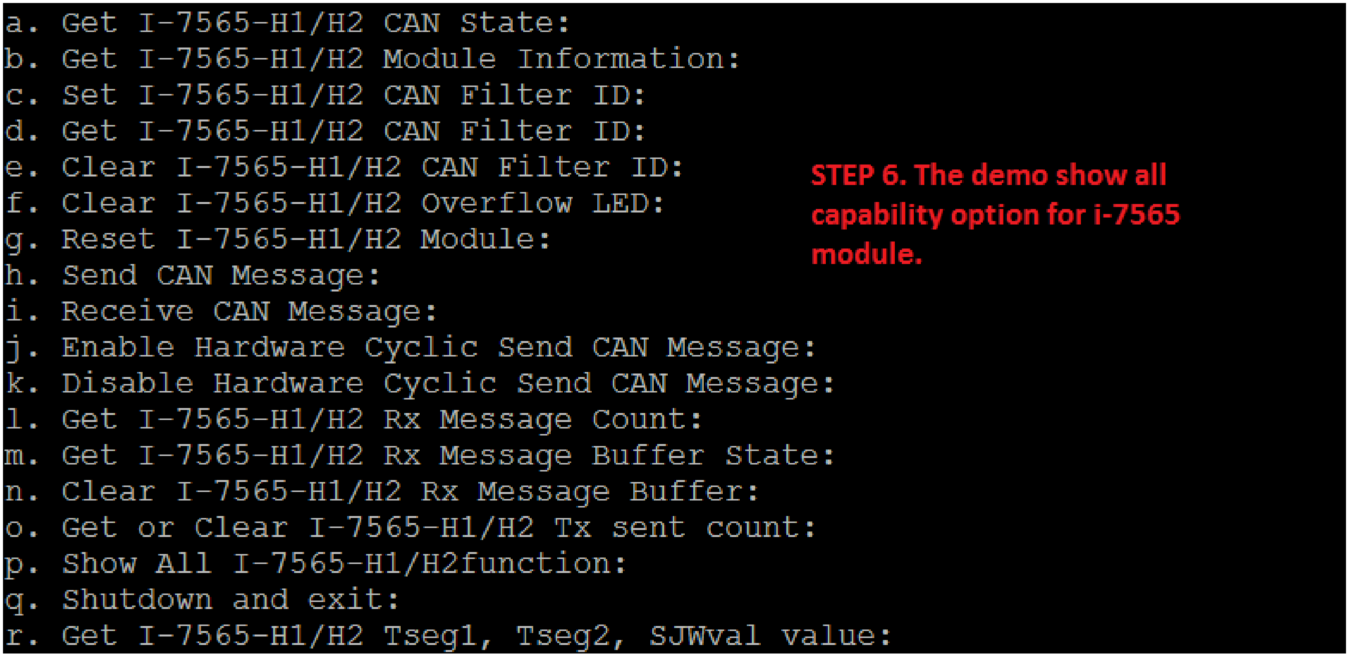
STEP 3. Choose the baud rate of CAN port

STEP 4. Set the CAN port Tseg2 value

STEP 5. Set linsten only mode

Figure 3-1

Step 6: After user initial i-7565-H1 module well, the demo would show all capability. Please refer to figure 3-2.

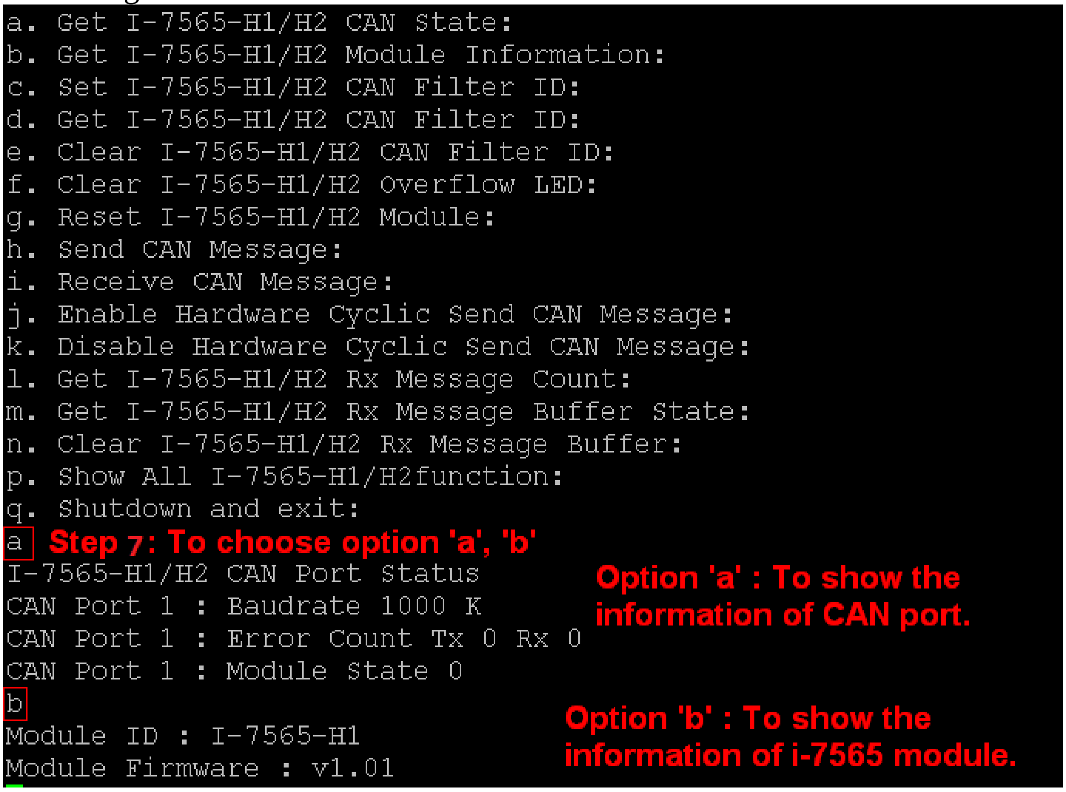


```
a. Get I-7565-H1/H2 CAN State:
b. Get I-7565-H1/H2 Module Information:
c. Set I-7565-H1/H2 CAN Filter ID:
d. Get I-7565-H1/H2 CAN Filter ID:
e. Clear I-7565-H1/H2 CAN Filter ID:
f. Clear I-7565-H1/H2 Overflow LED:
g. Reset I-7565-H1/H2 Module:
h. Send CAN Message:
i. Receive CAN Message:
j. Enable Hardware Cyclic Send CAN Message:
k. Disable Hardware Cyclic Send CAN Message:
l. Get I-7565-H1/H2 Rx Message Count:
m. Get I-7565-H1/H2 Rx Message Buffer State:
n. Clear I-7565-H1/H2 Rx Message Buffer:
o. Get or Clear I-7565-H1/H2 Tx sent count:
p. Show All I-7565-H1/H2function:
q. Shutdown and exit:
r. Get I-7565-H1/H2 Tseg1, Tseg2, SJWval value:
```

STEP 6. The demo show all capability option for i-7565 module.

Figure 3-2

Step 7: To choose option ‘a’, ‘b’ to get the information of i-7565-H1 module. Please refer to figure 3-3.



```
a. Get I-7565-H1/H2 CAN State:
b. Get I-7565-H1/H2 Module Information:
c. Set I-7565-H1/H2 CAN Filter ID:
d. Get I-7565-H1/H2 CAN Filter ID:
e. Clear I-7565-H1/H2 CAN Filter ID:
f. Clear I-7565-H1/H2 Overflow LED:
g. Reset I-7565-H1/H2 Module:
h. Send CAN Message:
i. Receive CAN Message:
j. Enable Hardware Cyclic Send CAN Message:
k. Disable Hardware Cyclic Send CAN Message:
l. Get I-7565-H1/H2 Rx Message Count:
m. Get I-7565-H1/H2 Rx Message Buffer State:
n. Clear I-7565-H1/H2 Rx Message Buffer:
p. Show All I-7565-H1/H2function:
q. Shutdown and exit:
a
I-7565-H1/H2 CAN Port Status
CAN Port 1 : Baudrate 1000 K
CAN Port 1 : Error Count Tx 0 Rx 0
CAN Port 1 : Module State 0
b
Module ID : I-7565-H1
Module Firmware : v1.01
```

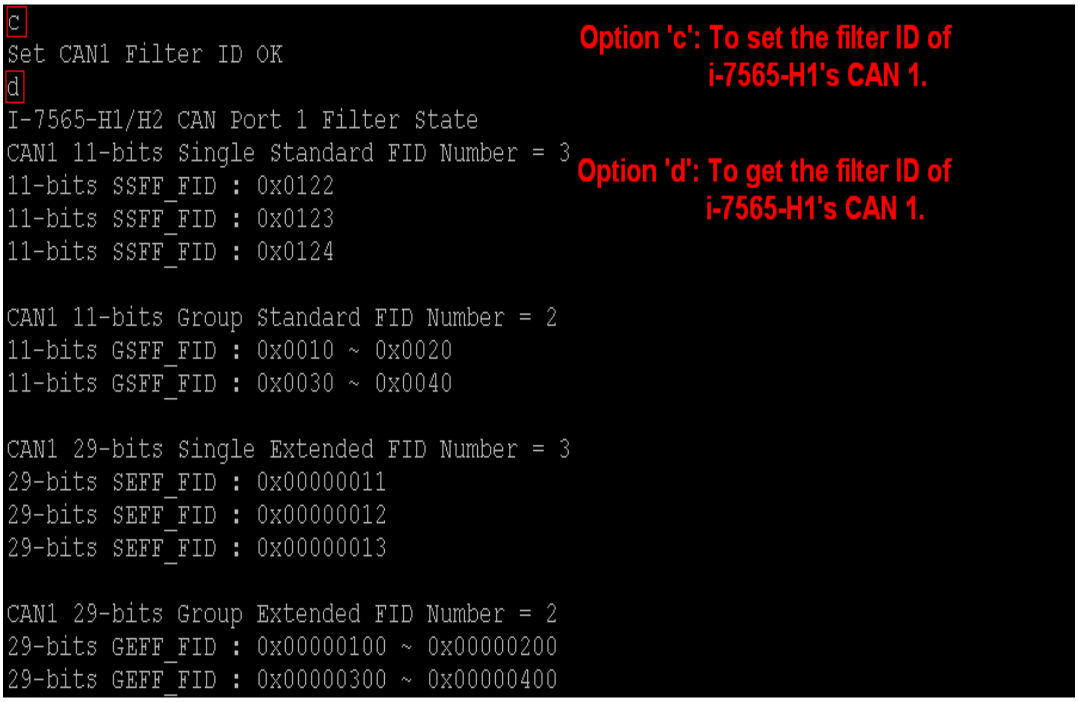
Step 7: To choose option 'a', 'b'

Option 'a' : To show the information of CAN port.

Option 'b' : To show the information of i-7565 module.

Figure 3-3

Step 8: 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.

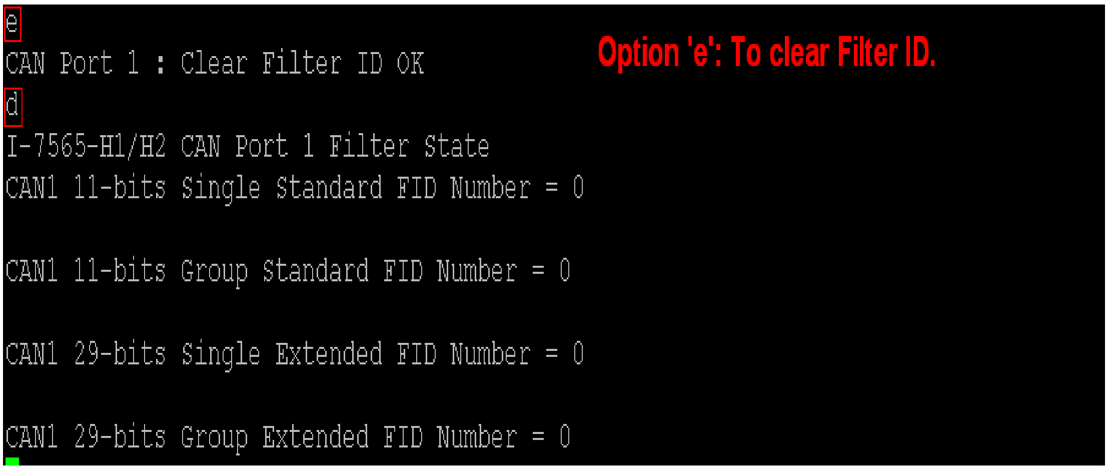


```
c
Set CAN1 Filter ID OK
d
I-7565-H1/H2 CAN Port 1 Filter State
CAN1 11-bits Single Standard FID Number = 3
11-bits SSFF_FID : 0x0122
11-bits SSFF_FID : 0x0123
11-bits SSFF_FID : 0x0124

CAN1 11-bits Group Standard FID Number = 2
11-bits GSFF_FID : 0x0010 ~ 0x0020
11-bits GSFF_FID : 0x0030 ~ 0x0040

CAN1 29-bits Single Extended FID Number = 3
29-bits SEFF_FID : 0x00000011
29-bits SEFF_FID : 0x00000012
29-bits SEFF_FID : 0x00000013

CAN1 29-bits Group Extended FID Number = 2
29-bits GEFF_FID : 0x00000100 ~ 0x00000200
29-bits GEFF_FID : 0x00000300 ~ 0x00000400
```

Option 'c': To set the filter ID of i-7565-H1's CAN 1.

Option 'd': To get the filter ID of i-7565-H1's CAN 1.

Figure 3-4

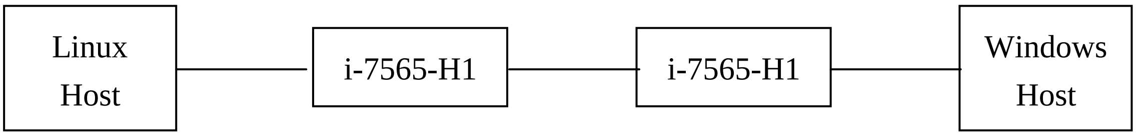


```
e
CAN Port 1 : Clear Filter ID OK
d
I-7565-H1/H2 CAN Port 1 Filter State
CAN1 11-bits Single Standard FID Number = 0

CAN1 11-bits Group Standard FID Number = 0

CAN1 29-bits Single Extended FID Number = 0

CAN1 29-bits Group Extended FID Number = 0
```

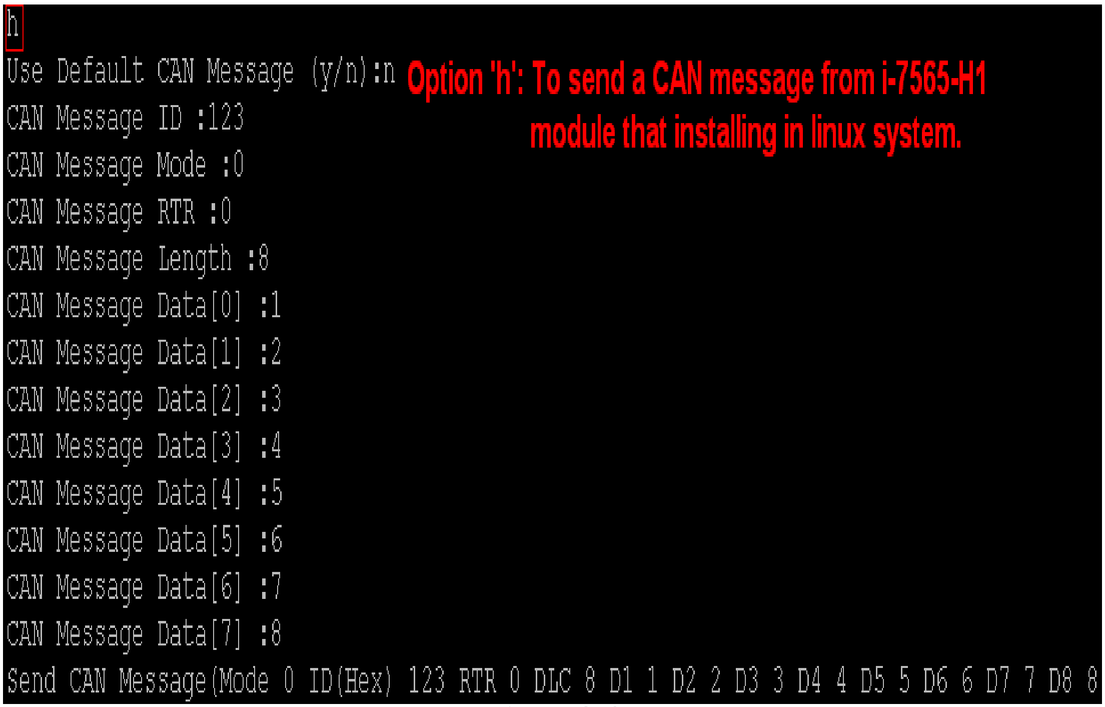
Option 'e': To clear Filter ID.

Figure 3-5

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



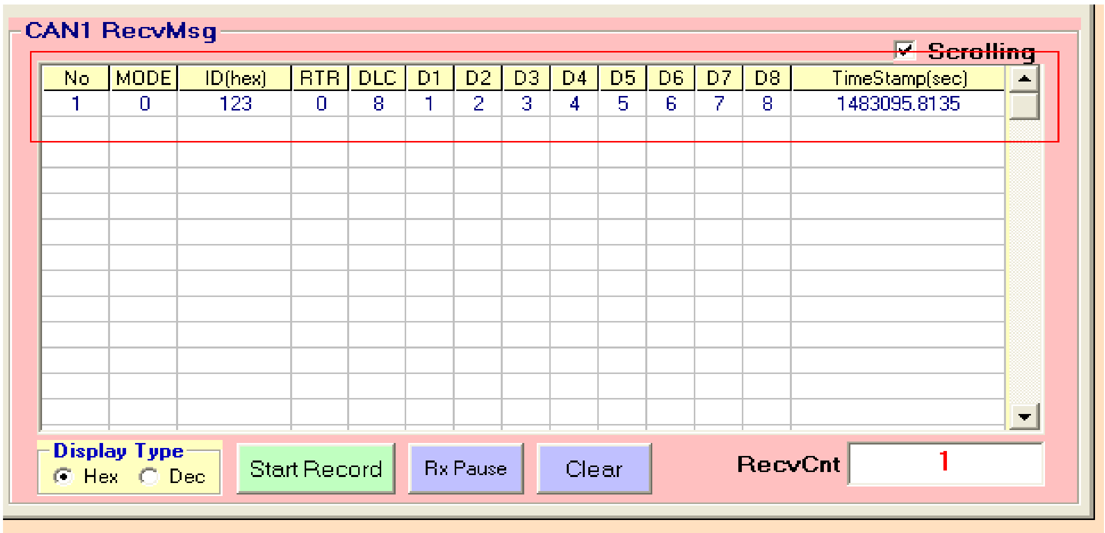
To choose option ‘h’ to send a CAN message to another i-7565-H1 module that installing in Windows system. Please refer to figure 3-6, 3-7(Windows host use i-7565 utility to get CAN message).



```
h
Use Default CAN Message (y/n):n
CAN Message ID :123
CAN Message Mode :0
CAN Message RTR :0
CAN Message Length :8
CAN Message Data[0] :1
CAN Message Data[1] :2
CAN Message Data[2] :3
CAN Message Data[3] :4
CAN Message Data[4] :5
CAN Message Data[5] :6
CAN Message Data[6] :7
CAN Message Data[7] :8
Send CAN Message(Mode 0 ID(Hex) 123 RTR 0 DLC 8 D1 1 D2 2 D3 3 D4 4 D5 5 D6 6 D7 7 D8 8
```

Option 'h': To send a CAN message from i-7565-H1 module that installing in linux system.

Figure 3-6



CAN1 RecvMsg

☑ Scrolling

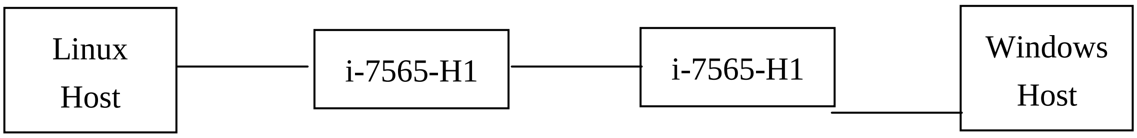
No	MODE	ID(hex)	RTR	DLC	D1	D2	D3	D4	D5	D6	D7	D8	TimeStamp(sec)
1	0	123	0	8	1	2	3	4	5	6	7	8	1483095.8135

Display Type: ● Hex ○ Dec

RecvCnt: 1

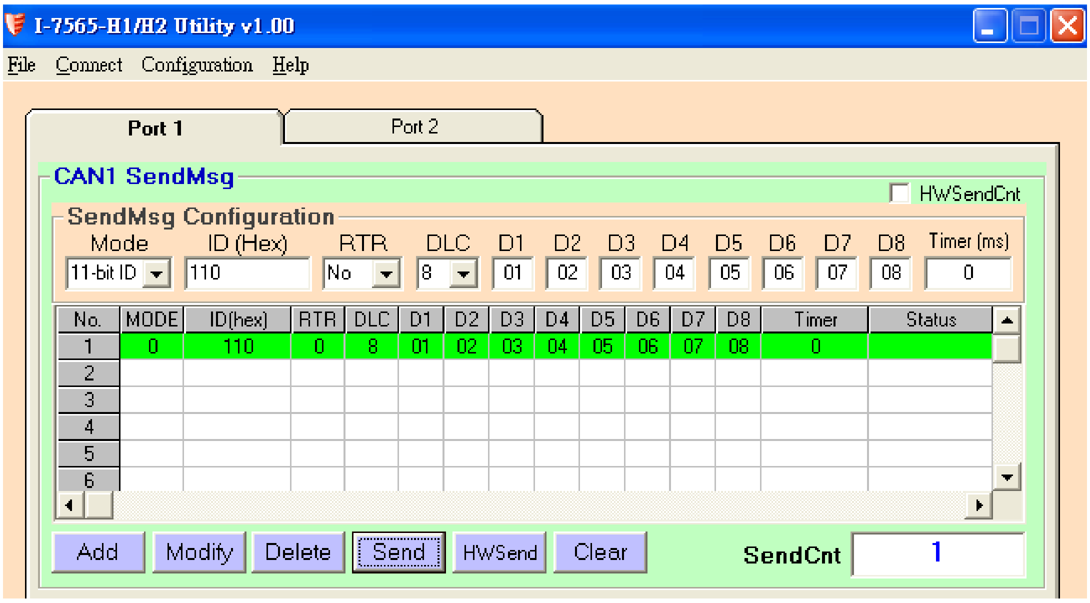
Figure 3-7

Step 10: 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”), figure 3-9.

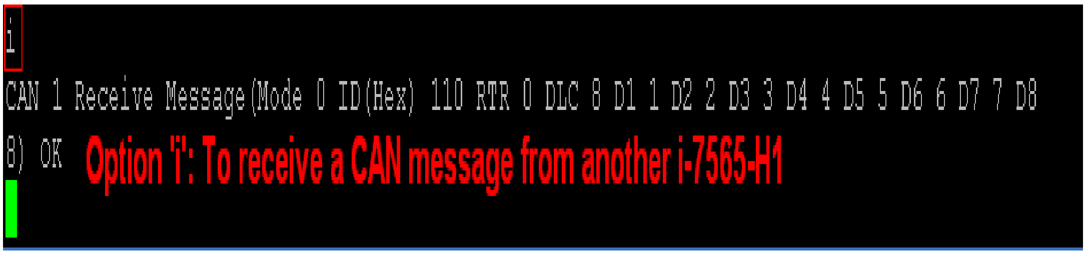


Mode	ID (Hex)	RTR	DLC	D1	D2	D3	D4	D5	D6	D7	D8	Timer (ms)
11-bit ID	110	No	8	01	02	03	04	05	06	07	08	0

No.	MODE	ID(hex)	RTR	DLC	D1	D2	D3	D4	D5	D6	D7	D8	Timer	Status
1	0	110	0	8	01	02	03	04	05	06	07	08	0	
2														
3														
4														
5														
6														

SendCnt 1

Figure 3-8

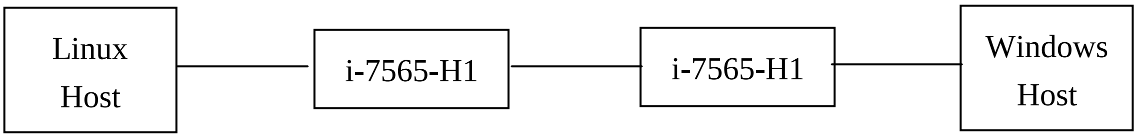


```
i
CAN 1 Receive Message(Mode 0 ID(Hex) 110 RTR 0 DLC 8 D1 1 D2 2 D3 3 D4 4 D5 5 D6 6 D7 7 D8
8) OK
```

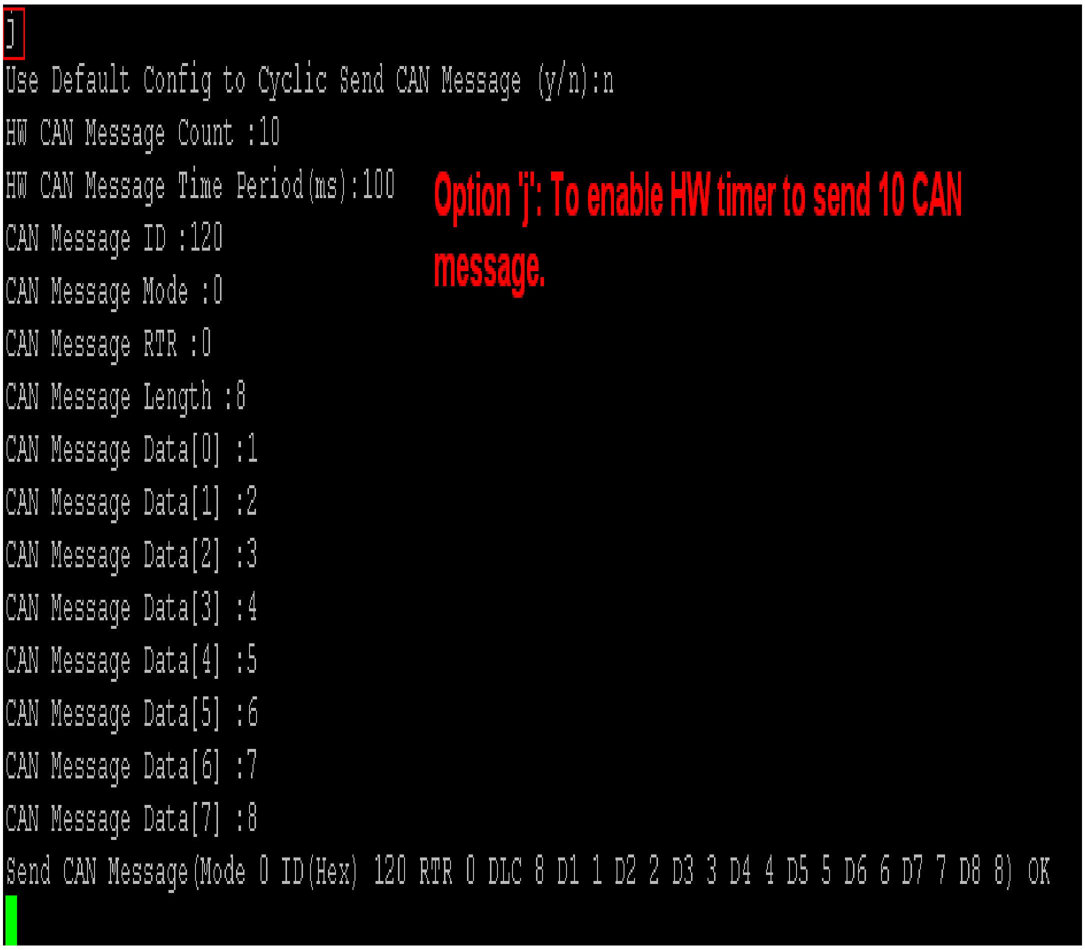
Option 'i': To receive a CAN message from another i-7565-H1

Figure 3-9

Step 11: Before choosing the option, user should build the test environment first. Please refer to below test environment.



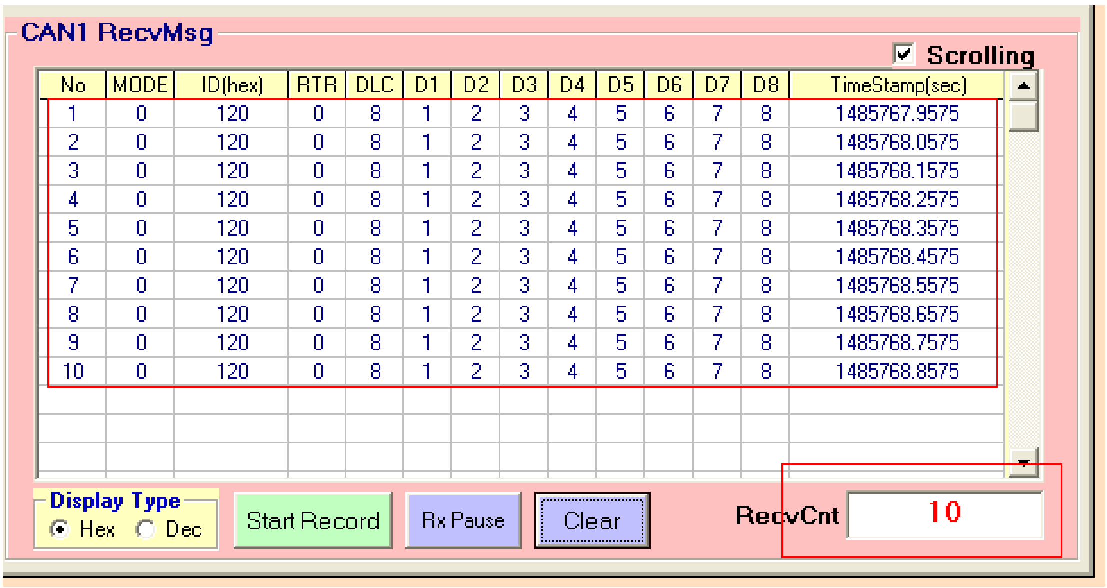
To choose option ‘j’ to enable hardware timer to send CAN message. Please refer to figure 3-10, 3-11.



```
j
Use Default Config to Cyclic Send CAN Message (y/n):n
HW CAN Message Count :10
HW CAN Message Time Period(ms):100
CAN Message ID :120
CAN Message Mode :0
CAN Message RTR :0
CAN Message Length :8
CAN Message Data[0] :1
CAN Message Data[1] :2
CAN Message Data[2] :3
CAN Message Data[3] :4
CAN Message Data[4] :5
CAN Message Data[5] :6
CAN Message Data[6] :7
CAN Message Data[7] :8
Send CAN Message(Mode 0 ID(Hex) 120 RTR 0 DLC 8 D1 1 D2 2 D3 3 D4 4 D5 5 D6 6 D7 7 D8 8) OK
```

Option 'j': To enable HW timer to send 10 CAN message.

Figure 3-10



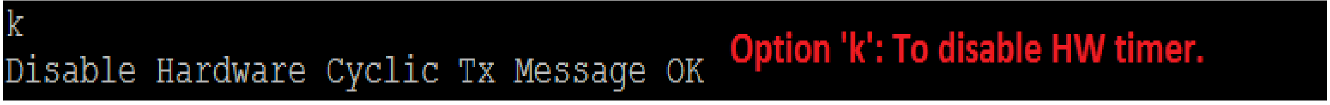
No	MODE	ID(hex)	RTR	DLC	D1	D2	D3	D4	D5	D6	D7	D8	TimeStamp(sec)
1	0	120	0	8	1	2	3	4	5	6	7	8	1485767.9575
2	0	120	0	8	1	2	3	4	5	6	7	8	1485768.0575
3	0	120	0	8	1	2	3	4	5	6	7	8	1485768.1575
4	0	120	0	8	1	2	3	4	5	6	7	8	1485768.2575
5	0	120	0	8	1	2	3	4	5	6	7	8	1485768.3575
6	0	120	0	8	1	2	3	4	5	6	7	8	1485768.4575
7	0	120	0	8	1	2	3	4	5	6	7	8	1485768.5575
8	0	120	0	8	1	2	3	4	5	6	7	8	1485768.6575
9	0	120	0	8	1	2	3	4	5	6	7	8	1485768.7575
10	0	120	0	8	1	2	3	4	5	6	7	8	1485768.8575

Display Type ● Hex ○ Dec

RecvCnt 10

Figure 3-11

Step 12: To choose option ‘k’ to disable hardware timer to send CAN message. Please refer to figure 3-12

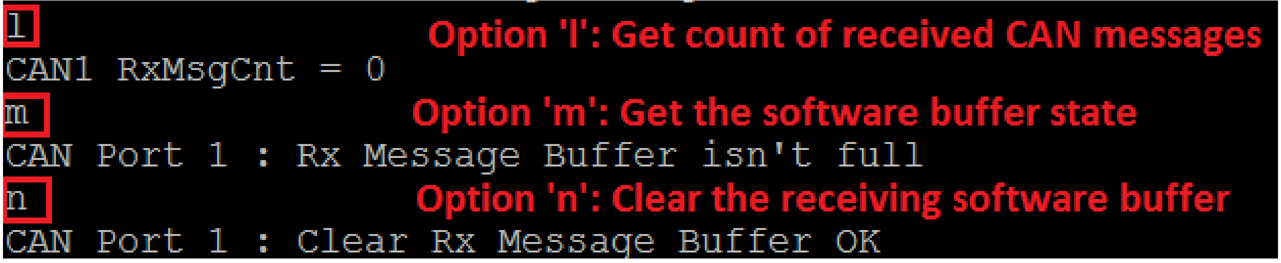


```
k
Disable Hardware Cyclic Tx Message OK
```

Option 'k': To disable HW timer.

Figure 3-12

Step 13: To choose option ‘l’,’m’,’n’ to get received software buffer status. Please refer to figure 3-13



```
l
CAN1 RxMsgCnt = 0
m
CAN Port 1 : Rx Message Buffer isn't full
n
CAN Port 1 : Clear Rx Message Buffer OK
```

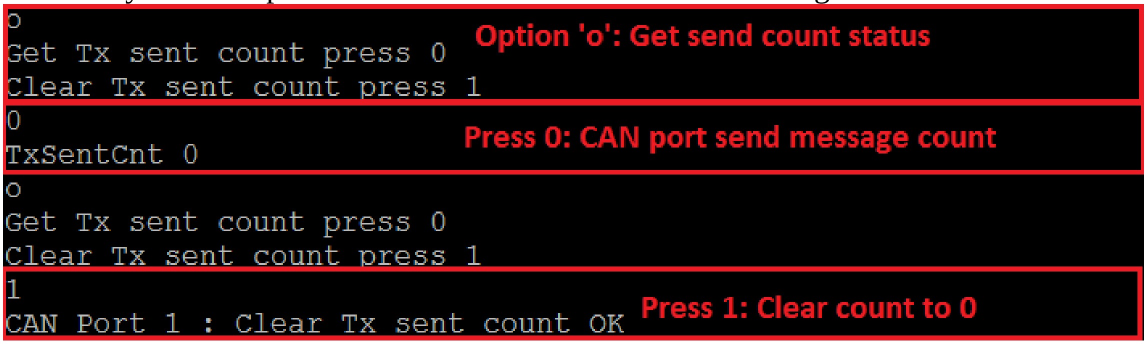
Option 'l': Get count of received CAN messages

Option 'm': Get the software buffer state

Option 'n': Clear the receiving software buffer

Figure 3-13

Step 14: To choose option ‘o’ to get send software buffer status. This option has two option to choose, choose 0 can get your CAN port send software buffer count, choose 1 can clear your CAN port send software buffer. Please refer to figure 3-14



```
o
Get Tx sent count press 0
Clear Tx sent count press 1
0
TxSentCnt 0
o
Get Tx sent count press 0
Clear Tx sent count press 1
1
CAN Port 1 : Clear Tx sent count OK
```

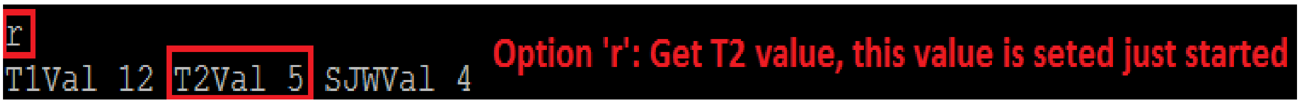
Option 'o': Get send count status

Press 0: CAN port send message count

Press 1: Clear count to 0

Figure 3-14

Step 15: To choose option ‘r’ to get assigned CAN port T2 value. Please refer to figure 3-15



```
r
T1Val 12 T2Val 5 SJWVal 4
```

Option 'r': Get T2 value, this value is seted just started

Step 16: To choose option ‘q’ to finish demo.

Appendix

1. **I7565-H1/H2 coexists with cdc_acm driver**

In default setting, when you install I7565-H1/H2 driver (use script "I7565H1H2_install"), we'll blacklist cdc_acm driver. This driver will probe I7565-H1/H2 device and init it.

If you have other devices need cdc_acm driver, you can follow steps below.

1.1. Execute script "I7565H1H2_install" to install I7565-H1/H2 driver and keep cdc_acm driver

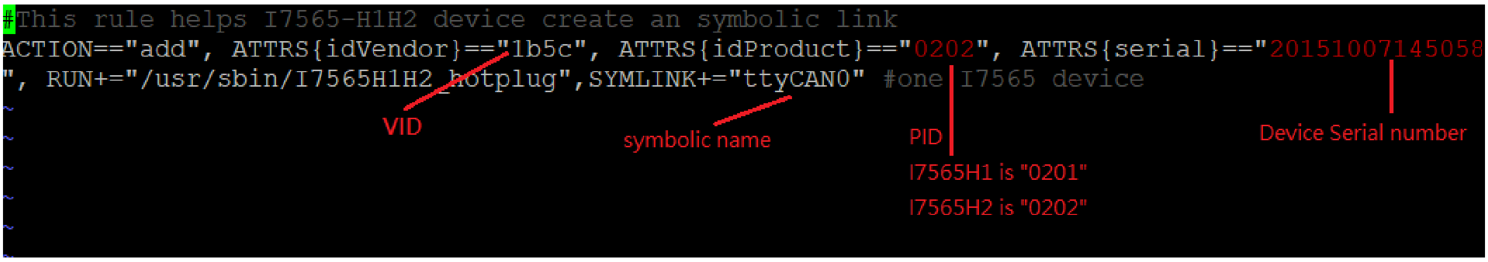
./ I7565H1H2_install cdc_acm-unblacklist

If cdc_acm driver still not install, you can execute script "I7565H1H2_install cdc_acm" to remove cdc_acm driver from blacklist and test.

1.2. Modify "99-I7565H2.rules" and "I7565H1H2_hotplug" example.

The screenshots below provides only one I-7565H1/H2 device to use. If you have two or more devices, please follow example and add device by yourself.

Modify file "99-I7565H2.rules"



```
#This rule helps I7565-H1H2 device create an symbolic link
ACTION=="add", ATTRS{idVendor}=="1b5c", ATTRS{idProduct}=="0202", ATTRS{serial}=="20151007145058
", RUN+="/usr/sbin/I7565H1H2_hotplug",SYMLINK+="ttyCAN0" #one I7565 device
~
~
~
~
~
```

In the VID section, always 1b5c

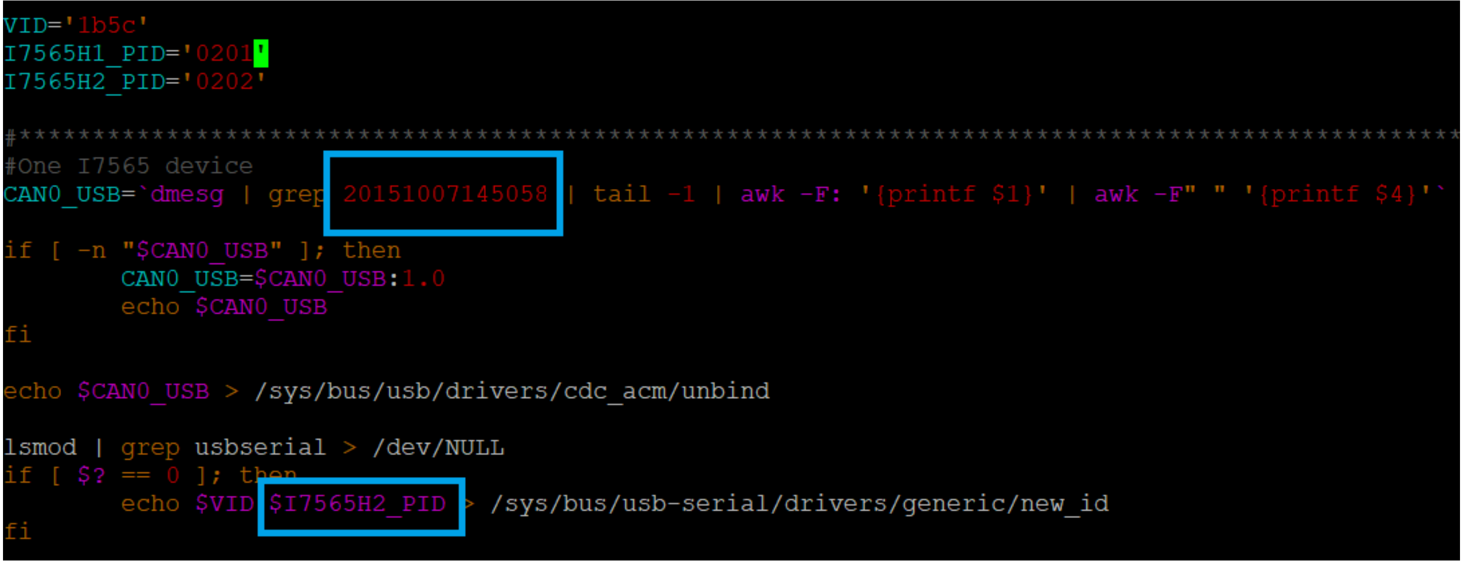
In the PID section, I7565H1 is 0201, I7565H2 is 0202

In the serial number section, you can use command

"**dmesg | grep "Product: I-7565-H**" -A2"to check your device serial number

In the symbolic name section, it can fixed device name.

Modify file “I7565H1H2_hotplug”



```
VID='1b5c'
I7565H1_PID='0201'
I7565H2_PID='0202'

#**************************************************************************************************
#One I7565 device
CAN0_USB=`dmesg | grep 20151007145058 | tail -1 | awk -F: '{printf $1}' | awk -F" " '{printf $4}'`

if [ -n "$CAN0_USB" ]; then
        CAN0_USB=$CAN0_USB:1.0
        echo $CAN0_USB
fi

echo $CAN0_USB > /sys/bus/usb/drivers/cdc_acm/unbind

lsmod | grep usbserial > /dev/NULL
if [ $? == 0 ]; then
        echo $VID $I7565H2_PID > /sys/bus/usb-serial/drivers/generic/new_id
fi
```

In first blue box, change it to your device serial number

In second blue box, according to your device to set right PID

1.3. Execute script “I7565H1H2_install” to move these two files to right place.

./ I7565H1H2_install udev_setup

1.4. You can test and find I7565-H1/H2 device and cdc_acm driver can coexist.