

# XY-Plotter

von Thomas Wagner

Im folgendem wird ein XY-Plotter beschrieben, der universell einsetzbar ist, jedoch für einen speziellen Einsatzzweck entworfen wurde: die Übertragung, Anzeige und Speicherung von Oszillogrammen eines digitalen Speicheroszilloskops Tektronix 2232 zu b.z.w. auf einem PC.

Dieses Oszilloskop kann je nach Option über eine RS232 oder eine GPIB Schnittstelle verfügen. Auf jeden Fall ist aber der so genannte „Auxiliary Connector“ vorhanden, der unter anderen einen X-Y Plotter Ausgang besitzt.

Diese Schnittstelle besteht aus zwei analogen Signalen (Spannungsbereich von ca. -2,5V bis +2,5V), welche die X- und Y- Koordinaten eines Bildpunktes repräsentieren und einem wahlweise Low- oder High- aktiven digitalen Signal, welches den Zustand „Stift aufsetzen“ (Pen down) darstellt.

## *Hardware*

Verwendet wurde das R8C-Mikrokontroller-Board. Vom Mikroprozessor wird die serielle Schnittstelle 1 zur Übertragung der Daten zum PC genutzt. Die Schnittstelle 1 wurde deshalb gewählt, um auch „Firmware“-Updates ohne zusätzlich Hardware durchführen zu können. Weiterhin werden AN8 und AN9 zur AD-Wandlung der beiden analogen Kanäle im 10-Bit Modus und P1\_2 als Digitaleingang verwendet.

Zur Pegelwandlung der seriellen Schnittstelle und zur Stromversorgung wurde die bekannte Variante mit Einzeltransistoren (Q1, Q2) und einem Spannungsregler 78L05 (IC2) gewählt, ähnlich wie im Elektorheft 12/2005 auf Seite 22 beschrieben. Die Analogeingänge sind mit einem spezielle Spannungsteiler (R5/R6, R7/R8) versehen, der es erlaubt, einen Spannungsbereich von -2,5V bis +2,5V abzudecken. Entsprechende Dioden (D2/D3, D4/D5) schützen die Eingänge vor zu hohen positiven oder negativen Spannungen. Dieser Eingangsbereich von 5V wird in 1024 diskrete Werte „übersetzt“.

## *Software $\mu P$*

Die Software für den R8C ist leicht überschaubar. In Abhängigkeit vom PenDown-Signal werden die Analogen Eingänge nach einem 8fach Oversampling in die diskreten Werte gewandelt. Das Datentelegramm besteht dabei aus einer Zeile, welche die durch Komma getrennten Werte für X- und Y- Koordinate enthält. Zusätzlich ist eine Zeile mit einem einzelnen Zeichen ‚d‘ für den Zustand „Pen down“ (Stift unten) und ‚u‘ für den Zustand „Pen up“ (Stift oben) möglich. Die Koordinaten während der „Stift oben“-Phase werden nicht übertragen.

Die Software enthält weiterhin noch Code zur Ausgabe der Daten auf ein zweizeiliges LC-Display. Dieser Code (und noch weitere Codeteile zu Debugzwecken) wurde nicht entfernt. Und würde im Falle einer Veröffentlichung auf jeden Fall noch einmal überarbeitet werden.

## *Software PC*

Für die Erstellung der PC-Software wurde die Sprache Python gewählt. Grund dafür ist die Einfachheit und Effektivität dieser Sprache (nach einer gewissen Einarbeitungszeit kann man mit dieser Sprache sehr effektiv programmieren) und die damit erreichbare Plattformunabhängigkeit. der damit erstellten Software (diese müsste unverändert auch unter Linux laufen).

Die Software wurde bewusst auf das nötigste beschränkt, um sie übersichtlich zu halten und die Überprüfung der oben genannten Behauptung möglich zu machen.

Die Software bestehe aus drei Einzelprogrammen:

- `plotter.py` - Anzeige des übertragenen Oszillogrammes in Echtzeit.
- `recorder.py` - Entgegennahme des Oszillogrammes und Speicherung in eine Datei.
- `simuplotter.py` – Anzeige gespeicherter Oszillogrammes

Als Beispiele habe ich vier gespeicherte Aufzeichnungen beigelegt. Diese zeigen das vom Oszilloskop bereitgestellte Probe Adjust Signal, ausgegeben mit den verschiedensten Plotgeschwindigkeiten (Einzustellen am Oszilloskop, mögliche Werte 1...10, 10 ist die höchste Geschwindigkeit)

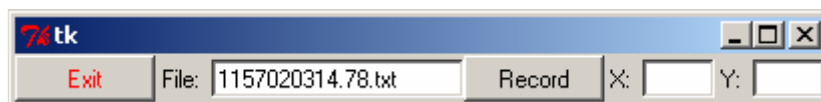
- `sample1.txt` - Plotgeschwindigkeit 10, mit Rasterlinien
- `sample2.txt` - Plotgeschwindigkeit 10, ohne Rasterlinien
- `sample3.txt` - Plotgeschwindigkeit 5, mit Rasterlinien
- `sample4.txt` - Plotgeschwindigkeit 1, mit Rasterlinien

Auch ohne die Schaltung aufbauen zu müssen, kann man sich so einen Überblick über die Möglichkeiten dieser Lösung verschaffen. Einfach `simuplotter.py` durch Doppelklick starten und in das Textfeld „File“ den gewünschten Dateinamen angeben und anschließend die Schaltfläche „Plot“ betätigen.

### *Anmerkung*

Wie bei `sample4.txt` gut zu sehen, ist es mir trotz voll geschirmten Aufbaus nicht gelungen, alle Störungen auf das Analogsignal (+/- ein Digit) zu eliminieren. Ich führe das auf den Aufbau auf einer Lochrasterleiterplatte zurück (Je ein PCB-Layout für bedrahtete und SMD-Bauelemente sind angefangen, jedoch nicht rechtzeitig fertig geworden). Im Schaltplan sind jedoch getrennte Analog- und Digital- Massen (GND, AGND) berücksichtigt worden. Auch wurde AVCC (VEE) getrennt betrachtet.

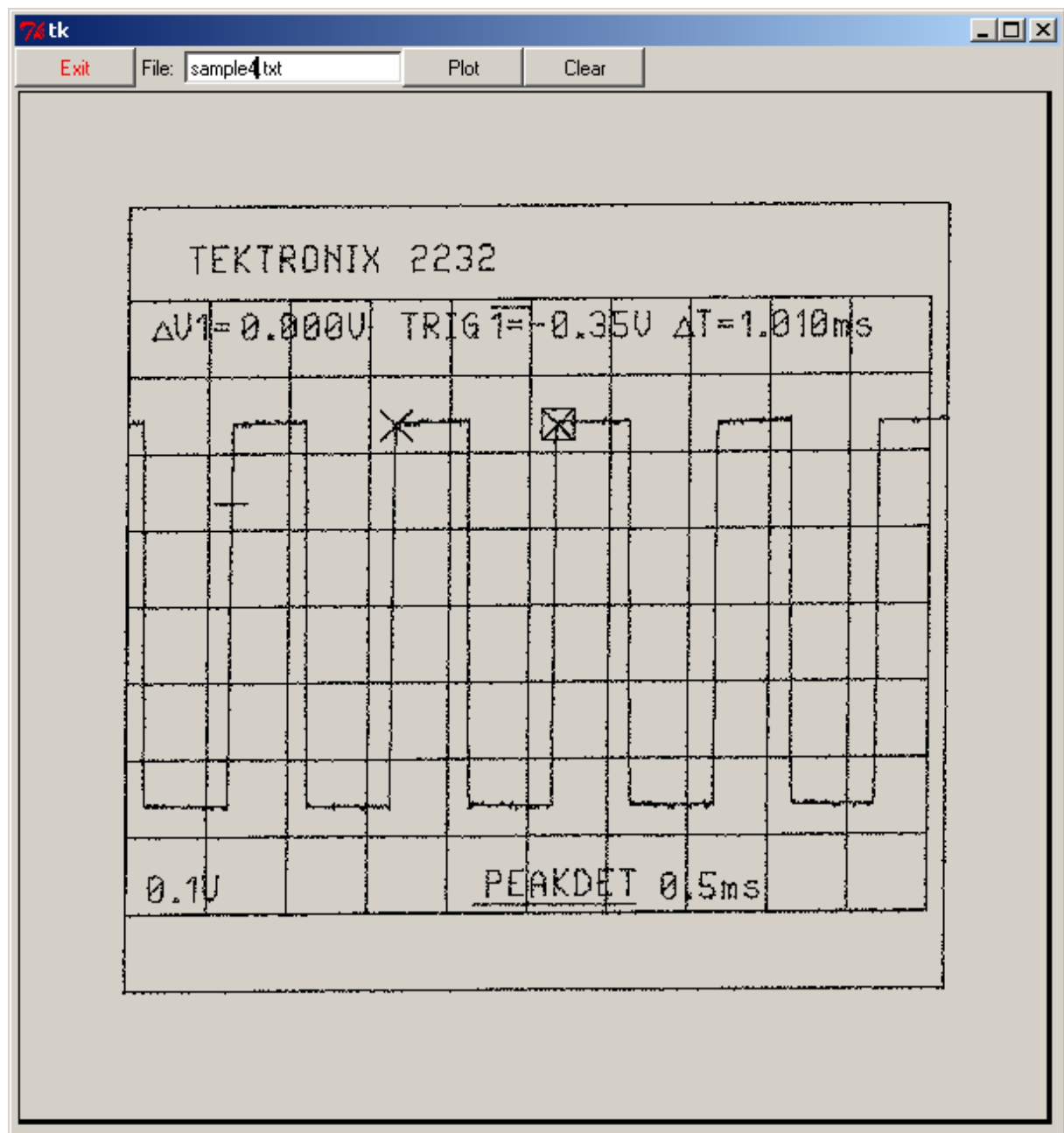
### *Screenshots*



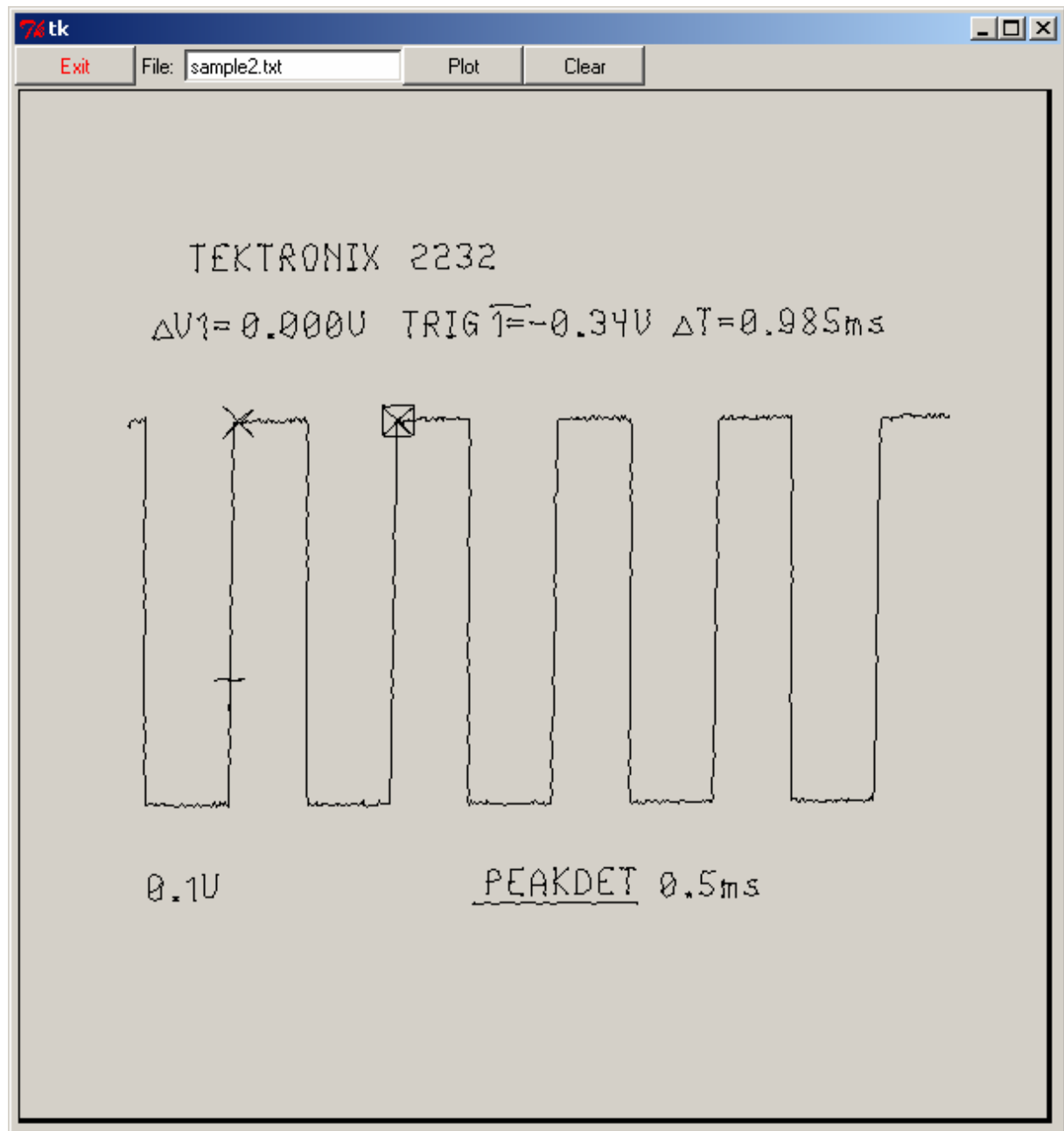
**Screenshot recorder.py mit vorgeschlagenem Dateinamen vor der Aufnahme**



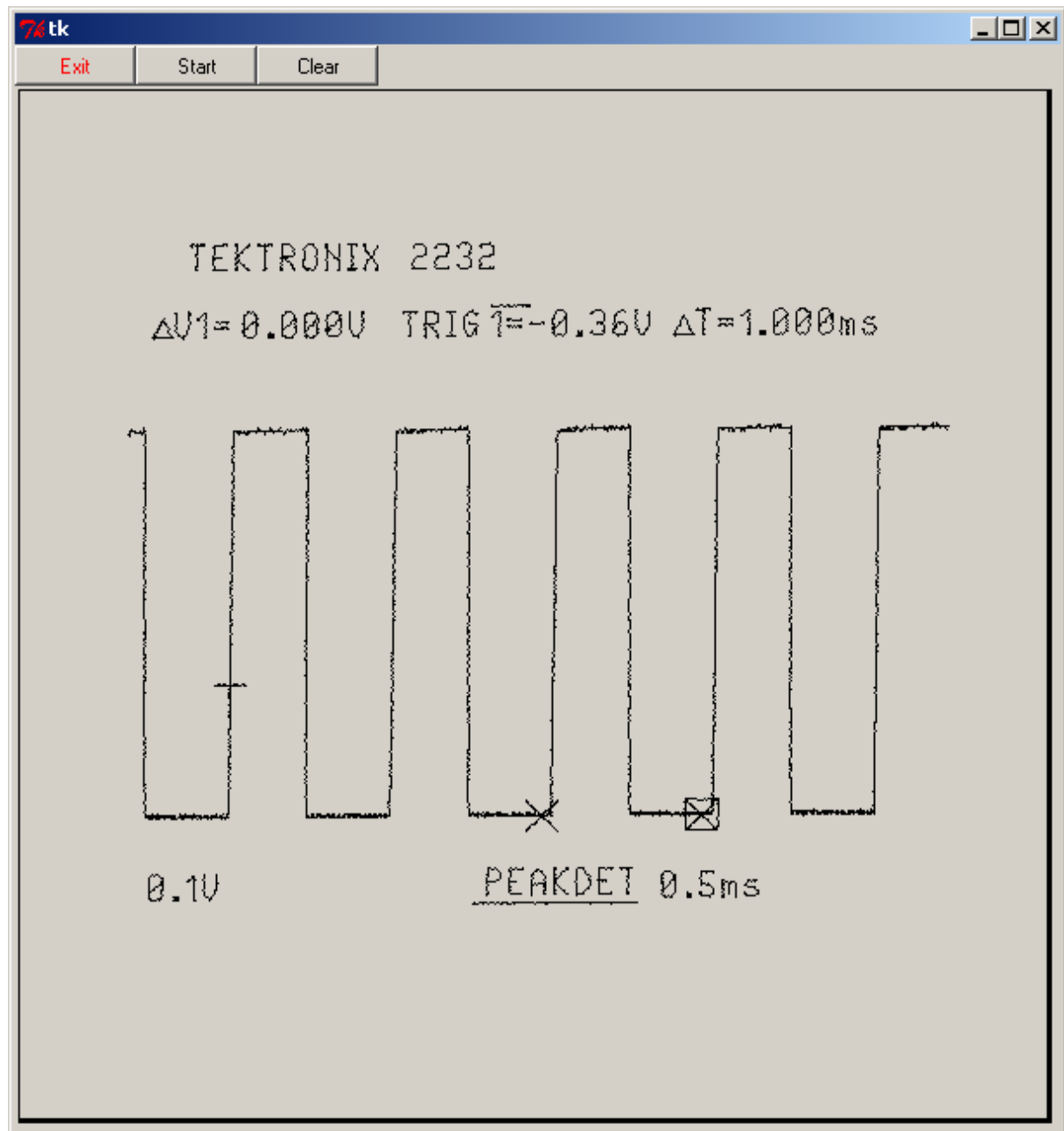
**Screenshot recorder.py mit selbstgewähltem Dateinamen während der Aufnahme**



Screenshot simuplotter.py mit sample4.txt (langsamste Plotgeschwindigkeit)



Screenshot simuplotter.py mit sample2.txt (schnellste Plotgeschwindigkeit)



## Screenshot plotter.py ohne Raster und mit Plotgeschwindigkeit 2

### *Python Softwareinstallation*

Benötigt werden folgenden Dateien, die teilweise zu installieren (\*.msi, \*.exe) sind:

- python-2.4.3.msi
- pywin32-208.win32-py2.4.exe
- pyserial-2.2.win32.exe
- mfc71.dll

Diese Dateien sind auf den einschlägigen Python-Websites frei erhältlich.