



Aqserver Dokumentation

Release 1a

Michael Taxis

29.01.2020

Inhaltsverzeichnis

Über Aqserver	2
Neu in dieser Version	3
Installation	4
Installation mit dem Quellcode	4
Installieren mit dem Windows Setup	5
Händische Installation	5
Konfigurationsdatei für Aqserver	6
Regeln in der Konfigurationsdatei	6
Einstellungen in der Konfigurationsdatei	6
Aqdata Einstellungen	6
Einstellungen für die Kommunikation	7
Verschiedene Einstellungen	8
Trigger Einstellungen	10
Debug Einstellungen	11
Werteeinstellungen	12
Parameter für die Kommandozeile	14
Benutzung des Programms	15
Starten aus einem Python Terminal	15
Benutzung der Windows Installer Version	16
Mehrere Instanzen starten	17
Probleme	18
Häufig gestellte Fragen FAQ:	18
Einführung in Kst2	19
Benutzung	19
Hinzufügen einer Anzeige des aktuellen Werts	24
Changelog	27
Licensing	30
Disclaimer of Warranty	30

Dieses Programm ist ein Aufzeichnungswerkzeug (Datenlogger) für Simatic S7 Steuerungen. Es wurde in Python unter Verwendung des Moduls `python-snap7` von <https://pypi.python.org/pypi/python-snap7/> sowie der `snap7` Kommunikationsbibliothek von <http://snap7.sourceforge.net> geschrieben. Die Datei Routinen beruhen auf Beispielcode aus dem Buch “Real-World Instrumentation with Python”.

Homepage <http://www.taxis-instruments.de>

Contact <aqserver at taxis-instruments dot de>

Author Michael Taxis <michael at taxis-instruments dot de>

Copyright of this document:

Copyright CC BY-SA 3.0

Copyright of the program:

Copyright LESSER GNU GENERAL PUBLIC LICENSE 3.0

Please note that soft- and hardware designations and brand names of their respective companies used in this document are generally subject to trademark protection or patent protection.

Über Aqserver

Mit diesem Programm können Werte aus einer Simatic S7 Steuerung in eine CSV-Datei aufgezeichnet werden. In Abhängigkeit von der Anzahl der aufzuzeichnenden Werte verändert sich die Zykluszeit der Aufzeichnung, mit steigender Anzahl der Werte steigt auch die Zykluszeit.

Es können Trigger Bedingungen definiert werden, die eine neue Aufzeichnungsdatei starten. Alte Aufzeichnungsdateien werden komprimiert und in einem definierbaren Verzeichnis abgelegt. Über Zeiten für Pre- und Posttrigger kann man eine Überlappung von alter und neuer Datei realisieren. Wenn keine Triggerbedingung definiert ist, kann auch von Hand getriggert werden.

In einer Konfigurationsdatei (Standard: aqserver.cfg) werden die Werte für Kommunikation, die aufzuzeichnenden S7 Variablen und noch andere Parameter eingestellt. Öffne die Datei in einem beliebigen Editor und lies die Kommentare, um zu erfahren, wie die Parameter eingestellt werden müssen oder lies das Kapitel "Konfiguration" in diesem Handbuch.

Werte werden in einer CSV-Datei im Unterverzeichnis "data" gespeichert. Der Dateiname kann in der Konfigurationsdatei eingestellt werden. Wenn eine definierte Triggerbedingung erfüllt ist, beginnt eine neue Aufzeichnungsdatei. Nach einem Trigger oder wenn das Programm beendet wird, wird die Datendatei komprimiert und in einem Verzeichnis gespeichert, welches ebenfalls in der Konfigurationsdatei eingestellt werden kann. Es ist möglich mehrere Instanzen, mit einer jeweils anderen Konfigurationsdatei, zu starten, um Werte aus unterschiedlichen Steuerungen gleichzeitig aufzuzeichnen.

Bitte beachten, dass sich das Programm noch in der Entwicklung befindet. Falls es Probleme bei der Benutzung gibt, lies das Kapitel "Probleme" in diesem Handbuch. Dass ein Problem mit dem Programm in der Steuerung entsteht, ist jedoch nicht zu befürchten.

Für die Visualisierung der aufgezeichneten Daten benutze das Programm Kst2 von <https://kst-plot.kde.org>. Die Datendatei kann damit online während der Aufzeichnung gelesen und geplottet werden. Dekomprimierte Archive können ebenfalls angezeigt werden. Natürlich können die gespeicherten CSV-Dateien auch mit LibreOffice Calc, MS Excel und anderen ähnlichen Programmen geöffnet und angezeigt werden.

Neu in dieser Version

28 März 2017

- Neue Funktion Einheiten: Es wird eine Zeile für die Einheiten der Werte in der Datendatei angelegt
- Neue Funktion demo mode: die Daten werden über eine Zufallsfunktion erzeugt, ohne Kommunikation zu einer Steuerung. DEMO wird in Aqserver in der Überschrift angezeigt
- Neue Funktion booloffset: boole'sche Werte innerhalb eines Bytes können mit einem Offset versehen werden. Dies hat den Vorteil, dass die Werte in Kst in einem Plot angezeigt werden können ohne sich zu überlappen.
- Für die Werte in der Konfigurationsdatei muss nun auf Groß- und Kleinschreibung geachtet werden. ACHTUNG: Die Werte IP, RACK and SLOT müssen in alten Konfigurationsdateien auf Kleinschreibung geändert werden (ip, rack,slot). Das wurde geändert, da die Wertenamen immer nur in Kleinbuchstaben angezeigt wurden.

Für eine komplette Änderungshistorie siehe Changelog!

Installation

Das Programm kann unter <https://github.com/franzhefekranz/aqserver> heruntergeladen werden.

Das Programm wurde mit Python 2.7.10 unter Windows 7 - 32 bit geschrieben.

Installation mit dem Quellcode

Wenn direkt mit dem Quellcode gestartet werden soll, muss eine lauffähige Python Umgebung installiert sein. Empfehlenswert dafür sind für ein Windows Betriebssystem z.B. die Python (x,y) Distribution von <https://python-xy.github.io/> oder Winpython von <https://winpython.github.io/> . Unter Linux oder MacOS sollte Python bereits als Teil des Betriebssystems installiert sein.

Im Programm werden folgende Bibliotheken zumindest teilweise verwendet (nicht in bestimmter Reihenfolge):

- time
- sys
- os
- shutil
- ctypes
- struct
- logging
- collections
- win32com
- python-snap7
- socket
- datetime
- gzip
- bz2
- binascii
- ConfigParser
- distutils

Diese Bibliotheken können, wenn nicht bereits vorhanden, mit pip installiert werden.

Python-snap7 ist ein Python Wrapper für die snap7 Bibliothek von <http://snap7.sourceforge.net/> . Bitte sicherstellen, dass auch diese Bibliothek installiert ist.

Installieren mit dem Windows Setup

Für die Installation kann auch das im Repository im Unterverzeichnis nsi verfügbare Installationsprogramm verwendet werden. Starte das setup.exe als Administrator und folge den Anweisungen des Programms. Eine Verknüpfung zum Programm wird im Startmenü installiert. Das Programm selbst wird im Verzeichnis "C:\Program Files (x86)\Aqserver\" installiert. Zusätzlich wird das Verzeichnis "Aqserver" mit den Unterverzeichnissen "help", "log" und "data" im Verzeichnis "Eigene Dokumente" des aktuellen Benutzers angelegt. Im "Aqserver" Verzeichnis findest du folgende Dateien:

- test.bat: eine Windows Command/Batch Datei, welche Aqserver mit einer Testkonfiguration startet. Diese Datei kann kopiert und bearbeitet werden, um sie an die entsprechenden Anforderungen anzupassen. Ich benutze diese Datei, um Aqserver aus dem Windows Explorer zu starten oder um mehrere Instanzen des Programms zu starten (eine Batch Datei pro Instanz).
- test.cfg: ein Beispiel für eine Konfigurationsdatei, die an die persönlichen Anforderungen angepasst werden muss. (Es ist eine gute Idee, vorher eine Kopie der Datei zu erstellen..)

Um Aqserver zu deinstallieren, starte den Uninstaller aus dem Startmenü oder deinstalliere aus der Systemsteuerung.

Händische Installation

Um Aqserver auf einem Windows Betriebssystem ohne Python Umgebung zu installieren, benutze ein Verzeichnis deiner Wahl und kopiere die Dateien aqserver.exe, aqserver.cfg, aqserver.chm und aqserver.pdf in dieses Verzeichnis. Kopiere "snap7.dll" in dein "Windows\system32" Verzeichnis. Lösche diese Dateien, um zu deinstallieren.

Konfigurationsdatei für Aqserver

Zum Starten des Programms wird eine Konfigurationsdatei benötigt, in der alle notwendigen Parameter zur Aufzeichnung hinterlegt sind. Beim Start wird die Konfigurationsdatei eingelesen, die Parameter auf Korrektheit überprüft und danach die Aufzeichnung gestartet.

Regeln in der Konfigurationsdatei

Benutze eine Editor wie z.B. Notepad um die Konfigurationsdatei zu bearbeiten.

Bemerkung: Wenn die Datei gespeichert wird, vergewissere dich, dass im ANSI Format gespeichert wird.

In der Konfigurationsdatei starten Kommentare mit “#”:

“#” in der ersten Spalte definieren die Zeile als Kommentar, dies kann auch zum Auskommentieren von Einstellungen benutzt werden. Es muss jedoch immer eine gültige Einstellung vorhanden sein.

Benutze “;” um nach einer Einstellung in derselben Zeile einen Kommentar einzufügen

Einstellungen in der Konfigurationsdatei

Aqdata Einstellungen

Zur Information können verschiedene Texte in der Konfigurationsdatei hinterlegt werden, diese werden im Kopf der Datendatei eingefügt. Es kann ein Feldname ab der ersten Spalte gefolgt von “=” und dem gewünschten Text, hinzugefügt werden.

Bemerkung: Alle Einstellungen werden in der Datendatei eingefügt. Wird ein entpacktes Archiv im Editor geöffnet, können die Einstellungen herauskopiert werden, um eine neue Konfigurationsdatei zu erstellen, falls die Originaldatei verloren gegangen ist.

```
#####
# aqdata settings
#####
[aqdata]
place = somewhere

creator = Michael Taxis

machine = MyMachine

order = 123456

yourText = type whatever you desire
```

```
remarks = ; use tab for multiline values
      trials with double spreader, to get rid of married rolls
      trials for SF production
```

Einstellungen für die Kommunikation

Zur Kommunikation mit der S7 werden folgende Parameter benötigt:

- demo: Aqserver kann in Demomodus laufen die Werte werden zufällig erzeugt, ohne Kommunikation mit einer Steuerung
- IP address: IP Adresse der Steuerung die erreicht werden soll. Bestätige mit einem PING vor dem Start!
- rack number: ist normalerweise immer 0
- slot number: das ist normalerweise die Nummer links neben der CPU in der Hardware Konfiguration, im Bild unten die 3.
- maximum connection attempts: Das Programm versucht sich mit der Steuerung zu verbinden. Falls nach dieser Anzahl Verbindungsversuche keine Verbindung zustande kommt, wird das Programm beendet. Prüfe dann die Konfiguration und deine Verbindung (Kabel, PING, evtl. Firewall...). Wird dieser Wert auf 0 gestellt, wird in einer Endlosschleife versucht die Verbindung aufzubauen und das Programm bricht nicht ab. Dies kann hilfreich sein, wenn das Programm unbeobachtet z.B. über Nacht aufzeichnen soll. Sollte in diesem Fall keine Verbindung zustande kommen, kann das Programm mit CTRL + C bzw. STRG + C abgebrochen werden. Wird die Kommunikation während der Aufzeichnung unterbrochen, versucht das Programm die Verbindung wieder aufzubauen. Es können aber natürlich keine Werte aufgezeichnet werden, wenn die Verbindung unterbrochen ist.

(0) UR2ALU	
1	PS 405 10A
3	CPU 417-5 H PN/DP
X2	DP
X1	MPI/DP
IF1	
IF2	
X5	PN-IO
X5 P1 R	Port 1
X5 P2 R	Port 2
5	CP 443-1
X1	PN-IO
X1 P1 R	Port 1
X1 P2 R	Port 2
6	CP 443-1(1)
X1	PN-IO-1-1
X1 P1 R	Port 1
X1 P2 R	Port 2
7	CP 443-5 Ext
8	
9	

```
#####
# communication settings
#####
```

```
[communication]

# setting up communication parameters to S7-PLC
# ATTENTION: config values are case sensitive
# ip, rack and slot must be in lowercase letters!

# demo switch
# if 1 communication will be ignored and values will be randomly created
# if 0 then aqserver tries to connect to a plc
demo = 0

# IP address
# you can leave several settings , just comment with a leading "#"
#ip = 192.168.1.107
ip = 172.16.13.174

# rack number, see HW Config
rack = 0

# slot number of CPU, see HW Config
slot = 3

# maximum attempts for connection, 0 is trying forever
maxattempts = 10
```

Bemerkung: Aqserver kann auch mit PLCSIM und NetToPlcsim von <http://nettoplcsim.sourceforge.net/> getestet werden.

Verschiedene Einstellungen

Wir brauchen ein paar grundsätzliche Einstellungen für Aqserver:

- delimiter: Dies ist das Trennzeichen, das die Werte in der Datendatei voneinander trennt, falls nichts anders dagegen spricht benutze ";". Bitte nicht das Dezimaltrennzeichen des Betriebssystems verwenden (also "." oder ", " NICHT verwenden)!
- datafileprefix: Es kann ein Name definiert werden, der im Namen der Archivdatei verwendet wird. Dies ist ein Namensvorsatz, da der Dateiname auch noch einen Zeitstempel enthält, z.B.: MyProject20150804_173035.csv.gz
- datafile ist der Dateiname (ohne Erweiterung) der Datendatei für die aktuelle Aufzeichnung. Die Datei ist eine CSV-Datei. Falls mehrere Instanzen für unterschiedliche Steuerungen aufgezeichnet werden sollen, muss für jede Instanz eine anderer Dateiname in den separaten Konfigurationsdateien verwendet werden!
- autostart: definiert ob die Aufzeichnung direkt mit Programmstart anläuft oder ob auf ein manuelles Startsignal (Taste "S") gewartet wird.
- datapath: hier stellen wir ein, wo die Archive abgelegt werden.
- usedir: definiert ob wir eine Verzeichnisstruktur (\JJJ\MM\TT\) zur Ablage der Archive verwenden.
- scantime: Zykluszeit in [ms], minimale Zykluszeit ist auf 20 ms im Programm begrenzt. Die Zeit ist nur eine ungefähre Angabe und ist auch von der Anzahl der zu lesenden Variablen abhängig. Der Wert kann verwendet werden um die Dateigröße zu reduzieren. Je kleiner der Wert umso größer wird die Datei. Wird der Wert auf "0" gesetzt, werden die Daten so schnell wie möglich gelesen (Vorsicht: große Datei!). Je nach Anzahl der Variablen können Zykluszeiten bis ~10 ms erreicht werden.

- maxrecords: Diese Zahl definiert die maximale Anzahl der Aufzeichnungen in einer Datei. Damit kann die Größe der Datei begrenzt werden. In Abhängigkeit von der Anzahl der Variablen sollte geprüft werden, welcher Wert hier anwendbar ist.
- booloffset: wenn dieser Wert auf 1 gesetzt wird, wird zu den Bits in einem Byte ein Offset addiert, wie folgt:

$$\text{Wert} + \text{Bit Nummer} * 2$$

Dadurch können die Bits in Kst in einem Plot angezeigt werden ohne zu überlappen

bit	true	false
0	1	0
1	3	2
2	5	4
3	7	6
4	9	8
5	11	10
6	13	12
7	15	14

Ist der Wert 0 wird nur der boole'sche Wert gespeichert (1 für true, 0 für false).

```
#####
# miscellaneous settings
#####
[misc]

# miscellaneous values for setting up the acquisition server
# value delimiter in storage file
delimiter = ;

# prefix of data file name, e.g. a customer/project name or whatever
datafileprefix = MyProject

# data file name for actual data recording, without extension!
# e.g. if you use "filename", actual name will be "filename.csv"
datafile = recording

# autostart: when program is started decide whether acquisition is started(1)
# immediately or wait for start signal (0)
autostart = 0

# path for data files, use "\" for directory separation, with "\" at the end !
# e.g. datapath = D:\mydata\
datapath = F:\aqdata\MyProject\

# if 1 use directory structure datapath\yyyyy\MM\dd otherwise use only datapath
usedir = 1

# scantime in milliseconds [ms]
# if you just put 0 program will scan as fast as possible
# this will produce rather large data files!
# depending on number of values this value is just a hint ;-)
scantime = 100

# maximum number of records
# to avoid too big data files, a new one will be started after this number
# of recordings
maxrecords = 50000

# switch for offset of boolean values
# if 1 then boolean values in a byte (see values settings) will be offset by 2 as follows:
#
```

```
# value + bit number * 2
#
# bit | true | false
# ----+-----+-----
# 0 | 1 | 0
# ----+-----+-----
# 1 | 3 | 2
# ----+-----+-----
# 2 | 5 | 4
# ----+-----+-----
# 3 | 7 | 6
# ----+-----+-----
# 4 | 9 | 8
# ----+-----+-----
# 5 | 11 | 10
# ----+-----+-----
# 6 | 13 | 12
# ----+-----+-----
# 7 | 15 | 14
# if booloffset is 0 then only the boolean value (1 for true, 0 for false) will be stored
booloffset = 1
```

Trigger Einstellungen

Die Trigger Einstellungen werden benutzt, um die Bedingung für den Start einer neuen Datendatei zu definieren. Ein Trigger kann auch per Hand (Taste "T") ausgelöst werden. Eine Triggerbedingung wird durch die folgenden 3 Einstellungen definiert:

- **trgsignal**: dies ist der Name des Signals aus den Werteeinstellungen (s.u.), auf welches getriggert werden soll. Kopiere den Namen aus den Werteeinstellungen.
- **trgcondition**: dies ist die Vergleichsbedingung für das Triggersignal mit dem Triggerwert. Wenn z.B. die Bedingung "==" ist, dann wird getriggert wenn das Signal gleich dem Wert ist.
- **trgvalue**: dies ist ein Wert bzw. eine Konstante mit dem das Triggersignal verglichen wird, um den Trigger auszulösen.

Desweiteren werden 2 Einstellungen benötigt, wenn alte und neue Datendatei sich überlappen sollen:

- **pretrg**: Zeit in [s] die in der neuen Datei VOR dem Trigger aufgezeichnet wird. Basiert auf der Einstellung scantime, pretrg geteilt durch scantime ergibt die Anzahl der Datensätze.
- **posttrg**: Zeit in [s] die in der alten Datei NACH dem Trigger aufgezeichnet wird. Basiert auf der Einstellung scantime, posttrg geteilt durch scantime ergibt die Anzahl der Datensätze.

```
#####
# trigger settings
#####
# when trigger condition is matched, then we close the old file after
# post-trigger time and start the new file and copy pre-trigger time
# and post-trigger recordings to new file
#   # condition is, with example:
#   trgsignal trgcondition trgvalue
#   rewind diameter [mm] = 0
#
[trigger]

# trigger signal, copy the name of the signal in [values] section,
# that you want to use as trigger signal
trgsignal = rewind diameter [mm]

# trigger condition, use >,>=,==, <=,<,! = as condition
# when conditon is matched, then we close the old file and start a new one
```

```
# trgcondition = >
# trgcondition = >=
# trgcondition = ==
trgcondition = <=
# trgcondition = <
# trgcondition = !=

# trigger value, with this value we compare the trigger signal
trgvalue = 0

# pre-trigger time in seconds [s]
# will still add pre-trigger/scantime lines to old file after trigger event
# e.g. pre-trigger is 60 seconds and scantime is 100 ms, then 600 lines
# will be recorded after trigger event
pretrg = 30

#post-trigger time in seconds [s]
# will copy last post-trigger/scantime lines from old to new file
# e.g. post-trigger is 60 seconds and scantime is 100 ms, then 600 lines will
# be copied after trigger event
posttrg = 30
```

Debug Einstellungen

Die Debug Einstellungen definieren wie und ob der Programmablauf zur Fehlersuche geloggt wird.

Dazu müssen wir einen Debug Level einstellen, der festlegt, was geloggt wird.

Mit dem Wert "0" wird das Logging deaktiviert, mit Level "1" wird alles geloggt. Bitte beachten, dass bei jedem Neustart des Programms, das Verzeichnis mit den Log-Dateien geleert wird, so dass nur die jeweils neueste Log-Datei erhalten bleibt.

Der Parameter logfile definiert den Namen der Log-Datei, ohne Erweiterung. Erweiterung ist immer ".log".

Wird der Parameter logts auf 1 gesetzt, wird jedesmal, wenn das Programm gestartet wird, eine neue Log-Datei angelegt. Ist logts = 0, dann wird an eine bestehende Log-datei angehängt.

```
#####
# debug settings
#####
[debug]

# debug level
# set logging level to debug, write program actions
# to logfile
# 0 - no logging
# 1 - log INFO messages (default setting)
# 2 - log WARNING messages
# 3 - log DEBUG messages
# 4 - log ERROR messages
# 5 - log CRITICAL messages
# 6 - log EXCEPTION messages
dbglevel = 2

# name of logfile, without extension. Extension will be added as ".log"
logfile = aqserver

# add timestamp to logfile name 1 = yes, 0 = no
# if set to 1 a timestamp will be added to the logfile name. pls. note that a
# new logfile will be created, every time you start the server,
# when dbglevel is > 0
logts = 1
```

Werteeinstellungen

In den Werteeinstellungen listen wir die Werte bzw. Steuerungsvariablen auf, die aufgezeichnet werden sollen. Eine Wertedefinition besteht aus einem Wertennamen gefolgt von einem Gleichheitszeichen und der Adresse der Variablen, die gelesen werden soll. Im Namen kann innerhalb von eckigen Klammern [] die Einheit des Wertes angegeben werden. Die Einheit wird aus dem Namen extrahiert und in eine extra Zeile in der Datendatei geschrieben.

Die Definition der Adresse folgt allerdings nicht der S7 Syntax, da unsere Syntax die Adresse, das Format (bool, int, float) und die Größe der Variablen in bytes (bool, byte, word, double word) in einem Parameter enthält. Dies ist bei der S7 Syntax nicht eindeutig, da z.B. ein Doppelwort sowohl das Format DINT als auch REAL haben kann. Die Syntax ist im Detail unten beschrieben.

Die Definition von booleschen Variablen ist etwas speziell, da die kleinste Größe die gelesen werden kann, ein Byte ist. Deshalb wird ein byte in 8 einzelne boolesche Variablen aufgesplittet. Um festzulegen welches bool von diesen 8 aufgezeichnet werden soll, müssen die Namen der einzelnen booleschen Variablen mmit einem „;“ getrennt werden (in einer Zeile). Wird dann der Text zwischen 2 Kommas weggelassen, wird diese Variable zwar gelesen, jedoch nicht in die Datendatei geschrieben.

Zum besseren Verständnis folgend eine Tabelle, wo wir die S7 Syntax mit unserer Syntax vergleichen:

S7 Syntax	Format	Aqserver syntax
DB4615.DBD714	REAL	DB4615.DF714
ED 4	DINT	ED4
DB4615.DBD714	DINT	DB4615.DD14
AW 4	INT	AW 4
DB4615.DBB6	INT	DB4615.DB6
DB4615.DBX6.1	BOOL	DB4615.DX6 (byte!)

```
#####
# value settings
#####
# here we define the S7 variables we want to read, and their formats
# here we define the S7 variables that we want to observe
# use following syntax:
#
##### how to define the names: #####
# use config value name with [ ] - brackets to define the unit of the value
# units will be separated from the name and put into the datafile
#
# boolean values:
# For boolean values (see format X above) a complete byte is read and then
# split into 8 bits
# To define names for the single bits use ',' to separate the names, e.g.:
#
# bit0,bit1,bit2,bit3,bit4,bit5,bit6,bit7 = DB1234.DX5
# Ventil 1, Ventil 2, Ventil 3, Ventil 4, Res1, Res2, Res3, Res4 = DB1234.DX5
#
# If you do not want all the bits, leave the name empty e.g.:
#
# bit0,,bit2,,,, = DB1234.DX5
#
# This reads only bit0 and bit2
#
##### how to define the values: #####
# (S7 variable and format)
# DBn.AFn.x
#
# where:
# - DB is for data blocks or omitted if other area
# - n is DB number or omitted if other area
#
```



```

# - . only when data, omitted otherwise
#
# - A is area
#   - D for data
#   - M or F for flags
#   - E or I for inputs
#   - A or Q for outputs
#   - T for timers
#   - Z or C for counters
#
# - F is format:
#
#   - X - for BYTE in BOOL format, followed by byte address:
#   - n is whole number for byte address
#     (attention to address ranges of PLC)
#     will always be split in 8 single booleans
#
#   - B - for BYTE in int format, followed by byte address
#   - n is whole number for byte address
#     (attention to address ranges of PLC)
#
#   - W - for WORD, followed by byte address
#   - n is whole number for byte address
#     (attention to address ranges of PLC)
#
#   - D - for DOUBLE WORD, followed by byte address
#   - n is whole number for byte address
#     (attention to address ranges of PLC)
#
#   - F - for DOUBLE WORD in REAL format, followed by byte address
#   - n is whole number for byte address
#     (attention to address ranges of PLC)
#
#
[values]
rewind diameter [mm] = DB4615.DF714
webspeed actual [m/min] = DB4615.DF574
vibration left core chuck [mm/s] = DB4614.DF560
vibration right core chuck [mm/s] = DB4614.DF564
vibration rider roll [mm/s] = DB4614.DF568
#Klemmventil UM1,Klemmventil UM2,Klemmventil UM3,Klemmventil UM4,,, = DB4614.DX564

```

Parameter für die Kommandozeile

Aqserver hat folgende Parameter für die Kommandozeile:

(starte 'python aqserver.py -help', um den Text unten anzuzeigen)

```
>python aqserver.py --help
usage: aqserver.py [-h] [-c CONFIG]
```

Siemens S7 data acquisition server

optional arguments:

```
-h, --help            show this help message and exit
-c CONFIG, --config CONFIG
                        specify name of configfile, defaults to aqserver.cfg
```

This program was written using the python-snap7 module and the snap7 library.
A configfile must exist (default file is aqserver.cfg) that defines
communication and other parameters and of course the S7 variables to scan.
Open this file in your preferred editor and read the comments on how to
configure.

Values will be stored to csv-file in subdirectory data. Filename can be
specified in configfile. Once a configurable trigger is raised, a new datafile
begins.

After trigger or when program is stopped, the data file will be compressed and
stored to a path, that can also be specified in the config file.

Benutzung des Programms

Starten aus einem Python Terminal

In einem Python Terminal kann das Programm wie folgt gestartet werden:

```
>python aqserver.py
```

oder

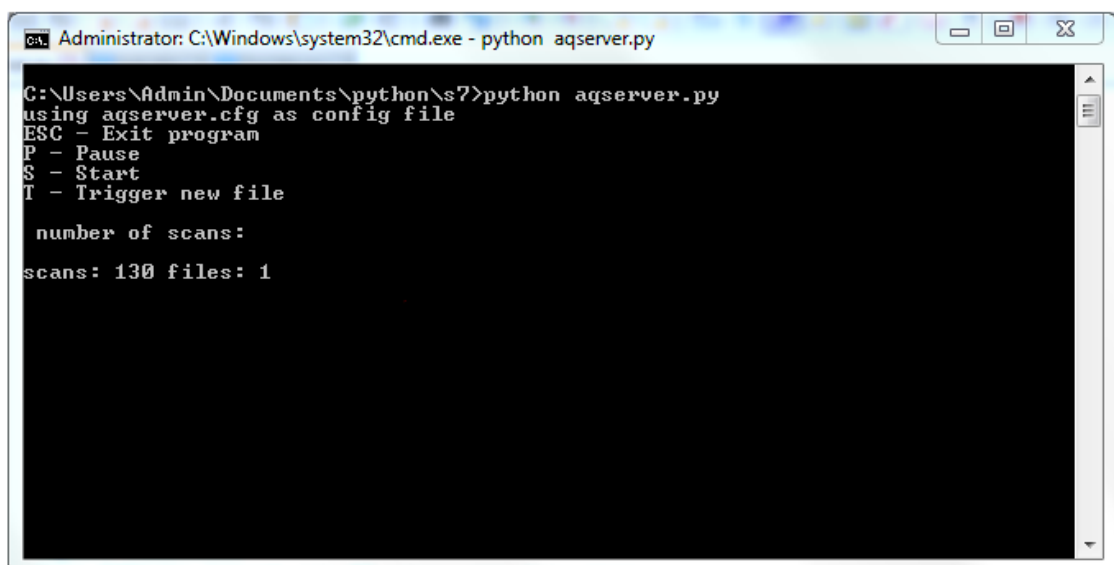
```
>aqserver.py
```

Wird das Programm ohne angehängte Parameter gestartet, wird der Standardname für die Konfigurationsdatei "aqserver.cfg" verwendet.

Es ist eine gute Idee, für jede Installation, an der aufgezeichnet werden soll, eine separate Konfigurationsdatei zu erstellen. In diesem Fall, wenn nicht die Standard-Konfigurationsdatei verwendet wird, wird so gestartet:

```
>aqserver.py -c my3rd.cfg
```

oder eben wie der Name der verwendeten Konfigurationsdatei ist. Es ist empfehlenswert die Erweiterung ".cfg" für die Konfigurationsdatei zu benutzen. Prinzipiell funktioniert es auch ohne die Erweiterung, benutze immer den kompletten Namen, inklusive der Erweiterung.



```
Administrator: C:\Windows\system32\cmd.exe - python aqserver.py
C:\Users\Admin\Documents\python\s7>python aqserver.py
using aqserver.cfg as config file
ESC - Exit program
P - Pause
S - Start
T - Trigger new file
number of scans:
scans: 130 files: 1
```

Wie man dem obigen Bild entnehmen kann, zeigt das Programm ein paar Optionen sowie die Anzahl der Datensätze in der aktuellen Datendatei und die Anzahl der bis jetzt aufgezeichneten Archivdateien an. Die Optionen sind:

- ESC - Programm beenden : Mit Drücken der "ESC" Taste wird die aktuelle Datendatei gespeichert und das Programm wird beendet.

- P - Pause: die Aufzeichnung kann mit der Taste “P” unterbrochen werden. D.h. es wird nicht von der Steuerung gelesen und auch nichts in die Datendatei geschrieben. Dies kann nützlich sein, wenn z.B. die Produktion einer Maschine gestoppt wurde, die aktuelle Datendatei jedoch weiter verwendet werden soll.
- S - Start: Wenn die Aufzeichnung mit “P” unterbrochen wurde oder die Einstellung “autostart” auf 0 steht, dann kann die Aufzeichnung mit der Taste “S” gestartet / fortgesetzt werden.
- T - Neue Datei triggern: Mit der Taste “T” kann ein Trigger für eine neue Datendatei von Hand ausgelöst werden. Die aktuelle Datendatei wird archiviert und eine neue Datendatei wird gestartet. Dies wird mit der Anzahl der aufgezeichneten Dateien angezeigt, außerdem startet die Anzahl der Datensätze wieder bei 1.

Bemerkung: Für die Tasten muss natürlich nicht auf Groß- oder Kleinschreibung geachtet werden.

Benutzung der Windows Installer Version

Wenn Aqserver mit dem Windows Setup Programm installiert wurde, dann ist die Benutzung etwas anders als oben beschrieben. Es kann auch sein, dass keine Python Umgebung installiert ist. Das Setup Programm installiert die Aqserver.exe nach “C:\Program Files (x86)\Aqserver\”. Eine Beispielkonfiguration und eine Batch Datei wird nach “..\My Documents\Aqserver\” kopiert. Wenn die Konfigurationsdatei bearbeitet und fertiggestellt wurde, sollte eine Kopie der Batch Datei bearbeitet werden, so dass Aqserver mit deiner eigenen Konfigurationsdatei gestartet wird. Inhalt der Batch Datei:

```
"C:\Program Files (x86)\Aqserver\aqserver.exe" -c "path_to\test.cfg"
pause
```

Passe Pfad und Name deiner Konfigurationsdatei an. Danach kann Aqserver mit einem Doppelklick auf die Batch Datei aus dem Windows Explorer gestartet werden.

Mehrere Instanzen starten

Wenn von 2 (oder mehr..) Steuerungen gelesen werden soll, müssen mehrere Instanzen von Aqserver gestartet werden:

Erzeuge eine zweite Konfigurationsdatei, die die entsprechenden Parameter für die zweite Verbindung enthalten.

Bemerkung: Darin muss sich der Name der Datendatei vom Namen in der anderen Instanz unterscheiden!

- benutze einen anderen Namen für die Konfigurationsdatei
- stelle darin einen anderen Namen für die Datendatei ein
- starte Aqserver aus einer Kommandozeile und benutze dazu den -c Parameter
- z.B. unter Windows:

```
>python aqserver.py -c my_new_config_file.cfg
```

Wenn Aqserver mit dem Windows Setup Programm installiert wurde, kann die Batch Datei im ”..\My Documents\Aqserver\” Verzeichnis verwendet werden , um Aqserver zu starten (siehe dazu das Kapitel “Installation” in diesem Handbuch). Erstelle eine Batch Datei für jede zu startende Instanz, welche die korrekten Einstellungen für die entsprechende Konfigurationsdatei enthält.

Probleme

Wenn sich das Programm unerwartet beendet oder abstürzt, dann sollte das Debugging eingeschaltet werden. Standardmäßig ist das Debugging ausgeschaltet. Um das Programm mit eingeschaltetem Debugging zu starten, öffne die Konfigurationsdatei und setze den Parameter “dbglevel” auf einen Wert > 0 (siehe unten). Der Wert 1 für INFO ist dabei die redseligste Einstellung. Bei Problemen sollte mindestens auf 3 eingestellt werden.

```
#####  
# debug settings  
#####  
[debug]  
  
# debug level  
# set logging level to debug, write program actions  
# to logfile  
# 0 - no logging  
# 1 - log INFO messages (default setting)  
# 2 - log WARNING messages  
# 3 - log DEBUG messages  
# 4 - log ERROR messages  
# 5 - log CRITICAL messages  
# 6 - log EXCEPTION messages  
dbglevel = 0
```

Das Programm sollte danach noch einmal gestartet werden. Nach Beendigung muss die Log-Datei im Unterverzeichnis “log” geprüft werden. Darin sollten detailliertere Angaben, warum das Programm nicht ordnungsgemäß funktioniert, zu finden sein. Korrigiere (vermutlich) deine Konfiguration entsprechend und versuche es erneut.

Probleme können verursacht werden durch:

- die Steuerung kann mit den eingestellten Parametern nicht erreicht werden
- Werte in deiner Liste sind in der Steuerung nicht vorhanden
- Schreibfehler in den Konfigurationswerten

Wenn all dies nicht zum Erfolg führt, schicke eine Kopie deiner Log- und der Konfigurationsdatei an meine Mailadresse (aqserver at taxis-instruments dot de).

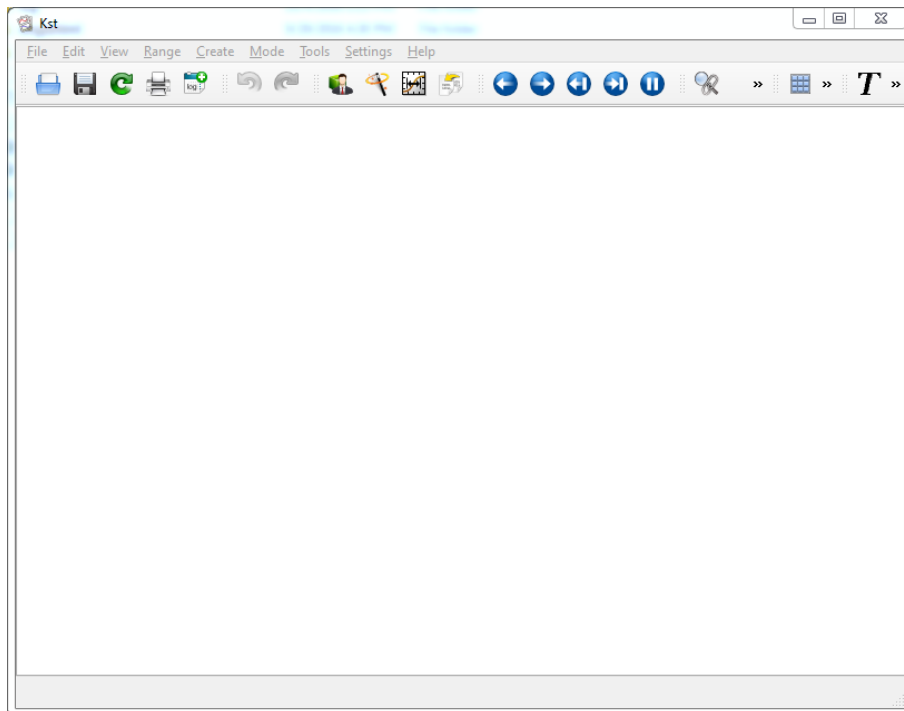
Häufig gestellte Fragen FAQ:

Keine bis jetzt. Dieser Abschnitt wird aktualisiert, sobald sich Fragen ergeben. Fragen können im Kontaktformular unter <http://taxis-instruments.de/sample-page/kontakt> gestellt werden oder schicke eine Email an <aqserver at taxis-instruments dot de>

Einführung in Kst2

Kst2 ist ein Open Source Programm, mit dem Daten visualisiert und geplottet werden können. Das Programm kann von <https://kst-plot.kde.org> heruntergeladen werden. Die aktuelle Windows Version ist 2.08. Auf der Homepage werden einige Tutorial Videos zur Verfügung gestellt. Du solltest vielleicht einen Blick darauf werfen, bevor du das Programm zusammen mit Aqserver benutzt.

Benutzung

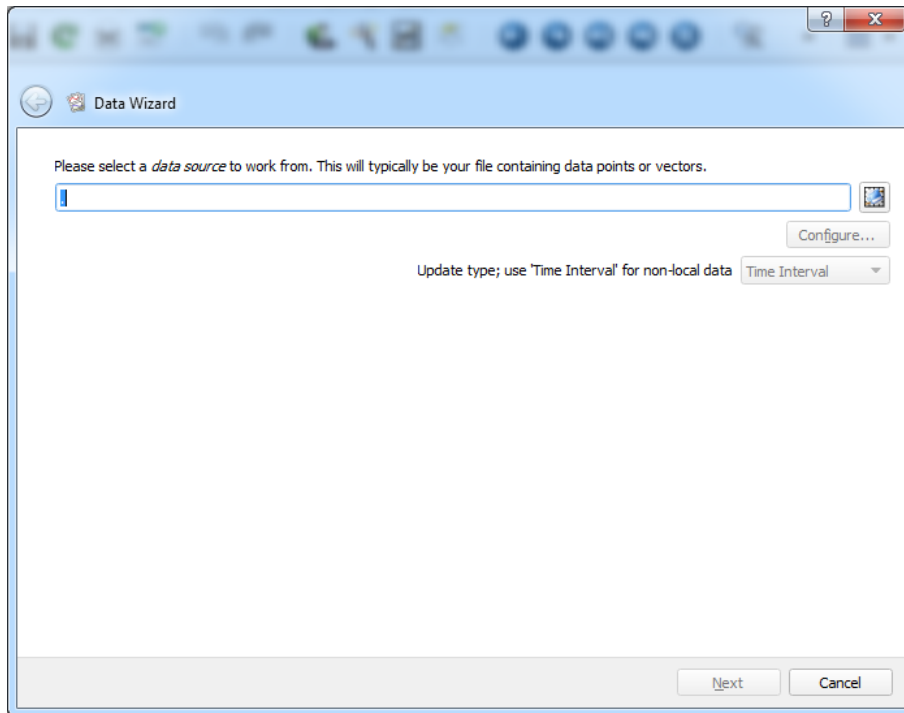


Die einfachste Art Kst zusammen mit Aqserver zu benutzen, ist mit dem “Data Wizard” zu beginnen.

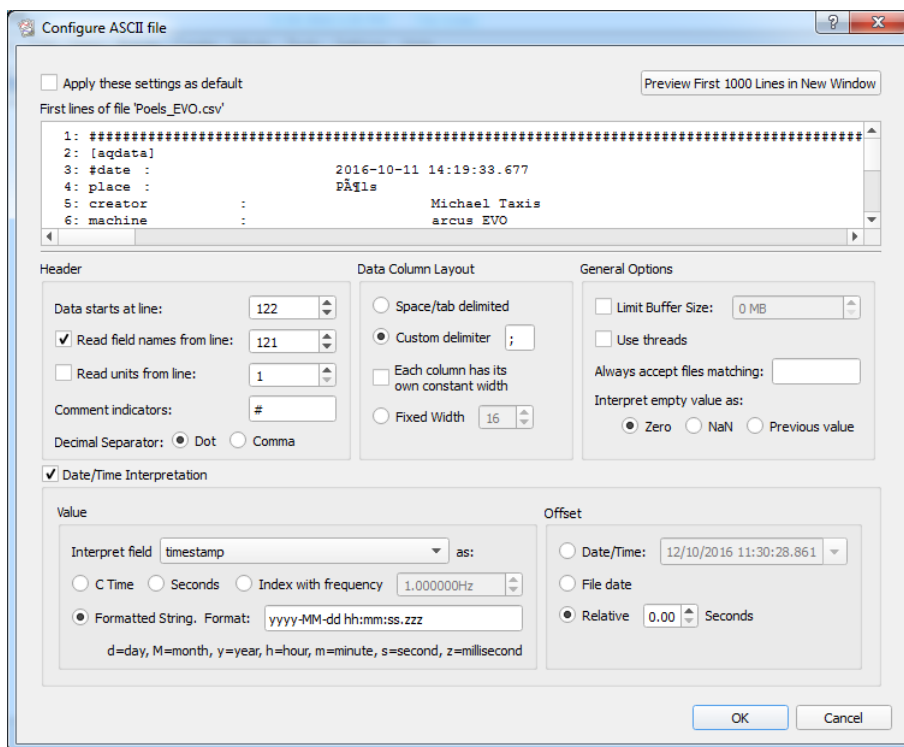
Bemerkung: Übrigens kann man den Data Wizard auch mehrmals hintereinander starten, wenn man noch Diagramme zur bisherigen Anzeige hinzufügen möchte. Wenn mehrere Instanzen von Aqserver gestartet wurden, können dabei auch die unterschiedlichen Datendateien gewählt werden.

Starte den Data Wizard mit folgendem Icon:





Wenn sich das Data Wizard Fenster öffnet, muss als erstes die aktive Datendatei ausgewählt werden (diese befindet sich in deinem “data” Unterverzeichnis). Wähle die Datei aus und klicke dann auf den “Configure” Knopf.

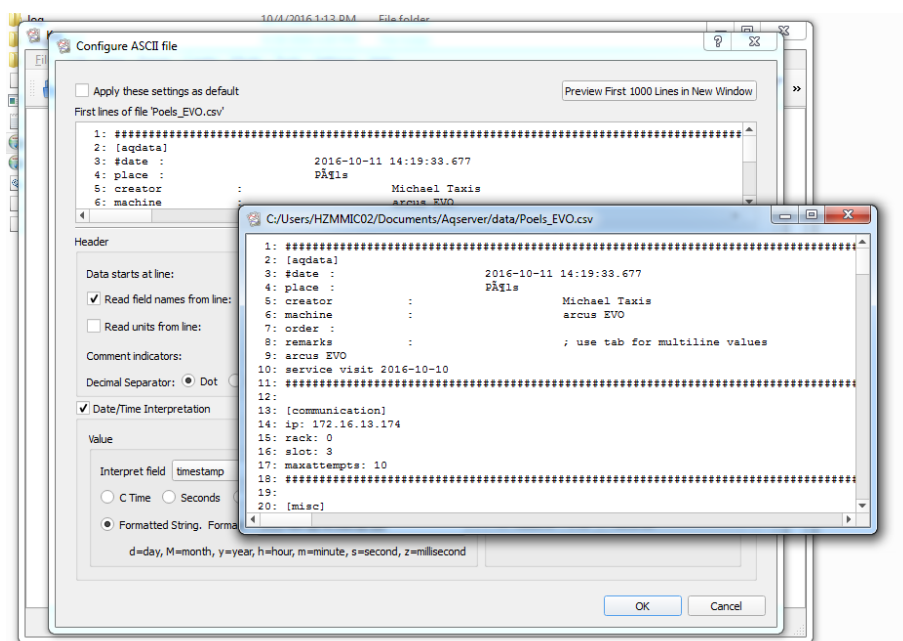


Jetzt öffnet sich das “Configure ASCII file” Fenster. Hier müssen die Eigenschaften der Datendatei eingestellt werden. Folgende Werte müssen (mindestens) eingestellt werden:

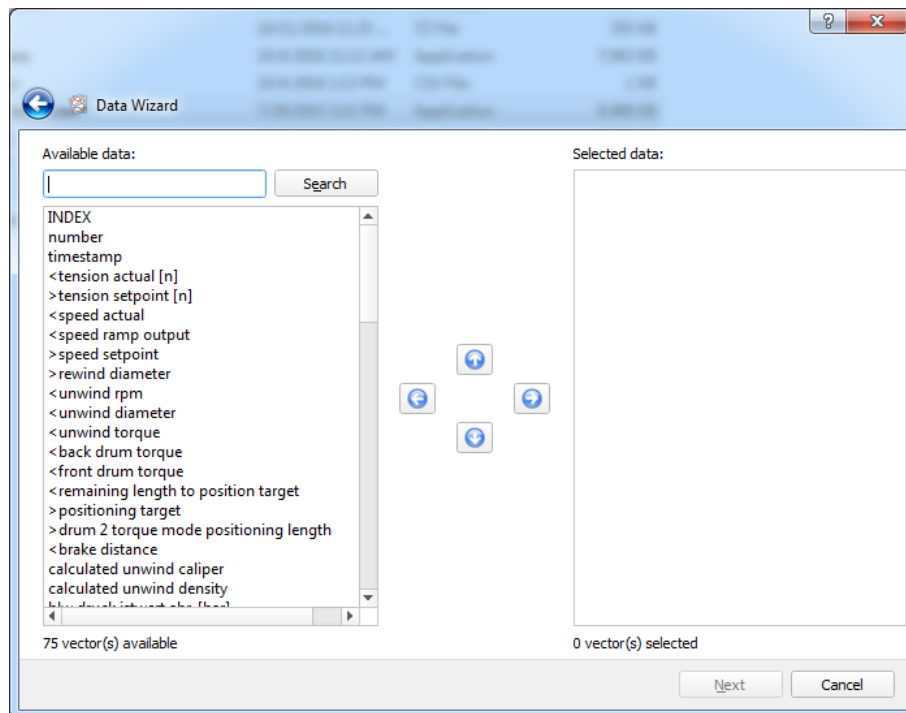
- Data starts at line: Scrolle in dem Fenster, das deine Datendatei zeigt, herunter, bis du den ersten Datensatz mit aufgezeichneten Werten erreichst. Merke dir die Zeilennummer und schreibe sie in das entsprechende Feld. Sollte diese Zeile nicht angezeigt werden, hilft dir vielleicht der Knopf “Preview first 1000 lines in new window”.

- Offensichtlich muss die “Read field names from:” Option aktiviert werden und die entsprechende Zeilennummer eingetragen werden (obige Zeilennummer - 1)
- stelle das Dezimaltrennzeichen ein
- stelle das Werttrennzeichen so ein wie in der Datendatei.
- Aktiviere “Date/Time Interpretation”
- Scrolle die Dropbox “Interpret field as” herunter und wähle den Wert “timestamp”. Wenn keine Auswahl angeboten wird, verlasse das Fenster mit “ok” und öffne es dann erneut. Jetzt sollte eine Auswahl vorhanden sein. Wähle “timestamp”.
- aktiviere “formatted string” und stelle das Format auf “yyyy-MM-dd hh:mm:ss.zzz” ein. Dann wird unser Zeitstempel korrekt interpretiert.

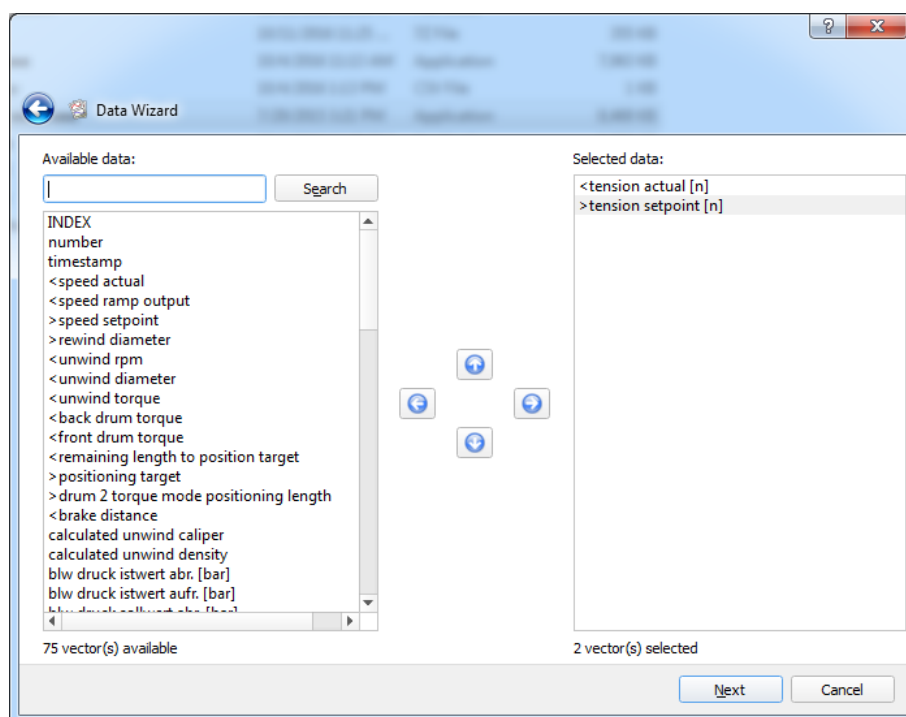
Verlasse das Fenster jetzt mit “Ok”. Setze den Update Typ auf entweder “time interval” oder du kannst auch “change detection” ausprobieren. Wenn nur eine dekomprimierte Archivdatei angezeigt werden soll, kann “no update” verwendet werden. Klicke dann auf “Next” im Data Wizard.



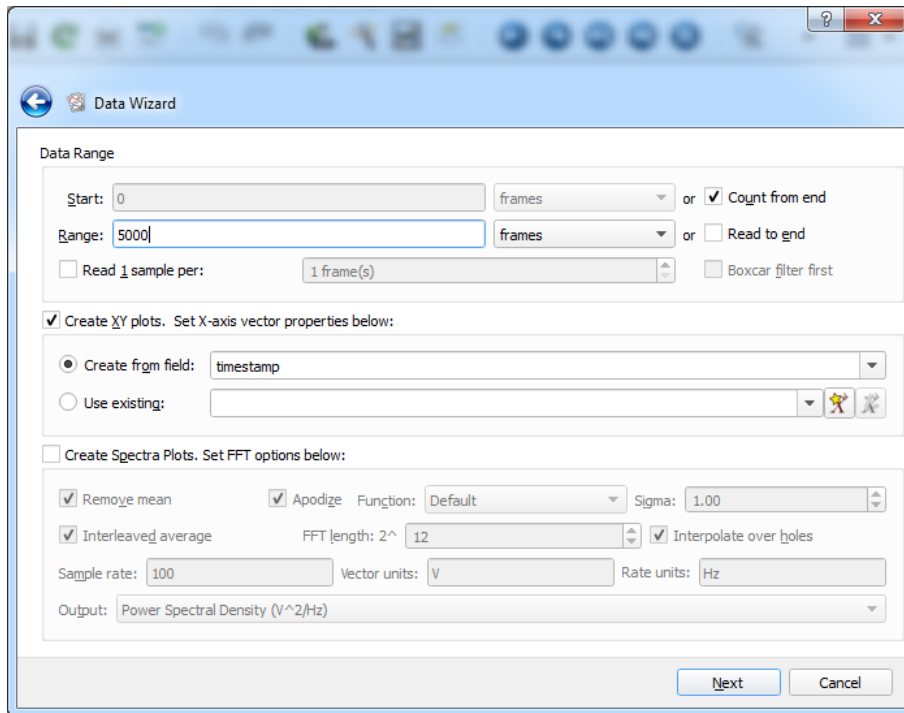
Data wizard mit “Preview first 1000 lines in new window”.



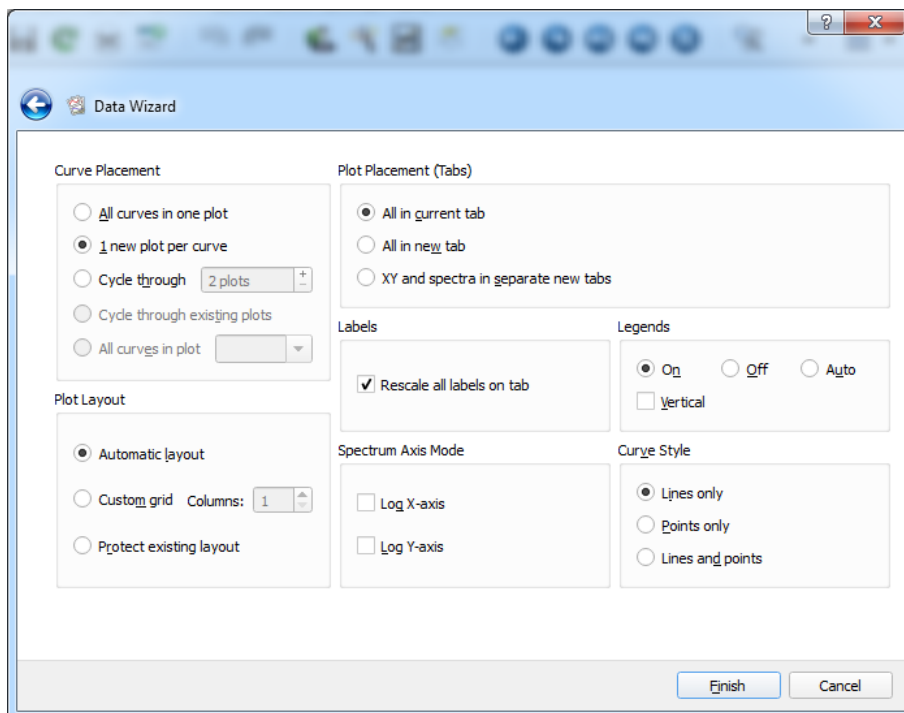
Wähle die Werte, die in einem Diagramm angezeigt werden sollen und bringe sie auf die rechte Seite mit den Pfeiltasten. Klicke dann auf "Next".



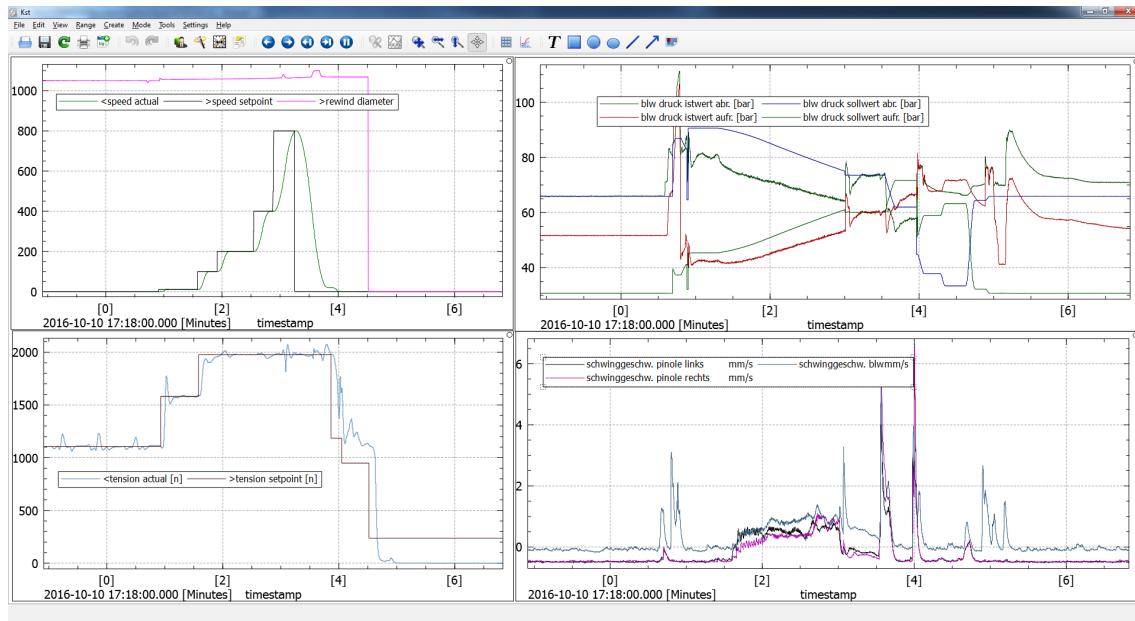
Das Gleiche mit einigen ausgewählten Werten.



Experimentiere mit den Datenbereichseinstellungen. Aktiviere “Create XY-plots”. Für den X-Achsen Vektor wählen wir unser Feld “timestamp”. Klicke dann auf “Next”.



Stelle die Anzeige der Diagramme je nach Wunsch auf “in one plot” oder “1 plot per curve”.



Jetzt sollten die aufgezeichneten Werte angezeigt werden. Falls du eine aktuelle Datendatei gewählt hast, sollten sich die Diagramme auch aktualisieren.

Bemerkung: Wenn ein Trigger für eine neue Datendatei ausgelöst wurde, benutze den Knopf mit dem grünen runden Pfeil, um die Datei zu aktualisieren. Sonst wird bei einem Wechsel auf eine neue Datei nichts mehr angezeigt.

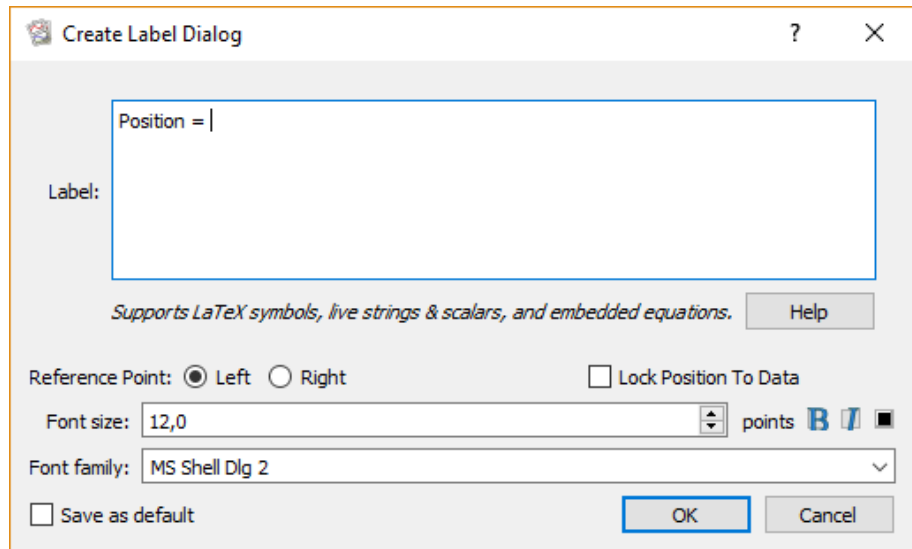
Für detaillierte Information über Kst2, lies die Dokumentation oder browse durch die Foren usw.

Hinzufügen einer Anzeige des aktuellen Werts

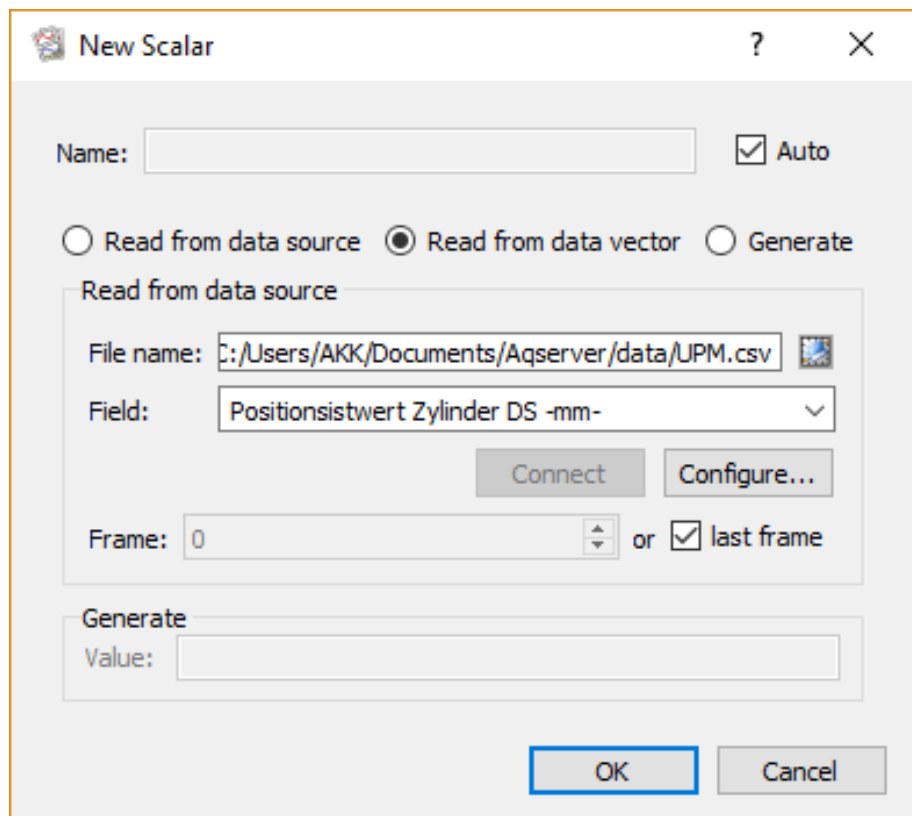
Soll im Diagramm der aktuelle Wert angezeigt werden, kann man dies mit dem Textwerkzeug realisieren. Dazu gehe wie folgt vor:

- Aktiviere das “Label” Werkzeug, es öffnet sich ein Fenster zur Eingabe des Textes

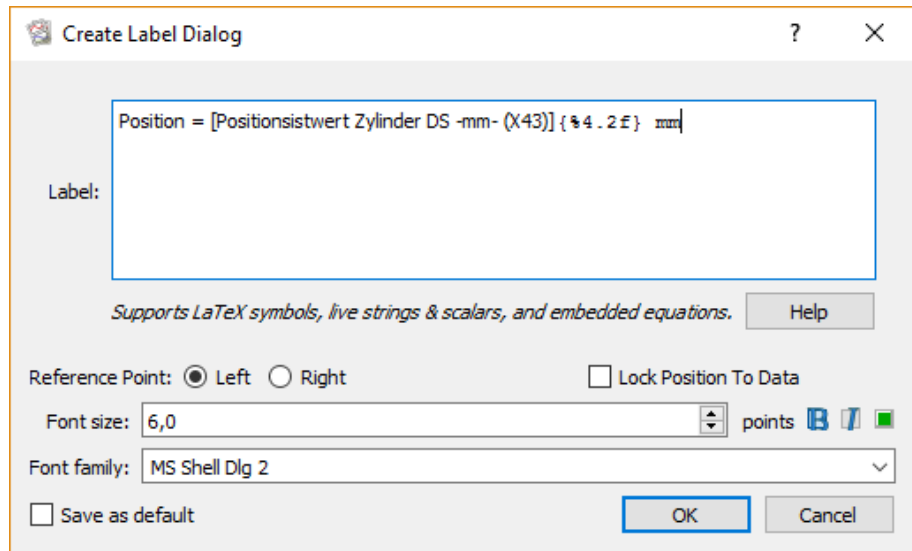




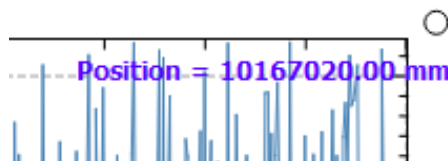
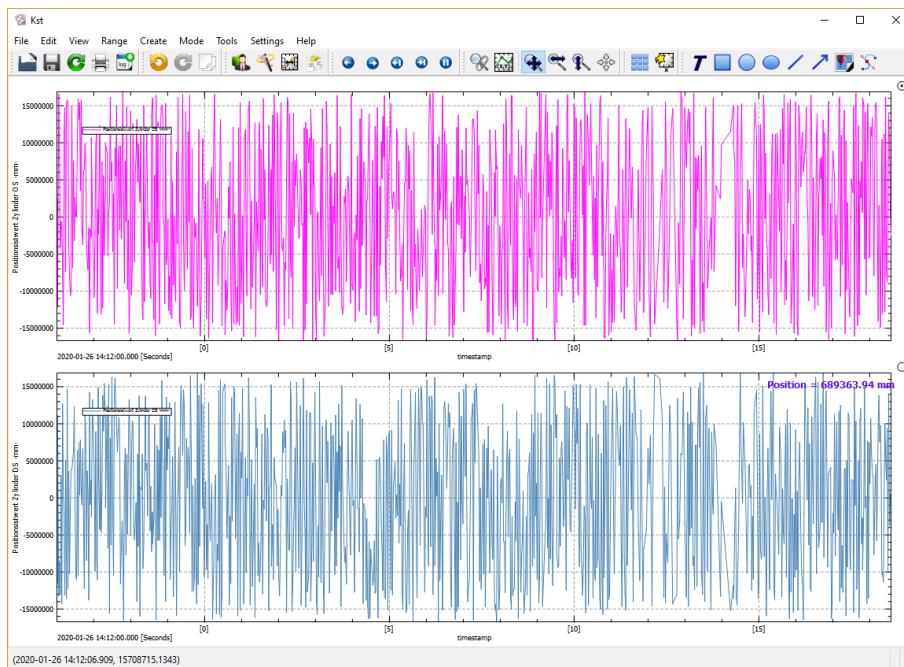
- Gib deinen Text ein. Um den aktuellen Wert einzufügen, klicke rechts und wähle “Insert new scalar”. Es öffnet sich das Fenster zur Einstellung für den neuen Skalar.



- Hier stellen wir folgendes ein:
 - aktiviere “Read from data vector”
 - wähle den gewünschten Wert aus der Dropdownliste
 - aktiviere das Kästchen “last frame” und akzeptiere die Einstellungen mit Ok
- Zurück im Eingabefenster für das Label, kann noch das Format des Wertes hinzugefügt werden (Näheres steht in der Hilfe, Button “Help”) Nach Fertigstellung verlasse die Eingabe mit “Ok”



- Platziere jetzt das Label mit der Maus, bei Klick wird das Label platziert.



Das Label sollte sich jetzt aktualisieren und den aktuellen Wert anzeigen.

Changelog

28 March 2017

- aqserver.py: added writing units for values to datafile
- aqserver.py: added functionality for demo: creating data from random function, without connection to plc. DEMO is shown in headline of aqserver
- aqserver.py: added functionality for booloffset: boolean values within one byte are added an offset, so they can be shown in one plot without overlapping each other
- PrgUtils.py: added config.optionxform=str for get_config for case sensitive config values. ATTENTION: config values IP, RACK and SLOT must be changed to lowercase in your new config files (ip, rack,slot)

17 March 2017

- aqserver.py: added config filename to display, so we know what is running
- aqserver.py: delete header file and temp files when program exits

15 March 2017

- aqserver.py: fixed error: when using multiple instances still same name for header file (and temp file) was used. This messed up data files. Now included datafileprefix in these filenames so they are unique.
- aqserver.py: for same reason logfile name was abandoned, using datafileprefix instead to have unique name

13 February 2017

- added german docs

13 October 2016

- updated docs
- created new installer using pyinstaller and NSIS. Problems before with py2exe.

11 October 2016

- added check for maximum records (from config) to avoid files getting too big
- added try ..except for reading config file

06 October 2016

- PrgUtils.py: added get username for all OS (Windows, Linux, MacOS)
- aqserver.py: change data and log directory, depending on OS. In Windows this will be sub directories of .. My Documents\Aqserver folder, in Linux/MacOS this will be in current users home directory (//home//user//Aqserver//..)

05 October 2016

- NSIS: tried to make setup.exe for Windows
- aqserver.py: added manual trigger always possible

- aqserver.py: added check for variables, and exit with config fault, problem to logfile

30 September 2016

- updating docs
- aqserver.py: check if argument for config file is list or string
- aqserver.py: added purging log directory, when debug is 0 (= no debugging), only actual logfile remains, with entry program start and end
- aqserver.py: improved error detection, connect and disconnect of client

29 September 2016

- updated docs
- compiled to exe with py2exe
- can test now with local Snap7 server
- added verification for configfile, data dir and log dir, if a dir is missing it will be created
- fixed keypress problem, program was not responding to every keypress

15 September 2016

- added try..except for communication error, program exits normal now, with message to user and logfile
- when client was already connected program tries to disconnect and to connect again (no exit)
- when scantime is set to 0 then program scans as fast as possible (no sleep)

28 November 2015

- modified aqserver(none OOP version), can now use directory structure yearmonthday for saved files

11 August 2015

- started refactoring and making OOP aware, with acqserver class (not yet finished)

10 August 2015

- tested and corrected trigger functionality
- tested S7 300 with plcsm and nettoplcsim

07 August 2015

- added filecopy routines to PrgUtils
- added trigger functionality to server (aqserver.py)

06 August 2015

- added config section for debug, with settings for debug level, logfile
- corrected error in endless loop that creates string with values
- create header template at start, that can be copied to further data files in case of trigger
- removed all sys.args, only*c configfile remains, all other settings in config file
- tried to start 2 instances of program
- use different configfile
- set different recording filename
- when started from 2 cmd environments with*c parameter and different configfile, it works!
- tested filecopy, rename, append, replace in file as preparation for trigger event actions

05 August 2015

- added data file name to config, so we can start aqserver several times using different settings, when scanning several PLCs

Licensing

Aqserver is distributed as python source code or Windows setup program under Lesser General Public License version 3.0 (LGPLv3)

Basically this means that you can distribute your commercial software linked with Aqserver without the requirement to distribute the source code of your application and without the requirement that your application be itself distributed under LGPL.

A small mention to the project or the author is however appreciated if you include it in your applications.

Disclaimer of Warranty

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

IF ANYONE BELIEVES THAT, WITH THIS WEBSITE OR WITH AQSERVER PROJECT SOME COPYRIGHTS HAVE BEEN VIOLATED, PLEASE EMAIL US, AND ALL THE NECESSARY CHANGES WILL BE MADE.