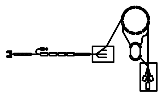

STD - Standard-Routinen zur Konfiguration von einfachen Gerätemodellen

Gerätemodell und Softwareentwurf

P. Kainberger

GSI, Gesellschaft für Schwerionenforschung mbH
Planckstraße 1, D-64291 Darmstadt
Tel.: 06159/71-0

Issued: 15. Sep. 1998
Key: U-GSW-05
Printed: 1. März 2010

Dieses Dokument ist gedacht als Reference Manual für die Benutzung der Standard-USRs zur Konfigurierung einfacher Gerätemodelle mit Standardkomponenten für die Anbindung primitiver DC-Geräte an das Kontrollsystem.

Document Revision History		
Date	Name	Comment
20. Oct. 1994	P. Kainberger	erste Version
13. Jan. 1995	P. Kainberger	Anpassung an letzte Modifikationen
15. Sep. 1998	L. Hechler	`\subsubsection` fr set_dev_fct

Inhaltsverzeichnis

1 Überblick

In der GSI gibt es eine ganze Menge *primitiver* DC-Geräte (Geräte, die keine Timinginformationen benötigen und auch sonst keine Realtime-Anforderungen an das Kontrollsystem stellen), die am Kontrollsystem angeschlossen sind oder noch angeschlossen werden müssen. Oftmals sind diese Komponenten (in Ermangelung eines eigenen ordentlichen Gerätemodells) als *Sondergeräte* an einem bereits existierenden Gerätemodell angeschlossen. Dabei erfüllen diese Komponenten meist nicht alle Anforderungen des jeweiligen Gerätemodells und bedürfen dann auf Operatingebene einer Spezialbehandlung.
Z.B. gibt es Druckmeßgeräte, die an der Magnetsoftware *MD* angeschlossen sind und als *CURRENTI* nicht den Strom in A sondern einen Druckwert in Torr liefern. Zudem kann man sie nicht Ein/Ausschalten und der Status ist auch nicht belegt (nur wenn man Glück hat, sind alle Bits 1). Um solche, aus der Not geborenen, Schweinereien vermeiden zu können, wurde die sog. *Standard*-Gerätesoftware entwickelt, die im Folgenden beschrieben wird.

2 Zielsetzung

Sinn und Zweck des *Standard-Gerätemodells* ist, *primitive* DC-Geräte unterschiedlichster Art (verschiedene Statusbelegung, Soll- und Istwerte mit verschiedener Bedeutung und unterschiedlichen Normierungen, ...) mit möglichst wenig Aufwand über eine einzige SE am Kontrollsystem so anzuschließen, daß zur Operatingebene hin jede einzelne Geräteeigenschaft auch als sinnfällig bezeichnete Property (mit individueller Normierung) abgebildet wird.

3 Elementarfunktionen auf der SE-Ebene

Die EQMs der SE-Ebene erhalten alle notwendigen Funktionsparameter (Funktionscode, MIL_timeout, Daten, ...), als Parameter und Daten in einer *Command_Communication_Area*.
Folgende Elementarfunktionen (EQMs) stehen zur Versorgung der Geräte zur Verfügung.

3.1 EQM_Std_DataWrite

Das *EQM_Std_DataWrite* kann mit folgender Parameter und Datenversorgung bis zu 16 Sollwerte mit einem einzigen Aufruf an ein Gerät schicken.

```
{--------------------}
{ Property : Data, WA }
{--------------------}
CONST
  Data_W__P_Count = 1;
  Data_W__D_Count = 3;  { 3 entries }
TYPE
  Data_W_Workpara_Type =
    RECORD
      count : uns_word;      { count of data to write }
    END;

  Data_W_Workdata_Type =
    RECORD
      data_desc : ARRAY [1..16] OF RECORD
                    fct_code : uns_word; { function-code to send data }
                    timeout  : uns_word; { timeout for communication  }
```

```
                  value    : word;     { value to send to device    }
                END;
  END;
```

3.2 EQM_Std_DataRead

Das *EQM_Std_DataRead* kann mit folgender Parameter und Datenversorgung bis zu 16 Istwerte mit einem einzigen Aufruf von einem Gerät lesen.

```
{--------------------}
{ Property : Data, RA }
{--------------------}
TYPE
  Data_R_Workpara_Type =
    RECORD
      count : uns_word;      { count of data to read }
      data_desc : ARRAY [1..16] OF RECORD
                    fct_code : uns_word; { function-code to send data }
                    timeout  : uns_word; { timeout for communication  }
                  END;
    END;

  Data_R_Workdata_Type = ARRAY [1..16] OF word; { data read from device }
```

3.3 EQM_Std_FctWrite

Das *EQM_Std_FctWrite* kann mit folgender Parameter und Datenversorgung bis zu 16 Funktionscodes mit einem einzigen Aufruf an ein Gerät schicken.

```
{--------------------}
{ Property : Fct, WA }
{--------------------}
CONST
  Fct_W__P_Count = 1;
  Fct_W__D_Count = 3;  { 3 entries }
TYPE
  Fct_W_Workpara_Type =
    RECORD
      count : uns_word;      { count of functioncodes to write }
    END;

  Fct_W_Workdata_Type =
    RECORD
      Fct_desc : ARRAY [1..16] OF RECORD
                   fct_code : uns_word; { function-code to send data }
                   timeout  : uns_word; { timeout for communication  }
                   length   : uns_word; { length (in ms) of function on device }
                 END;
    END;
```

3.4 EQM_Std_DataPoll

Das *EQM_Std_DataPoll* kann mit folgender Parameter und Datenversorgung auf bis zu 16 Datenzustände mit einem einzigen Aufruf "pollen".
Dabei wird mit dem angegebenen Funktionscode (fct_code) ein 16-Bit Wert vom Gerät gelesen, die für den Vergleich relevanten Bits (sel_mask) selektiert und entsprechend der Vergleichsanweisung (comp_opt) mit dem erwarteten Bitmuster (val_mask) verglichen. Die ganze Aktion wird maximal *rep_count*-mal wiederholt mit einem Abstand von *pollrate* s zwischen den Abfragen.

```
{-------------------------}
{ Property : DataPoll, WA }
{-------------------------}
CONST
  DataPoll_W__P_Count = 1;
  DataPoll_W__D_Count = 7;  { 7 entries }
TYPE
  DataPoll_W_Workpara_Type =
    RECORD
      count : uns_word;      { count of data to poll for }
    END;

  DataPoll_W_Workdata_Type =
    RECORD
      data_desc : ARRAY [1..16] OF RECORD
                    fct_code : uns_word; { function-code to read data }
                    sel_mask : uns_word; { describe data to wait for  }
                    val_mask : uns_word; { describe value of data to wait for  }
                    comp_opt : (comp_eq, comp_ne, comp_ge, comp_gt, comp_le,
                                comp_lt);{ compare option             }
                    timeout  : uns_word; { timeout for communication  }
                    pollrate : uns_word; { repetition rate for polling }
                    rep_count: word;     { count of repetitions }
                  END;
    END;
```

4 Standardfunktionen auf der GμP-Ebene

Normalerweise ist in den USRs und EQMs eines Gerätemodells festgelegt, mit welchen Normierungskonstanten Gerätesollwerte (in physikalischen Einheiten) auf gerätespezifische Form (16-Bit-Sollwert an der Interfacekarte) umzurechnen sind und mit welchem Funktionscode die Daten zur Interfacekarte zu übertragen sind.
Bei den *StandardUSRs* sind *alle* gerätespezifischen Eigenheiten (Funktionscode, Normierungskonstanten, ...) für jedes einzelne Gerät und für jede einzelne Property frei wählbar. Es gibt einen Satz von USRs (immer 1 ... 10) zum Setzen und Lesen von Sollwerten, zum Lesen von Istwerten und zur Durchführung von Schaltfunktionen. Jede einzelne USR findet in der Konstantentabelle der VME-DBS eine exakte Beschreibung der zu berücksichtigenden Geräteeigenschaften und benutzt zur Realisierung die Elementarfunktionen der SE-Ebene mit Hilfe von *Write_Command*- und *Read_Command*-Aufrufen.
Auf der GμP-Ebene stehen zur Versorgung der Geräte folgende Standardfunktionen (u.a. USRs) zur Verfügung, die in einem PASCAL-Sourcefile *includiert* werden können.

4.1 STD_Default_USRS

Im `STD_Default_USRs.PIN` werden Funktionen zum vereinfachten Gerätehandling definiert und die eigentlichen *Default_USRs* aus `LIB68:Default_USRS_IMP.PIN` includiert.
Folgende Funktionen werden zum Gerätehandling zur Verfügung gestellt.

4.1.1 act_dev_constants

Diese *Function* aktualisiert die gerätespezifischen Konstanten im DPRAM der SE. Sind die Konstanten im DPRAM nicht *valid*, so werden aus der *const*-Tabelle in der VME-DBS die Einträge ins DPRAM übertragen und als *valid* gekennzeichnet. Die Konstanten im DPRAM werden von der SE-Seite aus bei einem *Restart* der SE oder beim *Init* eines Gerätes als *invalid* markiert.
Als Besonderheit sei hier vermerkt, daß die Einträge in der VME-DBS nicht alle vom gleichen Typ sind. Die meisten Einträge sind *Integer*-Werte, lediglich die Faktoren zur Normierung der Soll- und Istwerte sind vom Typ *Real*.
`act_dev_constants` wird auch von den *Default_USRs* aufgerufen.

4.1.2 set_dev_soll

Diese *Function* handhabt das Setzen eines Sollwertes am Gerät per *Write_Command*-Aufruf. Die Parameter, die dazu nötig sind, werden aus dem Konstantensatz im DPRAM der SE entnommen.

4.1.3 get_dev_ist

Diese *Function* handhabt das Lesen eines Istwertes vom Gerät per *Read_Command*-Aufruf. Die Parameter, die dazu nötig sind, werden aus dem Konstantensatz im DPRAM der SE entnommen.

4.1.4 set_dev_fct

Diese *Function* handhabt das Durchführen einer Schaltfunktion am Gerät per *Write_Command*-Aufruf. Die Parameter, die dazu nötig sind, werden aus dem Konstantensatz im DPRAM der SE entnommen.

4.2 STDW_PowerUSR

Die *STDW_PowerUSR* führt die Ein-/Ausschaltfunktion eines Gerätes durch. Folgendes Beispiel zeigt die Verwendbarkeit derselben:

```
CONST
  w_power_name = 'POWER   ';
  w_power_timeout = 181;

%INCLUDE eqp$std:std$w_power$usr.pin
```

4.3 STD$R_Constant$USR

Die *STD$R_Constant$USR* liefert die gerätespezifischen Konstanten eines Gerätes, also die Parametersätze aller möglichen Geräteeigenschaften. Folgendes Beispiel zeigt die Verwendbarkeit derselben:

```
CONST
  r_constant_nr   = 1;
  r_constant_name = 'CONSTANT';
  r_constant_timeout = timeout;

%INCLUDE eqp$std:std$r_constant$usr.pin
```

4.4 STD$W_Copyset$USR

??? Braucht man das eigentlich explizit ??? ??? Könnte man auch implizit vereinbaren ???

```
CONST
  w_copyset_name = 'COPYSET ';
  w_copyset_timeout = timeout;

%INCLUDE eqp$std:std$w_copyset$usr.pin
```

4.5 STD$W_Soll{1 ... 10}$USR

Die *STD$W_Soll{1 ... 10}$USR* ermöglicht das Setzen und Lesen eines Sollwertes. Dabei ist der gewünschte *Propertyname* und ein Gerätemodellspezifischer *timeout* zu vereinbaren.
Mit der Sollwertnummer (1 ... 10) ist der Index in der Sollwerttabelle im DPRAM der SE und der Index in der Gerätekonstantentabelle (auch im DPRAM) festgelegt. D.h. in der Konstantentabelle (jedes einzelnen Gerätes, das mit diesem Standardgerätemodell betrieben wird) in der VME-DBS muß es einen Parametersatz mit diesem Index (= Sollwertnummer) geben. Daran muß man also denken, denn dabei kann man Fehler machen!
Folgendes Beispiel zeigt die Verwendbarkeit:

```
CONST
  w_Soll{1...10}_nr = <USR-Nummer>;
  w_Soll{1...10}_name = 'CURRENTS';
  w_Soll{1...10}_timeout = 5;

%INCLUDE eqp$std:std$w_Soll{1...10}$usr.pin
```

Zum Setzen des Sollwertes am Gerät wird die Funktion `set_dev_soll` benutzt.

4.6 STD$R_Ist{1 ... 10}$USR

Die *STD$R_Ist{1 ... 10}$USR* ermöglicht das Lesen eines Istwertes. Dabei ist der gewünschte *Propertyname* und ein Gerätemodellspezifischer *timeout* zu vereinbaren. Folgendes Beispiel zeigt die Verwendbarkeit:

```
CONST
  r_Ist{1...10}_nr = <USR-Nummer>;
  r_Ist{1...10}_name = 'CURRENTI';
  r_Ist{1...10}_timeout = 5;
```

```
%INCLUDE eqp$std:std$r_Ist{1...10}$usr.pin
```

Zum Lesen des Istwertes vom Gerät wird die Funktion `get_dev_ist` benutzt.

4.7 STD$N_Switch{1 . . . 10}$USR

Die *STD$N_Switch{1 . . . 10}$USR* ermöglicht das Durchführen einer Schaltfunktion. Dabei ist der gewünschte *Propertyname* und ein Gerätemodellspezifischer *timeout* zu vereinbaren. Folgendes Beispiel zeigt die Verwendbarkeit:

```
CONST
  n_switch{1...10}_nr = <USR-Nummer>;
  n_switch{1...10}_name = 'SWITCH  ';
  n_switch{1...10}_timeout = 18;

%INCLUDE eqp$std:std$n_switch{1...10}$usr.pin
```

Zur Durchführung der Schaltfunktion am Gerät wird die Funktion `set_dev_fct` benutzt.

5 Beschreibung der Konstantentabelle in der VME-Datenbasis

Für jedes einzelne Gerät muß in der Konstantentabelle der VME-Datenbasis für jede USR, die in einem Standardgerätemodell verwendet wird, ein Satz von Parametern definiert werden.
Die Funktion *act_dev_constants* liest die Liste der Einträge in der VME-DBS und sortiert sie in eine feste Struktur im DPRAM der SE. Die Funktion *act_dev_constants* wird von jeder Standard-USR explizit aufgerufen.
Die Konstanten müssen in folgender Reihenfolge eingetragen werden:

Bitmasken zur Statusinterpretation:

- **1:** Bitmaske (32 Bit), mit der die Bits im Status ausgewählt werden, die für die Anzeige des Ein/Aus-Zustandes relevant sind
- **2:** Bitmaske, in der die Bitwerte (0 oder 1) angegeben sind, die die ausgewählten Bits im Ein-Zustand haben müssen
- **3:** Bitmaske (32 Bit), mit der die Bits im Status ausgewählt werden, die für die Anzeige des Rechner/Hand-Betriebs relevant sind
- **4:** Bitmaske, in der die Bitwerte (0 oder 1) angegeben sind, die die ausgewählten Bits im Rechner-Betrieb haben müssen

Parameter zur POWER-Property:

- **5:** Pulslänge in ms (mindestens solange bleibt der Ein/Aus-Funktionscode an der Interfacekarte anstehen)
- **6:** Funktionscode, mit dem vom Gerät ein Datum gelesen werden kann, aus dem ersichtlich ist, ob die Ein/Aus-Funktion abgeschlossen ist

7: Bitmaske, mit der die Bits im gelesen Datum ausgewählt werden, die für die Anzeige des Ein/Aus-Zustandes relevant sind

8: Bitmaske, in der die Bitwerte (0 oder 1) angegeben sind, die die ausgewählten Bits im Ein-Zustand haben müssen

9: Pollrate in s (mit wieviel Sekunden Abstand soll der Erfolg der Ein/Aus-Funktion überprüft werden)

10: Maximale Anzahl der Überprüfungen

Parameter der Sollwerte 1. . . 10:

11: Anzahl *n1* der verschiedenen Sollwerte (0. . . 10)
Für jeden verwendeten Sollwert (falls n1 > 0) müssen folgende Einträge vorhanden sein:

11+1+6*(n1-1): Nummer des Sollwerts (1. . . 10) (zugleich Index in der Sollwerttabelle)

11+2+6*(n1-1): Minimalwert

11+3+6*(n1-1): Maximalwert

11+4+6*(n1-1): DAC-Maximalwert

11+5+6*(n1-1): DAC-Offset

11+6+6*(n1-1): Funktionscode zum Schreiben

Parameter der Istwerte 1. . . 10:

12+6*n1: Anzahl *n2* der verschiedenen Istwerte (0. . . 10)
Für jeden verwendeten Istwert (falls Anzahl > 0) müssen folgende Einträge vorhanden sein:

12+6*n1+1+5*(n2-1): Nummer des Istwerts (1. . . 10) (zugleich Index in der Istwerttabelle)

12+6*n1+2+5*(n2-1): Maximalwert

12+6*n1+3+5*(n2-1): ADC-Maximalwert

12+6*n1+4+5*(n2-1): ADC-Offset

12+6*n1+5+5*(n2-1): Funktionscode zum Lesen

Parameter der Schaltfunktionen 1. . . 10:

13+6*n1+5*n2: Anzahl *n3* der verschiedenen Schaltfunktionen (0. . . 10)
Für jede verwendete Schaltfunktion (falls Anzahl > 0) müssen folgende Einträge vorhanden sein:

13+6*n1+5*n2+1+3*(n3-1): Nummer (oder Index in der Schaltertabelle) der Schaltfunktion (1. . . 10)

13+6*n1+5*n2+2+3*(n3-1): Funktionscode zum Schalten

13+6*n1+5*n2+3+3*(n3-1): Pulslänge in ms (mindestens solange bleibt der Funktionscode an der Interfacekarte anstehen)

Beispiel hierzu:

```
/type      STDX
/structure 4i   ! Statusmake
           6i   ! Power-Property
            i   ! Anzahl Sollwertproperties:
            i   ! Nr. der Sollwertproperty
           4r i ! Beschreibung (erste) Sollwertproperty
            i   ! Anzahl Istwertproperties
            i   ! Nr. der Istwertproperty
           3r i ! Beschreibung (erste) Istwertproperty
            i   ! Anzahl Schaltproperties
            i   ! Nr. der Schaltproperty
           2i   ! Beschreibung (erste) Schaltproperty

/device    XXXSTD03  3
/constant
!
                ! Definition der Statusmasken:
    16#0100     !   power_select_mask (Bit 8)
    16#0100     !   power_value_mask (Bit 8)
    16#0200     !   remote_select_mask (Bit 9)
    16#0200     !   remote_value_mask (Bit 9)
                !
                ! Definitionen zur POWER-Property:
        200   !   pulslength of fct-code on device (in ms)
        192   !   fct-code for status-polling
    16#0100     !   poll_select_mask (Bit 8)
    16#0100     !   poll_value_mask (Bit 8)
          1     !   poll-rate for status-polling (in s)
        100     !   max. poll-count
                !
                ! Definitionen zu den Sollwert-Properties:
                !
          1     ! count of 'Sollwert'-Properties (Soll 1..10)
                ! Beschreibung der (ersten) Sollwert-Property
          1     !   index of element in structure (Soll 1..10)
        0.0 !   min. value
     1000.0 !   max. value
    32767.0 !   DAC-max
        0.0 !   DAC-offset
          6   !   function code
                !
                ! Definitionen zu den Istwert-Properties:
                !
          1   ! count of 'Istwert'-Properties (Ist 1..10)
                ! Beschreibung der (ersten) Istwert-Property
          1   !   index of element in structure (Ist 1..10)
     1000.0 !   max. value
    32767.0 !   ADC-max
        0.0 !   ADC-offset
        129   !   function code
                !
```

```
        ! Definitionen zu den Schalt-Properties:
        !
  1     ! count of 'Schalt'-Properties (Switch 1..10)
        ! Beschreibung der (ersten) Schalt-Property
  1   !   index of element in structure (Switch 1..10)
 19   !   function code
200   !   pulselength
```

6 Ausblick

Künftig könnte das Konfigurieren der Standardfunktionen zu einem Gerätemodell auch mit Hilfe eines kleinen Prográmmchens (DCL) erfolgen, welches das einzig notwendige USR-Sourcefile aus einer Konfigurationsdatei generiert.

A Beispielprogramm

Folgendes Beispielprogramm ist auf `ALICE` unter

```
SIS$ROOT:[EQP68K.STD.EXAMPLE]STD$USRS.PAS
```

zu finden und soll zeigen, wie die einzelnen Standardfunktionen zu einem individuellen Gerätemodell konfiguriert werden können.

Index