```
diff --git a/CurrentInterface.c b/CurrentInterface.c
index 3a60d46..27b786b 100644
--- a/CurrentInterface.c
+++ b/CurrentInterface.c
@@ -29,13 +29,17 @@
 void       CurrentInterface(long lValue2Convert);
 char       CurrentInterface_Calibration(unsigned char chWhat2Do,float
 fTrueValue);

+unsigned char    CurrentInterface_EvaluateDeviation(unsigned char
bCurrentOn, float fDeviation, unsigned char *pbyDisableFeedbackCounter);
 void       CurrentInterface_FeedBack(void);
 float      CurrentInterface_GetCurrent(unsigned char byMode);
 char       CurrentInterface_GetFeedBackIntegralPortion(void);
 char       CurrentInterface_GetFeedBackProportionalPortion(void);
 char       CurrentInterface_GetFeedBackState(void);
-unsigned int   CurrentInterface_GetTimeHysteresis(void);
+unsigned int     CurrentInterface_GetTimeHysteresis(void);
 char       CurrentInterface_Ini(unsigned char byMode);
+
+void       CurrentInterface_Ini_2(void);
+
 void     CurrentInterface_LoadDACSettings(unsigned char chWhat2Load);
 void       CurrentInterface_PrepareCalibration(unsigned char
 byCalPoint,float fWeight);

@@ -342,6 +346,119 @@ char CurrentInterface_Calibration(unsigned char
chWhat2Do,float fTrueValue)
    }
    return 0;
 }
+/*~+:unsigned char     CurrentInterface_EvaluateDeviation(unsigned char
bCurrentOn, float fDeviation, unsigned char *pbyDisableFeedbackCounter)*/
+/*#LJ:CurrentInterface_EvaluateDeviation=15*/
+unsigned char CurrentInterface_EvaluateDeviation(unsigned char bCurrentOn,
float fDeviation, unsigned char *pbyDisableFeedbackCounter)
+{
+   /*~+:Beschreibung*/
+   /*!
+   \fn unsigned char CurrentInterface_EvaluateDeviation(unsigned char
bCurrentOn, float fDeviation, unsigned char *pbyDisableFeedbackCounter)
+
+   <b>Beschreibung:</b><br>
+   Diese Funktion wertet die Ist-/Sollstrom-Abweichung aus und schaltet bei
mehrfacher Grenzwertüberschreitung in den Sicherheitszustand. Der Aufruf
erfolgt im Rahmen der Ausführung der Funktion 'CurrentInterface_Feedback()',
welche ihrerseits nur durchlaufen wird, wenn ein neuer Wandlungswert vom ADC
der Rückkoppelspannung vorliegt
(ADuC836_ADCIsNewConversionValue(ADuC836_ADC_PRIMARY_TOGGLE)). Die Taktrate
des ADCs liegt bei 5.35Hz, was ein Aufrufinterval von etwa 190ms zur Folge
hat. Da nach 13facher (s. System.cnd =>
SYSTEMCND CURRENT DEVIATION COUNTER LIMIT) Erkennung einer Stromabweichung
in den Sicherheitszustand gewechselt wird, kann eine Reaktionszeit von unter
3s (genau: 2.5s) eingehalten werden, sofern das Vorzeichen der Abweichung
nicht wechselt.
+
+   \param
+   bCurrentOn: Es fließt ein Ausgangsstrom von über 1.5mA
+   \param
+   fDeviation: Ermittelte Stromabweichung in mA.
+   \param
```

```
+   pbyDisableFeedbackCounter: Zeiger auf den Feedback-Sperr-Zähler
+
+   \return
+   Status: 0: Bürde abgeklemmt, 1: Bürde angeklemmt, 2: Mehrfache
Stromabweichung erkannt
+
+   <b>Zugriff:</b><br>
+   [X] Öffentlich / [ ] Privat
+   */
+
+   /*~+:Variablendeklarationen*/
+   static char chDeviationSign = 0;      // Vorzeichen der Stromabweichung
+   unsigned char byRetVal;
+   /*~+:Variableninitialisierungen*/
+   byRetVal = 1; //
+   if (bCurrentOn != 0) // Strom fließt - Bürde angeschlossen
+   {
+      /*~+:Berücksichtige das Vorzeichen bei der Untersuchung der
Stromabweichung*/
+      if (fDeviation > g CurrentInterface.FeedBack.fMaxDeviation)
+      {
+         /* Positive Stromabweichung über Grenzwert erkannt */
+         if (chDeviationSign <= 0 )
+         {
+            /* Die letzte Stromabweichung war negativ oder es wurde zuvor
noch keine ausgewertet (=0) */
+            // Vorzeichen der Stromabweichung setzen
+            chDeviationSign = 1;
+            /* Stromabweichungszähler initialisieren */
+            g_CurrentInterface.FeedBack.byMaxDeviationCounter =
SYSTEMCND CURRENT DEVIATION COUNTER LIMIT - 1;
+         }
+         if (g CurrentInterface.FeedBack.byMaxDeviationCounter == 0)
+         {
+            /*~+:Stromabweichung lag mehrmals über dem Grenzwert*/
+            /* Stromabweichung lag mehrmals über dem Grenzwert */
+            // Sicherheitsfunktion aufrufen
+            Diagnosis_SecurityCurrentInterface(CURRENT_DEVIATION);
+            // Korrekturwert auf 0 setzen
+            ADuC836_DACSetCurrentDeviation(0);
+            // Flag für neue ermittelte Rückkoppelspannung löschen
+            g_CurrentInterface.FeedBack.Results.byNewMeasurement = 0;
+            // Mehrfache Stromabweichung erkannt
+            byRetVal = 2;
+         }
+         else
+         {
+            /* Zähler der Stromabweichungs-Erkennung dekrementieren */
+            g_CurrentInterface.FeedBack.byMaxDeviationCounter--;
+         }
+      }
+      else
+      {
+         if (-fDeviation > g_CurrentInterface.FeedBack.fMaxDeviation)
+         {
+            /* Negative Stromabweichung über Grenzwert erkannt */
+            if (chDeviationSign >= 0 )
+            {
+               /* Die letzte Stromabweichung war positiv oder es wurde
zuvor noch keine ausgewertet (=0) */
+               // Vorzeichen der Stromabweichung setzen
```

```
+               chDeviationSign = -1;
+               /* Stromabweichungszähler initialisieren */
+               g_CurrentInterface.FeedBack.byMaxDeviationCounter =
SYSTEMCND_CURRENT_DEVIATION_COUNTER_LIMIT - 1;
+            }
+            if (g_CurrentInterface.FeedBack.byMaxDeviationCounter == 0)
+            {
+               /*~+:Stromabweichung lag mehrmals über dem Grenzwert*/
+               /* Stromabweichung lag mehrmals über dem Grenzwert */
+               // Sicherheitsfunktion aufrufen
+               Diagnosis_SecurityCurrentInterface(CURRENT_DEVIATION);
+               // Korrekturwert auf 0 setzen
+               ADuC836_DACSetCurrentDeviation(0);
+               // Flag für neue ermittelte Rückkoppelspannung löschen
+               g_CurrentInterface.FeedBack.Results.byNewMeasurement = 0;
+               // Mehrfache Stromabweichung erkannt
+               byRetVal = 2;
+            }
+            else
+            {
+               /* Zähler der Stromabweichungs-Erkennung dekrementieren */
+               g_CurrentInterface.FeedBack.byMaxDeviationCounter--;
+            }
+         }
+         else
+         {
+            /* Keine Stromabweichung über Grenzwert erkannt -
Stromabweichungszähler initialisieren */
+            g_CurrentInterface.FeedBack.byMaxDeviationCounter =
SYSTEMCND_CURRENT_DEVIATION_COUNTER_LIMIT - 1;
+         }
+      }
+   }
+   else
+   {
+      /* Zähler der Stromabweichungs-Erkennung setzen - Da die Bürde
abgeklemmt war, einen Leerdurchlauf der Rückkopplung initieren */
+
+      g_CurrentInterface.FeedBack.byMaxDeviationCounter =
SYSTEMCND_CURRENT_DEVIATION_COUNTER_LIMIT - 1;
+
+      *pbyDisableFeedbackCounter = 1;
+      byRetVal = 0;
+   }
+   return(byRetVal);
+}
 /*~+:void      CurrentInterface_FeedBack(void)*/
 void CurrentInterface FeedBack(void)
 {
@@ -393,67 +510,49 @@ void CurrentInterface FeedBack(void)
                if (ADuC836_DACGetCurrentDeviationCorrectionState())
                {
                   // Stromrückführung ist eingeschaltet
-                  /*~+:Abweichung zu groß ?*/
-                  if (fabs(fDeviation) >
g_CurrentInterface.FeedBack.fMaxDeviation)  // Abweichung zu groß
+                  switch
(CurrentInterface_EvaluateDeviation((ActualRMW_Current.fFloat > 1.0),
fDeviation, &byDisableFeedbackCounter))
                   {
-#ifdef MIT_SICHERHEITSALARM_FUNKTION
```

```
-
-                     if (g_CurrentInterface.FeedBack.byMaxDeviationCounter
> 20)
-
-#else
-
-                     if ((g_CurrentInterface.FeedBack.byMaxDeviationCounter
> 20)||(SYSTEMSTATE != SYSTEM_ERROR_CURRENT_DEVIATION))
-
-#endif
+                     case 0: // Bürde abgeklemmt
                      {
-#ifndef SPEZIALVERSION_FUER_TESTSYSTEME
-#ifdef MIT SICHERHEITSALARM FUNKTION
-                        /*~+:Abweichung mehrmals hintereinander zu hoch*/
-                        // Sicherheitsfunktion aufrufen
-
Diagnosis SecurityCurrentInterface(CURRENT DEVIATION);
-                        // Korrekturwert auf 0 setzen
-                        ADuC836 DACSetCurrentDeviation(0);
-
Limit_SetLimitState(SYSTEM_ERROR_CURRENT_DEVIATION);
-#else
-                        /*~+:Abweichung mehrmals hintereinander zu hoch*/
-                        // Korrekturwert auf 0 setzen
-                        ADuC836_DACSetCurrentDeviation(0);
-
Limit_SetLimitState(SYSTEM_ERROR_CURRENT_DEVIATION);
-#endif
-#endif
-                        // Flag für neue ermittelte Rückkoppelspannung
löschen
-
g_CurrentInterface.FeedBack.Results.byNewMeasurement = 0;
-                        // Ausstieg
-                        return;
+                        ADuC836_DACClearCurrentDeviation();  /* Den im
DAC-Modul abgelegten Offset löschen und neu setzen, sobald eine Bürde
angeklemmt und die Stromabweichung kleiner 5mA ist. */
+                        break;
                      }
-                     else
+                     case 1: // Bürde angeklemmt - keine mehrfache
Stromabweichung erkannt
                      {
-
g_CurrentInterface.FeedBack.byMaxDeviationCounter++;
-                     }
-                  }
-                  else
-                  {
-                     /*~+:Abweichung nicht größer als ein vorgegebener
Grenzwert*/
-                     g_CurrentInterface.FeedBack.byMaxDeviationCounter = 0;
-#ifndef MIT_SICHERHEITSALARM_FUNKTION
-                     Limit_ClearLimitState(SYSTEM_ERROR_CURRENT_DEVIATION);
-#endif
-                  }
+                        /* Die nachfolgende Abfrage wurde notwendig, um
nicht beim Ab- und Anklemmen der Bürde in den Sicherheitszustand zu
gelangen. Ist die Bürde abgeklemmt, liefert das Stellglied die maximalste
```

```
Spannung und beim erneuten Anklemmen wird dadurch ein Iststrom ermittelt,
welcher weit weg vom Sollstrom liegt. Der Software-Regler korrigiert diese
große Abweichung nach. Zusätzlich wird auch das Stellglied nachregeln. Beide
Maßnahmen haben eine Überkompensation zur Folge und bewirken, dass die
Stromabweichung innerhalb der geforderten Zeit von 3s nicht ausgeregelt
werden kann. Darum ist es sinnvoll, erst bei einer Stromabweichung von z.Z.
kleiner 5mA den Softwareregler zu aktivieren. Darüber arbeitet nur das
Hardware-Stellglied. */
+                        if (fabs(fDeviation) <
SYSTEMCND_CURRENT_DEVIATION_DISABLE_FEEDBACK)
+                        {
+                           /* Die Stromabweichung ist kleiner
SYSTEMCND_CURRENT_DEVIATION_DISABLE_FEEDBACK (z.Z. 5mA). Jetzt kommt der
Software-Regler zum Einsatz. */
 #ifdef MIT_VARIABLEM_I_ANTEIL
-                   // Regelgröße bestimmen
-                   fIntegralPortion =
g CurrentInterface.FeedBack.chIntegralPortion;
+                           // Regelgröße bestimmen
+                           fIntegralPortion =
g_CurrentInterface.FeedBack.chIntegralPortion;

-                   fDeviation *= (fIntegralPortion * 0.01);
+                           fDeviation *= (fIntegralPortion * 0.01);
 #else
-                   // 50% - I-Anteil
-                   fDeviation *= 0.5;
+                           // 50% - I-Anteil
+                           fDeviation *= 0.5;
 #endif
-                   // Regelabweichung setzen
-                   ADuC836_DACSetCurrentDeviation(fDeviation);
+                           // Regelabweichung setzen
+                           ADuC836_DACSetCurrentDeviation(fDeviation);
+
+                           // Analog-Ausgang aktualisieren
+                           lValue2Convert =
g_CurrentInterface.DAC.lRatedDACOutput;
+                           ADuC836_DACConvert(&lValue2Convert,0);
+                        }
+                        else
+                        {
+                           /* Die Stromabweichung ist größer oder gleich
SYSTEMCND_CURRENT_DEVIATION_DISABLE_FEEDBACK (z.Z. 5mA). Jetzt kommt kommt
nur das (Hardware-) Stellglied zum Einsatz. */
+                           ADuC836_DACClearCurrentDeviation(); /* Den im
DAC-Modul abgelegten Offset löschen und mit dem nächsten Durchlauf des
Softwarereglers (Stromabweichung < 5mA) neu setzen. */
+                        }
+                        break;
+                     }
+                     default:
+                     {
+                        /* Mehrfache Stromabweichung erkannt */

-                   // Analog-Ausgang aktualisieren
-                   lValue2Convert = g_CurrentInterface.DAC.lRatedDACOutput;
-                   ADuC836_DACConvert(&lValue2Convert,0);
+                        return;
+                     }
+                  }
```

```
                }
             }
             else
@@ -499,7 +598,7 @@ char CurrentInterface_GetFeedBackState(void)
 {
    return ADuC836_DACGetCurrentDeviationCorrectionState();
 }
-/*~+:unsigned int  CurrentInterface_GetTimeHysteresis(void)*/
+/*~+:unsigned int    CurrentInterface_GetTimeHysteresis(void)*/
 unsigned int CurrentInterface_GetTimeHysteresis(void)
 {
    return ADuC836_DACGetLimitHysteresis(ADUC836_DAC_UPPER_LIMIT) /
    CURRENTINTERFACE_TIMETICS2SEC;
@@ -678,8 +777,45 @@ char CurrentInterface Ini(unsigned char byMode)
          return 1;
       }
    }
+   CurrentInterface Ini 2();
    return 0;
 }
+/*~+:void      CurrentInterface_Ini_2(void)*/
+/*#LJ:CurrentInterface_Ini_2=6*/
+void CurrentInterface_Ini_2(void)
+{
+   /*~+:Beschreibung*/
+   /*!
+   \fnvoid CurrentInterface_Ini_2(void)
+
+   <b>Beschreibung:</b><br>
+   Zusätzliche Initialisierung der Stromschnittstelle für V2.000
+
+   \return
+   ./.
+
+   <b>Zugriff:</b><br>
+   [ ] Öffentlich / [X] Privat
+   */
+   /*~+:Variablendeklarationen*/
+
+   /*~+:Variableninitialisierungen*/
+   g_CurrentInterface.FeedBack.byMaxDeviationCounter =
SYSTEMCND_CURRENT_DEVIATION_COUNTER_LIMIT - 1;
+
+   /*~+:Parameter für Rückkoppelzweig*/
+   /* Maximale Stromabweichung über den Verstärkungsfaktor der
Stromschnittstelle ermitteln */
+   /* Dieser liegt für eine 500kg-Zelle bei etwa 0.64 und für eine
1000kg-Zelle bei 0.32 */
+   if (ADuC836_DACGetGain(0) < 0.48)
+   {
+      /* 1000kg - Zelle */
+      g CurrentInterface.FeedBack.fMaxDeviation =
SYSTEMCND_MAX_CURRENT_DEVIATION_1000KG;
+   }
+   else
+   {
+      /* 500kg - Zelle */
+      g_CurrentInterface.FeedBack.fMaxDeviation =
SYSTEMCND_MAX_CURRENT_DEVIATION_500KG;
+   }
+}
```

```
/*~+:void      CurrentInterface LoadDACSettings(unsigned char
chWhat2Load)*/
void CurrentInterface LoadDACSettings(unsigned char chWhat2Load)
{
```