PCI-D64HU

Linux Software 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. Linux Software Installation

The PCI-D64HU can be used in below Linux distribution.

Linux Distribution	Linux Kernel
Ubuntu 12.10(32bit)	3.5.0-6-generic
openSUSE 12.1(32bit)	3.1.0-1.2-default
openSUSE 12.1(32bit)	3.1.0-1.2-desktop
openSUSE 12.1(32bit)	3.1.10-1.29-desktop

For Linux O.S, the recommended installation and uninstall steps are given in Sec 1.1 ~ 1.2

1.1 Linux Driver Installing Procedure

Step 1: Copy the linux driver “pcid64hu.tgz” from the below web link

http://ftp.icpdas.com/pub/cd/iocard/pci/napdos/pci/pci-d64/dll/driver/

Step 2: Decompress the tarball “pcid64hu.tgz”.

Step 3: Type `cd' to the directory containing the package's source code and type `make' to compile the package.

Step 4: Type `./drv_load' to install the PCI-D64HU Linux driver module. Please refer to the Fig 1-1:

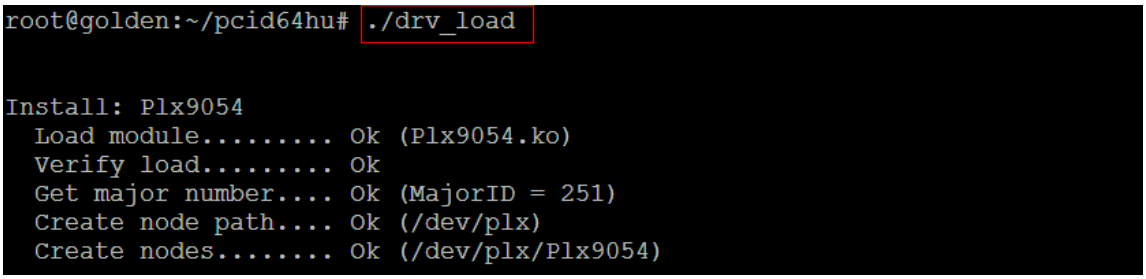

Fig 1-1

1.2 Linux Driver Uninstalling Procedure

Step 1: Type `cd' to the directory containing the package.

Step 2: Type `./drv_unload ' to remove the PCI-D64HU Linux driver module. Please refer to the Fig 1-2:

```
root@golden:~/pcid64hu# ./drv_unload

Remove: Plx9054
  Unload module.................. Ok (Plx9054)
  Clear device nodes.............. Ok (/dev/plx/Plx9054)
  Delete device node path......... Ok (/dev/plx)
```

Fig 1-2

2. Static Libary Function Description

The static libary is the collection of function calls of the PCI-D64HU card for Linux OS. The application structure is presented as following figure. The user application program developed by C language can call library “libd64hu.a” in user mode.

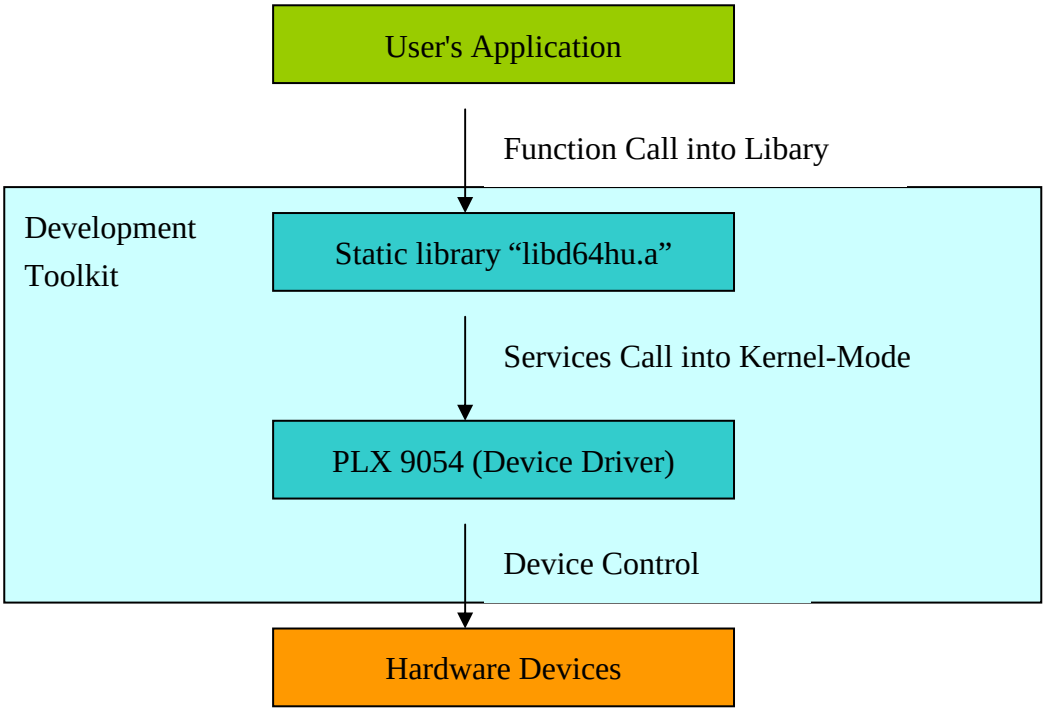

Figure 2.1

2.1 Table of ErrorCode

Error Code	Error ID
0	SUCCESS_NO_ERROR
-100	ERROR_ROUTINE_FAIL_BASE
-101	ERROR_GET_CARD_ID
-102	ERROR_DEVICE_OPEN
-103	ERROR_DEVICE_CLOSE
-104	ERROR_DOUBLE_BUFFER_MODE
-105	ERROR_DI_CONFIG
-106	ERROR_DI_ASYNC_CLEAR
-107	ERROR_DI_ASYNC_CHECK
-108	ERROR_DI_ASYNC_HALF_READY
-109	ERROR_DI_PIO_READ
-110	ERROR_DO_CONFIG
-111	ERROR_DO_ASYNC_CLEAR
-112	ERROR_DO_ASYNC_CHECK
-113	ERROR_DO_PIO_READ
-114	ERROR_DO_PIO_WRITE
-115	ERROR_DO_PIO_LINE_WRITE
-116	ERROR_CONTINUE_DI_START
-117	ERROR_CONTINUE_DO_START
-118	ERROR_FIFO_STATUS_GET
-119	ERROR_DO_EXT_TRIGLINE_WRITE
-120	ERROR_DO_HALF_READY
-121	ERROR_DIGITAL_FILTER_SET
-122	ERROR_OREQ_WIDTH_SET
-200	ERROR_INVALID_PARAMETER_BASE
-201	ERROR_INVALID_CARD_ID
-202	ERROR_INVALID_SCANNED_INDEX
-203	ERROR_CARD_ID_DUPLICATED
-204	ERROR_INVILID_DOUBLE_BUFFER_MODE
-205	ERROR_INVALID_PORT_LINE
-206	ERROR_INVALID_TRIGGER_SOURCE
-207	ERROR_INVALID_TRIGGER_ENABLE

-208	ERROR_INVALID_TRIGGER_POLARITY
-209	ERROR_INVALID_IREQ_POLARITY
-210	ERROR_INVALID_OTRIG_LEVEL
-211	ERROR_INVALID_OREG_ENABLE
-212	ERROR_INVALID_DOUBLE_BUFFER_MODE
-213	ERROR_INVALID_SYNCH_OP_MODE
-214	ERROR_SAMPLE_COUNT_IS_ODD
-215	ERROR_INVALID_DO_ITERATIONS
-216	ERROR_INVALID_EVENT_ACTION
-217	ERROR_INVALID_CALLBACK_ADDRESS
-218	ERROR_INVALID_BUFFER_ADDRESS
-219	ERROR_INVALID_PATTERN_SIZE
-220	ERROR_INVALID_FILTER_SETTING
-221	ERROR_INVALID_OREG_WIDTH
-300	ERROR_RUNTIME_BASE
-316	ERROR_NO_CARD_FOUND
-317	ERROR_MEMORY_MAP
-318	ERROR_MEMORY_UNMAP
-319	ERROR_ACCESS_VIOLATION_DATA_COPY
-320	ERROR_VARIABLE_PITCH_SET
-321	ERROR_DI_EVENT_ATTACH
-322	ERROR_DI_EVENT_DETACH
-323	ERROR_DO_EVENT_ATTACH
-324	ERROR_DO_EVENT_DETACH
-325	ERROR_EVENT_CREATE_FAILED
-326	ERROR_OVERLAP_EVENT_CREATE
-361	ERROR_IOCTL_FAILED
-362	ERROR_UNDEFINED_EXCEPTION
-500	ERROR_DIO_RUNTIME_BASE
-501	ERROR_DEVICE_POWER_DOWN
-502	ERROR_INVALID_MAPPED_ADDRESS
-503	ERROR_BUFFER_NOT_ENOUGH
-504	ERROR_DMA_NOT_AVAILABLE
-505	ERROR_DMA_NOT_COMPLETE
-506	ERROR_DMA_NOT_STARTED
-507	ERROR_DMA_NOT_PAUSED
-508	ERROR_DMA_IS_PAUSED

-509	ERROR_DMA_IN_PROGRESS
-510	ERROR_INVALID_SAMPLE_RATE
-511	ERROR_SAMPLE_COUNT_TOO_LARGE
-512	ERROR_SYNCH_OP_WITH_DOUBLE_BUFFER_MODE
-513	ERROR_SYNCH_OP_WITH_ASYNC_CHECK
-514	ERROR_UNSUPPORTED_SETTING
-515	ERROR_PATTERN_OUT_WITH_DOUBLE_BUFFER_MODE
-516	ERROR_PATTERN_OUT_IN_PROGRESS

Table 2.1

2.2 System Functions

2.2.1 d64h_scan

- **Description:**

  This function scans all active PCI-D64HU cards in your system. The pCardNum saves the numbers of active PCI-D64HU cards. The optional user-provided Array, pAvailCards, indicates the presence of active PCI-D64HU card. (1: present, 0: absent).

- **Syntax:**

  short d64h_scan(short* pCardNum, BYTE* pAvailCards = NULL)

- **Parameter:**

  pCardNum: The pointer to the memory that stores the numbers of active PCI-D64HU cards. pAvailCard: The address of user-provided BYTE-Array. Based on the Card ID, each element indicates the presence of active PCI-D64HU card. The user must prepare one BYTE-Array with D64H_MaxCards elements. For instance, there are three active PCI-D64HU cards with Card ID 3, 5 and 7. The content of pAvailCard Array will be { 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0 }

- **Return:**

  Please refer to "Section 2.1 Error Code".

2.2.2 d64h_get_cardinfo

- **Description:**

  This function returns the Card ID based on the scanned-index. This routine will get the Card ID configured with on-board Dip-Switch.

- **Syntax:**

  short d64h_get_cardinfo(int ScannedIndex, BYTE* pCardID)

- **Parameter:**

  ScannedIndex: The index that the active PCI-D64HU is scanned. This index begins from 0, and is less than the active PCI-D64HU cards.
  pCardID: The pointer to the memory that stores the specific Card ID.

- **Return:**

  Please refer to "Section 2.1 Error Code".

2.2.3 d64h_open

- **Description:**

  This function opens the device node of PCI-D64HU based on the specific Card ID. If this function returns successfully, the process that calls this function will own that specific device until d64h_close() is called. The device node of PCI-D64HU is ought to be owned before calling other functions. It's recommended to call d64h_scan() and d64h_get_cardinfo() to get the Card ID.

- **Syntax:**

  short d64h_open(BYTE bCardID)

- **Parameter:**

  bCardID: The specific Card ID that is configured with the on-board Dip-Switch.

- **Return:**

  Please refer to "Section 2.1 Error Code".

2.2.4 d64h_close

- **Description :**

  This function closes the device node of PCI-D64HU based on the specific Card ID. After calling this function, the PCI-D64HU card will be released, and other process can open it..

- **Syntax :**

  short d64h_close(BYTE bCardID)

- **Parameter :**

  bCardID: The specific Card ID that is configured with the on-board Dip-Switch.

- **Return:**

  Please refer to "Section 2.1 Error Code".

2.2.5 d64h_di_available_memory

- **Description :**

  This function gets the size of DI buffer that is allocated in driver. The unit of allocated buffer is reported in kilo-bytes.

- **Syntax :**

  short d64h_di_available_memory(BYTE bCardID, U32 *pMemSize)

- **Parameter :**

  bCardID: The specific Card ID that is configured with the on-board Dip-Switch.

  pMemSize: The pointer to the size of DI buffer, in kilobytes (KB).

- **Return:**

  Please refer to "Section 2.1 Error Code".

2.2.6 d64h_do_available_memory

- **Description :**

  This function gets the size of DO buffer that is allocated in driver. The unit of allocated buffer is reported in kilo-bytes.

- **Syntax :**

  short d64h_do_available_memory(BYTE bCardID, U32 *pMemSize)

- **Parameter :**

  bCardID: The specific Card ID that is configured with the on-board Dip-Switch.

  pMemSize: The pointer to the size of DO buffer, in kilobytes (KB)

- **Return:**

  Please refer to "Section 2.1 Error Code".

2.2.7 d64h_di_buffer_get

- **Description :**

  This function gets the size of DI buffer and the virtual address to access this DI buffer. These buffer addresses help programmer to get the DI acquisition data directly.

- **Syntax :**

  short d64h_di_buffer_get(BYTE bCardID, U32 *pMemSize, PVOID* pLowBufAddr, PVOID* pHighBufAddr)

- **Parameter :**

  bCardID: The specific Card ID that is configured with the on-board Dip-Switch.
  pMemSize: The pointer to the size of DI buffer, in kilobytes (KB)
  pLowBufAddr: The pointer to the address of low-part buffer. pHighBufAddr: The pointer to the address of high-part buffer.

- **Return:**

  Please refer to "Section 2.1 Error Code".

2.2.8 d64h_do_buffer_get

- **Description :**

  This function gets the size of DO buffer and the virtual address to access this DO buffer. These buffer addresses help programmer to update the DO output data directly.

- **Syntax :**

  short d64h_do_buffer_get(BYTE bCardID, U32 *pMemSize, PVOID* pLowBufAddr, PVOID* pHighBufAddr)

- **Parameter :**

  bCardID: The specific Card ID that is configured with the on-board Dip-Switch.
  pMemSize: The pointer to the size of DO buffer, in kilobytes (KB)
  pLowBufAddr: The pointer to the address of low-part buffer. pHighBufAddr: The pointer to the address of high-part buffer.

- **Return:**

  Please refer to "Section 2.1 Error Code".

2.3 Digital Input/Output Functions

2.3.1 d64h_di_config

- **Description :**

  This function configures the functionality of the following continuous digital-input.

- **Syntax :**

  short d64h_di_config(BYTE bCardID, U16 wTrigSource, U16 wExtTrigEnable, U16 wTrigPolarity, U16 wI_REQ_Polarity)

- **Parameter :**

  bCardID: The specific Card ID that is configured with the on-board Dip-Switch.

  wTrigSource: TRIG_SOURCE_INT_PACER,
  TRIG_SOURCE_EXT_STROBE or
  TRIG_SOURCE_HANDSHAKE.

  wExtTrigEnable: DI_WAITING or DI_NOWAITING.

  wTrigPolarity: DI_TRIG_RISING or DI_TRIG_FALLING. This parameter is required only when wExtTrigEnable is set as DI_WAITING.

  wI_REQ_Polarity: IREQ_RISING or IREQ_FALLING. This parameter is required only when wTrigSource is set as TRIG_SOURCE_EXT_STROBE or TRIG_SOURCE_HANDSHAKE.

- **Return:**

  Please refer to "Section 2.1 Error Code".

2.3.2 d64h_do_config

- **Description :**

  This function configures the functionality of the following continuous digital-output.

- **Syntax :**

  short d64h_do_config(BYTE bCardID, U16 wTrigSource, U16 wO_REQ_Enable, U16 wOutTrigHigh)

- **Parameter :**

  bCardID: The specific Card ID that is configured with the on-board Dip-Switch.
  wTrigSource: TRIG_SOURCE_INT_PACER or
  TRIG_SOURCE_HANDSHAKE.
  wO_REQ_Enable: OREQ_ENABLE or OREQ_DISABLE.
  wOutTrigHigh: OTRIG_HIGH or OTRIG_LOW. This parameter is required only when wO_REQ_Enable is set as OREQ_ENABLE.

- **Return:**

  Please refer to "Section 2.1 Error Code".

2.3.3 d64h_di_readport

- **Description :**

  This function reads all digital-input, and stores the data into one 32-bit variable.

- **Syntax :**

  short d64h_di_readport(BYTE bCardID, U32* pValue)

- **Parameter :**

  bCardID: The specific Card ID that is configured with the on-board Dip-Switch.

  pValue: The pointer to the 32-bit variable that stores all digital-input.

- **Return:**

  Please refer to "Section 2.1 Error Code".

2.3.4 d64h_di_readline

- **Description :**

  This function reads specific digital-input line.

- **Syntax :**

  short d64h_di_readline(BYTE bCardID, U16 wLine, U16* pValue)

- **Parameter :**

  bCardID: The specific Card ID that is configured with the on-board Dip-Switch.

  wLine: The parameter indicates the specific line to be input.

  pValue: The pointer to the 16-bit variable that stores the specific digital-input line.

- **Return:**

  Please refer to "Section 2.1 Error Code".

2.3.5 d64h_di_async_dblbuf_mode

- **Description :**

  This function configures the operation mode of digital-input acquisition.

- **Syntax :**

  short d64h_di_async_dblbuf_mode (BYTE bCardID, U16 wDblBufEnable)

- **Parameter :**

  bCardID: the specific Card ID that is configured with the on-board Dip-Switch.

  wDblBufEnable: DOUBLE_BUFFER_MODE_ENABLE or
  DOUBLE_BUFFER_MODE_DISABLE.

- **Return:**

  Please refer to "Section 2.1 Error Code".

2.3.6 d64h_di_async_dblbuf_halfready

- **Description :**

  This function checks the availability of ring-buffer when the digital-input acquisition is configured as double-buffer mode. If the pHalfReady indicates the ring-buffer is ready, please call d64h_di_async_dblbuf_transfer() to copy the available acquisition data.

- **Syntax :**

  short d64h_di_async_dblbuf_halfready (BYTE bCardID, BOOLEAN *pHalfReady)

- **Parameter :**

  bCardID: the specific Card ID that is configured with the on-board Dip-Switch.
  pHalfReady: the pointer to the memory that stores the availability of ring-buffer.

- **Return:**

  Please refer to "Section 2.1 Error Code".

2.3.7 d64h_di_async_dblbuf_transfer

- **Description :**

  This function extract the DI acquisition data from the ready half-buffer.

- **Syntax :**

  short d64h_di_async_dblbuf_transfer (BYTE bCardID, void *pBuffer)

- **Parameter :**

  bCardID: The specific Card ID that is configured with the on-board Dip-Switch.

  pBuffer: The pointer to the user-provided buffer that the acquisition data will be transferred to.

- **Return:**

  Please refer to "Section 2.1 Error Code".

2.3.8 d64h_di_async_check

- **Description :**

  This function checks if the asynchronous operation is completed. If the digital-input is completed, the number of samples will be returned.

- **Syntax :**

  short d64h_di_async_check (BYTE bCardID, BOOLEAN *pStopped, U32 *pAccessCount)

- **Parameter :**

  bCardID: The specific Card ID that is configured with the on-board Dip-Switch.

  pStopped: The pointer to the variable that stores the status of digital-input acquisition.

  pAccessCount: The pointer to the variable that stores the number of samples when the digital-input is completed.

- **Return:**

  Please refer to "Section 2.1 Error Code".

2.3.9 d64h_di_async_clear

- **Description :**

  This function terminates the in-progress digital-input acquisition, d64h_continue_readport()

- **Syntax :**

  short d64h_di_async_clear (BYTE bCardID, U32 *pAccessCount)

- **Parameter :**

  bCardID: The specific Card ID that is configured with the on-board Dip-Switch.

  pAccessCount: The pointer to the variable that stores the number of acquired-samples

- **Return:**

  Please refer to "Section 2.1 Error Code".

2.3.10 d64h_continue_readport

- **Description :**

  This function starts the continuous digital-input. The 32-bit acquisition data will be recorded into driver buffer continuously.

- **Syntax :**

  short d64h_continue_readport(BYTE bCardID, void *pBuffer, U32 dwSampleCount, F64* pSampleRate, U16 wSyncMode)

- **Parameter :**

  bCardID: The specific Card ID that is configured with the on-board Dip-Switch.

  pBuffer: The address of user-provided DI buffer. This parameter is ignored when enabling double-buffer mode.

  dwSampleCount: The samples of each acquisition.

  pSampleRate: The pointer to the address that stores sampling-rate. This call-by-reference parameter passes the desired sampling-rate to library; and stores the actual sampling-rate when this function returns. (0.001 ≦ valid sampling-rate ≦ 10,000,000)

  wSyncMode : The digital-input acquisition mode, SYNCH_OP or ASYNCH_OP.

- **Return:**

  Please refer to "Section 2.1 Error Code".

2.3.11 d64h_conti_di_status

- **Description :**

  This function reports the buffer-overrun status of digital-input buffer.

- **Syntax :**

  short d64h_conti_di_status (BYTE bCardID, U16 *pStatus)

- **Parameter :**

  bCardID: The specific Card ID that is configured with the on-board Dip-Switch.

  pStatus: The pointer to the variable that stores the status of digital-input buffer.
  The value in this variable will be 0 or DI_OVERRUN_STATUS.

- **Return:**

  Please refer to "Section 2.1 Error Code".

2.3.12 d64h_do_writeport

- **Description :**

  This function writes all digital-output with the user-assigned data.

- **Syntax :**

  short d64h_do_writeport (BYTE bCardID, U32 dwValue)

- **Parameter :**

  bCardID: The specific Card ID that is configured with the on-board Dip-Switch.

  dwValue: The data to be written to digital-output port.

- **Return:**

  Please refer to "Section 2.1 Error Code".

2.3.13 d64h_do_writeline

- **Description :**

  This function writes specific digital-input line.

- **Syntax :**

  short d64h_do_writeline (BYTE bCardID, U16 wLine, U16 wValue)

- **Parameter :**

  bCardID: The specific Card ID that is configured with the on-board Dip-Switch.

  wLine: The parameter indicates the specific line to be input.

  wValue: The data to be written to specific line of digital-output port.

- **Return:**

  Please refer to "Section 2.1 Error Code".

2.3.14 d64h_do_readport

- **Description :**

  This function reads all digital-output data, and stores in one 32-bit variable.

- **Syntax :**

  short d64h_do_readport(BYTE bCardID, U32* pValue)

- **Parameter :**

  bCardID: The specific Card ID that is configured with the on-board Dip-Switch.

  pValue: The pointer to the 32-bit variable that stores all digital-output data.

- **Return:**

  Please refer to "Section 2.1 Error Code".

2.3.15 d64h_do_readline

- **Description :**

  This function reads specific digital-input line.

- **Syntax :**

  short d64h_do_readline(BYTE bCardID, U16 wLine, U16* pValue)

- **Parameter :**

  bCardID: The specific Card ID that is configured with the on-board Dip-Switch.

  wLine: The parameter indicates the specific line to be output.

  pValue: The pointer to the 16-bit variable that stores the specific digital-output line.

- **Return:**

  Please refer to "Section 2.1 Error Code".

2.3.16 d64h_do_async_dblbuf_mode

- **Description :**

  This function configures the operation mode of digital-output acquisition.

- **Syntax :**

  short d64h_do_async_dblbuf_mode (BYTE bCardID, U16 wDblBufEnable)

- **Parameter :**

  bCardID: the specific Card ID that is configured with the on-board Dip-Switch.

  wDblBufEnable: DOUBLE_BUFFER_MODE_ENABLE or
  DOUBLE_BUFFER_MODE_DISABLE.

- **Return:**

  Please refer to "Section 2.1 Error Code".

2.3.17 d64h_do_async_dblbuf_halfready

- **Description :**

  This function checks the availability of ring-buffer when the digital-output data is transferred as double-buffer mode. If the pHalfReady indicates the ring-buffer is ready, the d64h_do_async_dblbuf_transfer() helps to update the output data.

- **Syntax :**

  short d64h_do_async_dblbuf_halfready (BYTE bCardID, BOOLEAN *pHalfReady)

- **Parameter :**

  bCardID: the specific Card ID that is configured with the on-board Dip-Switch.
  pHalfReady: the pointer to the memory that stores the availability of ring-buffer.

- **Return:**

  Please refer to "Section 2.1 Error Code".

2.3.18 d64h_do_async_dblbuf_transfer

- **Description :**

  This function updates the DO output data to the idle half-buffer.

- **Syntax :**

  short d64h_do_async_dblbuf_transfer (BYTE bCardID, void *pBuffer)

- **Parameter :**

  bCardID: The specific Card ID that is configured with the on-board Dip-Switch.

  pBuffer: The pointer to the user-provided buffer that contains the DO output data.

- **Return:**

  Please refer to "Section 2.1 Error Code".

2.3.19 d64h_do_async_check

- **Description :**

  This function checks if the asynchronous operation is completed. If the digital-output is completed, the number of samples will be returned.

- **Syntax :**

  short d64h_do_async_check (BYTE bCardID, BOOLEAN *pStopped, U32 *pAccessCount)

- **Parameter :**

  bCardID: The specific Card ID that is configured with the on-board Dip-Switch.

  pStopped: The pointer to the variable that stores the status of digital-output operation.

  pAccessCount: The pointer to the variable that stores the number of samples when the digital-output is completed..

- **Return:**

  Please refer to "Section 2.1 Error Code".

2.3.20 d64h_do_async_clear

- **Description :**

  This function terminates the in-progress digital-output operation, d64h_continue_writeport() and d64h_continue_pattern_write().

- **Syntax :**

  short d64h_do_async_clear (BYTE bCardID, U32 *pAccessCount)

- **Parameter :**

  bCardID: The specific Card ID that is configured with the on-board Dip-Switch.

  pAccessCount: The pointer to the variable that stores the number of acquired-samples.

- **Return:**

  Please refer to "Section 2.1 Error Code".

2.3.21 d64h_continue_writeport

- **Description :**

  This function starts the continuous digital-input. The 32-bit acquisition data will be recorded into driver buffer continuously. If the digital-data is needed to be output repeatedly, please call d64h_do_async_dblbuf_mode() and enable double-buffer mode.

- **Syntax :**

  short d64h_continue_writeport (BYTE bCardID, void *pBuffer, U32 dwSampleCount, U16 wIterations , F64* pSampleRate, U16 wSyncMode)

- **Parameter :**

  bCardID: The specific Card ID that is configured with the on-board Dip-Switch.
  pBuffer: The address of user-provided DO buffer.
  dwSampleCount: The samples of each acquisition.
  wIterations: This parameter is reserved for future use.
  pSampleRate: The pointer to the address that stores sampling-rate. This call-by-reference parameter passes the desired sampling-rate to library; and stores the actual sampling-rate when this function returns.
  (0.001 ≦ valid sampling-rate ≦ 10,000,000)
  wSyncMode : The digital-output operation mode, SYNCH_OP or ASYNCH_OP.

- **Return:**

  Please refer to "Section 2.1 Error Code".

2.3.22 d64h_continue_pattern_write

- **Description :**

  This function reads the output-pattern and stores it into a circular buffer, and loops the output infinitely. This feature is supported by onboard-circuit, and neither computing-power nor bus-bandwidth is consumed.

- **Syntax :**

  short d64h_continue_pattern_write (BYTE bCardID, void *pBuffer, U32 dwSampleCount, F64* pSampleRate)

- **Parameter :**

  bCardID: The specific Card ID that is configured with the on-board Dip-Switch.

  pBuffer: The address of user-provided DO buffer.

  dwSampleCount: The samples of each acquisition.

  (2 < dwSampleCount ≦ 2048)

  pSampleRate: The pointer to the address that stores sampling-rate. This call-by-reference parameter passes the desired sampling-rate to library; and stores the actual sampling-rate when this function returns.
  (0.001 ≦ valid sampling-rate ≦ 10,000,000).

- **Return:**

  Please refer to "Section 2.1 Error Code".

2.3.23 d64h_conti_do_status

- **Description :**

  This function reads the output-pattern and stores it into a circular buffer, and loops the output infinitely. This feature is supported by onboard-circuit, and neither computing-power nor bus-bandwidth is consumed.

- **Syntax :**

  short d64h_conti_do_status (BYTE bCardID, U16 *pStatus)

- **Parameter :**

  bCardID: The specific Card ID that is configured with the on-board Dip-Switch.
  pStatus: The pointer to the variable that stores the status of digital-output buffer.
  The value in this variable will be 0 or DO_UNDERRUN_STATUS.

- **Return:**

  Please refer to "Section 2.1 Error Code".

2.3.24 d64h_do_ext_trigline_write

- **Description :**

  This function reports the buffer-overrun status of digital-input buffer.

- **Syntax :**

  short d64h_do_ext_trigline_write (BYTE bCardID, U16 wValue)

- **Parameter :**

  bCardID: The specific Card ID that is configured with the on-board Dip-Switch.

  wValue: The state to be set on external trigger line, OTRIG_HIGH or OTRIG_LOW.

- **Return:**

  Please refer to "Section 2.1 Error Code".

2.4 Digital Filter Functions

2.4.1 d64h_DI_filter_set

- **Description :**

  This function set the digital filter for all DI lines.

- **Syntax :**

  short d64h_DI_filter_set (BYTE bCardID, U16 wDIFilter)

- **Parameter :**

  bCardID: The specific Card ID that is configured with the on-board Dip-Switch.
  wDIFilter: the delay factor for digital filter (0 ≦wDIFilter≦ 127). wDIFilter * 25ns < delay for data-latching ≦ (wDIFilter +1) * 25ns.

- **Return:**

  Please refer to "Section 2.1 Error Code".

2.4.2 d64h_IREQ_filter_set

- **Description :**
  The PCI_180X_M_FUN_2 is designed for arbitrary waveform generation.
- **Syntax :**
  short d64h_IREQ_filter_set (BYTE bCardID, U16 wIREQFilter)
- **Parameter :**
  bCardID: The specific Card ID that is configured with the on-board Dip-Switch.
  wIREQFilter: the delay factor for digital filter (0 ≦wIREQFilter≦ 127).
  wIREQFilter * 25ns < delay for data-latching ≦ (wIREQFilter +1) * 25ns.
- **Return:**
  Please refer to "Section 2.1 Error Code".

2.4.3 d64h_OACK_filter_set

- **Description :**

  This function set the digital filter for O_ACK input signal.

- **Syntax :**

  short d64h_OACK_filter_set (BYTE bCardID, U16 wOACKFilter)

- **Parameter :**

  bCardID: The specific Card ID that is configured with the on-board Dip-Switch.
  wOACKFilter: the delay factor for digital filter (0 ≦wOACKFilter≦ 127).
  wOACKFilter * 25ns < delay for data-latching ≦ (wOACKFilter +1) * 25ns.

- **Return:**

  Please refer to "Section 2.1 Error Code".

2.4.4 d64h_ITRG_filter_set

- **Description :**

  This function set the digital filter for I_TRG input signal.

- **Syntax :**

  short d64h_ITRG_filter_set (BYTE bCardID, U16 wITRGFilter)

- **Parameter :**

  bCardID: The specific Card ID that is configured with the on-board Dip-Switch.
  wITRGFilter: the delay factor for digital filter (0 ≦wITRGFilter≦ 127).
  wITRGFilter * 25ns < delay for data-latching ≦ (wITRGFilter +1) * 25ns.

- **Return:**

  Please refer to "Section 2.1 Error Code".

2.4.5 d64h_OREQ_width_set

- **Description :**

  This function set the width of O_REQ output signal.

- **Syntax :**

  short d64h_OREQ_width_set (BYTE bCardID, U16 wOREQWidth)

- **Parameter :**

  bCardID: The specific Card ID that is configured with the on-board Dip-Switch.
  wOREQWidth: the delay factor for digital filter (0 ≦wOREQWidth≦ 7).

  Width of O_REQ = (2^ (wOREQWidth + 1)) * 25ns.

- **Return:**

  Please refer to "Section 2.1 Error Code".

3. PCI-D64HU Demo Programs For Linux

All of demo programs will not work normally if the Linux driver would not be installed correctly. During the installation process of PCI-D64HU Linux driver, the install-scripts “drv_load” will setup the correct kernel driver. After the user install the driver, the related demo programs, development library and declaration header files for different development environments are presented as follows.

Package	Directory Path	File Name	Description
pcid64hu	Include	D64HDLL.h	Linux library header
	lib	libd64hu.a	Linux static library
	examples	dio.c	DI/O demo
		pattern_output	DO Pattern Demo
		time_span.c	Delay setting time
		timer_dma_double_buff	DI/O DMA demo
		timer_dma_async	DI/O DMA demo
		timer_dma_sync	DI/O DMA demo

Table 3.1