APSpec Programmers guide

Document Reference Number: APSpec-Prog-001
Version Number: 0.001
Issue Date: Jul 2022

Table of Contents

1 Preface

The APSpec library libAPSpec.lib and the header libAPSpec.h are a proprietary interface for Aipplica Spectrometer Devices. This document provides an explanation of the functions available to application developers via the APSpec library.

Any software code examples given in this document are for information only. The examples are for reference only and no guarantees or support by Aipplica Systems is implied.

2 Introduction

Unde-the-hood, Aipplica spectrometer use FTDIs ICs with Multi-Protocol Synchronous Serial Engine (MPSSE) hardware block. Use of the MPSSE requires certain components be in place, both software and hardware:

Please familiarize yourself & install FTDI's drivers and libraries as indicated, on FTDI's website. The following links may be helpful.

FTDI D2XX Device Drivers The latest D2XX device drivers are required.Installation guides for various operating systems are available at the FTDI Website. See http://ftdichip.com/Drivers/D2XX.htm for the latest downloads.

For Linux systems APSpec library is built on top of LibMPSSE_spi and libftd2xx provided by FTDI. Please refer to FTDI's webpage for documentation on these two libraries and installation instructions.

For Windows FTDI provides FTD2XX.DLL drivers as part of the common device driver (CDM) package. Please install those and Include WinTypes.h, libMPSSE_spi.h and ftd2xx.h and library file libMPSSE.lib during compilation.

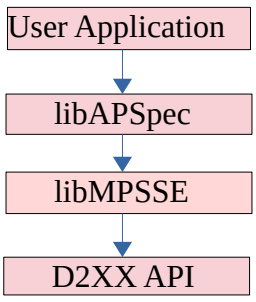

Fig 1

libMPSSE has three different APIs, one each for I2C, SPI and JTAG. This application will use the SPI section only. The libMPSSE (Linux and Windows versions) sample code, release notes etc can be downloaded from the FTDI website at : http://www.ftdichip.com/Support/SoftwareExamples/MPSSE.htm

3 APSpec functions

The functions listed in this section are compatible with Aipplica Spectrometers

3.1 AP_ReadID

Supported Operating Systems: Linux, Windows

Summary:

A command to read a MCU, Sensor, Firmware and hardware ID parameters stored in the device. This will allow the application to cater for different configurations.

Definition
AP_STATUS AP_ReadID(uint* ptr)
Parameters: pointer to a block of 4 words

Return Value
AP_OK if successful, otherwise the return value is an AP error code.
On successful completion,location pointed to by memory pointed is updated with MCU ID, Sensor ID, Firmware ID and Hardware ID

Remarks

The ID uniquely identify the Spectrometer device parameters and are stored in Onboard Read only Nonvolatile memory.

Sensor ID = 0x0001 for 1500 pixel Sensor
Sensor ID = 0x0002 for 2048 pixel Sensor
Sensor ID = 0x0003 for 3278 pixel Sensor

Hardware ID = 0x001 50 pixel data in a block,
15 pixel gap between successive pixel data.

Example

```
uint32 i;
uint8 idbfr[8];
uint8 *pidbuf;
pidbuf = &idbfr[0];


AP_ReadID(pidbuf);
for(i=1;i<8;i+=2)  {
      if (i==1) printf("MCUID: %d ",  *(pidbuf+i));
      if (i==3) printf("SENSID: %d ", *(pidbuf+i));
      if (i==5) printf("HWID: %d ",   *(pidbuf+i));
      if (i==7) printf("SENSID: %d\n",*(pidbuf+i));
}

printf("Read ID  Successful\n");
```

3.2 AP_ReadPattern

Supported Operating Systems: Linux, Windows

Summary:

A command returns a 50 word ascending pattern from 0x0000 to 0x0049 in the memory location pointer provided. This will allow the application to do a handshake with the hardware and verify robustness of command and data transfer. It is typically called immediately after AP_initUSB and retried a maximum of 16 times to ensure the communication and data transfer with underlying hardware is robust. Number of retries is reported as return value.

Definition

AP_STATUS AP_ReadID(uint* ptr)
Parameters: pointer to a block of 50 words

Return Value

Return value is an number of retries before command is successfully executed.
On successful completion, a 50 word ascending pattern from 0x0000 to 0x0049 is written into the memory location pointed to.

Remarks

Example:

```
uint8 ptrnbfr[100];
uint8 *pptrnbfr = &ptrnbfr[0];

AP_ReadPattern(pptrnbfr);
for(i=0;i<50;i++)  {
        if ((ptrnbfr[2*i] != 0) || (ptrnbfr[2*i+1] != i)) {
                printf("Ascending pattern verification fail\n");
        }
}
```

3.3 AP_SetIntTime

Supported Operating Systems: Linux, Windows

Summary:

This command updates the Hardware integration counter value which sets the charge integration time for pixel.

Definition
AP_STATUS AP_SetIntTime(uint16 Integration_Time)
Parameters: 16 bit word

Return Value

AP_OK if successful, otherwise the return value is an AP error code.
On successful completion, Integration time Hardware counter value is updated.

Remarks

One count of integration counter corresponds to 4usec. Minimum value of integration counter is 10msec which corresponds to a count of 2500, Maximum is 262msec corresponding to a count to 65536.

Example

AP_SetInTime(3500)

3.4 AP_ReadArray

Supported Operating Systems: Linux, Windows

Summary:

This command read the entire pixel from the sensors and loads then in word buffer for further processing by the application.

Definition
AP_STATUS AP_ReadArray(uint16* ptr)
Parameters: pointer to 1500 word array buffer

Return Value

AP_OK if successful, otherwise the return value is an AP error code.
On successful completion, word array buffer will be loaded with digitized value read from pixels

Remarks

Entire pixel array is read by scanning the array multiple times determined by parameters such as total number of pixels and pixels per read-block. These are coded in Sensor ID and Hardware ID. Each scan will follow integration time parameter.

Example

```
uint8  ptrnbfr[3000];
uint8 *pptrnbfr = &ptrnbfr[0];

AP_ReadArray(pptrnbfr);
for(i=0;i<3000;i++) { printf("%d\t",*pptrnbfr++); }
```

3.5 AP_ReadCalData

Supported Operating Systems: Linux, Windows

Summary:

This command returns Spectrometer Calibration Coefficients stored in the on-board-nonvolatile-memory.

Definition
AP_STATUS AP_ReadCalData(float* ptr)
Parameters: pointer to floating pointer number array

Return Value

AP_OK if successful, otherwise the return value is an AP error code.
On successful completion, floating pointer number array will be loaded with number of calibration coefficients and each calibration coefficient.

Remarks
A maximum of seven (7) calibration coefficients can be stored in the Non volatile memory.

Example

```
uint8 ptrnbfr[100];
uint8 *pptrnbfr = &ptrnbfr[0];
uint8 NumCalCoeff;
uint8 fourbytes[4];

AP_ReadCal(pptrnbfr);
NumCalCoeff = ptrnbfr[1]/4;
printf("Num-of-Cal-Coeff: 0%d\n", NumCalCoeff);
for(i=0;i<NumCalCoeff;i++) {
       for(j=1;j<5;j++) {
            fourbytes[j-1] = ptrnbfr[8*i+2*j+1]; }
            printf("C[%d]%f\n",i,*(float *)fourbytes);
       }
printf("Read Calibration Data Successful\n");
```

3.6 AP_WriteCalData

Supported Operating Systems: Linux, Windows

Summary:

This command stores supplied Spectrometer Calibration Coefficients in the on-board-nonvolatile-memory.

Definition
AP_STATUS AP_WriteCalData(float Coeff[7])
Parameters: Up to floating pointer number array containing 4 byte floating point coefficients C[0] to C[6].

Return Value

AP_OK if successful, otherwise the return value is an AP error code.
On successful completion, floating pointer number array calibration coefficient will be stored in Onboard-Nonvolatile Memory.

Remarks
A maximum of seven (7) calibration coefficients can be stored in the Non volatile memory. This allows users to implement up 6^{th} order polynomial for correction.

Example

```
float Coeff[7];

Coeff[0] = 12.34; Coeff[1] = 56.78; Coeff[2] = 90.12; Coeff[3] = 23.45;
Coeff[4] = 4.5; Coeff[5] = 5.6; Coeff[6] = 6.7;

AP_WriteCalData(Coeff);
```

3.7 AP_ReadFrame

Supported Operating Systems: Linux, Windows

Summary:

This command reads the first block of non-consecutive pixels from the sensor array. Hardware ID determines number of pixels read and spacing between pixels.

Definition
AP_STATUS AP_ReadArray(uint* ptr)
Parameters: pointer to 50 word array buffer

Return Value

AP_OK if successful, otherwise the return value is an AP error code.
On successful completion, word buffer will be loaded with digital value read from non-consecutive pixels.

Remarks

Example

```
uint8 ptrnbfr[100];
uint8 *pptrnbfr = &ptrnbfr[0];

AP_ReadFrame(pptrnbfr);
for(i=0;i<100;i++) { printf("%d\t",*pptrnbfr++); }
```

3.8 AP_Init

Supported Operating Systems: Linux, Windows

Summary:

This command performs a sequence of hardware initialization steps and must be issued before any other command. Specifically it get the number of available SPI channels, for the first available channel gets channel information such as serial number and device information, then it proceeds to open that channel and proceeds to set channel configuration options such as clock rate, latency timer and SPI Mode.

Definition
AP_STATUS AP_Init()
Parameters: nil

Return Value

AP_OK if successful, otherwise the return value is an AP error code.
On successful completion, further AP commands can be issued.

Remarks

Example

3.9 AP_Close

Supported Operating Systems: Linux, Windows

Summary:

This command performs close all open connections and must be the last command issed before exiting the application program

Definition
AP_STATUS AP_Close()
Parameters: nil

Return Value

AP_OK if successful, otherwise the return value is an AP error code.
On successful completion, a clean closure of software connection has been performed.

Remarks

Example

Appendix A: Type Definitions

```
UCHAR       1 byte unsigned character
PUCHAR      pointer to unsigned character
PCHAR       pointer to character
DWORD       Unsigned long
LPDWORD     Pointer to Unsigned long

AP_STATUS (DWORD)
      AP_OK                   =          0
      AP_INVALID_HANDLE       =          1
      AP_DEVICE_NOT_FOUND     =          2
      AP_DEVICE_NOT_OPENED    =          3
      AP_IO_ERROR             =          4
      AP_INSUFFICIENT_RESOURCES =        5
      AP_INVALID_PARAMETER    =          6
      AP_CAL_READ_FAILED      =          7
      AP_CAL_WRITE_FAILED     =          8
      AP_INVALID_ARGS         =          9
      AP_NOT_SUPPORTED        =          10
      AP_OTHER_ERROR          =          11[1]
```

Appendix B: References

Document References: NA

Acronyms and Abbreviations

Terms	Description
CDM	Combined Driver Model
D2XX	FTDI's proprietary "direct" driver interface via FTD2XX.DLL
MPSSE	Multi Protocol Serial Engine

Appendix C: Figures and Tables

List of Tables: Nil

List of Figures: Fig 1: Driver Stack

Appendix D - Revision History

Document Title: APSpec Programmer's Guide
Document Reference No.: AP_0001
Product Page: http://www.aipplica.com.com/Downloads.html

Revision	Changes	Date
0.0	Initial Release	7 Jul 2022

1