

**Wichtige Hinweise:**

1. Bewahren Sie diese Betriebsanleitung an einem für den Benutzer sicheren und zugänglichen Platz auf.
2. Lesen Sie diese Gebrauchsanweisung sorgsam, bevor Sie die Software einsetzen und in Betrieb nehmen.

**Voraussetzungen zum Betrieb der gelieferten Software:**

- PC mit Festplatte (mind. 1GB), USB2.0-Anschluß
- Windows® XP und höher (32 Bit Version)
- Delphi 10.1 Berlin oder Delphi 7 © installiert auf dem PC

Die MCP2221.dcu und die extra mitgelieferte Software wurden in Delphi 7 und Delphi 10.1 Berlin geschrieben.

***Hinweis zu den Delphi-Versionen:***

***Diese Softwareroutinen sind mit und für Delphi 7 und Delphi 10.1 Berlin geschrieben worden.***

## Inhaltsverzeichnis

<b>Thema</b>	<b>Seite</b>
<b>Allgemeine Beschreibung</b>	4
<b>Technische Zusammenhänge</b>	4
<b>Vorteile der gelieferten Software MCP2221.dcu</b>	5
<b>Lieferumfang</b>	6
<b>Installation und Inbetriebnahme der Software</b>	6
<b>Beschreibung der einzelnen Prozeduren</b>	7
<b>DLL Initialisation</b>	7
DLLStart	7
<b>USB</b>	7
Get_VID_PID	7
Get_PowerSource	7
Get_Remote_Wakeup_Enable	8
Get_Current_Req	8
Get_Str_Manuf	8
Get_Str_Descr	8
Get_Ser_Number	9
Get_Fac_SerNo	10
Get_Ser_Num_En_Enable	10
Get_Firmware_Version	10
Get_Hardware_Version	11
Get_Connection_Status	11
Get_Dev_Count	11
Set_VID_PID	12
Set_PowerSource	12
Set_Remote_Wakeup_Enable	13
Set_Current_Req	13
Set_Ser_Num_En_Enabled	14
<b>Interrupt Pin &amp; GP Clock</b>	14
Get_Interrupt_Pin_Mode	14
Get_Clock_Pin_Divider_Value	15
Get_Clock_Pin_Duty_Cycle	15
Set_Interrupt_Pin_Mode	16
Set_Clock_Pin_Configuration	16
<b>GPIO Configuration</b>	17
Get_GP_Pin_Configuration	17
Get_Gp_Pin_Direction	18
Set_GP_Pin_Configuration	18
Set_Gp_Pin_Direction	19
<b>DAC &amp; ADC</b>	19
Get_Dac_Voltage_Reference	19
Get_Dac_Value	19
Get_Adc_Voltage_Reference	20
Set_Dac_Voltage_Reference	20
Set_Dac_Value	20
Set_Adc_Voltage_Reference	21
<b>Initial Pin Values</b>	21
Get_Initial_Pin_Value_LedUartTx	21
Get_Initial_Pin_Value_LedUartRx	22
Get_Initial_Pin_Value_LedI2C	22
Get_Initial_Pin_Value_Sspnd	22
Get_Initial_Pin_Value_Usbcfg	23
Set_Initial_Pin_Value_LedUartTx	23
Set_Initial_Pin_Value_LedUartRx	23
Set_Initial_Pin_Value_LedI2C	24
Set_Initial_Pin_Value_Sspnd	24
Set_Initial_Pin_Value_Usbcfg	24

<b>Device Operations, Functions</b>	25
Reset_Device	25
Get_Adc_Data	25
Clear_Interrupt_Pin_State	26
Read_GPIO_PinValue	26
Write_Gpio_PinValue	26
<b>I2C Options</b>	27
Read_I2c_Data	27
Write_I2c_Data	27
Stop_I2c_Data_Transfer	28
<b>Allgemeine Hinweise zum MCP2221 am I2C-Bus</b>	28
<b>Beschreibung der Im Quelltext mitgelieferten Delphi 7 Software Project1.pas</b>	29
<b>USB-Überwachung</b>	31
<b>Anhang: RS232-Schnittstelle</b>	32
<b>Impressum, Copyrights</b>	33

## **Allgemeine Beschreibung:**

Die von Microchip® gelieferten dll Dateien liefern dem Anwender eine Fülle von Möglichkeiten, den Baustein MCP2221 über Software an zu sprechen und zu betreiben. Jedoch bieten die dll einige Nachteile.

Laut Aussage von Microchip wurden die dll in C++ geschrieben und sind somit für C++ Programmierer gedacht. Eine Verwendung mit Delphi® wurde seitens Microchip nie beabsichtigt. Entsprechend treten folgende Probleme auf, wenn man Delphi 7 und die dll für MCP2221 verwendet:

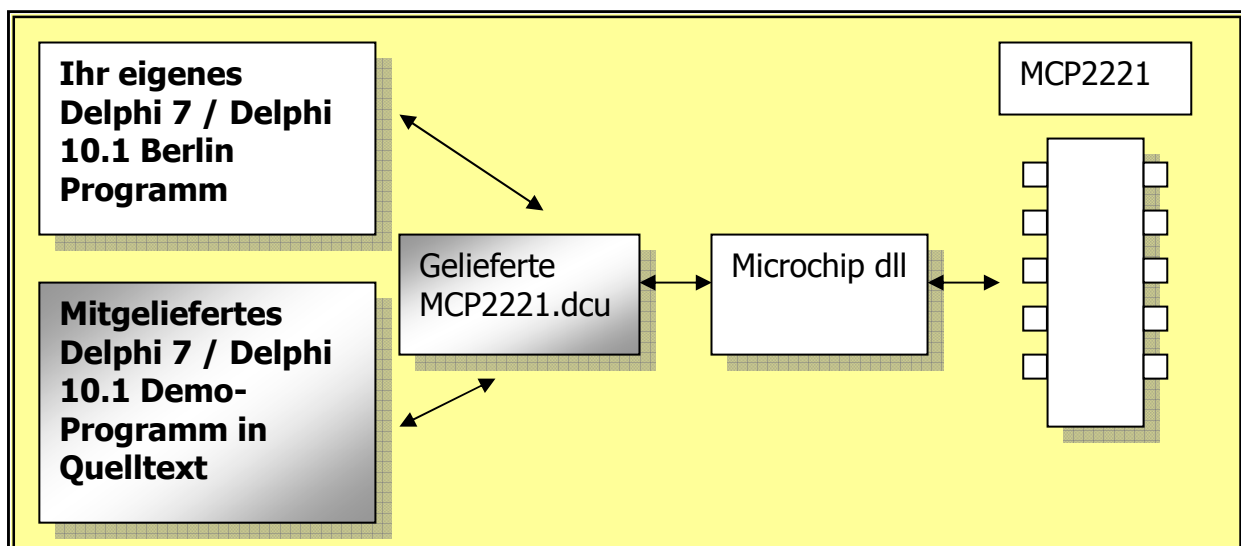
1. Teilweise komplizierte Konstruktoren (so genannte Handles), die den Zugriff auf bestimmte Funktionen erschweren.
2. Manche Parameter sind im UNICODE-Format zu übergeben. Delphi 7 kennt aber kein UNICODE-Format
3. Es sind unterschiedliche Parametertypen zu berücksichtigen je nach Funktion: Char, Integer, Arrays etc.

Auch der exzellente Microchip-Support konnte hierzu keine Lösung bieten.

## **Technische Zusammenhänge:**

Folgende Zusammenhänge sind vorhanden:

1. Die von Microchip mitgelieferten dll erlauben den Zugriff auf den Microchip Baustein MCP2221.
2. Die von uns gelieferte MCP2221.dcu ist die Schnittstelle zwischen dem Delphi - Anwender / Programmierer und der MCP2221-dll



Das zusätzlich zur MCP2221.dcu in Quelltext mitgelieferte Delphi 7 bzw. delphi 10.1 Berlin Programm Zeigt dem Anwender die Verwendung der dcu anhand von Beispielen zu jeder Funktion der dcu.

**Vorteile der gelieferten Software MCP2221.dcu:**

1. Alle von der MCP2221.dcu verwendeten Variablen sind vom Typ „string“. Der Anwender braucht sich um die verschiedenen Parametertypen keine Sorgen mehr zu machen. Ob Sie eine Variable an den MCP2221-Baustein senden oder von ihm empfangen, diese Variablen sind immer vom Typ „string“. Diese Vorgehensweise vereinfacht die Programmierung erheblich.
2. Die Kommunikation über die MCP2221.dcu geschieht immer über Prozeduren (keine Funktionen). Auch diese Art vereinfacht die Programmierung.
3. Die MCP2221.dcu prüft intern, ob die übergebenen Parameter korrekt sind. Ein Absturz der selbst geschriebenen Software wird dadurch minimiert.
4. Keine Softwareinstallation nötig: man kopiert lediglich die MCP2221.dcu ins Delphi-Arbeitsverzeichnis und hat zugriff auf alle Prozeduren der dcu.
5. Es werden zwei Varianten der MCP2221.dcu geliefert. Eine Voll- und eine Demo-Version. Damit kann auch der Anwender zwei Varianten seiner Software an den Endkunden weiter geben. Der Unterschied zwischen den beiden Varianten besteht darin, dass die Demo-Variante ab und zu Meldungen auf den Bildschirm bringt, die Vollversion hingegen nicht.
6. Alle Funktionen stehen in komplettem Umfang auch in der Demo-Variante zur Verfügung. Die Demo-Variante besitzt also – im vergleich zur Vollvariante – denselben Funktionsumfang ohne Einschränkungen. Sie können also die Demo-Variante in aller Ruhe testen und später die Vollversion erwerben.
7. Die zusätzlich mitgelieferte Software ist ein Delphi Vollprogramm im Quelltext, das die einzelnen Prozeduren der MCP2221.dcu zeigt. Man kann diese Quelltexte 1:1 in sein Delphi-Programm übernehmen.
8. Die MCP2221.dcu in der Demo-Variante und die Quelltexte zum Test aller Prozeduren der dcu sind frei verfügbar. Die Vollvariante der MCP2221.dcu ist kostenpflichtig.

## **Lieferumfang:**

### Vollversion:

- MCP2221.dcu
- Vollumfang Delphi 7 bzw. Delphi 10.1 Berlin Software (in Quelltextform) zur Verdeutlichung des Funktionsumfangs der MCP2221.dcu

### Demo-Version

- MCP2221.dcu als Demo. Kompletter Funktionsumfang wie bei der Vollversion. In der Demo-Version erscheint ab und an ein Meldungsfenster im Display.
- Vollumfang Delphi 7 bzw. Delphi 10.1 Berlin Software (in Quelltextform) zur Verdeutlichung des Funktionsumfangs der MCP2221.dcu

Zu beiden Versionen wird eine pdf-Bedienungsanleitung geliefert.

Zusätzlich wird die Microchip dll gesendet, damit Sie nicht suchen müssen bzw. damit Sie nicht die falsche dll verwenden (Microchip stellt mehrere dll für den Baustein MCP2221 zur Verfügung).

Ebenfalls wird der Treiber für den Baustein MCP2221 mitgeliefert. Auch dies erspart Ihnen die Suche auf der Microchip-Webseite.

## **Installation und Inbetriebnahme der Software:**

Eine Installation der Software ist nicht nötig.

### Demo-Version:

Kopieren Sie alle mitgelieferten Dateien in ein Verzeichnis Ihrer Wahl. Starten Sie Delphi und öffnen Sie die Projektdatei Project1.dpr aus dem gewählten Arbeitsverzeichnis. Sie können nun alle Prozeduren testen und sich deren Art im Quelltext ansehen.

### Vollversion:

Kopieren Sie die Dateien MCP2221.dcu und MCP2221DLL-UM\_x86.dll in Ihr Delphi Arbeitsverzeichnis.

Starten Sie Delphi und binden Sie die unit MCP2221 ein. Anschließend stehen Ihnen alle Prozeduren zur Kommunikation mit dem MCP2221-Baustein zur Verfügung.

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,  
Dialogs, StdCtrls, ComCtrls, Grids, **MCP2221, XPMAN**;

Die obige Einbindung der unit XPMAN ist optional. Sie verleiht der Oberfläche von Form1 sowie aller enthaltenen Buttons lediglich ein zeitgemäßes Äußeres. Das einbinden der Unit MCP2221 in Ihrem Hauptprogramm ist hingegen zwingend notwendig.

In allen Fällen müssen Sie die Microchip-Treiber des MCP2221 auf Ihrem PC installieren!

## **Beschreibung der einzelnen Prozeduren:**

Entscheidend für den Anwender sind die von der MCP2221.dcu zurück gelieferten Werte. Zu jeder Prozedur werden ein oder zwei Variablen des Typs string zurück geliefert bzw. Variablen vom Typ string an die Prozedur übergeben. Anhand der Überprüfung der zurück gelieferten Werte kann Ihr Programm Entscheidungen treffen, ob eine Prozedur korrekt oder inkorrekt ausgeführt wurde.

## **// DLL Initialisation //**

Prozedurname	DLLStart(s1,s2);
Funktion	Initialisiert die dll
Zu übergebende Parameter	Keine
Zurückgelieferte string Parameter	s1, s2

Die Prozedur ist vor jedem Start einmalig aus zu führen. Ist die Microchip dll korrekt initialisiert, erhalten s2 den Wert „0“ und s1 den Wert „OK“. Ansonsten bekommt s1 den Wert „FAIL“, s2 erhält einen negativen Wert.

---

## **// USB //**

Prozedurname	Get_VID_PID(VID, PID, s);
Funktion	Ermittelt die VID und PID des MCP2221
Zu übergebende Parameter	Keine
Zurückgelieferte string Parameter	VID, PID, s

Fehlerlos erhält String s den Wert 0, VID und PID werden als Strings im Hex-Format zurückgegeben.  
Ansonsten erhält s einen negativen Wert und VID und PID = „FAIL“.

---

Prozedurname	Get_PowerSource(power, powRet);
Funktion	Ermittelt die Art der Versorgungsspannung
Zu übergebende Parameter	Keine
Zurückgelieferte string Parameter	Pow, powRet

Ist pow = 0, dann ist powRet = „Bus-Powered“.  
Ist pow = 1, dann ist powRet = „Self-Powered“.  
Ist pow < 0 dann ist powRet = „FAIL“.

---

Prozedurname	Get_Remote_Wakeup_Enable(RWE,RWEret);
Funktion	Ermittelt das Remote Wakeup
Zu übergebende Parameter	Keine
Zurückgelieferte string Parameter	RWE, RWEret

Ist RWE = 0, dann ist RWEret = "Disabled".

Ist RWE = 1, dann ist RWEret = „Enabled“.

Ist RWE < 0, dann ist RWEret = „FAIL“.

---

Prozedurname	Get_Current_Req(Cur, CurRes);
Funktion	Ermittelt die Stromstärke (in mA) der USB-Versorgung
Zu übergebende Parameter	Keine
Zurückgelieferte string Parameter	Cur, CurRes

Ist Cur >= 0, wird die Stromstärke angegeben und CurRes bekommt den Wert „mA“. Ist Cur < 0, so erhält CurRes den Wert „FAIL“.

---

Prozedurname	Get_Str_Manuf(Manufact, ManRes);
Funktion	Gibt den Herstellernamen an.
Zu übergebende Parameter	Keine
Zurückgelieferte string Parameter	Manufact, ManRes

Ist ManRes >= 0, so ergibt Manufact den Herstellernamen.

Im Fehlerfalle ist ManRes <0.

---

Prozedurname	Get_Str_Descr(Desc, DesRes);
Funktion	Ermittelt den Bausteinnamen
Zu übergebende Parameter	Keine
Zurückgelieferte string Parameter	Desc, SesRes

Im übergeordneten Programm muß eine Variable vom Typ Desc: string[52] definiert werden.

DesRes >= 0 bewirkt, dass die Variable Desc den Wert

„MCP2221 USB-I2C/UART Combo“ annimmt, was dem Bausteintyp entspricht.

Erhält DesRes den Wert < 0, so ist ein Fehler eingetreten und die Variable Desc stellt sich aus undefinierten Zeichen zusammen.



Beispiel:

```
procedure TForm1.Button1Click(Sender: TObject);
var Desc: string[52];
    DesRes: string;
    i: integer;
begin
    Get_Str_Descr(Desc, DesRes);
    Edit1.Text:= ''; // Textfeld leeren
    for i:= 0 to 51 do
    begin
        Edit1.Text:= Edit1.Text + Desc[i]; //Textfeld mit Descriptor füllen
    end;
    Edit2.Text:= DesRes; //Antwort als Zahl anzeigen
end;
```

---

Prozedurname	Get_Ser_Number(Ser, SerRes);
Funktion	Ermittelt die Seriennummer des Bausteins
Zu übergebende Parameter	Keine
Zurückgelieferte string Parameter	Ser, SerRes

Im übergeordneten Programm muß eine Variable vom Typ Ser: string[20] definiert werden.

Ist SerRes > 0 erhält die Variable Ser die Seriennummer des MCP2221, ansonsten  
Stellt sich die Variable Ser aus undefinierten Zeichen zusammen.

Beispiel:

```
procedure TForm1.Button1Click(Sender: TObject);
var Ser: string[20];
    SerRes: string;
    i: integer;
begin
    Get_Ser_Number(Ser, SerRes);
    Edit1.Text:= ''; // Textfeld leeren
    for i:= 0 to 19 do
    begin
        Edit1.Text:= Edit1.Text + ser[i]; //Textfeld mit Seriennummer füllen
    end;
    Edit2.Text:= SerRes; //Antwort als Zahl anzeigen
end;
```

---

Prozedurname	Get_Fac_SerNo(FSer, FSerRes);
Funktion	Ermittelt die Werks-Seriennummer des Bausteins
Zu übergebende Parameter	Keine
Zurückgelieferte string Parameter	FSer, FSerRes

Gleiches Verhalten wie die Funktion zuvor (Get\_Ser\_Number(Ser, SerRes);

---

Prozedurname	Get_Ser_Num_En_Enable(SNEE, SNEEres);
Funktion	Ermittelt die Seriennummer-Enumeration
Zu übergebende Parameter	Keine
Zurückgelieferte string Parameter	SNEE, SNEEres

Für SNEE = 0, erhält SNEEres den Wert "Disabled".

Für SNEE = 1, erhält SNEEres den Wert "Enabled".

Für SNEE < 0, erhält SNEEres den Wert "FAIL".

---

Prozedurname	Get_Firmware_Version(FV, FiVeRes1, FiVeRes2);
Funktion	Ermittelt die Version der Firmware
Zu übergebende Parameter	Keine
Zurückgelieferte string Parameter	FV, FiVeRes1, FiVeRes2

Parameter FV muß im übergeordneten Programm vom Typ „array[1..4] of Char“ definiert werden. Die Abfrage / Darstellung des Parameters kann jedoch als string definiert werden (s. nachfolgendes Beispiel).

Variable FiVeRes1 enthält eine Zahl.

Ist FiVeRes1 >= 0, dann erhält FiVeRes2 den Wert "OK" und FV zeigt die Firmware Versionsnummer an.

Ist FiVaRes1 < 0, erhält FiVaRes2 den Wert „FAIL“ und FV bleibt leer.

Beispiel:

```
procedure TForm1.Button1Click(Sender: TObject);
var FV: array[1..4] of Char;
    FiVeRes1, FiVeRes2: string;
begin
  Get_Firmware_Version(FV, FiVeRes1, FiVeRes2);
  Edit1.Text:= FV;
  Edit2.Text:= FiVeRes1;
  Edit3.Text:= FiVeRes2;
end;
```

---

Prozedurname	Get_Hardware_Version(HV, HaVeRes1, HaVeRes2);
Funktion	Ermittelt die Version der Hardware
Zu übergebende Parameter	Keine
Zurückgelieferte string Parameter	HV, HaVeRes1, HaVeRes2

Gleiche Funktionsweise wie die Prozedur zuvor namens „Get\_Firmware\_Version“.

Auch hier ist zu beachten, dass eine Variable HV als „array[1..4] of Char“ definiert werden muß, die jedoch im Nachhinein als ein string abgefragt werden kann.

---

Prozedurname	Get_Connection_Status(ConStat, CSRes);
Funktion	Gibt an, ob eine Verbindung zwischen PC und MCP2221 existiert
Zu übergebende Parameter	Keine
Zurückgelieferte string Parameter	ConStat, CSRes

Existiert eine Verbindung zwischen PC und MCP2221, so wird die Variable CSRes als „0“ übergeben und ConStat wird zu „CON“. Andernfalls wird CSRes mit „-66“ antworten und ConStat wird zu „NO CON“.

---

Prozedurname	Get_Dev_Count(s1);
Funktion	Gibt die Anzahl der angeschlossenen MCP2221 zurück
Zu übergebende Parameter	Keine
Zurückgelieferte string Parameter	S1

S1 enthält die Anzahl der MCP2221, die am USB-Bus angeschlossen sind.



**ACHTUNG:** zuvor ist die Prozedur „Get\_Connection\_Status“ auf zu rufen!

---

Prozedurname	Set_VID_PID(vid, pid, result1, result2)
Funktion	Setzt die neuen VID und PID des MCP2221
Zu übergebende Parameter	vid, pid
Zurückgelieferte string Parameter	result1, result2

**Übergabeparameter:**

Die neuen vid und pid sind als string im Hex-Format der Prozedur zu übergeben.

Antworten der Prozedur:

**Im Erfolgsfall:**

result1 = „0“ und result2 = „OK“.

**Im Fehlerfall:**

result1 = „-4“ und result2 = „FAIL“, wenn ein allgemeiner Fehler auftritt.

result 1 = „-66“ und result 2 = „No HEX“, wenn keine Hexzahl eingegeben wurde.

---

Prozedurname	Set_PowerSource(ps, power, powRet);
Funktion	Setzt die Art der Spannungsversorgung.
Zu übergebende Parameter	ps
Zurückgelieferte string Parameter	power, powRet

**Übergabeparameter:**

Über die variable ps wird die neue Art der Spannungsversorgung gesetzt. Korrekte Werte sind 0 oder 1.

**Antworten der Prozedur:**

**Im Erfolgsfall:**

Ist ps = „0“, wird power zu „0“ und powerRet = „Bus-Powered“.

Ist ps = „1“, wird power zu „0“ und powerRet = „Self-Powered“.

**Im Fehlerfall:**

Im Falle eines Fehlers wird power zu „-4“ und powerRet zu „FAIL“, oder power = „-66“ und powerRet = „No HEX“, wenn keine Hexzahl über die Variable ps eingegeben wurde.

---

Prozedurname	Set_Remote_Wakeup_Enable(RWE, RWEret1, RWEret2);
Funktion	Setzt die Art des Remote Wakeup
Zu übergebende Parameter	RWE
Zurückgelieferte string Parameter	RWEret1, RWEret2

**Übergabeparameter:**

RWE als 0 oder 1

**Antworten der Prozedur:**

**Im Erfolgsfall:**

Ist RWE als „0“ übergeben worden, wird RWEret1 zu „0“ und RWEret2 zu „Disabled“.

Ist RWE als „1“ übergeben worden, wird RWEret1 zu „0“ und RWEret2 zu „Enabled“.

**Im Fehlerfall:**

RWEret1 = „-1“ und RWEret2 = „FAIL“ im Fehlerfalle oder RWEret1 = „-66“ und RWEret2 = „No HEX“, wenn keine positive Ganzzahl für RWE übergeben wurde.

---

Prozedurname	Set_Current_Req(NewCur, NewCurRes1, NewCurRes2);
Funktion	Setzt den neuen Stromwert in mA für die USB-Spannungsversorgung
Zu übergebende Parameter	NewCur
Zurückgelieferte string Parameter	NewCurRes1, NewCurRes2

**Übergabeparameter:**

NewCur als mA-Wert.

**Antworten der Prozedur:**

**Im Erfolgsfall:**

NewCurRes1 = „0“ und NewCurRes2 = „mA“

**Im Fehlerfall:**

NewCurRes1 = „-201“ und NewCurRes2 = „FAIL“, wenn der mA-Wert nicht akzeptiert wurde.

NewCurRes1 = „-66“ und NewCurRes2 = „No HEX“, wenn keine positive Ganzzahl für NewCur übergeben wurde.

---

Prozedurname	Set_Ser_Num_En_Enabled(NewSNEE, NewSNEERes1, NewSNEERes2);
Funktion	Setzt die neue Serial Number Enumeration
Zu übergebende Parameter	NewSNEE
Zurückgelieferte string Parameter	NewSNEERes1, NewSNEERes2

**Übergabeparameter:**

NewSNEE als "0" oder "1".

**Antworten der Prozedur:**

**Im Erfolgsfall:**

NewSNEERes1 = "0" und NewSNEERes2 = "Disabled", wenn NewSNEE als „0“ übergeben wurde.

NewSNEERes1 = "0" und NewSNEERes2= "Enabled", wenn NewSNEE als „1“ übergeben wurde.

**Im Fehlerfall:**

NewSNEERes1 = "-1" und NewSNEERes2= "FAIL", wenn ein allgemeiner Fehler aufgetreten ist.

NewSNEERes1 = "-66" und NewCurRes2 = "No HEX", wenn keine positive Ganzzahl für NewSNEE übergeben wurde.

---

## **// Interrupt pin and GP Clock //**

Prozedurname	Get_Interrupt_Pin_Mode(witoget, Resp1, Resp2);
Funktion	Gibt den Interrupt Pinmode zurück
Zu übergebende Parameter	witoget
Zurückgelieferte string Parameter	Resp1, Resp2

**Übergabeparameter:**

Witoget wie folgt:

Current setting = 0, Power-up default = 1

**Antworten der Prozedur:**

**Im Erfolgsfall:**

Resp1 > 0 und Resp2 = „OK“

**Im Fehlerfall:**

Resp1 = "-4" oder = „-201“ und Resp 2= "FAIL", wenn ein allgemeiner Fehler aufgetreten ist.

Resp1 = "-66" und Resp2 = "No HEX", wenn keine positive Ganzzahl für witoget übergeben wurde.

---

Prozedurname	Get_Clock_Pin_Divider_Value(wtg, Resp1, Resp2, Resp3);
Funktion	Divisorwert für den Clock-Ausgang
Zu übergebende Parameter	wtg
Zurückgelieferte string Parameter	Resp1, Resp2, Resp3

**Übergabeparameter:**

wtg wie folgt:

Current setting = 0, Power-up default = 1

**Antworten der Prozedur:**

**Im Erfolgsfall:**

Resp1 = „0“ und Resp2 = „OK“. Resp3 enthält den Divisorwert für den Clock-Ausgang.

**Im Fehlerfall:**

Resp1 = „-4“ oder = „-201“ und Resp 2= „FAIL“, wenn ein allgemeiner Fehler aufgetreten ist.  
Resp3 wird zum Leerstring.

Resp1 = „-66“ und Resp2 = „No HEX“, wenn keine positive Ganzzahl für wtg übergeben wurde.

---

Prozedurname	Get_Clock_Pin_Duty_Cycle(wtg, Resp1, Resp2, Resp3);
Funktion	Gibt das Tastverhältnis des Clock-Ausgangs an
Zu übergebende Parameter	wtg
Zurückgelieferte string Parameter	Resp1, Resp2, Resp3

**Übergabeparameter:**

wtg wie folgt:

Current setting = 0, Power-up default = 1

**Antworten der Prozedur:**

**Im Erfolgsfall:**

Resp1 = „0“ und Resp2 = „OK“. Resp3 enthält das Tastverhältnis am Clock-Ausgang.

**Im Fehlerfall:**

Resp1 = „-4“ oder = „-201“ und Resp 2= „FAIL“, wenn ein allgemeiner Fehler aufgetreten ist.  
Resp3 wird zum Leerstring.

Resp1 = „-66“ und Resp2 = „No HEX“, wenn keine positive Ganzzahl für wtg übergeben wurde.

---

Prozedurname	Set_Interrupt_Pin_Mode(wtg, imts, resp1, resp2);
Funktion	Setzt den Interrupt Pinmode
Zu übergebende Parameter	Wtg, imts
Zurückgelieferte string Parameter	Resp1, Resp2

**Übergabeparameter:**

wtg wie folgt:

Current setting = 0, Power-up default = 1

Imts wie folgt:

1 = steigende Flanke, 2 = fallende Flanke, 3 = beide

**Antworten der Prozedur:**

**Im Erfolgsfall:**

Resp1 = „0“ und Resp2 = „OK“.

**Im Fehlerfall:**

Resp1 = „-4“ oder = „-202“ und Resp 2= „FAIL“, wenn ein allgemeiner Fehler aufgetreten ist.

Resp1 = „-66“ und Resp2 = „No HEX“, wenn keine positive Ganzzahl für wtg und / oder imts übergeben wurde.

---

Prozedurname	Set_Clock_Pin_Configuration(wtg, CDiv, DCycl, Resp1, Resp2);
Funktion	Setzt die Konfiguration des Clock-Ausgangs
Zu übergebende Parameter	wtg, CDiv, DCycl
Zurückgelieferte string Parameter	Resp1, Resp2

**Übergabeparameter:**

wtg wie folgt:

Current setting = 0, Power-up default = 1

Cdiv wie folgt:

3 = 75%, 2 = 50%, 1 = 25%, 0 = 0%

DCycl wie folgt:

1 = steigende Flanke, 2 = fallende Flanke, 3 = beide

**Antworten der Prozedur:**

**Im Erfolgsfall:**

Resp1 = „0“ und Resp2 = „OK“.

**Im Fehlerfall:**

Resp1 = „-4“ oder = „-202“ und Resp 2= „FAIL“, wenn ein allgemeiner Fehler aufgetreten ist.

Resp1 = „-66“ und Resp2 = „No HEX“, wenn keine positive Ganzzahl für wtg und / oder CDiv bzw. DCycl übergeben wurde.

---



## // GPIO Configuration //

Prozedurname	Get_GP_Pin_Configuration(wtg, PinDe, PinDi, PinVa, s1, s2);
Funktion	Ermittelt die Konfiguration von GP0..GP3
Zu übergebende Parameter	wtg
Zurückgelieferte string Parameter	PinDe, PinDi, PinVa, s1, s2

### Übergabeparameter:

Wtg als „0“ oder „1“

### Antworten der Prozedur:

Um eine korrekte Antwort zu erhalten, müssen die Variablen PinDe, PinDi PinVa des Typs „array of string“ sein. Besser ist es, diese Variablen als vom Typ „PiCo“ (PinConfiguration) zu definieren, da die MCP2221.dcu diese Felder intern korrekt definiert und diese dem Anwender zur Verfügung stellt. Hier ein Beispiel der korrekten Variablendefinition:

```
var wtg, s1,s2: string;  
    PinDe, PinDi, PinVa: PiCo;
```

### Im Erfolgsfall:

s1 = „0“ und s2 = „OK“.

Zusätzlich erhalten PinDe[0..3], PinDi[0..3] und PinVa[0..3] die Informationen über die GPIO-Zustände. Die gewählten Namen stehen für Pin designation (Bestimmung, Typ), Pin direction (Richtung), Pin value (Wert) des jeweiligen Pins.

### Im Fehlerfall:

s1 = „-66“ und s2 = „No HEX“, wenn keine positive Ganzzahl für wtg und übergeben wurde.

s1 = „-4“ oder = „-201“ und s2= „FAIL“, wenn ein allgemeiner Fehler aufgetreten ist.

In allen Fällen werden dann PinDe[0..3], PinDi[0..3] und PinVa[0..3] zu Leerstrings.

---

Prozedurname	Get_Gp_Pin_Direction(wtg, pinNum, s1, s2);
Funktion	Fragt die Richtung einzelner Pins ab
Zu übergebende Parameter	wtg, pinNum
Zurückgelieferte string Parameter	s1, s2

Diese Prozedur kann nur dann angewendet werden, wenn der ab zu fragende Pin zuvor als GPIO konfiguriert wurde!

**Übergabeparameter:**

Wtg als „0“ oder „1“

**Antworten der Prozedur:**

**Im Erfolgsfall:**

s1 = „0“ und s2 = „Output“ oder s1 = „1“ und s2 = „Input“.

**Im Fehlerfall:**

s1 = „-66“ und s2 = „No HEX“, wenn keine positive Ganzzahl für wtg und übergeben wurde.

s1 = „-4“ oder = „-201“ und s2= „FAIL“, wenn ein allgemeiner Fehler aufgetreten ist.

---

Prozedurname	Set_GP_Pin_Configuration(wtg, PinDe, PinDi, PinVa, s1, s2);
Funktion	Setzt die Konfiguration von GP0..GP3
Zu übergebende Parameter	Wtg, PinDe, PinDi, PinVa
Zurückgelieferte string Parameter	s1, s2

**Übergabeparameter:**

Wtg als „0“ oder „1“

PinDe, PinDi, PinVa jeweils von 0..3 als Strings.

**Antworten der Prozedur:**

Um eine korrekte Antwort zu erhalten, müssen die Variablen PinDe, PinDi PinVa des Typs „array of string“ sein. Besser ist es, diese Variablen als vom Typ „PiCo“ (PinConfiguration) zu definieren, da die MCP2221.dcu diese Felder intern korrekt definiert und diese dem Anwender zur Verfügung stellt. Hier ein Beispiel der korrekten Variablendefinition:

```
var wtg, s1,s2: string;  
    PinDe, PinDi, PinVa: PiCo;
```

**Im Erfolgsfall:**

s1 = „0“ und s2 = „OK“.

**Im Fehlerfall:**

s1 = „-66“ und s2 = „No HEX“, wenn keine positive Ganzzahl für wtg und übergeben wurde.

s1 = „-4“ oder = „-201“ und s2= „FAIL“, wenn ein allgemeiner Fehler aufgetreten ist.

---

Prozedurname	Set_Gp_Pin_Direction(wtg, PiNum, PiDir, s1, s2);
Funktion	Setzt die Richtung einzelner Pins
Zu übergebende Parameter	wtg, PiNum, PiDir
Zurückgelieferte string Parameter	s1, s2

**Übergabeparameter:**

Wtg als „0“ oder „1“ oder „2“, PiNum als „0“ bis „3“ und PiDir als „0“ oder „1“

**Antworten der Prozedur:**

**Im Erfolgsfall:**

s1 = „0“ und s2 = „OK“

**Im Fehlerfall:**

s1 = „-66“ und s2 = „No HEX“, wenn keine positive Ganzzahl für wtg und übergeben wurde.

s1 = „-4“ oder = „-201“ und s2= „FAIL“, wenn ein allgemeiner Fehler aufgetreten ist.

---

## // DAC & ADC //

DAC = Digital to Analog Converter

ADC = Analog to Digital Converter

Prozedurname	Get_Dac_Voltage_Reference(s1, s2);
Funktion	Ermittelt die Referenzspannung desDAC
Zu übergebende Parameter	Keine
Zurückgelieferte string Parameter	s1, s2

**Übergabeparameter:**

**Antworten der Prozedur:**

**Im Erfolgsfall:**

s1 = „0“ bis „3“

s2 = Vdd (default) oder 1.024V oder 2.048V oder 4.096V

**Im Fehlerfall:**

s1 = „-4“ und s2= „FAIL“, wenn ein allgemeiner Fehler aufgetreten ist.

---

Prozedurname	Get_Dac_Value(s1, s2);
Funktion	Ermittelt den aktuellen DAC-Wert
Zu übergebende Parameter	Keine
Zurückgelieferte string Parameter	s1, s2

**Übergabeparameter:**

**Antworten der Prozedur:**

**Im Erfolgsfall:**

s1 = als DAC-Wert

s2 = „OK“

**Im Fehlerfall:**

s1 = „-4“ und s2= „FAIL“, wenn ein allgemeiner Fehler aufgetreten ist.

---

Prozedurname	Get_Adc_Voltage_Reference(s1, s2);
Funktion	Ermittelt die Referenzspannung des ADC
Zu übergebende Parameter	Keine
Zurückgelieferte string Parameter	s1, s2

**Übergabeparameter:**

**Antworten der Prozedur:**

**Im Erfolgsfall:**

s1 = „0“ bis „3“

s2 = Vdd (default) oder 1.024V oder 2.048V oder 4.096V

**Im Fehlerfall:**

s1 = „-4“ und s2= „FAIL“, wenn ein allgemeiner Fehler aufgetreten ist.

---

Prozedurname	Set_Dac_Voltage_Reference(wts, vref, s1, s2);
Funktion	Setzt die Referenzspannung des DAC
Zu übergebende Parameter	Wts, vref
Zurückgelieferte string Parameter	s1, s2

**Übergabeparameter:**

Wtg als „0“ oder „1“ oder „2“ und vref als „0“ bis „3“

**Antworten der Prozedur:**

**Im Erfolgsfall:**

s1 = „0“

s2 = „OK“

**Im Fehlerfall:**

s1 = „-66“ und s2 = „No HEX“, wenn keine positive Ganzzahl für wtg und / oder vref übergeben wurde.

s1 = „-4“ und s2= „FAIL“, wenn ein allgemeiner Fehler aufgetreten ist.

---

Prozedurname	Set_Dac_Value(wts, SetDacVal, s1, s2);
Funktion	Setzt den Wert des DAC
Zu übergebende Parameter	Wts, vref
Zurückgelieferte string Parameter	s1, s2

**Übergabeparameter:**

Wtg als „0“ oder „1“ oder „2“ und SetDacVal als „0“ bis „31“

**Antworten der Prozedur:**

**Im Erfolgsfall:**

s1 = „0“

s2 = „OK“

**Im Fehlerfall:**

s1 = „-66“ und s2 = „No HEX“, wenn keine positive Ganzzahl für wtg und / oder SetDacVal übergeben wurde.

s1 = „-4“ und s2= „FAIL“, wenn ein allgemeiner Fehler aufgetreten ist.

---

Prozedurname	Set_Adc_Voltage_Reference(wts, vref, s1, s2);
Funktion	Setzt den Wert des DAC
Zu übergebende Parameter	Wts, vref
Zurückgelieferte string Parameter	s1, s2

**Übergabeparameter:**

Wtg als „0“ oder „1“ oder „2“ und vref als „0“ bis „3“

**Antworten der Prozedur:**

**Im Erfolgsfall:**

s1 = „0“

s2 = „OK“

**Im Fehlerfall:**

s1 = „-66“ und s2 = „No HEX“, wenn keine positive Ganzzahl für wtg und / oder vref übergeben wurde.

s1 = „-4“ und s2= „FAIL“, wenn ein allgemeiner Fehler aufgetreten ist.

---

## // Initial Pin Values !!

Prozedurname	Get_Initial_Pin_Value_LedUartTx(TransmitPin, s2);
Funktion	Ermittelt den Ursprungswert des LED-Uart-Transmit-Pins
Zu übergebende Parameter	keine
Zurückgelieferte string Parameter	TransmitPin, s2

**Übergabeparameter:**

**Antworten der Prozedur:**

**Im Erfolgsfall:**

s1 = Wert des LED Sende-Pins

s2 = „OK“

**Im Fehlerfall:**

s1 = „-4“ und s2= „FAIL“, wenn ein allgemeiner Fehler aufgetreten ist.

---

Prozedurname	Get_Initial_Pin_Value_LedUartRx(ReceivePin, s2);
Funktion	Ermittelt den Ursprungswert des LED-Uart-Receive-Pins
Zu übergebende Parameter	keine
Zurückgelieferte string Parameter	ReceivePin, s2

**Übergabeparameter:**

**Antworten der Prozedur:**

**Im Erfolgsfall:**

s1 = Wert des LED Empfang-Pins

s2 = „OK“

**Im Fehlerfall:**

s1 = „-4“ und s2= „FAIL“, wenn ein allgemeiner Fehler aufgetreten ist.

---

Prozedurname	Get_Initial_Pin_Value_LedI2C(I2CPin, s2);
Funktion	Ermittelt den Ursprungswert des LED-I2C-Pins
Zu übergebende Parameter	keine
Zurückgelieferte string Parameter	I2CPin, s2

**Übergabeparameter:**

**Antworten der Prozedur:**

**Im Erfolgsfall:**

s1 = Wert des LED I2C-Pins

s2 = „OK“

**Im Fehlerfall:**

s1 = „-4“ und s2= „FAIL“, wenn ein allgemeiner Fehler aufgetreten ist.

---

Prozedurname	Get_Initial_Pin_Value_Sspnd(SspndPin, s2);
Funktion	Ermittelt den Ursprungswert des SSPND-Pins
Zu übergebende Parameter	keine
Zurückgelieferte string Parameter	SspndPin, s2

**Übergabeparameter:**

**Antworten der Prozedur:**

**Im Erfolgsfall:**

s1 = Wert des SSPND-Pins

s2 = „OK“

**Im Fehlerfall:**

s1 = „-4“ und s2= „FAIL“, wenn ein allgemeiner Fehler aufgetreten ist.

---

Prozedurname	Get_Initial_Pin_Value_Usbcfg(USBcfgPin, s2);
Funktion	Ermittelt den Ursprungswert des USBCFG-Pins
Zu übergebende Parameter	keine
Zurückgelieferte string Parameter	USBcfgPin, s2

**Übergabeparameter:**

**Antworten der Prozedur:**

**Im Erfolgsfall:**

s1 = Wert des USBCFG-Pins

s2 = „OK“

**Im Fehlerfall:**

s1 = „-4“ und s2= „FAIL“, wenn ein allgemeiner Fehler aufgetreten ist.

---

Prozedurname	Set_Initial_Pin_Value_LedUartTx(npv, s1, s2);
Funktion	Setzt den Ursprungswert des LED-Uart-Transmit-Pins
Zu übergebende Parameter	Npv
Zurückgelieferte string Parameter	S1, s2

**Übergabeparameter:**

Npv als „0“ oder „1“

**Antworten der Prozedur:**

**Im Erfolgsfall:**

s1 = „0“ und s2 = „OK“

**Im Fehlerfall:**

s1 = „-66“ und s2 = „No HEX“, wenn keine positive Ganzzahl für npv übergeben wurde.

s1 = „-4“ oder „-201“ und s2= „FAIL“, wenn ein allgemeiner Fehler aufgetreten ist.

---

Prozedurname	Set_Initial_Pin_Value_LedUartRx(npv, s1, s2);
Funktion	Setzt den Ursprungswert des LED-Uart-Receive-Pins
Zu übergebende Parameter	Npv
Zurückgelieferte string Parameter	S1, s2

**Übergabeparameter:**

Npv als „0“ oder „1“

**Antworten der Prozedur:**

**Im Erfolgsfall:**

s1 = „0“ und s2 = „OK“

**Im Fehlerfall:**

s1 = „-66“ und s2 = „No HEX“, wenn keine positive Ganzzahl für npv übergeben wurde.

s1 = „-4“ oder „-201“ und s2= „FAIL“, wenn ein allgemeiner Fehler aufgetreten ist.

---

Prozedurname	Set_Initial_Pin_Value_LedI2C(npv, s1, s2);
Funktion	Setzt den Ursprungswert des LED-I2C-Pins
Zu übergebende Parameter	Npv
Zurückgelieferte string Parameter	S1, s2

**Übergabeparameter:**

Npv als „0“ oder „1“

**Antworten der Prozedur:**

**Im Erfolgsfall:**

s1 = „0“ und s2 = „OK“

**Im Fehlerfall:**

s1 = „-66“ und s2 = „No HEX“, wenn keine positive Ganzzahl für npv übergeben wurde.

s1 = „-4“ oder „-201“ und s2= „FAIL“, wenn ein allgemeiner Fehler aufgetreten ist.

---

Prozedurname	Set_Initial_Pin_Value_Sspnd(npv, s1, s2);
Funktion	Setzt den Ursprungswert des LED-SSPND-Pins
Zu übergebende Parameter	Npv
Zurückgelieferte string Parameter	S1, s2

**Übergabeparameter:**

Npv als „0“ oder „1“

**Antworten der Prozedur:**

**Im Erfolgsfall:**

s1 = „0“ und s2 = „OK“

**Im Fehlerfall:**

s1 = „-66“ und s2 = „No HEX“, wenn keine positive Ganzzahl für npv übergeben wurde.

s1 = „-4“ oder „-201“ und s2= „FAIL“, wenn ein allgemeiner Fehler aufgetreten ist.

---

Prozedurname	Set_Initial_Pin_Value_Usbcfg(npv, s1, s2);
Funktion	Setzt den Ursprungswert des LED-USBCFG-Pins
Zu übergebende Parameter	Npv
Zurückgelieferte string Parameter	S1, s2

**Übergabeparameter:**

Npv als „0“ oder „1“

**Antworten der Prozedur:**

**Im Erfolgsfall:**

s1 = „0“ und s2 = „OK“

**Im Fehlerfall:**

s1 = „-66“ und s2 = „No HEX“, wenn keine positive Ganzzahl für npv übergeben wurde.

s1 = „-4“ oder „-201“ und s2= „FAIL“, wenn ein allgemeiner Fehler aufgetreten ist.

---



## // Device Operations, Functions //

Prozedurname	Reset_Device(s1,s2);
Funktion	Erzeugt einen RESET
Zu übergebende Parameter	Keine
Zurückgelieferte string Parameter	S1, s2

**Übergabeparameter:**

**Antworten der Prozedur:**

**Im Erfolgsfall:**

s1 = „0“ und s2 = „OK“

**Im Fehlerfall:**

s1 = „-4“ und s2= „FAIL“, wenn ein allgemeiner Fehler aufgetreten ist.

---

Prozedurname	Get_Adc_Data(AD_Value, s1, s2);
Funktion	Ermittelt die ADC-Werte aller 3 ADCs
Zu übergebende Parameter	Keine
Zurückgelieferte string Parameter	AD_Value, s1, s2

**Übergabeparameter:**

**Antworten der Prozedur:**

Um eine korrekte Antwort zu erhalten, muss die Variable AD\_Value des Typs „array[1..3] of string“ sein. Besser ist es, diese Variable als vom Typ „ADCValue“ (ADC-Wert) zu definieren, da die MCP2221.dcu diese Felder intern korrekt definiert.

**Beispiel:**

```
procedure TForm1.Button1Click(Sender: TObject);
var AD_Value: ADCvalue;
    s1, s2: string;
begin
  Get_Adc_Data(AD_Value, s1, s2);
  Edit1.Text:= s1;
  Edit2.Text:= s2;
  Edit3.Text:= AD_Value[1];
  Edit4.Text:= AD_Value[2];
  Edit5.Text:= AD_Value[3];
end;
```

**Im Erfolgsfall:**

s1 = „0“ und s2 = „OK“

AD\_Value[1..3] enthält die Werte der drei ADCs.

**Im Fehlerfall:**

s1 = „-4“ und s2= „FAIL“, wenn ein allgemeiner Fehler aufgetreten ist. AD\_Value[1..3] werden dann zu leeren Strings.

---

Prozedurname	Clear_Interrupt_Pin_State(s1, s2);
Funktion	Löscht den Status des Interrupt-Pins
Zu übergebende Parameter	Keine
Zurückgelieferte string Parameter	S1, s2

**Übergabeparameter:**

**Antworten der Prozedur:**

**Im Erfolgsfall:**

s1 = „0“ und s2 = „OK“

**Im Fehlerfall:**

s1 = „-4“ und s2= „FAIL“, wenn ein allgemeiner Fehler aufgetreten ist.

---

Prozedurname	Read_GPIO_PinValue(PN, s1, s2);
Funktion	Ermittelt den Wert des spezifizierten GPIO-Pins
Zu übergebende Parameter	PN
Zurückgelieferte string Parameter	S1, s2

**Übergabeparameter:**

PN als „0“ bis „3“

**Antworten der Prozedur:**

**Im Erfolgsfall:**

s1 = „0“ oder „1“ und s2 = „OK“

**Im Fehlerfall:**

s1 = „-66“ und s2 = „No HEX“, wenn keine positive Ganzzahl für pn übergeben wurde.

s1 = „-4“ und s2= „FAIL“, wenn ein allgemeiner Fehler aufgetreten ist.

---

Prozedurname	Write_Gpio_PinValue(PN, PV, s1, s2);
Funktion	Schreibt den Wert des spezifizierten GPIO-Pins
Zu übergebende Parameter	PN, PV
Zurückgelieferte string Parameter	S1, s2

**Übergabeparameter:**

PN als „0“ bis „3“, PV als „0“ oder „1“

**Antworten der Prozedur:**

**Im Erfolgsfall:**

s1 = „0“ oder „1“ und s2 = „OK“

**Im Fehlerfall:**

s1 = „-66“ und s2 = „No HEX“, wenn keine positive Ganzzahl für pn übergeben wurde.

s1 = „-4“ und s2= „FAIL“, wenn ein allgemeiner Fehler aufgetreten ist.

---

## // I2C Operations //

Prozedurname	Read_I2c_Data(Ad, ReLen, ReSpeed, DatToRe, s1, s2);
Funktion	Liest Daten vom I2C-Bus
Zu übergebende Parameter	Ad, ReLen, ReSpeed, DatToRe
Zurückgelieferte string Parameter	

### Übergabeparameter:

Alle Parameter beziehen sich auf die zu lesenden Werte am I2C-Bus.

Ad = I2C-Adresse

ReLen = Länge des zu lesenden Wertes

ReSpeed = I2C Geschwindigkeit

DatToRe = gelesene Daten

Dabei ist im Vorfeld DatToRe vom Typ „DataReceived“ zu definieren. In der MCP2221.dcu ist diese Variable als „array[1..1024] of string“ definiert und für das übergeordnete Programm auch nach außen „sichtbar“ zur Verfügung gestellt.

### Antworten der Prozedur:

#### Im Erfolgsfall:

s1 = „0“ und s2 = „OK“

#### Im Fehlerfall:

s1 = „-66“ und s2 = „No HEX“, wenn keine positiven Ganzzahlen übergeben wurden.

s1 = „-4“ und s2= „FAIL“, wenn ein allgemeiner Fehler aufgetreten ist.

---

Prozedurname	Write_I2c_Data(Ad, DatToSe, TrLen, TrSpeed, s1, s2);
Funktion	Schreibt Daten auf den I2C-Bus
Zu übergebende Parameter	Ad, DatToSe, TrLen, TrSpeed
Zurückgelieferte string Parameter	S1, s2

### Übergabeparameter:

Alle Parameter beziehen sich auf die zu schreibenden Werte auf I2C-Bus.

Ad = I2C-Adresse

ReLen = Länge des zu lesenden Wertes

ReSpeed = I2C Geschwindigkeit

DatToRe = gelesene Daten

Dabei ist im Vorfeld DatToSe vom Typ „DataToSend“ zu definieren. In der MCP2221.dcu ist diese Variable als „array[1..1024] of string“ definiert und für das übergeordnete Programm auch nach außen „sichtbar“ zur Verfügung gestellt.

### Antworten der Prozedur:

#### Im Erfolgsfall:

s1 = „0“ und s2 = „OK“

#### Im Fehlerfall:

s1 = „-66“ und s2 = „No HEX“, wenn keine positiven Ganzzahlen übergeben wurden.

s1 = „-4“ und s2= „FAIL“, wenn ein allgemeiner Fehler aufgetreten ist.

---

Prozedurname	Stop_I2c_Data_Transfer(s1,s2);
Funktion	Stoppt den Datenverkehr auf dem I2C-Bus
Zu übergebende Parameter	Keine
Zurückgelieferte string Parameter	S1, s2

**Übergabeparameter:**

**Antworten der Prozedur:**

**Im Erfolgsfall:**

s1 = positive Zahl und s2 = „OK“

**Im Fehlerfall:**

s1 = „-4“ und s2= „FAIL“, wenn ein allgemeiner Fehler aufgetreten ist.

---



**Allgemeine Hinweise zum MCP2221 am I2C-Bus:**

1. Alle zu übergebenden Parameter sind als string im Format von Dezimalzahlen zu übergeben.
2. Beachten Sie bitte, dass sich die Adressen der Bausteine am I2C-Bus (die am MCP2221 angeschlossen sind) aus einer Hardware- plus einer Softwareadresse zusammensetzen können.
3. Die Adressen zum Schreib- und Lesezugriff (der Bausteine, die mit dem MCP2221 kommunizieren) können um eine Stelle variieren. Beispiel: Schreibadresse = 64, Leseadresse = Schreibadresse + 1 = 65.
4. Der MCP2221 ist am I2C-Bus nur als Master zu betreiben. Ein Betrieb als Slave ist von Microchip nicht vorgesehen.

Beachten Sie bitte die Spezifikationen der jeweiligen Hersteller der Bausteine, die am I2C-Bus kommunizieren!

## Beschreibung der Im Quelltext mitgelieferten Delphi 7 Software Project1.pas

Zur besseren Verdeutlichung der einzelnen Prozeduren der Unit MCP2221.dcu nutzen wir die mitgelieferten Delphi Quelltexte des Programms „Project1“.  
Alle Prozeduren lassen sich durch entsprechende Buttons aufrufen. Im Editiermodus lassen sich die Quelltexte beim Doppelklick auf den entsprechenden Button im Edit-Feld anzeigen.

### Übersicht Form1, TabSheet „USB“:

Form1

Initialize | USB | Interrupt pin and GP Clock | GPIO Config. | DAC and ADC | Initial Pin Val. | Device Oper. | I2C Operations

**READ / GET**

Button	Input Fields	Response 2	Response 1
Get_VID_PID	VID, PID, HEX	Res_VID_PID	
Get_Power_Source		PS_Status, PS	
Get_Rem_Wakeup_Enable		RWEret, RWE	
Get_Current_Requirement	Unit	Cur_Res	
Get_Manufacturer	Manufacturer	Manuf_Res	
Get_Descriptor	Descriptor	Desc_Res	
Get_Ser_Number	SerNumber	SerRet	
Get_Fac_Ser_Number	Fac_SerNumber	FSRet	
Get_Ser_Num_En_Enable	Se_Nu_En_En_Res	Se_Nu_En_En	
Get_Firmware_Version	Firmware_Version, Fi_Ver_Res2	Fir_Ver_Res1	
Get_Hardware_Version	Hardware_Version, Ha_Ver_Res2	Ha_Ver_Res1	
Get_Connection_Status	Connection_Status	CS_Res	
Get_Device_Count	Num_Of_Dev		

HINWEIS: Sie müssen die Prozedur 'Get\_Connection\_Status' zuerst aufrufen, bevor Sie diese Prozedur nutzen!

**WRITE / SET**

Button	Eingaben (grüne Felder)	Response 2	Response 1
Set_VID_PID	New_VID, New_PID	VIDPID_Result2	VIDPID_Result1
Set_Power_Source	New_PS	NewPs_Result	PSret
Set_Rem_Wakeup_Enable	New_RWE	RWE_Result	RWEset
Set_Current_Requirement	New_Curr	New_Curr_Res2	New_Curr_Res1
Set_Ser_Num_En_Enable	Set_SNEE	SNEE_Res2	SNEE_Res1

GroupBox2: TGroupBox  
Origin: 504, 8; Size: 489 x 441  
Tab Stop: False; Order: 1

### Übersicht Form1.exe

Form1

Initialize | USB | Interrupt pin and GP Clock | GPIO Config. | DAC and ADC | Initial Pin Val. | Device Oper. | I2C Operations

**READ / GET**

Button	Eingaben (grüne Felder)	Response 2	Response 1
Get_GP_Pin_Config	Which_To_Get	GP_Con_Res2	GP_Con_Res1

**PIN informations**

	GP0	GP1	GP2	GP3
Pin Designat.	PinDes1	PinDes2	PinDes3	PinDes4
Pin Directions	PinDir1	PinDir2	PinDir3	PinDir4
PinOutLatches	PinVal1	PinVal2	PinVal3	PinVal4

**WRITE / SET**

Button	Eingaben (grüne Felder)	Response 2	Response 1
Set_GP_Pin_Config	Which_To_Set	GP_Con_Res2	GP_Con_Res1

**PIN settings**

	GP0	GP1	GP2	GP3
Pin Designat.	PinDes1	PinDes2	PinDes3	PinDes4
Pin Directions	PinDir1	PinDir2	PinDir3	PinDir4
PinOutLatches	PinVal1	PinVal2	PinVal3	PinVal4

**Get\_GP\_Pin\_Dir**

Button	Eingaben (grüne Felder)	Response 2	Response 1
Get_GP_Pin_Dir	Which_To_Get, GPin_Num	GP_PiDir_Res2	GP_PiDir_Res1

The direction (input/output) only matters if the pin is designated as a GPIO.

**Set\_GP\_Pin\_Dir**

Button	Eingaben (grüne Felder)	Response 2	Response 1
Set_GP_Pin_Dir	Wich_To_Set, SPin_Num, SPin_Dir	SP_PiDir_Res2	SP_PiDir_Res1

Um einen besseren Überblick zu behalten, gelten folgende Definitionen:

1. Alle TabSheets sind nach Themen sortiert. Somit wird der Überblick in den Befehlen erhalten. Wer lediglich die GPIO-Prozeduren nutzen muß, kann die entsprechende TabSheet anklicken und findet darin die nötigen Prozeduren. Die restlichen nicht benötigten Sheets können außer Acht gelassen werden.
2. Die Felder aller an die MCP2221.dcu zu übergebenden Parameter sind mit grüner Farbe hinterlegt.
3. Die Felder aller von der MCP2221.dcu erhaltenen Werte sind mit weißer Farbe hinterlegt.



**ACHTUNG:** Sie finden im Quelltext Namen der Buttons, Editierfelder und Variablen, die der jeweiligen Prozedur ähneln. Dies ist übersichtlicher, als die Editierfelder mit Edit1, Edit2....Edit146 zu benennen.

Selbstverständlich können Sie den Buttons, Editierfeldern, Variablen etc. eigene Namen vergeben. Dies beeinträchtigt den Funktionsumfang der MCP2221.dcu nicht.

Der Delphi 7 Editor unterstützt Sie bei der Verwendung der Prozeduren aus der MCP2221.dcu.

Beispiel:

Sie möchten einen Befehl verwenden und haben den im Editor korrekt eingetippt. Dann zeigt Ihnen der Delphi 7 Editor die korrekte Eingabe weiterer Parameter, wie Sie es von anderen Befehlen gewohnt sind.

<code>Set_Dac_Voltage_Reference (</code>	<code>WhichToSet: String; Vref: String; out Res1: String; out VRefSet: String</code>
--	--

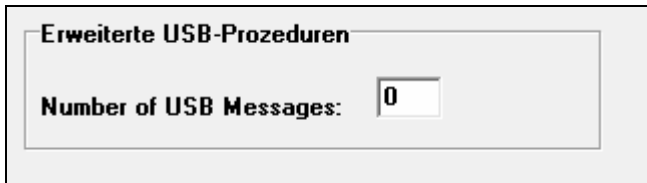
In unserem obigen Beispiel ist der Befehl zum setzen der DAC-Referenzspannung korrekt eingetippt und anschließend die Klammer gesetzt worden. Der Editor zeigt dann sofort im Kontext Fenster hinter dem Befehl die zu erwartenden Parameter.

In diesem Falle erwartet der Befehl zwei zu übergebende Parameter sowie zwei Parameter, die von der Prozedur zurückgeliefert werden (mit „out“ gekennzeichnet).

Das Programm „Project1“ wird im Quelltext geliefert. Es soll dazu dienen, Ihnen die verschiedenen Prozeduren der MCP2221.dcu zu verdeutlichen. Sie können im Editiermodus einen Button doppelklicken und der Editor zeigt Ihnen die Verwendung des gewählten Befehls in den MCP2221.dcu.

## USB-Überwachung

Auf der ersten TabSheet namens „Initialize“ finden Sie eine Extra-GroupBox namens „Erweiterte USB-Prozeduren“. Diese können Sie optional nutzen und in Ihre Software einbinden. DIESE PROZEDUR IST FÜR DEN BETRIEB DER MCP2221.dcu NICHT NOTWENDIG!!!! Sie ist lediglich ein extra Bonus.



Die sich dahinter befindliche Prozedur hat mit dem MCP2221-Baustein direkt nichts zu tun. Im Editor finden Sie dazu den Quelltext. Diese Prozedur erfasst Änderungen am USB-Bus. Bei jedem Einstecken oder Entfernen des USB-Steckers am PC erzeugt Windows® Ereignisse, die von der mitgelieferten Routine erfasst und in einem Edit-Feld in Form eines Ereigniszählers angezeigt werden.

Die Prozedur kann also vom Anwender als Ereignisüberwachung am USB-Bus benutzt werden. Über die Ereignismethode „OnChange“ des Editierfeldes „Edit1“ kann der Anwender auf USB-Ereignisse reagieren.

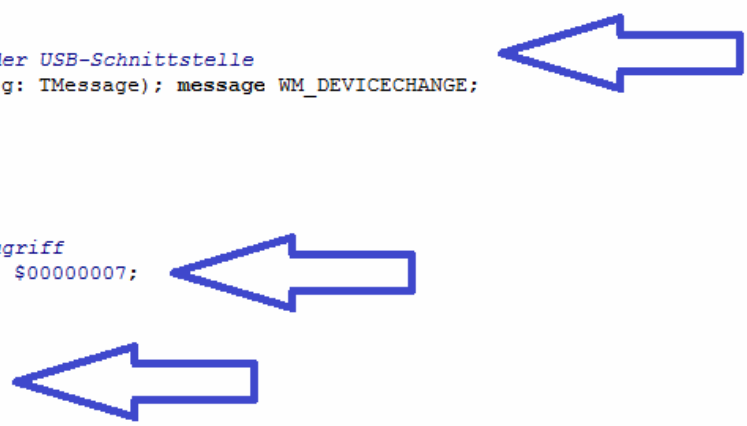
Bei der Implementierung der Routine in Ihr eigenes Programm sollten Sie folgendes beachten:

```
private
  { Private declarations }
protected
  //Prozedur zur Überwachung der USB-Schnittstelle
  procedure USB_MESSAGE(var Msg: TMessage); message WM_DEVICECHANGE;
public
  { Public declarations }
end;

const
  //Windows-Konstante bei USB-Zugriff
  DBT_DEVNODES_CHANGED = $00000007;

var
  Form1: TForm1;
  No_Of_USB_Messages: integer;

implementation
  {$R *.dfm}
```



1. Implementieren Sie im „protected“-Bereich (und nicht woanders) die procedure USB\_MESSAGE(var Msg: TMessage); message WM\_DEVICECHANGE;
2. Definieren Sie eine globale Konstante, wie im mittleren Pfeil angegeben.
3. Definieren Sie die globale Variable vom Typ integer (s. unterer Pfeil).

Hier der Quelltext der USB-Prozedur:

```
procedure TForm1.USB_MESSAGE(var Msg: TMessage);  
//Prozedur zur Überwachung der USB-Schnittstelle  
begin  
  inherited;  
  No_Of_USB_Messages:= No_Of_USB_Messages + 1;  
  if No_Of_USB_Messages >= 10 then No_Of_USB_Messages:= 1;  
  case Msg.WParam of  
    //Irgend ein USB-Ereignis hat stattgefunden  
    DBT_DEVNODES_CHANGED:  
      begin  
        //Zeige Ereignis  
        Edit1.Text:= IntToStr(No_Of_USB_Messages);  
      end;  
  end;  
end;
```

### **Anhang: RS232-Schnittstelle**

Die Nutzung der RS232 wird von der MCP2221.dcu nicht unterstützt. Die von Microchip gelieferten dll unterstützen die RS232 allesamt nicht.

Zur UART-Verwendung verweist Microchip auf die Windows CDC (Communication Device Class).

Abhilfe schaffen einige DLL und Programme, die Sie im Internet herunterladen können. Herr B. Kainka stellt im Internet folgende Dateien zur freien Verfügung:

- RSCOM.pas
- Rsapi.dll
- Terminal.exe

Hier einige Links dazu:

<http://www.b-kainka.de/pcmessfaq.htm>

<http://www.b-kainka.de/msrwefaq.htm>

<http://www.elo-web.de/messen-steuern-regeln/rscom.dll/unit-rscom.pas>

(Es ist eventuell nötig, sich auf der Elo-Webseite an zu melden, um die Unit RSCOM.pas herunter zu laden)

Alle obigen Programme und dll zur RS232-Unterstützung sind sehr einfach in der Handhabung und können leicht in Delphi 7 implementiert werden.

Auf unserem PC laufen die rsapi.dll und die RSCOM.pas einwandfrei. Aus diesem Grunde wurde auch in der gelieferten MCP2221.dcu keine RS232-Unterstützung eingebunden, da man das Rad neu erfinden würde.

Microchip-Link zum Baustein MCP2221: <http://www.microchip.com/wwwproducts/en/MCP2221>



© 2016 Daschke Ltd.

Vervielfältigung und die Weitergabe dieser Unterlagen (schriftlich, als Kopie oder im Internet) sowie der von uns gelieferten Softwarepakete oder von Teilen davon ist nur mit unserer schriftlichen Genehmigung erlaubt.

Für Angaben und deren Folgen auf den von uns verlinkten Webseiten distanzieren wir uns ausdrücklich. Verantwortlich für den Inhalt verlinkter Webseiten sind alleine deren Inhaber.

Alle angegebenen Namen sind Copyrights des jeweiligen Herstellers.

Verantwortlicher Mitarbeiter für die Entwicklung, Tests & Beschreibung:

**Rudolf Rautert, Daschke Ltd.**

**Krusenhof 42**

**45731 Waltrop**

**Tel.: +49 (0) 2309 - 540 99 42**

**Mobil.: +49 (0) 163 - 33369 17**

**Internet: [www.daschke-ltd.de](http://www.daschke-ltd.de)**

**Email: [info@daschke-ltd.de](mailto:info@daschke-ltd.de)**