1 Anleitung für PeakTech Oszilloskop

1.1 Datennahme

Am Oszilloskop mit `run/stop` eine Messung starten. Sind die gewünschten Daten aufgezeichnet, mit `run/stop` anhalten und in Ruhe die Daten mit `save` auf einen USB-Stick speichern.

1.2 Daten plotten

Die Binärdaten können mit Gnuplot direkt dargestellt werden. Die Achsenbeschriftung ist dabei noch willkürlich.

1.2.1 Plot

mit gnuplot

```
FILE='20141003_533052'
plot \
FILE.'.bin' binary skip=112 for-
mat="%int8" using ($1) every 1024 with lines
```

FILE='20141003_533052' definiert die Variable FILE mit dem basename
FILE.'.bin' setzt den basename mit der Endung '.bin' zusammen
binary teilt gnuplot mit, dass es sich um Binärdaten handelt
skip=112 überspringt die ersten 112 Byte (der Header der Datei)
format="%int8" jedes Byte entspricht einem vorzeichenbehafteten 8-Bit Integer-Wert
using ($1) die Datei enthält nur eine Spalte: die Y-Werte
every 1024 Stelle nur jeden 1024sten Wert dar. Nimmt man Daten mit 10MegaSamples auf, lassen sich kaum alle Samples plotten
with lines Zeichne Linien statt Punkte, wie es auch auf dem Oszi dargestellt wird

1.2.2 Tabelle

Mit gnuplot kann man die Binärdaten auch in eine Tabelle exportieren:

```
set table FILE.'.csv'
plot [:] \
FILE.'.bin' binary skip=128 format="%int8" using ($1) every 1024
unset table
```

Das Funktioniert ähnlich wie oben, nur dass der Plot in eine Tabelle ausgegeben wird. Eine 10 Megasample große Messung ergibt dabei leicht eine 200MB große Tabelle.

1.3 Details zu den gespeicherten Daten

Die Softwareversion, der Kanal, die Eingestellte X- und Y-Auflösung und der Offset werden im Header gespeichert. Die angezeigten Messwerte (`measure`) wie Frequenz, Effektivwert usw. werden nicht gespeichert. Die Bildschirmgröße dient der Skalierung, er ist 10 Kästchen (divs) hoch und 15,2 Kästchen (divs) breit.

1.3.1 Berechnung X-Achse

Der Bildschirm ist 15,2 Kästchen breit. Die Zeit pro Kästchen wird gepeichert.

1.3.2 Berechnung Y-Achse

Die Bildschirmhöhe wird auf 250 Werte aufgeteilt (8 bit Sampletiefe aber 25 Punkte pro Bildschirmeinheit (Kästchen) mit 10 Einheiten (Kästchen)[1]). Mit dem Wert aus dem header *VoltScale* berechnet sich die Spannung daher zu int8 · *VoltScale*/25.

Eine weitere Möglichkeit ist die Darstellung als 16bit Zahlen. In den langsamen Modi (`slow mode`) werden scheinbar 16bit pro Sample gespeichert, dafür jedoch die gespeicherte Samplerate begrenzt: $1520 \cdot 2 = 3040$ Samples, $3040 \cdot 16\,\text{bit} = 6080\,\text{byte}$, also z.B. bei 100s/div max. 2Sa/s.

Die Zahlen sind little endian gespeichert. Besteht ein Wert aus mehreren Byte, kann man die Zahl umrechnen: Beginnt eine 4 Byte lange int32 an der Stelle 1, berechnet sich die Zahl zu int8[0x04] · 2^{24} + int8[0x03] · 2^{16} + int8[0x02] · 2^8 + int8[0x01].

[1] Tektronix teilt den Bildschirm genauso auf: 25 Werte pro Kästchen (div) mit 10 Kästchen. Bei einem 8 Kästchen Bildschirm liegt je ein Kästchen ober- und unterhalb des sichtbaren Bildschirms. http://www.tek.com/support/faqs/why-should-my-signal-fit-within-screen-my-tektronix-oscilloscope

byte #	Werte	Verwendung
0x00..0x05	SPBS01	Dateiformat
0x06 07	101=0x65 00	Protocol version
	30 byte	Proto 101: Sequence number
0x08 09	0x10 80	Sequence number v. + with/without Protocol v.
0x0A..0E		30 byte SerienNr.
0x0F..0x1A	P12401408489	Softwareversion Oszilloskop
0x1B..0x27		SerienNr.
	14 byte	Proto 100: extra header
0x28		Trigger auto, ready, trig (4), scan (3), stop (2), play (✓)
0x29		Kanal Status (bit1 = Kanal 1, bit2 = Kanal 2) (✓)
0x2A 2B 2C 2D	float32	Trigger Position [µs]
0x2E	ASCII	's'=single 'a'=alternate trigger(✓)
0x2F	int8	Kanal, auf den getriggert wird: CH1, CH2, AC Line (4)
0x30	ASCII	'e'=edge, 'v'=video, 's'=slope, 'p'=pulse Trigger Typ (✓)
0x31 32 33 34	int32	Trigger level [pixel]
0x35	';'	Separator (✓)
		Header
0x36..0x38	CH1 bzw. CH2	Kanal (✓)
0x39 3A 3B 3C	0x39 + int32[0x39..3C] => CH2	Sprungbefehl zum nächsten Kanal (✓)
0x3D 3E 3F 40	0x02=16bit, 0x03=8bit	8bit oder 16bit Samples (✓)
0x41 42 43 44		Offset linker Bildschirmrand zum linken Datenrand
0x45 46 47 48	$t_{div} \cdot {}^{15,2}/_{\text{int32[0x45..48]}}$	Anzahl der Punkte, die ins Display passen (✓)
0x49 4A 4B 4C	= 0x4D 4E 4F 50	Anzahl aufgenommener Punkte (✓)
0x51 52 53 54	0x00..0x1F	Zeitauflösung t_{div} 5ns..100s. Beschriftet die t-Ache (✓)
0x55 56 57 58	int32	Offset in Y-Richtung
0x59 5A 5B 5C	0x00..0x0B	Spannungsbereich 2mV..10V (✓)
0x5D 5E 5F 60	$10^{\text{int32[0x5D..60]}}$	Eingestellte Dämpfung wegen Tastkopf (attenuation)
0x61 62 63 64	float32	Zeit zwischen zwei Punkten [µs] (scheint ✓)
0x65 66 67 68	int32	Sampling rate in [Hz] (?)
0x69 6A 6B 6C	int32	Sampling Cycle in [µs] (?)
0x6D 6E 6F 70	int32	Spannung zwischen zwei Punkten [mV]
ab 0x71	int8 oder int16	jedes Byte ein Sample

Tabelle 1: Bytes im Header

Inhalt Byte 0x59	Spannung/div
0x00	2mV
0x01	5mV
0x02	10mV
0x03	20mV
0x04	50mV
0x05	100mV
0x06	200mV
0x07	500mV
0x08	1V
0x09	2V
0x0A	5V
0x0B	10V

Tabelle 2: Spannungsbereiche, Header 0x59..5C

Inhalt 0x55..0x58	Dezimal	Offset in Kästchen	Offset Software
06 FF FF FF	-250	-10.00	
CE FF FF FF		-2.00	
E7 FF FF FF		-1.00	
FF FF FF FF	-1	-0.04	-0.03
00 00 00 00 (0)	0	0.00	
01 00 00 00	+1	0.04	0.03
19 00 00 00		1.00	
FA 00 00 00	+250	10.00	

Tabelle 3: Offset in Y-Richtung. Die Software scheint sehr seltsam zu runden. Header 0x55..58

Wert Byte 0x51	Zeit/div	Wert Byte 0x51	Zeit/div
0x00	5ns	0x10	1ms
0x01	10ns	0x11	2ms
0x02	20ns	0x12	5ms
0x03	50ns	0x13	10ms
0x04	100ns	0x14	20ms
0x05	200ns	0x15	50ms
0x06	500ns	0x16	100ms
0x07	1µs	0x17	200ms
0x08	2µs	0x18	500ms
0x09	5µs	0x19	1s
0x0A	10µs	0x1A	2s
0x0B	20µs	0x1B	5s
0x0C	50µs	0x1C	10s
0x0D	100µs	0x1D	20s
0x0E	200µs	0x1E	50s
0x0F	500µs	0x1F	100s

Tabelle 4: Zeit und Samplerate. Die Zeitskala kann am Oszi nachträglich geändert werden, was aber an den Daten nichts ändert. Lediglich die Darstellung wird aufgeblasen. Ändert man bei konstanten Daten die Zeitskala, variieren die Bytes 0x41, 0x42 und 0x51.