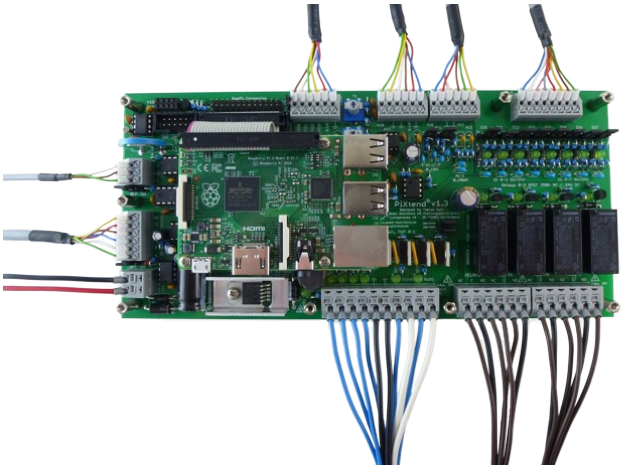
PiXtend mit CODESYS – Demo Projekt

Beschreibung des PiXtend Demo Projektes



Stand 18.08.2016, V1.3

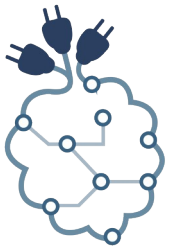
Qube Solutions UG (haftungsbeschränkt)
Arbachtalstr. 6, 72800 Eningen, Germany
http://www.qube-solutions.de/
http://www.pixtend.de

Inhaltsverzeichnis

PiXtend Application-Note:

PiXtend mit CODESYS – Demo Projekt

1. Einleitung



Diese Anleitung beschreibt das von der Qube Solutions UG bereitgestellte PiXtend Demo Projekt. Das Demo Projekt beinhaltet Programmcode und eine Visualisierung, mit der fast alle momentan verfügbaren CODESYS-Features von PiXtend getestet werden können.

Diese Anleitung hilft Ihnen dabei den Aufbau des Demo Projektes zu verstehen.
Das Demo Projekt ist gleichzeitig eine Art Referenz für Sie, so dass Sie die für Sie relevanten Features in Ihrem eigenen PiXtend CODESYS-Projekt implementieren können.

Für Ihre eigenen Projekte können Sie das Demo Projekt modifizieren und alle nicht benötigten Features entfernen. Wir empfehlen aber ein leeres Standard PiXtend-Projekt zu erstellen wie in der **App-Note PiXtend mit CODESYS – Projekt erstellen** beschrieben wird und nur die in Ihrem speziellen Projekt benötigten Features zu implementieren (empfohlen).

Wir beschränken uns in dieser Anleitung auf die Beschreibung der wichtigsten Hauptbestandteile des Demo Projektes damit Sie deren Interaktion verstehen und eigene Projekte erstellen können.

1.1 Allgemeine Hinweise

1.1.1 Urheberrecht von Texten und Bildern:

Texte und Bilder, welche mit dem Kürzel (3S) versehen sind, stammen von der Firma 3S-Smart Systems GmbH in Kempten – www.codesys.com
Texte und Bilder, welche mit dem Kürzel (RPI) versehen sind, stammen von der Raspberry Pi Foundation – www.raspberrypi.org
Texte und Bilder, welche nicht markiert oder mit dem Kürzel (QS) versehen sind, stammen von der Firma Qube Solutions UG – www.qube-solutions.de

1.1.2 Warnhinweise

PiXtend darf nicht in sicherheitskritischen Systemen eingesetzt werden. Prüfen Sie vor der Verwendung die Eignung von Raspberry Pi und PiXtend für Ihre Anwendung.

1.1.3 Einsatzbereiche PiXtend

PiXtend ist hervorragend geeignet für private und auch kommerzielle Projekte:

- Haus-Automation, Smart Home
- Zur Evaluierung von Teil-Systemen, Proof of Concept, Vorserie, Serie
- Als Lern- und Lehrplattform für Steuerungstechnik und Automation
- Als Lern- und Lehrplattform für Mikrocontroller Hard- und Software-Techniken
- Amateurfunk-, Bastler- und Maker-Projekte

1.2 Haftungsausschluss

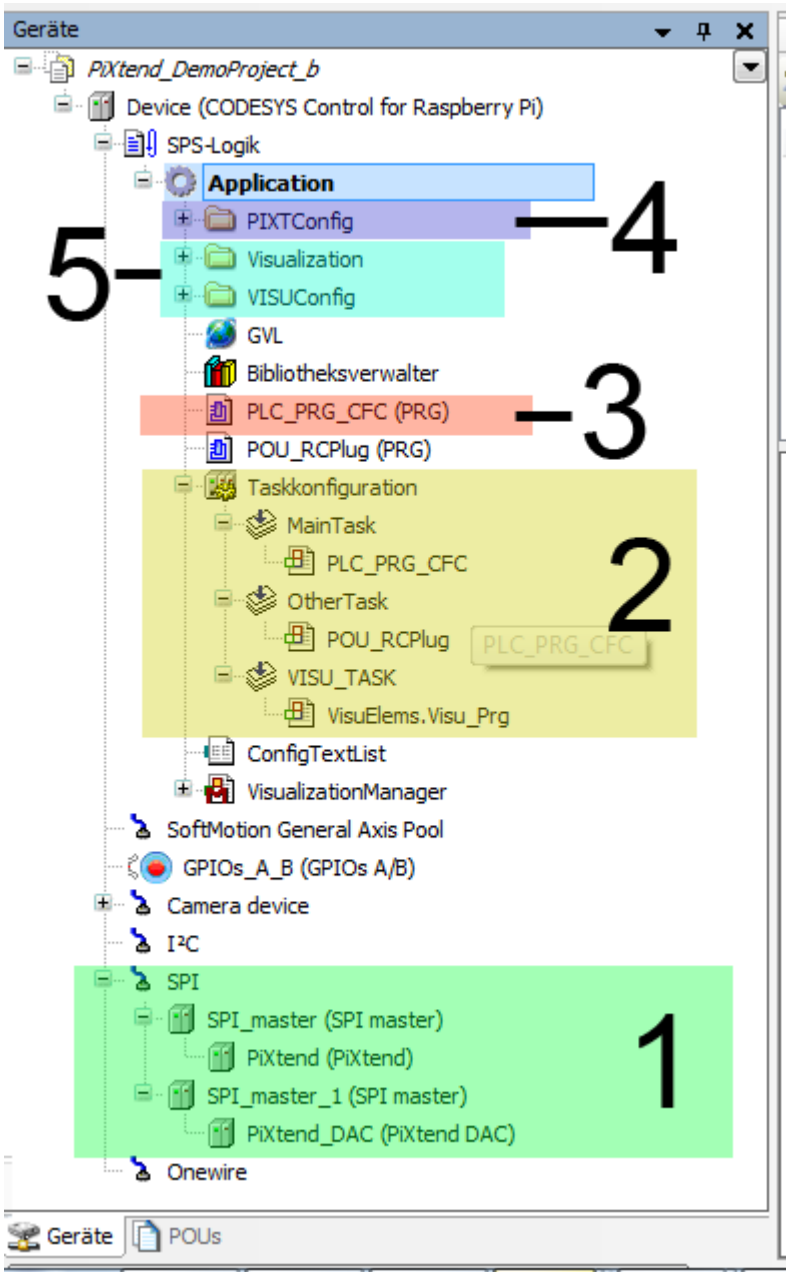
Weder Qube Solutions UG noch 3S-Smart Software Solutions können für etwaige Schäden verantwortlich gemacht werden die unter Umständen durch die Verwendung der zur Verfügung gestellten Software, Hardware oder der hier beschriebenen Schritte entstehen können.

2. Voraussetzungen

Diese Anleitung setzt voraus, dass Sie die Installation aller Software-Komponenten gemäß der Application Note **PiXtend mit CODESYS – Installation** durchgeführt haben.

Zum besseren Verständnis des hier beschriebenen Demo Projektes hilft es sehr, wenn Sie zuerst die **App-Note PiXtend mit CODESYS – Projekt erstellen** durcharbeiten oder zumindest überfliegen.

3. Aufbau des Demo Projektes



Das Demo Projekt beherbergt eine Vielzahl von Features und wirkt daher auf den ersten Blick etwas komplex. Lassen Sie sich davon nicht verwirren, sondern versuchen Sie die Funktionsweise der einzelnen Komponenten nach und nach zu verstehen.

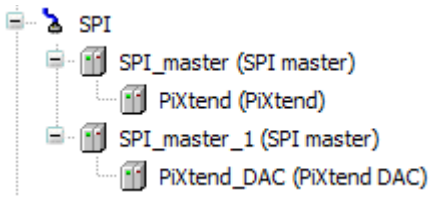
Im Folgenden werden wir die zunächst die Hauptbestandteile des Demoprojektes vorstellen:

1. SPI Master Konfiguration
2. Taskkonfiguration
3. Hauptprogramm *PLC_PRG_CFC*
4. *PIXTConfig*
5. Visualisierung und *VISUConfig*

3.1 SPI Master Konfiguration

Das PiXtend Demo Projekt verwendet, im Gegensatz zu dem wesentlich einfacheren Test Projekt, das sie bereits in der Application Note **PiXtend mit CODESYS – Projekt erstellen** kennengelernt haben, ein zusätzliches SPI Gerät.

Die SPI Geräte werden verwendet, um den Datenaustausch zwischen CODESYS, Raspberry Pi und den Chips auf PiXtend zu ermöglichen.



Das erste CODESYS SPI Gerät "PiXtend" erlaubt den Zugriff auf folgende Features von PiXtend:

- Digitale Eingänge
- GPIO Eingänge
- Analoge Eingänge
- Temperatur Eingänge für DHT11/22 Sensoren (an GPIO)
- Luftfeuchte Sensor Eingänge für DHT11/22 Sensoren (an GPIO)
- Digitale Ausgänge
- Relais Ausgänge
- GPIO Ausgänge
- PWM Ausgänge
- Control Eingänge
- Status Ausgänge

Das „PiXtend“ Gerät verwendet immer SPI-Port „/dev/spidev0.**0**“

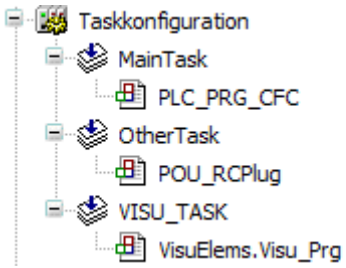
Das zweite CODESYS SPI Gerät „PiXtend_DAC“ erlaubt den Zugriff auf die beiden analogen Ausgänge. DAC steht für Digital-Analog-Converter.

Eine ausführlichere Beschreibung des PiXtend Digital-Analog-Converters finden sie in der Application Note **PiXtend mit CODESYS – Digital-Analog-Converter**

Das „PiXtend_DAC“ Gerät verwendet immer SPI-Port „/dev/spidev0.**1**“

3.2 Taskkonfiguration

Die Taskkonfiguration besteht aus 3 Tasks:



Der „Main Task“ wird alle 100 ms ausgeführt, führt das Hauptprogramm PLC_PRG_CFC aus und ist gleichzeitig der Buszyklus-Task für die beiden SPI Geräte PiXtend sowie PiXtend_DAC.

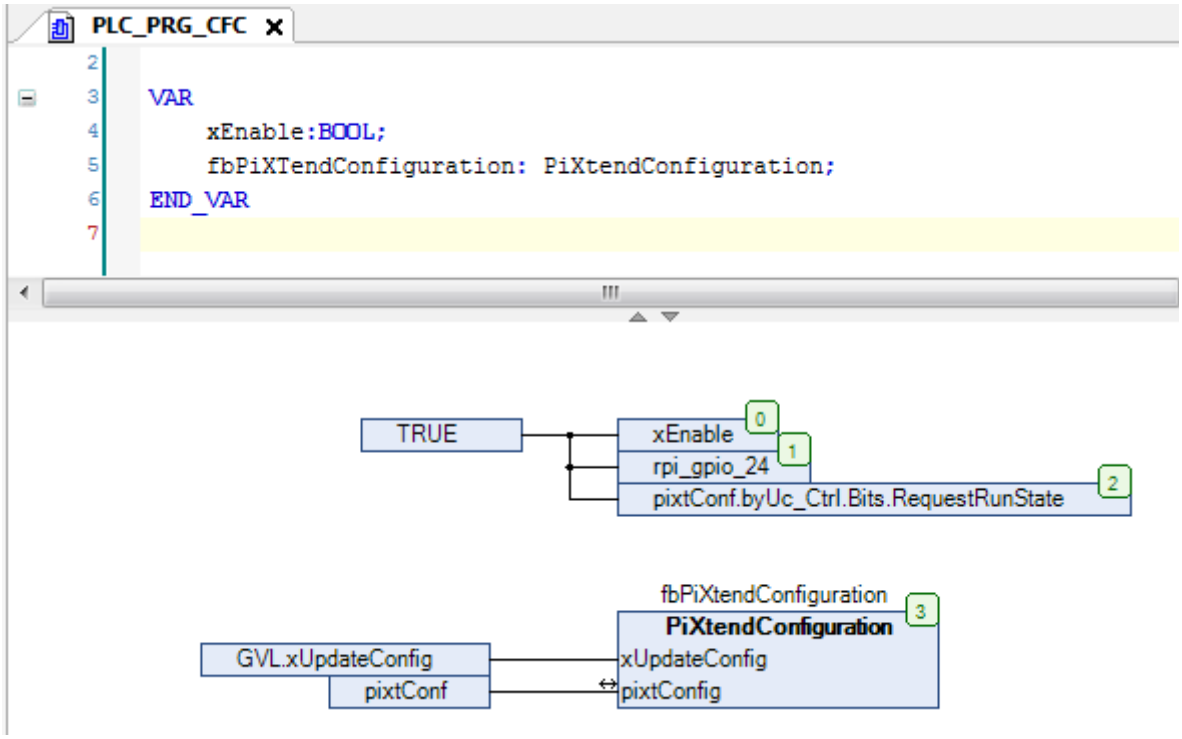
Der „Other Task“ wird alle 300 ms ausgeführt und führt das Programm „POU_RCPlug“ aus, das für das Steuern von eventuell vorhandenen Funksteckdosen verwendet wird.

Der „VISU_TASK“ wird für die Visualisierung benötigt.

Hinweis: Wenn keine DHT11/22 Sensoren an PiXtend angeschlossen und verwendet werden, so kann die Zykluszeit (Buszyklus) bis auf 25 ms reduziert werden (für das Gerät „PiXtend“). Das kann besonders dann interessant werden, wenn eine schnelle Regelung realisiert werden soll. Jeder DHT11/22 Sensor verlängert die minimale Zykluszeit (des „PiXtend“ Geräts) um ca. 20-25 ms.

Das Gerät „PiXtend_DAC“ hingegen kann mit einer Zykluszeit von 1 ms arbeiten. Es empfiehlt sich dann aber einen separaten Task für den DAC anzulegen.

3.3 Hauptprogramm PLC_PRG_CFC



Das Hauptprogramm *PLC_PRG_CFC* bildet das Herzstück des Projektes und wird mit dem MainTask alle 100 ms aufgerufen.
Mittels der dauerhaften Zuweisung der Konstanten „TRUE“ wird das *xEnable* Bit gesetzt, das Raspberry Pi GPIO Bit 24 gesetzt, welches die SPI Kommunikation ermöglicht, sowie das Mikrocontroller Control Bit *RequestRunState* gesetzt.

Des weiteren beherbergt das Hauptprogramm eine Instanz des Funktionsblockes *PiXtendConfiguration* der für Konfiguration der PiXtend Hardware Parameter verantwortlich ist.

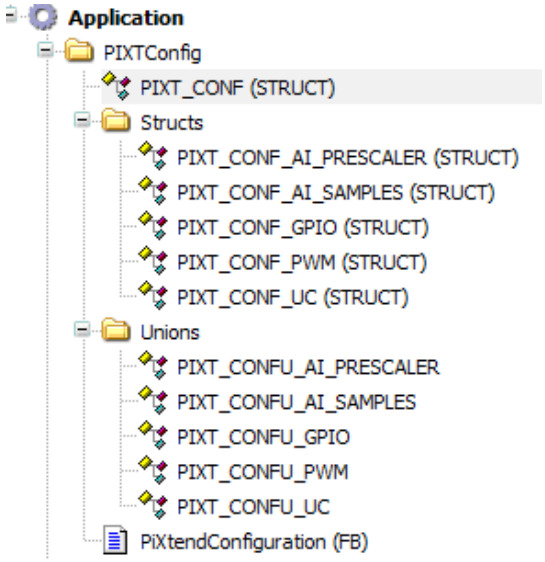
Im Vergleich zum bereits kennengelernten Test-Projekt wird im Demo-Projekt eine leicht abweichende Vorgehensweise gewählt, mit der der Zugriff auf einzelne Konfigurations-Bits und Bytes für PiXtend erleichtert werden soll.
Die im Hauptprogramm verwendete Variable *pixtConf* ist ein Struct vom Typ *PIXT_CONF* und wurde in der Globalen Variablen Liste (GVL) angelegt.

Sobald Einstellungen über die WebVisu geändert wurden, wird das Bit *GVL.xUpdateConfig* gesetzt, welches den Funktionsblock (FB) *PiXtendConfiguration* dazu veranlasst, die PiXtend Controlbytes entsprechend den gewählten Einstellungen zu befüllen.

Im Demoprojekt ist es möglich, die Konfiguration der PiXtend Hardware Parameter im laufenden Betrieb zu beeinflussen. Dafür stehen in der Visualisierung PopUp-Fenster mit

Auswahlfeldern zur Verfügung. Sie können beispielsweise die Datenrichtung von GPIOs ändern, oder die Abtastrate (sample rate) der analogen Eingänge mit Drop-Down Boxen „on the fly“ ändern.
Eine genauere Beschreibung der Komponenten des *PIXT_CONF Structs* folgen im nächsten Abschnitt.

3.4 PIXTConfig



Der *PIXTConfig* Ordner beinhaltet eine Sammlung von Structs, Unions, sowie den Funktionsblock *PiXtendConfiguration* zum einfachen Bit- als auch Byteweisen Zugriff auf alle benötigten PiXtend Konfigurationsbytes. Keine Angst, hier sind keinerlei Änderungen notwendig, wir wollen lediglich den Aufbau und die Funktionsweise erklären.
Das *PIXT_CONF Struct* beherbergt alle Control-Bytes von PiXtend.

PIXT_CONF

```
1  TYPE PIXT_CONF :
2  STRUCT
3      byPwm_Ctrl0 : PIXT_CONFU_PWM;
4      byPwm_Ctrl1 : BYTE;
5      byPwm_Ctrl2 : BYTE;
6      byGPIO_Ctrl : PIXT_CONFU_GPIO;
7      byUc_Ctrl   : PIXT_CONFU_UC;
8      byAi_Ctrl0  : PIXT_CONFU_AI_SAMPLES;
9      byAi_Ctrl1  : PIXT_CONFU_AI_PRESCALER;
10 END_STRUCT
11 END_TYPE
12
```

Im Folgenden wird exemplarisch das Union *PIXT_CONFU_UC* dargestellt:

PIXT_CONFU_UC | PIXT_CONF_UC

```
1  TYPE PIXT_CONFU_UC :
2  UNION
3      Bits : PIXT_CONF_UC;
4      Data : BYTE;
5  END_UNION
6  END_TYPE
7
```

Mit *pixtConf.byUc_Ctrl.Data*, welches ein einfaches BYTE ist, kann z.B. auf das gesamte Control Byte zugegriffen werden. Dieses Byte wird beispielsweise im Mapping der PiXtend Variablen direkt verwendet:

Control				
Application.GVL.pixtConf.byPwm_Ctrl0.Data	CtrlPwm0	%QB12	BYTE	
Application.GVL.pixtConf.byPwm_Ctrl1	CtrlPwm1	%QB13	BYTE	
Application.GVL.pixtConf.byPwm_Ctrl2	CtrlPwm2	%QB14	BYTE	
Application.GVL.pixtConf.byGPIO_Ctrl.Data	CtrlGpio	%QB15	BYTE	
Application.GVL.pixtConf.byAi_Ctrl0.Data	CtrlAi0	%QB16	BYTE	
Application.GVL.pixtConf.byAi_Ctrl1.Data	CtrlAi1	%QB17	BYTE	
Application.GVL.pixtConf.byUc_Ctrl.Data	CtrlUC	%QB18	BYTE	

Mit *pixtConf.byUc_Ctrl.Bits* kann auf einzelne Bits mit ihrem definierten Namen zugegriffen werden, ohne wissen zu müssen wo die Bits liegen. Dies ist alles im *PIXT_CONF_UC* Struct definiert:

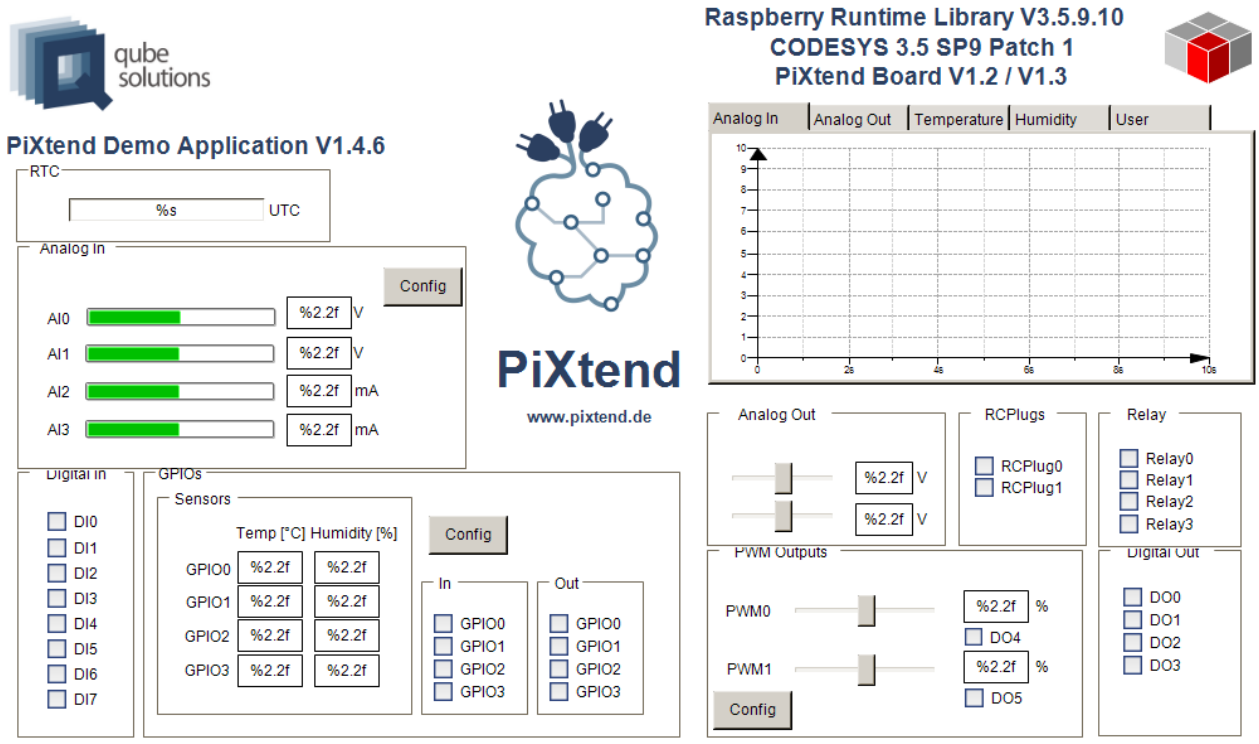
PIXT_CONFU_UC | PIXT_CONF_UC

```
1  TYPE PIXT_CONF_UC :
2  STRUCT
3      WatchdogEnable: BIT;
4      _NotUsed1: BIT;
5      _NotUsed2: BIT;
6      _NotUsed3: BIT;
7      RequestRunState: BIT;
8      _NotUsed5: BIT;
9      _NotUsed6: BIT;
10     _NotUsed7: BIT;
11 END_STRUCT
12 END_TYPE
```

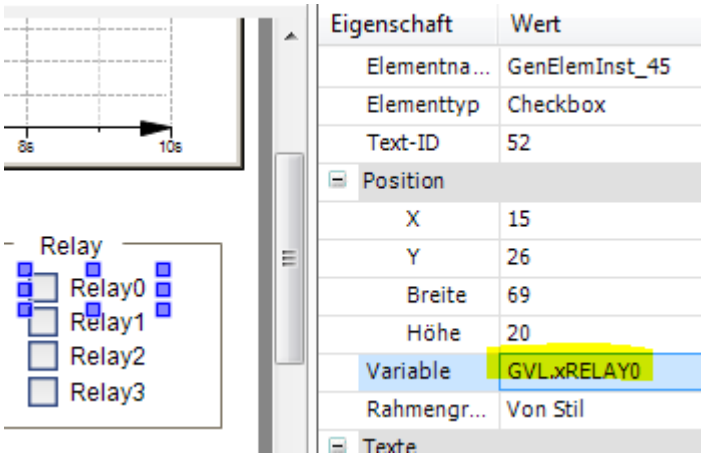
Die selbe Vorgehensweise gilt selbstverständlich für die restlichen Control Bytes. Verwenden sie das Demo Projekt als Referenz.

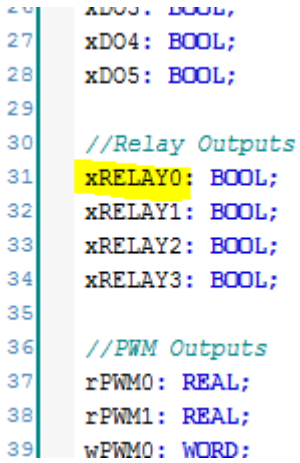
3.5 Visualisierung

3.4.1 Zugriff auf Ein- / Ausgangsvariablen

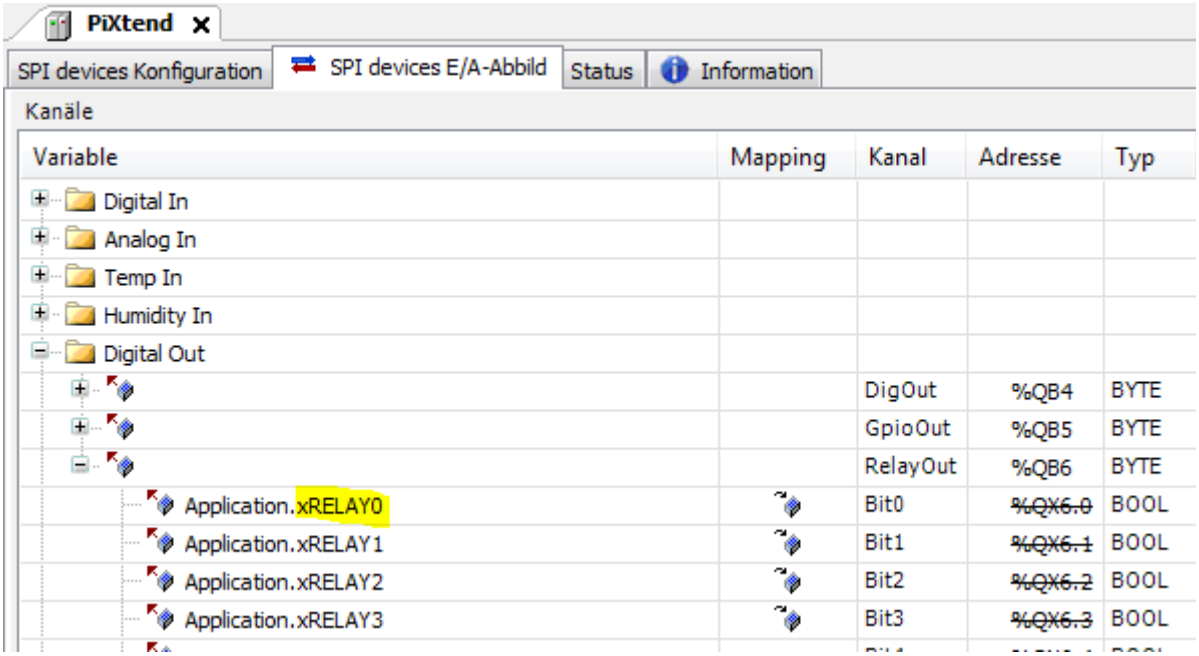


Zur Visualisierung der unmittelbaren Ein-/Ausgangsvariablen wird direkt auf die entsprechende Variable zugegriffen, die in der Globalen Variablen Liste angelegt wurde:

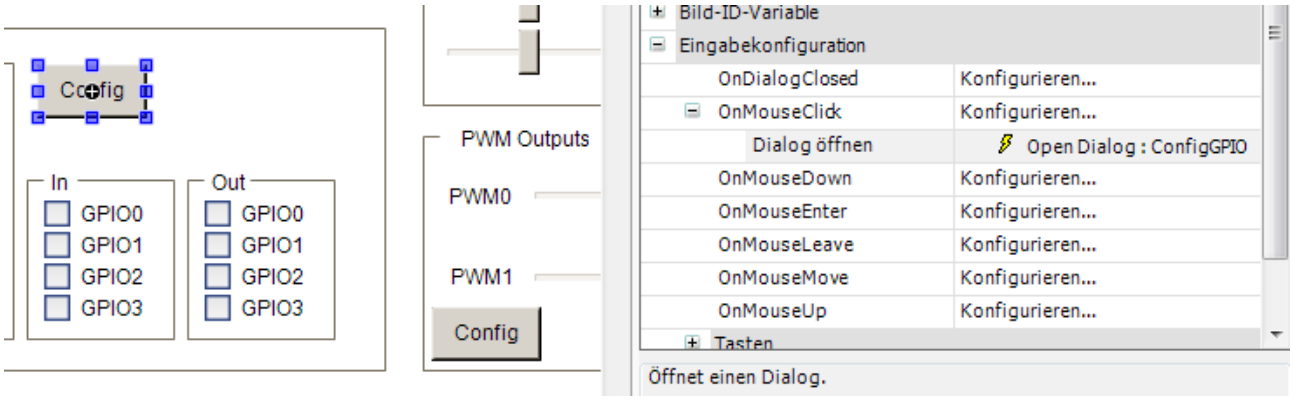




In der GVL existiert folgender Eintrag, der direkt auf das PiXtend-Gerät gemappt wurde:



3.4.2 Config-PopUps



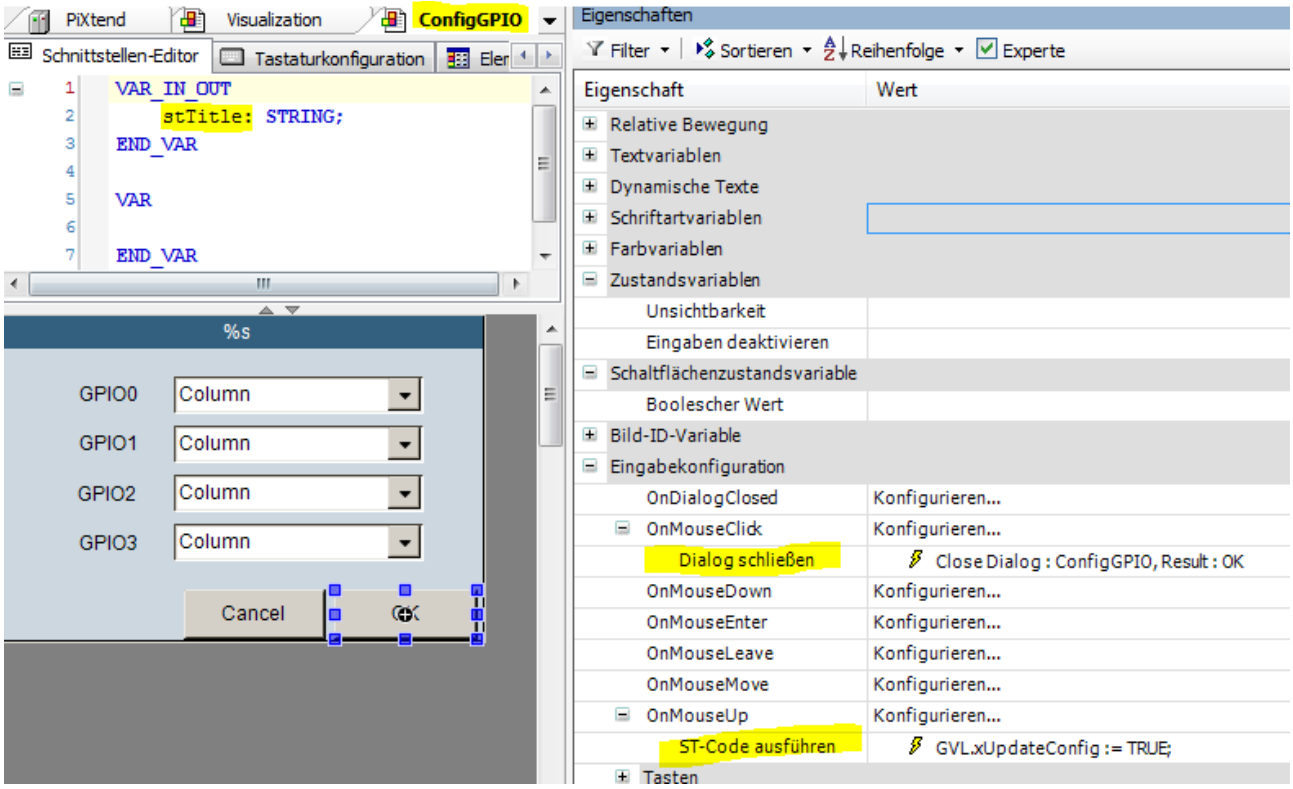
Um ein Config Popup zu öffnen, wird eine Schaltfläche verwendet die eine Visualisierung (hier *ConfigGPIO*) öffnet.

ConfigGPIO wurde als Visualisierung vom Typ „Dialog“ mit fester Größe angelegt.

Beim Schließen des PopUps mittels der Schaltfläche „OK“ wird der Dialog geschlossen, und gleichzeitig die Variable *GVL.xUpdateConfig:=TRUE;* gesetzt.

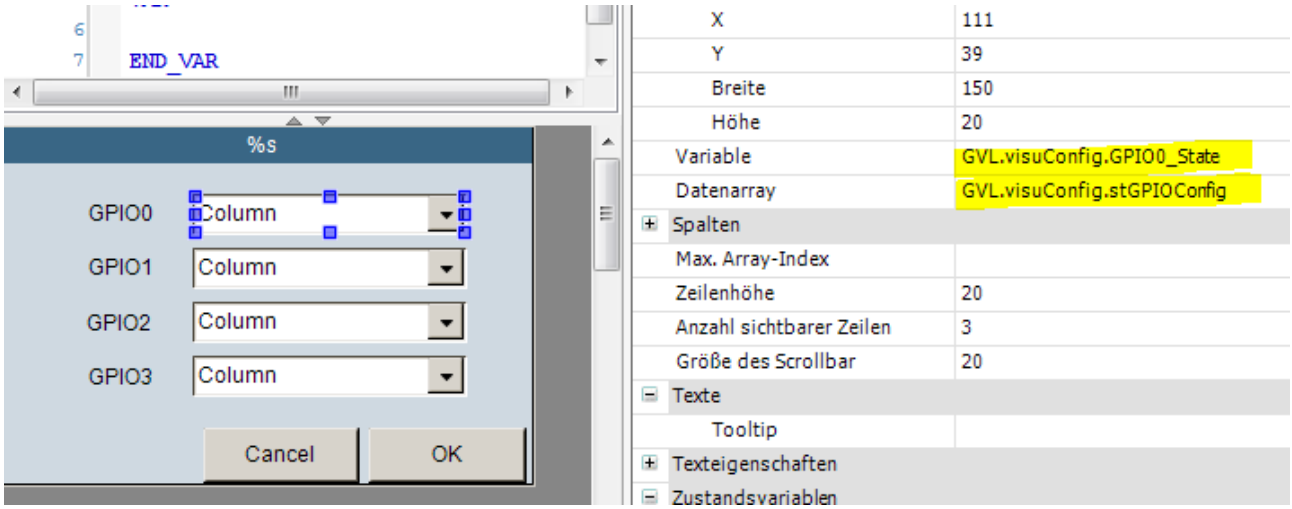
PiXtend Application-Note:

PiXtend mit CODESYS – Demo Projekt



3.4.3 Combo Boxen

Um Combo Boxen mit Strings zu füllen, wird auf das struct *GVL.visuConfig* zugegriffen.




```
VISU_CONF
1  TYPE VISU_CONF :
2  STRUCT
3      //GPIO Config Dialog
4      stGPIOConfig : ARRAY [0..3] OF STRING := ['Input', 'Output', 'DHT11', 'DHT22'];
5      GPIO0_State : INT;
6      GPIO1_State : INT;
7      GPIO2_State : INT;
8      GPIO3_State : INT;
9
10     //Analog In Config Dialog
11     stAIConfig : ARRAY [0..3] OF STRING := ['1', '5', '10', '50'];
12     AI0_State : INT := 2;
```

GPIO0_State ist vom Typ *Integer* (Ganzzahl) und enthält den Wert 0, 1, 2 oder 3 abhängig vom gewählten Eintrag „Input“, „Output“, „DHT11“ oder „DHT22“.

Der Funktionsblock *PixtendConfiguration* im Hauptprogramm prüft den Inhalt von *GPIO0_State* und setzt die benötigten Konfigurationsbytes in *pixtConf*, nachdem der Dialog geschlossen wurde und *xUpdate* gesetzt wurde.

PiXtendConfiguration

```
1   FUNCTION_BLOCK PiXtendConfiguration
2   VAR_INPUT
3       xUpdateConfig : BOOL;
4   END_VAR
5   VAR_IN_OUT

105     // GPIO
106     CASE GVL.visuConfig.GPIO0_State OF
107         0:  // Input
108             pixtConf.byGPIO_Ctrl.Bits.GPIO0_IsOutput := FALSE;
109             pixtConf.byGPIO_Ctrl.Bits.GPIO0_DHT1122  := FALSE;
110         1: // Output
111             pixtConf.byGPIO_Ctrl.Bits.GPIO0_IsOutput := TRUE;
112             pixtConf.byGPIO_Ctrl.Bits.GPIO0_DHT1122  := FALSE;
113         2: // DHT11
114             pixtConf.byGPIO_Ctrl.Bits.GPIO0_IsOutput := TRUE;
115             pixtConf.byGPIO_Ctrl.Bits.GPIO0_DHT1122  := TRUE;
116             pixtConf.byDHT.0 := TRUE;
117         3: // DHT22
118             pixtConf.byGPIO_Ctrl.Bits.GPIO0_IsOutput := TRUE;
119             pixtConf.byGPIO_Ctrl.Bits.GPIO0_DHT1122  := TRUE;
120             pixtConf.byDHT.0 := FALSE;
121     END_CASE
122     CASE GVL.visuConfig.GPIO1_State OF
123         0:  // Input
```

Wir wünschen Ihnen viel Spaß bei der Benutzung von *PiXtend mit CODESYS und ein gutes Gelingen für Ihre Projekte!*

Wir sind immer an Feedback interessiert. Sollten Sie PiXtend in einem Projekt verwenden würden wir uns über eine Erwähnung freuen.