FC2 - <offline>

"FC2:PW to RE" Normalize perifery analog input
Nome: PW_RE **Famiglia:** Funct
Autore: SimoneS **Versione:** 0.1
Versione blocco: 2
Data e ora Codice: 15/12/00 09:44:18
Interfaccia: 31/10/00 12:11:23
Lunghezze (blocco / codice / dati): 00940 00662 00056

Indirizzo	Dichiarazione	Nome	Tipo	Valore iniziale	Commento
0.0	in	IN_value	ANY		Input value to normalize
10.0	in	OUT_MAX	ANY		Maximun output value
20.0	in	OUT_MIN	ANY		Minimum output value
30.0	in	IN_MAX	ANY		Maximun input value
40.0	in	IN_MIN	ANY		Minimum input value
50.0	in	IN_Smooth_factor	REAL		Smoothing factor (0 - 15)
54.0	out	OUT_value	ANY		Normalized output value
64.0	out	Overflow	BOOL		Input value out of nominal range alarm
66.0	out	Ret_val	WORD		Output error code
68.0	in_out	IN_Smoothing_CLK	BOOL		Smoothing clock
70.0	in_out	Dummy_REAL	REAL		Auxiliary
74.0	in_out	Dummy_BOOL	BOOL		Auxiliary
76.0	in_out	OUT_value_temp	REAL		Normalized output value (real format)
80.0	in_out	Smooth_val	REAL		Auxiliary
0.0	temp	IN_value_real	REAL		Recovered input value (real format)
4.0	temp	OUT_MAX_real	REAL		Recovered output maximum (real format)
8.0	temp	OUT_MIN_real	REAL		Recovered output minimum (real format)
12.0	temp	IN_MAX_real	REAL		Recovered input maximum (real format)
16.0	temp	IN_MIN_real	REAL		Recovered input minimum (real format)
20.0	temp	Norm_range	REAL		Auxiliary
24.0	temp	YmaxMinYmin	REAL		Auxiliary
28.0	temp	XmaxMinXmin	REAL		Auxiliary
32.0	temp	Over_load_aux_1	BOOL		Auxiliary
32.1	temp	Over_load_aux_2	BOOL		Auxiliary
34.0	temp	DB_Nr	INT		DB Number
36.0	temp	Loop	INT		

Blocco:FC2 Normalize IN value

This function normalize an analog input value in proper scale

Segmento: 1 [IN_value] value recovery

This network manages the recovery of [IN_value] value from any to real

```
      L     5                 // Loar # of any parametrs
//                        *** Loop start ****
Lab:  T     #Loop             // Loop index
      L     P##IN_value       // Recover ANY structure start point
      LAR1
      L     10
      L     #Loop             // Calculate actual any parametr address
      *I
      TAK
      -I
      ITD
      SLD   3
      +AR1                    // Load AR1 register with actual any parameter address
//;
      L     P##IN_value_real  // Recover [IN_value_real] address
      LAR2
      L     4
      L     #Loop             // Calculate actual real parametr address
      *I
      TAK
      -I
      ITD
      SLD   3
      +AR2                    // Load AR2 register with actual real parameter address
//;
      L     0
      L     W [AR1,P#4.0]     // # of DB
      ==I                     // If == 0 jump
      SPB   go1
      T     #DB_Nr            // Open DB
      AUF   DB [#DB_Nr]
//
go1:  L     B [AR1,P#1.0]     // Load type of input
      L     B#16#2            // Formato byte
      ==I
      SPB   byte
      TAK
      L     B#16#4            // Formato word
      ==I
      SPB   word
      TAK
      L     B#16#5            // Formato intero
      ==I
      SPB   word
      TAK
      L     B#16#6            // Formato doppia word
```

```
      ==I
      SPB   doub
      TAK
      L     B#16#7            // Formato doppio intero
      ==I
      SPB   doub
      TAK
      L     B#16#8            // Formato real
      ==I
      SPB   real
      L     W#16#1            // Error code 0001 "type mismatch"
      T     #Ret_val
      NOT
      SAVE
      BEA
//
byte: L     D [AR1,P#6.0]     // Load imput address
      LAR1
      L     B [AR1,P#0.0]     // Imput value byte interpretated
      ITD                     // Conv. from integer to double
      DTR                     // Conv. from double to real
      SPA   fine
//
word: L     D [AR1,P#6.0]     // Load imput address
      LAR1
      L     W [AR1,P#0.0]     // Imput value word interpretated
      ITD                     // Conv. from integer to double
      DTR                     // Conv. from double to real
      SPA   fine
//
doub: L     D [AR1,P#6.0]     // Load imput address
      LAR1
      L     D [AR1,P#0.0]     // Imput value long interpretated
      DTR                     // Conv. from double to real
      SPA   fine
//
real: L     D [AR1,P#6.0]     // Load imput address
      LAR1
      L     D [AR1,P#0.0]     // Imput value long interpretated
//
fine: T     LD [AR2,P#0.0]    // Save recovered value
      L     #Loop
      LOOP  Lab
```

Segmento: 2 Normalize

The formula used is:

((Ymax-Ymin)/(Xmax-Xmin))*(X-Xmin)+Ymin=Y

Xmax = Maximum input scale (MAX IN value; IN_MAX)

```
Xmin = Minimum input scale (MIN IN value;             IN_MIN)
Ymax = Maximum output scale* (MAX OUT value;         OUT_MAX)
Ymin = Minimum output scale** (MIN OUT value;        OUT_MIN)
X    = Input actual value (IN value;                IN_value)
Y    = Output normalized value (OUT value;         OUT_value)

* Maximum real refered to a PW maximum in full range filed (PW_max)
** Minimum real refered to a PW minimum in full range field (PW_min)
```

```
      L     3.251100e+004               // check limit area
      L     #IN_value_real              // PW value must be beetwen
      <R                                // PW_max and PW_min
      =     #Over_load_aux_1
      L     -4.864000e+003
      <R
      =     #Over_load_aux_2
//                              ^^^ Start normalization ^^^
      L     #OUT_MAX_real
      L     #OUT_MIN_real
      -R
      T     #YmaxMinYmin                // (Ymax-Ymin)
      L     #IN_MAX_real
      L     #IN_MIN_real
      -R
      T     #XmaxMinXmin                // (Xmax-Xmin)
      L     #YmaxMinYmin
      L     #XmaxMinXmin
      /R
      T     #Norm_range                 // (Ymax-Ymin)/(Xmax-Xmin)
      L     #IN_value_real
      L     #IN_MIN_real
      -R                                // (X-Xmin)
      L     #Norm_range
      *R                                // (Ymax-Ymin)/(Xmax-Xmin)*(X-Xmin)
      L     #OUT_MIN_real
      +R                                // ((Ymax-Ymin)/(Xmax-Xmin)*(X-Xmin))+ Ymin
      T     #OUT_value_temp             // Assign output value
//;
//                    *** Smoothing output ***
//;
      CALL  "FC11:Smoothing"            // Smoothing method
       Smooth_factor:=#IN_Smooth_factor
       Input_value  :=#OUT_value_temp
       CLK          :=#IN_Smoothing_CLK
       Out_smoothed :=#Smooth_val
       Prev_value   :=#Dummy_REAL
       Dummy_bit_1  :=#Dummy_BOOL

//;                           ^^^ Alarm control ^^^
      U     #Over_load_aux_1
      O     #Over_load_aux_2
      =     #Overflow
```

Segmento: 3 [OUT_value] value recovery
This network manages the recovery of [OUT_value] value from any to real

```
      L     P##OUT_value    // Recover ANY structure start point
      LAR1
//
      L     0
      L     W [AR1,P#4.0]   // # of DB
      ==I                   // If == 0 jump
      SPB   go6
      T     #DB_Nr          // Open DB
      AUF   DB [#DB_Nr]
//
go6:  L     B [AR1,P#1.0]   // Load type of output
      L     B#16#2          // Formato byte
      ==I
      SPB   byt5
      TAK
      L     B#16#4          // Formato word
      ==I
      SPB   wor5
      TAK
      L     B#16#5          // Formato intero
      ==I
      SPB   wor5
      TAK
      L     B#16#6          // Formato doppia word
      ==I
      SPB   dou5
      TAK
      L     B#16#7          // Formato doppio intero
      ==I
      SPB   dou5
      TAK
      L     B#16#8          // Formato real
      ==I
      SPB   rea5
      L     W#16#1          // Error code 0001 "type mismatch"
      T     #Ret_val
      NOT
      SAVE
      BEA
//
byt5: L     D [AR1,P#6.0]   // Load imput address
      LAR1
      L     #Smooth_val     // Smoothed value
      RND                   // Convert from real to integer
      T     B [AR1,P#0.0]   // Assign real value to output
      SPA   fin5
```

```
//
wor5: L     D [AR1,P#6.0]  // Load imput address
      LAR1
      L     #Smooth_val    // Smoothed value
      RND                  // Convert from real to integer
      T     W [AR1,P#0.0]  // Assign real value to output
      SPA   fin5
//
dou5: L     D [AR1,P#6.0]  // Load imput address
      LAR1
      L     #Smooth_val    // Smoothed value
      RND                  // Convert from real to integer
      T     D [AR1,P#0.0]  // Assign real value to output
      SPA   fin5
//
rea5: L     D [AR1,P#6.0]  // Load imput address
      LAR1
      L     #Smooth_val    // Smoothed value
      T     D [AR1,P#0.0]  // Assign real value to output
//
fin5: NOP   0
```

Segmento: 4 Force RLC = TRUE and save it in BIE register
This network set the RLC to TRUE and save it in BIE register

```
      L     0         // Error code reset
      T     #Ret_val
      SET             // Force RLC to TRUE
      SAVE            // RLC save in BIE register
```