tau4

Devel's Manual

Ver. 2016-11-27.2143

Table Of Contents

List Of Tables

List Of Figures

Revision History

Datum	Änderung
2016-04-19	Erstausgabe.

1 Glossary

2 tau4

3 automation

3.1 ces.py – Control Engineering Systems

3.2 plc.py – PLC

3.3 sm.py – State Machines

```
class SM:

    """State Machine.

    An app can have more than one state machine.
    But be aware, that state machine states are singletons and you have to decide at
    runtime, which state machine they belong to!
    """

    def __init__( self, sms_initial):
        self.__sms = sms_initial
        return

    def execute( self):
        self.__sms.execute()
        return self


class SMState(metaclass=Singleton):

    def __init__( self, sms):
        self.__sm = sms.sm()
        return

    @abc.abstractmethod
    def execute( self):
        pass

    @abc.abstractmethod
    def is_condition_met( self):
        pass

    def select( self):
        self.sm().state( self)

    def sm( self):
        return self.__sm
```

3.3.1 Usage

```
class _SMStates:

    class Idle(SMState):

        def execute( self):
            if _SMState.Ready( self).is_condition_met():
                _SMStates.Ready( self).select()
                return

            return
```

4 ce

4.1

5 com

5.1 tau4com.mbus

tau4com.mbus ist eine Schicht über tau4.sweng.PublisherChannel, die eine Entkopplung vornimmt zwischen Publisher und Subscriber.

5.1.1 Interface

App-Code für den Subscriber sieht so aus (Beispiel):

```
mbus.gps.NewReading.RegisterSubscriber( self._mbus_on_new_gps_reading_)
```

App-Code des Publishers sieht so aus (Beispiel):

```
v = SomeGpsDevice().value()
mbus.gps.NewReading( v).publish()
```

6 data

Dieses Package besteht aus folgenden Packages:

- varbls

6.1 .varbls

6.1.1 FlexValue

Deckt die Features

- Reading / writing

ab. Alles weitere ist durch Subclasses zu realisieren, damit Objekte dieser Klasse schnell bleiben.

Das „Flex“ im Namen bedeutet, dass der Typ, den das Objekt hält, vom Wert abhängig ist, der dem Ctor übergeben wird.

Eine Identifikation des Values ist immer möglich, indem man

```
tau4.Objects.add( v, u”your sophisticated name goes here”)
```

ausführt, wobei `v` z.B. per

```
v = FlexValue( 42)
```

erzeugt worden ist.

6.1.2 FlexVarbl

Hinzu kommen die Features

- Identification
- Timing
  - Created
  - Modified
- Publishing
  - on_modified
  - on_limit_violated

Clipping und Scaling sind Strategien, die übergeben werden können. Eine Strategie wird von der Basisklasse `ValueMangler` abgeleitet.

6.1.3 FlexQuant

Hinzu kommen die Features

- Name
- Dimension

6.1.4 ValueMangler

Jede Strategie muss die beiden Methoden `app2value()` und `value2app()` implementieren.

6.1.4.1 Clipper

Wenn man clippen möchte, dann funktioniert das so:

```
v = varbls.FlexVarbl( 0.0)
v.value_manglers().add( varbls.Clipper( -42, 42)))
```

Das Clipping erfolgt immer beim Schreiben. Für eine geclippte Varbl gilt also immer die Invariante

```
min <= v <= max
```

6.1.4.2 Scaler

Wenn die App in anderen Dimensionen rechnet als das System, dann kann man so vorgehen:

```
v = varbls.FlexVarbl( 0.0)
                                  # Rechnet in mm
v.value_manglers().add( varbls.Scaler( app2value=1000)))
                                  # Rechnet in m.
                                  # Beim Schreiben des value wird mit 1000 mult., beim
                                  # Lesen durch 1000 div.
```

Index