Das Oszi Protocol

The Hantek DSO5xxxB (aka Das_Oszi Project) series oscilloscopes, rebadged under various other names, have a USB interface intended for communicating with software running on a PC. A proprietary protocol is used by the Hantek Windows software. This page documents that protocol.

Hantek provides the TTScope software, which is for Windows only. No SDK is available for any platform.

Contents

Credits

- The initial import of this documentation comes from the work "tinhead" has done over on the www.mikrocontroller.net (http://www.mikrocontroller.net/articles/Hantek_Protokoll) pages.

Introduction

The DSOs are identifiable only by the USB VID:PID 049f:505a (this VID is actually assigned to Compaq), and the bus and device address. When operating several DSO5xxxBs on a PC they cannot be distinguished by their serial number as Hantek sets the iSerial field in the USB descriptor to 0.

The CPU is the MIZI Research/Samsung S3C2410 SMDK. It contains the USB char driver (usb-char.c, Linux 2.4.18), and which uses the VID:PID 049f:505a. Although the DSO is based on a newer kernel (2.6.13), this driver was taken from the previous model Tekway DST1000 (not B!).

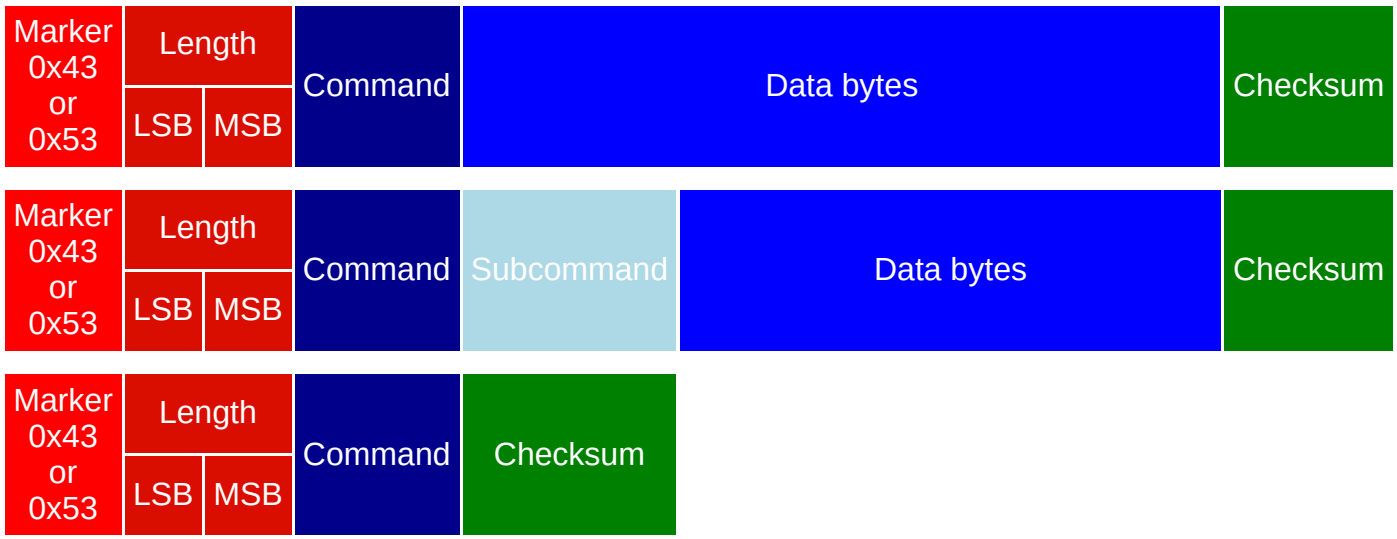
Apparently the DSO5xxxBM/BMV models, running Linux 2.6.30.4, use the "serial gadget" driver. Whether this is an improvement is currently not known.

Another effect of this driver is that the DSO briefly shows up at boot with another VID:PID on the USB bus. The bootloader (vivi) uses a SoC monitor, such as the Samsung S3C2440 SMDK (u2440mon), which when booting reports its own VID:PID on the USB port. While this may be very useful for hacking, backup or similar, changing the VID:PID causes BSODs on Win64 PCs.

The protocol uses USB bulk transfers (outbound endpoint 1, inbound endpoint 2) with a maximum packet size of 64 bytes. DSO protocol messages longer than 64 bytes are split up over several bulk packets.

Hantek's DSO protocol is layered directly on top of the USB bulk transfers. It is request/response oriented, meaning the USB host sends a request and receives a list of answers. The protocol can therefore be used in a single-threaded application using synchronous USB communication, even though Hantek's TTScope application uses separate transmit and receive threads.

Basic Structure



The message format is the same for requests and responses. Each message begins with a marker byte. There are two different markers:

- 0x53: normal message
- 0x43: debug message

This is followed by two bytes indicating the length of the message, LSB first, excluding the marker and length fields.

The word length is followed by the command byte. The reply to this command will have bit 7 set in the command byte. Therefore command bytes from PC to device are always smaller than 0x80, and the responses are always greater than or equal to 0x80.

The command byte is followed by command-specific data bytes. Some commands use the first data byte as a sub-command. The number of data bytes can be zero.

The data byte is followed by a primitive checksum byte. To calculate this checksum, add up all the bytes of the message. The LSB of this sum is used as the checksum.

0x53 Normal messages

Most 0x53 messages are used by the TTScope software.

0x00 Echo

All data bytes in the request are simply returned unchanged.

0x01 Read DSO settings

This gives a long record in which the current settings of the DSO are binary encoded.

The request:

0x53	0x02	0x00	0x01	0x56

The reply contains the DSO settings (SYSData):

0x53	Length		0x81	SYSData	Checksum
	LSB	MSB			

Details of the SYSData format can be found at http://www.mikrocontroller.net/articles/Datei:SysDATA_v1.0.zip

To ensure that the SYSData structure is decoded correctly it is recommended to read the **/protocol.inf** file immediately after connecting (see Read file).

If the protocol.inf file on the DSO is not readable or available the reply will be empty, i.e. it will contain no data byte:

0x53	0x02	0x00	0x81	0xD6

0x02 Read sample data

Only the sample data of the two channels (CH1/CH2) can be retrieved. The REF, MATH and FFT data can not be accessed, which is probably why Hantek's TTScope software shows nonsense data under "PowerSpectrum" and "WaveTabulator".

The request:

0x53	0x04	0x00	0x02	0x01	0x00 for CH1 or 0x01 for CH2	Checksum

The DSO responds with at least 3 and at most 202 packets. The first packet (subcommand 0x00) describes the total length of the sample data. The second packet (subcommand 0x01) contains a channel number and the sample data itself. If the sample data is longer than 10000 bytes, further packets with subcommand 0x01 are sent.

This is an example of a small transfer from a DSO5xxxB set to 4ns/DIV and 4K buffer:

0x53	0x06	0x00	0x82	0x00	Sample data length (3 bytes)	Checksum
0x53	0x5c	0x02	0x82	0x01	CHn + 0x..0x.. (600 bytes of sample data)	Checksum

0x53	0x04	0x00	0x82	0x02	CHn (0x00 for CH1 or 0x01 for CH2)	Checksum

Here's an example with more data (150 packets with 10000 sample bytes per packet) from a DSO1202BV with 2ms/DIV and 2M buffer:

0x53	0x06	0x00	0x82	0x00	Sample data length (3 byte)	Checksumme
0x53	0x14	0x27	0x82	0x01	CHn + 0x..0x.. (10000 bytes of sample data)	Checksum
					148 more packets with subcommand 0x01 and CHn sample data	
0x53	0x14	0x27	0x82	0x01	CHn + 0x..0x.. (10000 bytes of sample data)	Checksum
0x53	0x04	0x00	0x82	0x02	CHn (0x00 for CH1 or 0x01 for CH2)	Checksum

If an errors occurs during transmission, a special packet is sent with subcommand 0x03. This can also be sent if the DSO is in STOP mode, in which case there is no data available.

0x53	0x04	0x00	0x82	0x03	CHn (0x00 for CH1 or 0x01 for CH2)	Checksum

Since the sample data contains no information about the current DSO settings (channel, volts/DIV, timebase), it is important to read the settings first. The following commands ensure a consistency data set:

- 0x12 Lock panel
- 0x01 Read DSO settings
- 0x12 Unlock panel
- 0x02 Read sample data
- 0x00 or 0x01 can be used as "idle" commands

If both channels need to be read, the samples can be read sequentially:

- 0x12 Lock panel
- 0x01 Read DSO settings
- 0x12 Unlock panel
- 0x02 Read CH1 sample data
- 0x02 Read CH2 sample data
- 0x00 or 0x01 can be used as "idle" commands

Originally (in the Tekway DST1000 (http://www.tekwayins.com/product.asp?ArticleID=7) with 2.5Kpoint memory) the panel was also locked during the sample data transmission. This is not recommended for the current DSOs, since the data volumes are much higher than in this original model.

The sample data are post-processed from the image memory. The values are 10 DIV vertical (-127 to 127) and 20 DIV horizontal.

Note: the actual number may be closer to 10.2 DIV vertically (510 pixels). Horizontally, the number is 16 DIV (640 pixels) with menu visible and 19.2 DIV (768 pixels) without menu.

A list of the amount of memory depth per channel is available here in PDF format: http://www.mikrocontroller.net/articles/Datei:Sampledaten.pdf

If the window timebase is smaller than the main timebase, only the visible part of the data is transmitted (see http://www.mikrocontroller.net/topic/205820?page=4#2613537).

0x10 Read file

This function can be used to read any file on the DSO, but is primarily intended to read the **protocol.inf** (see Read DSO settings) and **keyprotocol.inf** (see Keypress trigger) files.

The request must include the complete file path.

The response uses two subcommands:

- 0x01: Data
- 0x02: Checksum of all data

If the contents of the file don't fit in one response packet, further packets with the subcommand 0x01 are sent. The packet with subcommand 0x02 marks the end of the file transfer, and contains a checksum of the entire file.

0x11 Write DSO settings

To write settings, the data format specified in Read DSO settings must be used:

The response contains a status byte, where 0x00 denotes success and anything else is an error:

0x12 Lock/unlock control panel, start/stop acquisition

Lets you start/stop DSO acquisition (subcommand 0x00) or lock/unlock the DSO's front panel controls (subcommand 0x01).

Start DSO acquisition:

Stop DSO acquisition:

The response contains the sent start/stop command:

As far as we know the "start/stop DSO acquisition" function triggers the same action internally as pressing the "run stop" button on the control panel. However it appears to be somewhat buggy. Call this function before retrieving sample data from the DSO.

Lock control panel:

0x53	0x04	0x00	0x12	0x01	0x01	Checksum 0x6b

Unlock control panel:

0x53	0x04	0x00	0x12	0x01	0x00	Checksum 0x6a

When the control panel is locked a red key appears on the top status bar of the DSO's display, and the controls no longer respond.

The response contains the sent lock/unlock command:

0x53	0x04	0x00	0x92	0x01	0x01 or 0x00	Checksum

0x13 Keypress trigger

Lets you simulate the press of nearly any button on the DSO's control panel. The desired key is selected by two bytes of data:

0x53	0x04	0x00	0x13	0x..	0x..	Checksum

The first byte is the key code. The second byte is the number of key presses, according to Hantek. However values higher than one appear to result in only one keypress regardless, so there is no point is specifying anything other than 0x01 for this.

The following keycodes are available for the Hantek DSO5xxxB/BM/BMV, Tekway DST1xxxB and Voltcraft DSO-3062C bench scopes:

```
0x00 0x01 - F0
0x01 0x01 - F1
0x02 0x01 - F2
0x03 0x01 - F3
0x04 0x01 - F4
0x05 0x01 - F5
0x06 0x01 - F6
0x07 0x01 - F7
0x08 0x01 - V0 left turn
0x09 0x01 - V0 right turn
0x0A 0x01 - V0 press
0x0B 0x01 - Save/Recall
0x0C 0x01 - Measure
0x0D 0x01 - Acquire
0x0E 0x01 - Utility
0x0F 0x01 - Cursor
0x10 0x01 - Display
0x11 0x01 - Autoset
0x12 0x01 - Single Seq Taste
0x13 0x01 - Run/Stop
0x14 0x01 - Help
0x15 0x01 - Default Setup
0x16 0x01 - Save to USB
0x17 0x01 - Math Menu
0x18 0x01 - CH1 Menu
0x19 0x01 - CH1 Position left turn
0x1A 0x01 - CH1 Position right turn
0x1B 0x01 - CH1 Position press
0x1C 0x01 - CH1 Volts/Div left turn
0x1D 0x01 - Ch1 Volts/Div right turn
0x1E 0x01 - CH2 Menu
0x1F 0x01 - CH2 Position left turn
0x20 0x01 - CH2 Position right turn
0x21 0x01 - CH2 Position press
0x22 0x01 - CH2 Volts/Div left turn
0x23 0x01 - Ch2 Volts/Div right turn
0x24 0x01 - Horz Menu
0x25 0x01 - Horizontal Position right turn
0x26 0x01 - Horizontal Position left turn
0x27 0x01 - Horizontal Position press
```

```
0x28 0x01 - Horizontal Sec/Div left turn
0x29 0x01 - Horizontal Sec/Div right turn
0x2A 0x01 - Trig Menu
0x2B 0x01 - Trigger Level left turn
0x2C 0x01 - Trigger Level right turn
0x2D 0x01 - Trigger Level press
0x2E 0x01 - Set to 50%
0x2F 0x01 - Force Trig
0x30 0x01 - Probe Check
```

The following keycodes are available for the Hantek DSO1xxxB/BV handheld scopes:

```
0x00 0x01 - Fx NULL
0x01 0x01 - Menu on/Off
0x02 0x01 - F1
0x03 0x01 - F2
0x04 0x01 - F3
0x05 0x01 - F4
0x06 0x01 - F5
0x07 0x01 - DMM V
0x08 0x01 - DMM A
0x09 0x01 - DMM Ohm
0x0A 0x01 - DMM Diode
0x0B 0x01 - DMM Continuity
0x0C 0x01 - DMM Capacitance
0x0D 0x01 - DMM/DSO switch
0x0E 0x01 - Save/Recall
0x0F 0x01 - Measure
0x10 0x01 - Utility
0x11 0x01 - Cursor
0x12 0x01 - CH1 Menu
0x13 0x01 - Math Menu
0x14 0x01 - CH2 Menu
0x15 0x01 - Default Setup
0x16 0x01 - Save to USB
0x17 0x01 - Math Menu
0x18 0x01 - CH1 Volts/Div +
0x19 0x01 - CH1 Volts/Div -
0x1A 0x01 - CH1 Position +
0x1B 0x01 - CH1 Position -
0x1C 0x01 - CH2 Position +
0x1D 0x01 - CH2 Position -
0x1E 0x01 - CH2 Volts/Div +
0x1F 0x01 - CH2 Volts/Div -
0x20 0x01 - Horizontal Sec/Div -
0x21 0x01 - Horizontal Sec/Div +
0x22 0x01 - Horizontal Position +
0x23 0x01 - Horizontal Position -
0x24 0x01 - Trigger Level +
0x25 0x01 - Trigger Level -
0x26 0x01 - Run/Stop
0x27 0x01 - Autoset
0x28 0x01 - Multifunktion Enter
0x29 0x01 - Multifunktion Up
0x2A 0x01 - Multifunktion Down
0x2B 0x01 - Multifunktion Left
0x2C 0x01 - Multifunktion Right
```

To make sure keycodes are valid on the DSO, you should read the **/keyprotocol.inf** immediately after connecting (see Read file).

The keyprotocol.inf file is structured as follows:

- the first line represents the number of key codes
- the second line marks the beginning of the list
- the following lines contain the key name as used by the firmware, followed by the number of bytes used for each key code
- the last line marks the end of the list

The line number after [START] - 1 corresponds to that key's keycode. This is the keyprotocol.inf file for a Hantek DSO5202B:

```
[TOTAL]  49
[START]
[FN-0-KEY] 1
```

```
[FN-1-KEY] 1
[FN-2-KEY] 1
[FN-3-KEY] 1
[FN-4-KEY] 1
[FN-5-KEY] 1
[FN-6-KEY] 1
[FN-7-KEY] 1
[FN-MLEFT-KEY] 1
[FN-MRIGHT-KEY] 1
[FN-MZERO-KEY] 1
[MENU-SR-KEY] 1
[MENU-MEASURE-KEY] 1
[MENU-ACQUIRE-KEY] 1
[MENU-UTILITY-KEY] 1
[MENU-CURSOR-KEY] 1
[MENU-DISPLAY-KEY] 1
[CT-AUTOSET-KEY]  1
[CT-SINGLESEQ-KEY] 1
[CT-RS-KEY] 1
[CT-HELP-KEY] 1
[CT-DS-KEY] 1
[CT-STU-KEY] 1
[VT-MATH-MENU-KEY] 1
[VT-CH1-MENU-KEY] 1
[VT-CH1-PSUB-KEY] 1
[VT-CH1-PADD-KEY] 1
[VT-CH1-PZERO-KEY] 1
[VT-CH1-VBSUB-KEY] 1
[VT-CH1-VBADD-KEY] 1
[VT-CH2-MENU-KEY] 1
[VT-CH2-PSUB-KEY] 1
[VT-CH2-PADD-KEY] 1
[VT-CH2-PZERO-KEY] 1
[VT-CH2-VBSUB-KEY] 1
[VT-CH2-VBADD-KEY] 1
[HZ-MENU-KEY] 1
[HZ-PSUB-KEY] 1
[HZ-PADD-KEY] 1
[HZ-PZERO-KEY] 1
[HZ-TBSUB-KEY] 1
[HZ-TBADD-KEY] 1
[TG-MENU-KEY] 1
[TG-PSUB-KEY] 1
[TG-PADD-KEY] 1
[TG-PZERO-KEY] 1
[TG-PHALF-KEY] 1
[TG-FORCE-KEY] 1
[TG-PROBECHECK-KEY] 1
[END]
```

The reply to a key command contains a data byte that refers to the menu that was displayed in response to the keypress. The menu need not actually be displayed.

0x14 Set system time

Lets you set the system time.

The data bytes:

- Byte 1: Year (LSB)
- Byte 2: Year (MSB)
- Byte 3: Month (1...12)
- Byte 4: Day (1...31)
- Byte 5: Hour (0...23)

- Byte 6: Minute (0...59)
- Byte 7: Second (0...59)

The reply is a packet without data bytes:

0x53	0x02	0x00	0x94	Checksumme

0x20 Screenshot

After receiving this command, the DSO sends a screenshot image without size or color palette information. It is sent back in multiple reply messages, up to a total of 384000 bytes. This corresponds to the 800x480 resolution of the display, with a color depth of 8 bits.

The PC sends a request without data bytes:

0x53	0x02	0x00	0x20	0x75

The DSO responds with 37 packets of 10208 bytes, with subcommand 0x01. The 38th packet is a little shorter, containing only 6304 bytes. After this another packet is sent with subcommand 0x02, containing a primitive checksum.

0x53	0xE3	0x27	0xA0	0x01	0x.. 0x.. 0x.. .. .. .. (10208 bytes)	Checksum

0x53	0xA3	0x18	0xA0	0x01	0x.. 0x.. 0x.. .. .. .. (6304 bytes)	Checksum
0x53	0x04	0x00	0xA0	0x02	Image checksum (1 byte)	Checksum

These DSO protocol packets are split up into multiple USB packets, but your USB library should resolve this transparently.

The received image is in the format of a windows bitmap with 256 colors. The image is flipped vertically, i.e. the bottom line is sent first. The color palette is compiled into the `dso.exe` program on the DSO, and TTScope on the PC, and has a length of 1024 bytes.

In the handheld Hantek DSO1202B/BV, DSO1102B/BV and DSO1062B/BV the display has a 640x480 resolution. This results in a total 307200 bytes of data transmitted for the screenshot.

The PC sends a request without data bytes:

0x53	0x02	0x00	0x20	0x75

The DSO responds with 30 packets containing 10208 bytes, subcommand 0x01. The 31st packet contains only 960 bytes, and is again followed by a final packet containing a checksum for the screenshot data:

0x53	0xE3	0x27	0xA0	0x01	0x.. 0x.. 0x.. .. .. .. (10208 bytes)	Checksum

0x53	0xA3	0x18	0xA0	0x01	0x.. 0x.. 0x.. .. .. .. (960 bytes)	Checksum
0x53	0x04	0x00	0xA0	0x02	Image Checksum (1 byte)	Checksum

As of August 2013 (verified on the DSO5072P), Hantek has updated both the Samsung SoC from S3C2440 to S3C2416, and upgraded the Linux kernel from 2.6.13 to 3.2. The display is still 800x480, but the newer hardware sends twice as much data per screenshot, for a total of 768000 bytes.

0x21 Read system time

Lets you read the device's system time.

The PC sends a request without data bytes:

The reply contains seven data bytes:

The data bytes:

- Byte 1: Year (LSB)
- Byte 2: Year (MSB)
- Byte 3: Month (1...12)
- Byte 4: Day (1...31)
- Byte 5: Hour (0...23)
- Byte 6: Minute (0...59)
- Byte 7: Second (0...59)

0x43 Debug messages

The 0x43 messages are not used by the TTScope software, but nevertheless expose interesting functionality. Hantek may be willing to answer some questions about these messages.

0x00 Read FPGA register

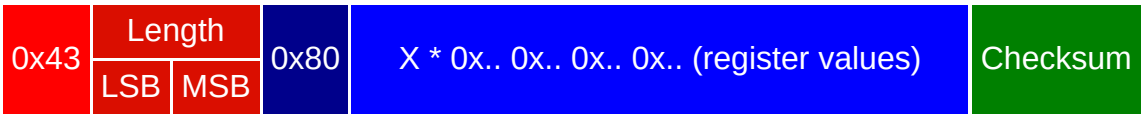
Lets you read one of more FPGA registers.

The request for a single register:

The response contains the 4-byte register value:

The request for multiple registers contains a starting register and the number of registers to fetch. The last register in this list (start register + number of registers) must not be higher than 0x4f.

The response contain the register values, 4 bytes per register:



Internally, this calls the firmware function **PcUartReadFpgaReg**.

Please note: this syntax is temporary, and may be incorrect.

0x01 Read FPGA FIFO contents

Lets you read directly from the FIFO memory. The request contains the buffer length in bytes:

The response contains the FIFO contents:

Please note: this syntax is temporary, and may be incorrect.

0x02 Read DSO firmware variables

The PC sends a request without data bytes:

The reply contains the DSO firmware variable values. These are stored on the DSO in the file `/param/sav/run1kb_yymmdd` and are loaded at boot time. If the file `chk1kb` is not available at boot time, default values are used. These default values are also used when the "Default" button is pressed on the DSO. The stored setups also include these values.

The DSO variables themselves are not yet documented; but at least the following information is stored:

- current UI (current menu, all user-selectable settings)
- DSO serial number
- Firmware version
- PCB hardware revision

0x03 Read DSO self-calibration

The request contains no data bytes:

The response contains the self-calibration info read from the DSO's memory. This is stored on the DSO in the file `/param/sav/chk1kb_yymmdd` and is loaded at boot time. If this file is not available, default values are used.

0x10 Read file

See Read file under **Normal messages**.

0x11 Virtual remote shell

Allows you to run shell commands. The command sent is stored on the DSO in a temporary file, which is then executed and the output is sent back in the reply packet. Instructions such as "**cd / tmp**" therefore do not make sense.

Commands that output over 10239 bytes cause the main "dso.exe" process to crash. If you have a shell on the DSO via the UART, you can run "**/dso.exe**" to restart it. Otherwise, powercycling the unit is necessary.

The request contains only the command:

The reply contains the shell output, followed by LF:

0x40 Write DSO firmware variables

The request contains the DSO firmware variables (see Read DSO firmware variables):

The response contains no data bytes:

To activate these new variables, the Init DSO function must be called.

0x41 Write DSO self-calibration

The request contains the self-calibration info (see Read DSO self-calibration).

The response contains no data bytes:

To activate this setting, the Init DSO function must be called.

0x42 Write FPGA register

The request contains the register number and value (see 0x00 Read FPGA register).

The response contains no data bytes:

Internally, this calls the firmware function **PcUartWriteFpgaReg**.

Please note: this syntax is temporary, and may be incorrect.

0x43 Send keycode

This function lets you send a keycode. The difference from the normal **0x53 message** Keypress trigger function is that this supports additional "special" keycodes, not just actual keys.

For example on the DST1xxxB/BV when the F6 key is pressed, it generates the debug keycode 0x42 (normal keycode 0x06 + 0x3c). When released, this generates the debug keycode 0xc2 (normal keycode 0x06 + 0xbc). This holds true for all keys in the **keyprotocol.inf** file.

For the Hantek DSO5xxxB/Voltcraft DSO-3062C/Tekway DST1xxxB, the values to add to the normal keycodes are 0x01 for keypress, and 0x81 for key release.

This function allows you to send both keypress and release events, and even the number of times to fire this event. It is thus possible to use this to step through menus, e.g. 2 x Utility, 2 x Page, 1 x F1.

The response contains no data bytes:

0x44 DSO Buzzer

Activates the DSO's buzzer for a period of time corresponding to the value specified * 100ms.

The response contains no data bytes:

0x45 Adjust interpolation

Lets you (temporarily) adjust changes in the Lagrange interpolation. You can set the boundaries (0-1000ps) and change the phases of all ADCs (0x00=0° up to 0xffff=360°). Actually the data from the FIFO are shifted in phase, not the ADCs.

The response contains no data bytes:

The Lagrange interpolation is enabled only in the TBID < 3 (i.e. horizontal deflection 2ns/DIV, 4ns/DIV and 8ns/DIV).

Please note: this syntax is temporary, and may be incorrect.

0x50 Write file

This function can be used to store a file on the DSO.

The request uses three subcommands. The first packet (subcommand 0x01) has the full file path on the DSO's filesystem. The second packet (subcommand 0x01) contains the file contents. If longer than 10000 bytes, further packets with this subcommand can be sent as needed. A packet with subcommand 0x02 marks the end of transmission, and contains a checksum of the file contents sent.

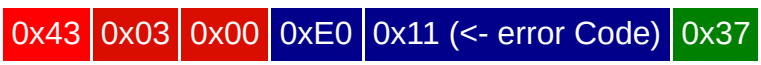
A response should be requested after each of the above packets. It contains no data bytes:

0x60 Update file

The function can be used to replace any existing file on the DSO's filesystem.

The structure of the request, and the response, is identical to Write file.

For some reason, after the first packet I get a reply with error code 0x11:



This stands for "file not found". However, the specified file is correctly replaced.

0x7F Init DSO

The request contains no data bytes:

0x43	0x02	0x00	0x7F	0xC4

Internally, this calls the following functions in the firmware:

- PauseSysAcq
- Acq_InitAcqParam
- Acq_SyncEqualAcq
- SyncChDispWhenEnterX
- SyncBodeAssiant
- SyncDsoScanSwitch
- InitCalculationManageParam
- InitAcqWaveEvent
- InitDispWaveEvent
- InitWaveSoftEvent
- Init_SwapTrig
- Tdc_SyncTdcTable
- Limites_Init
- InitLcdUnwaveareaShow
- InitLcdWaveAreaShow
- InitDsoOtherStat
- UpdateSysRunParam
- Fpga_SetTrigHoldTime
- Sync_AutoDispInterval
- ContinueSysAcq
- ForceShowWinBar

This is not entirely the same as a reboot since not all init functions are called. However it's enough to use before calling Write DSO firmware variables or Write DSO self-calibration.

The response contains no data bytes:

0x43	0x02	0x00	0xFF	0x44
