Data Transfer for Linux

Version 1.0

Programmer's Guide

Document No. M2T-1QA28500061-00

Document Version History

Version	Date	Notes
1.0	2017/09/29	USB3.0 only support
	2017/10/31	PCIe support

Chapter – 1 : Introduction to Data Transfer for Linux

This guide helps you get started quickly with Data Transfer for Linux.

The software is an external data transfer API library(libiqdata.so) of the instrument. It is the same function with Software for High Speed Data Transfer for Windows. It helps you get IQ data from the instrument at high speed using a Linux host computer.

The demo application gets you started quickly with the API. It runs as a Command Line Interface [CLI].

The whole block diagram of IQ data transfer from the instrument is shown below.

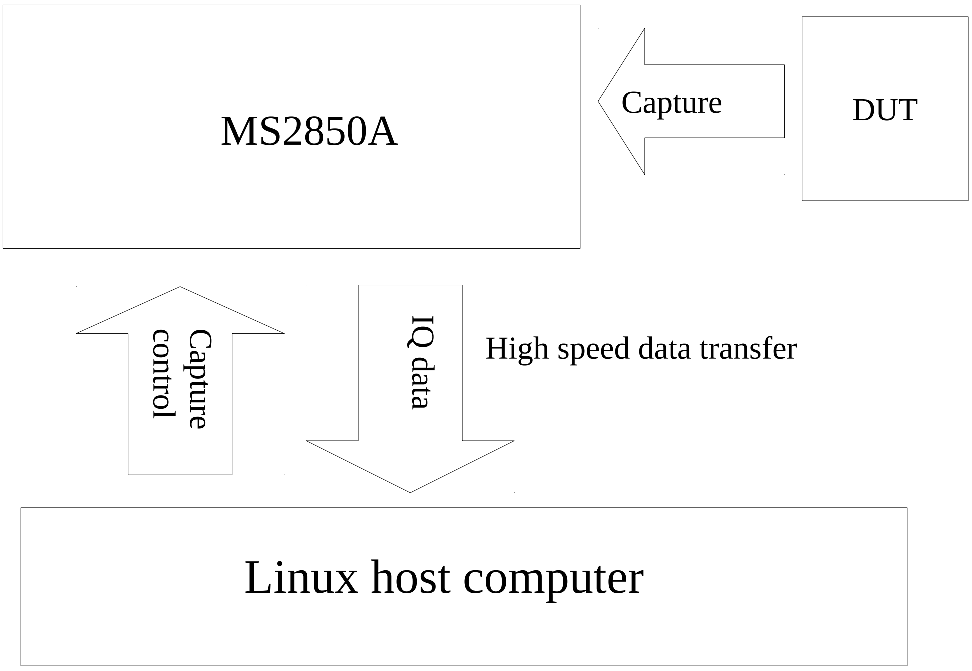

Figure-1 : Block Diagram of Data Transfer for Linux

Chapter – 2 : Folder structure

Once the files are extracted, the following folders are created.

```
/home/user/DataTransfer_Linux_version/
                                      |
                                      |-bin/
                                      |-demo_src/
                                      |
                                      |-docs/
                                      |-driver/
                                      |        |-config/
                                      |        |-img/
                                      |        |-module/
                                      |
                                      |-include/
                                      |-lib/
```

- The **bin** folder is the location to save demo application binary file.
- The **demo_src** folder is the location to save demo application source file.
- The **docs** folder is the location to save software related documents.
- The **driver** folder is the location to save driver related files. It contains **config** folder, **img** folder and **module** folder where the USB3.0 configuration file, firmware file and PCIe module file are saved.
- The **include** folder is the location to save the API related header file.
- The **lib** folder is the location to save the API object file.

Chapter – 3 : Getting started

It is assumed that the installation procedure documented in the user guide document has been followed before starting the following steps.

Demo application

$ datatransfer_demo

The following interpreter message is displayed.

```
|-------------------------------------------------------------|
|This is a demo program of Data Transfer for Linux.  |
|It shows how to read the IQ data of the instrument   |
|from the external PC and evaluate the speed of        |
|data transfer.                                                   |
|-------------------------------------------------------------|
Open start...
Open is OK
Before reading the data, it is necessary to capture the instrument at least once.
If you are ready to read, enter 'y' or 'Y',or enter another key to quit.
Is it OK? ['y' or 'Y'/others]: y ←Enter
Please intput datasize of IQ data to read[Unit:MB]: 100 ←Enter
If you want to save read IQ data, enter 'y' or 'Y', or enter another key to read only.
Is it OK? ['y' or 'Y'/others]: y ←Enter
Read start...
Read is OK.
Speed of data transfer: 330.000 MB/s.
Elapsed time of data transfer: 303.030 ms.

Do you want to continue? ['y' or 'Y'/others to quit]: q←Enter
Quitting demo.
Close start...
Close is OK.
```

In the interpreter message, the user has 4 inputs. First, it is necessary to capture the instrument manually when the “Open is OK” message is shown before entering user first input. After capturing, enter “y” or “Y” to proceed to next step. Input the reading data size. It should be an integer greater than zero and the unit is in megabytes (MB). The third input is whether to save read IQ data or not. In the example, “y” is to save data. Lastly, the user is asked whether or not to run the demonstration again. In the example, “q” is to quit demo application while “y” or “Y” is to continue.

Chapter – 4 : Programmers Guide to the API Library

This chapter gives details on how to develop code using the **libiqdata.so** library. (Note:when compiling code, link options '*-liqdata*', '*-lnsauwb*' '*-lcyusb*' and '-lnsawb' are required. See '*makefile*' for details.)

The **iqdata.h** header file present in **$./include** subdirectory, needs to be included in your program. For example, if you are in the **$./demo_src** subdirectory, you would see
#include “../include/iqdata.h”

This header file defines all APIs for users.

1. **Opening device**

   ```
   bool iqdata_open(void);
   ```

   This function initializes API to prepare for transferring IQ data. In order to read IQ data, it will be called first.

2. **Closing device**

   ```
   bool iqdata_close(void);
   ```

   This function closes API to finish transferring IQ data. If the application opens the device, it will always be called last.

3. **Reading data**

   ```
   bool iqdata_read(void* pbuffer,
                    int* offset,
                    uint32_t datasize,
                    uint64_t read_offset);
   ```

   This function reads IQ data from the instrument. It cannot save the data to the local PC. If you want to save the data to the local PC, the function is overloaded.

   ```
   bool iqdata_readplus(void* pbuffer,
                        int* offset,
                        uint32_t datasize,
                        uint64_t read_offset,
                        bool output = false,
                        char* path = (char*)"",
                        char* filename = (char*)"");
   ```

This function reads IQ data from the instrument. It can save the data to the local PC

4. Getting data size

uint64_t iqdata_get_data_size(void);

This function gets size of IQ data. It is the size of one capture of IQ data inside the FPGA of the instrument.

5. Checking level overflow

bool iqdata_get_level_overflow(void);

This function gets sign of level overflow whether it occured or not.

6. Getting error message

char iqdata_get_fail_msg(void);*

This function gets last information if run failed. When calling another API, if it fails, it is a function to get failure information.

7. Getting version information

char iqdata_get_version(void);*

This function gets version information of API.

Using the library – Detailed Function Documentation

Function Documentation

```
/*****************************************************************************
  Prototype    : bool iqdata_open(void);
  Description  : This function initializes API to prepare for transferring IQ data.
  Parameters   : None
  Return Value : Returns a boolean,
                 if success,true;else false.
 *****************************************************************************/
bool iqdata_open(void);

/*****************************************************************************
  Prototype    : bool iqdata_close(void);
  Description  : This function closes API to finish transferring IQ data.
  Parameters   : None
  Return Value : Returns a boolean,
                 if success,true;else false.
 *****************************************************************************/
bool iqdata_close(void);

/*****************************************************************************
  Prototype    : bool iqdata_readplus(void* pbuffer,
                                      int* offset,
                                      uint32_t datasize,
                                      uint64_t read_offset,
                                      bool output = false,
                                      char* path = (char*)"",
                                      char* filename = (char*)"");
  Description  : This function reads IQ data from the instrument.
                 It can save the data to the local PC.
  Parameters   :
             void* pbuffer                : The memory pointer for reading IQ data
             int* offset                  : The memory pointer offset for reading IQ data
             uint32_t datasize            : The data size in bytes
             uint64_t read_offset         : The data shift of DMA reading in bytes
             bool output = false          : The switch for saving IQ data, default value is 'false'
char* path = (char*)""                    : The path of saved IQ data, default value is """ (<=512bytes)
char* filename = (char*)""             : The filename of saved IQ data, default value is """(<=64bytes)
  Return Value : Returns a boolean,
                 if success,true;else false.
 *****************************************************************************/
```

```
bool iqdata_readplus(void* pbuffer,
                                          int* offset,
                                          uint32_t datasize,
                                          uint64_t read_offset,
                                          bool output = false,
                                          char* path = (char*)"",
                                          char* filename = (char*)"");

/**************************************************************************
  Prototype    : bool iqdata_read(void* pbuffer,
                                          int* offset,
                                          uint32_t datasize,
                                          uint64_t read_offset);

  Description  : This function reads IQ data from the instrument.
                          It can not save the data to the local PC.
  Parameters   :
                  void* pbuffer                : The memory pointer for reading IQ data
                  int* offset                  : The memory pointer offset for reading IQ data
                  uint32_t datasize            : The data size in bytes
                 uint64_t read_offset          : The data shift of DMA reading in bytes

  Return Value : Returns a boolean,
                  if success,true;else false.
 **************************************************************************/
bool iqdata_read( void* pbuffer,
                                     int* offset,
                                     uint32_t datasize,
                                     uint64_t read_offset);

/**************************************************************************
  Prototype    : char* iqdata_get_fail_msg(void);
  Description  : This function gets last information if run failed.
  Parameters   : None
  Return Value : Returns a string, shows the error message of last run failed.
 **************************************************************************/
char* iqdata_get_fail_msg(void);

/**************************************************************************
  Prototype    : uint64_t iqdata_get_data_size(void);
  Description  : This function gets size of IQ data.
  Parameters   : None
  Return Value : Returns size of IQ data in bytes.
 **************************************************************************/
```

```
uint64_t iqdata_get_data_size(void);

/**************************************************************************************
  Prototype    : bool iqdata_get_level_overflow(void);
  Description  : This function gets sign of level overflow whether it occurred or not.
  Parameters   : None
  Return Value : Returns a boolean.
                 if level overflow occurred,true;else false.
 **************************************************************************************/
bool iqdata_get_level_overflow(void);

/**************************************************************************************
  Prototype    : char* iqdata_get_version(void);
  Description  : This function gets version information of API.
  Parameters   : None
  Return Value : Returns a string, shows the version information.
 **************************************************************************************/
char* iqdata_get_version(void);
```