DPScope SE
Programming Interface Description

Version 1.0.0
July 24, 2012

1 Introduction

The DPScope SE is acting as a standard USB HID (Human Interface device). Key connection parameters are:

Vendor ID (VID): 0x04D8 (hex)
Product ID (PID): 0xF891 (hex)

Report size: 64 bytes

To get started with USB HID communication on the PC side, here is a thread where you find a complete demo program in Visual Basic (VB6):

http://www.mikroe.com/forum/viewtopic.php?f=88&t=26891

Other than that, the webpage of Jan Axelson has a lot of code for different programming platforms.

A good idea is first to implement readback of the scope's software revision, and control of the frontpanel LED, before venturing into more complex territory.

For any questions, feedback, suggestions, bug reports please send an emaul to:

support@dpscope.com

2 Command Description

The general communication scheme is as follows: Communication in HID mode is through data packets of fixed length, in the case of the DPScope SE the size is 64 bytes (the maximum allowed for HID). There can be up to 1000 exchanges per second, thus maximum data rate for HID is close to 64 KB/sec.

The PC always initiates the communication by sending a data packet to the scope. The first byte of the packet is the command code (one byte), given below for each command (note that the values are binary, not ASCII, so e.g CMD_ARM is binary value 5, not the ASCII character '5'. Additional bytes hold command parameters when necessary. The scope processes the command and sends back an acknowledge packet, which indicates that it has finished processing and is ready to receive the next command. Per default the first byte in the packet is the command code, except for commands where explicitly noted. The PC must wait until it receives this response before it can send another command. This handshaking protocol assures that neither the PC nor the DPScope SE can overload the other side, and no commands can get lost.

In the command description below there are the following conventions:

- Command code (in header) is the first byte of the packet from PC to scope.
- Parameters are subsequent bytes of the packet from PC to scope.
- Acknowledge byte is first byte of answer packet from scope back to PC.
- Data bytes are additional data bytes of answer packet from scope back to PC.

The acknowledge byte may be missing. In this case the first data byte is the first byte of the packet (i.e. at the same location where the acknowledge byte would have been).

To repeat, no matter how many or how few actual data bytes there are, the packets always consist of 64 bytes.

2.1 CMD_PING (2)

Short description: Requests echo from controller

Parameters:	none
Acknowledge byte:	yes
Data bytes:	10 bytes identifier (“DPSCOPE SE”)

Requests echo from controller. Use this command to check if DPScope SE is connected and functional. If this is the case the command will return a 10-byte ASCII string (without termination character) with the contents “DPSCOPE SE” (without the quotation marks).

2.2 CMD_REVISION (3)

Short description: Requests firmware revision number from controller

Parameters:	none
Acknowledge byte:	no
Data bytes:	Byte 1: major version number Byte 2: minor version number

Determines the version number (e.g. major version 2, minor version 1 would mean V2.1). The firmware numbers are sent in binary format. This information is useful if you want to handle functional differences between different firmware or hardware versions.

2.3 CMD_ARM (5)

Short description: Sets all acquisition parameters and arms scope (so it waits for trigger and then acquires data).

Parameters:

Byte 1 First ADC channel to acquire
Byte 2 Second ADC channel to acquire
Byte 3 ADC acquisition parameters
Byte 4 timer 0 preload MSB (determines sample rate)
Byte 5 timer 0 preload LSB (determines sample rate)
Byte 6 timer 0 prescaler bypass (0 = use prescaler, 1 = bypass prescaler)
Byte 7 timer 0 prescaler selection as power of 2 (7=div256, 0=div2)
Byte 8 sample shift first channel
Byte 9 sample shift second channel
Byte 10 sample subtract first channel
Byte 11 sample subtract second channel
Byte 12 trigger source (0 = auto, 1 = triggered)
Byte 13 trigger polarity (0 = falling edge, 1 = rising edge)
Byte 14 trigger level MSB (currently not used)
Byte 15 trigger level LSB (only applicable if triggering on CH1, not for ext. trigger)
Byte 16 sampling mode: (0 = real time, 1 = equivalent time)
Byte 17 equivalent time sample interval in 0.5 usec increments
Byte 18 equivalent time trigger stability check period (half of byte 17 value is a good choice)
Byte 19 trigger channel to use (1 = CH1 gain 1, 2 = CH1 gain 10, 3 = ext. trigger)

Acknowledge byte:	yes
Data bytes:	none

Channel selection: 0-10 selects ADC channel 0-1 of the PIC18F14K50. 15 selects the fixed voltage reference (4096mV) that can be used e.g. to determine the actual supply voltage.

The acquisition parameter is SFR register ADCON2 (refer to the datasheet of the PIC18F14K50 for details): ADCON2 = 128 + ACQT * 8 + ADCS, where

ADCS = 2 ' (FOSC/16) is the fastest that seems to work (outside spec!)
ACQT = 5 ' minimal valid values: >= 3 for FOSC/64; >=5 (12 Tad) for FOSC/32; 7 (20 Tad) for FOSC/16

Fosc=48 MHz

ADCS 6 = FOSC/64, ACQT 3 = 6 Tad --> Tacq_min = 64 * (11 + 6 + 2) / 48 = 25.33 us
ADCS 2 = FOSC/32, ACQT 5 = 12 Tad --> Tacq_min = 32 * (11 + 12 + 2) / 48 = 16.67 us
ADCS 5 = FOSC/16, ACQT 7 = 20 Tad --> Tacq_min = 16 * (11 + 20 + 2) / 48 = 11.00 us

The ADCs are 10 bits (0…1023), but due to limited RAM space the scope only stores a bit values. Value 512 corresponds to 0V input, <512 are negative voltages, >512 positive voltages).

Thus the raw 10-bit numbers need to be scaled (shifted and offset) to fit into 8 bits. Different scaling selections effectively implement “digital” gain selection. Parameter sets used for this "software gain" in the scope software are: (shift,subtract) = (2,0), (1,128), or (0,192) for gain 1, 2, 4, respectively.

A rough gain selection is made through the ADC channel to use (gain 1 path or gain 10 path).

So e.g. shift = 1, subtract = 128, and path = gain 10 results in an overall effective gain of 2*10 = 20.

2.4 CMD_DONE (6)

Short description: Query whether scope has already finished the acquisition

Parameters:	none
Acknowledge byte:	no
Data bytes:	Byte 1: done flag

Queries the scope whether the current acquisition has finished or not (0 = acquisition not yet finished, >0 = acquisition finished). Once acquisition has finished you can read back the acquired data.

2.5 CMD_ABORT (7)

Short description: Disarms the scope, so it's read for a new command

Parameters:	none
Acknowledge byte:	yes
Data bytes:	none

Disarms the scope (i.e. terminates a pending acquisition), so the scope is read for a new command. Use this command in case the latest acquisition has not finished yet but you want to send a new command.

Also refer to the commands CMD_ARM and CMD_READBACK for further information.

2.6 CMD_READBACK (8)

2.7 CMD_READADC (9)

Short description: Reads back ADC directly (with 10 bit resolution, returns 2 bytes per channel)

Parameters:	Byte 1: first ADC channel to read Byte 2: second ADC channel to read Byte 3: ADC acquisition parameters
Acknowledge byte:	no
Returned data:	Byte 1: signal on scope CH1 (MSB) Byte 2: signal on scope CH1 (LSB) Byte 3: signal on scope CH2 (MSB) Byte 4: signal on scope CH2 (LSB)

This command reads back the ADC (analog-to -digital converter) for both channels directly. The resolution of the result is 10 bits. This command allows slow-speed data acquisition under full user control since it does not use the scope's internal sampling engine and memory.

The acquisition parameter is SFR register ADCON2 (refer to the datasheet of the PIC18F14K50 for details): ADCON2 = 128 + ACQT * 8 + ADCS, where

ADCS = 2 ' (FOSC/16) is the fastest that seems to work (outside spec!)
ACQT = 5 ' minimal valid values: >= 3 for FOSC/64; >=5 (12 Tad) for FOSC/32; 7 (20 Tad) for FOSC/16

Fosc=48 MHz

ADCS 6 = FOSC/64, ACQT 3 = 6 Tad --> Tacq_min = 64 * (11 + 6 + 2) / 48 = 25.33 us
ADCS 2 = FOSC/32, ACQT 5 = 12 Tad --> Tacq_min = 32 * (11 + 12 + 2) / 48 = 16.67 us
ADCS 5 = FOSC/16, ACQT 7 = 20 Tad --> Tacq_min = 16 * (11 + 20 + 2) / 48 = 11.00 us

2.8 CMD_STATUS_LED (10)

Short description: Turn the status LED on the front panel on/off

Parameters:	Byte 1 LED status (1 = on, 0 = off)
Acknowledge byte:	yes
Data bytes:	none

2.9 CMD_WRITE_MEM (11)

Short description: Writes a byte to a memory location on the microcontrollers SFR

Parameters:	Byte 1 Address MSB Byte 2 Address LSB Byte 3 Data to write to SFR address

Acknowledge byte:	yes
Data bytes:	none

Address range is restricted to SFR.

2.10 CMD_READ_MEM (12)

Short description: Reads a memory location on the microcontrollers SFR

Parameters:	Byte 1 Address MSB Byte 2 Address LSB
Acknowledge byte:	yes
Data bytes:	Byte 1 Data byte read from SFR address

Address range is restricted to SFR. Addresses outside will return zero.

2.11 CMD_WRITE_EEPROM (13)

Short description: Writes a memory location on the microcontroller’s data EEPROM

Parameters:	Byte 1 Address MSB Byte 2 Address LSB Byte 3 Data to write to EEPROM address
Acknowledge byte:	yes
Data bytes:	none

2.12 CMD_READ_EEPROM (14)

Short description: Writes a memory location from the microcontroller’s data EEPROM

Short description: Reads a memory location on the microcontrollers SFR

Parameters:	Byte 1 Address MSB Byte 2 Address LSB
Acknowledge byte:	yes
Data bytes:	Byte 1 Data byte read from EEPROM address

2.13 CMD_READ_LA (15)

Short description: Reads back the state of the logic analyzer pins (port B)

Parameters:	none
Acknowledge byte:	no
Data bytes:	Byte 1: status of PORT B

Returns full 8 bits, but only 4 highest correspond to logic analyzer channels.

2.14 CMD_ARM_LA (16)

Short description: Sets logic analyzer acquisition parameters (sample rate, trigger condition)

2.15 CMD_INIT (17)

Short description: Re-initialize microcontroller

Parameters:	Byte 1 blink LED (1) or not (0)
Acknowledge byte:	yes
Data bytes:	none

Note that if blinking LED is selected, the init process will take so long that USB connection will be lost and will have to be restored.

2.16 CMD_SERIAL_INIT (18)

Short description: Initialize external trigger pin for serial data output

Parameters:	none
Acknowledge byte:	yes
Data bytes:	none

Sets up the external trigger pin as an output, so subsequently data can be sent using CMD_SERIAL_TX. You can use this to control peripherals (e.g. auxiliary boards) connected to the logic analyzer header (ground, power, and serial data).

2.17 CMD_SERIAL_TX (19)

Short description: Send one byte over trigger pin at 9600 baud

Parameters:	Data byte to send out
Acknowledge byte:	yes
Data bytes:	none